Engineer. In addition, all sections can be looped via another switch on the console. The implementation of microorders to sense switches is simple in comparison to the technique used in conventional diagnostics. Switches on the console are also labeled as an aid for the CE. Conventional diagnostics on the other hand, require special instructions in order to sense switches on the console. The sensing of these switches is rather lengthy in terms of execution time, therefore, it is not normally done during scoping loops.

**ROS simulation**

Writable Control Storage in System/360 Model 85 has been used to test resident diagnostics implemented in Read Only Storage (ROS). This procedure has proved to be a very effective and efficient way of implementing code found in Read Only Storage because turnaround time for manufacture of bit planes necessary for ROS is relatively long in comparison to that for assemblies used for WCS. In addition to the improvement in speed and flexibility, significant cost savings have been realized by eliminating unnecessary manufacture of incorrect bit planes. Since WCS can be expanded to the same size as Read Only Storage it can be used to check the basic machine set as well as the resident microdiagnostics.

**LIMITATIONS OF MICRODIAGNOSTICS**

Limitations in implementing System/360 Model 85 microdiagnostics fall into one of five main categories:

- First - it is impractical to implement functional testing in microcode. Functional tests imply that the functional design of the machine is tested. If we substitute WCS code for the functional ROS code, we would not be testing the design of the machine.
- Second - limited access is available to the I unit and the Storage Control Unit. This is understandable because microcoding was used in the design of the System/360 Model 85 primarily for the control of the execution unit. The I unit and the SCU are primarily controlled by hardware sequencing.
- Third - coding inefficiencies for large data handling are prevalent because of the lack of power of micro-instructions.
- Fourth - interactive problems (timing) require a considerable amount of test setup which is more efficiently done in conventional code. In addition, the interactive problems of prime concern are those which exist between instructions of the basic instruction set (functional).
- Fifth - skill level - because microprogramming is such a low level language, detailed knowledge of the hardware is required before any attempt can be made to program the machine.

**SOFTWARE SUPPORT OF MICRODIAGNOSTICS**

The system which was initially adopted for System/360 Model 85 support is the Controlled Automated System (CAS). This system yields a flow chart of microinstructions being executed. It is designed for use by engineers for implementation of their code in Read Only Storage and has proved to be an asset in development of design. This system, however, does not provide flexibility needed for the programming environment of microdiagnostics. Because of this lack of flexibility an interim assembler was developed for the support of the microdiagnostics debug. This program provides a listing format. Utilities are available to permit changes in program decks and a loader which can be used to load these decks into WCS for debugging purposes.
Use of read only memory in ILLIAC IV*

by H. J. WHITE and E. K. C. YU
Burroughs Corporation
Paoli, Pennsylvania

INTRODUCTION

Because of its high speed operation, large instruction repertoire and centralized control, the ILLIAC IV Computer uses a Read Only Memory (ROM) to translate instructions into control enables. These control signals are broadcast to the array of parallel processors to control the step by step operation of each processor. Each of the over 260 instructions is decoded into a microsequence (microprogram) used to address the ROM. Each microsequence consists of from one to 69 microsteps (microinstructions).

The ROM is a transistor cross point matrix and is configured with discrete transistors mounted on large multilayer circuit boards. The memory size is 720 words (microsteps) by 280 bits (control enables). Cycle time is 50 nsec.

To simplify instruction decoding, up to five words are simultaneously addressed in many microsequences, i.e., control enable ORing. This ORing also improves memory speed and reliability by greatly reducing the number of transistors required. To save execution time, up to two instructions are simultaneously addressed, i.e., instruction overlap. Thus, up to ten memory words could be simultaneously addressed. Because the Read Only Memory uses linear addressing, the simultaneous addressing of any number of words is easily achieved.

THE ILLIAC IV SYSTEM

By using extensive parallel processing, the ILLIAC IV System (1) offers computing power capable of solving a number of problems beyond the power of currently available or proposed computers. Some problems involve manipulations of very large matrices (e.g., linear programming), others involve the solution of sets of partial differential equations over large grids (e.g., numerical weather prediction); and still others require extremely fast correlation techniques (e.g., phased array radar).

As shown in Figure 1, the ILLIAC IV System consists of a large parallel-array computer coupled to an I/O subsystem. The I/O section contains

1. A Burroughs B6500 Computer which functions as the executive element.
2. Two Burroughs disk files. (Each file is capable of up to 16 parallel disks. Each disk contains about $79 \times 10^6$ bits of storage. The effective bit transfer rate of each disk file is over $500 \times 10^6$ bits per second. Both files may operate concurrently for a net maximum transfer rate of $10^9$ bits per second.
3. An I/O Controller, buffer memory and switch for interfacing with the computer.

