NOTE: Art still being compiled; not included in this draft.

III. THE PDP-11 FAMILY

last edit 1/18/78

latest edit 2/16/78

PDP-11 GLUE

This part could stand alone as a book on the evolution of the PDP-11 computer structure, although it does rely on the conceptual framework of Chapter 1, and the ISPS language description given in appendix 00. The 11 has evolved quite differently from the other computers in this book, and as such provides an independent and interesting story. The factors that have created the machines are clearly market and technology based; they have generated a large number of implementations (ten) over a relatively short (eight-year) lifetime. Because there are multiple implementations spanning a performance range at the same points in time, the PDP-11 provides problems and insight which did not occur in the evolutions of the traditional mini (8 Family); the best cost/performance machines (18-bit), and the high performance machines (the DECsystem 10). The 11 designs cover a range of 500:1 in system price ($500 to $250,000) and 500:1 in memory size (4 Kwords to 2 Mwords).

The part is divided into six sections.

1. Introduction.

The first chapter (00), published when the 11 was announced, introduces the architecture, gives its goals, and predicts how it might evolve.

The family notion is quite strong, although not specific about members. Chapter 00, ~~What Have We Learned From~~ The PDP-11 after three design generations, might be read next in order to get the broadest overview, and best immediate critique of the 11 evolution.

2. Conceptual Basis.

This section contains two papers that form some of the conceptual basis for the models. Strecker, Chapter 00, describes the cache memory mechanism for building high performance models (specifically the 11/70). Levy describes the intercommunication problem among physical components and how busses carry out this task.

3. Implementations.

Of the four implementation chapters, three are on specific implementations (LSI-11, CMU-11 and 11/60) and the fourth (Chapter 00) is a study of all models. This latter chapter provides comparative data on the various implementations. ~~together with how the designs fit a conceptual model.~~

4. Evaluation.

Chapter 00 evaluates the 11 as a machine for executing FORTRAN. ~~What We Have Learned From~~ The PDP-11 after three design generations, Chapter 00, discusses aspects of the PDP-11 evolution in terms of the ~~models~~ views introduced in Chapter 1.

5. VAX-11.

This paper, by the architect of VAX-11, discusses the new architecture and its first implementation, the VAX-11/780.

6. ~~The~~ multiprocessor research computers ~~of Carnegie-Mellon University.~~

built at Carnegie-Mellon University are discussed

Three multiprocessors: C.mmp (Chapter 00), Cm* (Chapter 00) and C.vmp (Chapter 00) ~~give examples of~~ a 16-processor multiprocessor, a set of computer modules based on LSI-11, ~~and~~ a triplicated, voting multiprocessor computer for high reliability.

?

A fourth multiprocessor, ~~built at DEC~~ [illegible] PULSAR, is also discussed.

INTRODUCTION

A NEW ARCHITECTURE FOR MINICOMPUTERS--THE DEC PDP-11

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed, to describe the first (11/20) implementation, and to show possible extensions. This was attempted at a time when the whole family architecture had not been ~~worked out or even~~ fully considered.

The computer class definitions (given in 1970) of micro, mini and midi have stood the rest of time for they correspond quite closely to those of view 3 in

Chapter 1.

The major reasons (elaborated upon in Chapter 00) for the disparity between the predicted and actual evolution are:

1. The notion of designing with improved technology, especially for a family, was not understood in 1970. This understanding came later and was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus proved unacceptable for intercommunications at the very high and low end designs. Although this chapter posits a multiprocessor and multiple Unibusses for high end designs, such a ~~particular~~ ~~this exact~~ structure did not evolve as a standard. Levy's chapter elaborates on the bus evolution.

3. The address space for both physical and virtual memory was too small.

4. The particular data-type extensions were not predicted. While floating point arithmetic was discussed, the character string and decimal operations were not described. These data types evolved in response to market need and COBOL -- factors which did not exist in 1970.

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete (by adding memory management and floating point) ISPS description given in Appendix 0 of the book.

CACHE MEMORIES FOR PDP-11 FAMILY COMPUTERS

Chapter 00 by Strecker is included for four reasons: it is a clear exposition of the cache memory structure and its design parameters; the cache is the basis of a fast PDP-8 [Bell et al, 1974], the 11/60 (Chapter 00) the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is good engineering; and finally, the paper is well-written--in fact, it received the award for the Best Paper at the Third Computer Architecture Conference. We also publish it simply to serve as an encouragement and an example for those who should understand and describe their work -- such well-written and relevant papers are only too rare.

The cache design process is implicit in the way in which the work is carried out to determine the structure parameters. The relevant sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge (Chapter 00) uses Strecker's program traces and methodology.[1] Footnote 1 Note that it is easy to collect statistics about PDP-11 program behavior since the trace bit in the PS word, permits the 11 to interpret itself on a single-instruction basis. One of the important parameters to understand is the time between changes of context because all real time and multiprogrammed systems have many context switches. To the best of our knowledge this study is unique. ~~The PDP-8 cache design (Chapter 00) shows the effect of segmenting the cache for instructions and data.~~

A cache design for a PDP-8 [Bell *et al*, 1974] is summarized on page 00. The 8 study shows the effect of separating instructions and data whereas the 11 studies does not. Strecker gives the performance evaluation in terms of cache miss ratios whereas the reader is probably interested in performance or speed-up. These two measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

[illegible]

p	= total no. of memory accesses by the processor, Pc
m	= no. of memory accesses that are missed by the cache and how to be referred to Mp
t.c	= cycle time of cache memory, Mc
t.p	= cycle time of primary memory, Mp
R	= t.p/t.c (ratio of memory speeds), where R is typically 3 to 10

the relative execution speeds are:

t(no cache) = pR

t(to cache) = p + mR

speedup = pR/(p + mR) = R/(1 + (m/p) R) = a = miss ratio = m/p

therefore

speedup = R/(1 + aR) = 1/(a + 1/R)

note that if a = 0 (100% hit), the speedup is R; while
if a = 1 (100% miss), the speedup is R/(1 + R), i.e.,
the speedup is less than 1 (i.e., time to
reference both memories)

IMPLEMENTATIONS

A MINICOMPUTER-COMPATIBLE MICROCOMPUTER SYSTEM: THE DEC LSI-11

Although the paper is a descriptive narrative about the design at each of the chip, board, and backplane levels, it lacks insight that the designers at Western Digital or DEC (Duane Dickhut, Lloyd Dickman, Rich Olsen, or Mike Titelbaum) might have provided. It was written from the viewpoint of a knowledgeable user. An account of the chip-level design is available (Soha and Pohlman, 1974). The design was done at Western Digital by Roberts, Soha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with the designers to effect interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP, and the particular PMS-level components as they are integrated into a backplane to form a hardware system. Sebern points out the microprogramming tradeoff that took place between the chip and module levels to carry out functions normally in hardware: the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. For example, a bounded design that is typical of one board microcomputers was considered, though not described. However, a lower-cost, one-board system, like the VT78, (page 00) has evolved and is shown in Fig. RXT-11. The RXT-11 is an integrated system containing an LSI-11 chip set, 32 Kwords of memory, connectors for six EIA interfaces, and a controller for two floppy disk drives. One-hundred and seventy-five i.c.'s were used -- to implement the same functionality using standard LSI-11 modules, 375 i.c.'s would have been used.

Fig. RXT-11

The initial module-level design for the LSI-11 was predicated on a quad-sized form factor with a conventional backplane. The modules are shown on page 00. (Fig. LSI-11/2) Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was devised. It provided interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus (requires a double sized module). This permitted multiple board, complex options, e.g., a disk controller, to be packaged as a single option with no interconnection between the boards except via the second half of the quad board. As DEC began to build special ICs to interface to the bus, the option sizes decreased to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options. The effect of a lower cost LSI-11 system is to provide additional applications as we discuss in Chapter 1.

Fig. LSI-11/2

Table LSI-11 Options

Table of LSI-11 Options and Extensions (1978) for LSI-11/2

	LSI-11	LSI-11/2	Additions
Basic			
Processor + 4 Kw + ~~send~~ serial line	quad		
Processor		double	
Real time clock	quad		
Power controller/ sequences	double		
1/18/78 **Memories**			
4 Kw RAM	double		
4 Kw core	quad		
16 Kw RAM	quad		
4 Kw PROM	double		
4 Kw PROM; 256 RAM	double		
32 Kw RAM	quad	double	
Interfaces			
Parallel (16 line)	double		
4 line asynchronous	quad	double	
Asynchronous (serial)	double		
Instrument (IEEE-488)	double		
1 line synchronous	quad		

Direct Memory Access	quad
Foundation (general purpose)	quad
Fancy RT clock	quad
Analog-to-digital	quad
Digital-to-analog	quad
Floppy interface	double
Hard disk interface (RK05)	4 quads
" " " (RL01)	2 quads
Backplanes line printer	double
4 x 8	for quads
2 x 4	
2 x 8	
2 x 12	

~~1 line synchronous~~	~~quad~~
~~4 line asynchronous~~	~~quad~~

USING LSI PROCESSOR BIT-SLICES TO BUILD A PDP-11--A CASE STUDY IN MICROCOMPUTER DESIGN

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, ~~herein called a Microcomputer (for Microprogrammed Processor),~~ is used to interpret ~~a~~ the PDP-11 ISP. The purpose of the design was to test the assertion that ~~the~~ a bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (a real ISP) to fit the building blocks (the Intel 3000 processor). Indeed, the authors observed awkwardnesses that ultimately resulted in lower (than desired) performance. In retrospect, the Intel 3000 has not become the standard bit-slice architecture that the AMD 2900 series has; perhaps it suffered from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was the evaluation of the computer-aided-design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the major components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction of a design. The authors claim that 95% of the errors were detected in this manner. The simulated-time/real-time ratio ($\cancel{10^{6/1}}$) did

$10^6/1$

constrain the design process, e.g., the number of different runs used to find worst-case conditions.

DESIGN DECISIONS FOR THE PDP-11/60 MID-RANGE MINICOMPUTER

Unlike the reports from an architect's or reporter's viewpoint this chapter is a direct account of the design from the close proximity of the project. A mid-range machine is an inherently difficult design for the reasons of the designer characteristics we presented in Chapter 00, page 00. ~~[CM: unclear].~~ As neither the lowest cost nor highest performance PDP-11, it has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60: the cache to reduce Unibus traffic; trace-driven design of floating-point arithmetic processors; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus was previously thought to be inadequate for high performance systems, by using a cache most processor references do not use the Unibus and so leave it free for i/o traffic. This gives high performance without the attendant cost~~*~~. The cache work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since fetching multiple words would tie up the Unibus with additional traffic.

~~*Using Amdahl's constants, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus. [CM: We should do this for all models in "What We've Learned" Chapter]~~

The use of trace data to design the floating point arithmetic is described together with the resulting design. Note that the 11/60 performs roughly at 11/70 speeds but at lower cost. The implementation of the two can be compared in the following table.

Table - Implementation of 11/60 and 11/70 (count of printed circuit boards).

	11/60	11/70
Base Pc	~~5~~4	8 [CM: check 11/70 sys. manual]
Floating point	4	4
Cache	1	4
Memory management	1	2
Total	10	18

Microprogramming is used to provide both increased user-level capability and increased reliability, availability and maintainability. The ~~large~~ writeable control store option is described together with its novel use for data storage. ~~and various applications~~. This option has been recently used for emulating the PDP-8 at the OS/8 operating system level.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 2, page 00). Other

~~*Using Amdahl's constants, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus. [CM: We should do this for all models in "What We've Learned" Chapter]~~

does this belong here too?

semiconductor technology improvements are described together with how they affect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

all CAPS

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

This chapter presents a ~~most~~ comprehensive comparison of the eight processor implementations used in the ten PDP-11 models. The work was carried out to investigate various design styles for a given problem, namely ~~-~~ the interpretation of the PDP-11 ISP. The tables alone give more insight into processor implementations than is available from any single source we know. The usefulness of the data also comes from having an outside observer examine the machines and share his insight.

The tables include:

1. a set of instruction frequencies, by Strecker, for a set of ten different applications. The reader should note the frequencies do not reflect all uses, e.g., there are no floating point instructions, nor has operating system code been analyzed.

2. implementation cost (modules, ICs, control store widths) and performance (micro- and macro-instruction times) for each model; and

3. a canonical data path for all 11 implementations, against which each

*Ability to interconnect a number of subsystems together through a wired-or connection.

OR
Caps

processor is compared.

With this background data, a ~~"~~top-down~~"~~ model is built which explains the performance (macro-instruction time) of the various implementations in terms of the micro-instruction execution, and primary memory cycle time.

Since these two parameters do not fully explain (model) performance, a ~~set of~~ bottom-up approach is also used ~~factors must be introduced.~~ These factors include various design techniques and the degree of processor overlap. We believe that this analysis of a constrained problem should provide useful insight to both computer and general-digital-systems designers.

EVALUATION

TURNING COUSINS INTO SISTERS: THE ROLE OF SOFTWARE IN SMOOTHING HARDWARE DIFFERENCES

Since FORTRAN is probably ~~quite possibly~~ the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first FORTRAN compiler and object (run) time system are described together with the evolutionary extensions to improve performance. The FORTRAN IV-PLUS compiler is only briefly discussed since its improvements, largely due to compiler optimization technology, are less relevant to the 11 architecture.

The chapter title overstates the compatibility problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by essentially providing five separate object (run) time systems and a single compiler. This transparency is provided quite easily using a concept called threaded code discussed in the chapter. ~~This concept appears to be a very simple interpreter for the PDP-11--and might not be called an interpreter by many. With threaded code, one 1-word instruction requiring two memory cycle times is executed each time a high level operation code is to be interpreted, otherwise the processor is carrying out the desired op code. When a simple integer expression like I = I + 1, which occupies 2 memory words and requires 3 memory cycles to execute, is transformed into a threaded code version the program still only occupies 2 words, but instead requires 5 memory cycles to execute (nearly a factor of 2). For more complex operations requiring longer execution times, like floating point arithmetic, the overhead turns out to be quite low and the space utilization is quite good. It is also possible to move efficiently between threaded and directly executed code, although this is not done. Jim Bell (Ref) discovered the technique; it has been used extensively for other compilers because of its low time and space overhead. The ability to carry out the interpretation so elegantly was not part of the original PDP-11 design, but rather was a consequence of the generality of the 11's addressing modes.~~

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. In the second version, by recognizing the special high-frequency-of-use cases, e.g., A = 0, A = A + 1, and by having better conventions for three-address operations (to and from the stack), speedups of 1.3 and 2.0 for floating point and integers, were obtained.

It is interesting to compare the virtual machine described with the FORTRAN IV-PLUS machine which uses the floating point processors (on the 11/34, 11/45, 11/55, 11/60, and 11/70). If the FORTRAN machine described in the paper is implemented in microcode and made to operate at FPP speeds, the resulting machines turn out to operate at roughly the same speed and programs occupy roughly the same program space.

THE PDP-11 AFTER THREE DESIGN GENERATIONS

~~WHAT HAVE WE LEARNED FROM THE PDP-11?~~

This chapter is a substantially revised version of a paper called "What Have We Learned From the PDP-11?" written for the CMU Computer Science 10th Anniversary, September 1975. This paper was written to critique the original expository paper on the PDP-11 (Chapter 00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined.

The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially critical. The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further aid in understanding the design process.

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standrad for ~~of~~ interconnection. ISP

We try to provide a set of evolutionary cost-performance metrics so the 11 can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique (within DEC) problem of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a more useful evaluation of the architecture for FORTRAN execution. A complete section is given on the addressing extension, beyond the 11/45 and 11/70 extensions, which required a major perturbation: VAX-11.

The final section, addressed to the research community, describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs.

VAX-11/780: A VIRTUAL ADDRESS EXTENSION TO THE DEC PDP-11 FAMILY

This chapter ~~Chapter 00, by Bill Strecker,~~ provides a clean, somewhat terse, yet comprehensive description of the VAX-11 architecture. It is among the best papers on a specific architecture that we know. Since the VAX part of the architecture is so complete in terms of data-types, operators, addressing and memory management, it can also serve as a textbook model and base, case study for architecture in general. Goals, constraints and various design choices are given. The first model, VAX-11/780, is also briefly described.

VAX-11 is the extension to the PDP-11 ~~ISP~~ to provide a large, 32-bit virtual address for each user process. ~~When operating with 32-bit addresses, call native mode, the VAX part of the architecture is in effect.~~ The architecture includes a ~~PDP-11~~ compatibility mode ~~(~~~~)~~ for PDP-11 programs written for the RSX-11/M program environment to run unchanged. In this way, PDP-11 programs can be moved among VAX and PDP-11 computers, depending on the user's address size, computational and generality needs.

~~THE~~ MULTIPROCESSOR RESEARCH COMPUTERS ~~OF CARNEGIE-MELLON UNIVERSITY AND DEC'S PULSAR~~

Three computers, which use the 11 as a basis, were built at Carnegie-Mellon University to carry out research in computer structures and operating systems for multiprocessors. A fourth multiprocessor based on the 16 LSI-11s, called the DEC PULSAR is also discussed.

The first computer, C.mmp, is a 16 processor (11/40's and 11/20's) system with 2.5 million words of shared primary memory. It was built to investigate the programming (and resulting performance) questions associated with having a large number of processors.

The chapter on the second computer, Cm*, is located physically in the

Modules Part, as the modules that form Cm* are LSI-11's. Cm* was based on the premise that ultimately, the smallest modular unit, i.e., integrated circuit, used to build digital systems would be a computer. With this premise, we need to understand how to interconnect computers physically, how to assign parts of the application program to the various computers, and how to program the complete structure.

The third computer, C.vmp, was designed to investigate how a production microcomputer, the LSI-11, could be used to build a triplicating, voting high availability computer.

The goals of the first two are performance while the third has reliability as its goal.

We believe that technology will <u>force</u> the evolution of computing structures to converge to three styles of multiprocessor computers: (1) C.mmp-style, for high performance, incremental performance and availability; (2) loosely coupled computers like Cm* to handle specialized processing, e.g., front end, file, and signal processing; and (3) C.vmp-style for high availability motivated by ~~based on~~ increas~~ed~~ing maintenance costs.

The technology-force argument is based on history, near term technology, and resulting price extrapolations:

1. MOS technology is increasing in both speed and density faster than the technology, e.g., ECL, from which high performance machines are built.

2. The price per chip of the single-MOS-chip processors decreases at a substantially greater rate than for the low-volume high-performance, special designs. Chips in both designs have high design costs, but the special design enjoys a much lower volume.

3. Relative to all other costs of a system, the processor cost in a low end system is essentially zero.

4. Standards in the semiconductor industry tend to form more quickly for high volume products. For example, in the 8-bit microcomputer market, one type supplies about 50% of the market and three types supply over 90%.

5. A 16-bit processor-on-a-chip, with both an address space matching its performance and appropriate data-types, has been announced. Such a commodity will form the basis for nearly all future computer designs.

DEC's PULSAR, described below, is a good example of one of the more straightforward applications of this technology. As a result of these factors, the two classes of machines (MOS-processor-on-a-chip-based and low-volume, high-performance-processor-based) have rapidly diverging costs per operation per chip. Furthermore, large scale applications have been slow to form since problem complexity increases more rapidly than program size. Therefore, most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at essentially zero cost from processor-on-a-chip-based processors, large scale computing will come from arrays of processors, just as we build arrays of 64 Kbit ICs to form memory subsystems.

Insert ②

Caps→ C.mmp--A Multi-Mini-Processor

C.mmp was motivated by the need for more computing power to solve speech recognition/signal processing problems and to understand the multiprocessor software problem. Until C.mmp, only one large, tightly coupled multiprocessor had been built--the Bell Laboratory's Safeguard Computer (BSTJ issue?).

The introductory section describes the economic and technical factors influencing multiprocessor feasibility and argues for the timeliness of the research. Various problems to be researched are given together with a discussion of particular design aspects. For example, since C.mmp is predicated on a common operating system there are two sources of degradation: memory contention and lock contention. The machine's theoretical performance as a function of memory-processor interference is based on Strecker's (1971) work. In practice, because the memory was not built with low-order address interleaving, memory interference was greater than expected. This problem was solved by moving program segments.

As the number of memory modules and processors becomes very large, the theoretical performance (as measured by the number of accesses to the memory by the processors) approaches the memory bandwidth (m/memory-cycle-time) x 1/e.[ref.?] Thus, with infinite processors, there

¶ The multiprocessor research projects at CMU have emphasised synthesis and measurement. Operating systems have been built for them and the executions of user programs have been carefully analyzed. All the multiprocessor interferences, overheads, and synchronization problems have been faced; the resultant performance helps to put their actual costs in perspective. Figure HARPY looks at one of applications, the HARPY speech recognition program, in detail. ~~the perf~~ Here the performance of C.mmp and Cm* is compared with three uniprocessors (KA-10, KL-10, and 11/40).

is not a maximum limit on performance, provided all processors are not contending for the same memory.

Although there is a discussion outlining the design direction of the operating system, Hydra, later descriptions should be read [Wulf et al, 1975]. Since the ~~11's~~ small address of the PDP-11 necessitated frequent map changes, 11/40s with writeable control stores were used to implement the operating systems calls ~~to~~ which change the segment base registers.

The C.mmp which was actually built is shown in Fig. PMSC.mmp.

There are three basic approaches to the effective application of multiprocessors:

1. System-level workload decomposition. If a workload contains a lot of inherently independent activities, e.g., compilation, editing, file processing, and numerical computation, it will naturally decompose.

2. Program decomposition by a programmer. Intimate knowledge of the application is required for this time-consuming approach.

3. Program decomposition by the compiler. This is the ideal approach. However, results to date have been disappointing [Ref. Illiac IV FORTRAN compiler projects].

C.mmp was predicated on the first two approaches. Since the original paper, ALGOL 68, a language with facilities for expressing parallelism in programs, has been implemented. It has assisted greatly with program decomposition. See Figure x.

× Fig.

As can be seen in the paper, a model of the lock problem influenced the number of critical sections in the scheduler. Since the paper, the operating system Hydra has been designed and implemented. An extensive description is given in [Wulf et al, 1975].

Two experiments analyzing the performance of C.mmp and Hydra have been reported. The first, by ~~Ran~~Marathe [1977], used a hardware monitor to measure the degradation due to the locking mechanism which is invoked when shared data is accessed. The second, by Oleinick [1977], analyzed user-program-level synchronization. We summarize the results of both below.

The contention for shared resources in a multiprocessor system occurs at several levels. At the lowest level, processors contend at the cross-point switch level for memory. This memory interference is discussed in the chapter. On a higher level there is contention for shared data in the operating system kernel. At higher levels, processes contend for i/o devices and for software processes, e.g., for memory management. The following table points to models on experimental data at these different levels in C.mmp.

Table?

Contention Level	Reference

user-program	[Oleinick, 1977]
	[Fuller and Oleinick, 1976]
	Marathe and Fuller, 1977]
Hydra Kernel objects	[Marathe
cross-point switch	Chapter XX
	[Fuller, 1976]

~~Ran~~ Marathe's data show that the shared data of Hydra is organized into enough separate objects that a very small degration (less than 1%) results from contention for these objects. Table LOCK is reproduced from his paper. He also built a queueing model which projected that the contention level would be about 5% in a 48 processor system.

Table LOCK

Replace with section (A)

~~Oleinick uses a root finding algorithm to study various implementations of a single problem.~~

~~[CM: elaborate after studying his thesis chapter.]~~

Caps →

Multi-Microprocessors: An Overview and Working Example

The Cm* work, sponsored by NSF and ARPA, is an extension of earlier NSF-sponsored research [Bell, ~~etc~~ et al. 1973] on register-transfer-level modules. As LSI and VLSI enable construction of the processor-on-a-chip, it is apparent that low-level, register-transfer modules are passe' for the construction of all but low-volume computers. Although the research is

[Oleinick's data on C.mmp
— for 11 Glue — follows Marathe's
data in the section on C.mmp]

Section A

Oleinick [1978] has used C.mmp to conduct an experimental, as opposed to theoretical, study of implementation of parallel algorithms on a multiprocessor. He ~~uses a rootfinding algorithm~~ studies an extension of the bisection method for finding the roots of an equation. ~~Thus~~ A decomposition of the binary search for a root into n parallel processes is to evaluate the function simultaneously at n points.

Under ideal conditions, ~~each~~ all processes would finish ~~evaluat~~ the function evaluation (required at each step) at the same time, and then some brief bookkeeping would take place to determine the next subinterval for the n processes to work on. Figure Sdeal shows this case. ~~Since~~ However, because the time to evaluate the function is data dependent, some processes complete before others. Moreover, if the bookkeeping task is time consuming relative to the time to evaluate the function, the speedup ratio will suffer.

Oleinick systematically studies each source of fluctuation in performance. The dominant one is the mechanism used for process synchronization. Fig 4.2 ~~shows~~ compares the speed up obtained with the four different synchronization primitives ~~below~~ used. These are as follows:

PMØ

PM1

Kernel

Spin lock – the processor ~~simply waits~~ continues to executes a short sequence of instructions which repeatedly test the lock.

Clearly the impact of process synchronization is a function of the ratio of synchronization time to function evaluation time. Figure 4.5 ~~indicates that spin lo~~ gives the ranges over which ~~the~~ each lock ~~can~~ shared be used without dominating execution time.

predicated on structures employing a hundred or so processors, this chapter describes the culmination of the first (ten-processor) phase.

The authors motivate their work by appealing to the diseconomy-of-scale arguments which we advanced at the beginning of this section (page 00)~~.~~ and elaborate upon in Part IV. To provide additional context for their research, computer modules (Cm*), multiprocessors (C.mmp) and computer networks are described in terms of performance and problem suitability. The chapter gives a description of the modules structure, together with their associated limitations and potential research problems.

Insert section (S)

The final, most important, part of the chapter evaluates the performance of Cm* for five different problems.

~~new page~~ →

Insert Section (B)

Caps

C.vmp: The Architecture and Implementation of a Fault Tolerant Multiprocessor

C.vmp is a triplicated, voting multiprocessor designed to understand the difficulty (or ease) of using standard, off-the-shelf LSI-11's to provide greatly increased reliability. There is concern for increased reliability because systems are becoming more complex, are used for more critical applications, and because maintenance costs for all systems are increasing. Because the designers themselves carry out and analyze the work, this chapter provides first-hand insight into high reliability designs and the design process--especially its evaluation. The system has operated for several months and the first phase of work is complete.

Section 3

A 50-Computer-Module Cm* is now under construction and is planned to be operational by the end of 1978 for evaluation in 1979. The extension of Cm* is known as Cm*/50 and is shown in Fig Cm*/50. It will be used to test the CMU ideas on parallel programming methods, fault tolerance, modularity, and the extensibility of the Cm* structure.

Section 5

The grouping of processor and memory into modules and the hierarchy of bus structures - LSI-11 bus, Map bus, and Intercluster bus - are radical departures from more conventional computer systems.

Several design goals are initially predicated and the work is carried out against the goals.

