Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

Microprocessor Hits and Misses

Panel at Hot Chips Symposium Reviews 20 Years of Successes and Failures

Edited By Tom R. Halfhill {10/20/08-01}

Designing a new microprocessor is risky business, it turns out. Most projects don't achieve anywhere near the success anticipated by their designers. Riskier yet is creating a new microprocessor architecture, which might be an even bigger gamble than subprime loans. The failure rate of new processors and architectures is alarmingly high—and there are no government bailouts.

This year marked the 20th anniversary of the Hot Chips Symposium at Stanford University in Palo Alto, California, sponsored by the IEEE Technical Committee on Microprocessors and Microcomputers. To celebrate, the organizers invited six industry experts to join a discussion panel: "Ready, Fire, Aim—20 Years of Hits and Misses at Hot Chips." They reviewed new microprocessors and architectures presented at the symposium since 1989 and attempted to sort out the successes and failures. Panel members included the following:

Nathan Brookwood, founder and principal analyst of Insight64, a technology-analysis firm

Dave Ditzel, vice president of Hybrid Parallel Computing at Intel and formerly CEO of Transmeta

John R. Mashey, founder and principal consultant at Techviser, formerly a microprocessor architect at Silicon Graphics and MIPS Technologies

David Patterson, professor of computer science at the University of California, Berkeley, and coauthor (with John

PHOTO COURTESY OF HOT CHIPS SYMPOSIUM

Members of the "Hits and Misses" discussion panel at the Hot Chips Symposium were (from left): Howard Sachs, Telairity; David Ditzel, intel; Michael Slater, Webvanta; Nathan Brookwood, Insight64; John R. Mashey, Techvisor; and David Patterson, University of California at Berkeley. (Note: To reproduce an acceptable photograph of the panel, MPR assembled a composite image from different frames of a video recording.)

L. Hennessy) of *Computer Architecture: A Quantitative Approach.*

Howard Sachs, president and CEO of Telairity

Michael Slater, CEO of Webvanta, former vice president of Adobe Systems, and founder of *Microprocessor Report*

The session was chaired and moderated by Nick Tredennick, a technology analyst for Gilder Publishing and formerly an engineer at Altera, IBM, and Motorola. (Tredennick was also a member of the *MPR* editorial board for many years.)

To prepare for the panel discussion, each panelist gave a brief presentation. This article omits those presentations but includes the transcript of the lively discussion that followed. *MPR* has lightly edited the transcript for clarity and has added comments and article references to help put some remarks into context.

PHOTO COURTESY OF HOT CHIPS SYMPOSIUM

Microprocessor Report **founder Michael Slater made a rare reappearance at Hot Chips to participate in the "Hits and Misses" discussion panel.**

Three Weeks to Design the x86

Patterson: When you look at the history of computer architecture, what's our batting average? It seems like the disasters are so easy to find. It's hard to find the successes.

Mashey: Well, consider this. The IBM 360 was designed in the early '60s. There is still an upward compatible thing being built. I actually wrote code in 1970 that is still running today—assembler code—and it does instruction modification!

[laughter]

Tredennick: So who's done the most damage to computer engineering over the last 20 years? Yes, which one of you guys, which one of us?

Brookwood: I think Ditzel.

[laughter]

Tredennick: So who's wasted the most money over the last 20 years?

Patterson: It's not me, I'll tell you that.

Slater: It depends on who has the most money to waste.

Sachs: It's probably between Dave and me, I suspect. And we're not going to tell, right?

Ditzel: Howard didn't know about all the other secret gallium arsenide programs at Sun that never were announced. One of the things I was going to put on the slides was I remember vividly a giant all-hands Sun meeting where Eric Schmidt, now CEO of Google, got up and announced that Sun would be the first company to ship a gallium arsenide processor.

Patterson: You guys may not know your computer history, but Convex successfully shipped a gallium arsenide computer. They made it work, and it sold, and people bought it. So, I guess, better engineers were able to make them work.

[laughter]

Mashey: Whoa, whoa, whoa! Be fair, be fair!

Brookwood: We were talking about the [Intel] 432 [microprocessor]. These programs do have human consequences. And one of the side effects of Intel focusing on the 432 is that they took a young engineer, pretty much out of college, and threw him on a program to see what they could do to the [Intel] 286 to keep it going for a couple more years—

Patterson: That's not quite right.

Brookwood: Oh?

Patterson: No, what happened was this. Gordon Moore, of Moore's law, was and is pals with Gordon Bell. *[Editor's note: Bell, now at Microsoft, was vice president of research and development at DEC for many years.]* And Gordon could see that Intel was eventually going to be a competitor, but they got along. Gordon Moore saw—and this was when they [Intel] were doing the 8080 and 8-bit instruction sets—that probably the next instruction set they did was going to last forever. So he hired a bunch of smart people, many from Carnegie Mellon, and sent them up to Oregon, and they were going to do the next great thing. And it was a stealth project. If you applied for a job, they wouldn't tell you what you were working on. But it was going to change the world, and you should take it.

So after a couple of years, they had this thing that was going to change the world, but it was going to be late. So Gordon Moore saw it coming, set it all up, and then the guys took too long to do the 432. So he had an emergency to design a 16-bit instruction set. They had three months to come up with the instruction set, because they had to have a product to market. And that product is x86.

