A Custom LSI Approach to a Personal Computer

by Todd R. Lynch

THE NEW HP-85 PERSONAL COMPUTER, which was featured in last month's issue, is a system totally integrated into a single package. Included in this system are a CRT, printer, tape drive, and keyboard. To control the I/O (input/output) devices and to interface to ROM (read-only memory) and RAM (random-access memory), the design uses LSI (large-scale integrated) circuits designed and fabricated by Hewlett-Packard.

The design of custom chips minimizes the cost of the electronics. Also, the power dissipation is reduced to a level that permits the use of air-convection cooling, eliminating the need for a fan. The LSI designs save large amounts of printed-circuit-board space, making a small system package possible. By designing LSI circuits dedicated to each mechanical subassembly, features can be added to the overall system that would be nearly impossible to incorporate with discrete logic designs.

HP-85 LSI Chip Set

There are nine custom LSI circuits in the HP-85. These circuits are interconnected as shown in Fig. 1. Eight of the LSI circuit designs use NMOS technology. The ninth circuit uses a bipolar technology with two layers of metallization.

The heavy emphasis on LSI design makes the HP-85 very compact. The electronics in the machine consists of these circuits plus a power supply, clock generator, CRT raster scan circuitry, and various motor and printhead drive circuits.

The system is partitioned as follows.

- CPU (central processing unit). The CPU commands the bus control lines and interfaces directly to the system operating commands (firmware) stored in the four ROMs.
- ROMs. Each ROM chip has the same basic custom design, differing only in the bit pattern permanently stored on each chip.
- Read/Write Memory. The user memory consists of eight commercially-available dynamic 16K×1 RAMs.
- RAM Controller. This chip is designed to interface between the CPU and the RAMs.
- Buffer. This chip is designed to provide the capability for system expansion by adding plug-in units in the rear of the machine.
- Keyboard Controller.
- Printer Controller.
- CRT Controller.
- Display Memory. Four commercial dynamic 16K×1 RAMs are used to store data for the display. Approximately one-fourth of this memory is used for the alphanumeric data and the remainder is used for graphic data.
- Cartridge Controller.
- Sense Amplifier. This bipolar chip interfaces the tape cartridge controller to the magnetic tape head.

Table I is a summary of the custom LSI chip sizes, number of pins (wiring connection points), and power dissipation.

The machine's sixteen-bit address allows direct access to 65,536 bytes of information. The memory map in Fig. 2 shows how this space is allocated. Note that 32K bytes of the address space are devoted to ROM while 16K bytes are for RAM. Additional RAM and ROM capacity may be added to the system. The I/O address space occupies the upper 256 bytes of memory. Each I/O device controller has from two to four dedicated addresses assigned to it.

