A. Baldwin

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

T. M. McWilliams, S. H. Fuller, and W. H. Sherwood
Departments of Computer Science and Electrical Engineering
Carnegie-Mellon University
Pittsburgh, Pa. 15213

January, 1976

This work was partially supported by the Advanced Research Projects Agency (ARPA) of the Department of Defense under contract F44620-73-C-0074, monitored by the Air Force Office of Scientific Research.

ABSTRACT

In this article we give the design and evaluation of the CMU-11: a fully operational implementation of the PDP-11 computer architecture built with Intel 3000 Schottky bipolar microcomputer bit-slices. This project was initiated to test in detail the claims that LSI processor bit-slices simplify the design of microprogrammed processors. The CMU-11 executes approximately 240,000 instructions per second, which is about 63% the speed of the PDP-11/40 and twice the speed of the LSI-11.

We explore in some detail the additional logic that was added to enable the Intel 3000 circuits to emulate the PDP-11 instruction set. We specified full DEC Unibus compatibility and 29% of the integrated circuits used to implement the CMU-11 were required to provide buffering and control of the Unibus. The other main sources of inefficiency were the lack of arithmetic overflow logic in the bit-slices and the organization of the microinstruction control store. We show how improved LSI circuits in this area can substantially reduce the size (and cost) of the processor.

The set of design aids currently available at Carnegie-Mellon University was of critical assistance in this project and we include a critique of our use of these design aids to show their utility in prototype design efforts.

Key words and phrases: Microcomputers, Computer Organization, LSI circuits, computer implementation techniques, microprogramming, design automation.

1. INTRODUCTION

Several semiconductor manufacturers have recently developed high speed LSI circuits that are designed to simplify the construction of microprogrammed processors and device controllers. These integrated circuits are called "bit-slices" because they implement 2 or 4 bits of the registers, arithmetic units, and primary data paths of a processor. This article presents the design and evaluation of the processor built at Carnegie-Mellon University that uses the Intel 3000 bit-slices and that is microprogrammed to emulate the PDP-11 computer architecture [Intel 75; Signetics 75].* The purpose of this project was to investigate the assertions of semiconductor manufacturers that their LSI bit-slices would in fact simplify the design and construction of processors.

Rather than specify a new architecture (i.e. instruction set) for this experiment in processor design, we decided to reimplement an established computer architecture: the PDP-11. We chose the PDP-11 architecture for several reasons. Using an existing and well-known architecture would allow others to more easily evaluate the results of our experiment and kept us from consciously or unconsciously tailoring the processor architecture to fit the capabilities and idiosyncrasies of the LSI bit-slices. Another reason is that PDP-11's are in extensive use at Carnegie-Mellon Univ. in a wide variety of applications and if our experiment was successful, the processor could be put to work on any one of several practical tasks. It was this second reason that helped establish a criteria that proved to be critical: we demanded that the processor we constructed support the standard DEC Unibus [DEC 73B] that is common to all PDP-11's except the LSI-11. Finally,

---

*We gratefully acknowledge the donation of 3000 microcomputer sets by both Intel and Signetics Corporations.

the PDP-11 architecture is an unusually good test of the capabilities of a bit-slice circuit family because it is a relatively complete architecture with numerous addressing modes and instruction formats.

This paper continues in the next section with a description of the design of the CMU-11 processor. We then discuss the performance, cost and implementation difficulties uncovered during the design and testing of the machine. In addition to the evaluation of the LSI bit-slice circuits for general-purpose processors, we are interested in the problems of computer design in general. For this reason, a fairly complete set of digital design automation aids are available at Carnegie-Mellon University: an interactive drawing package that generates engineering drawing, wire-lists, and aids in engineering changes; a digital simulation system that is interfaced to the drawing system; and microprogram assemblers. Section 6 reviews our experiences with these design aids and we draw some conclusions concerning the process of designing and debugging prototypes of digital systems built with LSI circuits.

2. ORGANIZATION OF THE CMU-11

Figure 2.1 is a register-transfer level diagram of the CMU-11 microprogrammable processor. The processor's components are arranged in the diagram into three sections: the data part, control part, and Unibus interface. We were able to build the entire processor on a single board and Figure 2.2 is a top view of the CMU-11.

2.1 The Data Paths and Working Registers

The data part of the processor is designed around the 3002 (central processing element) bit-slice. A single 3002 circuit implements a 2-bit slice of the data paths and hence eight 3002's have been used in the CMU-11. Although not explicitly shown in Figure 2.1, the 3003 carry-lookahead circuit is also used. With the 3003, the 3002 array is capable of cycling through operations at every 150 ns. However, other delays in the clock and control part dictate that the CMU-11 has a 200 nsec micro-cycle time. The eight general-purpose working registers of the PDP-11 architecture can be kept in the register scratchpad on the 3002's, and the three remaining registers, R8, R9, and T are sufficient for source and destination operand computations as well as other intermediate results. The Program Status (PS) and Instruction Register (IR) were not possible to assign within the 3002's without a severe loss in performance.

The relatively generous number of input and output lines of the 3002's are used to good advantage. The D<15:0> and A<15:0> buses feed the Unibus Data and Adress lines, respectively. In addition, the D bus allowed access to the extra data paths necessary to include the PS register and to facilitate the byte swap operation needed by many of the PDP-11's instruction. The

M<15:0> bus is used as the principle data input bus. The Function bus, F<6:0>, specifies both the operation to be performed by the arithmetic/logic unit as well as the selection of the register in the scratchpad to be involved in the operation. The K<15:0> bus is used to input masks or constants from the microinstruction. The 3000 circuit set makes frequent use of the K lines to specify masks (usually all zeros or all ones) that effectively extend the operation code on the Function bus.

2.2 Control Part

The control part of the CMU-11 uses the 3001 Microprogram Control Unit and a 512 word control store* with 32 bit microinstructions. Figure 2.4 shows the format of the microinstruction and Table 2.1 briefly describes the function of each of the fields. A microinstruction buffer register was included in the design to allow the overlap of the fetch of the next microinstruction with the execution of the current microinstruction which is common technique to improve the performance of microprogrammed processors.

The "next-address logic" of the 3001 has been augmented by additional microbranch control logic external to the 3001. This external logic uses the contents of the Instruction Register, the condition codes in the PS, and the PLA field from the microinstruction register to determine the AC<6:0> lines to input to the 3001.

The other major section of control logic that had to be added to the design was the Processor Status logic to control the setting of the 4-bit condition

---

*In order to expedite the debugging of the microprogram for the CMU-11, we built a fast, simple writable control store for the CMU-11. 45 μsec access time, 1024 bit RAM packages were used to assure a writable control store as fast as the final ROM control store. The writable control store is interfaced to a Unibus (of a PDP-11 other than the CMU-11) for initial loading of microprograms. Figure 2.3 shows the CMU-11 interfaced to the supporting PDP-11 and writable control store.

code in the PS register and control access to the PS. The fact the PS register is defined as primary memory location 177776 in the PDP-11 architecture requires special logic to load and store the PS.

