The MicroVAX I Data-path Chip

Glenn Louie, Tom (Iu-meng) Ho, and Ed Cheng, Silicon Compilers, Inc., Los Gatos, CA

During 1983, Silicon Compilers, Inc. (SCI) and Digital Equipment Corp. (DEC) implemented the MicroVAX I Data-path component, which contains most of the computational logic of the new MicroVAX I computer manufactured by DEC. The capabilities of the data path meet the requirements of system designers who are currently using bipolar bit-slice components such as members of the AMD 2901/29116 family. These designers can also customize microarchitectural details to fit their applications exactly. This full-custom nMOS chip was developed with the use of SCI's silicon compilation tools. The development strategy shows how systems companies can gain the advantages of VLSI components in a timely manner.

Project Goals

The goals of the MicroVAX computer project were:

- To ship the product within a year
- To make the product VAX compatible, thus offering significant cost savings to DEC customers
- To make the MicroVAX I computer reside on two quad boards

The team formed for this project at DEC in Bellevue, WA (DEC WEST) consisted of system designers who had worked on the original VAX design. In developing a low-cost VAX microcomputer, DEC wanted to take advantage of VLSI technology to reduce the cost and size of the entire system. DEC also wanted the performance of the product to be competitive with that of the aggressive new high-end microprocessor-based systems that were beginning to appear on the market. Of course, the product had to be compatible with the existing VAX product line, in which the company and its customers had made an enormous software investment. DEC's primary aim was to introduce the product quickly.

A major design goal was to implement the entire VAX on two quad-size boards (79 in^2 per board). (The original VAX 11/780 occupied 27 hex-size boards (123 in^2 each), the VAX 11/750 required 5 extended hex-size boards (160 in^2), and the VAX 11/730 required 3 hex-size boards.) This requirement meant that MSI and gate-array technology, which had been used on earlier VAXs, would not provide the necessary circuit density. Furthermore, the VAX compatibility requirement barred the use of a standard microprocessor.

At approximately the same time that the DEC WEST team was beginning to investigate ways to develop the new VAX product, the management of Silicon Compilers, Inc. decided that the company's prototype tools for full-custom VLSI circuit design needed to be tested by another chip project. (The earlier version of the silicon-compilation tools had been used to develop the first commercial Ethernet™ controller.)

Our goals were to exercise our prototype tools, and to obtain working parts within six months.

SCI's Approach to the Project

DEC's goal of shipping the product one year after inception meant that a "normal" approach to custom-IC development would be impossible. There was no way SCI could design a custom microprocessor chip or chip set that implemented the entire VAX computer, and still meet the schedule. More specifically, this project did not appear to permit the enormous development risk of any full-custom VLSI project, mainly with regard to multiple design iterations and considerable schedule slippage.

However, we believed that we could reach all of the project's goals by developing a single custom component that incorporated much of the VAX processor, but not all of it. The project began to look feasible and perhaps even possible. Of course, "feasible" and "possible" were relative terms. "Feasible" meant that for the end product to be available in one year, the chip would have to be working in nine months; the rest of the system design and production would have to be done simultaneously. All of these elements would have to match up at the end, and work perfectly. Therefore, all component-design work would have to be done in six months and would have to work right the first time. "Possible" meant that in addition to fast negotiation of a development contract and agreement on design details, nothing could be allowed to get in the way of the schedule: no surprises.

System Architecture

During the very early discussions of the system architecture, someone suggested that a good starting point would be to implement the VAX 11/730. This machine was the smallest VAX available at the time, and its overall size would need to be reduced by no more than about 56 percent to meet the size goal. It is always attractive to attempt a size and cost reduction by re-implementing an existing internal architecture in a new technology, instead of starting from scratch with a new implementation architecture. However, we soon realized just how much an implementation is affected by the available technology. The VAX 11/730 implementation uses 2901 bit-slice bipolar parts in

its main computational unit. This approach achieves system performance essentially by using fairly simple circuits running very fast. MOS VLSI parts, on the other hand, are most effective when complex circuits are run relatively slowly. Copying the architecture of the VAX 11/730 would have meant that the custom VLSI circuit had to run at a 70-ns cycle time—not an attractive prospect for a part that had to be designed very quickly and that had to be manufacturable by several commercial foundries.