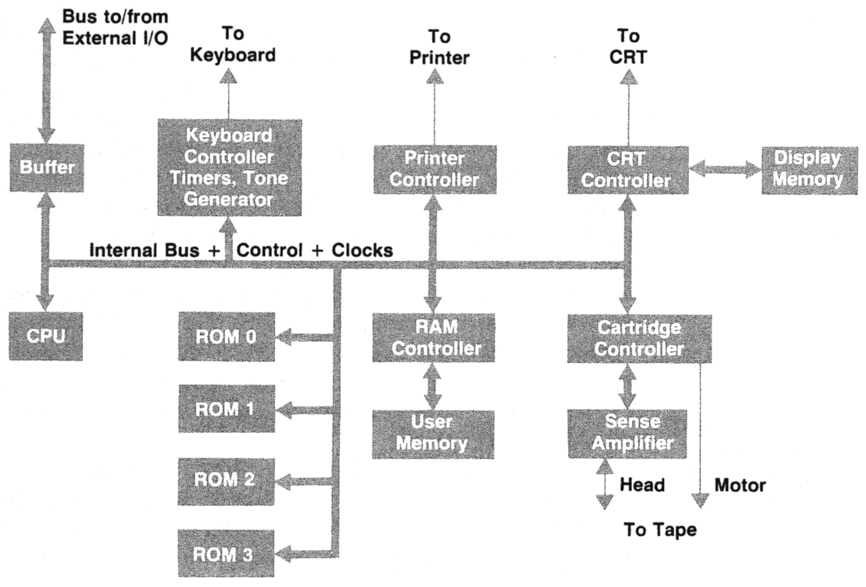


Fig. 1. *The HP-85 system block diagram contains nine custom LSI circuit designs. The user and display memories are the only parts in the diagram that use commercially available circuits. The development of custom circuits enabled the system to be contained in a compact package at low cost and without the need for a cooling fan.*

Table I
Custom LSI circuit characteristics. All of the circuits are fabricated with NMOS silicon gate technology except for the sense amplifier which is made with dual-layer-metallization bipolar technology.

Circuit:	Size (mm)	#Pins	Power (mW)
CPU	4.93×4.01	28	330
ROM	4.75×5.41	28	200
RAM Controller	2.67×3.56	40	220
Buffer	2.54×3.35	40	340
Keyboard Controller	3.78×4.39	42	200
Printer Controller	4.78×5.44	40	300
CRT Controller	4.14×5.51	40	200
Cartridge Controller	3.63×3.86	28	55
Sense Amplifier	1.50×1.55	16	150

CPU Design

Several commercially available CPUs were considered at the beginning of the project, but none could provide all the features needed to efficiently implement a powerful scientific BASIC language machine. BASIC requires high-precision arithmetic, which is best accomplished with decimal rather than binary numbers. Many different stacks are needed to parse (separate a statement into executable steps) and execute the language. The ability to handle variable-length data is required for variable-length tokens (bit sequences from one to several bytes in length), and multilevel vectored interrupts are needed to handle the I/O devices in real time. The design of the custom NMOS CPU incorporates all of these requirements plus many more.

Fig. 3 shows a block diagram of the HP-85 CPU. Major blocks are the 7000-bit PLA (programmable logic array), 64-byte register bank, and eight-bit ALU (arithmetic logic unit) and shifter. Each CPU instruction is decoded by a microprogram in the PLA that directs the rest of the chip to perform the desired function.

A very powerful feature of the CPU, which is incorporated into the PLA microprogram, is the ability to handle data ranging from one to eight bytes in length. This multibyte feature lets the programmer, for example, add two eight-byte mantissas, increment a sixteen-bit address, or load into the CPU a three-byte token from memory, each with a single instruction. With an off-the-shelf CPU, this would require setup and iteration within a software loop.

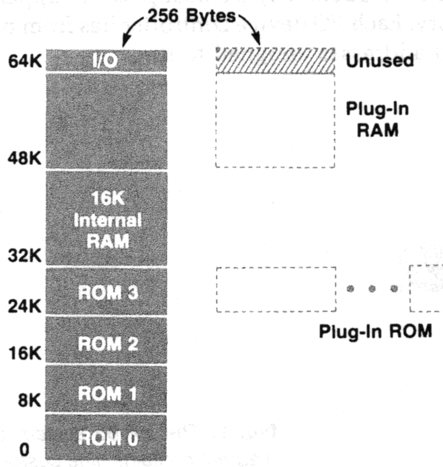


Fig. 2. *The HP-85 memory map allocates four 8K-byte system ROMs and one 16K-byte internal RAM. The RAM is used for user memory and 256 dedicated I/O locations. By using bank selection, up to six more 8K-byte ROMs may be added. The RAM may be expanded by 16K bytes if a plug-in module is added to the system.*