The parallel array computer consists of four identical array processors or quadrants. Figure 2 illustrates the quadrant diagram. Each quadrant consists of a Control Unit (CU) and 64 Processing Units (PU's). Each PU, in turn, contains a 10,000 gate arithmetic unit called the Processing Element (PE) (2), a 2048 word by 64 bit semiconductor memory called the Processing Element Memory (PEM) and a 1000 gate interface unit called the Memory Logic Unit. All the controls normally associated with an arithmetic unit are extracted from the PE's and placed in the CU. These controls are shared in parallel, by all 64 PE's in the quadrant.

Both PE data and CU instructions are contained in the array of 64 PEM's which serves as the main memory for the quadrant. The CU has access to the entire PEM array, while each PE can only directly reference its own PEM.

The following data give some idea of the size and complexity of the ILLIAC IV Computer (not including the I/O subsystem). Each quadrant is 53 feet long, 6.5 feet deep and 8 feet high. Each quadrant contains about

---

*Project supported by Advanced Research Projects Agency under Contract No. AF30(602)4144 as administered by University of Illinois, Urbana, Illinois.

197
850K, specially developed, high speed, emitter-coupled-logic (ECL) gates in a total of about 240K dual-in-line packages. Each quadrant requires about 210 KW of input power. Each quadrant contains 128 K words of storage at 64 bits per word. Memory access time is 188 nsec and cycle time is 200 nsec. The entire system operates at a 20 mHz clock rate. It performs a 64 bit floating point add in 2.50 nsec, a 64 bit floating point multiply in 450 nsec and a 64 bit floating point divide in 2800 nsec.

INSTRUCTION FLOW THROUGH THE CONTROL UNIT

The primary functions of the CU are:

1. Control and decode instruction streams.
2. Generate the control signals transmitted to the processing elements for instruction execution.
3. Generate and broadcast data words and those components of memory address that are common to all PE's.
4. Receive and process traps signals arising from arithmetic faults in the PE's or operations in the I/O subsystem.

The major parts of the CU, as shown in Figure 3, are:

1. ILA—Instruction Look Ahead, whose function is to fetch and store large blocks of contiguous code. It fetches 8 word blocks having two instructions per word. It stores up to 8 blocks, i.e., 128 instructions.
2. ADVAST—Advanced Station, is the principal housekeeper of the system wherein such functions as address arithmetic, loop control, interrupt processing and configuration control are performed. It receives and processes instructions from ILA.
3. FINST—Final Station has the function of directly controlling the operation of the PE's in the array.
4. MSU—Memory Service Unit resolves conflicts among all users requesting access to the array memory.
5. TMU—Test and Maintenance Unit.
From a programming standpoint, FINST and the PE's perform "inner-loops" of a program and ADVAST performs "outer-loops" and control functions. When ADVAST receives a PE instruction, it performs the necessary indexing (if required) and then sends it directly to FINST for processing. Instructions enter FINST thru a first-in-first-out queue (FINQ) which contains up to 8 instructions. The queue decouples FINST execution time from ADVAST execution time and permits instruction overlap between the two stations. Any synchronism during overlap is handled by hardware.

Using a Read Only Memory (ROM), FINST decodes instructions into control enables which are then broadcast to the array of 64 PE's. The control enables control information flow both in direction (register to register) and time. The ROM is a transistor crosspoint matrix.

In FINST two instructions are examined concurrently for purposes of instruction overlap, e.g., to see if data fetches for the next instruction can take place during the end of the present instruction. Instructions first appear in an overlap section from which preliminary commands are generated and then appear in an instruction section in which the remainder of the instruction is completed. For many instructions, two or more words are simultaneously addressed, i.e., control enable ORing. This ORing simplifies branching within the instruction and also improves memory speed and reliability by greatly reducing the number of transistors required. Thus, up to ten words are simultaneously addressed.

REASONS FOR USING A READ ONLY MEMORY

The ILLIAC IV instruction repertoire consists of over 260 instructions. This large number of instructions provides for great system flexibility.

For example, there are six kinds of signed and unsigned arithmetic:

1. Normalized floating point.
2. Normalized floating point rounded.
3. Unnormalized floating point.
4. Unnormalized floating point rounded.
5. Mantissa-sized fixed point.
6. Mantissa-sized fixed point rounded.