Fortunately, the VAX architecture was very well documented, and SCI had access to people at DEC WEST who could answer any question about the VAX architecture. Thus, we could proceed knowing what the final design had to do, what was feasible in MOS, and what the SCI tool set could do. Most important, we did not have to deduce the requirements by disassembling logic diagrams. Had this not been the case, we would have had little choice but to try to re-implement the same logic in the latest technology. (We already knew that the technology had not come far enough since the earlier implementation to meet the goal of multi-source manufacturing.)

System Partitioning

Once we had decided to implement a new internal architecture, we also had to decide which portions of it could best be placed on a single chip. This trade-off had to consider DEC's constraint of a two-board system. The computation engine of a computer is sometimes called the *data path*, because it provides a thoroughfare for information transfers between the data-processing elements in the computer (*e.g.*, it performs register-transfer operations). Data-processing elements include arithmetic logic units (ALUs), shifters, registers, latches, and counters. When a computer contains many data-processing elements, it is most efficient to connect these elements through common buses. Therefore, data paths are usually formed by buses and other common lines. The elements communicate with each other not only for direct data transfer, but also while performing various micro-operations. The communicating elements are controlled by a set of digital signals, which usually come in from outside the data-path block. We selected a straightforward two-bus data-path structure for the central processor, and designed the memory-management system as a separate logical unit operating in parallel.

The key question then became, "What parts go on the chip?" Random logic is often the first choice for integration of CPUs and controllers. However, in our case, the amount of random logic was reduced by the regular structure of the architecture, and could be implemented readily with microcode, programmable array logic, and programmable logic arrays (PLAs). The data path itself accounted for much of the chip count in MSI. The data path's regular design and stable functional specification made it the ideal candidate for quick implementation using SCI's block compiler. Thus, we selected the data-path section of the processor and its control logic for integration on the custom VLSI chip.

As shown in Figure 1, the control store, which contains the microprograms that implement the VAX instruction set, was not integrated onto the chip. Our decision to leave this section outside the chip, for implementation with ROMs, was based on the following considerations:

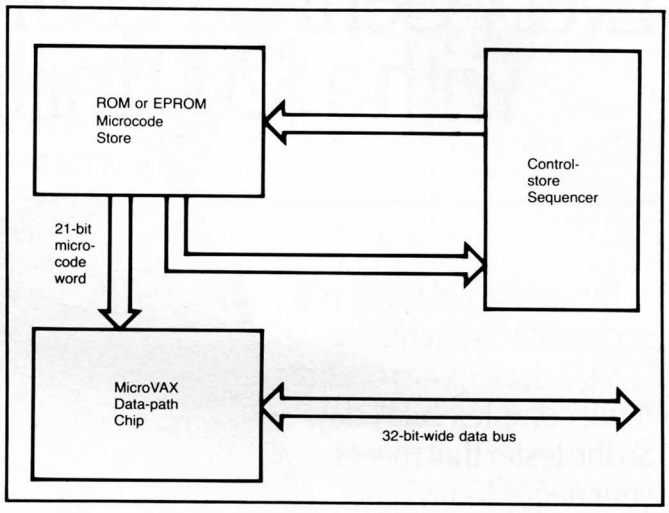


FIGURE 1. MicroVAX system partitioning.

- Incorporating the large control store on the chip would have made the chip prohibitively large.
- The use of standard off-the-shelf EPROMs and ROMs still offered the advantages of high-density VLSI technology.
- The VAX design team at DEC WEST would be solely responsible for coding the VAX instruction set. This way, they could work on the microcode while the SCI team worked on the chip design.
- Any late changes in the microcode (*e.g.* to fix bugs) could be done easily with EPROMs even after the custom chip was completed; such changes would not entail long delays for reworking the IC masks.

We also left the control-store sequencer off the chip, primarily because of the pin limitations of the package. Pin limitations also greatly affected the design of the system/chip interface. We selected a 68-lead pin-grid array package because it gave a generous number of pins in a cost-effective package. We also used a single bidirectional 32-bit data bus, along with a separate 21-bit control bus. These two buses alone required 53 of the pins, without including power and ground.

Microarchitecture

Data is processed by 32-bit-wide data-path blocks connected via a pair of 32-bit buses. These data-path blocks provide the register storage and computational hardware for the data-processing part of the system. The data path consists of the following blocks:

- A 47-register dual-port RAM file to hold 8-, 16-, and 32-bit operands
- A 32-bit full-function ALU
- A 64-bit-wide barrel shifter
- A 32-bit program counter
- A 7-level operand-restore stack
- A 32-word constant ROM

The two buses and the dual-port RAM allow access to

Clock Cycle 0 ns	250 ns	500 ns	
Instruction 1	Instruction loading and decoding	Data transfers and instruction execution	
Instruction 2		Instruction loading and decoding	Data transfers and instruction execution

FIGURE 2. One instruction is executed every 250 ns in a simple two-stage pipeline.

two operands in a single clock cycle, for processing by the ALU or by the barrel shifter.