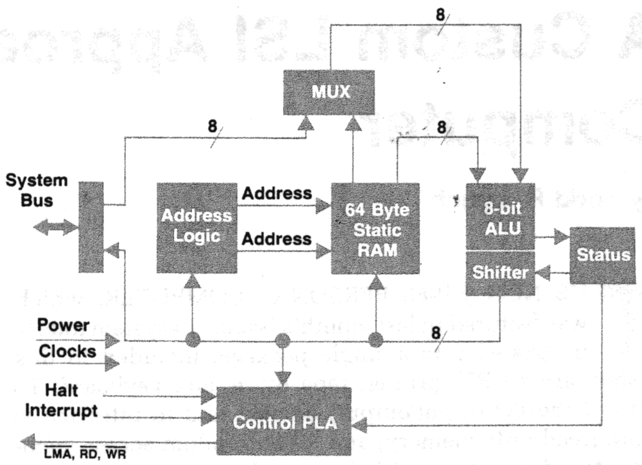


Fig. 3. *The CPU in the HP-85 is a custom NMOS circuit design. A 7K-bit programmable logic array (PLA) controls the interactions between the register bank, the ALU, and the rest of the HP-85 system.*

The eight-bit ALU is capable of shifting and can do both decimal and binary arithmetic. The programmer sets a mode bit to specify the type of shifting or arithmetic desired. This, combined with the multibyte feature, lets the programmer easily work with signed, floating-point mantissas up to sixteen digits in length or two's-complement binary integers up to sixty-four bits in length.

Of the 64 eight-bit registers contained in the CPU, one pair is dedicated to the program counter, one pair to the stack pointer, and one pair to internal index calculations. The rest are general-purpose registers. One advantage of having a register as the program counter is that it can easily serve as one of the operands for any CPU instruction. If it is modified by an instruction, then an immediate jump to the new location occurs. The stack pointer points to the subroutine return address stack stored in memory. Additional stacks can be created with any other consecutive pair of registers in the CPU. Thus, the programmer can maintain many different stacks at any one time. Sixteen instructions are dedicated to manipulating data on the currently designated stack. Any pair of consecutive registers can also be used as a sixteen-bit index register. This lets the programmer index into several data arrays at the same time.

To handle real-time I/O devices, the CPU has multilevel vectored interrupt handling ability. Up to 127 interrupt vectors can be accommodated by the architecture. The CPU can also be halted by an external device. This lets that device control the system bus, so it can have direct memory access at high speeds.

No accumulator is present in the CPU. This is made possible by the design of the 64-byte register bank. The registers are constructed as a two-read/one-write memory. This means that, at a given time, any two bytes can be read

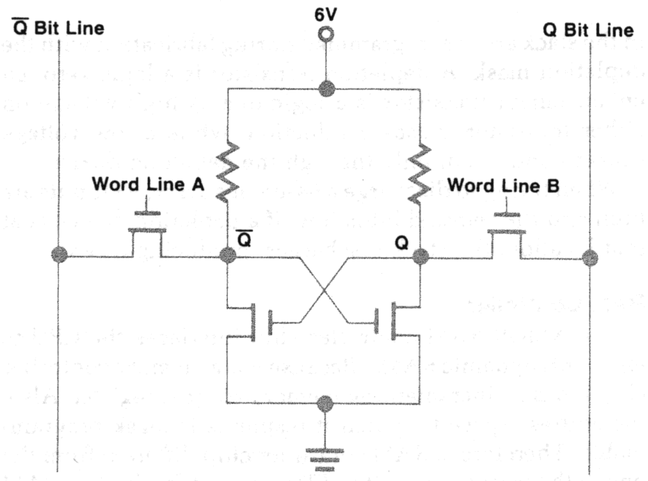


Fig. 4. *Independent control of each word line is used to achieve a cell that can be read out onto either bit line. The RAM made up with these cells can be read two words at a time.*

from the memory and operated on in the ALU. The result is returned to one of the accessed byte locations. This sequence takes one processor cycle (1.6 μs).

To design a two-read/one-write memory, a standard static RAM cell with a special variation is used (see Fig. 4). In a normal static RAM cell, a common word line enables both transmission gates between the bit lines and the cell. In a two-read cell, there must be two different word lines. Word line A enables complement information from one cell to be read out onto the $\overline{Q}$-bit line while word line B enables true information from another or the same cell to be read out on the Q-bit line.

