Oral History of Robert Garner, part 1 of 2

Interviewed by:
Roy Ogus

Recorded December 6, 2018
Mountain View, CA

Edited by Roy Ogus, March 2021
Edited by Robert Garner, January 2024

CHM Reference number: X8869.2019

Ogus: Hello. I'm Roy Ogus, and today is December 6th, 2018. I'm here at the Computer History Museum in Mountain View, California, with Robert Garner, who has graciously agreed to be interviewed for an oral history.

I've known Robert for many years. We first met around 1977, when Robert participated in a Digital Design class that I was teaching at Stanford, University, while I was involved with my post-doc research. Following that period, Robert and I have worked together on several product development activities at both Xerox and at Sun Microsystems.

We worked together as designers in the Xerox team in Palo Alto that developed the Dandelion workstation, which, of course, later became the platform for the seminal Xerox Star workstation product. Then a few years later, and actually by coincidence, we worked together again as part of the joint Xerox-Sun Microsystems SunDragon team that developed the Sun SPARCstation products. We'll hear a lot more about these activities in detail during this interview.

So Robert, we've had a long history together, and I'm delighted to be talking to you today about your story. Thanks so much for joining us. We really appreciate it.

Garner: Well, Roy, before you start your first question, I want to say I did work for you for a short-time, as you were my manager, and you were actually the best manager I've had --

Ogus: That's right. You were actually my manager as well, yes.

Garner: Well, could've flipped, but you were actually the best manager I've had, so...

Ogus: Okay, thank you.

Garner: And so that was--

Ogus: That's right. The first part of the Xerox thing.

Garner: Yes, and thank you for agreeing to do my oral history.

Ogus: Okay. My pleasure. All right. So why don't we start at the beginning? Can you tell me where you were born and when, and where you grew up? At the same time, why don't you also tell us what your parents did and how they influenced your various career decisions?

Garner: Yes. Well, those are all the easy questions.

Ogus: Yes, we start with--

Garner: So yes. So I was – I'm a native Californian, actually. I was born on the vernal equinox, 1954. That's when the day is half light and half dark. My parents lived in the Manhattan Beach area in L.A. When I was about one year old, I ran away to Arizona. What really happened is that the companies on the west coast were being encouraged to

expand inland because of the threat during World War II of invasion from Japan. So the company that my father worked for decided to move inland to Arizona.

My father, he grew up with a lot of responsibility early on. I think he taught that to me. When he was a young kid in high school he used to drive all the kids in the high school bus to the high school in Hollister. He would drive down Highway 25, drop them all off, stay at someone's house by candlelight, do homework. Next morning pick them all up one by one as a freshman in high school before he could even have a driver's license and drive all the kids into school.

He wanted to become an aeronautical engineer. His goal in school was to find a crashed airplane, because you couldn't afford to buy one, and to restore it, and to get it to fly again. He ended up joining AiResearch -- next to the Sky Harbor Airport in Phoenix, and worked on gas turbines his career. He kept all his college textbooks and so I ended up devouring his physics textbook. I just loved all the stuff I saw there.

Where I grew up, first in downtown Phoenix, and it was orange groves and being converted into housing divisions, we used to build, make things: Build tree forts. I would make all kinds of contraptions. I was known for building things. As a young kid, I would find strange things, like algebra papers thrown out of cars, and I was totally fascinated by those. Camera lenses in the field, all kinds of crazy things. I even later learned that Spielberg lived in my neighborhood, so it's even possible that I played with him as a young boy, and we certainly went to the same restaurant that he made his first movies in.

Starting somewhere in grade school, we formed-- two, of my best friends and I formed a science club and we met every weekend. We did crazy things: We built 3D chess sets. We built tube-based amplifiers. We did mathematics and calculus. We found adults who had access to computers. So we started writing small programs in seventh grade, sixth and seventh grade. I wrote a Tic-Tac-Toe program. You would kind of hand the program over to some adult and they would go find a computer somewhere to run it for you.

So the two close friends of mine were Jerry Jenkins and Chris Beall. Chris was a brilliant person who, I'll never forget, he wanted to check out a college physics book but they wouldn't let him because they thought he was too young and he was trying to design a particle accelerator. My friend Jerry Jenkins was a brilliant mathematician.

We had access to a terminal system at the high school first that did timesharing to a Scientific Data System, SDS Sigma, Sigma 9 computer or Sigma 7, located at a service bureau in downtown Phoenix called Transdata. That's where I learned that if you actually go to the computer center at night you can get full access to the computer. And it runs faster at night. They let us run the whole Sigma machine ourselves, like at 1:00 A.M. or 2:00 A.M. Quite a fun thing to do. Also, we found that if you searched the wastepaper baskets you found user IDs and passwords, <laughs> so we could get extra free time on the computer and get privileged accounts, because then people didn't realize they shouldn't be typing out user passwords.

Back to my father, I guess, a little bit. He loved the outdoors and we did some hunting and he was very observational. He could see animals before I could, and so he really taught me the skill of being the careful observer and watching things. I was really into botany and natural history. I would devour magazines like Science News but I couldn't figure out how people got their ideas. So I figured, well, maybe you had to study things, so I would take pictures of

landscapes to see how they might change over time. My dad would take me deer hunting. I later realized that wasn't a great thing to be doing. He had a playful, wry sense of humor. We'd go on car camping trips. We visited Lowell Observatory. Arizona has all these observatories -- islands in the sky observatories -- and I'll never forget being impressed by seeing Percival Lowell's hand drawings of the canals on Mars. Lowell so projected into what he was seeing through his refractor telescope that he thought he saw canals. But seeing those original documents really moved me into exploring science and the history of science.

Also, in seventh grade, I got a little carried away. My social studies project ended up being 700 pages of a history of the Conquistadors and the Spanish of the Southwest. I was the school class president in eighth grade and gave the graduation class speech. Oh, I almost forgot. In third and fourth grade with my friend Chris, we were school crossing guards, which is a very responsible position. In seventh and eighth grade I had the responsibility of lowering and raising the American flag on campus. So I had these very kind of responsible things to do.

But that was the time of the Vietnam War and a lot of protests against what was going on. I tended to kind of hang out with the more, with the people saying, "Let's end this war and stop this craziness." So I had kind of an anti-large-establishment mindset, more of the hippie mindset, you could say. So I wasn't really interested in business, but I was more interested in science and technology.

Oh, last thing on my father is that we, well, at that time, I guess, I also joined the Boy Scouts. We went on a lot of backpacking trips in the desert and the mountains in Arizona. I eventually became an Eagle Scout. On the way you do things like Order of the Arrow where they do unusual things like you don't speak for entire day, learning first hand what's it like to go through an entire day without talking. One of the things that came from joining the Boy Scouts was that I attended the International World Jamboree in 1972, which was held in Japan, at the base of Mount Fuji. You could just walk a hundred yards and be with another group of people from somewhere else around the world, which also strengthened my interests.

I had, I guess, something else I hadn't mentioned, is that at the time, Heathkits were very popular -- the electronic kits where you would build things. So I hand-assembled my family's color TV from a Heathkit. Through following the instructions, I hand-assembled a shortwave radio, and you could listen to broadcasts from around the world, late into the night. You could listen to programs from Russia, Cuba, South Africa, and elsewhere. My father and I actually continued our backpacking tradition as I became an adult, and we used to go backpacking into the canyon country of Utah and Arizona and New Mexico. So I was very close to my father.

My mother was raised on a small farm in West Virginia, and she was very caring and inquisitive and believed in science. She really believed in science, and she used to, when she would go shopping, to show you how geeky I was, she would drop me off at the library, which I just loved, and I would spend all day at the library reading books on relativity by Einstein, and math and all these physics things and I was just in heaven.

I just loved the libraries in the Phoenix area. I, as a student in high school, I used to go to the Arizona State Library and hang out in the stacks late at night, and that's where I got acquainted with books on computer design. I guess I did the usual science fair projects. I explored the electrical activity of fish using an oscilloscope. You saw kids doing little, simple computers as science fair projects, but I found them to be too simplistic: If you have access to a real computer, why bother building something which isn't very powerful?

