A RISCy Approach to VLSI

Daniel T. Fitzpatrick, John K. Foderaro, Manolis G.H. Katevenis, Howard A. Landman, David A. Patterson, James B. Peek, Zvi Peshkess, Carlo H. Séquin, Robert W. Sherburne, Korbin S. Van Dyke,
Computer Science Division, University of California at Berkeley

A general trend in computers today is to increase the complexity of architectures along with the increasing potential of implementation technologies. The consequences of this complexity are increased design time, more design errors, inconsistent implementations, and the delay of single chip implementation (Patterson and Ditzel, 1980). The Reduced Instruction Set Computer (RISC) Project investigates a VLSI alternative to this trend. Our initial design is called RISC I.

The judicious choice of a small set of the most often used instructions, combined with an architecture tailored to efficient execution of this set, can yield a machine of surprisingly high throughput. In addition, a single-chip implementation of a simpler machine makes more effective use of limited resources such as the number of transistors, area, and power consumption of present-day VLSI chips (Patterson and Séquin, 1980). Simplicity of the instruction set leads to a small control section, a comparatively short machine cycle, and reduced design cycle time.

Students taking part in a multi-term course sequence designed two different nMOS versions of RISC I. The "Gold" group (Fitzpatrick, Foderaro, Peek, Peshkess, and Van Dyke) designed a complete 32-bit microprocessor, currently being fabricated. The "Blue" group (Katevenis and Sherburne) started from the same basic organization, but introduced a more sophisticated timing scheme so as to shorten the machine cycle and also reduce chip area. (At present, only the data path of this more ambitious design has been completed.) The chips were designed using only "Manhattan" features with the simple and scalable Mead-Conway design rules (fabrication lambda=2 microns).

When we began to design RISC I, we defined the following goals and constraints: (a) find a reasonable compromise between high performance for high-level language programs and a simple single-chip implementation; (b) make the size of all instructions equal to one word, and execute all instructions in one machine cycle; (c) emphasize register oriented instructions, and (d) restrict memory access to the LOAD and STORE instructions. The resulting architecture has 31 instructions in two formats, uses 32-bit addresses, and supports 8-, 16-, and 32-bit data. The most visible impact of the reduced instruction set is that the area dedicated to control dropped from 50% (as in typical commercial microprocessors) to only 6% in RISC I.

The chip area saved by simplified control circuitry was devoted to an extra large set of 32-bit registers. Thus, the processor could allocate a new set of registers for each procedure call, and avoid the overhead of saving registers in memory. By overlapping these "windows" of registers, parameters may be passed by simply changing a pointer.

This so-called "overlapped register window" scheme (Patterson and Séquin, 1981) is largely responsible for the surprisingly good simulation of high-level language programs. Simulations of benchmark programs written in "C" indicate that RISC I can run faster than many commercial minicomputers. Table 1 shows the size and execution of six C programs on RISC I with an assumed machine cycle of 400 nanoseconds. Also in the table are the VAX 11/780 (a 32-bit Schottky-TTL minicomputer with a 200 ns microcycle time) and the Z8002 (a 16-bit nMOS microprocessor with a microcycle time of 250 ns). Even though the Z8002 uses only 16-bit addresses and data while RISC uses 32-bit addresses and data, RISC programs are typically only 10% larger and run about four times faster. The byte-variable length of VAX instructions reduces program size by about a third; on the other hand, every C program that we have run on the RISC simulator has run faster than on the VAX.

Micro-Architecture

The simplicity and regularity of RISC let most instruction executions to follow the same pattern: (1) read two registers, (2) perform an operation on their contents, and (3) store the result in a register. JUMP, CALL, and RETURN instructions add a register (which may be a program counter (PC) register) and an offset, and store the result in the appropriate PC latch. The LOAD and STORE instructions violate the one-cycle execution constraint: to allow enough time for main-memory access, they add the index register and immediate offset during the first cycle, and perform the memory access during the next cycle. In all cases, while the processor executes the first cycle of an instruction, the next instruction is fetched from memory.

