A Retrospective on "MIPS: A Microprocessor Architecture"

Thomas R. Gross
ETH Zurich

Norman P. Jouppi
Google

John L. Hennessy
Stanford University

Steven Przybylski
Verdande Group

Chris Rowen
Cadence Design Systems

The MIPS project started in early 1981. At that time, mainstream architectures such as the VAX, IBM 370, Intel 8086, and Motorola 68000 were fairly complex, and their operation was controlled by microcode. Designers of high-performance implementations discovered that pipelining of complex-instruction-set computer (CISC) machines, especially the VAX, was hard; the VAX 11/780 required around 10 processor cycles to execute a single instruction, on average. Tradeoffs that had been made when there were few resources available for implementation (for example, dense instruction sets sequentially decoded over many cycles by microcode) ended up creating unnecessary constraints when more resources became available. For example, as transistors were replacing the core, the cost of memory dropped much faster than logic—favoring a slightly less dense program encoding over logic-intensive interpretation. Also, the design of instruction sets did not assume the use of an optimizing compiler. CISC instruction features such as arithmetic operations with operands in memory were especially difficult to pipeline and did not take advantage of compiler capabilities (such as register allocation) that could provide more efficient execution.

One area of architecture research in the early 1980s was to create even more complex CISC architectures. Notable examples of these include high-level language machines such as Lisp machines, and later the Intel 432. However, the approach taken in the MIPS project as well as the RISC project at the University of California, Berkeley, was to design simpler instruction sets that did not require execution of microcode. These simpler instructions were a better fit for an optimizing compiler's output—more complex operations could be synthesized from several simple operations, but in the common case where only a simple operation was needed, the more complex aspects of a CISC instruction could be effectively optimized away. In the case of the MIPS project, we emphasized ease of pipelining and sophisticated register allocation, whereas the Berkeley RISC project included support for register windows in hardware.

The quarter before the MIPS project started, Stanford University had offered for the second time a (graduate) class on VLSI design, based on the approach developed by Carver Mead and Lynn Conway.[1] One key idea was that the design rules could be specified without close coupling to the details of a specific process. Most of the MIPS project team members had taken this class. Another difference was that in VLSI, tradeoffs differ from those made in multiboard machines built from hundreds of small-, medium-, and large-scale integration parts like the VAX 11/780. In a multiboard machine, gate transistors were expensive (for example, only four 2-input gates per transistor-transistor logic package), whereas ROM transistors were relatively cheap (for example, kilobits of ROM per package). Much of this was driven by pinout and other limitations of DIP packaging. This naturally favored the design of microcoded machines, in which control and sequencing signals could be efficiently stored as many bits per package instead of being computed by low-density gates requiring many packages.

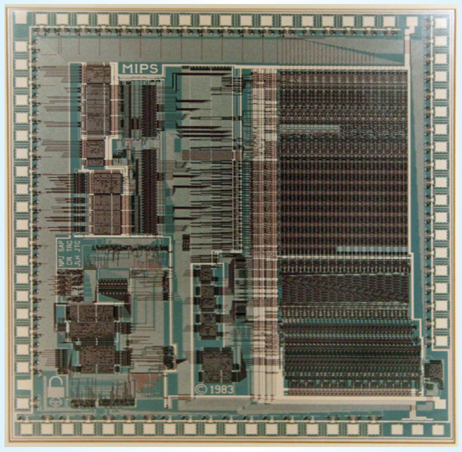

Figure 1. Die photo of MIPS processor.

However, in VLSI, a transistor has approximately the same cost no matter what logical function it is used for, which favors the adoption of direct control as in early RISC architectures instead of microcode. Another difference with widespread adoption of VLSI was that the topology of the wiring was important. In 1981, we had only a single level of metal, so long wires had to be largely planar. This also favored simple and regular designs. Finally, in a VLSI design it was clearly zero-sum: we had a maximum chip size available, so any feature put in would have to justify its value and would displace a less valuable feature. These constraints brought a lot of clarity of focus to the design process.