Later we moved to the north Scottsdale desert. This would've been about when I was in sixth grade, and there we lived in the desert but within a golf course, and so that's where I got into running. It was a very meditative thing. I would run the fairways at night through the sprinklers when it's over a hundred degrees outside and jump into the pool. I was a lifeguard at that pool, participated in swim meets, and it was a great -- a great balance between exploring the creatures of the desert -- I had snakes. I used to collect snakes. We had snakes in the house. Terrified my parents when they escaped from their cages and my father would see a snake slithering down the hallway in the middle of the night. <laughs> So I had a very strong interest in the outdoors and natural history.

Ogus: Maybe I could just ask a question here. It seems like many of your activities were technically oriented. I mean, your father was in the technical field.

Garner: Yes.

Ogus: But you also did a number of other activities that were not technical related. But somehow you kind of moved towards a technical career, and, in fact, you told me that your father actually made/moved you, from doing a natural history type of university education to an engineering one.

Garner: Yes, yes.

Ogus: So do you really understand how your interest in getting into the computer field evolved? Was it just innate or was it [as a result of] your influences while you were growing up?

Garner: Well, I think it was the -- I was interested in electronics, and I think it was the access to the computers, early access to computers via adults. I think at a very early age I realized that computers could be used to control almost anything mechanical in the world and I, so I -- it was a hobby, designing electronic things. I consumed every issue of Popular Electronics. Never forgot the Altair cover, but my allowance didn't allow me to purchase the kit.

Also in high school, at Saguaro High, I ran for freshman class president. Mike Boich, my, the friend who had built, also involved in our science club, beat me, but it was a very -- it was a Sputnik-era high school in the sense it was well funded. The math teachers were incredibly excellent, the physics teachers were incredibly excellent. I was a TA for the physics teacher one year. We actually, several of us at Saguaro High placed 4th in the MAA National High School Mathematic Examination. The high school just ahead of us was Palo Alto High, which we hadn't heard of before. <laughs>

So it was-- and I took Russian as well. Russia was dominating the news, Sputnik and all the spacecraft. It was thought to be important to learn Russian, so I took Russian in high school, and I visited Russia in a foreign language exchange program during the height of the Cold War. In fact, I was in the US consulate, in then Leningrad, the day of the American Bicentennial. It was quite an experience. One thing I remember about Russia is that they really promoted science and technology, so as a geek it was kind of fun. I remember going to a science exhibit featuring rockets, Soyuz and the cosmonauts. There was also a computer that looked a lot like an IBM-360. I later learned that Russia cloned 360s almost exactly. One thing that really helped with computing was that I took a typing class as in grade school, and so therefore a teletype terminal was not intimidating. Many people in the computer business never learned to type, but by taking a typing class first, that made that less intimidating. So that so…

Ogus: Was it really true that you were thinking of doing a natural history track?

Garner: Yes.

Ogus: We will get into college next, but--

Garner: Well, even before I get to college, one of the other projects I did is I built a -- the Scientific American magazine had an Amateur Scientist section that I consumed every month. One month I decided to build one of the projects, which was a mass spectrometer. I linked up with a physics professor at Arizona State who helped me build it. He called it the best plumbing system in the world, because you have to have a vacuum in them and was a very simple mass spectrometer, but that was lot of fun.

In high school I used to borrow the physics teacher's Geiger counter and I would inject radionuclides in plants. You could actually get samples of radionuclides over the mail by subscribing to free U.S. government publications, since I couldn't afford them. I would inject them into plants to see how they might travel.

Oh, the other thing that happened in high school that really led me to computers was that GE had a recycling center in South Phoenix near the airport [now Apache Electronics], where they would send all their defunct computers, maybe just one or two years old, and tear them apart. These giant mainframe computers, GE 225s or whatever. They would chop off the ends of the printed circuit cards for the gold and sell the disk platters as coffee tabletops. And then they would take all the discrete components and offer them to hobbyists. So we would go visit the GE recycling -- so that probably was my earliest interest in old computers as well, because, "How could they be chopping up old giants into pieces and selling them as coffee tabletops, the disk drives, four-feet diameter disk platters?" I never had enough money to buy one. There were also other old collectible places in town that had that kind of stuff.

I was also very much interested in math, and I remember in high school I was really fascinated by Gödel's Incompleteness Theorem and math philosophy. But I really did, like you say, I really wasn't sure whether to go into natural history or engineering, and my father said: "Well, you'll get paid a lot more doing engineering," and that led me to pursue that. So I went to Arizona State University (ASU). My friend Jerry Jenkins had already started there. At Arizona State you had to -- I wanted to get a B.S. in Electrical Engineering, but they had a B.S. in Engineering degree, so you had, EE would be a specialty, so they made you take a well-rounded engineering curriculum, which --

Ogus: This was 1972.

Garner: 1972. Yes, when I started at ASU. I didn't like the fact I had to take mechanical engineering classes necessarily, but it really helped me later in systems design understanding things like fluid and heat transfer. I had two jobs as an undergrad. I found that taking undergraduate classes, especially from perhaps not all inspiring teachers, was a grind. What really helped me get through college as an undergraduate was those two jobs, one for a math professor, Dr. Gregory Nielson, and the other for an experimental psychologist, Dr. Barry Leshowitz.

Ogus: So these were during the term time, they weren't like a summer job?

Garner: They were during the normal academic year. Dr. Nielson had an office in the Math Department with a Tektronix 4010 storage terminal. As part of his research into splines, he spearheaded one of the early 3D surface drawing programs based on parameterized equations as a function of two parameters, *u* and *v*. We ran the software on a PDP-10, located at a timesharing service over a 300-baud modem line. In would come the coordinates for the 3D surface which would show up on the storage tube. I hand-tuned some of the algorithms in assembly language [and wrote a simple hidden line removal algorithm. [Appendix Image 1-A shows two SURV surfaces on the Tektronix 4010 display]. I learned PDP-10 assembly language at that time. Most of our software was written in FORTRAN.

I also did a special project for Greg. He realized that we could take the 300-baud modem data stream coming in over the phone line, record it on a reel-to-reel tape recorder at the slow speed and then play it back at a higher speed and get a 4x faster rendering speed. I built a little box that converted the modem levels to levels appropriate for a magnetic tape recorder. It was a lot of fun. I also had the honor of access to his office lab at all hours of the day, so that's where I hung out, where I did my homework and reading. I could avoid the campus dorm life and retreat to the math office on Friday nights.

Dr. Barry Leshowitz in ASU's Department of Psychology had a real-time system, a DEC PDP-15, an 18-bit computer, and eventually had two of them. They were used to send auditory signals into a soundproof room. You'd walk into one of those rooms and you can't hear anything due to all the sound-absorbing cones. It was a thing to do those days in experimental psychology -- to determine how the human ears and brain detected distance or location for objects. You would send sounds to the students's ears in the room and they would be the guinea pigs.

One experience that really impacted me a lot working around the PDP-15 and other computers was that they were human-sized. They're in racks and they're mute, but they're smart. You can tell by the blinking lights -- so as a kid you're walking around these big computers with blinking lights. It's a pretty impressive experience and I think that really fashioned my feelings about computers at that time.

Ogus: And so your degree at--

Garner: Well, before-- let me finish with the PDP-15. So the technician, Woody [Sisson], he kept-- I noticed several things about that. He was kind of a crazy guy, but he kept a Bible on top of the system. When it booted up, it played a pre-recorded "Good morning, Woody!" greeting. Just imagine that this system in mid-1970s, cost around $150,000, or in today's dollars, 700 to $800,000. There were also many costing millions of dollars. It took that kind of system just to play a five-second audio. <laughs> So it was kind of hard to extrapolate into the future from there.

The ASU administration also had a UNIVAC 1110, a large business computer, although an astronomer in the Math Department would complain to me that it never got the same answer on floating point calculations. So I was really learning how computers could be fallible and not necessarily yield reliable results. A friend of mine on the UNIVAC 1110 was able to become superuser and take over command of the admin console right during the middle of a demo that a staffer was giving to visiting dignitaries. The operator freaked out because my friend had rendered a big eye that blinked on the console screen, took over the screen and blinked. <laughs>

Ogus: So talking about UNIVAC, you told me you did a summer job at Sperry on Univac?