2.3 Interface to the Unibus

A significant fraction of the components of the CMU-11 are devoted to the support of the Unibus. Given the demanding electrical requirements of the Unibus, the tri-state A, D, and M lines of the 3002 array could not be directly attached to the Unibus. Instead, separate transceiver packages had to be used to provide this buffering.

Because to the asynchronous operation of the Unibus and interrupt and non-processor requests (i.e. direct-memory access request via the Unibus) it was not practical to drive the Unibus directly from fields in the microinstruction. Instead, a bus control and timing section used added to the processor. The rest of the processor interfaces to this control unit via the UC<7:0> field in the microinstruction. See Table 2.1 for a description of the functions of the subfields within UC<7:0>.

2.4 Console Functions

In place of a standard front panel, the CMU-11 has front panel functions accessible from a standard teletype attached to the Unibus. Memory locations can be examined and loaded by typing the octal address followed by a slash. The current value is displayed and a new value may be entered if desired, followed by a carriage return. The processor may also be started and continued from the teletype. There is a halt switch on the front panel which causes the machine to return to the console microprogram. This use of a

teletype for a console is similar to the console teletype used by the LSI-11 [DEC 75]. In order to make it easier to maintain the processor we have added a microprocessor console which displays the microprogram address and allows the microprocessor to be single-stepped. The microconsole proved invaluable for debugging the prototype processor.

3. EVALUATION OF CMU-11 DESIGN

The critical questions to be asked about this design concern cost and performance. It has been fairly easy to evaluate the performance of the CMU-11 by looking at several representative instruction times and by running a set of benchmarks on the machine. Evaluating the cost of the CMU-11 has been more difficult. Rather than try to compare the price of existing PDP-11 implementations with the cost of the CMU-11, we chose instead to compare it with other PDP-11's with respect to circuit complexity. The other significant costs, i.e. development costs, are discussed in Section 6.

3.1 Performance of the CMU-11

The CMU-11 runs at a microinstruction cycle time of 200 nsec. The specifications for the Intel 3000 microcomputer family state that it is possible to build a 16 bit minicomputer with a 150 nsec. cycle time. However, given our objective to design as cost-effective an implementation as possible, we avoided the sensitive and complex timing circuits that would be required to approach a 150 nsec. cycle time.

If we had used clocks with sufficient buffering and pulse shaping, a worst-case analysis shows that with the particular IC packages used in the CMU-11, we could approach a 149 nsec. cycle time with Intel 3000 packages and a 126 nsec. cycle time with Signetics version of the 3000 set. We have in fact replaced the Intel 3000 circuits with the Signetics circuits and although the CMU-11 continues to run reliably at 200 nsec., we cannot reduce the cycle time below 200 nsec.: the critical path is in the control part and not the 3002 array.

Tables 3.1 and 3.2 show the execution time for six of the most frequently executed instructions and the eight addressing modes of the PDP-11. The instructions in Table 3.1 assume a register-to-register operation (i.e. a source and destination mode of 0). Table 3.2 shows the additional time that is added to the instruction execution time for the various source addressing modes.* The destination mode times are about the same as the given source mode times.

In order to measure the performance of the CMU-11 for various instruction mixes, several benchmarks were collected and run on the CMU-11, an LSI-11, and a PDP-11/40. Four benchmarks were collected that attempt to span a reasonable range of applications common to minicomputers:

Quicksort. This is a program that uses Hoare's quicksort procedure to sort a set of 16 bit integers. The benchmark also includes a pseudo-random number generator to provide the initial data.

Trigonometric Functions. A set of trigonometric, floating-point routines. We do not assume the existence of a floating point option on any of the processors and hence this benchmark heavily exercises software floating point emulation routines.

Partial Differential Equations. A program that uses a straightforward iterative relaxation technique to solve a partial differential equation over a two-dimensional space. Fixed-point values are used.

Text Searching. Searches an input string for names in a symbol table.

---

* In particular, the times in Table 3.2 are the source addresses modes time for the CMU-11 as measured on the BIS instruction. Addressing times on the other instructions are similar to the BIS times.

This benchmark makes extensive use of the byte and compare features in the instruction set.

Table 3.3 shows the execution times on the LSI-11, CMU-11, and PDP-11/40 for each of the four benchmarks. From these results we see the CMU-11 is approximately twice as fast as the LSI-11 and 63% of the speed of the PDP-11/40. As expected, there is a moderate amount of variation in the relative performance of the three machines for the different benchmarks. The two dominant effects that can be seen in Table 3.3 are that the PDP-11/40 design has optimized register-to-register operations more than either the LSI-11 of the CMU-11 (as demonstrated in the partial differential equation benchmark). Byte operations are more efficiently performed in the CMU-11 because of its byte-swap data path provided by the D and I buses. The last line in Table 3.3 is the data published by O'Laughlin [1975] in an article comparing the different DEC PDP-11 implementations.

It is mildly disappointing that the CMU-11, built with Schottky TTL bit-slices could not equal the performance of the PDP-11/40, built with standard TTL circuits. The next two sections will examine in detail where performance was lost (and gained) in the CMU-11 design. Before continuing with this review of the design, we turn to a brief discussion of the cost of the CMU-11.

A principle objective of the 3000 microcomputer bit-slice packages is to simplify the design of processors like the CMU-11. Table 3.4 is a summary of the complexity (measured in integrated circuits) of the CMU-11. There are two columns in Table 3.3: a simple count of the number of integrated circuit packages used in the CMU-11 and a column that converts the design to "16-pin equivalent" packages a measure of the size of the design in a standard unit). Table 3.6 gives a breakdown of the actual cost of the CMU-11 at January, 1976 prices.

It is surprising that less than 20% of the design is now in the data part of the processor: the part of the processor largely implemented with the LSI bit-slices. A larger part of the design, 29%, is needed just to interface to the PDP-11 Unibus.

In order to put the 144 package complexity of the CMU-11 in perspective, the IC package counts for other PDP-11's are: PDP-11/10 -- 203 packages; PDP-11/40 -- 417; and PDP-11/45 -- 696. The LSI-11 is able to implement the basic processor in 42 packages but does not interface to a Unibus. It is clear that the bit-slices do not approach the economy of the Western Digital NMOS microcomputer circuits which were specifically designed to emulate the PDP-11.

Another measure of the degree to which the CMU-11 processor of how efficiently the CMU-11 microprocessor is able to emulate the PDP-11 architecture is given by the size of the microprograms. Table 3.5 gives the size of microprograms for several PDP-11 processors. It is somewhat surprising that the CMU-11 uses fewer bits in its control store than any of the other processors except the LSI-11. This is in large part due to the fact the 11/10, 11/40, and 11/45 use MSI arithmetic/logic packages that did not have as useful a set of primitive operations as the 3002 ALU.