Tredennick: John? *[Editor's note: audience member John Wharton, former Intel engineer and* MPR *analyst.]*

Wharton: It was three weeks. It was not three months.

Patterson: Three *weeks?* It was three something.

Wharton: Eight man-weeks went into the development of the x86 spec, and they didn't implement the full spec when they actually introduced it.

Patterson: Yeah, but it was a—

Wharton: …filled in the gap with the 432…

Patterson: …They had a careful plan…and they had an emergency, and they needed an instruction set so they could ship a chip. And that's the one that's lasted, and will last for the rest of our lives.

Brookwood: I was talking about the 32-bit [x86] extensions, which is what [Intel engineer, now Senior VP Patrick] Gelsinger did. And that's because all the good people at Intel, all the hotshots, were off on the 432. And so they said, "Here Gelsinger, you go and do this, and see how you can keep the 286 going." And of course, you know, that turned out to be the salvation, and Pat has been able to use that forever.

VLIW: Success or Failure?

Audience question: VLIW and Itanium took a lot of heat. One of the problems that was mentioned was relying too much on the compiler and general software issues that sort of sank it. Yet, at the same time, in your last presentation, you said that multicore is the greatest success of the last 20 years. So can you justify, or do you think, that software issues have in fact been solved, and software programming issues have been solved, for multicore? Or is there a danger that it might end up like VLIW because it sinks in software?

Sachs: You know, I'm taking issue with this. Just because Nate jumped all over VLIW doesn't mean it's right. *[Editor's note: Earlier in the session, Nathan Brookwood's solo presentation criticized VLIW.]* Now, I think Dave and I—I don't know about the rest of you guys, but I think I can say "you"—for many years we've worked together on VLIW. And I really think it's a nice technology. Now what Itanium did, I can't speak for. But I think VLIW is a nice technology.

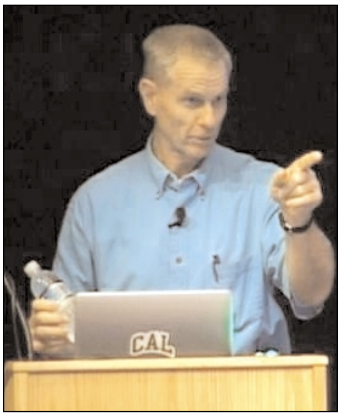

PHOTO COURTESY OF HOT CHIPS SYMPOSIUM

Nick Tredennick moderated the "Hits and Misses' discussion panel at the Hot Chips Symposium

Patterson: Well, you shouldn't call it "VLIW" [very long instruction words]. You should call it "LIW" [long instruction words]. Right? You guys like three instructions, right? I mean, that's kind of the tech…It's been very successful in the DSP market. And I think it's like three, right? Three or four?

Ditzel: Three or more. It depends on what you're doing. If you're doing graphics and video processing and other things, you can do it wider and use it, and it's one of the most power-efficient architectures out there. It's why DSPs are generally built with that form. I think you shouldn't brand all implementations as bad just because there's one implementation you don't like.

Mashey: Oh, there were more than that. Right? *[laughter]* No, no, I think…who were the guys?

Brookwood: Multiflow?

Mashey: Multiflow. OK. The really wide—the LIWs that deserve to be called "Vs," you know, compiler life was hard. Observation: one difference in multicore is a long history of multiprocessor stuff, and those at least get the benefit of throughput stuff, even if they don't get the speedup for the individual program. Right? And that's a huge benefit for lots and lots of environments. It doesn't make Word run any faster. *[See MPR 2/14/94-05, "VLIW: The Wave of the Future?," and MPR 12/5/94-06, "Architects Debate VLIW, Single-Chip MP."]*

Will Multicore Processors Sell PCs?

Patterson: Several of us at universities—including Stanford, Berkeley, and Illinois—many people are worried about the future of multicore. Now, in servers, or in cloud computing, I think multicore has a pretty safe niche. But the goal is to sell it in Mac Airs, that you and I are going to want to buy a Mac Air with 32 processors in it, in a few years. And we're going to tell everybody it's a good idea—don't get rid of your 16-processor Mac Air and buy the 32-processor Mac Air. If that doesn't happen, if the software can't effectively take advantage of that, sales of computers are going to slow. And, boy, given the history of multiprocessors, it's hard to imagine that any of us in this room, in whatever number of years, is going to say, "Yeah, get rid of your 16-processor laptop and buy the 32-processor one [because] it's going to run a lot of software faster."

Mashey: I thought you university guys were going to take care of it for us.

Patterson: Tell you what, we're going to try. Stanford, Berkeley, Illinois have big efforts, and we're going to give it our best shot. But what's strange for academic research projects is we have the sense that the industry is depending on these three academic projects to deliver, or else bad things are going to happen. And it's a very strange position to be in.

Slater: But there's something behind what you said. Will people buy new computers because they want the faster one? And I think that's actually ceased to be true for a number of years now. People buy new computers because, you know, it's a laptop and the keyboard wears out, or the screen is bad, or the power supply goes bad, or a new operating system comes out and it's easier to replace the computer. So, in some sense, I think progress in processors is nearly irrelevant to the rate at which computers are sold.