Garner: Oh, yes. One of the jobs that I had was in the summer, just a summer job. This was Sperry Flight Systems. In the room next door they were encoding and testing the words, "Pull up, pull up," in different voices for the pilot to pull up. In our area, a person had written a compiler for an HP 2100. This was the era when programmers debated whether compilers could generate good code and, in this case, the compiled code didn't fit in memory. So I had a summer job to be a human compiler. I would take the assembler output and rewrite it into a smaller form. I could see how dumb the compilers had been, so that was a great learning experience on compiler technology. <laughs>

Ogus: Were there any other summer jobs you had that were kind of similar type experience?

Garner: No. But I do remember a physics student in there, was coinhabiting in the Math Department, who wanted to build a TV terminal to display characters. He couldn't get it to work and I was astonished that somebody with a physics background didn't understand the practicalities of building circuits. So I helped him get his master's degree. So yes, that was my-- so I basically got through-- one thing that was really nice though is the Engineering Department allowed me to take Math Department courses instead of engineering math courses, which was really nice.

So I'd applied to universities. I applied to several. I was so delighted that Stanford admitted me. I didn't realize that by going to Stanford I'd be kind of homecoming to where my father's family had settled in California. [And that my two great aunts Ettie and Lettie Garner had attended Stanford in 1898 - 1902, to teach history and physics in at Hollister High]. I did one other project there at ASU. For the PDP-15, I designed a D-to-A converter, teaching me about that transition out to the analog world.

Ogus: So did you consider other universities as well?

Garner: Yes. Yes, I did. I had, I was accepted to some others. Case Western and a few others I can't remember now. For Stanford, I got admitted into the CS Information Theory curriculum.

Ogus: So you chose Stanford, and that was --

Garner: Yes. One thing I liked about Stanford is they had a short master's program, just one year.

Ogus: One year. Okay.

Garner: Yes, and that really appealed to me, because I really wanted to get out into the real world. Andy Bechtolsheim, was a student there and he also really wanted to get out in the real world. Of course, Bill Joy over at Berkeley did also, neither staying on for a doctorate.

So I was admitted into Information Theory, which I just loved. But I soon realized that it was an intense subject for just a year, so I asked permission to switch to Computer Science, which they granted. Stanford's EE curriculum had a lot of labs, a lot of work and time. I also took Russian again, which was probably not a good idea, because it didn't help my GPA, as I didn't get A's in that class. But it did imbue me with the beauty of Russian poetry. Political Russian sounds very hack-eyed, but since in the Russian language you can put the words in any order, it makes for lovely poetry.

Some professors that I recall were Bernard Widrow, who really understood analog computing and had his own neural net models. Ed McCluskey, who you remember, Roy, I'm sure, who --

Ogus: He was my advisor.

Garner: Your advisor, okay, who taught Karnaugh maps. This is where your brain actually did what's all done by software today, minimizing logic circuits. I had Jim Clark for discrete math, who later started SGI. I had Niklaus Wirth's compiler class, but felt that it was a step backwards because we used an IBM system with punch cards. I actually had to produce my compiler on punch cards. I thought, "for a well-known university this seems really backwards."

Ogus: You probably came into contact with Don Knuth as well?

Garner: Yes, I had Don Knuth for a class. I signed up for his Algorithms class and attended the first few sessions. Stanford allows you to drop a class after a week. I realized that his class, along with the labs I was taking, was going to be so much work that I couldn't pull it off, so I dropped his class. A few years ago here at the museum over dinner with Don and I said, "Don, I signed up for your Algorithms class and dropped it. I felt like it was two classes in one." He cheerfully responded, "That's right. I did run it as two classes in one." I felt off the hook a little bit.

Then I had electronics labs. Roy, you were my instructor in the electronics lab. I think I built a 3D clock based on a cube, as there are 12 edges on a cube and 12 hours in a day.

Ogus: Right. And you said you still have that device?

Garner: I still have it, yes. I have it somewhere.

Ogus: So you inherited the gene from your father to keep everything, right?

Garner: To keep everything. Yes, I guess so. Yes.

Ogus: Yes.

Garner: Yes, that's a good point. My apartment mates were in the OR program, Operations Research at Stanford, Rick Biedenweg, Lynne Weber and Linda Duyanovich. Lynne was brilliant person, National Bridge Championship. Rick was brilliant. He helped me with my math and physics. I met my wife-to-be, girlfriend, Robin, at a Keystone concert in Palo Alto, where the Jerry Garcia used to play on Sunday evenings. One evening, she was there to see the Any Old Time String Band and I was there to see Peter Rowan.

I was itching to get out of Stanford. I attended an on-campus interview hosted by Bob Metcalfe. Late at night he called me and said, "Well, why don't you get out of that easy school stuff!" I'm thinking, "Easy school stuff?" <laughs> "--and get to work?" I suspect he was impressed by my work I'd done for Greg Nielson on the analog tape recording device. [Later I learned that Greg had consulted at Xerox PARC.] So what I like to say is that all that happened was

that my biking route shifted, starting from Latham Square Apartments, two miles eastward from Stanford to the Xerox PARC campus on Coyote Hill Road.

Ogus: So what was it that attracted you about the SDD, System Development Department?

Garner: In 1977, that's when I finished my MSEE degree and Bob Metcalfe hired me to work on Ethernet in SDD. His charter was to commercialize the Ethernet as part of the huge Alto commercialization project, called Janus. The hardware plan was with Chuck Thacker… well, the Alto -- I won't go in much of the history, that's been covered so much, but -- Chuck was the chief designer of the Alto. This second system after the Alto that he was designing was called the Dolphin or D0, and it needed an Ethernet controller. So my first role in Xerox SDD, System Development Division, was to design the Ethernet controller for the Dolphin. That was an eye-opening experience!

The Alto was a microcoded machine. You needed to get your hands dirty at that low level so you could actually implement things like BITBLT microcode to move arbitrary rectangles on the screen quickly. Also, the Alto was architected such that device controllers could be mostly implemented in microcode. All the hardware interface needed was a small amount of hardware just to get a few words in and out, but everything else was done with microcode. The Dolphin expanded upon that paradigm, but it ended up falling into the "second system syndrome" trap. We'll do everything the Alto wanted to do, but better. So Chuck wanted to spruce the Alto up and polish it a little too much, and...

Ogus: So all this work was being done while you were reporting to Bob Metcalfe and his group, or were you actually in a different group at the beginning?

Garner: I don't recall who I reported to right from the very beginning. Could've been Bob. Yes, it was Bob Metcalfe.

Ogus: Okay.

Garner: Yes. I'll have to get the org charts to see [Appendix Illustration 1-B shows Xerox SDD organizational chart].

Ogus: Yes. You told me you were so amazed when you came to Xerox that everyone had an Alto on their desk.

Garner: Yes, yes, yes.

Ogus: Of course, it was very different back then.

Garner: Yes. Xerox gave each engineer an Alto. If you think about it, an Alto back then cost around $16,000 to build. In today's dollars that would be something like $50,000. So to give each engineer, including a kid out of college like me, a $50,000 tool was pretty remarkable. That was almost half the price of a home in Sunnyvale! I wasn't so impressed by the bitmap screen, but I was flabergasted that the bitmap screen consumed half of all memory. Alan Kay had wanted premium graphics, a 600-by-800 bitmap display. Half of all memory was consumed just to make a document or drawings look good. What was that all about?

But I was completely smitten with the Ethernet. Bob had been inspired by the ALOHAnet in Hawaii, and his idea was just do ALOHAnet on a coaxial cable [augmented with carrier sensing, collision detection and backoff]. The first Ethernet had been done at 3 megabits per second by Dave Boggs and Bob Metcalfe. I'll never forget doing my first FTP between two Altos and, like, the files transmitted instantly. If you're used to 300-baud modems (300 characters per second), 3 million characters per second, through a simple command line interface was shocking. It just really blew me away.

Of course, when I first came on board I couldn't believe they actually paid you for doing your hobby! <laughs> That's what struck me the most. "Oh, Yes, this is pretty cool. This is all fun and they're paying me." It was like $22,000 per year salary, equivalent to four times that today.