All data operations can be performed in 64-bit or 32-bit mode. Also, there is a limited amount of unsigned full word (64-bit) and 8-bit arithmetic. Thus, the system can be operated in an 8-, 32- or 64-bit mode which, for a 256 PE system, gives the capability of 512 32-bit processors or 2048 8-bit processors.

In FINST, the 260 instructions are decoded into 280 control enables of which about 260 are broadcast to the PE's with the remaining enables controlling areas in the CU. The reasons for the large number of PE enables are:

1. The large variety of arithmetic modes.
2. Centralizing all processing controls in the Control Unit.
3. High speed operation, i.e., control enables are changed every clock cycle (50 nsec).

Thus, the primary functions of FINST is to decode a large number of instructions (260) into a large number of enables (280). Two methods of decoding instructions were investigated: (1) a fully hard wired logic approach and (2) the use of an ROM. In the logic approach, all instructions generating the same control enable are essentially OR'd together. To meet the system speed requirements, all instructions generating the same control enable, should be on the same multilayer P.C. board. This leads to considerable duplication because the number of instructions generating a control enable may be so large that only a few of the complete set of enables can be on the same board. In this approach each board had an identical instruction register, timing counter and other basic logic required to decode instructions. A minimum of 60 boards was estimated to be required and the most likely number was about 80. Each board was unique which presented a serious board sparing problem. In the ROM approach, which was adopted, a total of 28 boards is required. There are eight word driver boards which generate the micro-sequences required to address the memory. Each of these boards is unique. There are 12 matrix boards which make up the memory. Each matrix board is identical except for the location of cross point transistors. There are eight sense amplifier boards. Although only one type of board is required, two types of boards were used as a convenience. Thus, the ROM approach occupies less than half the volume, had fewer board types and uses less than about 1/4 the number of components. In addition, the logical design is simpler in the ROM approach and changes are more easily implemented. Although an MSI or LSI ROM might be cheaper and faster, a discrete component ROM design was adopted for the following reasons:

1. A suitable, L.C., ROM would not be available in time (by first quarter of 1970).
2. Changes are easily made at any time in a discrete memory.
3. Linear addressing is easily accomplished in a discrete memory. Linear addressing, i.e., one address line per word, simplifies the simultaneous addressing.
required to achieve instruction overlap and control enable ORing.

INSTRUCTION OVERLAP

As stated earlier up to ten ROM words are simultaneously addressed as PE instructions are processed in FINST. For any instruction, up to five words are addressed to achieve control enable ORing. In addition, to realize the time saving of instruction overlap, up to two instructions are decoded at the same time.

Figure 4, illustrates how instruction overlap is accomplished. As instructions are received from ADVAST they are stacked in an eight word queue, FINQ. When an instruction is in position 1, it is transferred to the FINST Overlap Register (FOR). In FOR two actions take place. First, the instruction is examined to determine what parts of the Processing Elements will be used when the instruction is transferred to the FINST Instruction Register (FIR). This information is stored in the 1st level of a Busy Register. The Busy Register also contains the same PE usage information, in level 2, for the next instruction to be executed, which is in FIR, and contains like information, in level 3, for the instruction presently being executed, which is in FIAR (the ROM address register associated with FIR). The instruction in FOR is also examined to see if it is a candidate for instruction overlap. If it is, it is decoded into a microsequence which addresses the overlap section of the ROM (250 addresses) using the ROM address register FOAR. If the required sections of the PE are busy, the microsequence is inhibited until the PE sections are free.

FIAR addresses a 470 word section of the ROM which is sufficient to execute all instructions. When the instruction in FIAR is fully executed, the instruction in FIR is transferred to FIAR and the instruction pointers to the queue are incremented one position. This shifts the instruction from FOR to FIR and puts a new instruction in FOR. At the same time, the Busy Register is updated to determine if instruction overlap is possible.

To achieve the required high speed operation, the PE busy bits in word one are set while the instruction is in FOR. As the instruction moves from FOR to FIR to FIAR, the busy bits are transferred from level 1 to level 2 to level 3. Busy bits are reset via two paths. The normal path is via CU control enables out of the ROM. This path takes the longest (3 clock times) when the instruction in FIR is transferred to FIAR. Because this may delay the start of an instruction overlap, early resets are generated in FIR and enabled when the instruction transfers to FIAR.