4. SOME PITFALLS FOUND IN IMPLEMENTING THE PDP-11 WITH THE 3000 BIT-SLICES

Since the CMU-11 project was started, a number of different bit-slice chips have become available whose organizations are significantly different from the 3000 circuits and which provide an interesting contrast. Two of the more interesting bit-slice chips are the Advanced Micro Devices Am2901 and the Monolithic Memories Inc. MMI6701. These bit-slice chips have a very similar data path organization with only minor differences, the Am2901 being the faster device. Because of the similarity of these devices, we will limit the discussion here to the Am2901, but all of the microinstruction sequences discussed will work on both bit-slice sets.

The basic data path of the Am2901 is shown in Figure 4.1. The chip contains a register file of 16 4-bit accumulators and an accumulator extension register, the Q register. In one microinstruction, two operands can be read out of the register file, passed through the ALU, ~~and~~ the result ~~can be written~~ shifted left or right, and written back into the register file. In parallel with this, there is an addressing mode which controls the RAM and Q shifters allowing the output of the ALU and the Q register to be right shifted simultaneously, which is well suited for the inner loop of multiply or divide instructions.

4.1 I/O Buses

The main advantage of the 3000 bit-slice over the Am2901 is its five fully parallel data buses for transferring data in and out of the chip. It has two tri-state output buses (the A and D buses) and three input buses (M, I, and K). If the minicomputer to be emulated has a fairly short I/O and memory buses, the 3000 buses can directly be used to drive them, resulting in a

substantial savings in bus driver packages. In the CMU-11, we needed to drive a DEC Unibus, so we had to use separate bus drivers and receivers. Once external bus drivers are added, the advantage of the two output buses for the address and data is minimal, because an equivalent external address register can be loaded as fast as the ~~existing~~ internal address register in the 3000 and combination bus drivers/latches are available (e.g. Am2905). The savings realized by having three input buses is the cost of adding eight dual 4-to-1 line multiplexer chips at the input to the bit-slice chips. The ~~total~~ saving achieved with the five buses in the 3000 bit-slices over the ~~AMD~~2901's single input and single output bus is then 12 16-pin circuits, plus three bits in the control store (two for the select lines on the input multiplexer, and one to control loading of the address register).

4.2 Arithmetic Overflow with the 3000

One of the biggest problems encountered with the PDP-11 implementation using the 3000 bit-slice was detection of arithmetic overflow. The 3000 bit-slice has no overflow output and the signals needed to directly detect overflow are not available at the external pin connections. This results in considerable overhead in emulating instructions which must detect overflow (e.g. instructions that set the V bit in the PS register of the PDP-11). The CMU-11 overflow handling was implemented with two external flip-flops which contain the signs of the source and destination operands. After an instruction is fetched its operands are first fetched either from memory or from the register stack, and are put in the source and destination registers within the 3002. As the operands are fetched, the source and destination flip-flops are set to the signs of the operands. When an instruction

is executed the overflow logic can use the signs of the operands and result to detect overflow. This technique works well when the operands are from memory, but really slows down the register-to-register operations because the operands have to be moved to the AC so their signs can be latched in the external source and destination sign flip-flops.

The sequence of instructions needed to emulate a register-to-register ADD add is shown in Figure 4.2. The first instruction in the sequence loads the source operand into the AC, in order to get its sign out of the chip. The next instruction specifies for the source sign flip-flop to be set to the sign of the AC, and to store the AC into the T register. The following two instructions load the destination operand into the AC and set the destination sign flip-flop. The last two instructions do the add and store the result back in the destination register. Because of the multiple use of fields in the microinstruction it is not possible to specify that a register address comes from the instruction register in the same microinstruction that sets either the source sign or destination sign flip-flops, or which sets the condition codes. If the microprocessor were to be redesigned to allow this, the register-to-register add could be done in three rather than six microinstructions with the 3000 chips. However, we would pay for this performance improvement by having to use a wider microinstruction.

Because the Am2901 ~~provides external access to the~~ has an overflow detect output on the chip ~~and~~, the register-to-register add can be done with only one microinstruction, resulting in a considerable speed increase over the 3000 ~~chip~~ chips.

4.3 Example of a Multiply Instruction

The inner loop of a ~~16~~ 16x16 bit integer multiply instruction on the 3000 chips requires either three or six microinstructions, depending on whether that cycle is a double register shift and add, or just a shift. The high order word of the product is stored in the AC register, and the low order word is stored in the T register. Initially, AC is zero, and T holds the multiplicand. For each iteration of the multiply, the loop count is decremented and if the low order bit of the T register is a 1, then the multiplier is added into the AC and the AC and T registers are shifted right. Because the 3000 cannot add a register to the AC without also putting the result back in the register, it takes three microinstructions to perform the inner loop addition.

For the Am2901, the inner loop of the multiply can be done in two microinstructions with no external loop counter, and in one with an external counter. This is possible because the Am2901 in one microinstruction ~~can~~ can add two general registers together, shifting the result and the accumulator extension register right one bit. A similar speedup also occurs for division.

5. ADDITIONAL COMMENTS ON THE CMU-11 DESIGN

The 3000 microcomputer circuits are not the only area in which to look for improvements in the CMU-11 design. A major source of complexity was the Unibus interface (29% of processor's packages). The 3002 bit-slices provide tri-state drivers for their A and D lines and if Unibus compatibility is not essential, the outputs from the 3002 circuits could directly drive a memory and I/O bus of moderate size. If synchronous operation of the memory bus is adequate, further simplification of the bus interface section of the processor is possible.

A number of integrated circuit packages are now available that could help simplify the design of the control part of the processor. Most significantly, 4K bit PROM's appropriate for use in the control store are now available with internal latches for use as a microinstruction buffer. This would eliminate the need for the separate latches used in the CMU-11's microinstruction register. A related optimization to the CMU-11 would be to move from the partly encoded microinstruction format of the CMU-11 to a wider, fully horizontal format. The random logic needed to decode an encoded microinstruction is simply more expensive than the extra bits in the control store needed for the horizontal format.

We attempted to use programmable logic arrays (PLA's) in our initial design, but converted to ROM's when the PLA's we were designing with were discontinued. By now, however, several useful PLA's are readily available. For example, the Signetics FPLA, with its 16 inputs, is well suited to the decoding of PDP-11 instructions.

Below are the gains that might be expected in a second iteration of the CMU-11 design:

CMU-11	160 IC packages (16 pin equivalent)
Non-Unibus Design	128
Integrated ROM/MIR and horizontal micro-instruction format	113
Convert to Am2900 circuits	95

~~It should be noted that conversion to the Am2900 circuits~~

new section - not in original

It should be noted that the savings of 18-16 pin equalivent packages indicated by converting from the 3000 to the Am2901 circuits comes from two main factors: going from a 2 to a 4 bit slice in the data path, and the simplication of the logic needed to set the condition code bits in the program status register (the N, C, V and Z bits.) The setting of the the condition code bits is simplified because the Am2900 directly gives the overflow output, the carry output, zero detect, and the sign of the result, all during the current microinstruction. In the 3000, the external logic must generate the overflow output itself, and must wait until the next microinstruction to get the sign of the result, and to detect zero, resulting in extra logic to handle the resulting timing problems. It should also be noted that the real reason for converting from the 3000 to the Am2901 is not ~~just~~ the logic savings, but is mainly the speed increase the results from the more powerful microinstructions in the Am2901.