The Xerox System Development Division or SDD, was co-located with PARC at the time. So we kind of felt we were part of the PARC pantheon even though I didn't technically work for PARC at the time. We shared office space, we got to know all the people there. It was like you were in some altered state of reality. Everyone there was an expert. Roy, you might remember that.

There were the set of wizards, who were very generous, kind, unassuming people. The people who fell into that camp were folks like Bob McCreight, Ed Taft, Bob Sproull, Severo Ornstein, who also founded the Computer Professionals for Social Responsibility organization; Will Crowther, also a climber and caver, author of Adventure, the caving game that everybody played; Ken Pier, John Warnock, Hal Murray, Steve Purcell, Don Charnley, Larry Tesler, Carver Mead, Chuck Geshke. They were all very kind and unpretenous folk and just great to be around.

Alan Kay was this wild, inspirational, contrarian and practical man. His famous aphorism: "The best way to predict the future is to invent it!" -- is now embodied on the floor at the new Stanford Huang Engineering Center. He's related that he was angry with the Xerox planners that came to his office asking him: "Well, what's the future going to be like?" He just became upset: "Well, don't-- stop asking me that. Just give us the funds and we'll invent it!"

Ogus: That's one of his famous statements, right?

Garner: Yes. Well, it came out of being frustrated. I recall him being really frustrated with these planners from Xerox East Coast, and they had no idea what was going on at PARC really. So then there were the kind of the high priests, I call them, these people with very massive and significant egos. Very brazen, unreserved and frank and scrupulous designers. There was Butler Lampson, who was loud, had a booming laugh and an encyclopedic knowledge. You could walk into the PARC cafeteria and you knew he was there because there was a crowd of disciples around him. I don't know if you remember that. He--

Ogus: I remember. I also remember that we used to measure the speed of talking by how many "microLampsons" there were!

Garner: Oh, yes. <laughs> microLampsons. <laughter>

Garner: Yes, he was way too smart. He was the lord of Dealer, which was the weekly forum where you sat on bean bag chairs and people gave presentations. You'd think it would be relaxing but people just shot you with bullets from the bean bag chairs.

Chuck Thacker was an intense, brilliant man. Always happy to challenge you with a new physics puzzle. He had kind of a crooked smile and a laugh. He smoked -- that was something we didn't think about back then.

And then Bob Metcalfe, one of the most remarkable people I've known and worked for: a very vibrant imagination, very strong-willed, wry sense of humor, rebellious, politically agilent, inspiring. He was willing to be part of a big company like Xerox [for a while anyway]. Everyone felt they had created their own kingdom. PARC was a castle on the hill. Bob Taylor was very strong willed, paternalistic manager, [focused on keeping everyone working together while doing their best], but with a mischievous laugh as well.

One additional note about PARC is that there were two personality types: the free-thinking radicals and hippie-likes on one side versus the conservative, more traditional working people on the other. I think the tension between the two world views at PARC helped to engender success. Later over the years I noticed that the most successful companies or groups supported a natural tension between these two ways of thinking. It was certainly present at PARC.

Ogus: And one thing you told me was you felt that both SDD and PARC, even thought they were part of a very old, traditional company, really gave you the feeling of being in a startup culture there.

Garner: Yes, yes. Everyone felt like we were about to change the world. I mean, this is 1977. There were only character terminals and punch cards. And here they had created, the people at PARC, starting in 1973, an environment with WYSIWYG graphics, email, laser printers, you know, everything in place, and it was just amazing… I would take printouts of, let's say, Russian poetry, and show it to someone and they would react, "How did you generate that?" We all knew we were going to change the world, but we also didn't realize that we were ahead of ourselves and we were ahead of the market...

Ogus: Well, they were right. It just took a little longer than they planned.

Garner: Yea, unfortunately. So my first asignment, as I was saying, was to co-design the first commercial Ethernet controller [with you, where you focused on the D0 interface and authored the microcode]. It was then called the Xerox Wire internally. The experimental one at PARC had been called Ethernet, but the project code name, Xerox wanted to keep it to itself, and so renamed it the Xerox Wire. It was this "Information Outlet" with like a commodity wall plug outlet as the vision.

The initial speed goal that Bob had for the Ethernet was 20 megabits per second. Dave Boggs, who'd co-designed the first Experimental Ethernet at PARC, related that "The best way to implement it is to have separate transmit and receive sections on the board where you can do full loopback. A transmit packet can be sitting there ready to go or a packet can come in at any time. So I did that. I did a VME-sized board for the Dolphin about this big [one at 3-Mbps and another at 10-Mbps. This allowed the D0 to later be a gateway between the experimental and product Ethernets.] Thus, it was full-duplex: one-half was transmit, one-half was receive. [Appendix Photo 1-C shows the D0/Dolphin 3-mbps Ethernet controller]

But a challenge was that you have to eventually place and route all the TTL components on a printed-circuit board. According to the productization group in El Segundo, the chips weren't going to fit. On my desk was a Fairchild Data Book. One day I opened it up and there was a 10-megahertz CRC chip. So I went to Bob and I said: "This doesn't fit on the board at 20 megahertz." I even went to the Stanford library and investigated ways to do CRCs differently: parallel vs. serial and all this kind of stuff. (Today, you would just google it on the Web.) And I said: "Bob there's no way I can get this to fit. However, if we use this 10-megahertz Fairchild part, it was an F9401, it will all fit at 10 Mbps." So that's why the first commercial Ethernet ran at 10 megabits per second. [Unbeknowst to me at the time, Boggs had used the same F9401 in his 2nd spin of the Alto 3-mb Ethernet adapter.] Also, Ron Crane, who was working out the analog signaling on the cable at 20 MHz -- DC collision detect -- there was a bit too much reflection from the taps.

Ogus: I remember there were electrical issues as well.

Garner: Yes. So that helped push the speed back down to 10 Mbps. But 10 was still so much faster than anything in the world that it seemed ludicrous that we were trying for 20.

Ogus: Yes, we might note that the Alto Ethernet speed was two-point-something which was the clock on the Alto. There was nothing magic about that number.

Garner: Yes, the Alto Ethernet ran at 2.94 Mbps.

So the D0/Dolphin was an eye-opening experience. Chuck had a self-assured ego and I don't think he was always careful on some things. He wouldn't always do a comprehensive static timing analysis. The technology of the prototype boards was called stitch welding, adopted apparently from a satellite or spacecraft program. I recall the large stich-welding machine -- you could place a board into it and it would directly place a wire between two pins and then weld it to pins on each end. So you could spin a board in just in a few hours. [Appendix Photo 1-C shows the stichweld wiring on the back of the D0 3-mbps Ethernet controller.] If you needed a change to the netlist you could submit the new netlist to, I can't remember her name, the woman…

Ogus: Rosemary.

Garner: … Rosemary who ran the stitch welding machine and you'd get this new board back. So I'm not sure it encouraged -- it didn't encourage simulation or thinking about the design as well up front, but you could very rapidly turn around the board. We went through many iterations before committing them to an etched printed circuit board. But one of the problems with stitch welding is that the welds weren't always reliable. I'll never forget having a Dolphin board out on the card extender and twisting the board to get it to work. That was not a good sign.

Ogus: Yes, it was very good to rapidly prototype but it was not reliable.

Garner: Yes. I think there were timing issues in it as well. But, anyways, also, the Dolphin was very expensive. And it was kind of designed for a multiprocessor perhaps with a very capable memory system. One problem that it inherited from the Alto was that the microcode would dictate when to explicitly give up control. So a microcode task could take control of the CPU as long as it wanted. Chuck and I had a constant on going battle where I'd come in in the morning and find that the cursor had stopped working, as I was doing the cursor microcode for him. And I would say "Chuck,

I'm losing cycles to your disk microcode." So he would grumble and go fix his disk microcode and my cursor would work again. But the next day the cursor would stop working because he-- so we had this constant battle of who would get control of enough CPU cycles to get work done. And there was no way to allocate it, the bandwidth, among all the users. It was just a free-for-all and it was based on priority ranking, like a traditional I/O bus. So, it just didn't work that well, a painful lesson. Chuck was the interface to the productization group in El Segundo. He had to fly down to El Segundo all the time. The old Scientific Data Systems (SDS) people had been converted over to help bring our products to market and productize them.