Patterson: So if things grind to a halt, it doesn't matter.

Slater: Well, it matters, but I don't think it would be crippling. I don't think it would have a huge effect.

Sachs: I think I would be happy if you could get Vista off my laptop.

[laughter]

Slater: Buy a Mac.

Sachs: That is next.

Intel's 8086 Was a Rush Project

Wharton: This isn't why I stood up here, but to clarify on David Patterson's comments: When the 432 underwent a one-year slip, that was going to be a 32-bit architecture with a 16-bit underlying implementation. When they learned there would be a one-year slip in the schedule, Intel had a vice-presidential meeting, and they came up with three design criteria for something that would fill in that gap. And the three criteria were that it have a 16-bit ALU and a 20-bit address space. Secondly—

Patterson: That came from the management?

Wharton: Yeah, this was like the top 12 people in the company came together with this list of design criteria.

Something had to be on distributor shelves—a thousand units, 52 weeks from today. And I think it actually slipped to 54 weeks, but the 8086 was actually on the shelves. And the third thing is, we have to make an argument that it's upward compatible from the 8080. It didn't have to be upward compatible; we just had to be able to make the argument that it was upward compatible from the 8080. Which they did, at the source-code level, using a translate-86 program that took the same basic register set and moved it into the 8086.

Sachs: Wasn't that an OO machine? Object-oriented machine?

Wharton: The 432 was an object-oriented machine. It was wondrous. Thirty-three bits in every instruction word—

Patterson ***[to Sachs]*****:** He's talking about the 8086.

Sachs: Oh, I thought he said 432?

Patterson: No, he was saying because the 432 was late, they did the x86.

Sachs: Oh, before it slipped. Ah.

Wharton: So, yes, the x86 evolved out of—

Patterson: Three weeks on the instruction set on a 52-week schedule to silicon isn't a lot.

Wharton: Yes, the edict was a thousand units on distributor shelves 52 weeks from today. Working backward, the only way to do that, was we had to have a spec on paper two weeks from now.

Ditzel: To draw the obvious conclusion, we're giving designers too much time to design! *[laughter]* If this is a market success within three weeks, then stop these year study projects.

Slater: It's also worth noting that the reason it's a market success is that IBM chose it, right? The success of the x86 had absolutely nothing to do with any of the design decisions that were made.

Patterson: No, there was one important design decision. Intel had both the—well, two things, compared to the [Motorola] 68000. They had the 8086 and the 8088. They had an 8-bit [I/O] version and a 16-bit [I/O] version of the same basic package. And that was a big deal for IBM. Also, Intel was there with the chip. And I think Motorola was later. So those two things, right?

Tredennick: And Intel had a full set of peripheral chips already available. *[Editor's note: Tredennick designed the 68000 microarchitecture and control store.]*

Patterson: Yeah, and the other thing that the people involved in the 432 don't get credit for is the IEEE floating-point standard, of which the 8087 was the first implementation. All that work got started with the 432. So they asked all kinds of big questions. What's the proper floating-point arithmetic for the future? They had lots of big questions. And so the idea for the IEEE floating-point standard was funded and originated in the 432, and the first implementation was this coprocessor floating-point chip, the 8087, and the standard was closely related to that.

Wharton: Oh, by the way, I hadn't heard, Dave, your comment that Gordon Moore thought there would be one more architecture and that we better get it right. It turns out he was right in that regard.

Patterson: Yep. And you look back, and he was right about a whole lot of things.

Failure Is an Option

Wharton: The reason I stood up here was to ask the question: Frank Lloyd Wright supposedly said that surgeons may bury their mistakes, but architects can only plant ivy. *[laughter]* I'm wondering what computer architects do when they realize they may have had the wrong insight in an earlier generation.

Brookwood: Switch companies? *[laughter]* I think the market takes care of that. Or if it's one large company, they can ship it anyway and sell lots of them.

Patterson: I think what's interesting for me, as I went through all the Hot Chips programs—I don't think I'm that old—I forgot there's all these numbers I hadn't heard of in so long. I just forgot—the 88000—oh, yeah, Motorola made a RISC processor. I completely forgot that. The [AMD] 29000—I'm not even sure what that is anymore, but I think it was a RISC processor, too. So all these other ones, I just forgot. If you don't say anything, they just disappear and people stop talking about them.

Mashey: So I get to put a little plug in. Come to the Computer History Museum! You get to see lots of failures. *[Editor's note: Mashey is on the museum's board of trustees.]* But the other one, actually, if you've never been there, is the Science Museum in London, out in South Kensington, which is a wonderful museum. Think of the British Empire at its height. It's got all the old steam engines. And you know what? They tried everything! And we've just repeated that.

Teaching Parallelism to Programmers

Audience member: Multicore has been a real changer for our industry. I'm wondering if it's not going to affect something else as well, that's a little older, namely the tree-chopping and pulping industry. So this is a question for you, Dave [Patterson]. What are you and Hennessy going to do about your textbook [*Computer Architecture: A Quantitative Approach*] with regard to multicore?