INSTRUCTION DECODING

Instruction decoding is the same whether it occurs in the overlap or instruction stations. The format of the twelve bit instruction word is shown in Figure 5(a). Bit 0 indicates whether the operation is in 32-bit or 64-bit mode. Bits 9, 10 and 11 are operations on the data word associated with the instruction such as address indexing when the data word is an address. Bits 1 thru 8 contain the OP CODE. Each instruction is decoded into a microsequence (microprogram) used to address the ROM. Each microsequence consists of from one to 69 microsteps (microinstructions). Generally each microstep is an ROM word. In some operations, such as divide, the same word is addressed many times in succession. However, each time the word is addressed it is considered a microstep. Since a word is addressed
every clock cycle, microsteps are synonymous with clock times. When many words are addressed simultaneously to achieve control enable ORing, this is also considered a single microstep. Each microstep generates a full set of control enables which are stored in the FINST Control Register (FCR). From FCR they are broadcast to the 64 PE's in the quadrant (see Figure 4). Generally from one to 50 enables are active for each microstep.

The decoded instruction contains the starting address of the microsequence and all information for decision making, such as branching, within the microsequence. Typical branches are one of the six ways to do signed and unsigned arithmetic operations. Figure 5b illustrates a portion of a typical microsequence. The seven flip flops shown are “D” type flip flops, i.e., the “1” output is in the same state after the clock pulse as the “D” input was during the clock pulse. Each “1” output is designated by the ROM word it addresses. Related clock times are shown in ( ), e.g., (T2). The flip flops are part of the address register (FOAR or FIAR). Under control of the decoded instruction, the microsequence proceeds from flip flop to flip flop each clock time. Referring to Figure 5b, at time T1, word N is addressed. At time T2, either word N + 1 or N + 2 is addressed depending on the state of control bit I. At time T3, word N + 3 is addressed. At time T4 both words N + 4 and N + 6 are addressed to obtain enable ORing. Finally, word N + 5 is addressed at time T5.

The following microsteps are the microsequence for a floating point add and are accomplished in 250 ns (five clock times):

1. Fetch Operand. Transfer to B register (RGB), in the PE, the operand identified by the address field of the instruction.
2. Difference Exponent. Subtract exponent fields of operands in PE A register (RGA) and RGB.
3. Mantissa Alignment. Shift mantissa of operand in RGA or RGB by amount determined from step 2.
4. Add Mantissa. Add mantissa field of operands in RGA and RGB.
5. Normalize. Normalize sum in RGA.

BASIC ROM REQUIREMENTS

A Read Only Memory (ROM) may be thought of as a numerical conversion table, i.e., the selection of one of the input lines will present a set of predetermined numbers at the output of the memory. It is a simple matter to design a conversion table to read-out one set of numbers corresponding to the selection of either one or more than one input numbers. In block diagram form, such an ROM is shown in Figure 6. However, in this paper, only

From the collection of the Computer History Museum (www.computerhistory.org)
the functional block which performs the numerical conversion, is called the Read Only Memory. The other functional blocks are considered to be peripheral logical functions. The logic design of the ILLIAC IV calls for an ROM having a cycle time of 50 nanoseconds, i.e., the ability to read out a set of numbers every 50 nanoseconds. Referring to Figure 6, the timing from the input register (MIR) to the output register (MOR) is 50 nsec. The memory must accept 720 input lines and present an output of 280 bits. To be useful, an ROM must be alterable either electrically or mechanically. Note that an electrically alterable memory is essentially a read/write memory and is, as a rule, more difficult to construct than a memory that is altered by mechanical means. Because of its large size and fast cycle time, only an ROM in a matrix form, was found satisfactory for use in ILLIAC IV. It may be noted that an ROM can be constructed with sufficient speed by employing ECL gates with typically 2.5 nanosecond propagation delays. Such a design is schematically shown in Figure 7. In principle, the ROM is simply P number of gates where each gate is one output bit of the memory, and M line drivers, corresponding to the M input words. The gates are selectively connected to the input word lines, such that the activation of any one or more word lines provides a predetermined output bit pattern through the gates. The design of Figure 7 was considered early in the ILLIAC IV development. But careful examination of the design reveals that there must be 720 input lines and a minimum of 280 output gates. Each output gate must have 720 inputs. For gates with only 9-inputs (maximum available), each of the output gates must be connected with 80 gates in parallel to accept the 720 possible input lines. The 80 gates produce only one output bit, and must be buffered with multiple stages of OR-gates to provide that one bit output. Similarly, multiple buffering or amplifying stages must be used to drive the large number of output gates. Because of the large number of gates involved, inter-connection wiring is complex. An almost insurmountable difficulty in such a design is to change the content of the memory, mechanically or otherwise. By using a matrix design, the number of printed circuit boards is reduced and the memory is readily altered.