The goal of software and hardware transparency turned out to be easier to attain than expected, because of an idiosyncrasy of the floppy disk controller. Because the controller effects a word-at-a-time bus transfer from a one-sector buffer, voting can be carried out at a very low level. It is unclear how the system would have been designed without this type of controller; ~~as~~ at a minimum, some ~~form~~ part of the software transparency goal would not have been ~~violated~~ met and a significant controller modification would have been necessary.

A number of models are given by which the design is evaluated. From the discussion of ~~Various~~ component reliabilities ~~are used and~~ the reader should get ~~a great deal of~~ some insight into the factors contributing to reliability. It should be noted that a custom-LSI-~~dsg~~ designed ~~special hardware~~ voter is needed to get a sufficiently low cost for a marketable C.vmp. While the intent of C.vmp is not a product, it does provide much of the insight for such a product.

~~[PULSAR here]~~

Insert ~~Add~~ Section △

Section △ Insert for 11 glue – Don't Lose!

Jega Arulpragasam
2 February 1978

Pulsar: A Performance Range mP System

16 LSI-11 multiprocessor computer for investigating the cost-effectiveness of multiple microprocessors

PULSAR is a ~~project, begun in Autumn of 1976, aimed at investigating cost-effective ways of interconnection to create a symmetrical multiprocessing system~~ It covers a ~~specific~~ performance range of ~~This range is defined in terms of the total combined processor memory request rate and extends up to 5 megawords/sec. This is equivalent to a PDP-11 mip rate of about 2.0.~~ (approximately 1 LSI-11 to better than an 11/70 for simple instructions).

~~The particular interconnection chosen for breadboarding (so as to be able to explore more fully the software issues) is shown in Fig 1.~~

The breadboard system (Fig. 1) ~~has the general functionality~~ ~~similar~~ is based on ~~of~~ the 11/70 structure, including multiple interrupt levels and 22-bit physical addressing. It does not, ~~however~~ implement I&D space or supervisor mode, nor does it ~~make provision for attaching~~ share Floating Point processors.

The processors communicate with each other (P-Boards), the Unibus Interface (UBI) and a Common Cache and Control ~~Section~~ ~~(CC)~~ via a high-bandwidth, synchronous bus. ~~This~~ short (6 inch) ~~bus could achieve a bandwidth of 8.33 megawords per second by restricting its length to <6".~~

The ~~CC~~ Common Cache and Control ~~and Cache~~ contains a ~~large (8K word)~~, direct mapping, shared cache with a 2-word block size, interfacing to the 11/70 memory bus, ~~with 2 or more controllers providing block interleaving.~~ 2 or 4-way interleaved ~~Analysis shows this to be necessary to~~ prevents the memory subsystem from becoming a ~~severely limiting~~ bottleneck, in spite of the large reduction in bandwidth demand provided by the cache. ~~In addition, the CC~~ ~~It also~~ The control provides all the mapping functions for both Unibus and processor accesses to memory. The Unibus map registers (á la 11/70) and the process map ~~KT~~ registers for each processor are held in a single ~~monolithic bipolar chip array.~~ bipolar memory.

The UBI provides ~~all~~ the Unibus control functions ~~normally exercised by the CPU in~~ of a conventional PDP-11 ~~uniprocessor systems. It interfaces the Unibus to the P-bus to communicate with the CC for data transfer and console functions, and to the P-Boards for forwarding interruptions. The Interrupt Director mechanism in the UBI is capable of overlapping the handling of different priority levels on the Unibus side, but the forwarding of vectors to processors over the P-bus is not overlapped.~~ Interrupts will be fielded by the first enabled processor ~~reaching the Service Branch on I-fetch,~~ with preferential treatment for any processor in WAIT state.

~~The~~ Each P-Board~~s~~ contains two independent microprocessor chip sets with modified microcode ~~and some support logic to provide the extended functionality~~

Insert A, P3

~~needed. The two chip sets act as independent processors with functionally independent adapters to the P-bus.~~ Internal contention for the adapter is eliminated by running the two processors out of phase to each other. Such contention as does exist is resolved by the mechanism for arbitration of the P-bus itself.

The PULSAR has an ASCII console interfacing via a KMC-11, communications controller with modified microcode~~, to the Unibus.~~ In addition, ~~there exists~~ a debug panel ~~with~~ has displays for every stage of the pipeline.

~~Finally, 11/70 style Massbus Adapters are also provided.~~ P-bus and controller

OPERATION

The ~~heart of the~~ PULSAR System is ~~a resource~~ pipeline oriented. ~~in which a given transaction~~, ~~is~~ are allocated every resource in the pipeline at specific intervals bearing a fixed relation to each other. ~~This implies that if a particular transaction does not use a specific resource, there is an interval of time when that resource will be idle because no other transaction is allowed to use it.~~ with specific time slots for each processor

This permits a single, simple arbitration mechanism, rather than separate (complex) ones for each resource. ~~That single point of arbitration is for the P-bus Address/Function lines.~~

Once ~~these~~ pipeline is ~~are~~ assigned to a transaction, the successive intervals of time are assigned to the following resources in order:

1) The mapping array.

2) The address translation logic.

3) The cache.

4) The address validation logic.

5) The data lines of the P-bus.

~~Of course~~, the memory subsystem ~~itself~~, which is not a part of this resource pipeline, has an independent arbitration mechanism. Interfacing between these independent mechanisms is by means of queues.

There are some operations which require more than one access to the same resource in the pipeline. These operations are effectively handled as two transactions. Examples of such operations are Memory Writes and Internal I/O Page (say KT register) accesses. A memory write may need a second access to the cache for update, while the Internal I/O Page may need another access to the map array.

There are other operations in which the timing does not permit the use of a particular resource in the specific interval that is allocated to that transaction. This happens, for instance, when a Read operation results in a cache miss. The data is not available in time. In this case too a second transaction takes place, initiated when backing store data becomes available.

In the case of interruptions, the ~~CC~~ central control does not participate, and direct communication is established between the UBI and an appropriate Pc, with the currently chosen interrupt handling algorithm. ~~Consideration will be given to a change, after the breadboard is running, that would take into account software priority levels.~~

In that case, it is anticipated that the information regarding the software priority levels will be centralized in the CC.

In such a system a FORCE INTERRUPT message (which costs only the recognition hardware) suffices for all interprocessor communication.

The common cache has been proved to be more effective in both cost and in performance (provided the microprocessors' access time requirements are met) than a set of individual processor caches would be. It also eliminates the common stale data problem.

However, since the bandwidth of the pipeline is constrained by its slowest resource, the cache, the cache cycle has been separated into read - compare and compare - update cycles which are mutually exclusive during a single pass of a transaction through the pipeline. Careful design was needed to eliminate the stale-data-like problems this could cause, without resorting to cache invalidation.

If, however, P-bus bandwidth ever does become a problem, it would be possible to alleviate this with small on-board caches for Pc's that cached Read Only data.

-A- Console operations are effected by the UBI interrogating or changing a save area for each processor, physically held in the Mapping Array, in response to ASCII console messages over the unibus. Each processor places all appropriate status in the save area on every HALT, and restores from the save area prior to acting upon every CONTINUE or START.

SOFTWARE CONTENTION

Fig 2 shows the effective performance of the system, with and without the effects of software contention.

Preliminary measurements on the 11/70 Program Development System suggest that average executive occupancy is about 5%. Accordingly, we plan to run the system with a single global lock on the executive, as the degradation expected is small.

However, executive occupancy levels of 30% have been measured over periods of a few minutes. We have therefore designed a hierarchical multiple lock system with a lock of locks, which is set for very brief periods of time. But an implementation will await the collection of data on the average time spent under each of the several different potential "sublocks."

COST COMPARISON

Cost projections indicate that a multiprocessor will have an increase in parts count over ~~an~~ each possible equivalent performance uniprocessor ~~at each~~ in the range. ~~The cost penalt For~~ This will range from ~~about~~ a 20% increase for a 2 Pc mP, ~~down to close~~ approaching to 0% at the top of the range. It is to be noted that the 20% premuim can be reduced ~~to less than 10%~~ if provision is not made for expansibility over the entire range. Further, the premuim is based on parts count only and excludes considerations of cost benefits due to ~~volume~~. ~~mass~~ production learning, common spares and manuals, lower engineering costs, etc.

Clearly a sepeate 1 Pc structure can be cost-effective (since this is the LSI-11).

Marathe, M. and Füller, S. H. A Study of Multiprocessor Contention for Shared Data in C.mmp, Proceedings 1977 ACM SIGMETRICS Conferences, pp. 255-262.

Wulf, W. [and others] The Hydra Operating System, Fifth ACM SIGOPS Symposium on Operating Systems Principles (Nov. 1975).

new page

11 Glue Figures and Tables

last edit 2/16/78

latest edit 3/2/78

PDP-11 GLUE

This part could stand alone as a book on the evolution of the PDP-11 computer structure, although it does rely on the conceptual framework of Chapter 1, and the ISPS language description given in appendix 00. The 11 has evolved quite differently from the other computers in this book, and as such provides an independent and interesting story. The factors that have created the machines are clearly market and technology based; they have generated a large number of implementations (ten) over a relatively short (eight-year) lifetime. Because there are multiple implementations spanning a performance range at the same points in time, the PDP-11 provides problems and insight which did not occur in the evolutions of the traditional mini (8 Family); the best cost/performance machines (18-bit), and the high performance machines (the DECsystem 10). The 11 designs cover a range of 500:1 in system price ($500 to $250,000) and 500:1 in memory size (4 Kwords to 2 Mwords).

The part is divided into six sections.

1. Introduction.

    The first chapter (00), published when the 11 was announced, introduces the architecture, gives its goals, and predicts how it might evolve.

The family notion is quite strong, although not specific about members. Chapter 00, The PDP-11 After Three Design Generations, might be read next in order to get the broadest overview, and best immediate critique of the 11 evolution.

2. Conceptual Basis.

This section contains two papers that form some of the conceptual basis for the models. Strecker, Chapter 00, describes the cache memory mechanism for building high performance models (specifically the 11/70). Levy describes the intercommunication problem among physical components and how busses carry out this task.

3. Implementations.

Of the four implementation chapters, three are on specific implementations (LSI-11, CMU-11 and 11/60) and the fourth (Chapter 00) is a study of all models. This latter chapter provides comparative data on the various implementations.

4. Evaluation.

Chapter 00 evaluates the 11 as a machine for executing FORTRAN. The PDP-11 After Three Design Generations, Chapter 00, discusses aspects of the PDP-11 evolution in terms of the views introduced in Chapter 1.

5. VAX-11.

This paper, by the architect of VAX-11, discusses the new architecture and its first implementation, the VAX-11/780.

6. Multiprocessor research computers.

Three multiprocessors built at Carnegie-Mellon University are discussed: C.mmp (Chapter 00) a 16-processor multiprocessor, Cm* (Chapter 00) a set of computer modules based on LSI-11, and C.vmp (Chapter 00) a triplicated, voting multiprocessor computer for high reliability. A fourth multiprocessor, PULSAR, built at DEC is also discussed.

INTRODUCTION

A NEW ARCHITECTURE FOR MINICOMPUTERS--THE DEC PDP-11

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses what we have learned from the PDP-11. The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed, to describe the first (11/20) implementation, and to show possible extensions. This was attempted at a time when the whole family architecture had not been fully considered.

The computer class definitions (given in 1970) of micro, mini and midi have stood the rest of time for they correspond quite closely to those of view 3 in Chapter 1.

The major reasons (elaborated upon in Chapter 00) for the disparity between the predicted and actual evolution are:

1. The notion of designing with improved technology, especially for a family, was not understood in 1970. This understanding came later and was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus proved unacceptable for intercommunications at the very high and low end designs. Although this chapter posits a multiprocessor and multiple Unibusses for high end designs, such a structure did not evolve as a standard. Levy's chapter elaborates on the bus evolution.

3. The address space for both physical and virtual memory was too small.

4. The particular data-type extensions were not predicted. While floating point arithmetic was discussed, the character string and decimal operations were not described. These data types evolved in response to market need and COBOL -- factors which did not exist in 1970.

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete (by adding memory management and floating point) ISPS description given in Appendix 0 of the book.

CACHE MEMORIES FOR PDP-11 FAMILY COMPUTERS

Chapter 00 by Strecker is included for four reasons: it is a clear exposition of the cache memory structure and its design parameters; the cache is the basis of a fast PDP-8 [Bell et al, 1974], the 11/60 (Chapter 00) the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is good engineering; and finally, the paper is well-written--in fact, it received the award for the Best Paper at the Third Computer Architecture Conference. We also publish it simply to serve as an encouragement and an example for those who should understand and describe their work -- such well-written and relevant papers are only too rare.

The cache design process is implicit in the way in which the work is carried out to determine the structure parameters. The relevant sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge (Chapter 00) uses Strecker's program traces and methodology[1]. One of the important parameters to understand is the time between changes of context because all real time and multiprogrammed systems have many context switches. To the best of our knowledge this study is unique.

A cache design for a PDP-8 [Bell et al, 1974] is summarized on page 00. The 8 study shows the effect of separating instructions and data whereas the 11 study does not. Strecker gives the performance evaluation in terms of cache miss ratios whereas the reader is probably interested in

[1]Note that it is easy to collect statistics about PDP-11 program behavior since the trace bit in the PS word, permits the 11 to interpret itself on a single-instruction basis.

performance or speed-up. These two measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

p	= total no. of memory accesses by the processor, Pc
m	= no. of memory accesses that are missed by the cache and how to be referred to Mp
t.c	= cycle time of cache memory, Mc
t.p	= cycle time of primary memory, Mp
R	= t.p/t.c (ratio of memory speeds), where R is typically 3 to 10

the relative execution speeds are:

$$t(\text{no cache}) = pR$$

$$t(\text{to cache}) = p + mR$$

$$\text{speedup} = pR/(p + mR) = R/(1 + (m/p)\ R) = a = \text{miss ratio} = m/p$$

therefore

$$\text{speedup} = R/(1 + aR) = 1/(a + 1/R)$$

note that if a = 0 (100% hit), the speedup is R; while

if a = 1 (100% miss), the speedup is R/(1 + R), i.e.,

the speedup is less than 1 (i.e., time to reference both memories)

IMPLEMENTATIONS

A MINICOMPUTER-COMPATIBLE MICROCOMPUTER SYSTEM: THE DEC LSI-11

Although the paper is a descriptive narrative about the design at each of the chip, board, and backplane levels, it lacks insight that the designers at Western Digital or DEC (Duane Dickhut, Lloyd Dickman, Rich Olsen, or Mike Titelbaum) might have provided. It was written from the viewpoint of a knowledgeable user. An account of the chip-level design is available (Soha and Pohlman, 1974). The design was done at Western Digital by Roberts, Soha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with the designers to effect interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP, and the particular PMS-level components as they are integrated into a backplane to form a hardware system. Sebern points out the microprogramming tradeoff that took place between the chip and module levels to carry out functions normally in hardware: the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. For example, a bounded design that is typical of one board microcomputers was considered, though not described. However, a lower-cost, one-board system, like the VT78, (page 00) has evolved and is shown in Fig. RXT-11. The RXT-11 is an integrated system containing an LSI-11 chip set, 32 Kwords of memory, connectors for six EIA interfaces, and a controller for two floppy disk drives. One-hundred and seventy-five i.c.'s were used -- to implement the same functionality using standard LSI-11 modules, 375 i.c.'s would have been used.

The initial module-level design for the LSI-11 was predicated on a quad-sized form factor with a conventional backplane. The modules are shown on page 00. Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was devised. It provided interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus (requires a double sized module). This permitted multiple board, complex options, e.g., a disk controller, to be packaged as a single option with no interconnection between the boards except via the second half of the quad board. As DEC began to build special ICs to interface to the bus, the option sizes decreased to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options. The effect of a lower cost LSI-11 system is to provide additional applications as we discuss in Chapter 1.

Table of LSI-11 Options and Extensions (1978) for LSI-11/2

	LSI-11	LSI-11/2
Basic		
Processor + 4 Kw + serial line	quad	
Processor		double
Real time clock	quad	
Power controller/ sequences	double	
Memories		
4 Kw RAM	double	
4 Kw core	quad	
16 Kw RAM	quad	
4 Kw PROM	double	
4 Kw PROM; 256 RAM	double	
32 Kw RAM	quad	double
Interfaces		
Parallel (16 line)	double	
Asynchronous (serial)	double	
4 line asynchronous	quad	double
1 line synchronous	quad	

Instrument (IEEE-488)	double
Direct Memory Access	quad
Foundation (general purpose)	quad
Fancy RT clock	quad
Analog-to-digital	quad
Digital-to-analog	quad
Floppy interface	double
Hard disk interface (RK05)	4 quads
Hard disk interface (RL01)	2 quads
line printer	double

Backplanes

4 x 8	for quads
2 x 4	
2 x 8	
2 x 12	

USING LSI PROCESSOR BIT-SLICES TO BUILD A PDP-11--A CASE STUDY IN MICROCOMPUTER DESIGN

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, is used to interpret the PDP-11 ISP. The purpose of the design was to test the assertion that a bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (a real ISP) to fit the building blocks (the Intel 3000 processor). Indeed, the authors observed awkwardnesses that ultimately resulted in lower (than desired) performance. In retrospect, the Intel 3000 has not become the standard bit-slice architecture that the AMD 2900 series has; perhaps it suffered from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was the evaluation of the computer-aided-design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the major components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction of a design. The authors claim that 95% of the errors were detected in this manner. The simulated-time/real-time ratio ($10^6/1$) did constrain the design process, e.g., the number of different runs used to find worst-case conditions.

DESIGN DECISIONS FOR THE PDP-11/60 MID-RANGE MINICOMPUTER

Unlike the reports from an architect's or reporter's viewpoint this chapter is a direct account of the design from the close proximity of the project. A mid-range machine is an inherently difficult design for the reasons of the designer characteristics we presented in Chapter 00, page 00. As neither the lowest cost nor highest performance PDP-11, it has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60: the cache to reduce Unibus traffic; trace-driven design of floating-point arithmetic processors; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus was previously thought to be inadequate for high performance systems, by using a cache most processor references do not use the Unibus and so leave it free for i/o traffic. This gives high performance without the attendant cost. The cache work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since fetching multiple words would tie up the Unibus with additional traffic.

The use of trace data to design the floating point arithmetic is described together with the resulting design. Note that the 11/60 performs roughly at 11/70 speeds but at lower cost. The implementation of the two can be compared in the following table.

Table - Implementation of 11/60 and 11/70 (count of printed circuit boards).

	11/60	11/70	
Base Pc	4	8	~~[CM: check 11/70 sys. manual]~~
Floating point	4	4	
Cache	1	4	
Memory management	1	2	
Total	10	18	

Microprogramming is used to provide both increased user-level capability and increased reliability, availability and maintainability. The writeable control store option is described together with its novel use for data storage. This option has been recently used for emulating the PDP-8 at the OS/8 operating system level.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 2, page 00). Other semiconductor technology improvements are described together with how they affect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

IMPACT OF IMPLEMENTATION DESIGN TRADEOFFS ON PERFORMANCE: THE PDP-11, A CASE STUDY

This chapter presents a comprehensive comparison of the eight processor implementations used in the ten PDP-11 models. The work was carried out to investigate various design styles for a given problem, namely the interpretation of the PDP-11 ISP. The tables alone give more insight into processor implementations than is available from any single source we know. The usefulness of the data also comes from having an outside observer examine the machines and share his insight.

The tables include:

1. a set of instruction frequencies, by Strecker, for a set of ten different applications. The reader should note the frequencies do not reflect all uses, e.g., there are no floating point instructions, nor has operating system code been analyzed.

2. implementation cost (modules, ICs, control store widths) and performance (micro- and macro-instruction times) for each model; and

3. a canonical data path for all 11 implementations, against which each processor is compared.

*Ability to interconnect a number of subsystems together through a wired-or connection.

With this background data, a top-down model is built which explains the performance (macro-instruction time) of the various implementations in terms of the micro-instruction execution, and primary memory cycle time.

Since these two parameters do not fully explain (model) performance, a bottom-up approach is also used. These factors include various design techniques and the degree of processor overlap. We believe that this analysis of a constrained problem should provide useful insight to both computer and general-digital-systems designers.

EVALUATION

TURNING COUSINS INTO SISTERS: THE ROLE OF SOFTWARE IN SMOOTHING HARDWARE DIFFERENCES

Since FORTRAN is probably the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first FORTRAN compiler and object (run) time system are described together with the evolutionary extensions to improve performance. The FORTRAN IV-PLUS compiler is only briefly discussed since its improvements, largely due to compiler optimization technology, are less relevant to the 11 architecture.

The chapter title overstates the compatibility problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by essentially providing five separate object (run) time systems and a single compiler. This transparency is provided quite easily using a concept called threaded code discussed in the chapter.

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. In the second version, by recognizing the special high-frequency-of-use cases, e.g., A = 0, A = A + 1, and by having better conventions for three-address operations (to and from the stack), speedups of 1.3 and 2.0 for floating point and integers, were obtained.

It is interesting to compare the virtual machine described with the FORTRAN IV-PLUS machine which uses the floating point processors (on the 11/34, 11/45, 11/55, 11/60, and 11/70). If the FORTRAN machine described in the paper is implemented in microcode and made to operate at FPP speeds, the resulting machines turn out to operate at roughly the same speed and programs occupy roughly the same program space.

THE PDP-11 AFTER THREE DESIGN GENERATIONS

This chapter is a substantially revised version of a paper called "What Have We Learned From the PDP-11?" written for the CMU Computer Science 10th Anniversary, September 1975. This paper was written to critique the original expository paper on the PDP-11 (Chapter 00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined.

The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially critical. The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further aid in understanding the design process.

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standard for interconnection.

We try to provide a set of evolutionary cost-performance metrics so the 11 can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique (within DEC) problem of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a more useful evaluation of the architecture for FORTRAN execution. A complete section is given on the addressing extension, beyond the 11/45 and 11/70 extensions, which required a major perturbation: VAX-11.

The final section, addressed to the research community, describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs.

VAX-11/780: A VIRTUAL ADDRESS EXTENSION TO THE DEC PDP-11 FAMILY

This chapter provides a clean, somewhat terse, yet comprehensive description of the VAX-11 architecture. It is among the best papers on a specific architecture that we know. Since the VAX part of the architecture is so complete in terms of data-types, operators, addressing and memory management, it can also serve as a textbook model and base, case study for architecture in general. Goals, constraints and various design choices are given. The first model, VAX-11/780, is also briefly described.

VAX-11 is the extension to the PDP-11 to provide a large, 32-bit virtual address for each user process. The architecture includes a compatibility mode ~~for~~ which allows PDP-11 programs written for the RSX-11/M program environment to run unchanged. In this way, PDP-11 programs can be moved among VAX and PDP-11 computers, depending on the user's address size, computational and generality needs.

MULTIPROCESSOR RESEARCH COMPUTERS

Three computers, which use the 11 as a basis, were built at Carnegie-Mellon University to carry out research in computer structures and operating systems for multiprocessors. A fourth multiprocessor based on the 16 LSI-11s, called the DEC PULSAR is also discussed.

The first computer, C.mmp, is a 16 processor (11/40's and 11/20's) system with 2.5 million words of shared primary memory. It was built to investigate the programming (and resulting performance) questions associated with having a large number of processors.

The chapter on the second computer, Cm*, is located physically in the Modules Part, as the modules that form Cm* are LSI-11's. Cm* was based on the premise that ultimately, the smallest modular unit, i.e., integrated circuit, used to build digital systems would be a computer. With this premise, we need to understand how to interconnect computers physically, how to assign parts of the application program to the various computers, and how to program the complete structure.

The third computer, C.vmp, was designed to investigate how a production microcomputer, the LSI-11, could be used to build a triplicating, voting high availability computer.

The goals of the first two are performance while the third has reliability as its goal.

We believe that technology will <u>force</u> the evolution of computing structures to converge to three styles of multiprocessor computers: (1) C.mmp-style, for high performance, incremental performance and availability; (2) loosely coupled computers like Cm* to handle specialized processing, e.g., front end, file, and signal processing; and (3) C.vmp-style for high availability motivated by increasing maintenance costs.

The technology-force argument is based on history, near term technology, and resulting price extrapolations:

1. MOS technology is increasing in both speed and density faster than the technology, e.g., ECL, from which high performance machines are built.

2. The price per chip of the single-MOS-chip processors decreases at a substantially greater rate than for the low-volume high-performance, special designs. Chips in both designs have high design costs, but the special design enjoys a much lower volume.

3. Relative to all other costs of a system, the processor cost in a low end system is essentially zero.

4. Standards in the semiconductor industry tend to form more quickly for high volume products. For example, in the 8-bit microcomputer market, one type supplies about 50% of the market and three types supply over 90%.

5. A 16-bit processor-on-a-chip, with both an address space matching its performance and appropriate data-types, has been announced. Such a commodity will form the basis for nearly all future computer designs.

DEC's PULSAR, described below, is a good example of one of the more straightforward applications of this technology. As a result of these factors, the two classes of machines (MOS-processor-on-a-chip-based and low-volume, high-performance-processor-based) have rapidly diverging costs per operation per chip. Furthermore, large scale applications have been slow to form since problem complexity increases more rapidly than program size. Therefore, most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at essentially zero cost from processor-on-a-chip-based processors, large scale computing will come from arrays of processors, just as we build arrays of 64 Kbit ICs to form memory subsystems.

The multiprocessor research projects at CMU have emphasized synthesis and measurement. Operating systems have been built for them and the executions of user programs have been carefully analyzed. All the multiprocessor interferences, overheads, and synchronization problems have been faced; the resultant performance helps to put their actual costs in perspective. Figure HARPY looks at one of the applications, the HARPY speech recognition program, in detail. Here the performance of C.mmp and Cm* is compared with three uniprocessors (KA-10, KL-10, and 11/40).

C.mmp--A MULTI-MINI-PROCESSOR

C.mmp was motivated by the need for more computing power to solve speech recognition/signal processing problems and to understand the multiprocessor software problem. Until C.mmp, only one large, tightly coupled multiprocessor had been built--the Bell Laboratory's Safeguard Computer (BSTJ issue?).

The introductory section describes the economic and technical factors influencing multiprocessor feasibility and argues for the timeliness of the research. Various problems to be researched are given together with a discussion of particular design aspects. For example, since C.mmp is predicated on a common operating system there are two sources of degradation: memory contention and lock contention. The machine's theoretical performance as a function of memory-processor interference is based on Strecker's (1971) work. In practice, because the memory was not built with low-order address interleaving, memory interference was greater than expected. This problem was solved by moving program segments.

As the number of memory modules and processors becomes very large, the theoretical performance (as measured by the number of accesses to the memory by the processors) approaches the memory bandwidth (m/memory-cycle-time) x 1/~~e~~2.[~~ref.?~~ Baskett and Smith, 1976] Thus, with infinite processors, there is not a maximum limit on performance, provided all processors are not contending for the same memory.