6. COMPUTER-AIDED DESIGN TOOLS

Aside from freeing the designer of bookkeeping and clerical tasks, the main advantage of any design automation system is its inherent ability to maintain correct and consistent documentation (prints and wirelists) and the reduced turnaround time for design iterations. The fact that the total prototype development time for the CMU-11 was 39 (40 hours) man-weeks is an example of the savings possible with even modest design automation aids.

6.1 Description of Facilities Used at CMU

The Stanford University Drawing System was used to enter the schematic print set with a graphics display terminal. The drawing package includes a set of satellite programs to extract information for wirelists and cross-reference tables from its data base. Incorporated in the system are libraries of integrated circuit definitions which contain not only the pictorial representation of the gates but also pin section information and some loading data. Hard copy prints were conveniently generated by an XGP (digitally controlled Xerox Graphic Printer). The wirelist program can search the data base interactively for specific information or produce complete tables of run lists, stuff lists, error reports (wire-anding violations), and loading analyses, which all proved extremely helpful.

The logic simulator used was SAGE (Simulation of Asynchronous Gate Elements), which is a four-state {0, 1, high impedance [tri-state buses], and undefined (initialization and uncertainty in delay parameters)} gate-level simulator. It reads the data base directly from the output of the Stanford Drawing system. This proved to be of utmost convenience, since it allowed a turnaround time in the order of five minutes for print set corrections.

SAGE has models in its libraries for the TTL and Schottky families and special routines were written by us to emulate the 3000 microcomputer set. This allowed improvements in the efficiency of the simulation execution. Macro facilities are also available for quickly defining MSI circuits from more basic logic gates. The results of the simulations are in the form of register and signal reports and timing/trace diagrams.

6.2 Debugging with the Simulator

About 95% of the original design errors were eliminated through the use of the simulation program. Naturally, not all combinations and sequences of instructions can be simulated, but a standard PDP-11 diagnostic program was run in addition to a number of other programs. A total of about 100 milliseconds' worth of CMU-11 compute time was simulated before debugging on the actual hardware began.

The limitation here was that the SAGE simulation of the CMU-11 required about 10^6 seconds of CPU time on a PDP-10 to simulate 1 second of CMU-11 execution. We simply could not afford to consume more than about 30 hours of CPU time for this project.

Whatever amount of time is spent on simulation, the simulations cannot be exhaustive and the final set of errors must be tracked down with more extensive tests on the real machine. We discovered eight to ten errors in the actual CMU-11. However, when an error was found in the physical machine, the simulations were again run to help track down the bug through the use of timing traces and other results. The correction was then entered into the machine print set and the simulator was re-run before implementing the change on the processor wire-wrap board or in the microprogram.

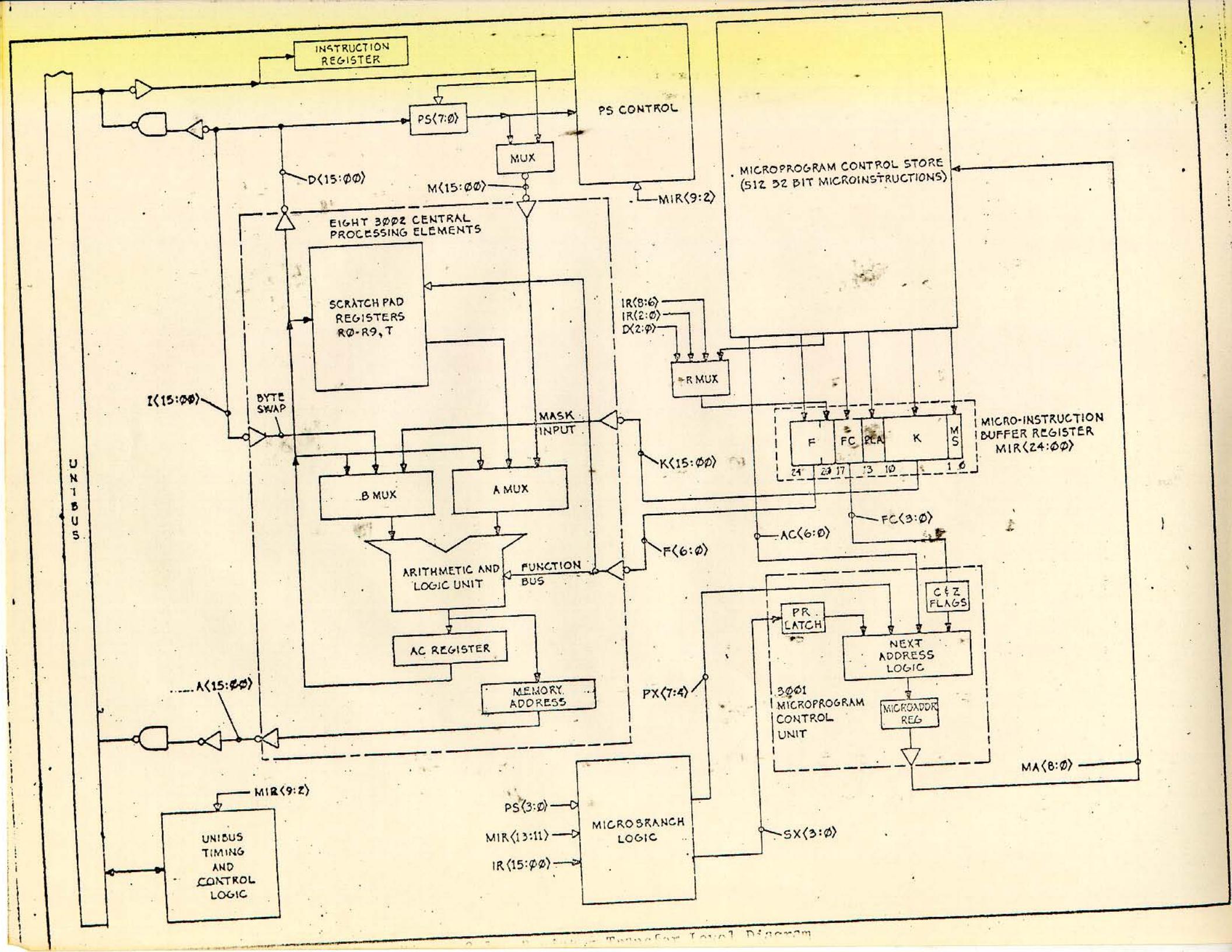
An example of the worth of the computer aided design system was when a major implementation change was made when several ROM's were incorporated into the design to replace a discontinued PLA (programmed logic array). Our design aids were essential in effecting this change within four man-days. In order to recover so quickly from such a massive wiring change, an ECO wrap/unwrap program was run using the old and new wirelists produced by the drawing package. Thus, at all times during development, the processor reflected the exact connectivity of the print set.