Multiplication can be done via the ALU using one clock cycle per bit. The barrel shifter can perform a multiple-bit left or right shift in a single clock cycle. We selected these computational blocks based on how well the microcode could use them to implement the instruction set, and on how efficiently they could be implemented in MOS technology. The barrel shifter and dual-port register file were much easier to implement with MOS than with MSI/TTL, and were attractive facilities for the microcoder.

The chip also has a 16-bit interval timer block, which can be used as the system's real-time clock or in microcode parity calculations to ensure system integrity.

The control section of the chip consists of a control-store register that is loaded from the external microprogram ROM with a 21-bit control word. Three separate PLAs were used to decode the control word and to generate explicit controls for the data-path blocks.

To minimize adverse effects of control-instruction latency from an off-chip microcode source, we used a simple two-stage pipeline for the controls (Figure 2). While one instruction is being loaded and decoded, the previous instruction is transferring its operands via the data paths and activating the block calculations. One microinstruction is executed every 250 ns, using an internal 4-MHz clock.

Compilation

The chip design contains 15 blocks, including four pad-frame blocks. Figure 3 shows the outline of these blocks and a photograph of the die. The chip was generated by a chip-specification file that explicitly called up the 15 blocks and described the interconnections between them. The chip-compiler executed this specification by calling up software routines (called *block generators*) to form the blocks from elemental cells in the data base. Once the blocks were generated, a router was invoked to make the chip interconnections according to the specification.

The chip was fabricated in 3-μm nMOS technology. The actual layout design rules were a set of composite, foundry-independent rules adjusted at tape-out time for the manufacturer. Thus, the rules enable compatibility with different silicon foundries. The chip was prototyped at two foundries, with slightly different design rules, to guard against any unforeseen production problems. The final chip size is 314 mils square (98,600 mil^2).

Data-path Block Design

Logically, the chip has a single 32-bit-wide data-path

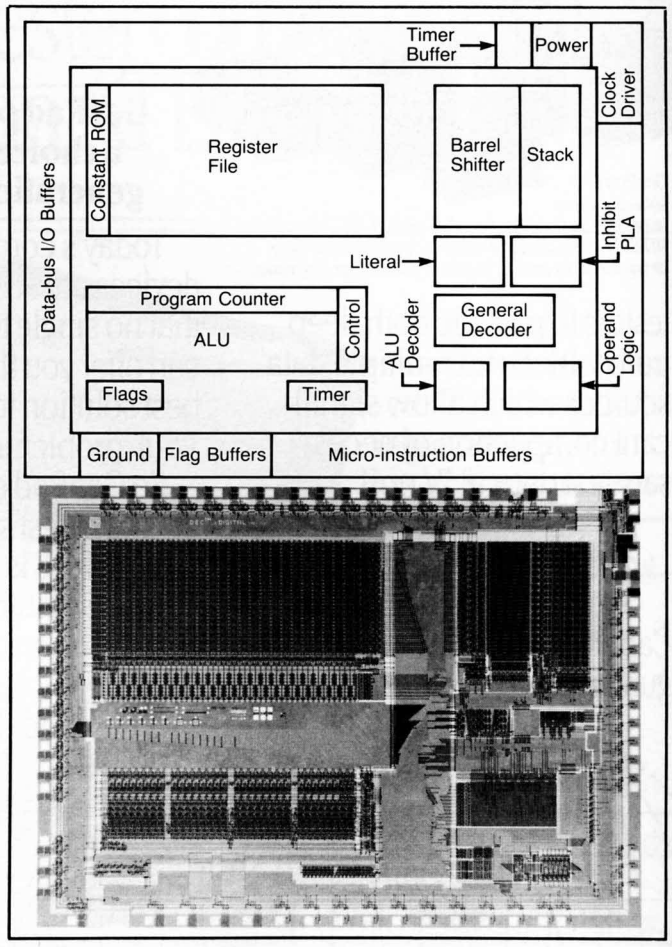


FIGURE 3. Floorplan of the MicroVAX I data-path, and photograph of the chip.

module which consists of many data-path blocks which are joined together via a pair of buses. Each data-path block provides a computational resource for the design. These blocks are designed to be lumped together into a data-path module, to improve layout efficiency. The buses provide a two-operand data-path connection between these elements that makes it convenient to implement the register-instruction set of the VAX architecture in very few microinstructions. All of the elements have a common control interface that lets them be controlled easily by a control word generated by a PLA structure.