Although there is a discussion outlining the design direction of the operating system, Hydra, later descriptions should be read [Wulf et al, 1975]. Since the small address of the PDP-11 necessitated frequent map changes, 11/40s with writeable control stores were used to implement the operating systems calls which change the segment base registers.

The C.mmp which was actually built is shown in Fig. PMSC.mmp.

There are three basic approaches to the effective application of multiprocessors:

1. System-level workload decomposition. If a workload contains a lot of inherently independent activities, e.g., compilation, editing, file processing, and numerical computation, it will naturally decompose.

2. Program decomposition by a programmer. Intimate knowledge of the application is required for this time-consuming approach.

3. Program decomposition by the compiler. This is the ideal approach. However, results to date have been disappointing [Ref. Illiac IV FORTRAN compiler projects].

C.mmp was predicated on the first two approaches. Since the original paper, ALGOL 68, a language with facilities for expressing parallelism in programs, has been implemented. It has assisted greatly with program decomposition. ~~See Figure x.~~

As can be seen in the paper, a model of the lock problem influenced the number of critical sections in the scheduler. Since the paper, the operating system Hydra has been designed and implemented. An extensive description is given in [Wulf et al, 1975].

Two experiments analyzing the performance of C.mmp and Hydra have been reported. The first, by Marathe and Fuller [1977], used a hardware monitor to measure the degradation due to the locking mechanism which is invoked when shared data is accessed. The second, by Oleinick [1978], analyzed user-program-level synchronization. We summarize the results of both below.

The contention for shared resources in a multiprocessor system occurs at several levels. At the lowest level, processors contend at the cross-point switch level for memory. This memory interference is discussed in the chapter. On a higher level there is contention for shared data in the operating system kernel. At higher levels, processes contend for i/o devices and for software processes, e.g., for memory management. ~~The following~~ table Contrefs points to models on experimental data at these different levels in C.mmp.

Contention Level	Reference
user-program	[Oleinick, 1978]
	[Fuller and Oleinick, 1976]
Hydra Kernel objects	[Marathe and Fuller, 1977]
cross-point switch	Chapter XX (C.mmp chapter)
	[Fuller, 1976]

Table Contrefs References for experimental data on contention at each of three levels in the C.mmp system.

Marathe's data show that the shared data of Hydra is organized into enough separate objects that a very small degration (less than 1%) results from contention for these objects. Table LOCK is reproduced from ~~his~~ the Marathe and Fuller paper. He also built a queueing model which projected that the contention level would be about 5% in a 48 processor system.

Oleinick [1978] has used C.mmp to conduct an experimental, as opposed to theoretical, study of implementation of parallel algorithms on a multiprocessor. He studies rootfinder, an extension of the bisection method for finding the roots of an equation. A natural decomposition of the binary search for a root into n parallel processes is to evaluate the function simultaneously at n points.

Under ideal conditions, all processes would finish the function evaluation (required at each step) at the same time, and then some brief bookkeeping would take place to determine the next subinterval for the n processes to work on. Figure ~~i~~Ideal shows this case. However, because the time to evaluate the function is data dependent, some processes complete before others. Moreover, if the bookkeeping task is time consuming relative to the time to evaluate the function, the speedup ratio will suffer.

Oleinick systematically studies each source of fluctuation in performance. The dominant one is the mechanism used for process synchronization. Figure Lockcomp ~~4.2~~ compares the speedup obtained with the four different synchronization primitives. These are as follows:

Replace with α

PM0

PM1

~~P~~ Four different locks for process synchronization are available to the C.mmp user. The spin lock is the most rudimentary. It does not cause an entry to HYDRA; it is a short sequence of instructions which continually tests a semaphore until it can set it successfully. The P and V operations are in fact the following PDP-11 code sequences.

```
P:    CMP SEMAPHORE, #1        ;SEMAPHORE=1?
      BNE P                    ;loop until it is 1
      DEC SEMAPHORE            ;Decrement SEMAPHORE
      BNE P                    ;If neq 0 go to P

V:    MOV #1, SEMAPHORE        ;Reset SEMAPHORE to 1
```

~~+~~ Altho this repeating polling is extremely fast it has two major drawbacks. The first is that the processor is not free to do useful work. The second is that the polling process consumes memory cycles of the memory bank that contains the semaphore.

The kernel semaphore, is implemented in HYDRA, is the low level synchronization mechanism ~~used to~~ intended to be used by ~~for~~ system processes. When a process blocks or is to wake up, a state change ~~is made~~ for that process is made inside the kernel of HYDRA. If a process blocks while trying to P a semaphore, ~~it is~~ the kernel swaps the process from the processor. However the pages belonging to the process are kept in primary memory

The policy module semaphore is intended ~~as~~ to be the user's primary mechanism ~~for p.s.g sync~~ performing synchronization. When a process is blocked, it is not only swapped from the processor, but its pages are written ~~back~~ onto second storage. Often a process is blocked for just a few milliseconds, ~~and y~~ a ~~much~~ ~~smaller length of time~~ ~~than~~ amount compared with the time it takes to update the pages on secondary storage (~~at least~~ a minimum of 32 milliseconds per page). The original policy module semaphore (PMØ) was modified by introducing a delay before beginning the updating. ~~The delay time for is a parameter, e.~~ The modified semaphore is referred to as ~~PM1(e)~~ PM1(e) where e is the delay time in milliseconds.

Figure Lockcomp compares the performance of rootfinder for each of the four methods of synchronization: spin lock, kernel semaphore, PMØ, and PM1(e = 300). The distribution of the F(x) computation was approximately normal with a mean ~~of 72 millisec & standard sd 18. The curve for the PMO sem- shows that as soon as // [illegible] is increased, degradat occurs because the o'head of synch is gtr than the avge comp time The spin lock imple has the best speed up max of about 2.8 for eight processes~~*.

of 72 milliseconds and a standard deviation of 18 milliseconds. The curve for the PMO semaphore shows that as soon as parallelism is increased, degradation occurs because the ~~ob~~ overhead of synchronization is greater than the average compute time. The spin lock implementation has the best speedup maximum of about 2.8 for eight processes.*

* The speed up of 2.8 for 8 processes appears ~~unimpressive~~ unimpressive until one observes that the rootfinder ~~alg~~ algorithm inherently provides less than linear speedup. The theoretical speedup for 8 processes is about 2.9. ~~f~~

~~Kernel~~

~~Spin-lock - the processor continues to execute a short sequence of instructions which repeatedly test the lock.~~

Clearly the impact of process synchronization is a function of the ratio of synchronization time to function evaluation time. Figure ~~4.5~~ Lockrange gives the ranges over which each lock should be used without dominating execution time.

MULTI-MICROPROCESSORS: AN OVERVIEW AND WORKING EXAMPLE

The Cm* work, sponsored by NSF and ARPA, is an extension of earlier NSF-sponsored research [Bell, et al. 1973] on register-transfer-level modules. As LSI and VLSI enable construction of the processor-on-a-chip, it is apparent that low-level, register-transfer modules are passe' for the construction of all but low-volume computers. Although the research is predicated on structures employing a hundred or so processors, this chapter describes the culmination of the first (ten-processor) phase.

The authors motivate their work by appealing to the diseconomy-of-scale arguments which we advanced at the beginning of this section (page 00) and elaborate upon in Part IV. To provide additional context for their research, computer modules (Cm*), multiprocessors (C.mmp) and computer networks are described in terms of performance and problem suitability. The chapter gives a description of the modules structure, together with their associated limitations and potential research problems.

The grouping of processor and memory into modules and the hierarchy of bus structures - LSI-11 bus, Map bus, and Intercluster bus - are radical departures from more conventional computer systems.

The final, most important, part of the chapter evaluates the performance of Cm* for five different problems.

A 50-Computer-Module Cm* is now under construction and is planned to be operational by the end of 1978 for evaluation in 1979. The extension of Cm* is known as Cm*/50 and is shown in Fig. Cm*/50. It will be used to test the CMU ideas on parallel programming methods, fault tolerance, modularity, and the extensibility of the Cm* structure.

C.vmp: THE ARCHITECTURE AND IMPLEMENTATION OF A FAULT TOLERANT MULTIPROCESSOR

C.vmp is a triplicated, voting multiprocessor designed to understand the difficulty (or ease) of using standard, off-the-shelf LSI-11s to provide greatly increased reliability. There is concern for increased reliability because systems are becoming more complex, are used for more critical applications, and because maintenance costs for all systems are increasing. Because the designers themselves carry out and analyze the work, this chapter provides first-hand insight into high reliability designs and the design process--especially its evaluation. The system has operated for several months and the first phase of work is complete.

Several design goals are initially predicated and the work is carried out against the goals.

The goal of software and hardware transparency turned out to be easier to attain than expected, because of an idiosyncrasy of the floppy disk controller. Because the controller effects a word-at-a-time bus transfer from a one-sector buffer, voting can be carried out at a very low level. It is unclear how the system would have been designed without this type of controller; at a minimum, some part of the software transparency goal would not have been met and a significant controller modification would have been necessary.

A number of models are given by which the design is evaluated. From the discussion of component reliabilities the reader should get some insight into the factors contributing to reliability. It should be noted that a custom-LSI-designed voter is needed to get a sufficiently low cost for a marketable C.vmp. While the intent of C.vmp is not a product, it does provide much of the insight for such a product.

PULSAR: A PERFORMANCE RANGE mP SYSTEM

PULSAR is a 16 LSI-11 multiprocessor computer for investigating the cost-effectiveness of multiple microprocessors. It covers a performance range of approximately 1 LSI-11 to better than an 11/70 for simple instructions.

The breadboard system (Fig. ~~1~~ PULSAR), is based on the 11/70 structure, including multiple interrupt levels and 22-bit physical addressing. It does not implement I&D space or supervisor mode, nor does it have Floating Point processors.

The processors communicate with each other (P-Boards), the Unibus Interface (UBI) and a Common Cache and Control via a high-bandwidth, synchronous bus.

The Common Cache and Control contains a large (8K word), direct mapping, shared cache with a 2-word block size, interfacing to the 2- or 4-way interleaved 11/70 memory bus. This prevents the memory subsystem from becoming a bottleneck, in spite of the large reduction in bandwidth demand provided by the cache. The control provides all the mapping functions for both Unibus and processor accesses to memory. The Unibus map registers (a la 11/70) and the process map registers for each processor are held in a single bipolar memory.

The UBI provides the Unibus control functions of a conventional PDP-11. Interrupts will be fielded by the first enabled processor with preferential treatment for any processor in WAIT state.

Each P-Board contains two independent microprocessor chip sets with modified microcode. Internal contention for the adapter is eliminated by running the two processors out of phase to each other. Such contention as does exist is resolved by the mechanism for arbitration of the P-bus itself. The PULSAR has an ASCII console interfacing via a KMC-11 communications controller, with modified microcode. In addition, a debug panel has displays for every stage of the P-bus and controller pipeline.

Console operations are effected by the UBI interrogating or changing a save area for each processor, physically held in the Mapping Array, in response to ASCII console messages over the unibus. Each processor places all appropriate status in the save area on every HALT, and restores from the save area prior to acting upon every CONTINUE or START.

OPERATION

The PULSAR System is pipeline oriented with specific time slots for each processor. This permits a single, simple arbitration mechanism, rather than separate (complex) ones for each resource.

Once the pipeline is assigned to a transaction, the successive intervals of time are assigned to the following resources in order:

1. The mapping array.

2. The address translation logic.

3. The cache.

4. The address validation logic.

5. The data lines of the P-bus.

The memory subsystem, which is not a part of this resource pipeline, has an independent arbitration mechanism. Interfacing between these independent mechanisms is by means of queues.

There are some operations which require more than one access to the same resource in the pipeline. These operations are effectively handled as two transactions. Examples of such operations are Memory Writes and Internal I/O Page (say ~~KT~~ memory-management register) accesses. A memory write may need a second access to the cache for update, while the Internal I/O Page may need another access to the map array.

There are other operations in which the timing does not permit the use of a particular resource in the specific interval that is allocated to that transaction. This happens, for instance, when a Read operation results in a cache miss. The data is not available in time. In this case too a second transaction takes place, initiated when backing store data becomes available.

COST COMPARISON

Cost projections indicate that a multiprocessor will have an increase in parts count over each possible equivalent performance uniprocessor in the range.

This will range from a 20% increase for a 2 Pc mP, approaching 0% at the top of the range. It is to be noted that the 20% premium can be reduced if provision is not made for expansibility over the entire range. Clearly a separate 1 Pc structure can be cost-effective (since this is the LSI-11). The premium is based on parts count only and excludes considerations of cost benefits due to production learning, common spares and manuals, lower engineering costs, etc.

Marathe, M. and Fuller, S. H. A Study of Multiprocessor Contention for Shared Data in C.mmp, Proceedings 1977 ACM SIGMETRICS Conferences, pp. 255-262.

Wulf, W. [and others] The Hydra Operating System, Fifth ACM SIGOPS Symposium on Operating Systems Principles (Nov. 1975).

Fuller, S.H., and Oleinick, P.N. "Initial measurements of parallel programs on a ~~multiprocess~~ multi-mini processor," IEEE CompCon, pp 358-363 (1976).

Baskett, F. and Smith, A.J. "Interference in Multiprocessor Computer Systems with interleaved Memory", CACM 19, 6 pp ~~[illegible]~~ 327-334 (June, 1976)

~~Fu~~

Fuller, S.H. "Price/Performance Comparison of C.mmp and the PDP-10" Proc. of the Third Annual Symposium on Computer Architecture, ~~pp. 19-24 (1968)~~; pp 195-202 (~~Jan~~ 1976).

~~Wulf et al 5th Sysops Nov 75~~

Oleinick P.N. The Implementation of Parallel Algorithms on a Multiprocessor. Ph.D. Thesis, Carnegie-Mellon University Computer Science Dept., (in preparation).

11 Glue Figures and Tables

<u>Note</u> There are corrections here to enter into file. 1/28/78. GB.

Gordon

[illegible]

last edit 12/16/77

latest edit 1/18/78

also, note Steve Teicher needs to ~~add~~ look at a figure.

PDP-11 GLUE

This part could stand alone as a book on the evolution of the PDP-11 computer structure, although it does rely on the conceptual framework of Chapter 1, and the ISPS language description given in appendix 00. The 11 has evolved quite differently from the other computers in this book, and as such provides an independent and interesting story. The factors that have created the machines are clearly market and technology based; they have generated a large number of implementations (ten) over a relatively short (eight-year) lifetime. Because there are multiple implementations spanning a performance range at the same points in time, the PDP-11 provides problems and insight which did not occur in the evolutions of the traditional mini (8 Family); the best cost/performance machines (18-bit), and the high performance machines (the DECsystem 10). The 11 designs cover a range of 500:1 in system price ($500 to $250,000) and 500:1 in memory size (4 Kwords to 2 Mwords).

The part is divided into six sections.

1. Introduction.

   The first chapter (00), published when the 11 was announced, introduces the architecture, gives its goals, and predicts how it might evolve.

The family notion is quite strong, although not specific about members. Chapter 00, What Have We Learned From PDP-11, might be read next in order to get the broadest overview, and best immediate critique of the 11 evolution.

2. Conceptual Basis.

This section contains two papers that form some of the conceptual basis for the models. Strecker, Chapter 00, describes the cache memory mechanism for building high performance models (specifically the 11/70). Levy describes the intercommunication problem among physical components and how busses carry out this task.

3. Implementations.

Of the four implementation chapters, three are on specific implementations (LSI-11, CMU-11 and 11/60) and the fourth (Chapter 00) is a study of all models. This latter chapter provides comparative data on the various implementations together with how the designs fit a conceptual model.

4. Evaluation.

Chapter 00 evaluates the 11 as a machine for executing FORTRAN. What We Have Learned From PDP-11, Chapter 00 discusses aspects of the PDP-11 evolution in terms of the models introduced in Chapter 1.

5. VAX-11.

This paper, by the architect of VAX-11, discusses the new architecture and its first implementation, the VAX-11/780.

6. The multiprocessor research computers of Carnegie-Mellon University.

Three multiprocessors: C.mmp (Chapter 00), Cm* (Chapter 00) and C.vmp (Chapter 00) give examples of a 16-processor multiprocessor, a set of computer modules based on LSI-11 and a triplicated, voting multiprocessor computer for high reliability.

INTRODUCTION

A NEW ARCHITECTURE FOR MINICOMPUTERS--THE DEC PDP-11

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed, to describe the first (11/20) implementation, and to show possible extensions. This was attempted at a time when the whole family architecture had not been worked out or even fully considered.

The computer class definitions (given in 1970) of micro, mini and midi have stood the rest of time for they correspond quite closely to those of Chapter 1.

The major reasons (elaborated upon in Chapter 00) for the disparity between the predicted and actual evolution are:

1. The notion of designing with improved technology, especially for a family, was not understood in 1970. This understanding came later and was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus proved unacceptable for ~~most~~ inter communications at the very high and low end designs. Although this chapter posits a multiprocessor and multiple Unibusses for high end designs, this exact structure did not evolve as a standard. Levy's chapter elaborates on the bus evolution.

3. The ~~physical memory~~ address space for both physical and virtual memory was too small.

4. The particular data-type extensions were not predicted. While floating point arithmetic was discussed, the character string and decimal operations were not described. These data types evolved in response to market need and COBOL -- factors which did not exist in 1970.

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete (by adding memory management and floating point) ISPS description given in Appendix 0 of the book.

CACHE MEMORIES FOR PDP-11 FAMILY COMPUTERS

Chapter 00 by Strecker is included for four reasons: it is a clear exposition of the cache memory structure and its design parameters; the cache is the basis of a fast PDP-8 [Bell et al, 1974], the 11/60 (Chapter 00) the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is good engineering; and finally, the paper is well-written--in fact, it received the award for the Best Paper at the Third Computer Architecture Conference. We also publish it simply to serve as an encouragement and an example for those who should understand and describe their work -- such well-written and relevant papers are only too rare.

The cache design process is implicit in the way in which the work is carried out to determine the structure parameters. The relevant sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge (Chapter 00) uses Strecker's program traces and methodology. Note that it is easy to collect statistics about PDP-11 program behavior since the trace bit in the PS word, permits the 11 to interpret itself on a single-instruction basis. One of the important parameters to understand is the time between changes of context because all real time and multiprogrammed systems have many context switches. To the best of our knowledge this study is unique. The PDP-8 cache design (Chapter 00) shows the effect of segmenting the cache for instructions and data.

A cache design for a PDP-8 [Bell et al, 1974] is summarized ~~in the~~ on page 00. ~~introduction to Part II. Two differences from the 11 work are of interest.~~

The 8 study shows the effect of separating instructions and data whereas the 11 does not.* ~~The second difference is that~~ Strecker gives the performance evaluation in terms of cache miss ratios whereas the reader is ~~8 data are~~ probably interested in performance or speed-up. ~~in terms of a speed up factor~~. These two ~~performance~~ measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

p	= total no. of memory accesses by the processor, Pc
m	= no. of memory accesses that are missed by the cache and how to be referred to Mp
t.c	= cycle time of cache memory, Mc
t.p	= cycle time of primary memory, Mp
R	= t.p/t.c (ratio of memory speeds), where R is typically 3 to 10

the relative execution speeds are:

$$t(\text{no cache}) = pR$$
$$t(\text{to cache}) = p + mR$$

$$\text{speedup} = pR/(p + mR) = R/(1 + (m/p)\ R) = a = \text{miss ratio} = m/p$$

therefore

$$\text{speedup} = R/(1 + aR) = 1/(a + 1/R)$$

note that if a = 0 (100% hit), the speedup is R; while

if a = 1 (100% miss), the speedup is R/(1 + R), i.e.,

the speedup is less than 1 (i.e., time to reference both memories)

IMPLEMENTATIONS

A MINICOMPUTER-COMPATIBLE MICROCOMPUTER SYSTEM: THE DEC LSI-11

Although the paper is a descriptive narrative about the design at each of the chip, board, and backplane levels, it lacks insight that the designers at Western Digital or DEC (Duane Dickhut, Lloyd Dickman, Rich Olsen, or Mike Titelbaum) might have provided. It was written from the viewpoint of a knowledgeable user. An account of the chip-level design is available (Soha and Pohlman, 1974). The design was done at Western Digital by Roberts, Soha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with the designers to effect interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP, and the particular PMS-level components as they are integrated into a backplane to form a hardware system. Sebern points out the microprogramming tradeoff that took place between the chip and module levels to carry out functions normally in hardware: the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

~~The design alternatives, and uniqueness of the board level structure are not described; hence we felt compelled to discuss them in Chapter 00.~~

[note I took out page 8?]

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. For example, a bounded design that is typical of one board microcomputers was considered, though not described. However, a lower-cost, one-board system, like the VT78, (page 00) has evolved and is shown in Fig. RXT-11. The RXT-11 is an integrated system containing an LSI-11 chip set, 32 Kwords of memory, connectors for six EIA interfaces, and a controller for two floppy disk drives. One-hundred and seventy-five i.c.'s were used -- to implement the same functionality using standard LSI-11 modules, 375 i.c.'s would have been used. ~~[GB: Not announced yet -- probably March]~~.

The initial module-level design for the LSI-11 was predicated on a quad-sized form factor with a conventional backplane. The modules are shown on page 00. Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was devised. It provided interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus (requires a double sized module). This permitted multiple board, complex options, e.g., a disk controller, to be packaged as a single option with no interconnection between the boards except via the second half of the quad board. As DEC began to build special ICs to interface to the bus, the option sizes decreased to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options.

~~Table LSI-11 Options~~

~~[Teicher to supply]~~ — see next page

no ¶

The effect of a lower cost LSI-11 system is to provide additional applications as we discuss in Chapter 1. ~~page 00.~~

USING LSI PROCESSOR BIT-SLICES TO BUILD A PDP-11--A CASE STUDY IN MICROCOMPUTER DESIGN

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, herein called a Microcomputer (for Microprogrammed Processor), is used to interpret a PDP-11 ISP. The purpose of the design was to test the assertion that the bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (a real ISP) to fit the building blocks (the Intel 3000 processor). Indeed, the authors observed awkwardnesses that ultimately resulted in lower (than desired) performance. In retrospect, the Intel 3000 has not become the standard bit-slice architecture that the AMD 2900 series has; perhaps it suffered from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was the evaluation of the computer-aided-design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the major components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction of a design. The authors claim that 95% of the errors were detected in this manner. The simulated-time/real-time ratio ($10^{6/1}$) did constrain the design process, e.g., the number of different runs used to find worst-case conditions.

DESIGN DECISIONS FOR THE PDP-11/60 MID-RANGE MINICOMPUTER

Unlike the reports from an architect's or reporter's viewpoint this chapter is a direct account of the design from the close proximity of the project. A mid-range machine is an inherently difficult design for the reasons of the designer characteristics we presented in Chapter 00, page 00. **[CM: unclear]**. As neither the lowest cost nor highest performance PDP-11, it has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60: the cache to reduce Unibus traffic; trace-driven design of floating-point arithmetic processors; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus was previously thought to be inadequate for high performance systems, by using a cache most processor references do not use the Unibus and so leave it free for i/o traffic. This gives high performance without the attendant cost*. The cache work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since fetching multiple words would tie up the Unibus with additional traffic.

The use of trace data to design the floating point arithmetic is described together with the resulting design. Note that the 11/60 performs roughly at 11/70 speeds but at lower cost. The implementation of the two can be compared in the following table.

Table - Implementation of 11/60 and 11/70 (count of printed circuit boards).

	11/60	11/70
Base Pc	5	8 **[CM: check 11/70 sys. manual]**
Floating point	4	4
Cache	1	4
Memory management	1	2
Total	10	18

Microprogramming is used to provide both increased user-level capability and increased reliability, availability and maintainability. The large

*Using Amdahl's constants, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus. [CM: **We should do this for all models in "What We've Learned" Chapter**] writeable control store option is described together with its use for data storage and various applications. This option has been recently used for emulating the PDP-8 at the OS/8 operating system level.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 1, page 00). Other semiconductor technology improvements are described together with how they affect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

This chapter presents a most comprehensive comparison of the eight processor implementations used in the ten PDP-11 models. The work was carried out to investigate various design styles for a given problem--the interpretation of the PDP-11 ISP. The tables alone give more insight into processor implementations than is available from any single source we know. The usefulness of the data also comes from having an outside observer examine the machines and share his insight.

The tables include:

1. a set of instruction frequencies, by Strecker, for a set of ten different applications. The reader should note the frequencies do not reflect all uses, e.g., there are no floating point instructions, nor

*Ability to interconnect a number of subsystems together through a wired-or connection.

has operating system code been analyzed.

2. implementation cost (modules, ICs, control store widths) and performance (micro- and macro-instruction times) for each model; and

3. a canonical data path for all 11 implementations, against which each processor is compared.

With this background data, a "top-down" model is built which explains the performance (macro-instruction time) of the various implementations in terms of the micro-instruction execution, and primary memory cycle time.

Since these two parameters do not fully explain (model) performance, a set of bottom-up factors must be introduced. These factors include various design techniques and the degree of processor overlap. We believe that this analysis of a constrained problem should provide useful insight to both computer and general-digital-systems designers.

EVALUATION

TURNING COUSINS INTO SISTERS: THE ROLE OF SOFTWARE IN SMOOTHING HARDWARE DIFFERENCES

Since FORTRAN is quite possibly the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first

FORTRAN compiler and object (run) time system are described together with the evolutionary extensions to improve performance. The FORTRAN IV-PLUS compiler is only briefly discussed since its improvements, largely due to compiler optimization technology, are less relevant to the 11 architecture.

The chapter title overstates the compatibility problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by essentially providing five separate object (run) time systems and a single compiler. This transparency is provided quite easily using a concept called threaded code. This concept appears to be a very simple interpreter for the PDP-11--and might not be called an interpreter by many. With threaded code, one 1-word instruction requiring two memory cycle times is executed each time a high level operation code is to be interpreted, otherwise the processor is carrying out the desired op code. When a simple integer expression like I = I + 1, which occupies 2 memory words and requires 3 memory cycles to execute, is transformed into a threaded code version the program still only occupies 2 words, but instead requires 5 memory cycles to execute (nearly a factor of 2). For more complex operations requiring longer execution times, like floating point arithmetic, the overhead turns out to be quite low and the space utilization is quite good. It is also possible to move efficiently between threaded and directly executed code, although this is not done. Jim Bell discovered the technique; it has been used extensively for other compilers because of its low time and space overhead. The ability to carry out the interpretation so elegantly was not part of the original PDP-11 design, but rather was a consequence of the generality of the 11's addressing modes.

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. In the second version, by recognizing the special high-frequency-of-use cases, e.g., A = 0, A = A + 1, and by having better conventions for three-address operations (to and from the stack), speedups of 1.3 and 2.0 for floating point and integers, were obtained.

It is interesting to compare the virtual machine described with the FORTRAN IV-PLUS machine which uses the floating point processors (on the 11/34, 11/45, 11/55, 11/60, and 11/70). If the FORTRAN machine described in the paper is implemented in microcode and made to operate at FPP speeds, the resulting machines turn out to operate at roughly the same speed and programs occupy roughly the same program space.