The micro-architectures of the two implementations are tailored according to the above characteristics. The CPU can be subdivided naturally into the following functional blocks: the register-file, the ALU, the shifter, a set of Program Counter (PC) registers, the Data I/O latches, the Program Status Word (PSW) register, and control (which contains the instruction register, instruction decoder, and internal clock circuits). Because two operands are required

An earlier version of this article, entitled "VLSI Implementations of a Reduced Instruction Set Computer," was presented at the CMU Conference on VLSI Systems and Computations, October 19-21, 1981, Pittsburgh, PA.

Name	Program Size (bytes) RISC	VAX	VAX/RISC	Z8000	Z8000/RISC	Execution Times (secs) RISC	VAX	VAX/RISC	Z8000	Z8000/RISC
acker	208	120	0.58	238	1.15	3.2	5.1	1.6	8.8	2.8
qsort	644	436	0.68	648	1.01	0.8	1.8	2.3	4.7	5.9
puzzle(sub)	2468	1668	0.68	1612	0.70	4.7	9.5	2.0	19.2	4.2
puzzle(ptr)	2480	1700	0.68	1656	0.7	3.2	4.0	1.3	7.5	2.3
sed	17368	14336	0.83	17500	1.01	5.1	5.7	1.1	22.2	4.4
towers	132	100	0.76	242	1.82	6.8	12.2	1.8	28.7	4.2
Average	3883	3060	0.7±.1	3649	1.1±.4	4.0	6.4	1.7±.4	15.2	4.0±1.2

TABLE 1. RISC program size and execution times relative to a VAX 11/780 and a Z8000.

simultaneously, the register file needs at least two independent buses and a two-port cell design. For speed, the registers are read from dynamically precharged bit lines. This characteristic requires the following basic timing sequence: (1) register READ, (2) arithmetic/logic/shift operations, (3) register WRITE, and (4) bus precharge for next READ. The cycle time is determined by this sequence of operations. For the price of a third bus, phase (4) can be eliminated: as the result is written back into the register file by this extra bus, the two READ buses are precharged for the following READ phase. This simple but robust 3-phase scheme was adopted for the Gold processor. Figure 1 shows the basic organization with buses A,B (READ only) and C (WRITE only). This organization permits simple instruction fetch and execution.

During the Gold RISC I design period, it became clear that a three-bus register cell incurred a significant area penalty. Because a large fraction of the chip area is devoted to the register file, more attention was given to the design of a compact bit cell. Because of its compactness, the classic six-transistor static RAM cell was chosen for the second, more ambitious Blue chip. Reading is done by selectively discharging the precharged bit line buses. Contrary to commerically available static RAMs, no sense amplifiers are used. This imposes a speed penalty, because the bit cell must discharge a high-capacitance bus. However, a two-port reading capability is gained. Writing is done by putting both the data and its complement onto the two buses, as for a typical static RAM. Before reading, the buses must be precharged for proper operation.

The Blue RISC I design (Figure 2) was based on this two-bus, two-port register cell. The reduced cell size allowed significant performance improvement due to the shorter RC delay in the polysilicon control lines running across the data path. The overall chip size was also considerably reduced.

Further improvements were made by allowing register writing to occur in parallel with execution of the next instruction. In effect, each instruction now requires three cycles: (1) instruction fetching and decoding; (2) register read, operate, and temporary latching of result; (3) writing of result into register file. However, in the Blue design, these three operations are pipelined so that a new instruction begins each cycle (except LOAD/STORES).

Both designs multiplexed the address and data pins, because, with current packaging technology we could not afford to use 64 separate lines. Power consumption for the Gold chip is estimated at between 1.2 and 1.9 watts.