The single logical data-path module on the chip is realized physically as three separate but tightly interconnected modules. This design divides the bus into several smaller buses which, with the use of buffering elements, improve the data-transfer time on the buses. It also permits a more flexible arrangement of the floorplan, because the pitch of the ALU/PC block is different from that of the register file and barrel shifter/stack block.

The Layout of the Elements

The basic elements from which the blocks are compiled are laid out and connected to each other by abutting elements on all four sides. The data and control-line connections are located in standard positions to allow this very dense packaging. The data-bus and control-line routing is

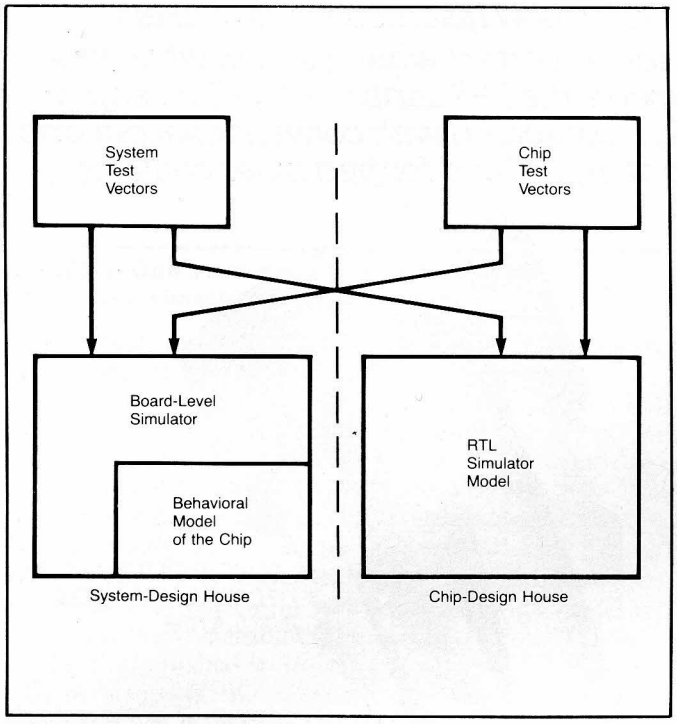


FIGURE 4. An exchange of test vectors couples two simulators.

done directly inside the element, so that the data bus wiring runs in one direction in metal, and the control wiring runs on a polysilicon layer in the other direction. At the edges of the block, control elements provide the clocking and buffering for the blocks. The resulting block-layout density approaches that of manual design.

When the project was started, very few production-worthy elements had already been designed. However, layout was required only for the elements used to make up the blocks. All of these elements were designed by Duane Hook, who was our single layout designer for a period of three months. We have added these elements to our library, and they are now available for future projects.

Interaction between the System-Design Team and the Chip-Design Team

A proposal for the chip was developed by SCI's chip-design team, based on discussions with the system-design team. The latter team supplied design analysis in terms of their knowledge of the VAX architecture and the rest of the board-level system. The chip-design team proposed chip architectures that could be implemented efficiently in MOS. A detailed functional specification was drafted, and the resulting negotiations led to an explicit design specification to prevent any re-design. Once the specification was "cast in concrete," the system hardware-design project leader and the chip-design project leader usually communicated one-to-one.

From the start, both design teams focused on the steps that were necessary to obtain a working chip on the first try. To keep both groups synchronized and to minimize delays, meetings were held every other week (alternating between DEC WEST and SCI). An electronic-mail link established during the project encouraged and improved communications between the two locations. Of course, we also generated and reviewed schematics and design reports.

However, our most important achievement was a simulation-linking strategy that eliminated most of the ambiguities in a functional specification that could cause unpleasant surprises late in the project. Top-down chip-design methods and extensive use of verification tools were required to ensure design correctness. Physical layout had to behave as predicted by the simulations. This requirement made it possible to control the complexity of the design, and helped to eliminate clerical errors.