WHAT HAVE WE LEARNED FROM THE PDP-11?

This chapter is a substantially revised version* of a paper called "What Have We Learned from the PDP-11?" written for the CMU Computer Science 10th Anniversary, September 1975. This paper was written to critique the original expository paper on the PDP-11 (Chapter 00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined.

The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially

~~*The paper is 50% longer.~~ ~~The introductary overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.~~

critical. The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further aid in understanding the design process.

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standrad of interconnection.

We try to provide a set of evolutionary cost-performance metrics so the 11 can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique (within DEC) problem of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a more useful evaluation of the architecture for FORTRAN execution. A complete section is given on the addressing extension, beyond the 11/45 and 11/70 extensions, which required a major perturbation: VAX-11.

The final section, addressed to the research community, describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs.

VAX-11/780: A VIRTUAL ADDRESS EXTENSION TO THE DEC PDP-11 FAMILY

[MJ add this to the 11ghee

when operating with 32-bit addresses, call native mode, the VAX part of the architecture is in effect.

ISP

is also briefly described, ~~to~~ ~~for~~

~~and the reader can observ~~

~~perharst the best~~

Chapter 00, by Bill Strecker, ~~is the cleanest, most comprehensive~~
~~description of and architecture that~~

provides a clean, somewhat terse, yet comprehensive ~~descriptive~~ description
of ~~ten~~ (a) VAX-11 Architecture. It is ~~t~~ among the best papers on a specific architecture
that we know ~~of~~ ~~also 11/780 imp.~~ The first model, VAX-11/780,

VAX-11 is ~~the~~ the extensi~~x~~on to PDP-11 to provide ~~a~~ large ~~size~~, 32 bit
virtual address for ~~a~~ each user process. The ~~extension~~ architecture includes a ~~to~~ PDP-11 (compatibility) mode for ~~running~~ PDP-11 programs written for the RSX-11/M program environment to run unchanged. ~~but~~ In this wa~~y~~y, ~~the~~
PDP-11 programs can be moved ~~to VAX and~~ among VAX and PDP-11 computers ~~in~~, depending on the user's address size, computational and generality needs.
~~a clealearly compatible fashion.~~

~~The description~~

~~The description proceeds around gives includes te the data-types~~

~~There is a~~
The ~~architectural description is~~ ~~also includes commentar a~~ ~~the~~ goals ~~and~~, constraints and
~~a that effected lead to~~ various design choices, ~~and the resultant architecutre is so comprehensive,~~ are ~~[illegible]~~ given. ~~as the~~ ~~The steps~~
~~that it can almost be used as a base study for all other architectures in taht it~~
~~includes a wide variedty of data-types and operators, and . The memory addressing~~
~~and segmentation scheme also seerves as can serve as a model of comparision.~~

Since ~~VAX~~ the VAX part of the architecture is so complete in terms of data-types, operators, ~~and~~ addressing and memory management it can also serve as a textbook model ~~for other arch company~~ and base, case study for architecture in general.

~~There~~
The descript

~~The evolution of the 11.~~

~~[Section on Strecker's paper to go here]~~

THE MULTIPROCESSOR RESEARCH COMPUTERS OF CARNEGIE-MELLON UNIVERSITY AND DEC's PULSAR

Three computers, which use the 11 as a basis, were built at Carnegie-Mellon University to carry out research in computer structures and operating systems for multiprocessors. A fourth multiprocessor, based on the 16 LSI-11s, called the DEC PULSAR, is ~~given in~~ also discussed

The first computer, C.mmp, is a 16 processor (11/40's and 11/20's) system with 2.5 million words of shared primary memory. It was built to investigate the programming (and resulting performance) questions associated with having a large number of processors.

The chapter on the second computer, Cm*, is located physically in the Modules Part, as the modules that form Cm* are LSI-11's. Cm* was based on the premise that ultimately, the smallest modular unit, i.e., integrated circuit, used to build digital systems would be a computer. With this premise, we need to understand how to interconnect computers physically, how to assign parts of the application program to the various computers, and how to program the complete structure.

The third computer, C.vmp, was designed to investigate how a production microcomputer, the LSI-11, could be used to build a triplicating, voting high availability computer.

The goals of the first two are performance while the third has reliability as its goal.

We believe that technology will <u>force</u> the evolution of computing structures to converge to three styles of multiprocessor computers: (1) C.mmp-style, for high performance, incremental performance and availability; (2) loosely coupled computers like Cm* to handle specialized processing, e.g., front end, file, and signal processing; and (3) C.vmp-style for high availability based on increased maintenance costs.

The technology-force argument is based on history, near term technology, and resulting price extrapolations:

1. MOS technology is increasing in both speed and density faster than the technology, e.g., ECL, from which high performance machines are built.

2. The price per chip of the single-MOS-chip processors decreases at a substantially greater rate than for the low-volume high-performance, special designs. Chips in both designs have high design costs, but the special design enjoys a much lower volume.

3. Relative to all other costs of a system, the processor cost in a low end system is essentially zero.

4. Standards in the semiconductor industry tend to form more quickly for high volume products. For example, in the 8-bit microcomputer market,

one type supplies about 50% of the market and three types supply over 90%.

5. A 16-bit processor-on-a-chip, with both an address space matching its performance and appropriate data-types, has been announced. Such a commodity will form the basis for nearly all future computer designs.

~~Pulsar~~ DEC's PULSAR, ~~see~~ described below, is a good example of one of the more straightforward applications of this technology.

As a result of these factors, the two classes of machines (MOS-processor-on-a-chip-based and low-volume, high-performance-processor-based) have rapidly diverging costs per operation per chip. Furthermore, large scale applications have been slow to form since problem complexity increases more rapidly than program size. Therefore, most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at essentially zero cost from processor-on-a-chip-based processors, large scale computing will come from arrays of processors, just as we build arrays of 64 Kbit ICs to form memory subsystems. ~~This type~~

C.mmp--A Multi-Mini-Processor

C.mmp was motivated by the need for more computing power to solve speech recognition/signal processing problems and to understand the multiprocessor software problem. Until C.mmp, only one large, tightly coupled multiprocessor had been built--the Bell Laboratory's Safeguard Computer (BSTJ issue?).

The introductory section describes the economic and technical factors influencing multiprocessor feasibility and argues for the timeliness of the research. Various problems to be researched are given together with a discussion of particular design aspects. For example, since C.mmp is predicated on a common operating system there are two sources of degradation: memory contention and lock contention. The machine's theoretical performance as a function of memory-processor interference is based on Strecker's (1971) work. In practice, because the memory was not built with low-order address interleaving, memory interference was greater than expected. This problem was solved by moving program segments.

As the number of memory modules and processors becomes very large, the theoretical performance (as measured by the number of accesses to the memory by the processors) approaches the memory bandwidth (m/memory-cycle-time) x 1/e.**[ref.?]** Thus, with infinite processors, there is not a maximum limit on performance, provided all processors are not contending for the same memory.

Although there is a discussion outlining the design direction of the operating system, Hydra, later descriptions should be read [Wulf et al, 1975]. Since the 11's small address necessitated frequent map changes, 11/40s with writeable control stores were used to implement the operating systems calls to change the segment base registers.

The C.mmp which was actually built is shown in Fig. PMSC.mmp.

There are three basic approaches to the effective application of multiprocessors:

1. System-level workload decomposition. If a workload contains a lot of inherently independent activities, e.g., compilation, editing, file processing, and numerical computation, it will naturally decompose.

2. Program decomposition by a programmer. Intimate knowledge of the application is required for this time-consuming approach.

3. Program decomposition by the compiler. This is the ideal approach. However, results to date have been disappointing [Ref. Illiac IV FORTRAN compiler projects].

C.mmp was predicated on the first two approaches. Since the original paper, ALGOL 68, a language with facilities for expressing parallelism in programs, has been implemented. It has assisted greatly with program decomposition. See Figure x.

As can be seen in the paper, a model of the lock problem influenced the number of critical sections in the scheduler. Since the paper, the operating system Hydra has been designed and implemented. An extensive description is given in [Wulf _et al_, 1975].

Two experiments analyzing the performance of C.mmp and Hydra have been reported. The first, by Ranathe [1977], used a hardware monitor to measure the degradation due to the locking mechanism which is invoked when shared data is accessed. The second, by Oleinick [1977], analyzed user-program-level synchronization. We summarize the results of both below.

The contention for shared resources in a multiprocessor system occurs at several levels. At the lowest level processors contend at the cross-point switch level for memory. This memory interference is discussed in the chapter. On a higher level there is contention for shared data in the operating system kernel. At higher levels, processes contend for i/o devices and for software processes, e.g., for memory management. The following table points to models on experimental data at these different levels in C.mmp.

<u>Contention Level</u>	<u>Reference</u>
user-program	[Oleinick, 1977]
	[Fuller and Oleinick, 1976]
Hydra Kernel objects	[Marathe and Fuller, 1977]
cross-point switch	Chapter XX
	[Fuller, 1976]

Ranathe's data show that the shared data of Hydra is organized into enough separate objects that a very small degration (less than 1%) results from contention for these objects. Table LOCK is reproduced from his paper. He also built a queueing model which projected that the contention level would be about 5% in a 48 processor system.

Oleinick uses a root finding algorithm to study various implementations of a single problem.

[CM: elaborate after studying his thesis chapter.]

Multi-Microprocessors: An Overview and Working Example

The Cm* work, sponsored by NSF and ARPA, is an extension of earlier NSF-sponsored research [Bell, etc. 1973] on register-transfer-level modules. As LSI and VLSI enable construction of the processor-on-a-chip, it is apparent that low-level, register-transfer modules are passe' for the construction of all but low-volume computers. Although the research is predicated on structures employing a hundred or so processors, this chapter describes the culmination of the first (ten-processor) phase.

The authors motivate their work by appealing to the diseconomy-of-scale arguments which we advanced at the beginning of this section (page 00). To provide additional context for their research, computer modules (Cm*), multiprocessors (C.mmp) and computer networks are described in terms of performance and problem suitability. The chapter gives a description of the modules structure, together with their associated limitations and potential research problems.

The final, most important, part of the chapter evaluates the performance of Cm* for five different problems.

C.vmp: The Architecture and Implementation of a Fault Tolerant Multiprocessor

C.vmp is a triplicated, voting multiprocessor designed to understand the difficulty (or ease) of using standard, off-the-shelf LSI-11s to provide greatly increased reliability. There is concern for increased reliability because systems are becoming more complex, are used for more critical applications, and because maintenance costs for all systems are increasing. Because the designers themselves carry out and analyze the work, this chapter provides first-hand insight into high reliability designs and the design process--especially its evaluation. The system has operated for several months and the first phase of work is complete.

Several design goals are initially predicated and the work is carried out against the goals.

The goal of software and hardware transparency turned out to be easier to attain than expected, because of an idiosyncrasy of the floppy disk controller. Because the controller effects a word-at-a-time bus transfer from a one-sector buffer, voting can be carried out at a very low level. It is unclear how the system would have been designed without this type of controller; as a minimum, some form of software transparency goal would have been violated and a significant controller modification would have been necessary.

A number of models are given by which the design is evaluated. Various component reliabilities are used and the reader should get a great deal of insight into the factors contributing to reliability. It should be noted that a special hardware voter is needed to get a sufficiently low cost for a marketable C.vmp. While the intent of C.vmp is not a product, it does provide much of the insight for such a product.

PULSAR here

Marathe, M. and Fuller, S. H. A Study of Multiprocessor Contention for Shared Data in C.mmp, Proceedings 1977 ACM SIGMETRICS Conferences, pp. 255-262.

Wulf, W. [and others] The Hydra Operating System, Fifth ACM SIGOPS Symposium on Operating Systems Principles (Nov. 1975).

11 Glue Figures and Tables

Table of LSI-11 Options and Extensions (1978) for LSI-11/2

	LSI-11	LSI-11/2	Additions
Basic			
Processor + 4 Kw + send line	quad	double	
Real time clock	quad		
Power controller/ sequences	double		
Memories			
4 Kw RAM	double		
4 Kw core	quad		
32 Kw RAM	quad	double	
4 Kw PROM	double		
4 Kw PROM; 256 RAM	double		
Interfaces			
Parallel (16 line)	double		
Asynchronous (serial)	double		
Instrument (IEEE-488)	double		
Direct Memory Access	quad		
Foundation (general purpose)	quad		
Analog-to-digital	quad		
Digital-to-analog	quad		
Floppy interface		double	

MJ — please ask Steve ~~to~~ Teicher to add to this set and check both

Table of LSI-11 Options and Extensions (1978) for LSI-11/2

	LSI-11	LSI-11/2	Additions
Basic			
Processor + 4Kw + serial line	quad		
Processor		double	
Memories			
4 Kw ram.	double.		
~~32~~16 "	quad	~~double~~	
~~4 kw prom~~	~~double~~		
4 Kw prom; 256 ram	double.		
4 Kw core	quad		
32 K w RAM	quad	double	
Interfaces			
Parallel (16 line)	double		
~~Seri~~ 4 line ~~serial~~ Asynchronous		double	—
Asynchronous (serial)	double		
Instrument (IEEE-488)	double		
Direct Memory Access	quad		
Foundation (general purpose)	quad		
~~4 Real time cl~~ Fancy RT clock	quad		
Analog-to-digital	quad		
Digital-to-analog	quad		
Floppy interface	←	double	
Real time clock	quad		
Power controller / sequencer	double		
Hard disk interface (RK05)	4 quads		
(RL01)	2 quads		
line printer	double		
Back planes			
4×8	for quads		
2×4			
2×8			
2×12		~~6~~ 6	
4-line synchronous ~~DUV~~	quad		
4-line ASYN mux ~~DZV~~	quad		

PDP-11 glue all caps

edited 12/15/77

structure

is quite capable of standing alone as

This part ~~could have~~ ~~stood alone as~~ a ~~single~~ book on the PDP-11 computer evolution, ~~For a number of reasons~~ the 11 has evolved quite differently than the other computers in this book, and as such ~~a has~~ ~~this is~~ ~~quite~~ provides an interesting story.

although it relies heavily on the conceptual framework of Chapter 1, and the ISPS language description given in ~~the~~ appendix 00.

~~a much diof id different story.~~ ~~For~~ ~~Per~~ ~~Two factors seem~~

The factors ~~th~~ that ~~seem to~~ have created the machines are clearly market and technology based, generating a ~~numb~~ large number (ten) of different implementations over ~~a~~ relatively short ~~1e~~ seven year lifetime. Thus, the fact ~~aht~~ that there is an expanding a range of machines ~~causes~~ as we move along in time provides ~~the engineeri~~ ~~we~~ ~~with a set f~~ of problems and insight ~~that is not natural to some of the~~ ~~do occur in~~ in addition to ~~other designs. of~~ the traditional mini (8 Family), ~~or~~ the best cost/performance (18-bit), and ~~or~~ the ~~higest~~ high performance, ~~that one can build with a that is~~ less than 500K (the DECsystem 10). ~~Here, we~~ The 11 designs cover a range of a factor of 500 in system price ~~and~~ (\$500 to \$250,000) and 500 in memory size (~~5~~ 4 Kwords to 2 Mwords). ~~The other~~ ~~Because~~

~~Becuase the 11 is a relatively low cost, it is an excellent vechicle to use in Computer Systems and C Computer Structures research to test various ideas in ordfer that structures may b evolve.~~

In

This ~~section is~~ Part is ~~really~~ divided into ~~a~~ five sections consisteing of :

1. ~~The~~ Introduction. ~~containsin the paper on the original paper on the 11 architecture~~ first chapter published when the 11 was announced

This ~~paper~~ ~~introud~~ introduces the architecture, ~~and~~ gives various goals for it, and predictes how it might eveolve. The family notion is quite strong, although not specific.

~~Alternatively,~~ Chapter 00, What Have We Learned From PDP-11,

May preferably be read next ~~after chapter 00~~ in order to get ~~an~~ the broadest overview, ~~of~~ and best immediate critique of chapter 00.

2. ~~Conceptual~~ ~~The~~ Conceptual basis ~~for the evoution of~~. This section ~~cont~~ ~~A has~~ ~~par Streckert's paper~~ on the Cache Memory ~~both describes the concept and shows how the implementation is carried~~ out.

5. the ~~Multip~~ multiprocessor research computers of Carnegie-Mellon University, Three ~~comp~~ multiprocessors: C.mmp (chapter 00), Cm* (chapter 00), and C.vmp (chapter 00) ~~are~~ ~~described~~
give examples of ~~mult actual~~ ~~multipro~~ a 16 processor multiprocessor, a ~~co~~ set of ~~compu~~ modules based on LSI-11 and a ~~voting~~ triplicated, voting ~~multi co~~ multiprocessor computer for high reliability.

~~Levy's chapter on busses as the form the backbone.~~

contains two papers, that combined with Chapter 00,

form ~~the concepts~~ ~~many of the~~ some of the

conceptual basis for basis for the models

Strecker, Chapter 00, describes the cache memory

mechanism as a basis for building high

performance models (specifically the 11/70). Levy describes ~~various~~

~~p~~

c the intercommunication problem ~~of~~ among physical components

and how busses carry out ~~the~~ this task.

3. Implementations. ~~Four chapters ~~ ~~give~~ Of the

~~Four cha~~,

~~sp~~ four chapters, three are on specific implementations

(LSI-11, ~~11/60 and~~ CMU-11 and 11/60) and the

fourth ~~describes~~ a (Chapter 00) ~~giv~~ ~~gives a~~ is on

~~model for the p~~ all models, ~~the~~ This latter

chapter provides both ~~a conceptual model~~ ~~as~~ ~~and it~~ details

of the various implementations together with how the designs fit a conceptual

model. This is especially ~~vital~~ useful to understand the performance.

4. Evaluation, Chapter 00 evaluates the 11 ~~fs~~ as a

machine for executing Fortran. What We Have Learned From

PDP-11, Chapter 00 ~~evaluat~~ discusses all aspects of

the evolution in terms of ~~Chapter 00.~~ the introductory chapter (00).

over

Note: When single-spaced, all footnotes will appear only once on the appropriate page.

Gardon

~~PDP-11 Glue 12/7/77~~

~~INTRODUCTION~~

A New Architecture for Minicomputers--The DEC PDP-11

allcaps

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed to describe the first (11/20) implementation at a time when the whole architecture had not been worked out or even fully considered; and to show possible extensions.

The reader might note that the computer class definitions (given in 1970) of micro, mini and midi correspond quite closely to those of Chapter 1.

Although we comment on the disparity between the predicted and actual evolution in Chapter 00, some of the important reasons are:

1. The notion of designing with improved technology especially for a family was not understood then. The understanding for one of us (Bell), was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus bandwidth proved unacceptable for all communications at the very high and low end designs. Whereas, this chapter posits a multiprocessor, and multiple Unibusses, this precise sturcture did not evolve. The bandwidth has subsequently been shown to be adequate for

all but the largest configurations when a cache is attached to the processor as in the 11/60 (see Chapter 00). Note the effect of a 90% cache h ~~bit~~ rate is to reduce the number of access to primary memory ~~Mp~~ via the Unibus by a factor of ten!

3. The memory address space was too small.

4. The particular data-type extensions weren't predicted. While floating point data was discussed, the character string and decimal operations weren't described. These data types evolved in response to market need (and COBOL) which didn't exist for DEC at the time the machine was designed.

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete (memory management and floating point) ISPS description that was used when the 11 was evaluated as a Computer Family Architecture (CFA) standard by the U. S. Defense Department.

Caps

<u>Cache Memories for PDP-11 Family Computers</u>

Chapter 00 by Strecker is included for four ~~three~~ reasons: it is a clear exposition of the cache memory structure and its design parameters; ~~since~~ the cache is the basis of the fast PDP-8 (Chapter 00), the 11/60 (Chapter 00), ~~and~~ the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is "good engineering"; and finally the paper is "well-written"--in fact, it received the award for the Best Paper at the

Third Computer Architecture Conference at which it was presented. Thus, since a relevant paper that provides a clear exposition, while illustrating good engineering and being well-written, is so rare we also publish it simply to serve as an encouragement and an example for those who should expose, understand and describe their work.

The design process is implicit in the way the work is carried out to determine the structure parameters. It should be noted that it was easy to collect data statistics about the program's behavior since the trace bit, T permits the 11 to interpret itself on a one at a time basis. The relevant parameters are varied and sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge and Strecker's traces. also uses the same methodology (Chapter 00). One of the important parameters, to understand is the time between changes of context, because all ~~is especially important~~ and to the best of our knowledge this study is ~~both~~ unique ~~and necessary for any~~ real time ~~or~~ and multiprogrammed system. ~~behave~~ have many context switches The PDP-8 cache design (Chapter 00) shows the effect of segmenting the cache for instructions and data. The PDP-8 chapter discusses the performance for a particular design, and the performance numbers are in terms of a speed-up factor. Strecker gives the performance evaluation in terms of cache miss ratios. the ~~P~~erformance measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

~~using~~ Fig. Cachespeed

~~P~~ = total no. of memory accesses by the processor, Pc

~~M~~ = no. of memory accesses that are missed by the cache and how to be referred to Mp

t.c = cycle time of cache memory, Mc

t.p ~~top~~ = cycle time of primary memory, Mp

R = t.p/t.c (ratio of memory speeds), where R is typically 3 to 10

the relative execution speeds are:

t(no cache) = pR

t(to cache) = p + mR

speedup = pR/(p + mR) = R/(1 + (m/p) R) = a = miss ratio = m/p

therefore

speedup = R/(1 + aR) = 1/(a + 1/R)

note that if a = 0 (100% hit), the speedup is R; while

if a = 1 (100% miss), the speedup is R/(1 + R), i.e.,

the speedup is less than 1

IMPLEMENTATIONS

[illegible]

Caps A Minicomputer-Compatible Microcomputer System: The DEC LSI-11

This first chapter of the implementation section was written from the knowledgeable user viewpoint. Although the paper is a descriptive narrative of the design from the chips, boards and backplane levels, it lacks ~~some~~ insight that ~~from one of~~ the designers at Western Digital or DEC (Lloyd Dickman, Rich Olsen, or Mike Titelbaum) might have provided. An account of the chip-level design is available (Zorsz. Soha and Pohlman, 1974). The design was done at Western Digital ~~corporation~~ by Roberts, ~~Zorsz.~~ S Zoha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with them for use in the interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP; and the particular PMS module level components as they are integrated into a backplane to form a hardware system. Sebern points out ~~an interesting~~ the microprogramming tradeoff that took place between the chip and module levels ~~can be observed in the use of microprogramming~~ to carry out functions ~~that are~~ normally ~~done with~~ hardware; the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. The initial module level design was predicated on a quad-sized form factor and plugged into a conventional backplane. For example, The alternative, ~~more~~ one board bounded design~~s~~ that ~~are~~ is typical of ~~other~~ microcomputers, was considered, though ~~is~~ not described. ~~However it should be noted that in order to get more reduced~~ A lower cost~~, such a~~ one board system, like the VT78 (page 00), has evolved and will eventually be marketed. This structure is shown in Fig. KXT11. The modules are shown on page 00. Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was used for providing interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus. (requires a double sized module) This permitted multiple board, complex options (e.g., a disk controller) to be packaged as a single option with no interconnection between the boards except the second half of the quad board. As DEc began to build special ICs to interface to the bus, the option sizes began to decrease to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options.

Table LSI-11 Options

[Teicher to supply]

The effect of a lower cost LSI-11 system ~~modules and backplane availability~~ is to provide additional applications as we discuss in Chapter 1, page 00.

Caps

Using LSI Processor Bit-Slices to Build a PDP-11--A Case Study in Microcomputer Design

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, herein called a Microcomputer (for Microprogrammed Processor) ~~IC~~, is used to interpret a PDP-11 ISP. The purpose of the design was to test the assertion that the bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design ~~of~~ and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (11 processor) to fit the building blocks (the Intel 3000 processor). As such, the authors observed the awkwardness that ultimately resulted in lower (than desired) performance. In retrospect, the Intel 3000 has not become the standard ~~(now~~ that the AMD 2900 series has ~~series)~~ bit-slice architecture, but perhaps suffers from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was to evaluate the computer aided design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the key components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction: The authors claim that 95% of the errors were detected by the simulation--and all errors were ultimately detectable once the simulation data or machine description was changed.

Caps

Design decisions for the PDP-11/60 Mid-Range Minicomputer

Unlike the reports from an architect's or reporter's viewpoint this chapter is a direct account of the design from the close proximity of the project. Because it is a mid-range machine, the 11/60 is a difficult design for the reasons of the designer characteristics, Chapter 00, page 00. The design is neither the lowest cost nor performs the ~~most~~ highest of the 11s, but has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60: the cache to reduce Unibus bandwidth; trace-driven design of floating-point arithmetic; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus had previously thought to be inadequate for high performance, the cache is used to unload the i/o traffic and provide high performance without the attendant cost†. The work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since additional traffic is generated by fetching multiple words ~~and thereby tying~~ which would tie up the Unibus ~~for~~ with additional traffic.

See page 8 footnote

The use of trace data to design the floating point arithmetic is described together with the resulting design. ~~It should be~~ note~~d~~ that the 11/60 performs roughly at 11/70 speeds with lower cost. ~~as~~ The implementation of the two can be compared in the following table.

Table — Implementation of 11/60 and 11/70.

	11/60	11/70
Base Pc	x	x
Floating point	x	
Cache	x	x
Memory management	0	x
	______	______
Total	x	x

Th [illegible]

Microprogramming is used to provide both increased user-level capability ?

*Ability to interconnect a number of subsystems together through a wired-or connection.

and increased reliability, availability and maintainability. The large writeable control store option is described together with its use for data storage and various applications. This option has been recently used for emulating the PDP-8 with OS/8 operating systems.

*Ability to interconnect a number of subsystems together through a wired-or connection.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 1, page 00). Other semiconductor technology improvements are described together with how they effect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

See page 7 footnote

Insert 400A, 100B

*6

Cups

EVALUATION

Turning Cousins into Sisters: The Role of Software in Smoothing Hardware Differences

Since FORTRAN is quite possibly the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first FORTRAN compiler and (run) object time system are described together with the evolutionary extensions ~~(evolution)~~ to improve performance. The FORTRAN IV+ compiler is only briefly discussed since its improvements are largely a matter of compiler optimization technology and are less relevant to the 11 architecture.

The chapter title overstates the problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by providing what amounts to five separate object (run) time systems and a single compiler. This transparency is provided quite easily using ~~comes about because~~ a concept called threaded code. This concept appears to be ~~is used to provide what is~~ a very simple interpreter for the PDP-11--and might not be called an interpreter by many. With threaded code, one 1-word instruction requiring two memory cycle times is executed ~~per transfer of control~~ each time the next high level operation code (the interpreted [illegible]) is to be

*Using Amdahl's ~~constraints,~~ constants page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus.