Chip Analysis

We have analyzed the completed Gold chip and the data path of the Blue chip to determine whether our design procedure resulted in noticeably different structure or statistics. Table 2 shows the chip-resource allocation for the various functional blocks in the Blue and Gold designs. (Because the Blue design is not completed, we have estimated the figures for the control and I/O portions.) These resources are measured in million square lambda, thousands of transistors, and thousands of rectangles. We can make the following observations based on Table 2:

1) *Control is only 6 to 8%* of the total chip. The next section of this article compares this important figure to those for commercial microprocessors.
2) *The Blue data path is much more compact than the Gold data path*. The register file pitch determined the height of the data path. The shifter, ALU, and PC modules were designed to meet the respective register file pitches in both designs. The two-bus Blue register cell with its smaller pitch resulted in a smaller shifter, ALU, and PCs. The compactness of the cross-coupled static RAM cell let the 138 32-bit registers of the Blue design occupy about half the area of the 78 32-bit registers of the Gold design.

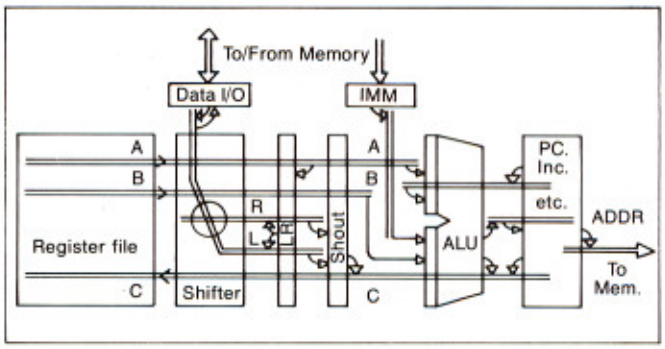

FIGURE 1. The Gold Data-Path.

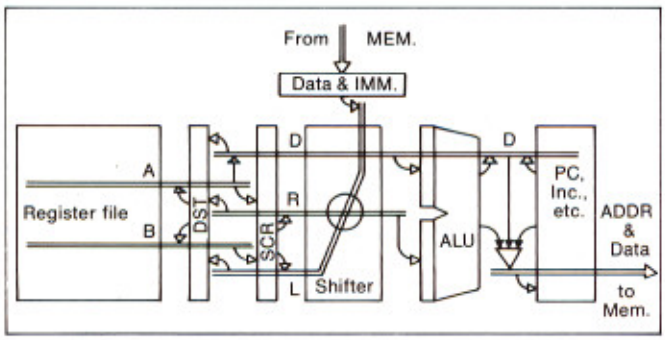

FIGURE 2. The Blue Data-Path.

Functional Block	AREA(MA²) Gold		Blue		FUNCTION (K transistors) Gold		Blue		COMPLEXITY (K rectangles) Gold		Blue	
Registers	7.72	39%	4.12	30%	30.05	66%	26.50	66%	372.2	69%	300.0	65%
Register decoder	1.58	8%	0.87	6%	3.33	7%	3.60	9%	30.8	6%	40.0	9%
Shifter	1.89	10%	0.89	6%	2.63	6%	2.93	7%	39.3	7%	39.4	9%
ALU	1.41	6%	0.44	3%	3.06	7%	2.26	6%	32.3	6%	23.3	5%
PCs	0.88	4%	0.36	3%	1.98	6%	1.64	4%	20.2	4%	16.6	4%
Data I/O logic	0.23	1%	0.10	1%	0.67	2%	0.37	1%	6.5	1%	3.5	1%
Scan In Scan Out	0.14	1%	0.06	0%	0.38	1%	0.14	0%	1.6	0%	1.2	0%
Total DATA PATH	13.84	69%	6.84	50%	42.09	95%	37.44	94%	502.8	93%	424.0	92%
PLAs	0.23	1%			0.84	2%			5.5	1%		
Latches	0.42	2%			0.83	2%			8.5	2%		
Routing	0.47	2%			-	-			6.0	1%		
Scan In Scan Out	0.04	0%			0.10	0%			0.6	0%		
Total CONTROL	1.16	6%	~1.16	~8%	1.77	4%	~1.66	~4%	20.6	4%	~20.0	~4%
Routing	2.05	10%	~2.75	~20%	-	-	-	-	3.8	1%	~5.0	~1%
Pads	1.13	6%	~1.10	~8%	0.84	2%	~0.84	~2%	9.9	2%	~10.0	~2%
Scan In Scan Out	0.73	4%	~0.65	~5%	0.08	0%	~0.06	~0%	1.2	0%	~1.0	~0%
Total I/O	3.91	20%	~4.50	~33%	0.92	2%	~0.90	~2%	14.9	3%	~16.0	~4%
Unused area	1.09	6%	~1.20	~9%	-	-	-	-	-	-	-	-
TOTAL CPU	20.00	100%	~13.70	100%	44.42	100%	~40.00	100%	538.3	100%	~460.0	100%