ROM DESIGN DETAILS

The ROM schematic is shown in Figure 8. The form is a standard, transistor cross point matrix consisting of m (720) word lines by n (280) bit output lines. Only those bit lines that are transistor coupled to a word line will switch when that word line switches. Bit line switching is then detected by sense amplifiers on the bit lines. Each word line is powered by a line driver which must tolerate the variations in word line loading caused by the variation in the number of bit lines coupled to the word line. For convenience word line drivers and bit line sense amplifiers are standard devices used elsewhere in ILLIAC IV. The line drivers are level converters that convert standard ECL levels of ±0.4 volts to CTL compatible levels of ±3.0 and 0.0 volts. The sense amplifiers are standard, ECL, 9 input, negative NAND gates.

For coupling word lines to bit lines, transistors offer several performance advantages compared to diodes or resistors. Multiple leakage paths thru resistively coupled cross points would be prohibitive in a memory

From the collection of the Computer History Museum (www.computerhistory.org)
of this size and speed. Compared to diodes, transistors isolate bit line capacity and dc loading from the word lines which alleviates word line driving problems.

**Bit lines**

The required nominal input levels at the sense amplifiers are ±0.4 volts. In order to conserve power, the coupling transistors are biased to be cut-off for a low level word line. Because a bit line may be driven by a coupling transistor at any location along the bit line, each bit line is terminated at both ends in its 50 ohm characteristic impedance. The bit line terminating resistors R02 and R03 have a Thevenin's equivalent of 50 ohms returned to −0.8 volts and quiescently bias the bit line at −0.4 volts. At a high bit line level of +0.4 volts, the coupling transistor must supply 8 ma to R01 and 16 ma to R02 and R03, for a total of 24 ma. Since the $V_{be}$ drop is about 0.8 volts, the word line is required to swing between 0 to +1.2 volts. The collector supply voltage is fed thru diodes D1 and D2, from the +4.8 volt supply, in order to reduce power dissipation in the coupling transistors.

**Word lines**

As explained before, because of the convenience of using available devices, the word line driver provides an output level that swings from 0. v to 3.0 v. As shown in Figure 8, R1 is inserted in the word line to attenuate the 3 volt signal to the desired 1.2 v level. Because of the large number of bit lines crossing the word lines, the length of the word line is electrically long, and is terminated at the far end in 50 ohms. The word lines are 50 ohm microstrip lines. To minimize the word line time delay each word line is divided into two segments, with each segment coupled to a maximum of 140 bit lines. Each segment is designed as shown in Figure 8 and has its own set of line drivers, and line terminations. Although there are 140 bit lines crossing each word line segment, system design requires only a maximum of 25 bit lines be coupled to any one word line segment. Each coupling transistor introduces a capacitive loading of about 1.5 pf, delays the signal propagation by about 0.15 nanosecond, and produces a peak negative reflection of about 30 mv. To prevent an excessively large reflection, no more than 3 transistors are located in succession along any word line or any bit line.

**Memory partitioning**

Because of speed considerations and space limitations, each word line is divided into two segments. Similarly, the bit lines are divided into 6 segments. The result is to divide the ROM into 12 matrix boards. Each board contains 120 word lines and 140 bit lines. The sense amplifiers and output register are mounted on 8 sensing boards. Eight boards are required because of the pin limitations. The partitioned ROM is shown in Figure 9. Note that only the sense amplifiers and bit line terminating resistors (R01) are on the sensing boards. The remaining components are mounted on the 12 matrix boards. The matrix board is a special design. The sensing boards are standard 12 layer boards used throughout the ILLIAC IV Control Unit.

**Mechanical description of the matrix board**

The bit lines are 50 ohm strip lines which gives the advantages of low parallel line cross-talk and close impedance control. The word lines are 50 ohm microstrip lines. Address selection lines, which are the input lines to the word line drivers, are nominally 100 ohm micro-strip lines to simplify board construction. The
address selection lines are constructed in two layers with both layers using the same ground plane as the signal return path. Consequently, the two layers of address lines are slightly different in line impedance. The lines in the top layer are about 110 ohms and the lines in the bottom layer are about 90 ohms. The matrix board cross section is shown in Figure 10. The board dimensions are otherwise the same as a standard CU board, i.e., 18 inches X 20 inches.