Patterson: What a great question! So the president of this university left it to me to do the next edition of the undergraduate textbook, because he claimed he had things to do. *[laughter]*

Mashey: What a wimp!

Patterson: So this edition is coming out in November. Every chapter has a section on parallelism. There's cache coherency. There's bus consistency. The problems of floating-point arithmetic in parallelism, you know. If you calculate with 32 processors, and you calculate with 16, you might not get the same answer. There's a new chapter on parallelism. So there's a brand new chapter and a new section in every chapter. Yeah, I think the challenge for us, as educators is—unquestionably, the future is parallel. A hundred percent

of architects think the future is going to be parallel computers. Yet at the same time, nobody knows what the programming model is. We know we have to train the people going to college *something*, but we don't know what it is. And you know, that's a tough challenge. At least on the hardware side, there's stuff we can talk about that's been done before, and there are some ideas that you should be aware of that help you understand where to start.

Mashey: Well, at least there's hope, right. Compared with the old days, when a multiprocessor was an expensive device, at least multicore things are cheap enough that every student can have one.

Patterson: I think I've got six reasons why we're going to succeed this time. The first one is there is no killer micro, right? No one is building a faster uniprocessor. In the past, you just did the lazy-boy programming. You just sit there and "I'm not going to learn any new programming. I'll just wait for Intel to make my processor faster." So kind of ironically, it's Intel's fault there are no multicores out there, because they can't build faster uniprocessors. It's all going to be in one chip. That will allow us to do things that maybe we've never done before. There's the open-source software industry, which is a really major force today, and that's a real meritocracy.

Audience member: Can they innovate?

Patterson: Can they? I think open-source software—

Audience member: So far, they've only been commoditizing.

Patterson: I think open-source software—these are volunteers. They like cool things. I think they want to give cool things to volunteers to work on. So I think there are some reasons for optimism. Before, only startup companies were working on it. Now the whole industry is working on it. Everybody's working on it now, so we've got a large fraction of people on it. So there's some reasons to hope we'll be successful this time.

Another audience member: A survivor worth mentioning is the [Zilog] Z80000. It had MMX instruction set. It is a survivor because it showed up, along with its floating-point bug in Pentium. If [former Pentium engineer] Don Alpert is here, he can reflect on that. Other smaller ones are FFT, ever since 200 years ago, and Huffman codes from 1950. Still, those have been resisting *[unintelligible]* for a long time.

Are There Opportunities for Startups?

Another audience member: I have two questions: the first one is very small and related to the second one. Given the complexity of today's chips, and all the failures of companies and startups that we have seen in the last 20 years, do you think our computer architecture market can accept any new successful startups? Well, I guess that's the first question. The second question is, How?

Ditzel: I think the big challenge isn't the market acceptance of a new startup. I think the problem is: Can a new startup actually get the funds to get a product to market? It is becoming so expensive now. I'm referring to the design of general-purpose processors to get something done. For example, the costs of taping out one of the first SPARC chips at Sun was maybe $20,000 or something? Today, it's on the order of $2 million to $4 million per tapeout, if you look at one of the most advanced [fabrication] technologies. If I look at one of the chips that was here in 1992 or something—the SuperSPARC chip—I think we did about 30 all-layer tapeouts. Many metal layers, trying to get things right.

Brookwood: Thirty? *Thirty?*

Ditzel: It was a very large number. It went for, I think, on the order of 24 months to get it right…This is a chip that finally made it to 32MHz, maybe a little bit more over time. But a lot of practices and things in Silicon Valley—where there are lots and lots of startups who can make chips and tape them out and do it with reasonable funding—I think they are going to find that the costs are just becoming prohibitive. So I think we're going to see a change in the type of startup companies and other things, just because it's so prohibitively expensive to figure out how to make a really significant advance. It's not easy. There's not a lot of low-hanging fruit.

The engineers have done a great job. Look at the Pentium Pro evolving into the [Intel] Nehalem processor, right? There's been a lot of years in between where people have very carefully made iterations there. And if you want to do something that gets dramatically better than that—wow! I wish you every success. But the history of 20 years of Hot Chips is that revolutionary ideas are rarely immediately successful.

And so I just think it's a very tough environment now. It's very tough for new startups to get funding. If you look at a recent example, like Montalvo [Systems], the rumors were that they spent on the order of $100 million to get the chip taped out. But it would have taken another $100 million to get to market. *[See MPR 5/27/08-02, "Editorial: A Tale of Two Companies."]*

From Transmeta's own experience, one of the biggest challenges, even if you get a product to market, is brand. And the question is, how many hundred million dollars do you have to spend to get things accepted? Because you could build a great product and people don't accept it. Transmeta had a bunch of notebooks that shipped, for example, in China. Tremendous sales in the first couple of weeks. And then they started to get returns three weeks later, because people hadn't heard what a Transmeta chip was.