10A

Impact of Implementation Design Tradeoffs on Performance:The PDP-11, A Case Study

This chapter presents ~~a~~

~~This paer presents one~~ ~~One of the~~ most comprehensive

the eight PDP-11 processor implementations. The work was carried out

comparison~~s~~ of ~~machine & (processor impleentatins) is mad is in this~~

to investigate various design styles for a given ~~(fix~~ problem —

~~chpater.~~ Aside from any con~~s~~clusions the~~y~~ ~~are significant~~ tables ~~that~~ give more insight into processor implementations than is available from any source we know. ~~Unlike some of the other dataa this is more objective~~

~~(and examinations include)~~

The ~~signifcant~~ tables include:

1. ~~A~~ set of instruction use frequenc~~i~~ies, ~~that s Ste Strecker determined from~~ by Strecker, for an operating system~~useage~~. (The reader should ~~be care~~ note ~~th hat they~~ the frequencies do not ~~represent a fully~~ reflect general pupose ~~mix,~~ use, i.e. eg. there are no ~~arti~~ arithmetic ~~or fixed poid~~ and floating point ~~data in the~~ instructions~~used.~~

2. ~~All~~ A canonical data path ~~is presented which~~ is posited for all 11 implementations, and each ~~m~~ processor is presented and compared ~~(and~~ with it~~)~~. ~~This is insight that we has not come from~~

~~3. All the execution times for the various models are present in] place.~~

3. Implementation ~~pack~~ cost ~~measures~~ (modules ~~packages and~~ ICs, control store widths) and performance (~~its~~ 'micro' and macro-instruction times)

The useful ness of the data also comes from having an outside ~~or~~ observer examin~~g~~e the machines and ~~and try to~~ provide insight ~~into the machine.~~

We believe

single

With this ~~These tables~~ are background data, ~~the maine puropose of the study~~ chapter

a "top-down" model is built ~~is to build a s modle wth~~ which explains the performance (in macro-instruction time ~~rate~~) of the various implementations in terms of the ~~given the detr d various metrics. The two metrics (as we might expect, are instruction time macro in microc~~

~~The two design parameters~~, micro-instruction execution time, and primary ~~dmem main~~ memory cylce time ~~are used (called the top-down approach) to expain (to withing~~ 85% ~~the behavior (macro-instrucioint times.~~

the ~~implem~~ interpretation of the PDP-11 ISP.

105B

~~Unfortunately~~, The design ~~tricks~~ and overlpa enter the pciture and must be used to finally explain the re tthe discepancises of the model. These design trc

Since these two parameters don't fully explain (model) performance, ~~other~~ a set of bottom-up factors ~~are~~ must be introduced. ~~to more~~ These factors include various design techniques and the ~~dedi~~ degree of processor overlap.

~~We believe the reader should~~ study
~~These implementation as they~~ in
We believe the fact that the problem is constrained should provide useful insight to ~~not only all~~ both computer ~~equipment~~ ~~designers~~ and general digital systems design.

interpreted, otherwise the processor is carrying out the desired op code. ~~For~~ when a simple integer expression like A = A + 1, which occupies 2 memory words and requires 3 memory cycles to execute, is transformed into a threaded code version; the program still only occupies 2 words, but instead requires 5 memory cycles to execute (nearly a factor of 2). For ~~longer~~ more complex operations requiring longer execution times, like floating point arithmetic, the overhead turns out to be quite low and the space utilization is quite good. Jim Bell discovered this technique and it has been used extensively for other compilers because the time and space overhead is so low. The ability to carry out the interpretation so elegantly was not part of the original design, but rather turned out to be possible based on the generality in the 11's indexing modes.

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. By recognizing the special high use frequency cases (e.g., A = 0, A = A + 1, etc.) and by having better conventions for three-address operations to and from the stack, speedups of 1.3 and 2.0 for floating point and integers, respectively, were obtained when a more complex machine was constructed for the second version.

Since this paper is really on the construction of an extension to the 11 to execute FORTRAN, it is interesting to compare it with the FORTRAN IV+ machine which uses the floating point processor (11/45, 11/55, 11/60, 11/70). ~~If~~ The two machines turn out to ~~be roughly~~ operate at roughly the same speed and ~~have~~ programs occupy roughly the same program space ~~efficiency assuming~~ If the FORTRAN machine described in the paper is microprogrammed and made to operate at FPP speeds,

~~*Using Amdahl's constraints, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare the needs with the Unibus.~~

Caps

What Have We Learned From the PDP-11?

This chapter is a substantially revised version* of a paper written for the CMU Computer Science 10th Anniversary, September 1975.

This paper was written to critique the original expository paper on the PDP-11 (Chapter 00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined. ¶ The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially critical. The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further ~~attempt an~~ aid in understanding ~~of~~ the design process.

The main section consists of; evaluating; the Unibus, examining the cost-performance evolution, the ISP and ~~asking~~ discussing the organizational issues behind why multiprocessors haven't evolved. As a companion, Chapter 00 ~~[illegible]~~, on C.mmp describes the technical problems associated with multiprocessors.

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standrad of interconnection.

We try to provide a set of evolutionary cost-performance metrics so the 11

*The paper is 50% longer. The introductory overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.

can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique problem the 11 has of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a useful evaluation of the architecture for FORTRAN execution.

A complete section is given on the addressing extension beyond the 11/45 and 11/70 extensions which required a major perturbation in the form of the VAX/11 extensions.

The final section to the research community describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs. (This part has been expanded too.)

*The paper is 50% longer. The introductory overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.

Center and caps

MULTI-PROCESSOR

The RESEARCH ~~MACHINES~~ COMPUTERS OF Carnegie-Mellon University

Irrespective

~~This series of machines~~ These three multiprocessor computers which use the 11 as a basis were built at Carnegie-Mellon University ~~as were designed to~~ to carry out ~~three different aspects of~~ various probl computer structures, operating system and application program Computer engineering and science research. ~~The first~~

Computer

The ~~Carnetgi CMU's~~ first computer, C.mmp, ~~was built to for~~ is a ~~mulit minicomputer, multi, minicomputer processor computer (or minicomputer- is a conventional~~, ~~large~~ ~~hardware structrue~~ 16 processor (model 11/40's) ~~multi-processor~~ system with ~~comp~~ shared 1 million words of primary ~~ma~~memory, ~~and~~ It was built ~~with the expressed purpose of~~ to investigate ~~evaluating~~ the ~~performance quest p a~~ programming (and resulting performance) questions associated with having a large number of ~~multiporcessors~~.

The second computer, Cm*, is ~~most~~ also~~, properly p~~ discussed in the ~~part on modu the mo~~ Modules Part as the modules that forms Cm* are LSI-11 computers. ~~This~~ Cm* ~~evolved from a design as~~ evolution was based on the premise ~~hta~~ that ultimately, (i.e. ultimately ~~the~~ all ICs) ~~modules such as~~ the smallest modular ~~us~~ unit that would be used to build digital systems is the computer ~~that would be built~~. With ~~the~~ this premise, it ~~was~~ is important ~~ot~~ to ~~undertake the research which showed~~ understand how to interconnect them physically, ~~and~~ how to assign parts of the problem to the various computers and how to ~~us uderstand how t the probrramm~~ programming ~~structure~~ them.

The C.vmp, for voting ~~mu~~ multiprocessor, was ~~desg~~ designed to investigate ~~the re d problems associated with taking a an available mult~~ how a production ~~comp~~ microcomputer, the LSI-11, ~~and (turned into)~~ can be used to ~~poon~~ build a triplicating, voting high availability computer.

The goals, ~~therefore, are per performance, perf~~ of the ~~three are~~ first two are performance while the third uses multiple computers for ~~reli~~ redundancy and reliability. To this end, Fig. Perf shows the effect of using multiprocessors to ~~solve~~ execute various algorithms (More to Come Here!)

EKISTICS
ΟΙΚΙΣΤΙΚΗ
REVIEWS ON THE PROBLEMS AND SCIENCE OF HUMAN SETTLEMENTS
EKISTICS is published by the Athens Center of Ekistics • 24, Strat. Syndesmou St. • Athens 136, Greece, P.O. Box 471
EDITORIAL OFFICE: Page Farm Road • Lincoln, Massachusetts 01773 • USA • 617-259-9144

Dear Reader,

Thank you for ordering EKISTICS. We hope that you will find it useful and become a regular reader.

EKISTICS is interdisciplinary and deals with questions of human settlements as a unified whole. This perspective is part of a broad concensus among planners, environmentalists, teachers, researchers, architects, and policymakers.

Each month we explore a different topic which is determined in consultation with our expert advisory board and through suggestions from our readers. If you would take a moment to indicate on the form below the urban problems of greatest concern to you, I would appreciate it. I also invite you to become a part of the EKISTICS network by subscribing to the journal.

I look forward to your reply.

Cordially,

Gwen Bell

Gwen Bell
Editor

To: Gwen Bell, Editor, EKISTICS
Page Farm Road
Lincoln, Mass. 01773 USA

From:

I consider the most pressing problem confronting human settlements is: ____________________

____

Please enter my subscription to EKISTICS:

____ One year (12 issues) $24.00

____ Two years(24 issues) $44.00

1 P

~~We must point out~~

We believe that

~~We believe we can guarantee~~ ~~that teh most likely eye evolution~~ technology ~~forces~~ will force the evolution

of ~~a most~~ computing structures ~~will~~ to be along ~~the~~ lines of ~~some form of~~ ~~both~~ all three

~~evolution:~~ — for high performance, ~~as~~ incremental performance and availability; multiprocessor computers; ~~such as C.mmp or less tightly coupled~~ Cm* — for ~~[illegible]~~ more ad hoc ~~specialized~~ structures to handle specialized processing (eg. front end, file, signal processing); and C.vmp for high availability, based on increased maintainence costs.

~~computers such as C.mmp based soleley on the evolving directions of~~

~~technology. Althou This pat is clear. The~~ P The technology force forcess ~~Forces and therefore~~ argument is ~~simply~~ based on history, ~~and~~ near term techology and resulting price extrapolations:

~~the directionis clear! the technology which forms the low cost processo processors~~ ~~, currently MOS, is evolving at more rapid rate~~ than

~~to density and the s speed if is imporvisn at a substantially greater rate than the ECL gate e aarray technology from which large machiens are mf formed;~~

1. the ~~tech~~ MOS technology ~~which drives~~ is increasing in both speed and density faster than the technology ~~from~~ (eg. ECL) from which high performance machines is formed;

2. the price per chip of the MOS one chip processors decreases ~~is coming down~~ at a substntially greater rate ~~due~~ than, ~~to~~ low volume special designs due to ~~it~~ the much higher volumes; and very high design costs (for both designs).

3. ~~and~~ the resulting performance (in operations) per chip and cost per operation per chip ~~is substantially~~ are rapidly diverging from ~~the~~ the ~~large~~ high performance semiconductor technology ~~form~~ which conventional high performance machines are formed.

(in terms of produce a better price and at a price/performance) computers

These factors ~~cost design costs for large machiens~~ is diverging ~~price and~~ rate compared with large scale computers. Furthermore, large scale applications have been slow to form since ~~there is more~~ problem complexity increases more rapidly than program size

~~(relatively s sloww growois market) high compared with the design cost per element of at the high volume products. This will c tend to create a small number of micro standard mciroprocessors (eg. thereer are essentially only 2 1 doinant type with which covers 1 type supplies about 50% of the market and 3 types supply 95% of the markt. AI a time when there is sufficient technolsoy to have a c sna adequate in terms of memory address space processor on a chipe, a standard witll created by which all all other other computer will be constructed.~~

at the low cost market end

4. Standards (in the semiconductor industry) for high volume products tend to form more quickly. For example, in the 8-bit microcomputer market, 1-type supplies about 50% of the market and 3-types supply over 90% of the market

For all intents and purposes, the processor cost at the low end is 0 relative to all other costs of the system!

Therefore, ~~all~~ most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at 0 cost from

5. A ~~new~~ 16-bit processor (-on-a-chip) ~~is eminent~~ with an acceptable (for the performance) address space ~~that~~ and ~~r~~ appropriate data-types is eminent.

a processor-on-a-chip based processors, large scale computing will come from ~~this source!~~ arrays ~~(just as~~ of processors, just as we build arrays of 16K bit ICs to form memory.

C.mmp--A mulit-mini-processor

~~C.mmp came from several The mov~~ C.mmp was motivated by the need for

recognition / problems

more computing power to solve ~~a particular set of problmes~~ (Speech signal procesing)

understand the

~~to get more computing power~~ and to ~~as a machine to w study~~ multiprocessor softwares problem.

large

Until C.mmp, ~~there~~ only one tightly coupled multiprocessor had been built --

BSTJ issue

~~only one~~ large

~~had be for, and it was for a different problem~~ The ~~safeguard~~ Bell Laboratory's Safegard Computer (?)

introductory

The ~~first~~ section ~~de~~ describes the ~~tech~~ economic and technical factors ~~behind~~ influencing

feasibility and argues that it's important

multiprocessors ~~an together with their adn why it is a matter of timeliness~~

to embark on the research ~~that mmakes them more and more practival in this t time periosd.~~ because ~~they are ultimately~~ of the timeliness.

Various problems to be researched are given together with

~~A number of reesearch ares are given and~~ a discussion of particular aspects ~~of~~ the design.

~~some of the pra particularly worrisonme aspects of the design are presented in d3etail~~

~~(e.g.~~ For example, since ~~a multiprocessors the multiprocess~~ C.mmp is predicated

both

one common operating system there ~~are both~~ is ~~both~~ degredation due to memory

contention when all the processors use the same block of memory and ~~also,~~ since

common for parts of

there are ~~software~~ synchronizing locks ~~around~~ the software, ~~a there can~~

may occur when a processor must idle and wait to

~~be significand~~ degredation ~~as waiiting ofr the software to~~ enter common critical sections.

The machine's ~~strucutre is proesneted in deta ssome detail and and the~~

due to

~~p maximu~~ theoretical performance is computed ~~based on~~ memory processor interference based on

~~this is computed based on teh work of~~ Strecker's (1971) work.

In practice,

~~Here it is interesting to note that the assumption that~~ ~~Ultimately,~~ memory

~~did~~ was

~~the problem f of i~~ interence ~~id di ultiamtely become~~ a problem ~~because~~ resulting in porrer than expected performance

with low order address interleaving.

~~programs were the mmemory was not built on an interle interleaved (lo order address~~

~~was not built, hence there was some intereecnce more interference thatn the interence~~

This problem ~~did materialize and~~ had (it) be solved ~~compensated for~~ by moving program segments of th ~~movememnt of~~ ~~programs~~ within ~~diffenrnet~~ ~~to different parts~~ within memory.

~~part of physical memosr.~~ ~~One aspect hat that is counter-intuitive should~~ One

It should be noted that when thenumber of memory moduels and ~~the number of~~ ~~(and equal)~~ processors ~~are equal (for exampel) and~~ become very large, the perfformance (as measured by the number of accesses to the memory by the processors) approaches the ~~number of memor~~ memory bandwidth ~~(m/tcycle)~~ ~~times 1/e.~~ (m / memory-cycle-time) × 1/e. Thus, there is not a maximum limit ~~of the~~ on performance ~~when~~ with infinite processors provided they are all not contending to access the same memory.

although ~~there~~ there is a ~~an~~ ~~extensive~~ discussion ~~of the~~

~~Operat~~

outlining the design ~~oper~~ direction of the operating system, Hydra, ~~the reader is urged to~~ ~~st~~ ~~ott~~ later descriptions ~~of Hydra~~ should be read (list the ~~re~~ references?? here). Since ~~#~~ the 11's small address ~~got in the way of~~ impaired ~~prob~~ ~~providing~~ use, 11/40's with microprogrammed writeable control stores were used to ~~carry out the memory mapping~~ ~~fast changes~~ implement ~~of~~ ~~ca~~ operating systems calls involving changing the ~~&~~ ~~base~~ ~~ad~~ ~~base ad~~ segment base registers.

One of the most pleasant surprises ~~that came out of~~ of C.mmp was the ~~work on~~ Algol 68 implementation because ~~for the two~~ it enables parallel programs to be specified. The results (see page 00) in terms of performance are quite encouraging! ~~approaches~~

~~Note~~ There are three basic approaches to ~~a~~ effectively applying ~~using~~ multiprocessors:

~~1. Only~~

~~1. Only us app~~

~~1. conventional, as partitioned mu~~

~~1. timesh~~ some

1. having lots of independent work to do do through multiprogrammed, ~~to~~ timeshared, ~~& use~~ and ~~or~~ multiple independent computers;

2. ~~taking using a conventional language (eg. Fortran)~~ ~~and~~ having it the compiler for a conventional language (eg. Fortran) detect ~~all~~ a ~~actual~~ and compile statements ~~or sets of statements~~ that can be executed in parallel; ~~and the~~

3. introducing new ~~co~~ ~~con~~ primitives into the programming language (eg. Algol 68) so that algorithms for parallel execution are specified by the programmer.

~~Algol 68 takes this~~

← although C.mmp was only predicated on ~~use of~~ ~~a~~ $\boxed{\text{use of type 1}}$, the Algol 68 implementation on C.mmp makes type 3 use possible, ~~so far~~ ~~the~~ (see ?)

~~5 The results of H~~

I. Isolated AC Input

This module provides eight transformer-isolated 120-volt AC inputs, with the following features.

1. 8 inputs per module; 2 wires per input.
2. Isolation to 1500 volts DC or peak AC between inputs or from input to ground
3. Jumper strip provided to allow commoning of one side of all inputs. Strip mounts on screw terminals.
4. Frequency range: 47-63 Hz
5. Nominal input voltage for logic "1": 120 volts
6. Individual input indicators on AC side of isolation.
7. Nominal input loading: 10 mA
8. Response time (turn-on or turn-off): less than 1/2 cycle
9. Inputs protected against overvoltage transients by MOV's.

II. Isolated DC Input

This module provides sixteen bits of optically-isolated DC input. There is an optional interrupt capability.

1. 16 inputs per module; 2 wires per input.
2. Isolation to 1500 volts DC or peak AC between inputs or from input to ground.
3. Jumper strip provided to allow commoning of one side of all inputs. Strip mounts on screw terminals.
4. Response time (turn-on or turn-off): 2.5 msec nominal

NSF ~~and ARPA~~ sponsored

Multi-Microprocessors: An Overview and Working Example

~~which can be interconnctedec~~ to form

Cm* is ~~the name~~ a system of high level modules, constructed

~~from~~ of LSI-11 computers,

~~W~~hich can be interconnected to form ~~a~~ large ~~computeing sytem.~~ computer structures. ~~The~~ The Cm* ~~are an extension of~~ are an extension

work sponsored by NSF and ARPA

~~work came out~~ of ~~a~~ the research (Bell etc. 1973) on register transfer modules. ~~because~~ As LSI and VLSI enable ~~the~~ construction of

I, with the processor-on-a-chip, it is

~~in which wit became~~ apparent that ~~the~~ low level register ~~mod~~

transfer modules ~~were passe'~~ are for the construction of low volume computers. Everyone is using and contributing to the standard. ~~in that~~ LSI and VLSI technology, ~~use~~ ~~enabling the construction of~~

~~to build complete~~

~~were driving the construction of modules made up of computers.~~ ~~current~~ Although the

research is predicated on ~~large~~ structures employing

~~up to~~ a hundred ~~pro~~ or so processors, this ~~first~~

for

chapter ~~com~~ describes the ~~firs~~ culmination of the first

~~The~~

~~the~~

10 processor phase.

he proposal for the first computer moudules we was funded by NS thef

National Science Foundation and the proposed direction was set describedin

1973 (Bell et al 1973) sda;l;l;l;laaslsfljsldfljljljjkkjjklklklklklklklkllkkjkkjjjjj

~~The~~ A movtivation ~~for the work~~ of Cm* is ~~described~~ basedon ~~the diversins cost~~ a diseconomy of scale for large computer introductions ~~factors for large computers (i.e there is diseconomy of scale) over hte last ttw tttwo years~~ during 1975-1977. ~~The need for the~~ C computer modules (Cm*) verrsus a multirprocessors ~~strucutre~~ (C.mmp) and computer networks are described in terms of performance and problem suitability to ~~further~~ provide additional context for the research. ~~wheat e the type os problesms each migh be approprate forl.~~

~~Most o Mus Much of the paper is~~ The chpater gives a ~~complete~~ ~~straightforward~~ descrimiption of the modules structure, together with their associated and ~~the design problems and the problems~~ potential ~~research~~ ~~topics~~ problems (research). ~~problelsm associated iwth building such a structure~~

~~In a Since this p chpater is the final~~

This chpater representd s the culminationof the v first phaase of research project involving the construction of al alarge mut multiprocessor strucutre, and although only 10 processors are eval used, there is a very good evaluationof the strucutrue agains in terms of a number of different program types.

The final? most important part of the chapter ~~is cor gives the Cm*~~ the performance & evaluates Cm* for ~~50~~ five different problems.

C.vmp: The Architecture and Implementation of a Fault Tolerant Multiprocessor

C.vmp a ~~voting~~ triplicated, voting multiprocessor designed to ~~inv~~ understand the difficulty (or ease) of using ~~a~~ standard, off-the -shelf ~~Ls~~ LSI-11s to ~~obtain~~ provide greatly increased reliability. ~~Since~~ There is a ~~increased~~ ~~growind~~ concern for increased reliability because systems are ~~settins~~ ~~m~~ more complex, ~~and~~ are used for more critical appplications, ~~but~~ and ~~move mov moreover there is a hisher cost of~~ basic maintainence cost for all systems ~~that~~ are increasing. ~~might be improved.~~

~~A number of~~ Several design goals are initially predicated ~~as~~ and the work is carried out against these ~~design~~ goals. ~~Some~~ Two of the more interesting ~~desins~~ goals include using off the shelf hardware and software with no modifications to the components.

~~Here,~~ Because the designer and analyze ~~and beca because the a designer is als the also~~ carries out the work, ~~and writers the paper,~~ this ~~paper~~ chapter provideds a great deal of ~~a~~ insight into the high reliability design and design process -- especially its evaluation. ~~Since~~ ~~also~~ The systems ~~i~~ has ~~been~~ operated for several months, ~~there is agreat deal to be~~ and ~~the fact that~~ the first phase of work is ~~also~~ complete ~~is especially valuable.~~

~~a One interesting observation is po possible~~

The goal ~~a~~ of software and hardware transparency turned out ~~at~~ be more easy because of an idiosyncrasy of the ~~possible because we even with~~ floppy ~~id disks drives because~~ the controller. This controller of ~~for floppy disks~~ operates on a word at a time transfer from a one sector buffer after a sector is transferred — thus voting is carried out ~~at a~~ at a very low level (i.e. as bus transfers are made).

(It is unclear ~~h~~ how the system ~~wer~~ would have been designed without this type of ~~disk~~ controller, but at ~~the~~ a minimum, some form of software transparency goal would have ~~to be relinquished, or some form of~~ been violated together with a ~~sif~~ significant controller modifications.

by the is evaluated.

A ~~nub~~ number of models are ~~gar~~ given which ~~evaluated hter~~ design~~, together~~

Thees are based on various component reliabilitys ~~whch come from~~

Various component reliabilities are used and the~~se~~

~~read~~

~~is a great deal~~

reader should get a great deal of insight ~~as~~ into

~~to~~ the factors contributing to reliability. It ~~is especiall~~ should be

noted that a special voter hardware is ~~[illegible]~~ needed ~~to~~ in

~~ma~~

order to ~~ma~~ build a ~~practical, m~~ marketable C.vmp.

While the intent of C.vmp is not a product, ~~there there is~~ it

~~much~~ much of

does provide ~~much~~ the insight ~~into the factors~~ ~~about what~~ ~~such a product might~~

for such a product

~~Con~~ ~~for a&~~

Correct and make ~~1~~ 6 2 copies for Fri

Gordon

PDP-11 GLUE 12/15/77

edited 12/16/77

This part ~~is quite capable of standing~~ can stand alone as a book on the PDP-11 computer structure evolution, although it relies ~~heavily~~ on the conceptual framework of Chapter 1, and the ISPS language description given in appendix 00. The 11 has evolved quite differently than the other computers in this book, and as such provides an independent and interesting story. The factors that have created the machines are clearly market and technology based, generating a large number (ten) of different implementations over its relatively short seven year lifetime. ~~Thus, the fact that~~ Because there is an expanding range of machines ~~we move along in~~ versus time provides problems and insight in addition to the traditional mini (8 Family); the best cost/performance (18-bit), and the high performance less than $500K (the DECsystem 10). The 11 designs cover a range of a factor of 500 in system price ($500 to $250,000) and 500 in memory size (4 Kwords to 2 Mwords).

This part is divided into five sections consisting of:

1. Introduction.

   This first chapter (00) published when the 11 was announced introduces the architecture, gives ~~various~~ its goals~~, for it~~, and predicts how it might evolve. The family notion is quite strong, although not specific. ~~Alternatively,~~ Chapter 00, What Have We Learned From PDP-11, ~~may~~ might ~~preferably~~ be read next in order to get the broadest overview, and best immediate critique of ~~Chapter 00.~~ the 11 evolution

---

2. Conceptual Basis.

This section contains two papers, that combined with Chapter 00, form some of the conceptual basis ~~for basis~~ for the models. Strecker, Chapter 00, describes the cache memory mechanism ~~as a basis~~ for building high performance models (specifically the 11/70). Levy describes the intercommunication problem among physical components and how busses carry out this task.

3. Implementations.

Of the four chapters, three are on specific implementations (LSI-11, CMU-11 and 11/60) and the fourth (Chapter 00) is a study of ~~on~~ all models. This latter chapter provides details of the various implementations together with how the designs fit a conceptual model. ~~This is especially useful to understand the performance.~~

4. Evaluation. Chapter 00 evaluates the 11 as a machine for executing FORTRAN. What We Have Learned From PDP-11, Chapter 00 discusses all aspects of the evolution in terms of the introductory Chapter (00).

5. The multiprocessor research computers of Carnegie-Mellon University. Three multiprocessors: C.mmp (Chapter 00), Cm* (Chapter 00) and C.vmp (Chapter 00) give examples of a 16-processor multiprocessor, a set of modules based on LSI-11 and a triplicated, voting multiprocessor computer for high reliability.

A NEW ARCHITECTURE FOR MINICOMPUTERS--THE DEC PDP-11

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed to describe the first (11/20) implementation at a time when the whole architecture had not been worked out or even fully considered; and to show possible extensions.

The reader might note that the computer class definitions (given in 1970) of micro, mini and midi correspond quite closely to those of Chapter 1.

Although we comment on the disparity between the predicted and actual evolution in Chapter 00, some of the important reasons are:

1. The notion of designing with improved technology especially for a family was not understood then. Th~~e~~is understanding ~~for one of us (Bell)~~, was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus bandwidth proved unacceptable for all communications at the very high and low end designs. Whereas, this chapter posits a multiprocessor, and multiple Unibusses, this precise structure did not evolve, as a standard. The bandwidth has subsequently been shown to be adequate for all but the largest configurations when a cache is attached to the

processor as in the 11/60 (see Chapter 00). Note the effect of a 90% cache hit rate is to reduce the number of access to primary memory via the Unibus by a factor of ten!