Ogus: So one question is, was there any serious consideration to productize Dolphin?

Garner: Yes, there was.

Ogus: Is that why he was working with the El Segundo people?

Garner: Yes. He was going down there. The D0 was on track to be productized but the decision was eventually made that it was too expensive. And it seemed to have timing problems. It was an occurance of the second system syndrome. It was a tough decision that someone made to cancel it as the go-to-market workstation. [It was however incorporated into the flagship Xerox 5700 Laser Printing System announced in 1980, with a list price of $91,000.] And then at the same time Butler Lampson and Roy Levin at PARC came up with a paper design which addressed the fundamental issue with the two previous designs with a new fixed tasking schedule to access memory and run device microcode.

The new microcode scheduling framework was called a "round" and there were five "clicks" in a round. [Three microinstructions and one memory access could be executed in one click.] I/O devices were assigned to one or more clicks depending on their memory bandwidth requirements. So an I/O controller always knew when its click's microcode was going to run next and get access to main memory within certain number of microseconds. So as long as the system met its cycle time, I/O was guaranteed to work [i.e., there could be no device buffer underruns or overruns]. It was just a remarkable aspect of the Dandelion. It was called Dandelion -- that was the name of the machine eventually.

Ogus: I remember it. It was actually called Wildflower in the paper design.

Garner: Wildflower was the name of the PARC design, right.

Ogus: But that really was the impetus to start Dandelion. So, what you next worked on was the Dandelion system?

Garner: Yes. Yes.

Ogus: And that, of course, was – I use the word – seminal system. It changed the world. So, perhaps you could tell us a little bit about what was Dandelion all about? You started talking about the architecture that was really ahead of its time. And then you started working on it.

Garner: Yes, yes. I went back and looked at my notes. It was interesting. Actually, myself and Bob Belleville, who was my manager at the time, we were both initially skeptical of actually using the Alto Dolphin and Wildflower architecture for a product because we had experienced the Dolphin's bad track record. I actually wrote a memo, "Is this like the dinosaur era?" But eventually I came around to the realization that the Wildflower was both low cost and had this unique feature that if we met the cycle time goal it'd be guaranteed to work.

And so the Dandelion was a very synchronous machine. It could be viewed as an appliance built for the set of I/O devices it supported. It was so synchronous that 7 display bit periods -- each one were like 19.6 nanoseconds -- equaled one CPU clock cycle, 137 nanoseconds. (By the way, 137 is my favorite number – it's the fine-structure constant from physics. So I always thought that was cool -- 137.14, very close to the fine structure constant.) And there were 12 display/pixel periods per disk bit. And 3 clock cycles per memory access. So the entire core of the machine was synchronous. Now, the Ethernet ran asychronously at 10-megahertz so it needed a FIFO and a synchronizer. Otherwise the entire machine was synchronous. So we just had to meet the cycle time goal or the machine wouldn't work. But if we did, it was just going to work.

Ogus: It was guaranteed to work.

Garner: Now, the I/O processor, which you worked on, was also asynchronous and handeled the low-speed I/O.

Ogus: But that was a whole separate processor. And then, of course, you had the interface between the two machines.

Garner: Well, and you used an 8085?

Ogus: Yes.

Garner: It's kind of ironic that the germ of the demise of Xerox's D machines lay in the Intel 8085 in the bowels of the Dandelion.

Ogus: The I/O processor is kind of like a PC, right, that Xerox had a little after that.

Garner: Yes, it was. It ended up eating the whole rest of the market alive a few years later.

So there was this round robin scheme and two slots were dedicated to the Ethernet: one click for the disk, one click for the display, one click for slow speed I/O, your stuff. And then when any I/O microcode wasn't running the Mesa language emulator ran. So I was responsible for the CPU card design which was based on the four-bit AMD 2901-bit slice. We had 48-bit microinstructions.

I was also responsible for the Dandelion 10-Mbps Ethernet design. And at this time because we were short on real estate I decided to balk at Boggs' advice and go half-duplex. Since the cable was half-duplex -- it was only sending or receiving at any one time -- why not make the controller half-duplex? And receiving would have priority over sending. Boggs wasn't too keen about that, but it ended up working just fine. Ron Crane designed the analog phase-locked

loop part of the Ethernet controller and its interface to the transceiver [designed by Tat Lam]. <Appendix Photo 1-D shows the Dandelion Ethernet controller and CPU board.>

And Ron specified the Ethernet cable signaling levels. And, a big event, he located and repaired several Ethernet transceivers after they were blown out by a lightning strike. [During the early morning of Jan 5th, 1978, a lightning strike blew out half a dozen transceivers in a 3-Mbps Ethernet strung between two Xerox Palo Alto buildings.] Ron also designed the display controller, where he even invented a clever addressing mechanism to smoothy scroll the screen [not adopted by the Star developers]. Dan Davies designed the disk controller. Roy, you designed the low-speed I/O processor (IOP) based on the Intel 8085.

My friend and mathematician Don Charnley wrote the Dandelion BITBLT microcode, the microcode that moves a rectangular area of memory to another rectangular area. He also wrote the Dandelion microcode assembler called MASS. I wanted to implement an arbitrary 16-to-1 rotation in the Dandelion processor because you needed that kind of thing for BITBLT. Don suggested an arbitrary four-bit rotation and followed up by two one-bit rotations to get an arbitrary rotation. So you could do an arbitrary 16-bit rotation in 3 microinstructions. That was a neat suggestion on his part. [Appendix 1-E shows the Dandelion hardware design team]

Other people did various microcode: Jim Frandeen, Amy Fasnacht, and Jim Sandman. Also at that time they asked me to become owner of the Mesa instruction set architecture (ISA). Mesa was the proprietary language that had been developed using an instruction set architecture document that had been modeled after the IBM PrincOps document, something I wasn't aware of at the time. But that taught me about the rigor of meticulously defining an instruction set with very precise definitions. I had started defiining them using Gordon Bell's ISP notation, but the Mesa developers asked me later to convert it over to Mesa code because then they could then take the Mesa code and just run it on a simulator. That was a better way to make sure the opcodes were defined properly.

Ogus: Do you remember how it ended up that you were given ownership of the Mesa language? Because you were someone in the hardware team so it's unusual that they chose you.

Garner: Yes, I don't remember.

Ogus: Very appropriate, I think. But just the fact that they did that was kind of unusual.

Garner: Yes, I thought it was pretty cool. I guess I was responsible for parts of the microcode for Mesa emulator. So in that sense it made sense for me to do both.

Ogus: Perhaps that's why.

Garner: Now, something that I characterized at the time, what I dubbed the "Mesa high priests." (I can supply names later. [Richard Johnsson, Ed Satterthwaite, Dick Sweet, Jim Sandman]) The Mesa instruction set was a classic what Dave Patterson called a CISC [Complex Instruction Set Computer]. Every week the Mesa developers would rejigger them to make the static code size smaller based on compiling some sample programs in order to fit in a small amount of memory. We only had -- we weren't sure whether we were going to have 384-kB max or 128-kB max main memory. That's not very much main memory (16-bit words). So they were remixing the instruction set every week to shrink

code size. And also to leave some data on the stack; so maybe subsequent instructions could -- further down the stack -- use those values. A very CISCy approach. Everyone was kind of doing that at the time so few thought anything of it.

Ogus: We'll later talk about CISC versus RISC. Can you talk a little bit about the bring-up process for Dandelion because you implemented many innovations there as well?

Garner: Yes. We had no experience in this, but I thought "Wouldn't it be cool to have all five boards be probeable simultaneously?" So I proposed what was called the Windmill which was a central structure with a flexible vertical circular backplane incorporating all five board connectors. [The technician Jim Cucinitti built it.] You plugged the boards in like petals of a wildflower or windmill. With my background in natural history the name made some sense. This was the same way that the Cray-I was constructed. I learned later that the IBM 801 RISC prototype was done exactly the same way, all its boards were mounted around a vertical column that you could squish together or open up. So that way we could examine signals simultaneously on boards without having to put them on card extenders.

Ogus: I remember Cray's production machines are like that too for access later.