And sometimes also, as a startup company, you don't have a lot of other products to back you up when one slips. It's very tough to get something so complicated out the door. One of Transmeta's biggest issues, most people don't know, was fab problems, which forced Transmeta to shut down shipments to customers for a year. That window that Nathan [Brookwood] picked on a little bit—that was fun, being stealthy—but really the time from announcement to when a competitor responds is very important there. And so in Transmeta's case, the tapeout was good, it was a good chip, it was a chip that worked, [but] it was just simply that the fab had trouble being able to manufacture it. The company never

really recovered from shutting all your customers down and then asking them to trust you, the next time, to use the product. So there are a lot of issues that affect companies, and it's a very tough business to get into for a new startup company. *[See MPR 12/26/07-01, "Transmeta's Second Life."]*

Brookwood: And if the money and the brand aren't enough problems, there are all the intellectual-property hurdles now. For established players, they can do cross-license agreements and kind of finesse the patents. But if you're the new guy on the block and you don't have any intellectual property to trade, then it's very hard to enter.

Tredennick: So you guys don't think programmable logic is the answer to this?

Patterson: Let me counterpoint this. You guys are the model where you go compete with Intel by selling chips. A whole lot of startups are in the model that the goal is to get acquired. The whole CAD industry is set up that way. These days, if you had a really great idea in multicore, some kind of software/hardware thing, and you could use FPGAs to emulate it, and your goal is to try to get acquired, and you have a genuinely good idea—

Ditzel: If you solve the multicore problem, you will be acquired. I guarantee it!

Patterson: Your goal has to be, we're not going to tape out chips. Here is our idea, we've got this incredible thing—

Tredennick: You need the proof of concept.

Patterson: Yeah, you need the proof of concept. It takes a lot less money than making the chip does. I think there's an opportunity for great ideas now. The industry needs great ideas. I think that somebody who claimed to have something, a significant advance, would be a very successful startup, you know, in an acquisition, which solves all the patent problems.

Sources of New Ideas: Academia vs. Industry

Audience question: So I have an interesting question, both looking back and looking forward. From my perspective, computer architecture is very interesting, and a lot of ideas that were novel came both from academia and industry. You know, the classic one, I think, would be the Tomasulo algorithm, which was developed in industry, not in academia. *[Editor's note: In 1967, Robert Tomasulo of IBM developed an algorithm for out-of-order instruction execution.]* On the other hand, you have things like simultaneous multithreading which was the University of Washington and the folks at DEC. So I'm curious, as you look back at all the great ideas, what you thought were good ideas, that proved to be great ideas, and going forward, what do you think the contribution of academia is and the contribution of industry? And let's try not to have a fight between all the five guys on the left and the guy on the right. *[laughter]*

Tredennick: Why not?

Patterson: I can handle myself pretty well.

Unknown: He's pretty tough. We're worried about the five guys.

Mashey: Let me make a comment here, because this is something I actually worry about. Dave [Patterson] and I were talking about this a little bit before. You know, there have been periods where in universities you could actually do clear leading-edge stuff that was pretty cool and pretty competitive. And then there have been other periods where it was harder to do that. And I would ask Dave, "Do things go back and forth? What kind of period are we in? Where are the classes of things that you feel you can do very cool stuff in the universities?" Because when I talk to university people, I say, "Do research to try stuff out. Don't do the stuff that worries about chip yields and stuff like that. That's a waste of time for you. Explore ideas. See what works. See what doesn't. But make sure, at least, it's not something that leaves all the software as an exercise."

Patterson: What's cool about computer architecture, at least if you're my age, is you could meet all the people who invented everything. I got to meet [Presper] Eckert and [John] Mauchly and Maurice Wilkes and all those guys. IBM was just this overwhelming thing in the early years of computers: caches, the Tomasulo algorithm. People who weren't around then don't quite realize how big IBM was. It was like Microsoft and Intel and Google all in one company, right? They just ran everything. And they had this huge money-maker in the mainframe, and so they were generous to universities, and they had huge research teams. They did a lot in the early years.

Having said that, I think there's a study that the National Academy of Sciences did about technological innovations, and they looked at where these things came from. You see this 50/50 mix between industry and academia. Now the thing to keep in mind is what they said is timed to a billion-dollar industry. They have 19 examples of billion-dollar industries, and it's about 10 years. So here are these ideas, some of them, usually, it was done both in industry and in academia, one before the other maybe, and 10 years later there's a billion-dollar industry. But I think that's our history. It's a little worrisome today, because a lot of the kind of researchy places in industry have gone away, you know, so I don't know what it's going to look like in the future.

Mashey: Once upon a time there was Bell Labs and Xerox PARC and stuff like that, and that's not the same anymore. Which to me says, particularly for those early stages of research, we've really got to count on the universities, more than we perhaps used to.

FPGAs Are Great Tools for Startups

Patterson: In answering Mashey's question—now I remember what it was—I'm pretty excited about FPGAs, because they're growing with Moore's law. And as prototyping technology, they're just about perfect. It's kind of like hardware and it's almost as easy to change as software. So in terms of building a prototype, it's a pretty exciting time. Now, you know, you're not going to get gigahertz with FPGAs ever, but can you faithfully build something that works and runs a lot of software, and you could accurately estimate timing and have a

very credible prototype. I think FPGAs are very exciting. Building chips has never been harder.