3. The memory address space was too small.

4. The particular data-type extensions weren't predicted. While floating point data was discussed, the character string and decimal operations weren't described. These data types evolved in response to market need (and COBOL) which didn't exist for DEC at the time the machine was designed.

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete (memory management and floating point) ISPS description that was used when the 11 was evaluated as a Computer Family Architecture (CFA) standard by the U. S. Defense Department.

CACHE MEMORIES FOR PDP-11 FAMILY COMPUTERS

Chapter 00 by Strecker is included for four reasons: it is a clear exposition of the cache memory structure and its design parameters; the cache is the basis of the fast PDP-8 (Chapter 00), the 11/60 (Chapter 00) the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is "good engineering"; and finally the paper is "well-written"--in fact, it received the award for the Best Paper at the

Third Computer Architecture Conference at which it was presented. Thus, since a relevant paper that provides a clear exposition, while illustrating good engineering and being well-written, is so rare we also publish it simply to serve as an encouragement and an example for those who should ~~expose,~~ understand and describe their work.

The design process is implicit in the way the work is carried out to determine the structure parameters. It should be noted that it was easy to collect data statistics about the program's behavior since the trace bit, T, permits the 11 to interpret itself on a one at a time basis. The relevant ~~parameters are varied and~~ sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge (Chapter 00) ~~also~~ uses ~~the same methodology and~~ Strecker's program traces and methodology. One of the important parameters to understand is the time between changes of context. To the best of our knowledge this study is unique, ~~and imp~~ this understanding is important because all real time and multiprogrammed systems have many context switches. The PDP-8 cache design (Chapter 00) shows the effect of segmenting the cache for instructions and data. The PDP-8 chapter discusses the performance for a particular design, and the performance numbers are in terms of a speed-up factor. Strecker gives the performance evaluation in terms of cache miss ratios. The performance measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

p	= total no. of memory accesses by the processor, Pc
m	= no. of memory accesses that are missed by the cache and how to be referred to Mp

$t.c$	= cycle time of cache memory, Mc
$t.p$	= cycle time of primary memory, Mp
R	= $t.p/t.c$ (ratio of memory speeds), where R is typically 3 to 10

the relative execution speeds are:

$t(\text{no cache}) = pR$

$t(\text{to cache}) = p + mR$

speedup = $pR/(p + mR) = R/(1 + (m/p) R)$ = a = miss ratio = m/p

therefore

speedup = $R/(1 + aR) = 1/(a + 1/R)$

note that if $a = 0$ (100% hit), the speedup is R; while

if $a = 1$ (100% miss), the speedup is $R/(1 + R)$, i.e.,

the speedup is less than 1 (i.e. time to reference both memories)

IMPLEMENTATIONS

A MINICOMPUTER-COMPATIBLE MICROCOMPUTER SYSTEM: THE DEC LSI-11

This first chapter of the implementation section was written from the knowledgeable user viewpoint. Although the paper is a descriptive narrative about ~~of~~ the design of ~~from~~ the chips, boards and backplane levels, it lacks insight that the designers at Western Digital or DEC (Lloyd Dickman, Duane Dickhut, Rich Olsen, or Mike Titelbaum) might have provided. An account of the chip-level design is available (Soha and Pohlman, 1974). The design was done at Western Digital by Roberts, Soha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with them for use in the interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP; and the particular PMS module level components as they are integrated into a backplane to form a hardware system. Sebern points out the microprogramming tradeoff that took place between the chip and module levels to carry out normally hardware functions: the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. For example, the alternative, bounded design that is typical of one board microcomputers was considered, though not described. A lower cost one board system, like the VT78, (page 00) has evolved and ~~will eventually be marketed. This structure~~ is shown in Fig. KXT11. The initial module level design was predicated on a quad-sized form factor and plugged into a conventional backplane. The modules are shown on page 00. Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was used for providing interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus (requires a double sized module). This permitted multiple board, complex options (e.g., a disk controller) to be packaged as a single option with no interconnection between the boards except the second half of the quad board. As DEC began to build special ICs to interface to the bus, the option sizes began to decrease to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options.

Table LSI-11 Options

[Teicher to supply]

The effect of a lower cost LSI-11 system is to provide additional applications as we discuss in Chapter 1, page 00.

USING LSI PROCESSOR BIT-SLICES TO BUILD A PDP-11--A CASE STUDY IN MICROCOMPUTER DESIGN

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, herein called a Microcomputer (for Microprogrammed Processor), is used to interpret a PDP-11 ISP. The purpose of the design was to test the assertion that the bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (11 processor) to fit the building blocks (the Intel 3000 processor). As such, the authors observed the awkwardness that ultimately resulted in lower (than desired) performance. In retrospect, the Intel 3000 has not become the standard bit-slice architecture that the AMD 2900 series has, but perhaps suffers from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was to evaluate the computer aided design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the key components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction. The authors claim that 95% of the errors were detected by the simulation--and all errors were ultimately detectable once the simulation data or machine description was changed.

DESIGN DECISIONS FOR THE PDP-11/60 MID-RANGE MINICOMPUTER

Unlike the reports from an architect's or reporter's viewpoint this chapter is a direct account of the design from the close proximity of the project. Because it is a mid-range machine, the 11/60 is a difficult design for the reasons of the designer characteristics, Chapter 00, page 00. The design is neither the lowest cost nor performs the highest of the 11s, but has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60: the cache to reduce Unibus bandwidth; trace driven design of floating-point arithmetic; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus had previously thought to be inadequate for high performance, the cache is used to unload the i/o traffic and provide high performance without the attendant cost*. The work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since additional traffic is generated by fetching multiple words which would tie up the Unibus with additional traffic.

The use of trace data to design the floating point arithmetic is described together with the resulting design. Note that the 11/60 performs roughly at 11/70 speeds with lower cost. The implementation of the two can be compared in the following table.

Table - Implementation of 11/60 and 11/70.

	11/60	11/70
Base Pc	x	x

*Using Amdahl's constants, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus.

Floating point	x	x
Cache	x	x
Memory management	0	x
	______	______
Total	x	x

Microprogramming is used to provide both increased user-level capability and increased reliability, availability and maintainability. The large writeable control store option is described together with its use for data storage and various applications. This option has been recently used for emulating the PDP-8 with OS/8 operating systems.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 1, page 00). Other semiconductor technology improvements are described together with how they effect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

This chapter presents a most comprehensive comparison of the eight PDP-11 processor implementations. The work was carried out to investigate various design styles for a given problem--the interpretation of the PDP-11 ISP.

Aside from any conclusions the tables give more insight into processor

*Ability to interconnect a number of subsystems together through a wired-or connection.

implementations than is available from any single source we know. The usefulness of the data also comes from having an outside observer examine the machines and provide insight.

The tables include:

1. ~~A~~ set of instruction use frequencies, by Strecker, for an operating system. ~~The~~ reader should note the frequencies do not reflect general purpose use, e.g., there are no arithmetic and floating point instructions;

2. Implementation cost (modules, ICs, control store widths) and performance (micro- and macro-instruction times) for each model; and

3. ~~A~~ canonical data path ~~is posited~~ for all 11 implementations, and each processor is presented and compared with it.

With this background data, a "top-down" model is built which explains the performance (macro-instruction time) of the various implementations in terms of the micro-instruction execution, and primary memory cycle time.

Since these two parameters don't fully explain (model) performance, a set of bottom-up factors must be introduced. These factors include various design techniques and the degree of processor overlap. We believe the fact that the problem is constrained should provide useful insight to both computer and general digital systems design.

~~*Ability to interconnect a number of subsystems together through a wired-or connection.~~

EVALUATION

TURNING COUSINS INTO SISTERS: THE ROLE OF SOFTWARE IN SMOOTHING HARDWARE DIFFERENCES

Since FORTRAN is quite possibly the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first FORTRAN compiler and object (run) time system are described together with the evolutionary extensions to improve performance. The FORTRAN IV+ compiler is only briefly discussed since its improvements are largely a matter of compiler optimization technology and are less relevant to the 11 architecture.

The chapter title overstates the problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by providing what amounts to five separate object (run) time systems and a single compiler. This transparency is provided quite easily using a concept called threaded code. This concept appears to be a very simple interpreter for the PDP-11--and might not be called an interpreter by many. With threaded code, one 1-word instruction requiring two memory cycle times is executed each time the next high level operation code is to be interpreted, otherwise the processor is carrying out the desired op code. When a simple integer expression like A = A + 1, which occupies 2 memory words and requires 3 memory cycles to execute, is transformed into a threaded code version the program still only occupies 2 words, but instead requires 5 memory cycles to execute (nearly a factor of 2). For more complex operations requiring longer execution times, like floating point arithmetic, the overhead turns out to be quite low and the space utilization is quite good. Although ~~this is~~ not used, it is possible to move ~~easily~~ efficiently between threaded and directly executed code. Jim Bell discovered this technique and it has been used extensively for other compilers because the time and space overhead is so low. The ability to carry out the interpretation so elegantly was not part of the original design, but rather turned out to be possible based on the generality in the 11's indexing modes.

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. By recognizing the special high use frequency cases (e.g., A = 0, A = A + 1, etc.) and by having better conventions for three-address operations to and from the stack, speedups of 1.3 and 2.0 for floating point and integers, respectively, were obtained when a more complex machine was constructed for the second version.

Since this paper is really on the construction of an extension to the 11 to execute FORTRAN, it is interesting to compare it with the FORTRAN IV+ machine which uses the floating point processor (11/45, 11/55, 11/60, 11/70). If the FORTRAN machine described in the paper is microprogrammed and made to operate at FPP speeds, the two machines turn out to operate at roughly the same speed and programs occupy roughly the same program space.

WHAT HAVE WE LEARNED FROM THE PDP-11?

This chapter is a substantially revised version* of a paper written for the CMU Computer Science 10th Anniversary, September 1975. This paper was written to critique the original expository paper on the PDP-11 (Chapter 00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined.

The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially critical, The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further aid in understanding the design process.

The main section consists of: evaluating the Unibus, examining the cost-performance evolution, the ISP and discussing the organizational issues behind why multiprocessors haven't evolved. As a comparison, Chapter 00 on C.mmp describes the technical problems associated with multiprocessors.

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standrad of interconnection.

*The paper is 50% longer. The introductary overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.

We try to provide a set of evolutionary cost-performance metrics so the 11 can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique problem the 11 has of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a more useful evaluation of the architecture for FORTRAN execution. A complete section is given on the addressing extension beyond the 11/45 and 11/70 extensions which required a major perturbation in the form of the VAX/11 extensions.

The final section to the research community describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs. (This part has been expanded too.)

MULTIPROCESSOR

THE RESEARCH COMPUTERS OF CARNEGIE-MELLON UNIVERSITY

These three multiprocessor computers which use the 11 as a basis were built at Carnegie-Mellon University to carry out various computer structures, operating system and application program computer engineering and computer science research.

The first computer, C.mmp, is a 16 processor (model 11/40's) system with 1 million words of shared primary memory. It was built to investigate the programming (and resulting performance) questions associated with having a large number of processors.

The second computer, Cm*, is also discussed in the Modules Part as the modules that form Cm* are LSI-11 computers. Cm* evolution was based on the premise that ultimately, the smallest modular unit (i.e., ultimately all ICs) that would be used to build digital systems is the computer. With this premise, it is important to understand how to interconnect ~~them~~ computers physically, how to assign parts of the problem to the various computers and how to program the~~m~~, complete structure.

The C.vmp, for voting multiprocessor, was designed to investigate how a production microcomputer, the LSI-11, can be used to build a triplicating, voting high availability computer.

The goals of the first two are performance while the third uses multiple computers for redundancy and reliability. To this end, Fig. Perf shows the effect of using multiprocessors to execute various algorithms (More to come here!).

We believe that technology will force the evolution of computing structures to be along all three lines of multiprocessor computers: C.mmp~~--~~, for high performance, incremental performance and availability; less tightly coupled computers, like Cm*~~--~~ for more ad hoc structures to handle specialized processing (e.g., front end, file, signal processing); and C.vmp--for high availability based on increased maintenance costs.

The technology force argument is based on history, near term technology and resulting price extrapolations~~.~~ ~~It~~ and follows:

1. The MOS technology is increasing in both speed and density faster than the technology (e.g., ECL) from which high performance machines is formed;

2. The price per chip of the MOS one chip processors decreases at a substantially greater rate than low volume special designs due to the much higher volumes and very high design costs (for both designs);

3. For all intents and purposes, the processor cost of the low end is 0 relative to all other costs of the system! The resulting performance (in operations) per chip and cost per operation per chip are rapidly diverging from the high performance semiconductor technology from which conventional high performance machines are formed.

4. Standards (in the semiconductor industry) for high volume products tend to form more quickly. For example, in the ~~1~~8-bit microcomputer market, 1-type~~s~~ supplies about 50% of the market and 3-types supply over 90% of the market.

5. A 16-bit processor (-on-a-chip) with an acceptable (for the performance) address space and appropriate data-types is eminent. Such a commodity will form the basis for nearly all future designs.

These factors produc~~t~~e better (in terms of price and price/performance) computers at the low cost market end at a diverging rate compared with large scale computers. Furthermore, large scale applications have been slow to form since problems complexity increases more rapidly than program size. Therefore, most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at 0 cost from a processor-on-a-chip based processors, large scale computing will come from arrays of processors, just as we build arrays of 16K bit ICs to form memory.

C.mmp--A Multi-Mini-Processor

C.mmp was motivated by the need for more computing power to solve speech recognition/signal processing problems and to understand the multiprocessor software problem. Until C.mmp, only one large, tightly coupled multiprocessor had been built--the Bell Laboratory's Safeguard Computer (BSTJ issue?).

The introductory section describes the economic and technical factors influencing multiprocessor feasibility and argues that it's important to embark on the research because of the timeliness. Various problems to be researched are given together with a discussion of particular design aspects. For example, since C.mmp is predicated on a common operating system there is degredation both due to memory contention when all the processors use the same memory block and since there are common synchronizing locks for parts of the software, degredation may occur when a

processor must idle and wait to enter common critical sections. The machine's theoretical performance due to memory processor interference is computed based on Strecker's (1971) work. In practice, memory interence was a problem resulting in poorer than expected performance because the memory was not built with low order address interleaving. This problem had to be solved by moving program segments of the memory. It should be noted that when the number of memory modules and processors become very large, the performance (as measured by the number of accesses to the memory by the 1
processors) approaches the memory bandwidth (m/memory-cycle-time) x $1/e$. Thus, there is not a maximum limit on performance with infinite processors provided they are all not contending to access the same memory.

Although there is a discussion outlining the design direction of the operating system, Hydra, later descriptions should be read (list the references?? here). Since the 11's small address impaired use, 11/40s with microprogrammed writeable control stores were used to implement operating systems calls involving changing the segment base registers.

One of the most pleasant surprises of C.mmp was the ALGOL 68 implementation because it enables parallel programs to be specified. The result (see page 00) in terms of performance are quite encouraging! There are three basic approaches to effectively applying multiprocesors:

1. having lots of independent work to do through multiprogrammed, timeshared and multiple independent computers;

2. having the compiler for a conventional language (e.g., FORTRAN) detect and compile statements that can be executed in parallel;

3. introducing new primitives into the programming language (e.g., ALGOL 68) so that algorithms for parallel execution are specified by the programmer.

Although C.mmp was only predicated on type 1 use, the ALGOL 68 implementation on C.mmp ~~explores~~ ~~makes type 3 use possible (see ?)~~. permits parallel program (type 3) to be written. The results of some of this programming is shown in Fig. ?.

Multi-Microprocessors: An Overview and Working Example

Cm* is a system of high level modules, constructed of LSI-11 computers, which can be interconnected to form large computer structures. The Cm* work sponsored by NSF and ARPA are an extension of NSF sponsored research (Bell, etc. 1973) on register transfer modules. As LSI and ULSI enable construction of the processor-on-a-chip, it is apparent that low level register transfer modules are passe' for the construction of all but low volume, specialized computers. A complete industry will ultimately design, contribute to and use a ~~Everyone is using and contributing to the~~ standard computer. Although the research is predicated on structures employing a hundred or so processors, this chapter describes the culmination of the first ten processor phase.

A motivation of Cm* is based on a diseconomy of scale for large computer introductions during 1975-1977 (see page 00). Computer modules (Cm*), multiprocessors (C.mmp) and computer networks are described in terms of performance and problem suitability to provide additional context for the research. The chapter gives a description of the modules structure, together with their associated and potential problems (research).

The final, most important part of the chapter evaluates the performance of Cm* for five different problems.

C.vmp: The Architecture and Implementation of a Fault Tolerant Multiprocessor

C.vmp is a triplicated, voting multiprocessor designed to understand the difficulty (or ease) of using standard, off-the-shelf LSI-11s to provide greatly increased reliability. There is concern for increased reliability because systems are more complex, are used for more critical applications, and basic maintenance cost for all systems are increasing. Because the designers carry out and analyze the work, this chapter provides a great deal of insight into high reliability designs and design process--especially its evaluation. The system has operated for several months and the first phase of work is complete.

Several design goals are initially predicated and the work is carried out against the goals. Two of the more interesting goals include using off-the-shelf hardware and software with no modifications to the components.

The goal of software and hardware transparency turned out to be easier than expected because of an idiosyncrasy of the floppy controller. This controller operates on a word-at-a-time transfer from a one-sector buffer after a sector is transferred--thus voting is carried out at a very low level (i.e., as bus transfers are made). It is unclear how the system would have been designed without this type of controller, but as a minimum, some form of software transparency goal would have been violated together with a significant controller modification.

A number of models are given by which the design is evaluated. Various component reliabilities are used and the reader should get a great deal of insight into the factors contributing to reliability. It should be noted that a special hardware voter is needed in order to build a marketable C.vmp. While the intent of C.vmp is not a product, it does provide much of the insight for such a product.

~~#~~ created 1/21/77 (from G.B.)

11 glue

A New ~~#~~ Architecture for Minicomputers-The DEC PDP-11

original PDP-11 description

It is somewhat ~~quite~~ anticlamactic to discuss this ~~paper~~ here because ~~the~~ Chapter 00 explicitly discusses ~~the subject what~~ what We Have Learned From ~~About~~ the PDP-11. ~~Nevert A sigini~~

~~The reader should might note that the defi various definitions of micro, computer classes into~~ definitions of micro, mini and midi ~~and thees se correspond quite closely to the claseses given in~~ those of Chpater 1 ~~, ev despite~~ despite age (1969) ~~the fact the definitions were made in 1969 and we have not tried to makee the two definitosn coincide.~~

The purpose of the chapter ~~paper~~ was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed; to ~~d specifically~~ describe the first (11/20) implementation ~~of the 11/20 (in~~ at a time when the whole architecture had not been worked out or even fully considered; and to ~~gently~~ show ~~what~~ extensions ~~were~~ possible ~~and might like ly follow~~ (even though ~~the a extensions were not even i at the proposal stage).~~

Although we comment ~~in chpa~~ on the ~~de~~ disparity between the predicted and actual evoution ~~of the 11~~ in Chapter 4 00, ~~it is worth of note simply noting that there wer why the two herer why the two are different. Simply,~~ some of the important reasons are:

1. ~~teh~~ the notion of designing with imporved technology ~~(and famil~~ especially for a family ~~in view of the family range,~~ was not understood ~~(t The evolutionary notion is given posited in a paper It is~~ then. The understanding for one of us (Bell), was put forth in a paper in 1972 (Bell, Chen, Rege, 1972)

2. The Unibus bandwidth proved unacceptable for ~~gave out as a single bu~~ for all communications at the high and very low end designs. ~~I~~ Whereas, ~~the chapter~~ this ~~caper~~ art posits a multiprocesor, ~~the~~

and ~~no especially~~ multiple Unibusses, this precise structure ~~form~~ did not ~~materialisze.~~ evolve. The ~~I~~ bandwidth

~~The cache strucuter did a develope and hendee the UNibus is can now still be shown to be adequate for even substantially higher performance machines when a cache is employed~~

has subsequently shown to be adequate ~~in terms of i/o~~ for all but the largest configurations when a cache is attached to the processor as in the 11/60 (see Chapter 00). ~~Flas~~

Note the effect of a 90% cache hit rate is to reduce the ~~ta~~ number of access to Mp via the unibus by a factor of ten!

3. The memory address space was too small.

4. The particular data-type extensions weren't predicted. While ~~we~~ floating point data was discussed, the character string and decimal operations weren't described. These ~~later~~ data ~~dt~~types eveolved in response to market need (and Cobol) which ~~DEC wasn't in at the time~~ didn't exist at the time the machine was designed.

~~One final~~ We have made a major change in the ~~appendicx, to (~~ ~~the ea~~ chapter by removing the original ISP description and replacing it ~~wasnt~~ with ~~and~~ a complete (memory managmt and floating point) ~~ISPS des ac the~~ ~~and~~ ISPS description that was ~~wed~~ used ~~in~~ when the 11 was evaluated ~~by the~~ as a Computer Family Architecture (CFA) standard by the U.S. Defense Department ~~bye group for standardization in the CFA~~. ~~This descripton also includes the floating point and memory management extensions.~~

~~It show~~

Cache Memories For PDP-11 Family Computers

Chapter 00 by

three exposition

Strecker'~~s chapter~~ is ~~given for~~ included ~~t~~ for ~~t two~~ reasons: it is a clear ~~description~~ of the cache memory

and its design parameters

~~p about the cleares t explanation of the cache meory that we would~~ ex structure ~~which is used in~~

~~beliieve exists;~~ ~~it describes the parameters from a designer computer designers~~

~~standpoint~~

since the cache is or

~~As such the cache is~~ the basis of ~~a number of products systems~~

~~described here~~ the ~~PS and N~~ fast PDP-8 (chapter 00), the 11/60 (chapter 00)

and the 11/70 (page 00~~)~~), and the KL10 (chapter 00),; the ~~methodu~~ design

methodology ~~of the ddesign for carrying out the~~ is well done, --

~~design i.e.~~ it is "good engineering ~~practice~~,"; and finally the paper

is "well-written"--in fact, it received the awarrd for the Best Paper

at the Third Computer Architecture Conference at which it was presented.

~~In this latter regard, it is rare~~

~~In all these regards it is rare to find a re~~

~~A combination of these three attributes: relevancy, g a good engn and example~~

~~of good~~ ~~Thus~~ ~~Since~~ provides a clear exposition, while

Thus since relevant while

A paper that ~~combines relevency, with~~ illustrating good enseeineering and

being and

~~is~~ well-written is ~~a~~ so really rare. ~~Therefore we also publish it~~ ~~that~~ we ~~would~~ also publish it simply

an and ~~a~~ example

to serve as ~~a~~ encouragement for thos~~e~~ e who should ~~r wirte (explain)~~ ~~explain~~,

~~and~~ ~~d~~ expose, underst~~r~~and and describe th~~i~~eir work~~s~~.

~~The explanastion is quite clear (and n together with~~ . The subtelties

of ~~a~~ design for working ~~wi w~~ with an operating system environment

It should be noted that it was easy to collect data ~~on the~~ statistics about the program's behavior since the trace bit, T, permits the 11 to interpret itself on a one at a time basis.

¶ The design process is implicit in the way the work is carried out to determine

structure

varied

the parameters. The relevant parameters are ~~tried~~, and sensitiivty plots

In the Mudge

(runs) are made to determine wht effect of each parameter on the design. ¶ ~~The~~ 11/60, also used the same ~~also~~ methodology (Chapter 00).

One of the important parameters, the time between ~~interrupts (or~~ changes

and the best of

of context is especially important, To our knowledge, ~~this is the only~~ is both unique and

necessary

~~work that deals wi specifically with htis issue.~~ ~~There is a The~~ ~~relevant~~ for ~~multi~~

~~The paper on the PDP-8 cache should be reused too, because together~~ any real time or

design Multiprogrammed system.

~~they have nearly all the design parameters. In the case of~~ The PDP-8, ~~the cac~~ cache (chapter 00)

shows the

effect of ~~s~~ segmenting the cache for instructions and data ~~are is examined.~~

¶

~~I in the originals w~~

~~Chapters)) and this chapter give different measures for performance~~

~~While~~ The PDP-8 chapter discusses ~~teh~~ the performance for a particular~~x~~ design, and the performance numbers~~x~~ are in ~~absolutes.~~ ~~Since this work is~~ ~~Strecker~~ Strecher ~~more general~~ gives ~~is~~ the performance evaluation ~~is~~ in terms of ~~the msis rac ratio.~~ cache miss ratios. The two ~~indic~~ performance measures are related (Lee, 1969) in the ~~may be coma compareed ed by the observing the followins (L22~~ following way (assuming an infinitely fast processor): terms of a speed-up factor.

Using Fig. Cachespeed

~~let~~
p = total no. of memory accesses by the processor, P_c
m = no. of memory accesses that are missed by ~~(i.e. not in the~~ the cache and have to be refered to M_p

~~t.fast = cycle time~~
$t.c$ = cycle time of cache ~~~~M_p~~~~ ~~(M_c)~~ memory, M_c
$t.p$ = cycle time of ~~primary M_p~~ primary memory, M_p
R = ~~M_p / M_c~~ ~~M~~ $t.p / t.c$ (ratio of memory speeds), where R is typically 3 to 10

The relative execution speeds are:

$$t\,(\text{no cache}) = pR$$
$$t\,(\text{to cache}) = p + mR$$

$$\text{Speedup} = pR / (p + mR) = R / (1 + (m/p)\,R) =$$

$$a = \text{miss ratio} = m/p$$

Therefore

$$\text{Speedup} = R / (1 + aR) = 1 / (a + 1/R)$$

note that if $a = 0$ (100% hit), the speedup is R; while if $a = 1$ (100% miss), the speedup is $R/(1+R)$, i.e. the ~~spe~~ speed is less than 1.