Garner: Yes. The Cray-1s were. They had trouble with card extenders, too. But the funny part, though, is that our printed-circuit cards were stitch-welded and the characteristic impedance of stitch-welded lines is higher than on a printed circuit board (PCB) so we were taking somewhat of a gamble implementing with stitch-weld first where signals propagate faster than in printed circuit boards. I was sort of aware of that, which also led me to be more conservative in the timing analysis because I knew when we switched over to PCBs with their lower characteristic impedance, like 50 ohms instead of 100 ohms, the propogations delays would be even slower with more crosstalk. It was a risk, but it seemed like it worked.

Ogus: Right. And then we had a debugging system that connected to an Alto to debug it.

Garner: Yes, that was so cool. So we used the previous computer the Alto to debug your current one. So Pitts Jarvis developed this thing called Burdock which was a beautiful user interface where you could set breakpoints, you could run microcode, you could look at registers. There was an umbilical cord (we called it that) which went over to the Dandelion. Did that go through the IOP, I think? I don't remember. Did it go through your board?

Ogus: Yes, it did, actually. Yes. So do you remember why it was called Burdock?

Garner: No.

Ogus: Well, there's an English drink called "Dandelion and Burdock". And there was an English researcher who was working with us, Dick Snow, and he came up with the name Burdock. So that's a little piece of trivia!

Garner: It was a beautiful debugging environment. It's too bad something like that couldn't be commercialized. So with microcoded machines you could have multiple sources going to multiple destinations. And I did a static timing analysis of all of them and some of them just didn't work. So that was kind of ugly. And we had the assembler, MASS, that disallowed those operations. So there was a big chart that showed which sources could be sent to which

destinations and what was illegal. Ugly. But I think that's what helped us to get it to work. So I was kind of proud that it was working, coming up in the lab.

And I remember one day we invited the product engineering people from El Segundo up into the lab and introduced them to the design. The CPU board was sitting there on an extender card and I was so proud of the thorough diagnostic program that I had written, I said: “Let's pull out a random chip from the board and see what happens.” Boy, what a learning experience that was! I amble up to the CPU board and I take the little TTL DIP chip remover and pull out a random chip. And, low and behold, the diagnostic did not detect an error and is still running! I \ looked at the chip in my hand and I'm like “I think I missed something! Or perhaps we don’t need this chip!” <laughs> I scratched my head, and examining it later, I had written the diagnostic -- and being new to diagnostics -- I had put the expected answer in a register and then computed the answer and then compared them. Well, the expected answer went through a driver onto a large shared bus called the X-bus. And since I had removed its source driver, the new answer didn't get put on the bus, but instead the expected answer that my diagnostic had supplied earlier was still there. There was so much capacitance on the X-bus that, without a source driving it, the expected value actually hung on for an entire click of 411 nanoseconds. And so even though I pulled out the driver for the new source value, the correct value that had been put there by my diagnostic was still capacitively sitting there on the bus. So I learned that in diagnostic programs you should never -- what you're expecting -- put the right value right before checking for the test value because it might hang around for some unforeseen reason.

Ogus: Perhaps we can talk some more about the Mesa development of the PrincOps instruction set. The Mesa work had been done for many, many years independent of Dandelion. There must have been some other tools that had to be developed for Dandelion like Burdock was one. But what about the microcode controller—microcode compiler. Do you remember who did those tools?

Garner: Well, the microcode assembler you mean?

Ogus: Sorry, assembler. Yes.

Garner: That was Don Charnley.

Ogus: That was Don Charnley.

Garner: Yes, Don Charnley did the microcode assembler, called MASS. The Mesa compilers and all of that, I don't remember the names. The museum should do a whole history on the Xerox Star. One thing I remember when I started -- I almost forgot to talk about it – when we made this transition from the paper Wildflower to the actual Dandelion design I queried Butler Lampson : “Butler why don't you want to help implement it?,” And ditto for Roy Levin. Their response was: “Well, it's easier to design it on paper than to actually get it to work!” And Butler also famously quipped: “The trouble with paper designs is you don't get bloody enough.” And so I thought that's interesting -- time to get bloody!

Ogus: Tell me a little bit about how long it took to develop the Dandelion?

Garner: So I thought we did it pretty quickly. We started the design, according to my notes here, in mid-1978, so a little less than a year after I joined. So the whole experience with the Dolphin got over pretty quickly. The first Dandelion CPU board was stitch-welded in July a year later, runnng diagnostics a month later, Ron Crane's display was operational December '79. The first Mesa program ran in March 1980. Then I started to work on the Ethernet. So I did the CPU, then the Ethernet in sequence, and that was the same time the formal Ethernet "Bluebook" spec was being hashed out. We transmitted packets to a Dolphin in August 1980, just 8 months later. We committed the board to etch in September 1980.

This was at the very time Digital Equipment Corp (DEC), Intel, and Xerox announced the "Bluebook" spec for Ethernet, DIX, as the triumvirate was called. The authhors involved included Gordon Bell from DEC [and Rich Seifert, Peter Nesbeda, Tom Ermolovich, Tony Lauck]. I can't remember who was from Intel [Rob Ryan, Bob Beach, George Marshall]. And from Xerox Bob Metcalfe [and John Shoch, Dave Redell, Ron Crane, Yogen Dalal, Will Crowther, Bob Printis, Bill Lynch].

Metcalfe recognized that Xerox would likely keep Ethernet proprietary. That was the thing you did back then, you kept your golden jewels private. Remember that? The STAR [also known as Janus internally] was a very secret project. Everything was proprietary and secret, not open. But Bob [and David Liddle] somehow convinced Xerox to open up [and license the Ethernet patent out for a minimal fee] – I don't know how they did it. [Dave convinced management that there would be little value to Xerox competing in a low-margin Ethernet hardware market.] That's for his oral history and I hope he's done it – convincing management to open up the Ethernet spec.

Ogus: Right because it was against the philosophy at the time.

Garner: It was totally against the prevailing philosophy. But for us, what was fun about it is that Ron Crane was actually authoring the low-level spec. Remember, he wrote it in Star, actually. It was a Star document. And I recall checking with Ron: the only thing that significantly changed from Ron's spec and our work was the bit ordering [high-order bit out first to low-order bit out first], and also, the CRC was enlarged from 16 to 32 bits [and the packet source and destination addresses were widened from 32 to 48 bits, mainly due to Will Crowther's foresight]. But those were the only changes made at the link level between our work and the official DIX spec. So we kind of felt like -- hey we've got the Ethernet controller going to market at the same time the spec is going public, so, great, we have a time-to-market advantage. We weren't bothered that it was becoming an open standard.

Ogus: Yes, and that happened after the announcement of Star.

Garner: Well, my dates show that it happened in the same month. So in September 1980 the DIX Spec was signed.

Ogus: Oh, so it was actually earlier because the Star was announced in '81.

Garner: It was actually earlier.

Ogus: So it was earlier.

Garner: One thing I remember working with Hal Murray, he did the device driver for the Ethernet and I did the microcode. He did the device driver. So we had the battle “Whose bug is this?” and endless nights debugging.

Working there at Xerox was like at a startup. We worked past midnight. I used to go home typically at 2:00 or 3:00 A.M., ride my bike by the Alta Mesa Memorial Park graveyard in Los Altos which was kind of spooky. I would wait until 11 o'clock at night to call my girlfriend Robin who had temporarily moved to Bakersfield to save money on the bills that you probably got. Xerox SDD was truly a startup environment.

And I think back about it at-- some of the things I didn’t mention earlier, all of my outdoor trips when I was younger. I did solo backpacking trips when I was in high school to the Chiricahua Mountains, Arizona. The cabin my father had built south of Mormon Lake near Flagstaff -- I’d go out wandering around by myself all day or sleeping at night with coyotes around me. So I had a very strong sense of self strength and fortitude and things didn't scare me. So I could stay up all night by myself if I needed to, working on schematics and then biking home by the graveyard at 3:00 A.M. So I think the confidence you get in solo backpacking -- truly in high school, I was doing solo backpacking -- really made hanging around at a workplace late at night not abnormal.

Ogus: So let's see. Sorry, did you have a thought?