Several of the errors discovered on the real machine were timing errors that were not caught in the simulation debugging. These errors were not detected because the simulation models did not consider the effects of loading on the propogation delays and only maximum delays in all gates were used as an approximation to worst case conditions. In fact, if time had permitted, minimum and typical (Gaussian distributed) parameters should also have been tested. However, we again face a fundamental problem with simulation in that the computation time becomes excessive as different sets of delays are simulated to find worst case conditions.

7. CONCLUDING COMMENTS

The CMU-11 project was initiated as an experiment in constructing general purpose (mini) processors with LSI bit-slice components. Table 7.1 is a summary of the results. As the table shows, the CMU-11 was implemented with significantly less components (IC packages) than either the PDP-11/10 or the PDP-11/40, which are processors built with MSI components, and the performance of the CMU-11 falls between these two MSI processors. However, the economy of implementation is not nearly as significant as was realized with the LSI-11 although the CMU-11 is able to perform at twice the speed of the LSI-11 The LSI-11 is a processor implemented with NMOS LSI microcomputer packages in which the entire data path (with 8 bit data paths) was put in a single package and both the control and data packages for the LSI-11 have been specialized to efficiently emulate the PDP-11 architecture.

In Sections 5 and 6 we discussed improvements that are possible in the CMU-11 design and argue that a second iteration on the design could boost the performance to that of the PDP-11/40 and could be implemented in about 95 rather than 144 packages. To achieve a more cost effective design than this will require either the development of some LSI control circuits specific to the processor's instruction set or will require the specification of a new computer architecture tailored to make the most efficient use of the functions provided in the LSI circuits.