TABLE 2. Area, transistors, and rectangles per Blue and Gold functional block.

3) *SCAN IN/SCAN OUT (SISO) is less than 5% of the chip*. The SISO technique improves chip testability by allowing access to each state bit in a module. The flip-flops are connected together as a large shift register, allowing serial reading and writing. The Gold chip has complete SISO on the shifter, ALU, and control. It also has SISO on a portion of the PCs. Because we had spare pins, we used 11 pins for SISO. The 11 SISO pads are responsible for 4% of the chip area.
4) *In each functional block, the number of transistors has a high correlation to the number of rectangles*. There are about ten rectangles for each transistor.
5) *Area utilization varies widely*. The highly regular data path, consisting mainly of carefully optimized cells, contains more than 90% of the transistors and rectangles but only half to two-thirds of the area. On the other hand, the I/O area, which is dominated by random interconnections, contains 20 to 30% of the area but less than 4% of the transistors or rectangles.

Tools

We are certain that appropriate software is the key to successful VLSI design. Although we have ambitious plans for a sophisticated design environment (Newton, *et al.* 1981), at the moment we have to work with a rather small subset. We used more than a dozen programs to design our chips. Among those, we felt that the tools listed in Table 3 were invaluable. Table 4 lists other tools which were particularly helpful for our project.

The "glue" that held all these various tools together and produced a cohesive design environment was provided by the UNIX operating system (4th Berkeley Software Distribution) (Joy and Fabry 1981) running on a DEC VAX 11/780.

We started the designs with an ISPS description of RISC I and a block diagram similar to Figure 1. (The ISPS description was not very useful, because it does not run on the VAX.) The logic, circuits, and initial layouts were designed on paper and then entered into Caesar. As the designers became more comfortable with Caesar, they used it to do the initial layout. Caesar converted the graphic description into CIF, the format in which the chip layout had to be submitted for fabrication. Caesar also let us use CIF-based tools such as CIFPLOT and DRC. The information needed to run STAT, SPICE, and POWEST was extracted from this same CIF by MEXTRA. After the bottom-level modules were designed, we used SLANG to describe the chip completely at the functional and logical levels. To reveal errors, we then ran RISC diagnostic programs on this description. The SLANG description was also used to specify many of the remaining connections in the chip and to drive the PLA tools to produce the PLAs for RISC automatically. Howard Landman acted as a "roving critic," scanning for errors that were overlooked by the design tools.

The final step was to use SLANG to compare the original description with the final masks. MEXTRA was first used to extract from the masks the information needed to drive MOSSIM. We then used SLANG to compare interactively, every half clock phase, the values of hundreds of nodes in

PROGRAM	TYPE	AUTHOR	INSTITUTION
CAESAR	color graphics editor	John Ousterhout	U.C. Berkeley
CIFPLOT	plot of mask layers	Dan Fitzpatrick	U.C. Berkeley
MEXTRA	Manhattan circuit extractor	Dan Fitzpatrick	U.C. Berkeley
SLANG	multi-level simulator	John Foderaro	U.C. Berkeley
MOSSIM	awitch-level simulator	Randy Bryant/ Chris Terman	M.I.T.
DRC	layout-rules checker	Clark Baker	M.I.T.