Garner: Well, I was going to say when it was announced. Yes. So, again, we worked with the ex-SDS crew in El Segundo to productize it. They did really all that work.

The Dandelion hardware was first announced in November 1980 as the Xerox Information Network System or Xerox INS 8000. So Xerox actually first announced Dandelion as a storage server, or what they called a file server, a communication server and a print server at the end of 1980. The hardware was barely working at that time but Xerox announced it anyways which was a bold marketing step. [Also Dave Liddle wanted to announce it before the Xerox 860 group in Dallas announced their verion of a server.]

And then Xerox announced the Xerox Star 8010 Professional Workstation in April 1981 at the World Trade Center. We weren’t there. Charles Irby, I think, was there. A professional information system: “A personal information system for business professionals whose main job is to create, interpret and manage information and distribute the results.” So email, graphics, printing, all the things that people waste their time on today.

Ogus: Yes. I remember that the innovation – this whole idea of servers was a new thing – a client/server paradigm. And that was to their credit – something new that came up.

Garner: Yes. That's a good point. They came there, right there.

Ogus: Because in the beginning everything was focused on the workstation. And no one thought about the servers that were needed.

Garner: No, that then made the entire system expensive, too, unfortunately.

Ogus: Yes, it was expensive.

Garner: The Xerox Star 8010 Workstation was announced with a 16, almost $17,000 purchase price, which would be $50,000 today for a 1-MIPS workstation (although it did move things quickly on the screen), or $700 per month rental which is $2000 per month today. It was also shown at the National Computer Conference (NCC) in May of that year (1981) and that's where Steve Jobs checked it out apparently. He was there to see what Xerox had brought to the market. The DIX Ethernet announcement happened in May, 1980 [and the Ethernet Bluebook 1.0 spec was published in Sept, 1980.]

Ogus: Okay. So, of course, the Star was a phenomenal technological success but not a business success.

Garner: No. Well, first of all, it was a huge investment. I mean there were eventually hundreds of people on the Star project, around 350 total [employess plus contractors] on the hardware, software and the operating system…

Ogus: Probably three quarters of them were software.

Garner: Were software people, yes. So to bring to market all of that—yes, way ahead of its time. So at $50,000 in today's currency for a single workstation, professionals said, “OK, look, I don't know how to type.”

Ogus: Or “I don't want to type.”

Garner: “I don't want to type.” I have secretaries that type for me at that time. “You're telling me you want me to purchase in today's dollars a $50,000 workstation to make my secretary more efficient?” And then you want me to purchase a file server, a print server and a network box as well. So the total cost is going to run to like $200K which would be almost $1 million today. “Just to make my secretaries more efficient?”

Ogus: So you think those are the kind of reasons because if it had been ten times cheaper it might've been different.

Garner: Yes, it would've been totally different. And: “What's this thing called a mouse?” And actually, I went back and looked at the original product release. The original press release called the mouse: “an unusual control device” because a mouse had never been announced to the public before.

Ogus: Yes, I think you're right. Ahead of its time is the right description for the problem.

Garner: And later I learned maybe only 20,000 were sold and that's what Dave Liddle has said is the number. But the major customers I saw and heard of were NSA (during a visit I had with the NSA later) and the Voice of America since they were multilingual. Star was a success in Japan -- the first Japanese Kanji workstation.

Ogus: Yes. So I remember that the Japanese were very interested in it because this was the way to produce their very complicated script in an easy way.

Garner: And in a system that cost quite a bit less. And it was very reliable. We never had heard of any field issues with the hardware. The software was very reliable even though it had a very basic virtual memory system. In November 1981 the Xerox Star received the Fortune magazine Product of the Year award which was kind of cool.

[Shown in Appendix Photo 1-U] Every year Fortune selects what are the coolest products that year in the United States and the world.

So, unfortunately for us, right after the Xerox Star was announced, just four months later, IBM announced this thing called the PC with a 4.8-Megahertz Intel 8088. It was announced just four months later. Everyone felt that the rug had just been jerked out from under them. And that Microsoft and Bill Gates were setting the world back three decades with DOS. The Star user interface had been designed to be intuitive, easy to use. The Star GUI design team consulted psychologists and sociologists. It made sense. You started with an existing document and modified it. You also didn't have these horrible DOS-like command interfaces. Unfortunately, Star didn't feature spreadsheets until later.

So it was all kind of downhill from there. The emphasis shifted from Xerox PARC to Apple at that point. I recall the day that Steve Jobs came by and visited PARC in December, 1979. Larry Tesler, who knew my girlfriend [Robin] at the time, gave the demo. He moved to Apple. Bob Belleville moved to Apple and managed the Macintosh hardware team. He had been gunning for… while at SDD -- remember, he built this little small PC… what did he call it?

Ogus: Cub.

Garner: A little tiny 6800-based or something, or was it Intel-based?

Ogus: Well, I think at that point Intel had come out with a 16-bit processor and he wanted to play with them. And so the Cub was based on that.

Garner: Yes. So he kind of knew that the high priests and Xerox doing their own language (Mesa) that needed a microcode machine and operating system was not going to make it. So Bob Belleville went to Apple and actually managed the Macintosh program at Apple. On which my friend Mike Boich worked against whom I had run for freshman class president in high school.

Ogus: That's right. So then you actually before you left for Sun you were still at Xerox for a while. You moved to PARC per se.

Garner: Yes. Right before I left, I remember there was one interesting thing that happened. This guy named Forest Baskett was coming around and he was into computer benchmarking. He benchmarked Cray's stuff and really understood it. He had this little benchmark called Puzzle which solved a block packing problem that he was running on all the computers he could get access to. He ran it on the Dandelion and the performance was really bad. I've tried to find what the results were, but even though we had a 7.3 megahertz clock, it took three cycles to do a memory access. So we were less than a 1-MIPS machine. So here we are taking what could be a 7-MIPS machine [for non-memory instructions] and making it into a less than a 1-MIPS machine. So that really got my attention. Why had we done this? But I had always wanted to work at PARC proper since we had been over there so much. So, in 1982 I joined PARC itself in Lynn Conway's VLSI group, who was doing…

Ogus: So you joined Lynn Conway's group?

Garner: Yes. So Lynn was doing VLSI design.

Ogus: Which is a completely different type of work from what you were doing before.

Garner: Entirely. Lynn Conway was a very kind and good-natured person. I never saw her angry. She really cared about people. She really engendered the best in people. She was really an educator with her VSLI design methodology. Her whole history is how she created that whole program to get students tuned into chip design.

I started working with Dick Lyon. He had done an innovative design for an optical mouse based on cross-coupled neural light sensors. It could even worked on blue jeans or any pattern that had equal-distant spots. It would stabilize on a pattern, asynchronously stabilize on another one, and if it detected a change in the dot pattern, it could figure out which way it had moved. A very brilliant design. And I helped productize it for the product group. I also worked on some other things. I actually designed a DARPA MOSIS chip using a Lynn's and Doug Fairbairn's Icarus integrated circuit layout program.

Other people did some more complicated chips but this got my fingers into chip design. I saw that the field, though, needed more advanced software tools. It just wasn't possible to design an interesting chip by hand. Lynn promoted the "stick diagram" method that, in jest, one could say all you needed was blue, red, green, and yellow pencils to design a chip since when red (poly) lines crossed green (diffusion) lines you got a transistor. Blue was metal. But there weren't good design tool methodologies in place yet. John Ousterhout was just formulating his IC layout tool Magic which was really needed. But that's research, that's the nature of research.

Ogus: And then you also worked on the PARC Dragon work which was kind of prescient for the future. We'll get to that later.

Garner: Right. Yes. So while I was realizing the tools really weren't in a place to do interesting chips there was an attempt to do an interesting chip program at PARC called the Dragon multiprocessor. It was a whole new multiprocessor architecture designed with careful attention to the cache coherency protocols. It was based on a split transaction packet-based bus: You could send a memory request and get the result back later so you wouldn't tie up the bus for a long period of time.