Given this context, we decided to build a single-chip high-performance microprocessor that could be realized with the fabrication technology available to university teams—a 4-μm single-level metal nMOS process that put severe limitations on the design complexity and size. (The DARPA MOSIS program brokered access to silicon foundries so that students and researchers in universities could obtain real silicon parts.) High performance at that time meant that the processor could execute more than 1 million (meaningful) instructions per second (MIPS), and the design target was 2 MIPS (that is, a clock rate of 4 MHz). Relative to other processor designs of this era, a board powered by a MIPS processor could deliver the same performance as a cabinet-sized computer built with lower levels of integration. At the time we wrote our paper in the fall of 1982,[2] the architecture and microarchitecture were defined, compilers for C and Pascal were written, and several test chips covering different parts of the design had been sent out for fabrication.

Developments After Publication

The MIPS design was completed in the spring of 1983 and sent out for fabrication.[3] At about the time the design was sent for fabrication, the Center for Integrated Systems at Stanford University developed a 3-μm VLSI fabrication process and used the MIPS design (shrunk optically) to tune its process. In early 1984, we received working 3-μm MIPS processors from MOSIS that operated at the speed we had expected for the design at 4 μm. Figure 1 shows the die photo. These processors were run on a test board developed by our fellow grad student, Anant Agarwal, and on 20 February 1984, the first program (8queens) was run to completion. In mid-1984, MIPS Computer Systems was founded. It designed a completely new processor that was heavily influenced by the Stanford MIPS design and was sold as the MIPS R2000. A final evaluation of the Stanford MIPS architecture was published in 1988.[4]

Reflections

Several papers described the implementation and testing of the MIPS processor[5] and discussed the design decisions that turned out to be right and those that deserve to be reconsidered. More than 30 years later, at a time when microprocessors contain multiple independent processing units (cores) and more than 5 billion transistors, a few of these have passed the test of time.

MIPS Design

As part of the MIPS project, we developed various CAD tools, such as tools for programmable logic array synthesis and timing verification.[6] These tools were driven by the designer's requirements and allowed a small team of students and faculty to complete the processor design in a timely manner.

We also invested heavily in compilers beyond the immediate needs of the MIPS processor development. Fellow student Fred Chow designed and implemented a machine-independent global optimizer, and fellow student Peter Steenkiste developed a Lisp compiler to investigate dynamic type checking on a processor that did not provide any dedicated hardware support for this task.

Personal workstations were a novelty at that time, and the MIPS project members were the first to enjoy workstations in their offices. These workstations let us support the implementation with novel tools and interactively experiment with different compiler and hardware optimizations. RISC designs emphasize efficient resource usage and encourage designers to employ resources where they can contribute the most; this RISC strategy drove many of the major design decisions (for example, to abandon microcode in favor of simple instructions, or to use precious on-chip transistors for frequent operations and leave other operations to be dealt with by software).

Hardware/Software Coupling

The MIPS project invested in software early on; the compiler tool chain worked before the design was finalized. We made several key microarchitecture decisions (including the structure of the pipeline) after running benchmark programs and assessing the impact of proposed design changes.

The MIPS project also emphasized tradeoffs across layers as defined by then-prevalent industry practice. Two sample design decisions illustrate this aspect: the MIPS processor uses word addresses, not byte addresses. The reasoning was that byte addressing would complicate the memory interface and slow down the processor, that bytes aren't accessed that often, and that an optimizing compiler could handle any programming language issues. This decision was probably correct for research processors (and saved us from debating if the processor should be big-endian or little-endian) and illustrates the freedom afforded by looking beyond a single layer. But all subsequent descendants supported byte addressing (of course, by that time, VLSI had advanced to CMOS with at least two levels of metal, so byte selection logic was easier to implement). However, word alignment of word accesses as in MIPS was still retained, unlike in minicomputers.

Sometimes the absence of interlocks on MIPS is seen as a defining feature of this project. However, it was a tradeoff, based on the capabilities of the implementation technology of the time. To meet the design goal of a 4-MHz clock, the designers had to streamline the processor, and still most of the critical paths involved the processor's control component.[7] For the MIPS processor, it made sense to simplify the design as much as possible; later descendants, with access to better VLSI processes, made different decisions. The absence of hardware interlocks (to delay an instruction if one of the operands wasn't ready) was a tradeoff between design complexity, critical path length, and software development costs. In addition to clever circuit design and novel tools, this ability to make hardware/software tradeoffs was a key factor for success.

The team wanted to pick a name for the project that emphasized performance. About nine months earlier, the RISC project at UC Berkeley had started, so we needed a catchy acronym. "Million instructions per second" (MIPS) sounded right, given the project's goals, but this metric was also known as the "meaningless indicator of processor speed." So, we settled on "microprocessor without interlocked pipeline stages."