The circuit design of such a cell must be done carefully. Since the bit lines are precharged before reading, it is possible that by enabling a single word line, a cell could be made to flip (change state) rather than be read. Inadvertent flipping is prevented by using a low voltage for the word line and a proper size ratio between the cell's pull-down transistor and the transmission gate.

The static RAM cell's pull-up device also received close scrutiny during the design phase. A cell with a low-power depletion-load pull-up requires an area larger than 6400 square micrometres (10 square mils) and a quiescent power of 120 microwatts. By using a polysilicon pull-up resistor, the RAM cell area is reduced by 40% and the power is reduced by 80%. The savings in area amounts to a reduction in size of more than 0.25 mm on each side of the die. The reduction in total power consumption is fifty milliwatts.

To obtain these reductions, a polysilicon resistor process was developed for the CPU. An additional masking step is required to define the regions on the chip where the polysilicon layer is lightly doped, hence creating the polysilicon resistors. The doping level was chosen for a sheet resistance of 10^7 $\Omega/\square$. An undoped layer could not be used because the sheet resistance was so high that the junction leakage currents at elevated temperatures resulted in unwelcome voltage drops across some resistors. It was also discovered that if the contact mask overlapped the polysilicon resistor area, the aluminum metallization could spread into the lightly doped resistor and lower its resistance.

System Control and Timing

Eight bus and three control lines leave the CPU. The eight-bit bus is used to time-multiplex a sixteen-bit address, instructions, and multibyte data quantities. The three control lines—$\overline{\text{LMA}}$ (load memory address), $\overline{\text{RD}}$ (read) and $\overline{\text{WR}}$ (write)—indicate to the system circuits what type of information is on the bus.

When $\overline{\text{LMA}}$ is low, all chips in the system know that one byte of a two-byte address will be placed on the bus. The $\overline{\text{LMA}}$ signals always come in pairs since all addresses are sixteen bits. Each chip in the system reads this address and compares it to the address range to which the chip is supposed to respond. Setting $\overline{\text{RD}}$ low indicates that the CPU wants to read the contents of the address most recently sent out. When $\overline{\text{WR}}$ goes low, the CPU wants to write into the location most recently addressed.

The multibyte feature is accomplished by sending out an address with two $\overline{\text{LMA}}$ signals followed by one to eight $\overline{\text{RD}}$ or $\overline{\text{WR}}$ signals. The circuit being read from, or written to, is expected to increment its memory address register every time it sees a $\overline{\text{RD}}$ or $\overline{\text{WR}}$. The CPU sometimes can fetch consecutive instructions from memory by simply sending out additional $\overline{\text{RD}}$ commands. This speeds up the instruction rate of the machine since sending out an address for every instruction and piece of data is not necessary.

Four nonoverlapping phases (Fig. 5) with 200 ns width and 200 ns spacing clock the circuits in the HP-85. In one cycle the system can access a preaddressed memory location, read it into the CPU, add it to a CPU register and store the result in the CPU register. An eight-byte add requires eight cycles plus the time to fetch the command, which is usually three cycles.

Before any chip writes on the bus, the bus gets precharged to a logic one. Therefore, the circuit desiring to put a one on the bus must merely continue to hold the bus high. If it wants a zero, it must discharge the bus. In NMOS technology it is easier to discharge a bus than to bring it high. Consequently, the circuits were designed with large devices to discharge the bus and relatively small devices to maintain a high level on the bus. This minimizes the chip area required for each circuit's driver logic.