TABLE 3.

PROGRAM	TYPE	AUTHOR	INSTITUTION
MKPLA	PLA generator	Howard Landman	U.C. Berkeley
PRESTO	PLA minimizer	Sheng Fang/ Richard Newton	U.C. Berkeley
EQNTOTT	PLP equation translator	Robert Cmelik	U.C. Berkeley
SPICE	circuit simulator	Donald Pederson/ Richard Newton	U.C. Berkeley
STAT	electrical-rules checker	Forrest Baskett	Stanford/ Xerox PARC
POWEST	power estimator	Robert Cmelik	U.C. Berkeley

TABLE 4.

both the SLANG description simulation and the MOSSIM simulation. We ran about a dozen diagnostic programs and found several errors.

Design Time and Regularity

Table 5 compares several design metrics for RISC to those of some commerical microprocessors. The information for the Z8000 and MC68000 comes from Frank and Sproull (1981), and the information on the 432 comes from Lattin1, *et al.* (1981) and Lattin2, *et al.* (1981). The data provides quantitative support for some of our conclusions listed above.

The Gold chip is larger than most Mead-Conway designs, but is still comparable to the latest microprocessors. It is about 6% larger than the 43203. On the other hand, the Blue chip may be 25% smaller than the 43203. The most significant difference is that the instruction set of RISC I leads to a much smaller control section than that of the comparison processors; control occupies less than 10%, compared to about 50% for the other chips.

The smaller control section leaves proportionally more area for the structured data path and, in particular, for the very regular and compact register file. Thus, the overall regularity of the chip is increased. The "regularity factor" (Lattin 1979) is defined as the total number of devices on the chip, excluding ROMs, divided by the number of drawn transistors. The table entries for the number of devices in the Z8000 and MC68000 are estimates, but the entries for the 432 and RISC are actual measurements. RISC is shown to be 2 to 5 times more "regular."

Increased chip regularity was a key factor leading to a short design cycle. The elapsed time was considerably shorter for RISC; the effort in man months was about five times less. Control was by far the most time-consuming part of design and layout. Because the control section was reduced from half the chip to less than 10%, the design effort was greatly reduced.

Another factor of the reduced design cycle is our integrated design environment. Special credit goes to Caesar, the Manhattan-based graphic layout editor. Caesar was developed by Ousterhout (see the article in this issue), working closely with the chip designers.

All the various tools for graphic editing, checkplotting, design-rule checking, layout-rule checking, architectural simulation, and switch-level simulation were at the designers' fingertips, and ran on the same machine. This reduced the "psychological overhead" associated with the use of such tools.

Other factors affecting layout time were the use of the simplified Mead-Conway design rules and the restriction to Manhattan features. Design time was saved by concentration on correct function rather than on highest performance. SPICE simulations were used only in some of the most critical paths. Extensive switch-level simulation was done for the entire chip from a file obtained by automated extraction from the mask geometry data base. Certainly the reduced design time is due partly to the fact that, as academics, we only had to deal with a set of rather loose, self-imposed constraints. For example, our master clock is generated off-chip, and the Gold chip has a rather poor external bus interface. The opcode selection was kept flexible to the very end. (These codes were modified the day before the design was submitted for mask generation.) An industrial product can rarely afford such luxury.