Brookwood: Speaking of FPGAs, when Sun announced that they were going to let people download the RTL of their SPARC processors, everybody said, "Oh, yeah, and how are you going to build it? Where's your foundry?" But in fact, they've now got FPGAs big enough so you can build and run a single core on an FPGA. And so these are really fantastic vehicles for students who are learning. It's almost like writing a compiler used to be—when I was doing it, you only got two runs a day through the punchcard machine—but basically, you don't have to wait for the fab to turn around a wafer. So I think FPGAs are very promising here. *[See MPR 5/3/04-01, "Microprocessor Sunset," and MPR 5/10/04-01, "Reconfigurable Illogic."]*

The Architecture Born of Paranoia

Tredennick: We'll just take the questions from people who are at the microphones and then we'll quit.

Audience member: One comment. The other wonderful thing about being a computer architect now, in your shoes, is that in 20 years, you're going to be collecting Social Security.

Sachs: I already am.

Patterson: Assuming it still works.

Sachs: I've been collecting it for 20 years.

Patterson: Yeah, none of us will *start* collecting Social Security for 20 years, except for maybe Ditzel. *[laughter]*

Ditzel: Alright, questions.

Another audience member: First of all, I'm a nitpicker, so I have to take that nonpossessive apostrophe thing a little bit further. In England, the spelling of Maurice is pronounced "Morris."

Tredennick: We're not in England. *[laughter]*

Audience member: And these are the same folks that take "Burlingame" and pronounce it "Blingham." But I want to take the two-week design of an architecture that changed the world and say that we had another architecture that was amazing. It took seven years to get it finished. It was a cooperation among a whole bunch of companies that were competitors. It was an architecture that had never been tried before. It was only a little bit of the problem. It was a floating-point architecture. And [IEEE] 754 is the only arithmetic you can find on a computer these days, and has been for a long time, since before it was a standard. It took seven years to get there. And it fixed a bunch of problems. It used to be that when you got a new computer, you had to approach it—if you were doing anything numerical with floating point—you had to approach it with an attitude of paranoia. And the guy who was the guru of that project, that standards project, wrote a program called "Paranoia," which would tell you what was wrong with the arithmetic on the machine that you ran it on. It also said, "If you get this error message, contact [Richard] Karpinski immediately." *[Editor's note: "Paranoia: A Floating-Point Benchmark," published in* Byte Magazine*, February 1985.]*

Patterson: That was a great success that was just before Hot Chips.

Advice for Young CPU Architects

Audience member Christos Kozyrakis: *[Editor's note: Kozyrakis is an assistant professor of electrical engineering and computer science at Stanford University.]* I have a quick comment and a question. The comment has to do with this business class I took eight years ago at Berkeley. I don't remember the professor's name, but the whole point was that failure of projects is a natural thing, and you just learn how to deal with it. More projects fail than succeed. And it's OK, as long as you save your career. And talking about this, it seems that even though you guys got involved with one or two failures over time, you've done great for yourselves. If nothing else, you've got 200 people listening to you at 10 p.m. and trying to correct your spelling or something. *[laughter]*

So my question has to do with those of us who are unfortunate enough to be young architects. Some of us will be here for another 30 years or something, unless we marry rich. So let's switch to a positive spin. Instead of saying "Merced was a bad idea," yes, OK, sure, let's talk about what's a positive thing that some of us can use over the next 30 years. One message seems to be, "Pay attention to software." That's good. Another message that came out was, "Be open, someone else will discover your mistakes." So why don't we just go around, and each one of you mentions one positive thing the youngsters should get from this discussion as they move forward for the next three decades or something.

Tredennick: I'll start it off. I like the FPGAs. I think that's a great way to prototype systems. And, in fact, it's a good idea for startups, because, for example, Altera has a thing called HardCopy. You can get a privately labeled product. You can do, I think, the NRE for HardCopy FPGAs for like $200,000. And you can get the development kit for almost nothing. *[See MPR 12/17/07-02, "Altera Aims For ASICs."]*

Patterson: I just heard of something called eASIC, which is kind of *[unintelligible]*, which is a little bit lower performance, but it's like $5,000 or $10,000 for the NRE.

Tredennick: That's a fabric, yeah. So anyway, that's mine.

Sachs: Mine is: Learn how to work in small teams, as compared with an individual contributor. Too often, you'll come up with a really good idea, you're not able to communicate with other people, or the worst case is you get pretty emotional about things and say things you're really sorry about, and your good idea goes down the drain. So learn how to deal with people.

Ditzel: I guess I'd say stick to it. If somebody tells you something is impossible, don't give up. A lot of good ideas take a while to develop. Over many projects, 10 years is not unusual—from you spending a couple of years to work on that idea, to taking a year or two to sell it, to get funding to build a team, to spend three or four years to build something, to help get it into the marketplace and out. So pick

your projects carefully, right? Because you're only going to get three or four or five 10-year projects in your career. But if you're onto the right thing, hang on to it, because you'll know it better than anybody. If you're really onto the right thing, you can push it, and quite often it will succeed. It's the ones that people hold on to the longest that actually make it through.

Slater: I'd say stay focused on what delivers real big jumps in actual value to real customers. I think lots of projects get pursued for ideas other than that, and they ultimately end up failing, because while they may gratify the architect and have some clever technical merit, they aren't really a meaningful increase in value delivered to customers.