A Minicomputer-Compatible Microcomputer System: The ~~T~~ DEC LSI-11

~~The LSI-11~~ This chapter ~~is the~~ first of ~~an~~ the impl~~l~~ementation section on ~~the various implementtations.~~ ~~Altho~~ It was written from the ~~a user view~~ knowledgemable user viewpoint. ~~as S Sev Sebern ws was not a Bii Bill Roberts of then of Western Digits~~ ~~The paper has none~~ Although the paper is a descriptive narr~~r~~ative of the design from ~~a~~ the chips, boards and backplane levels, it ~~has none of the insight~~ that lacks ~~the insis design~~ some insight that ~~w one might get from~~ at Western Digital or DEC one of the designers ~~(Roberts, Zoha, or Dickman)~~ ~~(eg Pohlman, Teitelbaum~~ (Dickman, Olsen, or Mike Teitelbaum) Lloyd Rich An account of the chip-level ~~des~~ design is ~~given in by roberts~~ available by Roberts, ~~Po~~ Zoha, and Pohlman (Soha ~~d~~ and Pohlman, 1974) (MJ please ask Dickman or Teicher for a cop~~y~~y of this paper or find out if Roberts wrote a paper for~~m~~ the 74 Nerem Record Part 2 in Oct. 74) ~~Since it~~ The design was done at ~~W~~ Western Digital Corporation, ~~it was~~ as a relatively general purpose microprogram computer ~~design~~ that could be used to emulate ~~any~~ many computers. When DEC ~~4DEC~~ started working with them for use ~~a~~ in the interpretation of the 11 ISP, the design ~~began to take~~ took on more of the 11 strucutre ~~of an 11.~~

Two design levels are described in the paper: the 3 chip ~~chips an micro~~ microprogrammed computer as it is used to interpret ~~i~~ the 11 ISP; and the module ~~board~~ level ~~he set~~ particular PMS components as they are ~~packased on a board~~ and integrated into a backp~~l~~lane to form ~~the basis of~~ a hardware system.

Sebern points out an ~~One very~~ interesting tradeoff, ~~is~~ between the chip and module levels ~~observed~~ can be observed ~~as~~ in the ~~case~~ use of microprogramming to carry out ~~[illegible]~~ functions that are normally done with hardware: the ~~clock~~ time clock, the console, ~~and~~ refreshing dynamic ~~ram~~ random access ~~semiconductor~~ MOS memory; and power fail control.

~~It is interesting to note that the se the board subtelties~~

The subt~~l~~leties and uniqueness of the ~~board~~ module structure are not ~~designed d is not~~ described. The initial ~~module~~ module level design~~s~~ ~~were~~ was predicated on a quad-sized form factor which and plugging into a conventional backplane. The alternative, more bounded designs that are typical of other microcomputers is not described. However it should be noted that in order to get more reduced cost, such a system, ~~seen~~ like the VT78, has evolved and will eventually be marketed.

modules are shown ~~in~~ on page 00. ~~It i~~ ~~The~~ Since there was not

special ICs beyond the 3 chip~~s~~ ~~from which the~~ processor, ~~was formed,~~ options

tended to be relatively large and often occupied a full quad module. For options

that were greater than a quad size, ~~a~~ an ingenious packaging scheme was

used for providing interconnection points on the extra half of the moudle

which was not used as a the LSI-11 Bus, ~~(also known as a Q Bus.~~ This a ~~permemtted~~ permitted

multiple ~~p~~ board, complex options (e.g. a disk controller) to be packa~~s~~ged as a single

option with no interconneciton between the boards except the second half of the board. As the ~~level of~~

~~int IC we~~ DEC began to build special ~~options~~ ICs ~~ot~~ to interface to the ~~Q Bus,~~ bus, the

~~average sixwe of~~ ~~the~~ option sizes began ~~p~~ to decrease to occupy a double module ~~so that a only a sisle double~~

~~board is needed fo the interface.~~ This sytem ~~began to be~~ is now known as the

LSI-11/~~f~~2. ~~A set~~ A backplane system with modules is shown in Fig. LSI-11/2.

The interconnection ~~forming~~ forming the ~~Q Bus~~ LSI-11 Bus ~~is~~ only ~~a~~ requires 1/2 of the pins, ~~or a Do~~ i.e. a double size.

This structure is shown in Fig. KXT11.

The ~~options are~~ options available for the two systems ~~are! shown in Table~~ have evolved as shown in Table LSI-11 Options

Table LSI-11 Options

Teicher to supply

~~Note~~

The effect of lower cost LSI-11 module and backplane ~~avail~~ availabality is to provide additional applications ~~as~~ as

~~because it is available at a new, lower level~~

we discuss in Chapter 1, page 00.

Using LSI Processor Bit-Slices to Build a PDP-11--A Case Study in Microcomputer Design

by the designers of CMU-11

This paper appears both in the moudle ~~section and~~ part and in this part on PDP-11's. implementations

~~Two~~

~~Here,~~ the Intel 3000 bit ~~clice~~ bit slice, ~~an~~ IC is used, herein called a Microcomputer

to interpret a PDP-11 ISP. (for Microprogrammed Proceessor)

The purpose of the design was to test the assertion ~~of semiconductor~~

~~manufacturers to~~ that the ~~microp~~ bit-slice based, arithmetic unit / with

~~with~~ register memory and microprogrammed controls would simplify

the design of and construction of processors. ~~It is worthy to note~~

~~that~~ In retrospect, ~~this~~ the Intel 3000 has not become ~~a~~ the standard *

bit-slice architecture, but ~~it~~ perhpas suffers from being ~~first~~ one of the

earliest, ~~and as such the current de facto standard~~

~~architecture, A American Micro by AMD has become hte standard.~~

~~Since the three authorst are also the proinciple desingers, the~~

~~there is a great deal of insight into the design.~~ ~~It shoul~~ ~~There are~~

~~a~~ Detailed comparisons ~~about~~ including a breakdown of the various parts of the processor design and are given.

~~these are~~ compared with the ~~11~~ LSI-11, and ~~the~~ 11/40 designs in

~~ther teer~~ terms of performance and cost (IC count, number of control bits ~~and~~ in the

the microprogrammed controller) ~~and performance~~

A key part of the ~~initial design goal~~ investigation ~~of the~~

~~r project originaialnlly~~ was ~~to~~ to evaluate the computer aided design

tools ~~available a to assist in the design. Here,~~ The Standford University

Drawing Systesm (SUDS) and the SAGE Logic simulator were the key ~~part~~ components.

~~The purpose of the design was to evaluate whetrher~~ was predicated on being able to SAGE ~~could~~ detect all the

design ~~flasws prior to~~ and timing flawss prior to construction ; ~~and in fact,~~

The authors claim that 95% of the errors were detected by the simulation — and all errors were ultimately detectable once the simulation data or ~~sim~~ machine description was changed.

~~* The AMD 2900 ser.~~

The 11 was selected as a target problem in order to avoid the temptation of changing the problem ("processor") to fit the building blocks (the Intel 3000 processor). As such, the authors observed the awkwardness that ultimately resulted in lower (than ~~dess~~ desired) performance.

now the AMD 2900 series)

Design Decisons for the PDP-11/6~~0~~0 Mid-Range Minicomputer

~~This In this pa~~ This ~~paper~~ chapter is a ~~ar dir~~ direct accou n t of the design from the close proximity of the project. ~~un~~ Unliske ~~sever several~~ the chapter ~~of the papers that~~ reports from an architect's or ~~desc descriptive~~ reporter's viewpoint

¶ Four interesting aspects of computer engineering ~~It shows~~ ~~Several good design principles~~ are shown in the 11/60: the cache to reduce Unibus bandwidth; trace-driven design of floating-point arithmetic; providing writeable control store; and increasing the ~~Reliat~~ reliability, availability and maintainability. Whereas the Unibus had previously thought to be inadequte for high performance, the cache is used to unload the ~~i.~~ i/o traffic and provide high performance ~~withought~~ without the attendant cost.* ~~The~~ The ~~use of the Strecker~~ work ~~on~~ is based on Strecker's modelling cache performance ~~is buit built on.~~ Chapter 00. ~~Like the PDP-8,~~ The study leading to determining the block size is given. ~~the~~ block size can only conveniently be ~~1~~ one since ~~undue~~ additional traffic ~~would be b~~ is generated by fetching ~~two~~ multiple words and thereby tying up the Unibus for ~~teh~~ additional traffic. ~~(This study is shown. described.~~

Because it is a mid-range machine, ~~Like all mid range designs,~~ The 11/60 is a ~~df~~ difficult design ~~as~~ for the reasons ~~we discuss~~ of the ~~when we talk about the~~ designer charactersitics, ~~in~~ Chapter 00, page 00. The design is neither the ~~chearerst~~ lowest cost or ~~s~~ performs the most, but has to be ~~in a~~ the right balance of features, price, and performance against criteria that are usually extrememly vague.

The use of trace data to design the floating point artithmetci is ~~definesd.~~ described, ~~or together~~ together with the resulting design ~~with a sketchy view of the~~ ~~There are various algorithms, but~~ ~~Here,~~ It should be noted that the 11/60 performs ~~roun~~ roughly at 11/70 speeds, ~~for a p processor floating point p cost of of x boards versus y boards and a an overall~~

	11/60	11/70
Base Pc	x	x
Floating Point	x	x
Cache	x	x

Memory Managemt O X

Total ~~X~~ ~~X~~

~~The third innovation that is char~~

~~The third aspect of the design that is carried forward to the user levle~~

~~is micro user microporgramming using a writeable control store~~

~~The micro~~

Microprogramming ~~is discussed as it~~ is used to provide both increased ~~user~~ user-level capability ~~with the~~ ~~when the~~ ~~writeable control store option is used~~ and increased reliability, ~~and~~ availability and maintainability. ~~Since a~~ the large,

~~The~~ writeable control store option is ~~described~~ ~~avail~~ ~~available~~ ~~des~~ described as together with its use for data storage and various application. This option has been recently used for ~~carry~~ emulating ~~a~~ the PDP-8 ~~RT-8~~ with ~~RF~~ OS/8 ~~Operating~~ Syst operating system.

¶ A general discussion of microprogramming is also given, especially with respect to its memory technology advances (see also chapter 1, page 20 ff).

~~At~~ ~~The~~ Other ~~aspects of~~ semiconductor technology improvements are described ~~togt~~ together with ~~their use~~ how they effect price and performance. It is interesting to note that the simple concept of tri-state logic* had ~~sot~~ such a great effect on the design.

* Ability to ~~dot wire~~ interconnect a number of subsystems together through a wired-or connection,

Turning ~~Sisters in~~ Cousins into Sisters: The Role ofSoftware in Smoothing Hardware Differences

Since Fortran is ~~the~~ quite possibly the most often executed language and the one we intended that it execute
for the PDP-11, it is important to observe ~~have this a d disc discussion of the archit~~ the 11 — its user,
architecture ~~aas~~ seen by the language processor. ~~On hte other hand, since~~
~~the we designed the 11 f to be able to execute Fortana p easily, this~~
~~paper also provided a critique of the 11 for the this ind intended~~
~~function.~~ The ~~original Fortran com~~ first Fortran compiler ~~are~~ and
(run) object time system are described, together with the evolutionary extensions
~~(evolution to)~~ to improve performance. The Fortran ~~4~~ IV+ compiler is only
briefly discussed, since ~~it the im~~ its improvements are largely
a ~~mete~~ matter of compiler ~~and~~ optimization technology and
~~have~~ are less relevant to the ~~arc~~ architecture 11.

~~The One of the problems~~

~~The title is not completely accurate, because it~~

The chapter title overstates the problem, ~~as it turned~~ since
~~out quite easy to have~~ the variations of the five ~~of the~~ 11 ISP ~~are~~ for
~~that dealt with~~ floating point arithmetic be compatible ~~4~~ by providing are made to
what amounts to ~~sep~~ five seperate object (run) time systems, and a B
single compiler, ~~becuase.~~ This transparency comes about because a simply
concept called threaded code is used to provide what is a very simple
interpreter for the PDP-11, --and might not be called a interpreter requiring two memory cycle times
by many. With threaded code, one 1 word ~~since only~~ 1 instruction is executed per transfer of control
~~for each one two memory cycle instruction is executed~~ each time the
next ~~subrouting e is~~ operation code to be interpreted is ~~is given.~~
~~Thus~~ For a simple integer expression like ~~A+A+1~~ A=A+1, which
~~can be exec~~ occupies ~~two~~ 2 memory words and ~~takes~~ requires 3 memory cycles to

execute is ~~interpreted~~ transformed into ~~ist~~ a ~~its~~ threaded code version,

the program still only occupies 2 words, but instead requires (nearly a factor of 2)

5 memory cycles to execute. ~~Since I C. Since, in the early machien~~

Jim Bell ~~arp~~ discovered this technique and it has been used extensively for time and space

other compilers, ~~but~~ The ability to carry out the interpretation because the overhead is so low

so elegantly was not part of the ~~d~~ original design, but rather turned

otu to be possilbe ~~from~~ based on the generality ~~provided~~ in the 11's indexing modes.

For longer ~~operai oper~~ operations ~~such as~~ o require longer execution times,

~~the~~ like floating point arithmetic, the overhead turns out ~~ot~~ to be quite ~~trivial.~~ low and the space utilization is quite good

The first version of the Fortan machine constructed ~~whas~~ was a simple stack

machine. As such, the execution ~~time~~ times turned out to be quite long. By ~~recod~~ recognizing

~~observing~~ the special cases (eg A=0, A=A+1, etc) and having better conventions

for three-address operaritons to ~~stack~~ and from the stack, speedups of

1.3 and 2.0 were ~~obs os observed~~ ~~obser~~ obtained when a more complex machine was constructed ~~in~~ for the

~~th~~ second version. for floating point and integers, respectively,

~~The finale of the paper deser compa~~

~~Since~~ Since this paper is really on the construction of an ~~alternative~~ extension

to the 11 ~~the 11 to~~ ~~machine~~ to ~~in~~ excecute Fortran, it is interesting to ~~note that~~ compare it with

~~in the final comparison~~ the Fortan IV+ machine which uses the Floating Point

Processor (11/45, 11/55, 11/60, 11/70). The two ~~take~~ turn out to be roughly the same speed and ~~have the same~~ have the same

program space efficiency assuming the

~~occupy the same number of words as a machien microprogrammed to interpret the evolved~~ the

Fortran machine described in the paper is microprogrammed.

* Using Amdahl's constants, page 00, the reader might compute for ~~with~~ this speed processor ~~on the~~ the bus bandwidth given the cache ~~induced needed~~ (for i/o traffic) ~~and the address space needed based~~ and the address space needs and compare these needs with the Unibus.

What Have We Learned From the PDP-11?

~~This chapter was originally written~~ for the CMU ~~10th anniversary~~ Computer Science ~~Sysm~~ 10th anniversary, September 1975. ~~in the summer of 1975. The cp chrater This~~

(... in content)

~~ttt~~ This chapter is a substantially revised & version* of a paper written

* The paper is 50% longer.

a single footnote

The ~~ins~~ introductory overview has been ~~bben~~ deleted (and is now placed in Chapter 1), and the section on the 11/45 and ~~12~~ 11/70 have been ~~b~~ greatly expanded to include the ~~a major~~ perturbations of the memory addressing due to and protection extensions. ~~and there has been a section added which gives a btt more extensive ev evaluation of the performance~~ ~~model of the~~ evolution as model ~~is given~~ ~~of the machine in terms of cost,~~ of the cost, and performance ~~as~~ characteristics ~~is given gov~~ has been added. More historical facts are introduced, particularly as they effect the design of the ~~11/45~~ extensions. Finally the basis (need) for the VAX/11 extension is discussed.

~~The original~~ This paper ~~whas intended to be a critaute of the~~ ~~or~~ was written to critique the original expository paper on the PDP-11 (chapter 00) and to ~~show how the actual~~ compare the actual ~~evolution~~ with the predicted evolution. ~~It does this generally with respect to~~ The four critical issues of ~~tech...~~ ~~as described o earlier in pas on page 00, adn it gives an elaborate comparision of the differences especially vis avis the issue of muliprocesors.~~

The role of semiconductor memories is especially critical.

~~A section~~ The first part of the chapter ~~d~~iscusses ~~the use of te~~ how the technology is used ~~to~~ as a basis for the evolution (something we did not understand when the ~~PDP~~ machies were originally ~~designed~~ planned). The ~~evolution f the~~ Unibus ~~ses~~ evolution is given ~~busses is givein~~ and the case is made for ~~the Unibus~~ its optimatlity. ~~Certainly~~ The Unibus has had greater longevitty and use than any of the other DEC busses and compares favorabley with the IBM ~~One~~ I/O Channel ~~Bus~~ Bus as a universal standard of interconnection.

technology ~~evolution~~, bus bandwidth, address space (and PMS structure) and data-type evolutions ~~evo~~ are ~~described~~ examined,

(2)

~~A~~ The next section ~~de~~ describes the ~~aspects of the evol~~ evolution ~~of the~~ from the point of view of the ~~people who led the~~ various projects development ~~toagether with how thye interactd.~~ and people. Some early historical documents are introduced ~~as further~~ to attempt further an understanding of the design process.

(4) Although ~~a section is given on the~~ ISP an evaluation is given, it is quite weak, ~~and the~~ By comparison of Chapter 00 by Brender, gives a ~~si far more~~useful ~~in terms of the~~ evluatinon of the architecture. ~~Some~~ for Fortran execution

(6) ¶ The ~~fi#~~ find section describes ~~some of the~~ some general problems ~~in d~~ encountered in computer structures ~~and~~ design and engineering, together with how ~~these~~ solving them might be useful in ~~to a s research problems to solve~~ subsequent designs.

¶ The main ~~part of the~~ section ~~paper co~~ ~~chapter~~ (1) consists ~~on~~ of evaluating: the Unibus, examining ~~cost~~ the cost-performance evolution, ~~the~~ ~~[illegible]~~ the ISP and asking why multiprocessors haven't evolved.

(3) We try to provide ~~a de~~ a set of ~~evolut~~ evolutionary cost performance (18-bit, 12- and 36-bit) metrics so the 11 can be compared with the other machines in the book (chapter 00, 01, and 02 ~~DB~~). Also here we go into the ~~[illegible]~~ unique problem the 11 has of designing a range of machines.

(5) A complete section is given on the ~~problems~~ addressing ~~p~~ extension beyond the 11/45 and 11/70 extensions ~~as~~ which ~~caused~~ required a major perturbation in the form of the VAX/11 extension.

PDP-11 GLUE 12/16/77

This part ~~can~~ could stand alone as a book on the evolution of the PDP-11 computer structure ~~evolution~~, although it ~~relies~~ does rely on the conceptual framework of Chapter 1, and the ISPS language description given in appendix 00. The 11 has evolved quite differently ~~than~~ from the other computers in this book, and as such provides an independent and interesting story. The factors that have created the machines are clearly market and technology based; ~~generating~~ they have generated a large number ~~(ten)~~ of ~~different~~ implementations (ten) over ~~its~~ a relatively short (~~seven~~ eight-year) lifetime. Because there are multiple implementations spanning a performance range ~~is an expanding range of machines versus time~~ at the same points in time, the PDP-11 ~~provides~~ provides problems and insight ~~in addition to~~ which did not occur in the evolutions of the traditional mini (8 Family); the best cost/performance machines (18-bit), and the high performance ~~less than $500K~~ machines (the DECsystem 10). The 11 designs cover a range of ~~a factor of 500~~ 500:1 in system price ($500 to $250,000) and 500:1 in memory size (4 Kwords to 2 Mwords).

~~This~~ The part is divided into ~~five~~ six sections. ~~consisting of~~:

1. Introduction.

   ~~This~~ The first chapter (00), published when the 11 was announced, introduces the architecture, gives its goals, and predicts how it might evolve. The family notion is quite strong, although not specific about members. Chapter 00, What Have We Learned From PDP-11, might be read next in order to get the broadest overview, and best immediate critique of the 11 evolution.

2. Conceptual Basis.

This section contains two papers, which when ~~that~~ ~~combined with Chapter 00,~~ form some of the conceptual basis for the models. Strecker, Chapter 00, describes the cache memory mechanism for building high performance models (specifically the 11/70). Levy describes the intercommunication problem among physical components and how busses carry out this task.

3. Implementations.

Of the four implementation chapters, three are on specific implementations (LSI-11, CMU-11 and 11/60) and the fourth (Chapter 00) is a study of all models. This latter chapter provides comparative data on ~~details of~~ the various implementations together with how the designs fit a conceptual model.

4. Evaluation. new line Chapter 00 evaluates the 11 as a machine for executing FORTRAN. What We Have Learned From PDP-11, Chapter 00 discusses ~~all~~ aspects of the PDP-11 evolution in terms of the models introduced in Chapter 1. ~~introductory Chapter (00).~~

5. VAX-11
This paper, by the architect of VAX-11, discusses the new architecture and its first implementation, the VAX-11/780.

6. The multiprocessor research computers of Carnegie-Mellon University.

insert blank line

Three multiprocessors: C.mmp (Chapter 00), Cm* (Chapter 00) and C.vmp (Chapter 00) give examples of a 16-processor multiprocessor, a set of computer modules based on LSI-11, and a triplicated, voting multiprocessor computer for high reliability.

A NEW ARCHITECTURE FOR MINICOMPUTERS--THE DEC PDP-11

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed, to describe the first (11/20) implementation, at a time when the whole family architecture had not been worked out or even fully considered; and to show possible extensions. This was attempted —

~~The reader might note that~~ The computer class definitions (given in 1970) of micro, mini and midi have stood the test of time for they correspond quite closely to those of Chapter 1.

The major reasons (elaborated upon in Chapter 00) for ~~Although we comment on~~ the disparity between the predicted and actual evolution ~~in Chapter 00, some of the important reasons~~ are:

1. The notion of designing with improved technology, especially for a family, was not understood in 1970 ~~then~~. This understanding came later and was put forth in a paper in 1972 (Bell, Chen, Rege).

2. The Unibus ~~bandwidth~~ proved unacceptable for most ~~all~~ communications at the very high and low end designs. Although ~~Whereas,~~ this chapter posits a multiprocessor, and multiple Unibusses for high end designs, this exact ~~this precise~~ structure did not evolve as a standard. Levy's chapter elaborates on the bus evolution. The bandwidth has subsequently been shown to be adequate for all but the largest configurations when a cache is attached to the processor as in the 11/60 (see Chapter 00). Note the effect of a 90% cache hit rate is to reduce the number of access to primary memory via the Unibus by a factor of ten!

delete, but make sure points are made in Levy

3. The physical memory address space was too small.

4. The particular data-type extensions were~~n't~~ not predicted. While floating point ~~data~~ arithmetic was discussed, the character string and decimal operations were~~n't~~ not described. These data types evolved in response to market need (and COBOL) — factors which did not exist in 1970. ~~which didn't exist for DEC at the time the machine was designed.~~

We have made a major change in the chapter by removing the original ISP description and replacing it with a correct and complete by adding (memory management and floating point) ISPS description given in Appendix Ø of the book. ~~that was used when the 11 was evaluated as a Computer Family Architecture (CFA) standard by the U. S. Defense Department~~.

CACHE MEMORIES FOR PDP-11 FAMILY COMPUTERS

Chapter 00 by Strecker is included for four reasons: it is a clear exposition of the cache memory structure and its design parameters; the cache is the basis of ~~the~~ a fast PDP-8 [Bell et al, 1974] ~~(Chapter 00)~~, the 11/60 (Chapter 00) the 11/70 (page 00), and the KL10 (Chapter 00); the design methodology is well done--it is "good engineering"; and finally, the paper is "well-written"--in fact, it received the award for the Best Paper at the Third Computer Architecture Conference. ~~at which it was presented. Thus, since a relevant paper that provides a clear exposition, while illustrating good engineering and being well-written, is so rare~~ We also publish it simply to serve as an encouragement and an example for those who should understand and describe their work — such well-written and relevant papers are only too rare.

The cache design process is implicit in the way in which the work is carried out to determine the structure parameters. ~~It should be noted~~ Note that it ~~was~~ is easy to collect ~~data~~ statistics about ~~the~~ PDP-11 program~~'s~~ behavior since the trace bit, in the PS word, ~~T,~~ permits the 11 to interpret itself on a ~~one at a time~~ single-instruction basis. The relevant sensitivity plots (runs) are made to determine the effect of each parameter on the design. In the 11/60, Mudge (Chapter 00) uses Strecker's program traces and methodology. One of the important parameters to understand is the time between changes of context. To the best of our knowledge this study is unique. ~~This understanding is important~~ because all real time and multiprogrammed systems have many context switches. The PDP-8 cache design (Chapter 00) shows the effect of segmenting the cache for instructions and data.

→ confirm KL10

→ differs between 11 and 8

¶ ~~The PDP-8 chapter discusses the performance for a particular design, and the performance numbers are in terms of a speed-up factor.~~ Strecker gives the performance evaluation in terms of cache miss ratios whereas the 8 data are in terms of a speed up factor. ~~The~~ These two performance measures, see Fig. Cachespeed, are related (Lee, 1969) in the following way (assuming an infinitely fast processor):

p	=	total no. of memory accesses by the processor, Pc
m	=	no. of memory accesses that are missed by the cache and how to be referred to Mp
t.c	=	cycle time of cache memory, Mc
t.p	=	cycle time of primary memory, Mp
R	=	t.p/t.c (ratio of memory speeds), where R is typically 3 to 10

¶ ~~A~~ A cache design for a PDP-8 [Bell et al, 1974] is summarized in the introduction to Part II. Two differences from the 11 work are of interest. The 8 study shows the effect of separating instructions and data whereas the 11 does not.* The second difference is that the relative execution speeds are:

$$t(\text{no cache}) = pR$$

$$t(\text{to cache}) = p + mR$$

$$\text{speedup} = pR/(p + mR) = R/(1 + (m/p)\ R) = a = \text{miss ratio} = m/p$$

therefore

$$\text{speedup} = R/(1 + aR) = 1/(a + 1/R)$$

note that if $a = 0$ (100% hit), the speedup is R; while

if $a = 1$ (100% miss), the speedup is $R/(1 + R)$, i.e.,

the speedup is less than 1 (i.e., time to

reference both memories)

IMPLEMENTATIONS

A MINICOMPUTER-COMPATIBLE MICROCOMPUTER SYSTEM: THE DEC LSI-11

~~This first chapter of the implementation section~~ was written from the viewpoint of a knowledgeable user, ~~viewpoint~~. Although the paper is a descriptive narrative about the design ~~of~~ at each of the chip~~s~~, board~~s~~, and backplane levels, ~~it~~ lacks insight that the designers at Western Digital or DEC (Duane Dickhut, Lloyd Dickman, Rich Olsen, or Mike Titelbaum) might have provided. It An account of the chip-level design is available (Soha and Pohlman, 1974).

The design was done at Western Digital by Roberts, Soha, and Pohlman as a relatively general purpose microprogrammed computer that could be used to emulate many computers. When DEC started working with the designers to effect ~~them for use in the~~ interpretation of the 11 ISP, the design took on more of the 11 structure. Two design levels are described in the paper: the 3 chip microprogrammed computer as it is used to interpret the 11 ISP, and the particular PMS-~~module~~ level components as they are integrated into a backplane to form a hardware system. Sebern points out the microprogramming tradeoff that took place between the chip and module levels to carry out functions normally in hardware: ~~functions:~~ the time clock, the console, refreshing dynamic random access MOS memory, and power fail control.