The system bus has fewer spurious transitions using the

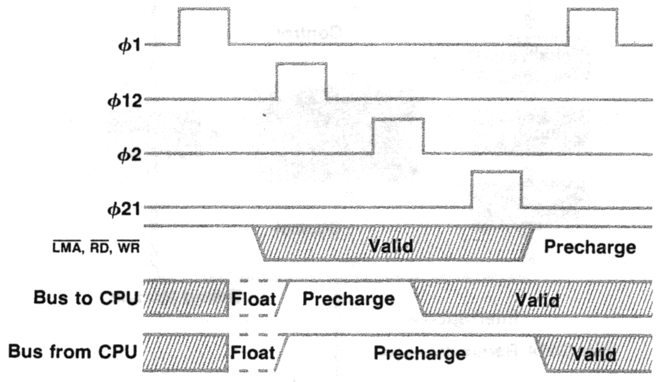


Fig. 5. *HP-85 system timing and control. The control lines are valid by the time the ϕ12 clock pulse occurs. If a system chip is to be read, it must respond during the ϕ2 clock pulse. Information then enters the CPU, goes through the ALU, and is stored during the next ϕ1 clock pulse. When the CPU sends out addresses or data, they are valid by the time the ϕ1 clock pulse occurs.*

precharged scheme. Timing is such that the driving circuit is not enabled until its data is valid and ready to go on the bus. It is not possible to discharge the bus inadvertently when the chip wants a logic one to be output.

ROM

The NMOS system ROM is organized as an 8K by eight-bit array. Because this circuit must respond to multibyte transfers, its memory address register is designed to increment as $\overline{RD}$ commands are given. The circuit can be enabled or disabled by a bank select command. When plug-in ROMs are used, it becomes necessary to selectively enable or disable ROMs in the address space from 24K to 32K (Fig. 2). Each ROM in this address space has a unique eight-bit bank number, and only one bank can be active at any given time.

The core of the ROM consists of 64-input, minimum geometry, NAND gate arrays. The principle of operation is to precharge the NAND stack from both ends. This requires 600 ns. At the beginning of phase $\phi12$, one 64-device stack per bit of output is selected. If the stack discharges, the bus bit will be a zero, otherwise it will be a one. The transistors in the stack are preprogrammed during fabrication with the depletion mask. A depletion transistor is a logic zero, an enhancement transistor is a logic one. A high voltage on either transistor causes conduction, while a low voltage causes conduction only through the depletion device.

When trying to discharge a 64-device stack, 63 inputs are high and the selected bit is low. If a depletion device is at that location, the stack discharges, producing a zero.

RAM Controller

The NMOS RAM controller chip interfaces the CPU to eight 16K dynamic RAMs. Because it is a memory controller chip it has an incrementing memory address register. Also, the address space to which it responds is mask programmable. Therefore, a RAM controller chip different from the one in the mainframe of the HP-85 is used for the 16K RAM plug-in. This chip knows not to respond to dedicated I/O addresses in the upper 256 bytes of memory.

An important function of the RAM controller chip is refreshing the dynamic memory. An internal timer on the chip tells it when the next sequential location is due to be

The HP-85 Software Development System

by Nelson A. Mills

The HP-85 is based upon a new eight-bit custom processor. One of the first steps in the development of the machine was to provide a set of software development tools. The HP-85 software development system consists of an assembler, a hardware interface for the HP-85 simulator, and software debug system (see Fig. 1).

The assembler for the HP-85 processor was designed to run on an HP 1000 Computer. The assembler supports the full range of the HP-85 processor's instruction set and provides several useful pseudo-operations. Programs may be either absolute or relocatable and the program origin may be reset at any time. Several instructions are provided to facilitate data definition and may exist locally within the program, or may be retrieved from a file of global data definitions.

The hardware interface designed for the HP-85 software development system provides the ability to use an HP 1000 to control execution of the HP-85 simulator. The interface contains an on-board RAM which can be downloaded from the HP 1000 and is used to simulate the HP-85 ROM. Included are two breakpoint registers which can be used to halt execution of the HP-85 processor at either of two specified addresses. The processor can be made to execute steps in either a continuous-run mode, or in a single-step mode where execution halts after each instruction. The interface also provides the ability to halt the HP-85 processor at any time, and to read data from or write data to the system RAM or CPU registers.