Brookwood: I'll borrow from Shakespeare: "Unto thine own self be true." When you're looking at something and you're working on something, don't necessarily believe all the spin that the marketing guys may be applying to your project or that you're applying to the project. Look at it scrupulously and honestly to assess whether or not it's living up to what you thought it was going to be. And if it's not, and you don't see how you can make it get to that state, then you should be pulling the plug on it, rather than somebody else.

Mashey: I think there's a particular area over the next couple years. We've already seen a bunch of it here. That's really low power. I currently happen to be fond of and involved in wireless-sensor network things that have to go five to ten years on two AA batteries. I believe there's a lot of room because of energy problems the world is about to get into, where processors and electronics can be used to substitute for energy. The nice thing about low power is that it's an extra dimension beyond price, performance, and binary compatibility, where there's actually some interesting stuff that people can do. I think there's a lot of turf out there that I think we need some good ideas in, because we have this little thing called peak oil coming, and then there's peak gas, and then there's other ugly stuff.

Patterson: I guess I would say find something you're passionate about. No matter what you do, it's going to be hard work. Like the example of the IEEE floating-point standard, which sounds like the dullest topic you could imagine. Velvel [William] Kahan, who was the godfather of that effort, would keep audiences rapt, because he was so passionate about the importance of accurate floating-point calculation. I think that kind of passion is particularly useful even in academic research projects to get people excited to keep working on them. And particularly at companies, because when people are trying to decide to join a startup or not—besides the supposedly getting-rich part, supposedly—it's do you believe the vision that you have? So find something that you're really passionate about that you really believe in, because you need that kind of enthusiasm to be contagious within whatever group you're going to lead.

Audience member: And besides that, it will be fun.

Patterson: Yeah!

Are We Repeating the Same Mistakes?

Audience member: I would add one more thing. If you don't eat your own lunch, somebody else will. So a company that's afraid to start a new project that's going to interfere with their old one, they're just going to get their lunch eaten. But that's not the question I had. In the beginning was the mainframe. And the mainframe did some pretty amazing things and made some really stupid mistakes, and it begat the minicomputer, which did some amazing things and made the same stupid mistakes, and eventually that begat the microprocessor, which did some really cool things and did the same stupid mistakes over again. So we're sort of the stepchild of mainframes, in many ways. My question is, if we've been following in their shoes, are we now grown up and we're making our own mistakes? Mistakes that the mainframes never had and never will? Do we have more mistakes to make in the same shoes that mainframes made? Or are we really on our own now, all grown up?

Mashey: Sometimes I said this thing where every mistake in computer architecture gets made at least three times with that exact set, except I had to change it to say four times because of SoCs. Except—

Patterson: Because of what? SoCs are?

Mashey: Well, because it was a new generation of stuff. What I was going to say is, I don't think it's actually true, in the sense that if you look at it people-wise, there was a pretty big break between mainframe designers and minicomputer designers and microprocessor designers. There were people that overlapped, but there were a lot of design communities that were sort of independent. Whereas an awful lot of people doing SoCs these days have done microprocessors before, or put microprocessor-based systems together. I used to see the same battles on SoC internal interconnects that completely replayed all the same battles we had with multiple microprocessors. So I think we're better off, OK? Of course, we still live with all the mistakes we made.

Sachs: Well, I worked on, in 1975, an IBM plug-compatible [System/370 Model] 158, so I have done those.

Mashey: But you're unique, Howard.

Sachs: I know, I know. And you know, it feels the same. It really is no different. Just, you know, different people, different stuff, different problems, but it really feels the same.

Patterson: I think, in this 21st century, with this multicore challenge facing us, we've got the chance to fix things. A lot of what today is mistakes that have happened, we're stuck with, you know. Nobody is willing to change all the software in the world to fix this terrible thing. Like security, right? The people that invented email, what was our spam technique? We would yell at anybody that sent spam. That was our technological solution to spam on email. *[Laughter]*

So there's a bunch of mistakes we clearly made in the past that we can't figure out how to change. Because of multicore, this challenge in the whole industry has to change. Here's our chance to fix everything. You've got to solve the multicore problem, which is a huge challenge. But you can fix all kinds

of things that we're embarrassed about computers today, and make a much more attractive foundation for the 21st century than we have so far. We would have thought, well, it's too late—like your story about the archaeologist—we can never change that stuff. It's possible we could change it. There is such a compelling need that we have the opportunity.

Ditzel: I question the magnitude of the change a little bit. When I think back to mainframes, there'd be a 16-processor mainframe system. The only difference between that and today's multicore is that it's on one die. But the years we've had since the first shared-memory multiprocessor, which was a Burroughs machine about 1962 or so, is there's been really embarrassingly poor progress in the programming model for that. That really is not changed by multicore. So while multicore has many benefits still, how do we use hundreds of cores effectively is the big issue. That's the challenge. There are other good ways of using multicores today that are the same ways we've used them for the last many years. I think the challenge is, are we going to find new ways to use multicore, which are really a lot better. And I don't know if we see the exact answer to using hundreds of cores yet, and I think that will be the big change. I guess I agree with Dave Patterson. That's what we've got to look for.