Transfer Level Diagram

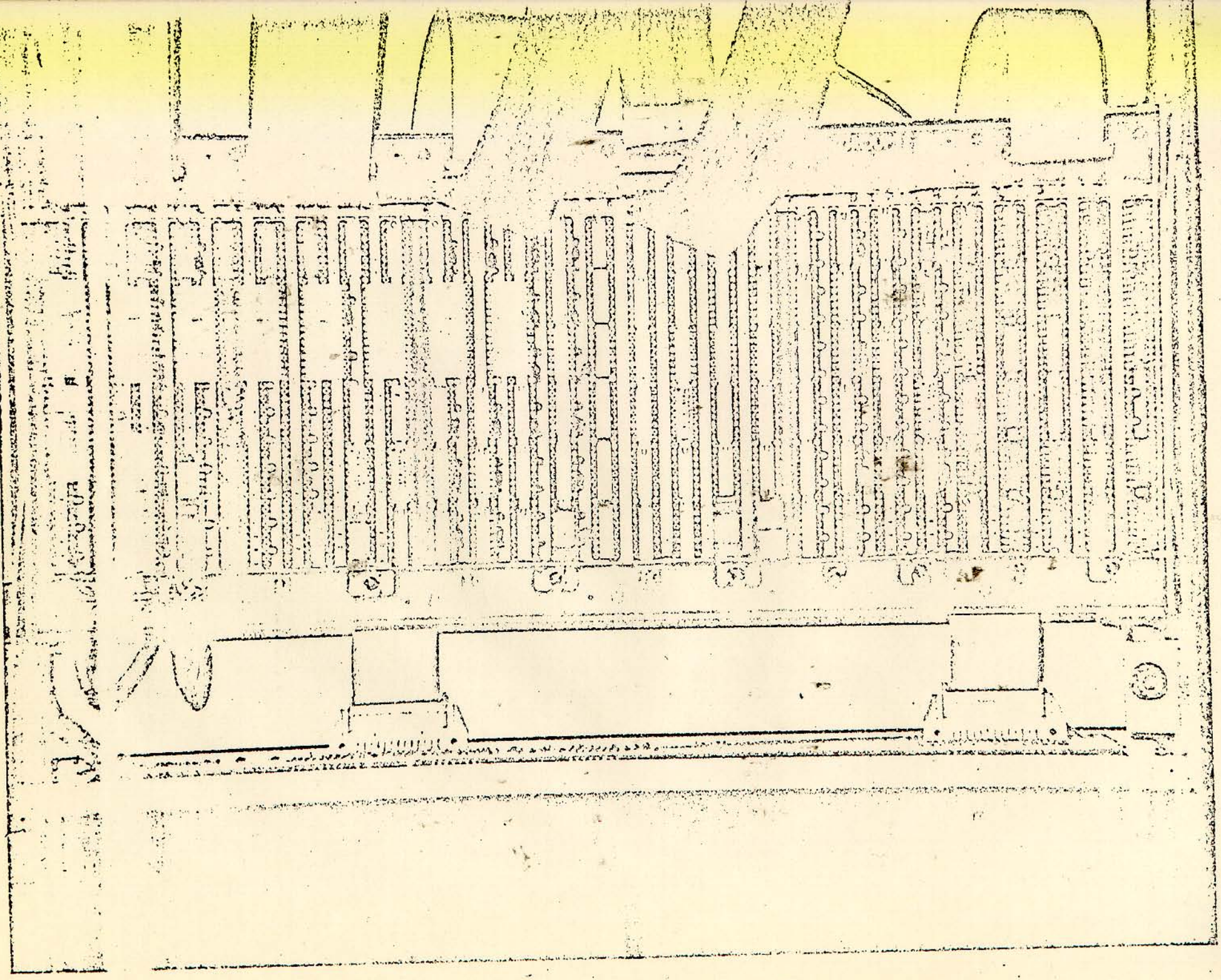


Figure 2.2. CMU-11 Processor Board

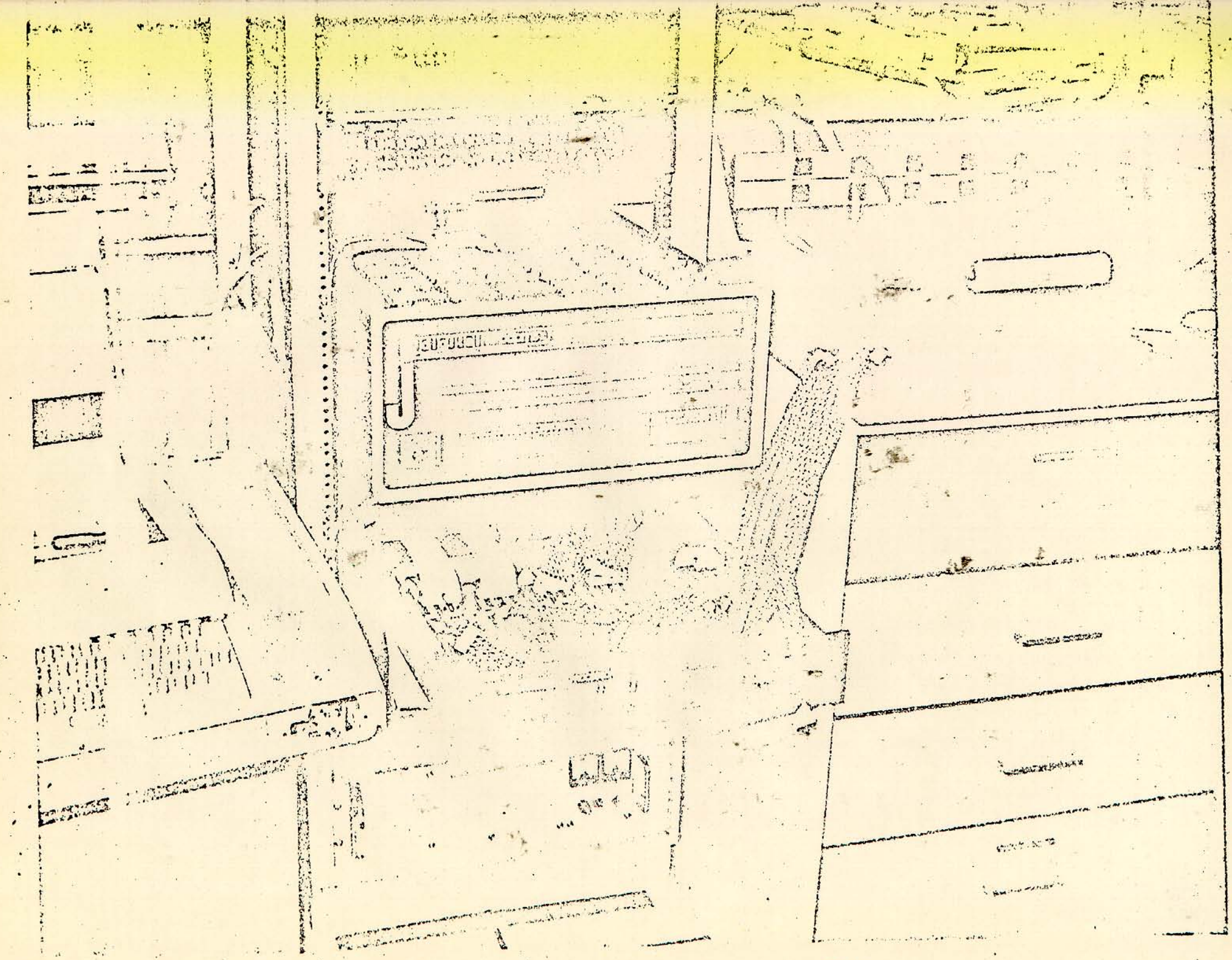
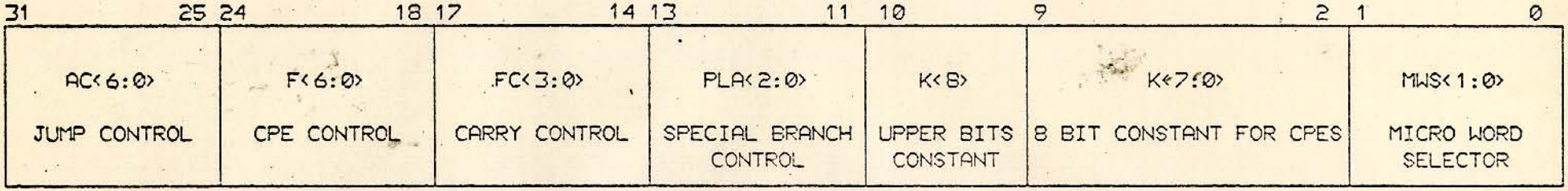
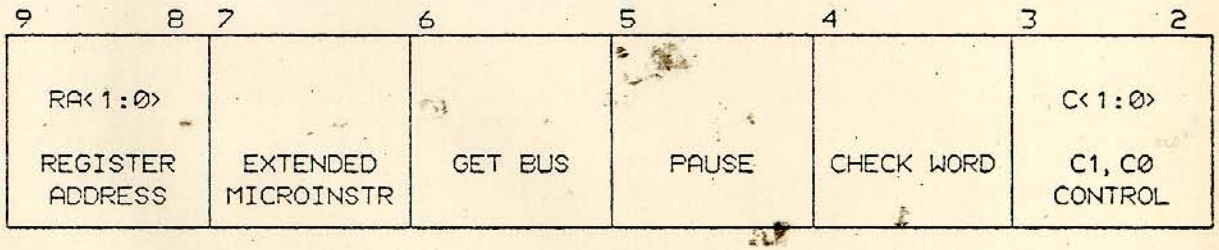


Figure 2.3. CMU-11 System with Associated PDP-11 to



UC<7:0>, UNIBUS CONTROL:



PS<7:0>, PS LOGIC CONTROL:

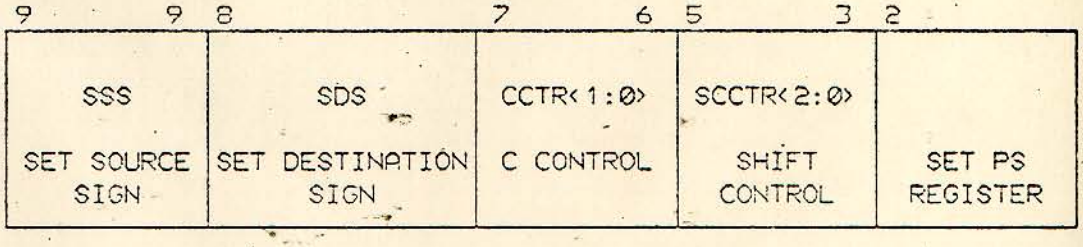


Figure 2.4. Microinstruction Format

TABLE 2.1. DESCRIPTION OF MICROINSTRUCTION FIELDS

MWS<1:0> := MI<1:0>	Micro Instruction Selector. Specifies if MI<9:2> should define a constant, unibus control, or PS control.
K<8:0> := MI<10:2>	Literal. K<7:0> is a byte constant used by the least significant byte of the K input lines of the 3002 array. K<8> is extended to feed the most significant byte of the K input lines.
UC<7:0> := MI<9:2>	Unibus Control
UC<1:0>	C1, C0 Control. Specified the C1 and C0 lines on the Unibus.
UC<2>	Check Word. Tests whether a word address is specified in Unibus operation.
UC<3>	Pause. Halt processor clock until completion of Unibus operation.
UC<4>	Get Bus. Request access of Unibus for a data transfer.
UC<5>	Extended Micro Instruction Code. If set, defines alternate meaning for PLA<2:0>.
UC<7:6>	Register Address. Specified which input register address multiplexor should be used.
PS<7:0> := MI<9:2>	Processor Status Control
PS<0>	Set PS Register. Controls loading of PS.
PS<3:1>	Shift Control
PS<5:4>	Carry Control
PS<6>	Set Destination Sign. Controls latching of sign of destination operand in flag external to 3002's.
PS<7>	Set Source Sign. Analogous to PS<6>.
PLA<2:0> := MI<13:11>	Special Branch Control. Used by microbranch logic to tell which fields of IR and PS to examine for branch conditions.
FC<3:0> := MI<17:14>	MCU Flag Control. Controls testing and setting of flags in 3001 (MCU).