	Zilog	Motorola	Intel iAPX-432			RISC I	
	Z8000	68000	43201	43202	43203	Gold	Blue
Total devices	17.5k	68k	110k	49k	60k	44k	37k+?
Total ROM	17.5k	37k	44k	49k	44k	44k	?
Drawn devices	3.5k	3k	5.6k	9.5k	5.7k	2k, 1.6k[1]	x+?
Regularization factor	5.0	12.1	7.9	5.2	7.7	21.7, 27.5[1]	?
Size of chip							
(Dimension in mils)	238x251	246x281	318x323	366x313	358x326	406x350	~400x~215
(Area in square mils)	60k	69k	103k	115k	117k	124k	~86k
Metal pitch							
(microns)	12	11	11	11	11	12	12
Size of control[2]							
(Area in square mils)	37k	42k	67k	45k	47k	7k	~7k
Percent control	53%	62%	65%	39%	40%	6%	~8%
Elapsed time to first silicon (months)	30	30	33[3]	33[3]	21[3]	17[4]	17+?[4]
Design time (man months)	60	100	170[3]	170[3]	130[3]	30/2[5]	(30+?)/2[5]
Layout time (man months)	70	70	90	100	50	12	5+?

[1] There are two ways to count drawn transistors; the pessimistic approach counts every transistor in a cell even if it derived from simple modifications to a basic cell. The optimistic approach only counts the transistors that were changed. For the Gold chip the difference is the transistor count for the register decoders. The optimistic count saves 433 drawn transistors thereby increasing the regularity factor.

[2] We estimated these sizes from the photomicrographs of the commercial chips.

[3] Data provided by Justin Rattner of Intel.

[4] We counted elapsed time from the beginning of the first class to the end of the last class plus three months which should be the time for fabrication (we hope).

[5] Since the designers also did layout this is a somewhat fuzzy distinction. All work before 1/1/81 is considered design and we have included circuit design as part of layout.

TABLE 5. Design metrics for Z8000, MC68000, iAPX-432, and RISC I.

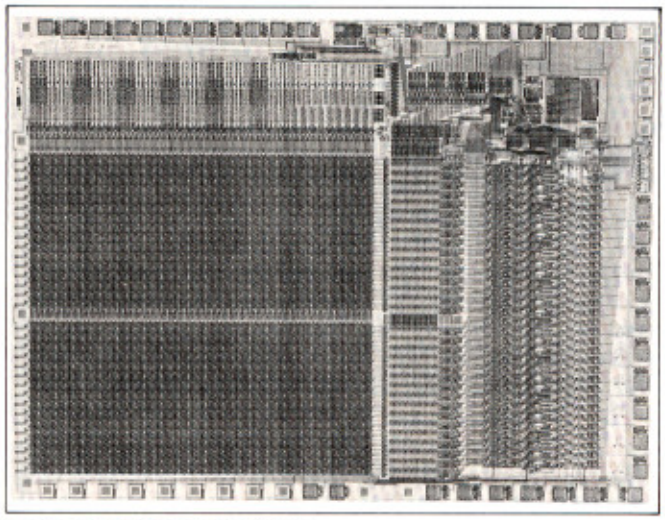

Photo of RISC chip checkplot.

Conclusions

The RISC Project has had a synergistic effect on research at Berkeley in architecture, VLSI, and CAD. Useful tools were often created in response to specific needs. For example, a special extractor for Manhattan geometry was formulated, because the older extractor, able to handle general geometry, would have taken too long to find all 44,000 transistors on the Gold chip. As a result of this synergism, in the last six months, our design environment has improved dramatically.

The design-tool performance improvements due to the restriction to Manhattan designs were well worth the inconveniences they caused in layout. We found only small areas in each chip where non-Manhattan geometry could save space. Figure 3 shows a mask of the Gold design.

Although we realize that more work is needed to turn RISC into a full-fledged microcomputer, we also believe that the most difficult and time-consuming part of the task is done. These results can be duplicated by industry. Reduction in elapsed design time, reduction in manpower, and high performance are available for those who are willing to take calculated risks.