Don't Forget the Applications

Patterson: I'd say one thing that strikes me, as I look back over all these years—and Illinois and Berkeley and Stanford all do the same thing—is that somehow this field got created independent of applications. We could do computer design without knowing how in the world they were going to be used. And that was OK, that was fine. How do I figure out if it's good? You run Dhrystone, and then you're all done and you don't have to talk about it. What a weird thing that we all thought we didn't have to know any applications. So all three of these new research projects are trying to work closely with applications developers.

I think it was Stan Mazor on the [Intel] 8080, I think he said, "Well, we tried to solve some problem, and hopefully by solving some real problem, it'll solve others." And that must have been one of our weaknesses in the multiple parallel-processing effort. There would be all these architects with this preconceived idea of the right thing that was going to solve the problem, and they never got around to apps. So maybe being more tightly tied to apps will give us a better chance this time around. To me it's startling that we don't really know anything about the apps that we're building the computers for.

Is the x86 Forever?

Tredennick: Last question.

Another audience member: We have to end on failure, I guess. It occurs to me, while watching all the slides, that Wow!, we had this wonderful failure from Intel called the 432, and that led to the great success of the 8086, also from Intel. And then we had another great failure from Intel, the Itanium, which led to the x86-64 opportunity from AMD. So first of all, I guess we've got to thank our colleagues at Intel for these wonderful failures that have promoted great, successful architects. And so, how do we get them to do it again? *[Laughter]* Is it possible, in seriousness—do you think there will be another general-purpose architecture? Or whatever it is 100 years from now, it will still be called x86?

Patterson: There is a group that looks only at the computer in front of them. That's the only computer in the world, right? But there's a whole lot of computers in the world, a lot of computers that aren't on your laptop. Those are pretty unbound by binary compatibility. I think there's plenty of opportunity for innovative designs. I think the question is going to be, in terms of binary compatibility, whether we can get the power efficiency that we desperately need to fit everything on the multicore, and still be binary compatible. Right? Or will that have to go by the wayside? I don't know the answer to it, but if that gets to be in the way, then maybe things will change. But I think there will be all kinds of system-on-a-chip things, like Mashey was talking about, where people have special-purpose cores, ARM processors, all kinds of things are out there. So that community, I think it's the more exciting community. I don't know how many people have iPhones. I'm very excited about my iPhone. It doesn't have an x86 in it, and I don't think it ever will.

Brookwood: I also think we're finally getting to the point where some of the emulation techniques that people have talked about for years are finally becoming workable. I mean, look at your Macintosh today, where they did the transition from PowerPC to x86. It went incredibly smoothly, far more smoothly than I had anticipated. Far more smoothly than when they went from 68K to PowerPC. And a lot of that had to with the Rosetta technology, which they get from Transitive, which is also being used in some IBM Power6 [systems] to run x86 and so forth. As that technology becomes not only more mature, but [also] better accepted

For More Information

For more information about the Hot Chips Symposium, visit:

www.hotchips.org/hc20/main_page.htm

To purchase DVDs recorded at Hot Chips 20, visit:

www.hotchips.org/hc20/store.htm

For additional information on the development of the Intel x86 architecture, the 8087 floating-point coprocessor, and the IEEE 754 floating-point specification, see:

"The Intel 8086 Microprocessor: A 16-bit Evolution for the 8080," by Stephen P. Morse, William B. Pohlman, and Bruce W. Ravenel, IEEE Computer Magazine, pages 18–27, June 1978, Volume 11, No. 6.

"The Intel 8087 Numeric Data Processor," by John F. Palmer, Proceedings of the National Computer Conference, 1980, pages 887–893.

by ISVs, then I think you do have the ability to innovate in the microarchitectural space, and even in the architectural space, and be able to introduce a product without having to worry too much about the chicken-or-egg issue with software. *[See MPR 1/30/06-08, "This Technology Is Virtually Here Now," and MPR 8/8/05-01, "Transitive's Tech Frees ISA Dependence."]*

Mashey: I'm actually an adviser at Transitive. They do some pretty interesting stuff. In general, the resurgence of virtual-machine technologies allows for some interesting things, I think. For innovation in the general-purpose market, I think that's one of the keys.

Patterson: Take another thing that was on one of our slides about the lack of innovation. I snuck out today and went to NVision—Nvidia's answer to MacWorld [Expo], I guess. That is a very exciting community. Their story is, "Moore's law is for slowpokes. We're going faster than that." And they're tied to apps. There's not this big disconnect between apps and them. And they have visual computing. It's hard to come away depressed about the future of computer architecture when you look what's going on with GPUs. Play forward 20 years from now—I don't really know which computer will be dominating the future. People in the future might use GPUs like we use CPUs today, right? Obviously, it's visual and media computing that's at the center of everything, because that's what exciting, and there's also these things that run operating systems. It didn't come up in this panel, but the GPU thing is a pretty interesting force that's certainly a wild card and may play an increasingly important role. *[See MPR 1/28/08-01, "Parallel Processing With CUDA," and MPR 9/29/08-01, "Intel's Larrabee Redefines GPUs."]*

Tredennick: Thanks, guys! *[Applause]*