F<6:0> := MI<24:18>

CPE Control. Drives Function Bus of 3002 (CPE) array.

AC<6:0> := MI<31:25>

Address Control. Connected directly to the AC<6:0> bus of the 3001 (MCU). This is the one field of the micro instruction not buffered in the micro instruction register. (The Microprogram Address Register internal to the MCU performs the buffering function.)

Instruction	Basic Execution Time (microseconds)		
	LSI-11	CMU-11	PDP-11/40
MOV	3.50	2.06	0.90
CMP	3.50	2.19	0.99
ASL	3.85	2.46	0.99
ADD	2.46	3.85	0.99
BRX (branch)	3.50	2.82	1.76
(no branch)	3.50	1.48	1.40
JSR	6.40	4.39	2.94

Table 3.1. Execution Times of Common Instructions

ADDRESSING MODE	LSI-11	CMU-11	PDP-11/40
0: Register	0.00 µsec	0.00 µsec	0.00 µsec
1: Register Deferred	1.40	1.21	0.78
2: Autoincrement	1.40	0.64	0.84
3: Autoincrement Deferred	3.50	1.91	1.74
4: Autodecrement	2.10	1.00	0.84
5: Autodecrement Deferred	4.20	2.28	1.74
6: Indexed	4.20	1.78	1.46
7: Indexed Deferred	6.30	2.99	2.36

Table 3.2. Execution Times for the Source Addressing Modes

Benchmarks	Execution Times Relative to PDP-11/40*					
	LSI-11	11/10	11/20	CMU-11	11/40	11/45
Quicksort	2.88 (366)			1.48 (188)	1.0 (127)	
Partial Diff. Eqn.	3.48 (268)			1.75 (135)	1.0 (77)	
Trig. Functions	3.36 (111)			1.57 (52)	1.0 (33)	
Text Searching	2.76 (204)			1.45 (107)	1.0 (74)	
Average	3.2	—	—	1.6	1.0	—
O'Loughlin's Data	—	2.32	1.85	—	1.0	0.91

Table 3.3. Performance of CMU-11 Relative to Other PDP-11's

*Numbers in parentheses are the absolute run times in seconds for the benchmarks.

Processor Component	No. IC Packages	No. 16 pin Equivalent Packages
DATA PART		
3002 (CPE) Array	8	20
PS and Instruction Registers	6	6
Misc.	4	5
subtotal	18	31 (19%)
CONTROL PART		
Control Store ROMs	8	8
Micro Instruction Register	10	10
3001 (MCU)	1	3
Microbranch logic	26	27
PS Control	16	16
Misc.	18	18
subtotal	79	82 (52%)
UNIBUS INTERFACE		
Bus Tranceivers and Inverters	19	19
Unibus Control	28	28
subtotal	47	47 (29%)
TOTAL	144	160

Table 3.4. Integrated Circuit Statistics

Components	Prices	
	Single Units	Quantities of 100+
LSI Microcomputer parts (Intel 3001,3002's,3003)	$207 (184)*	$125
PROMS (3601,3602,3604,745168)	204	136
SSI/MSI Parts	179	158
Integrated Circuit Subtotal	590	419
Augut Wirewrap Board	379	(use printed circuit)
Wirewrapping	107	—
TOTAL	$1076	—

Table 3.5. Cost Breakdown for CMU-11

*Signetics prices

LSI-11	PDP-11/10*	CMU-11	PDP-11/40*	PDP-11/45*
22 bits x 512 words (includes console)	40 bits x 249 words	32 bits x 287 words (without console) 414 words (with console)	56 bits x 251 words	64 bits x 256 words

Table 3.6. PDP-11 Control Store Sizes

*[O'Loughlin 1975]