The software debug system, provided as part of the development system, runs in the HP 1000 and performs four important functions. It includes a relocatable loader that is used to download the RAM on the interface board with the software under development. It controls execution of the HP-85 simulator via the interface board. It displays the current status of the breadboard, including CPU status, program counter, and the current contents of all CPU registers and any specified memory locations. Finally, it provides the means for system developers to modify the contents of memory, registers, status, or program counter, and to clear and set breakpoints.

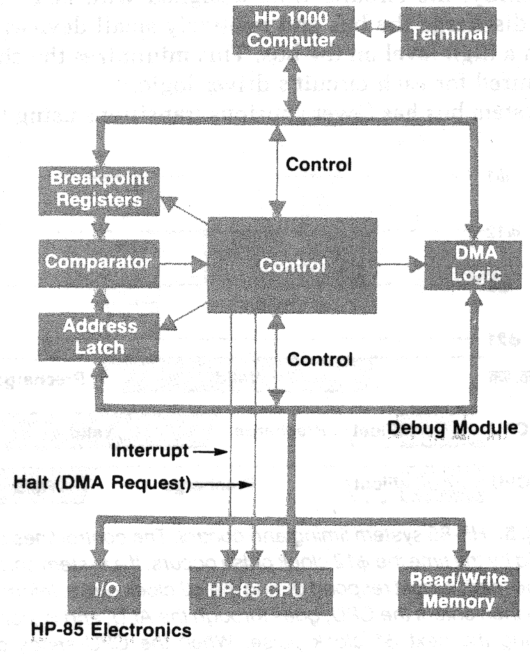


Fig. 1. *HP-85 software development system.*

Nelson A. Mills

Joining HP in 1976, Nelson Mills worked on the HP-85 operating system and interpreter. He is now the project manager for high-end firmware at the Corvallis Division. After receiving a BS in mathematics from Albion College, Michigan in 1961, Nelson spent five years in the U.S. Navy and then worked for ten years as a systems programmer. He and his family—wife and two children—live in Corvallis, Oregon. Outside of working hours Nelson enjoys photography, hiking, camping, and coaching basketball and AYSO soccer.

refreshed. The chip then waits for a cycle when memory is not being accessed to do the refresh. There is a two-element queue in case a vacant memory cycle does not come before the following location must be refreshed.

Keyboard Controller

The circuit needed just to interface to the keyboard would have been quite small. Consequently, four programmable timers were added to this NMOS chip. The timers can count up to 27 hours with one-millisecond resolution. Each has a maskable interrupt. When the timer gets to its preset count, it interrupts, resets and counts again.

The keyboard portion of the chip can scan an 8×10-key keyboard plus three dedicated keys—shift, control, and caps lock. The key debounce time is mask programmable from 1.67 to 11.69 ms. After the key has been debounced, the chip generates an interrupt. An internal ROM is used to convert the key position to its ASCII* equivalent.

Another feature of the chip is an output to a speaker for audio tones. Via firmware, a 1.2-kHz output tone can be obtained, or the frequency can be varied by periodically setting and clearing an internal flip-flop. This output can be accessed by the programmer using a command that can specify the frequency and duration of the tone.

CRT Controller

The NMOS CRT controller[1] interfaces the CPU to a 127-mm diagonal CRT and its dedicated display memory. The memory is four 16K×1 dynamic RAMs. This is enough storage to hold a full display of graphics information (256×192 dots) plus four displays of alphanumeric information (4×32 characters/line × 16 lines/display). The chip must make sure that this memory is properly refreshed during the vertical retrace. The interface to the video drive circuitry consists of a vertical sync pulse (60 Hz), a horizontal sync pulse (15.7 kHz), and a video line (4.9 MHz). An internal ROM provides a dot pattern translation of the stored ASCII characters.