The subtleties and uniqueness of the module structure are not described, nor are the design alternatives. For example, ~~the alternative,~~ a bounded design that is typical of one board microcomputers was considered, though not described. However, a ~~A~~ lower-cost, one-board system, like the VT78, (page 00) has evolved and is shown in Fig. ~~KXT11~~ RXT-11. The RXT11 is an integrated system containing an LSI-11 chip set, 32Kwords of memory, connectors for six EIA interfaces, and a controller for two floppy disk drives. 175 i.c.'s were used — to implement the same functionality using standard LSI-11 modules, 375 i.c.'s would have been used. [GB: Not announced yet – probably March]

The initial module-level design for the LSI-11 was predicated on a quad-sized form factor with ~~and plugged into~~ a conventional backplane. The modules are shown on page 00. Since there were not special ICs beyond the 3 chip processor, options tended to be relatively large and often occupied a full quad module. For options that were greater than a quad size, an ingenious packaging scheme was devised. It provided ~~used for providing~~ interconnection points on the extra half of the module (a double sized module) which was not used as the LSI-11 Bus (requires a double sized module). This permitted multiple board, complex options, (e.g., a disk controller), to be packaged as a single option with no interconnection

[illegible]

between the boards except via the second half of the quad board. As DEC began to build special ICs to interface to the bus, the option sizes ~~began t~~o decreased to occupy a double module. This system is now known as the LSI-11/2. A backplane system with modules is shown in Fig. LSI-11/2. The options available for the two systems have evolved as shown in Table LSI-11 Options.

Table LSI-11 Options

[Teicher to supply]

The effect of a lower cost LSI-11 system is to provide additional applications as we discuss in Chapter 1, page 00.

USING LSI PROCESSOR BIT-SLICES TO BUILD A PDP-11--A CASE STUDY IN MICROCOMPUTER DESIGN

This paper by the designers of CMU-11 appears both in the module part and in this part on PDP-11 implementations. The Intel 3000 bit slice, herein called a Microcomputer (for Microprogrammed Processor), is used to interpret a PDP-11 ISP. The purpose of the design was to test the assertion that the bit-slice based arithmetic unit with register memory and microprogrammed control would simplify the design and construction of processors. The 11 was selected as a target problem in order to avoid the temptation of changing the problem (a real ISP ~~11 processor~~) to fit the building blocks (the Intel 3000 processor). Indeed ~~As such~~, the authors observed ~~the~~ awkwardnesses that ultimately resulted in lower (than desired) performance.

In retrospect, the Intel 3000 has not become the standard bit-slice architecture that the AMD 2900 series has~~; but~~ perhaps it suffer~~s~~ed from being one of the earliest. Detailed comparisons including a breakdown of the various parts of the processor design are given and compared with the LSI-11 and 11/40 designs in terms of performance and cost (IC count and number of control bits in the microprogrammed controller).

A key part of the investigation was ~~to~~ the evaluate~~~~ion of the computer-aided-design tools. The Stanford University Drawing System (SUDS) and the SAGE logic simulator were the ~~key~~ major components. SAGE was predicated on being able to detect all the design and timing flaws prior to construction of a design. The authors claim that 95% of the errors were detected ~~by the simulation--and all errors were ultimately detectable once the simulation data or machine description was changed.~~ in this manner. The ~~simulated/real~~ simulated-time/real-time ratio ($10^6/1$) did constrain the design process, e.g., the number of different runs used to find worst-case conditions.

DESIGN DECISIONS FOR THE PDP-11/60 MID-RANGE MINICOMPUTER

Unlike the reports from an architect's or reporter's viewpoint, this chapter is a direct account of the design from the close proximity of the project. A Mid-range machine is an inherently difficult design ~~because~~ for the ~~Because it is a mid-range machine, the 11/60 is a difficult design for the reasons~~ reasons of the designer characteristics we presented in Chapter 00, page 00. [CM: unclear] ~~The design~~ It is neither the lowest cost nor ~~performs the highest of the 11s, but~~ highest performance PDP-11, it has to be the right balance of features, price, and performance against criteria that are usually extremely vague.

Four interesting aspects of computer engineering are shown in the 11/60:

√

the cache to reduce Unibus ~~bandwidth~~ traffic; trace-driven design of floating-point arithmetic processors; providing writeable control store; and increasing the reliability, availability and maintainability.

Whereas the Unibus ~~had~~ was previously thought to be inadequate for high performance systems, by using a cache ~~most processor~~ memory references do not use the Unibus and so leave it free for ~~, the cache is used to unload the~~ i/o traffic. ~~and provide high~~ This gives high performance without the attendant cost*. The cache work is based on Strecker's (Chapter 00). The study leading to determining the block size is given. The block size can only conveniently be one since ~~additional traffic is generated by~~ fetching multiple words ~~which~~ would tie up the Unibus with additional traffic.

The use of trace data to design the floating point arithmetic is described together with the resulting design. Note that the 11/60 performs roughly at 11/70 speeds ~~with~~ but at lower cost. The implementation of the two can be compared in the following table.

Table - Implementation of 11/60 and 11/70 (count of printed circuit boards)

	11/60	**11/70**
Base Pc	~~x~~ 5	~~x~~ 8
Floating point	~~x~~ 4	~~x~~ 4
Cache	~~x~~ 1	~~x~~ 4
Memory management	0	~~x~~ 2
	10	18

[CM: check 11/70 Sys Manual]

*Using Amdahl's constants, page 00, the reader might compute the bus bandwidth (for i/o traffic) and the address space needs for this speed processor given the cache and compare these needs with the Unibus. [CM: we should do this for all models in "What we've learned" chapter].

Total	~~x~~ 10	~~x~~ 18

Microprogramming is used to provide both increased user-level capability and increased reliability, availability and maintainability. The large writeable control store option is described together with its use for data storage and various applications. This option has been recently used for emulating the PDP-8 ~~with~~ at the OS/8 operating system~~s~~ level.

A general discussion of microprogramming is also given, especially with respect to memory technology advances (see also Chapter 1, page 00). Other semiconductor technology improvements are described together with how they affect price and performance. It is interesting to note that the simple concept of tri-state logic* had such a great effect on the design.

Impact of Implementation Design Tradeoffs on Performance: The PDP-11, A Case Study

This chapter presents a most comprehensive comparison of the eight ~~PDP-11~~ processor implementations~~s~~ used in the ten PDP-11 models. The work was carried out to investigate various design styles for a given problem--the interpretation of the PDP-11 ISP. ~~Aside from any conclusions t~~The tables alone ~~[illegible]~~ give more insight into processor implementations than is available from any single source we know. The usefulness of the data also comes from having an outside observer examine the machines and ~~provide~~ share his insight.

The tables include:

*Ability to interconnect a number of subsystems together through a wired-or connection.

1. a set of instruction ~~use~~ frequencies, by Strecker, for a set of ten different applications ~~an operating system~~. The reader should note the frequencies do not reflect ~~general~~ all uses ~~purpose use~~, e.g., there are no ~~arithmetic and~~ floating point instructions~~;~~, nor has operating system code been analyzed.

2. implementation cost (modules, ICs, control store widths) and performance (micro- and macro-instruction times) for each model; and

3. a canonical data path for all 11 implementations, ~~and~~ against which each processor is ~~presented and~~ compared~~ with it~~.

With this background data, a "top-down" model is built which explains the performance (macro-instruction time) of the various implementations in terms of the micro-instruction execution, and primary memory cycle time.

Since these two parameters ~~don't~~ do not fully ~~explain~~ (set) {model} performance, a set of bottom-up factors must be introduced. These factors include various design techniques and the degree of processor overlap. We believe ~~the fact~~ that this analysis of a constrained ~~that the~~ problem ~~is constrained~~ should provide useful insight to both computer and general-digital-systems designers.

EVALUATION

TURNING COUSINS INTO SISTERS: THE ROLE OF SOFTWARE IN SMOOTHING HARDWARE DIFFERENCES

~~*Ability to interconnect a number of subsystems together through a wired-or connection.~~

Since FORTRAN is quite possibly the most often executed language for the PDP-11 and the one we intended that it execute, it is important to observe the 11 architecture as seen by the language processor--its user. The first FORTRAN compiler and object (run) time system are described together with the evolutionary extensions to improve performance. The FORTRAN IV~~4~~-PLUS compiler is only briefly discussed since its improvements, ~~are~~ largely ~~a~~ due to ~~matter of~~ compiler optimization technology ~~and~~) are less relevant to the 11 architecture.

The chapter title overstates the compatibility problem since the five variations of the 11 ISP for floating point arithmetic are made to be compatible by essentially providing ~~what amounts to~~ five separate object (run) time systems and a single compiler. This transparency is provided quite easily using a concept called threaded code. This concept appears to be a very simple interpreter for the PDP-11--and might not be called an interpreter by many. With threaded code, one 1-word instruction requiring two memory cycle times is executed each time ~~the next~~ a high level operation code is to be interpreted, otherwise the processor is carrying out the desired op code. When a simple integer expression like ~~A~~ I = ~~A~~ I + 1, which occupies 2 memory words and requires 3 memory cycles to execute, is transformed into a threaded code version the program still only occupies 2 words, but instead requires 5 memory cycles to execute (nearly a factor of 2). For more complex operations requiring longer execution times, like floating point arithmetic, the overhead turns out to be quite low and the space utilization is quite good. ~~Although not used,~~ It is also possible to move efficiently between threaded and directly executed code, Although ~~[illegible]~~ this is not done. Jim Bell

~~*Ability to interconnect a number of subsystems together through a wired-or connection.~~

discovered ~~this~~ the technique~~; and~~ it has been used extensively for other compilers because ~~the~~ of its low time and space overhead~~ is so low~~. The ability to carry out the interpretation so elegantly was not part of the original PDP-11 design, but rather ~~turned out to be possible based on~~ was a consequence of the generality ~~in~~ of the 11's ~~indexing~~ addressing modes.

The first version of the FORTRAN machine constructed was a simple stack machine. As such, the execution times turned out to be quite long. ~~By~~ In the second version, by recognizing the special high~~-use~~ frequency of use cases (e.g., A = 0, A = A + 1, ~~etc.)~~) and by having better conventions for three-address operations (to and from the stack), speedups of 1.3 and 2.0 for floating point and integers, ~~respectively,~~ were obtained~~ when a more complex machine was constructed for the second version~~.

~~Since this paper is really on the construction of an extension to the 11 to execute FORTRAN,~~ It is interesting to compare ~~it~~ the virtual machine described with the FORTRAN IV~~+~~-PLUS machine which uses the floating point processors (on the 11/34, 11/45, 11/55, 11/60, and 11/70). If the FORTRAN machine described in the paper is implemented in microcode ~~microprogrammed~~ and made to operate at FPP speeds, the ~~two~~ resulting machines turn out to operate at roughly the same speed and programs occupy roughly the same program space.

WHAT HAVE WE LEARNED FROM THE PDP-11?

This chapter is a substantially revised version* of a paper written for the CMU Computer Science 10th Anniversary, September 1975. This paper was written to critique the original expository paper on the PDP-11 (Chapter

*The paper is 50% longer. The introductary overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.

00) and to compare the actual with the predicted evolution. The four critical issues of technology, bus bandwidth (and PMS structure), address space and data-type evolutions are examined.

The first part of the chapter discusses how the technology is used as a basis for the evolution (something we did not understand when the machines were originally planned). The role of semiconductor memories is especially critical. The next section describes the evolution from the point of view of the various development projects and people. Some early (historical) design documents are introduced to further aid in understanding the design process.

~~The main section consists of: evaluating the Unibus, examining the cost-performance evolution, the ISP and discussing the organizational issues behind why multiprocessors haven't evolved. As a comparison, Chapter 00 on C.mmp describes the technical problems associated with multiprocessors.~~

The Unibus evolution is given and the case is made for its optimality. The Unibus has had greater longevity and use than any of the other DEC busses and compares favorably with the IBM I/O Channel Bus as a universal standrad of interconnection.

We try to provide a set of evolutionary cost-performance metrics so the 11 can be compared with the other machines (18-, 12- and 36-bit) in the book (Chapter 00, 01, and 02). Also, here we go into the unique (within DEC) problem ~~the 11~~

~~*The paper is 50% longer. The introductary overview has been deleted (and is now placed in Chapter 1) and the sections on the 11/45 and 11/70 have been greatly expanded to include the perturbations due to the memory address and protection extensions. A detailed evolutionary model of the cost, and performance characteristics has been added. More historical facts are introduced, particularly as they effect the design of the extensions. Finally the basis (need) for the VAX/11 extension is discussed.~~

~~has~~ of designing a range of machines.

Although an ISP evaluation is given, it is quite weak. By comparison, Chapter 00 by Brender, gives a more useful evaluation of the architecture for FORTRAN execution. A complete section is given on the addressing extension, beyond the 11/45 and 11/70 extensions, which required a major perturbation: ~~in the form of the VAX/11 extensions.~~ VAX-11.

The final section, addressed to the research community, describes some general problems encountered in structure design and engineering, together with how solving them might be useful in subsequent designs. ~~(This part has been expanded too.)~~

VAX-11/780: A VIRTUAL ADDRESS EXTENSION TO THE DEC PDP-11 FAMILY

[section on Strecker's paper to go here]

~~MULTIPROCESSOR~~

THE MULTIPROCESSOR RESEARCH COMPUTERS OF CARNEGIE-MELLON UNIVERSITY

~~These~~ Three ~~multiprocessor~~ computers, which use the 11 as a basis, were built at Carnegie-Mellon University to carry out ~~various~~ research in computer structures, and operating systems ~~and application program computer engineering and computer science research.~~ for multiprocessors

The first computer, C.mmp, is a 16 processor (~~model~~ 11/40's and 11/20's) system with ~~4~~ 2.5 million words of shared primary memory. It was built to investigate the programming (and resulting performance) questions associated with having a large number of processors.

✓

The ~~[illegible]~~ chapter on the second computer, Cm*, ~~is also discussed~~ is located physically in the Modules Part, as the modules that form ~~Cm*~~ Cm* are LSI-11's ~~computers.~~ Cm* ~~evolution~~ was based on the premise that ultimately, the smallest modular unit ~~(i.e., ultimately all~~ , i.e., integrated circuit, ~~ICs) that would be~~ used to build digital systems ~~is the~~ would be a computer. With this premise, ~~it is important~~ we need to understand how to interconnect computers physically, how to assign parts of the ~~problem~~ application program to the various computers, and how to program the complete structure.

The third computer, C.vmp, ~~for voting multiprocessor,~~ was designed to investigate how a production microcomputer, the LSI-11, ~~can~~ could be used to build a triplicating, voting high availability computer.

The goals of the first two are performance while the third ~~uses multiple computers for redundancy and~~ has reliability as its goal. ~~To this end, Fig. Perf shows the effect of using multiprocessors to execute various algorithms (More to come here!).~~

We believe that technology will force the evolution of computing structures to ~~be along all~~ converge ~~[illegible]~~ to three ~~lines~~ styles of multiprocessor computers: (1) C.mmp-style, for high performance, incremental performance and availability; (2) ~~less tightly~~ loosely coupled computers like Cm* ~~for more ad hoc structures~~ to handle specialized processing, (e.g., front end, file, and signal processing); and (3) C.vmp-style for high availability based on increased maintenance costs.

The technology-force argument is based on history, near term technology, and ~~the~~ resulting price extrapolations. ~~and follows:~~

1. ~~The~~ MOS technology is increasing in both speed and density faster than the technology, (e.g., ECL), from which high-performance machines ~~is~~ are built. ~~formed;~~

2. The price per chip of the single-MOS- ~~MOS one~~ chip processors decreases at a substantially greater rate than for the low-volume high-performance, ~~tasks~~ special designs. ~~due to the~~ Chips in both designs have high design costs, but the special design enjoys a much lower volume. ~~much higher volumes and very high design costs (for both designs);~~

~~3. For all intents and purposes,~~ the processor cost ~~of the~~ in a low end system is ~~0~~ essentially zero.
3. Relative to all other costs of ~~the~~ a system. ~~The resulting performance (in operations) per chip and cost per operation per chip are rapidly diverging from the high performance semiconductor technology from which conventional high performance machines are formed.~~

4. Standards (in the semiconductor industry) for high volume products tend to form more quickly. For example, in the 8-bit microcomputer market, one ~~1~~-type supplies about 50% of the market and 3 types supply over 90%. ~~of the market.~~

5. A 16-bit processor (-on-a-chip), with both an address space matching its ~~an acceptable (for the performance) address space~~ and appropriate data-types, has been announced. ~~is eminent.~~ Such a commodity will form the basis for nearly all future computer designs.

~~These factors produce better (in terms of price and price/performance) computers at the low cost market end at a diverging rate compared with large scale computers.~~ Furthermore, large scale applications have been

As a result of these factors, the two classes of machines (MOS- ~~MOS~~ processor-on-a-chip-based and low-volume, high-performance-processor-based) have rapidly diverging costs per operation per chip.

slow to form since problems complexity increases more rapidly than program size. Therefore, most subsequent computers will be based on standard, high volume parts. For high performance machines, since processing power is available at ~~0~~ essentially zero cost from ~~a~~ processor-on-a-chip-based processors, large scale computing will come from arrays of processors, just as we build arrays of ~~16~~ 64K bit ICs to form memory subsystems.

C.mmp--A Multi-Mini-Processor

C.mmp was motivated by the need for more computing power to solve speech recognition/signal processing problems and to understand the multiprocessor software problem. Until C.mmp, only one large, tightly coupled multiprocessor had been built--the Bell Laboratory's Safeguard Computer (BSTJ issue?).

The introductory section describes the economic and technical factors influencing multiprocessor feasibility and argues ~~that it's important to embark on the research because of~~ for the timeliness of the research. Various problems to be researched are given together with a discussion of particular design aspects. For example, since C.mmp is predicated on a common operating system there ~~is~~ are two sources of degradation: ~~both due to~~ memory contention and lock contention. ~~when all the processors use the same memory block and since there are common synchronizing locks for parts of the software, degredation may occur when a processor must idle and wait to enter common critical sections.~~ The machine's theoretical performance ~~due to~~ as a function of memory-processor interference is ~~computed~~ based on Strecker's (1971) work. In practice, ~~memory interence~~ [illegible]

~~was a problem resulting in poorer than expected performance~~ because the memory interference was greater than expected. memory was not built with low-order address interleaving. This problem ~~had~~ was ~~to be~~ solved by moving program segments ~~of the memory.~~ ~~It should be noted~~

¶ As ~~that when~~ the number of memory modules and processors becomes very large, the theoretical performance (as measured by the number of accesses to the memory by the processors) approaches the memory bandwidth (m/memory-cycle-time) x 1/e. [Ref?]. Thus, with infinite processors, there is not a maximum limit on performance, provided ~~they are~~ all processors are not contending ~~to access~~ for the same memory.

Although there is a discussion outlining the design direction of the operating system, Hydra, later descriptions should be read ~~(list the~~ [Wulf et al, 1975] ~~references?? here).~~ Since the 11's small address ~~impaired use~~ necessitated frequent map changes, 11/40s with ~~microprogrammed~~ writeable control stores were used to implement the operating systems calls ~~involving changing~~ [illegible] to change the segment base registers.

~~One of the most pleasant surprises of C.mmp was the ALGOL 68 implementation because it enables parallel programs to be specified. The result (see page 00) in terms of performance are quite encouraging! There are three basic approaches to effectively applying multiprocesors:~~

~~1. having lots of independent work to do through multiprogrammed, timeshared and multiple independent computers;~~

~~2. having the compiler for a conventional language (e.g., FORTRAN) detect and compile statements that can be executed in parallel;~~

¶ The ~~actual~~ C.mmp which was actually built is shown in Figure PMS C.mmp.

Insert pages 20 a, b, c

There are three basic approaches to the effective application of multiprocessors:

1. system-level workload decomposition.

If a workload contains a lot of inherently independent activities, e.g., compilation, editing, file processing, and numerical computation, it will naturally decompose.

2. program decomposition by a programmer.

Intimate knowledge of the application is ~~applied~~ required for this time-consuming approach.

3. program decomposition by the compiler.

This is the ideal approach. However, results to date have been ~~disco~~ disappointing [Ref Illiac IV FORTRAN compiler projects].

C.mmp was predicated on the first two approaches. Since ~~ALGOL 68~~ the original paper, ALGOL 68, a language with facilities for expressing parallelism in programs, has been implemented. It has assisted greatly with program decomposition. See Figure X.

As can be seen in the paper, a model of the lock problem influenced the number of critical sections in the scheduler. Since the paper, the operating system ~~and~~ Hydra has been designed and implemented. ~~[Wulf et al, 1975]~~ An extensive description is given in [Wulf et al, 1975].

Two experiments ~~on~~ analyzing the performance of C.mmp and Hydra have been reported. The first, by Ranathe [1977], ~~measured the~~ used a hardware monitor to measure the degradation due to the locking mechanism which is invoked when shared data is accessed. The second, by Oleinick [1977], analyzed user-program-level synchronization. We summarize the results of both below.

The contention for shared resources in a multiprocessor system occurs at several levels. At the lowest level processors contend ~~for~~ at the cross-point switch level for memory. This memory interference is discussed in the ~~paper~~ chapter. On ~~At a~~ a higher ~~level~~ there is contention for shared data in the operating system kernel. At higher levels, processes contend for i/o devices and for software processes, e.g., for memory management. The following table points to ~~the~~ models or experimental data at these different levels in C.mmp.

Contention Level	Reference
User-program ~~[illegible]~~	[Oleinick ~~I~~, 1977] [Fuller and Oleinick, 1976]
~~Hydra Kerne~~ Hydra Kernel objects	[Marathe and Fuller, 1977]
cross-point switch	Chapter XX [Fuller, 1976]

Marathe's data show that the shared data of Hydra is organized into enough separate objects that a very ~~lit~~ small degration (less than 1%) ~~to~~ results from contention for these objects. Table ~~I~~ LOCK is reproduced from his paper. He also built a queuing model which projected that the contention level would be about 5% in a 48 processor system.

Oleinick uses a rootfinding algorithm to study various implementations of a single problem.

[CH: Elaborate after studying his thesis chapter]

~~3. introducing new primitives into the programming language (e.g., ALGOL 68) so that algorithms for parallel execution are specified by the programmer.~~

~~Although C.mmp was only predicated on type 1 use, the ALGOL 68 implementation on C.mmp permits parallel programs (type 3) to be written. The results of some of this programming is shown in Fig. ?.~~

<u>Multi-Microprocessors: An Overview and Working Example</u>

~~Cm* is a system of high level modules, constructed of LSI-11 computers, which can be interconnected to form large computer structures~~. The Cm* work, sponsored by NSF and ARPA, ~~are~~ is an extension of earlier NSF-sponsored research [Bell, etc. 1973] on register-transfer-level modules. As LSI and VLSI enable construction of the processor-on-a-chip, it is apparent that low-level, register-transfer modules are passe' for the construction of all but low-volume computers. ~~A complete industry will ultimately design, contribute to and use a standard computer~~. Although the research is predicated on structures employing a hundred or so processors, this chapter describes the culmination of the first (ten-processor) phase.

The authors motivate their work by appealing to the diseconomy-of-scale argument which we advanced at the beginning of this section (page 00). To provide additional context for their research, ~~A motivation of Cm* is based on a diseconomy of scale for large computer introductions during 1975-1977 (see page 00).~~ c~~C~~omputer modules (Cm*), multiprocessors (C.mmp) and computer networks are described in terms of performance and problem suitability. ~~to provide additional context for the research~~. The chapter gives a description of the modules structure, together with their associated limitations and potential research problems. ~~(research)~~.

The final, most important, part of the chapter evaluates the performance of Cm* for five different problems.

C.vmp: The Architecture and Implementation of a Fault Tolerant Multiprocessor

C.vmp is a triplicated, voting multiprocessor designed to understand the difficulty (or ease) of using standard, off-the-shelf LSI-11s to provide greatly increased reliability. There is concern for increased reliability because systems are becoming more complex, are used for more critical applications, and because ~~basic~~ maintenance costs for all systems are increasing. Because the designers themselves carry out and analyze the work, this chapter provides ~~a great deal of~~ first-hand insight into high reliability designs and the design process--especially its evaluation. The system has operated for several months and the first phase of work is complete.

Several design goals are initially predicated and the work is carried out against the goals. ~~Two of the more interesting goals include using off-the-shelf hardware and software with no modifications to the components.~~

The goal of software and hardware transparency turned out to be easier to attain than expected, because of an idiosyncrasy of the floppy disk controller. ~~This~~ Because the controller ~~operates on~~ effects a word-at-a-time bus transfer from a one-sector buffer,

~~after a sector is transferred--thus~~ voting ~~is~~ can be carried out at a very low level ~~(i.e., as bus transfers are made)~~. It is unclear how the system would have been designed without this type of controller; ~~but~~ as a minimum, some form of software transparency goal would have been violated ~~together~~ and ~~with~~ a significant controller modification would have been necessary.

A number of models are given by which the design is evaluated. Various component reliabilities are used and the reader should get a great deal of insight into the factors contributing to reliability. It should be noted that a special hardware voter is needed ~~in order to build~~ to get a sufficiently low cost for a marketable C.vmp. While the intent of C.vmp is not a product, it does provide much of the insight for such a product.

Marathe, M. and Fuller, S.H. A Study of Multiprocessor Contention for Shared Data in C.mmp. Proc 1977 ACM SIGMETRICS conferences, pp 255 - 262.

Wulf, W. [and others] The Hydra Operating System, Fifth ACM SIGOPS Symposium on Operating Systems Principles (Nov 1975).

11 Glue figures & tables

Fig.	Cachespeed	Structure of P_c, Mcache and M_p of cached computer
Fig.	LSI-11/2	Photograph of double height modules forming LSI-11/2
Fig	RXT11	~~Hog~~ Bounded LSI-11
Fig	PMSC.mmp	PMS diagram of C.mmp
Table	Lock	Measurement of the locking behaviour in Hydra/C.mmp.

It is somewhat anticlimactic to discuss this original PDP-11 description here because Chapter 00 explicitly discusses "What We Have Learned from the PDP-11". The purpose of the chapter was originally threefold: to give the PMS and ISP architecture of the PDP-11 as it was first proposed, to describe the first (11/20) implementation, at a time when the whole family architecture had not been worked out or even fully considered, and to show possible extensions. This was attempted

~~The reader might note that~~ The computer class definitions (given in 1970) of micro, mini and midi have stood the test of time for they correspond quite closely to those of Chapter 1.

The major reasons (elaborated upon in Chapter 00) for ~~Although we comment on~~ the disparity between the predicted and actual evolution ~~in Chapter 00, some of the important reasons~~ are:

1. The notion of designing with improved technology, especially for a family, was not understood in 1970 ~~then~~. This understanding came later and was put forth in a paper in 1972 (Bell, Chen, Rege).

[illegible]

2. The Unibus ~~bandwidth~~ proved unacceptable for most ~~all~~ communications at the very high and low end designs. Although ~~Whereas,~~ this chapter posits a multiprocessors and multiple Unibusses for high end designs, this exact ~~, this precise~~ structure did not evolve as a standard. Levy's chapter elaborates on the bus evolution. The bandwidth has subsequently been shown to be adequate for all but the largest configurations when a cache is attached to the processor as in the 11/60 (see Chapter 00). Note the effect of a 90% cache hit rate is to reduce the number of access to primary memory via the Unibus by a factor of ten!

delete, but make sure points are made in Levy