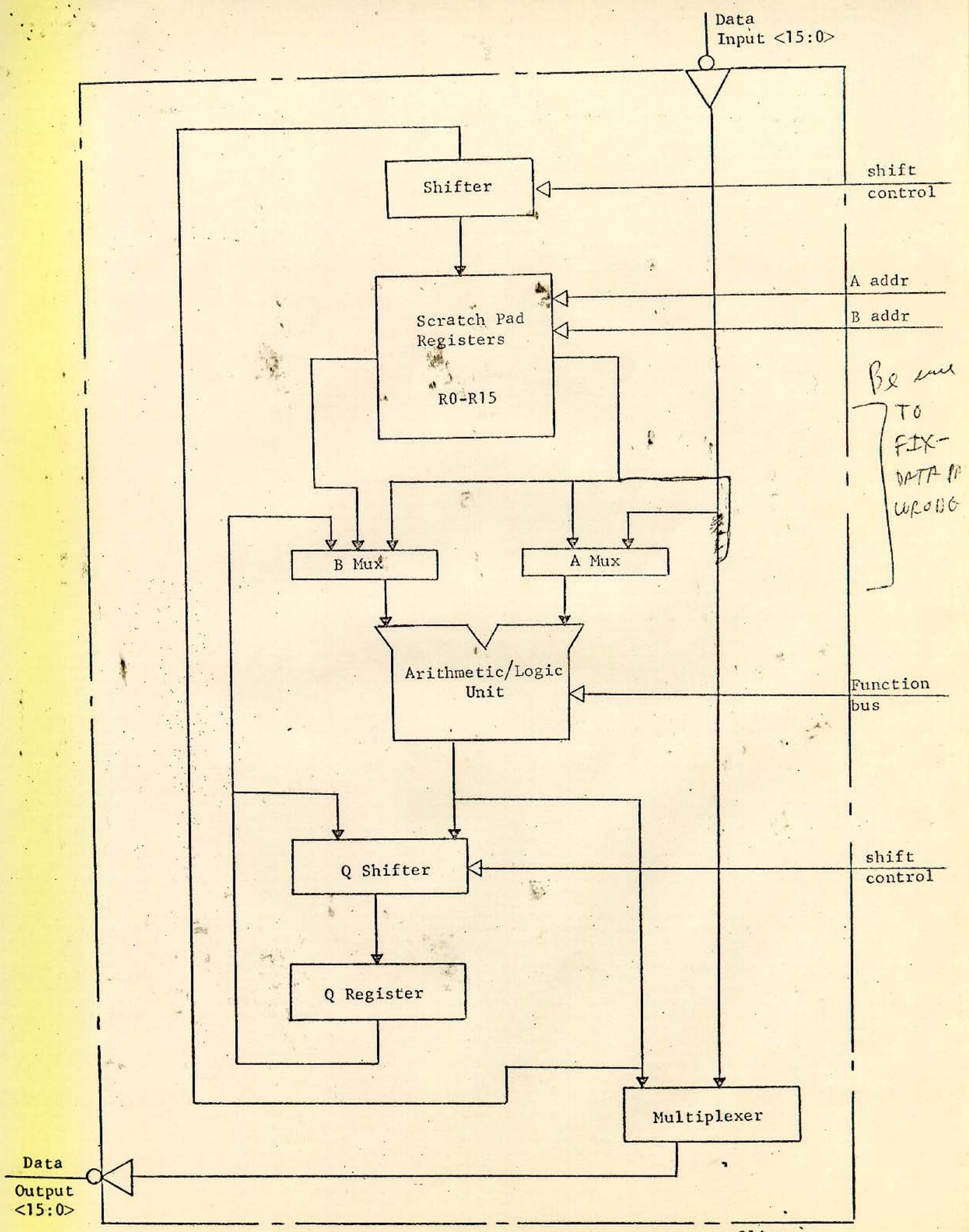


Figure 4.1. The Am2901 - a 4 bit Bipolar Microprocessor Slice.

```
ILR   SR              ;AC ← Source Register
SDR   T,1, SETSS      ;T ← AC and SET Source Sign
ILR   DR              ;AC ← Destination Register
NOP   SET DS          ;SET Destination Sign
ALR   T, SETCC        ;AC = AC + T and SET Condition Codes
SDR   DR,1            ;Destination Register ← AC
```

Figure 4.2. Microsequence Example: Register-to-Register Add with Overflow Detect

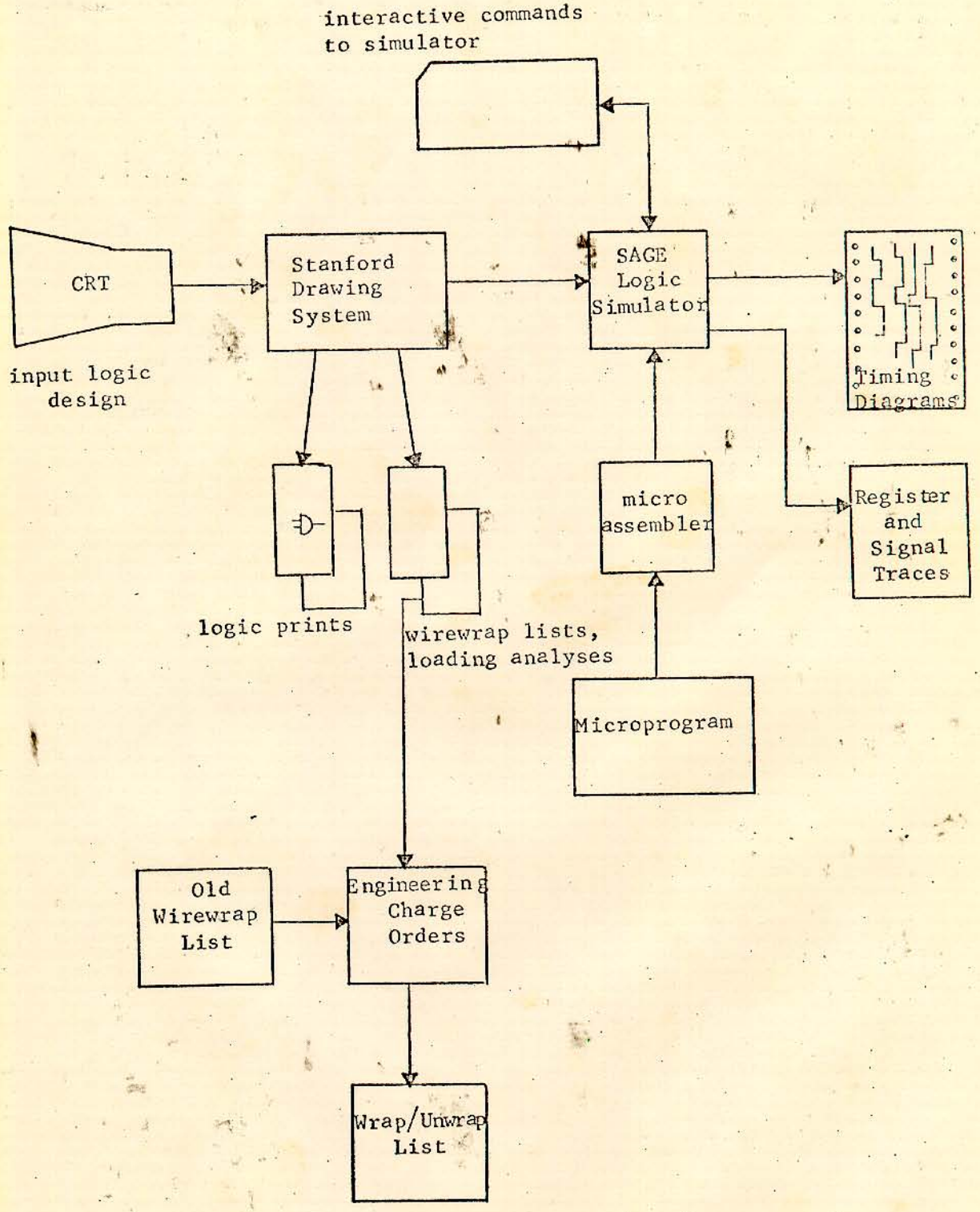


Figure 6.1. CAD System at CMU

Parameter	LSI-11	PDP-11/10	CMU-11	PDP-11/40
Microcycle time (nsec)	400		200	140,200,300
Relative Execution Times	3.2	2.32	1.6	1.0
IC Packages	42	203	144	417
Control Store Size (bits)	11264	9960	9184	14056

Table 7.1. Summary of Comparison between CMU-11 and Other PDP-11 Implementations

References

[AMD 75] Am2900 Bipolar Microprocessor Circuits, Advanced Micro Devices, Inc., Sunnyvale, California, 1975.

[CMU 76] Fuller, S. H., T. McWilliams, and W. Sherwood, CMU-11 Engineering Documentation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 1976.

[DEC 73A] PDP-11/05/10/35/40 Processor Handbook, Digital Equipment Corporation, Maynard, Mass., 1973.

[DEC 73B] PDP-11 Peripherals Handbook, Digital Equipment Corporation, Maynard, Mass., 1973.

[DEC 75] LSI-11, PDP-11/03 Processor Handbook, Digital Equipment Corporation, Maynard, Mass., 1975.

[Intel 75] Intel Schottky Bipolar LSI Microcomputer Set: 3001 Microprogram Control Unit, 3002 Control Progressive, Element, and 3003 Carry Look-ahead Generator, Intel Corporation, Santa Clara, California, 1975.

[O'Loughlin 75] O'Loughlin, J. F., Microprogramming a Fixed Architecture Machine, Infotech State of the Art Report 23, Infotech Information Limited, Maidenhead, England, 1975, 205-224.

[Signetics 75] Introducing the Series 3000 Bipolar Microprocessor, Signetics Corporation, Sunnyvale, California, 1975.