Acknowledgements

The RISC project was aided by several people at Berkeley and in the Bay Area. We would like to thank them all, but special thanks to the following individuals. John Ousterhout created Caesar, the main interface of the designers. The reliability and quality of this graphics editor and his responsiveness to our needs are major reasons for our reduced design time. Richard Newton let us use his graduate class to resolve RISC-related issues. In the Berkeley community, we also want to thank, Bob Cmelik, Sheng Fang, Richard Newton, and Donald Pederson for the use of their tools. We would especially like to thank the people in the ARPA-VLSI community who shared their tools: Randy Bryant and Chris Terman for the switch-level simulator MOSSIM, and Clark Baker for the layout-rule checker DRC. In addition to these tools from MIT, from Stanford we received STAT, a static electrical-rule checker created by Forrest Baskett. We gratefully acknowledge helpful discussions with Osamu Tomisawa about MOS circuit design and processing. Jim Beck, Bob Cmelik, and Robert Hyerle provided valuable chip-testing information and ideas. We would also thank the visitors from industry who gave us valuable design suggestions: Les Credele from Motorola, Dick Lyon from Xerox PARC, and Peter Stoll from Intel. Thanks also to Bob Fabry, Richard Fateman, Bill Joy, and Bob Kridle for providing the computing environment we needed to complete our designs. We thank Danny Cohen and Lee Richardson of the MOSIS group, and Alan Bell and Al Paeth of Xerox PARC, for fabricating our chips. Finally, we thank Duane Adams and DARPA for providing the resources that let universities attempt high-risk projects.

This research was sponsored by the Defense Advance Research Projects Agency (DoD), ARPA Order No. 3803, and monitored by Naval Electronic System Command under Contract No. N00039-78-G-0013-0004.

References

Frank, E.H. and R.F. Sproull. July 1981. "An Approach to Debugging Custom Integrated Circuits," *Carnegie-Mellon Computer Science Research Review 1979-80*.

Joy, W.N. and R.S. Fabry. May 1981. "Berkeley Software for UNIX on the VAX," U.C. Berkeley, Computer Science Department internal memo.

Lattin1, W.W., J.A. Bayliss, D.L. Budde, S.R. Colley, G.W. Cox, A.L. Goodman, J.R. Rattner, W.S. Richardson and R.C. Swanson. February 1981. "A 32b VLSI Micromainframe Computer System," *Proceedings of the IEEE International Solid-State Circuits Conference*.

Lattin2, W.W., J.A. Bayliss, D.L. Budde, J.R. Rattner, and W.S. Richardson. Second Quarter 1981. "A Methodology for VLSI Chip Design," *LAMBDA*, Vol. II, No. 2.

Lattin, W.W. January 1979. "VLSI Design Methodology: The Problems of the 80's for Microprocessor Designs," *First Caltech Conference on VLSI*.

Newton, A.R., D.O. Pederson, A.L. Sangiovanni-Vincentelli, and C.H. Séquin. July 1981. "Design Aids for VLSI: The Berkeley Perspective," *IEEE Transactions on Circuits and Systems*.

Patterson, D.A. and C.H. Séquin. May 1981. "RISC I: A Reduced Instruction Set VLSI Computer," *Proceedings of the Eighth International Symposium on Computer Architecture*.

Patterson, D.A. and D.R. Ditzel. October 15, 1980. "The Case for the Reduced Instruction Set Computer," *Computer Architecture News*, Vol. 8, No. 6.

Patterson, D.A. and C.H. Séquin. February 1980. "Design Considerations for Single-Chip Computers of the Future," *IEEE Transactions on Computers*, Vol. C-29, No. 2.

Authors

A project as large as that of the RISC chip entails the cooperation of a large number of individuals. The folks at Berkeley were kind enough to send us this informal group portrait of the RISC design team. First row: Korbin Van Dyke, Osamu Tomisawa, James Peek, Prof. David Patterson, Prof. Carlo Séquin, Peter Kessler; second row: Robert Sherburne, Manolis Katevenis, Prof. John Ousterhout, Ralph Campbell, Richard Piepho, Daniel Fitzpatrick, Daniel Halbert, John Foderaro; third row: Robert Cmelik, Robert Hyerle, Paul Hansen, Helen Davis, Michael Shiloh, Scott Baden, and Howard Landman.