The transistors are cubes of 80 mils on an edge. The transistors are spaced 100 mils apart and so are the spacing of the holes and mounting pads for the emitter, collector, and base leads. The tight spacing of the transistor requires a special mounting technique as shown in Figure 11.

CHECKOUT OF MATRIX BOARDS

The coupling transistors are located on the matrix boards in accordance with the system instruction table, i.e., the output bit pattern for every word selected. The simplest checkout fixture is to apply a +0.4 volt input level to each word line driver and read the output of every bit line with a voltmeter. Such a manual checkout system would require 320 man hours for the 12 matrix boards. A more automatic checkout system is shown in the flow chart of Figure 12. The design table of the ROM is transferred onto 80-column cards. Each card will have 80 bits and 12 words. Therefore, 2 of the 80-column cards are required to complete the contents of 12 words by 140 bits, and 20 cards are required for one of the 12 matrix boards. A card reader reads one card at a time and compares the reading with the bit outputs. Any mismatches are detected and displayed by indicating lamps. With this system the matrix boards can be checked out at the rate of 9 boards per hour or 2 man-hours for 12 boards. The cost of the checkout system is about $3,000.

In ILLIAC IV, neither of the above approaches is used. The Test and Maintenance Unit (TMU) area of the Control Unit has a 64-bit comparator which can be used to check out not only the matrix board but the entire area of instruction decoding. Via the I/O interface,
instruction decoding and ROM addressing can be controlled programmatically. The program generates the FINST instruction to be decoded and loads a TMU register with the expected ROM response. The comparator compares the expected and actual responses and the program proceeds if they agree. Because only 64-bits can be compared at a time, five compares are required to check each full 280 bit output of the ROM. If there is an error, the inputs being compared can be displayed on a CRT display in the TMU (Item 6, Figure 13).

APPLICATION OF MSI

For the entire function of decoding instructions into microsequences, storing enable patterns in the ROM and sensing and storing the ROM output, the present system uses a total of 28, 18 inch by 20 inch, multilayer boards. As the following example shows, an MSI approach would result in about a 50% reduction in volume. An all MSI ROM is composed of MSI cells where each cell is M words by N bits. To reduce interconnections and conserve pinouts, x-y addressing is used to address the M words in the cell. To eliminate buffers between cells, the N bits of one cell are "wire OR'ed" to the N bits of another cell.

Control Enable ORing is no longer possible because x-y addressing generates unwanted addresses if more than one x or more than one y input is active. The number of required ROM words is increased to about 1200 because this ORing cannot be used. The amount of logic required to generate microsequences also increases. However, the generation of microsequences is now more straightforward and better adapted to MSI. Based on the average number of words required by a microsequence, the optimum number of words in a cell is determined. Assume the optimum number is 16 words. Therefore, four address lines plus one address enable line are required. The address enable permits more than one cell, i.e., more than 16 addresses, to be used in a microsequence. Because many microsequences do not use whole multiples of 16 addresses, more than 1200 addresses are required. Assume the ROM size is increased by 20% to 1440 words to account for this affect. Assume each cell has 64 outputs. This requires less than 100 pins per cell which is consistent with present MSI packaging. To obtain 280 outputs, five cells are required for every word. Therefore a total of 450 cells are required to make up the complete ROM. Allowing four square inches of surface area per cell and sufficient space for terminating resistors, bypass capacitors and connectors, a total of 8, 18 inch by 20 inch, boards are required to make up the ROM. If the remaining 16 boards required for microsequence generation and ROM output sensing and storing can be reduced to 6 boards, then the original 28 boards are reduced to 14 boards. It is important to note that in order to achieve an ROM cycle time of 50 nsec (i.e., register to register), the cell access time (i.e., address input to bit output) will have to be, approximately, 20 nsec.

ACKNOWLEDGMENT

The final ROM is the result of the efforts of many people and the authors gratefully acknowledge their contributions. Especially noteworthy is work by E. S. Sternberg in the area of logic design and T. E. Gilligan in the area of circuit design.

REFERENCES

3 R A STOKES ILLIAC IV: Route to parallel computers Electronic Design Vol 26 pp 64-66 December 20 1967
4 Multilayer printed circuit board—technical manual The Institute of Printed Circuits

From the collection of the Computer History Museum (www.computerhistory.org)