Simulation Strategy

One of the most difficult tasks in custom-chip development is to specify the chip's functions precisely enough so that the system designer is not unpleasantly surprised by the behavior of the fabricated device. It is extremely difficult to isolate all of the subtle side-effects of a particular chip design; in many cases, the chip is not really wholly defined until after it has been manufactured. At the beginning of this project, both teams believed that simulation could somehow solve this problem. Both parties had proprietary simulation tools, and both had a certain success in using them to validate and debug the behavior of complex digital designs.

The system-design team intended to debug its board-level design and microcode by modeling the full system. This model would include a section that modeled the chip based on its detailed functional specification. The chip-design team intended to debug its product by developing an RTL functional model and then validating the functional simulation. Because the elements had not been debugged beforehand, it was necessary to correlate the results with block-level switch simulation and element-level transistor simulation. The problem was how to get these two simulators to match exactly, so that their results would become the *de facto* chip specification.

A simulation strategy was developed whereby each team would generate test vectors to check its own simulator's models. These vectors would then be exchanged and run on the other model (see Figure 4). This strategy worked out well, and both teams relied on it heavily.

First, a common test-vector format was defined for the chip/system interface. Each team then instrumented its own simulator to be driven by these test vectors. Several benefits were derived from a comparison of discrepancies in the behavior of the two models. At first, ambiguities in the specification came to light and were cleared up. Later, the chip-design team benefited from the early availability of additional test vectors to debug the chip design before committing it to silicon. The system-design team benefited through the refinement of its simulator to match the targeted chips, so that microcode debugging could be done more confidently. As the design of the microcode progressed, more and more test vectors were run on both models, and reinforced correct chip behavior. Thus, the system simulator could actually execute microcode long before the chip was manufactured. Finally, the test vectors and expected responses from the simulations became the nucleus of the chip-evaluation tests. Therefore, when the chips passed these tests, the probability that the system

would perform properly was virtually 100 percent.

Summary

The MicroVAX I Data-path chip has been deemed a success by both DEC and SCI. Its development met the primary goal of developing a custom chip in a very short time. The development contract was signed in January 1983. The operating system was running (with first-pass parts) by August 1983. The use of silicon-compilation tools provided a worthwhile trade-off between chip size and design time. The key to obtaining a working chip on the first pass was the close interaction of the system design team and the chip design team, and the use of verification tools and a comprehensive simulation strategy.

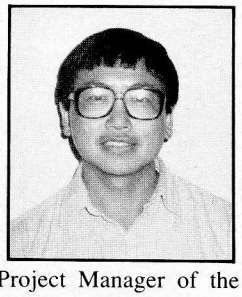
Of course, no project ever runs perfectly. We re-learned the following truisms:

- The tools will never be quite up to expectations when you need them (especially when you are expecting prototypes to stabilize right on schedule).
- The schedule is always optimistic no matter how well things go. □

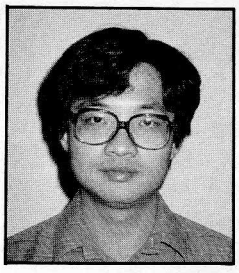
Acknowledgment

We wish to thank the design team at DEC WEST in Bellevue, WA.

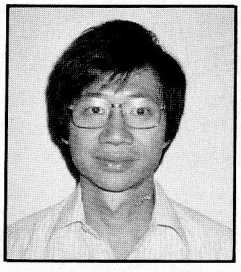
About the Authors



Glenn Louie received the BSEE degree from City University of New York in 1964 and the MS degree in electrophysics from the Polytechnic Institute of New York in 1968. He spent over 10 years at Intel Corp., where he worked on microprocessor peripheral chips and led a small design team in the logic design and RTL simulation of the iAPX-286 microprocessor chip. He joined Silicon Compilers, Inc. in late 1982, where he is Project Manager of the MicroVAX I Data-path chip effort.



Tom Ho received the BSEE degree from UC Berkeley in 1978. After graduation he worked at Intel Corp., where he was project leader for circuit and layout design of the iAPX-286. He joined Silicon Compilers, Inc. in November 1982 where he was responsible for circuit design of the MicroVAX I Data-path chip. Tom is currently involved in the development of various VLSI design tools.



Edmund Cheng is the Vice President of Engineering at Silicon Compilers, Inc. He received the BSEE degree from Ohio University and the MSEE and Ph.D. from California Institute of Technology. For six years he worked in the microcomputer design engineering department at Intel, where he developed the A/D converters for single-chip microcomputers, and managed automotive-engine-control projects.