The Mead/Conway approach to VLSI design emphasized a decoupling of architecture and fabrication. No longer was a chip design tied to a specific (often proprietary, in-house) process. The MIPS project demonstrated that a high-performance design could be realized in this framework and supported the view that VLSI design could be done without close coupling to a proprietary process. Vertical integration—that is, design and fabrication in one company—might offer benefits, but so does the separation of fabrication and design. The MOSIS service was an early (nonprofit) experiment; a few years after the end of the MIPS project, commercial silicon foundries started to offer fabrication services and allowed the creation of "fabless" semiconductor companies.

Benchmarks

The MIPS project used a quantitative approach to decide on various features of the processor and therefore needed a collection of benchmark programs to collect the data needed for decision making. In retrospect, the benchmarks we used were tiny and did not include any significant operating systems code. Consequently, the designers focused on producing a design that delivered performance for compiled programs but paid less attention to the operating system interface and the need to connect the processor to a memory hierarchy. At conferences, benchmark results started to become important, and the MIPS paper (as well as papers by other design groups, such as the UC Berkeley RISC project[8]) presented empirical evidence. Stanford University published the set of benchmarks used by the MIPS project (the Stanford Benchmark Suite), and despite their limitations, they were in use (at least) 25 years later to explore array indexing and recursive function calls. About five years later came the founding of the SPEC consortium, which eventually produced a large body of realistic benchmarks. In 2011, about 30 years later, the first ACM conferences initiated a process that lets authors of accepted papers submit an artifact collection (the benchmarks and tools used to generate any empirical evidence presented in a paper).[9] The MIPS project cannot claim credit for these developments, but it emphasized early on that end-to-end performance, from source program to executing machine instructions, is the metric that matters.

The Stanford MIPS project was an important evolutionary step. Later RISC architectures such as MIPS Inc. and DEC Alpha were able to learn from both our mistakes and successes, producing cleaner and more widely applicable architectures, including integrated floating-point and system support features such as TLBs. All new architectures that appeared afterward (since 1985) incorporated ideas from the RISC designs, and as the concern for resource and power efficiency continues to be important, we expect RISC ideas to remain relevant for processor designs. And, finally, this MIPS paper was also an important step in the transition of the SIGMICRO Annual Workshop on Microprogramming to the IEEE/ACM International Symposium on Microarchitecture.

References

1. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

2. J. Hennessy et al., "MIPS: A Microprocessor Architecture," *Proc. 15th Ann. Microprogramming Workshop*, 1982, pp. 17–22.
3. C. Rowen et al., "MIPS: A High Performance 32-bit NMOS Microprocessor," *Proc. Int'l Solid-State Circuits Conf.*, 1984, pp. 180–181.
4. T.R. Gross et al., "Measurement and Evaluation of the MIPS Architecture and Processor," *ACM Trans. Computer Systems*, Aug. 1988, pp. 229–258.
5. S. Przybylski, "The Design Verification and Testing of MIPS," *Proc. Conf. Advanced Research in VLSI*, 1984, pp. 100–109.
6. N. Jouppi, "TV: An NMOS Timing Analyzer," *Proc. 3rd CalTech Conf. VLSI*, 1983, pp. 71–85.
7. S. Przybylski et al., "Organization and VLSI Implementation of MIPS," *J. VLSI and Computer Systems*, vol. 1, no. 2, 1984, pp. 170–208.
8. D.A. Patterson and C.H. Sequin, "RISC-I: A Reduced Instruction Set VLSI Computer," *Proc. 8th Ann. Symp. Computer Architecture*, 1981, pp. 443–457.
9. S. Krishnamurthi, "Artifact Evaluation for Software Conferences," *SIGPLAN Notices*, vol. 48, no. 4S, 2013, pp. 17–21.

Thomas R. Gross is a faculty member in the Computer Science Department at ETH Zurich. Contact him at thomas.gross@inf.ethz.ch.

Norman P. Jouppi is a distinguished hardware engineer at Google. Contact him at jouppi@acm.org.

John L. Hennessy is president emeritus of Stanford University. Contact him at hennessy@stanford.edu.

Steven Przybylski is the president and principal consultant of the Verdande Group. Contact him at sp@verdande.com.

Chris Rowen is the CTO of the IP Group at Cadence Design Systems. Contact him at rowenchris@gmail.com.