Printer Controller

The NMOS printer controller[2] is the interface between the CPU and the thermal moving-head printer. To perform this function, the circuit must control a printhead drive motor, a paper advance motor and an eight-dot printhead. An internal 32-byte RAM allows firmware to buffer one print line of alphanumeric data at a time. An internal ROM translates an ASCII character into its appropriate dot pattern.

Tape Controller and Read/Write Amplifier

The NMOS tape controller and the bipolar read/write amplifier[3] control the tape unit in the HP-85. The read/write amplifier reads and writes data on the tape through a two-track magnetic head. The tape is formatted using a 1:1.75 delta distance code, in which 8-kHz flux reversals represent zeros and 4.6-kHz flux reversals represent ones.

The tape controller encodes digital information into this format and sends it to the read/write amplifier. When reading the tape, the signals from the read/write amplifier are decoded by the controller. The tape drive motor direction and speed are also controlled by the chip.

*American Standard Code for Information Interchange

Buffer

The NMOS buffer is the ninth chip in the system. With its help, the system bus can be expanded to include all of the plug-in I/O slots. It is capable of driving a 150-pF load. The delay between NMOS-level input and NMOS-level output is 50 ns. The signals that the buffer passes are eight bus lines, three control lines, and the interrupt and halt lines. The bus and control lines are designed to be bidirectional.

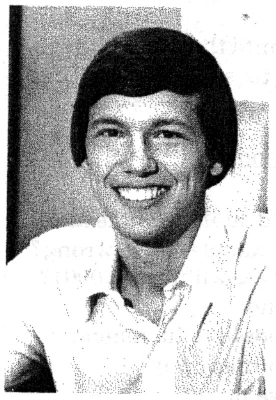
The clock lines are not buffered since this would cause unwanted skewing. The clock generator is capable of driving all internal as well as all external loads.

Acknowledgments

A great deal of credit must be given to the team that made this LSI chip set possible. Tim Williams designed the RAM controller and buffer. Donn Wahl designed the ROM. The keyboard controller was an effort from Jim Hutchins and Jerry Erickson. Jerry also designed the CRT controller. The printer controller was the work of Clement Lo. Implementation of the NMOS CPU was done by Jim Axtell. The cartridge controller design is that of Doug Collins. The sense amplifier is the result of the combined efforts of Mike Moore, Doug Collins and Mike Barbour. Special thanks must be given to Don Hale and Howard Shishido, who did the majority of the chip mask designs. Howard Honig contributed to the project by writing most of the test programs for the LSI circuits. Mike Pan, Rock Davidson, and Tom Kraemer also contributed to the chip set design effort. Bob Tillman and Steve Larsen helped get the NMOS process and polysilicon resistor development up to production standards. Payne Freret contributed some good ideas to the CPU design. Rick Bell did the design of the clock oscillator and generator.

Reference

1. J.F. Bausch, "A High-Quality CRT Display for a Portable Computer," Hewlett-Packard Journal, July 1980.
2. C.C. Lo and R.W. Keil, "A Compact Thermal Printer Designed for Integration into a Personal Computer," Hewlett-Packard Journal, July 1980.
3. D.J. Collins and B.G. Spreadbury, "A Compact Tape Transport Subassembly Designed for Reliability and Low Cost," Hewlett-Packard Journal, July 1980.



Todd R. Lynch

Todd Lynch is a native of Rochester, New York. He joined HP Laboratories in 1972 after working for a year on process control logic designs. Since coming to HP he has worked on computer architectures, and since transferring to what is now HP's Corvallis Division, Todd designed the architecture for the HP-85 CPU. He was the project manager for the integrated circuits in the HP-85 and is now project manager for high-end mainframes. He received a BS degree from Grove City College, Pennsylvania in 1970 and an MSEE from the University of Wisconsin in 1971. Todd is co-chairman for the Willamette Valley Junior Tennis Tournament and in addition to tennis enjoys woodworking, softball, and skiing. He lives in Albany, Oregon with his wife and new son.