Jean-Marc Frailong and Pradeep Sindhu were the lead architects. Chuck Thacker asked if I would work on the I/O chip for it which I agreed to start spec'ing. But right about that time PARC imploded or, I suppose, exploded would be the better word for whatever reason. PARC's director, George Pake, had come down from his nest on the first floor and poked at Bob Taylor's tight-knit cult leadership. CSL folks loved working for Bob Taylor. So Bob resigned in protest. They were trying to get him to do more practical work. Maybe they thought he had too much power. I don't know. But nearly everyone departed with him from PARC's Computer Science Lab (CSL): Butler Lampson and Chuck Thacker. They formed the new Systems Research Center (SRC) at DEC in downtown Palo Alto.

Ogus: Digital Equipment Corporation.

Garner: Yes. So I was standing around all these empty offices in CSL one day with Dave Patterson who had been consulting to the Smalltalk group, Alan Kay's Smalltalk group for some time. He knew me based on my reputation

designing the Xerox Dandelion hardware. And he asked if I might join-- he and his brilliant student from Berkeley, Bill Joy, at a startup doing 68000-based workstations but also planning to do a RISC-based design. And I kind of looked at him like: "A 68000-based workstation company? There are lots of those out there. Why is this one going to be any better than all the other "dirtball" 68K firms?" Dave had been consulting with Bill and the folks at Sun, which was about a year-and-a-half old at that point. And I just ignored him because I was in the PARC citadel. I had spent some time doing a hand design for a Mesa processor. I don't know how practical it was, but it seemed like somebody had to get going on one because there really needed to be a full-custom Mesa chip. Later SSD did that themselves. And then, fortuitously -- this is early '84 my girlfriend Robin Beresford asked me to go pick up an album at Tower Records.

Ogus: 1984?

Garner: Yes, '84, Tower Records on San Antonio Road. When I went in there to pick up the album there was Bill Joy and Eric Schmidt who had joined Sun a few months earlier from PARC walking up and down the aisles talking about work. You never stopped thinking about work at Sun. They spotted me and had contacted me before. They twisted my arm, insisting: "Why don't you come work here?" And so I gave in. If my wife hadn't sent me to Tower Records that night I don't know what my future would've been.

My interview with Andy Bechtolsheim at Sun was odd because it consisted of him explaining how he had developed a new static timing analysis tool -- a spreadsheet! I didn't realize at the time that he had a reputation for also turning an eye to validating timing. The Sun2 design had manufacturing challenges. In fact, my first day at Sun or maybe during the interview, one couldn't help but notice in one corner of the half-empty building, there was the CEO Scott McNealy, Bill Joy, Andy Bechtolsheim and Vinod Khosla plugging and unplugging 68010s until they could find one that worked because there were timing issues in Andy's board. That was quite an eye-opener. Vinod Khosla presented my employment offer and asked, "Do you ever want to be able to afford a house in Silicon Valley?" That was certainly a motivating factor for joining a startup.

Ogus: Let's see, so that's 1984. And so you decided to leave Xerox and move to this new little startup called Sun Microsystems. So what were your impressions when you got there?

Garner: Well, also, before I forget that was the year I married Robin, too. So it was…

Ogus: OK, a very busy year.

Garner: 1984. Sun was a very small company. We were just in one building. But the very first day I was kind of jolted by the realization that perhaps I hadn't moved that far because in the mailroom were printouts, Dover laser printer printouts from Stanford. People were driving to Stanford twice a day to pick up. Sun employees were using the Stanford University Network (SUN) Alto network….

Ogus: Right, Dover was a Xerox printer.

Garner: Yes, it was a Xerox printer. Xerox had installed at Stanford a large number, kind of granted to them a large number of Altos and a Dover laser printer. That's how Andy Bechtolsheim got his motivation to do a better-- a different job, do an open workstation instead of a proprietary workstation like the Alto. [Earlier, in 1978, Andy was a no-fee

consultant at Xerox PARC.] But the folks at Sun were actually using it. I felt like I was still at Xerox because the Dover printouts were in all the inboxes.

There were really amazing people at Sun. Bill Joy was, or is -- was and is -- a remarkable visionary. Just brilliant. He's a visionary of the highest order, a grand weaver of tales, a perception shifter and skilled hyperbolist. He went around doing what I called “cranial acupuncture.” He could walk in anyone's office, software, hardware or whatever, and within 30 seconds totally comprehend what they were up to, and immediately giving his advice. He'd walk out of the office and the stunned person would feel like "What just hit me?" He also had this technique of speaking so rapidly that as you tried to process it and keep up with him you'd start falling a sentence or two behind. And eventually, he just flummoxed you. You'd be speechless and tied in knots. It was one of his classic debating techniques. Dave Patterson had remarked that Bill would speak more than he did in class. He was just a very demonstrative person. On the other hand, Bill could also be extremely pragmatic. He had somewhat of a binary personality: Here he'd be crazy and visionary and then a few moments later he'd be practical and proclaim “We'll just do this!”

Dave Patterson was our eminent consultant. He had a fatherly persona, strong, generous, equanimous, pragmatic. He was really good at bringing people together and tying ideas together as he did at Berkeley. He wasn't afraid of conflict or arguments or fights. Later I learned that he was a wrestler in college, which kind of explained why he wasn't afraid of standing up for something. He would break up arguments, the cat fights we would have all the time at Sun. He would say; "That might be the way you think about it, but I think this is the way it's going to be." He was very pragmatic in getting people to adopt the most rational approach.

Andy Bechtolsheim was brilliantly meticulous and extremely detail-oriented. He could sometimes be stubborn. He only designed products that he knew could be built right then and now at low cost and delivered to market quickly. He didn't really look out two to three years, just the next year. Folks used to characterize his design method as “making a Porsche out of Volkswagen parts.” He's also the only person I knew of who could actually have two thought trains running concurrently. You could actually talk to him about two threads of thought and he wouldn't trip up. Just an amazing capability. He was a super optimizer who also kept himself grounded in the lab. That drove some people nuts years later when he became known to be extremely wealthy from all his investments, but nevertheless there he’d be in the lab doing technical work. He never backed down from his passion for doing world-class engineering.

Scott McNealy was bitingly and hilariously funny – cutting, jocular humor. He had a lot of fun being the rabble rouser for the computer industry. He didn't fully comprehend computing technology, but he fully trusted Bill and Andy. Totally trusted them, which was a great pairing. Also, I'll never forget, outside my office on the second floor, Scott held a company-wide beer bust every Friday. At the end he would give away all the freebies that executives and salesmen had given to him during the past week. He just gave them away, perhaps so he couldn't be overly influenced.

Bernie Lacroute, who came on later, was the Senior VP of Engineering, a very respected, passionate and caring person. We gave him the nickname “Napoleon” because he was so detailed, driven and forceful.

Ogus: So how big was Sun about at that time?

Garner: Well, I was employee 278. I don't know if--

Ogus: Oh, okay.

Garner: You can see that on the badge or not. I was one of the earliest hardware engineers there among, sort of in the top five. Sun had no marketing or sales people, it was all mainly engineering at that time.

Ogus: So a few hundred people it sounds like,..

Garner: Yes, a few hundred people. I started on March 1st, 1984. Sun was two years old. Speaking of that, attending one of the beer busts, Eric Schmidt drew the entire org chart of engineering on one piece of paper for me. That shows how small the company was. <Showing Eric's hand-drawn Sun Engineering organization chart, Appendix Image 1-F> There's Bill Joy, Bernie, Andy, Eric Schmidt, who was head of the software group. It was amazing that we were a fully functioning computer systems outfit at that time.There was a compiler group headed by Steve Muchnick; there was a Windows group, Jerry Farrell and other folks. We already had a GUI group and a graphics group -- Jerry Evans and others; there was a systems group; we already had a large operating systems group -- Bob and Tom Lyon, Bill Shannon, Dave Goldberg, John Gilmore [who later went to LINUX RedHat]; there was a documentation group; there were hardware groups; there was a CAD group/environment -- it was a complete systems organization. So to walk into that saying "Let's do a new RISC microprocessor," we were lucky that a lot of expertise was already in place, increasing our likelihood of success.

Ogus: Right. But on the other hand, you told me that you thought you'd gone a step back when you went there because--

Garner: Oh, Yes, so--

Ogus: You know, the WYSIWYG GUI at Xerox, and then you were UNIX now with this command line.

Garner: Yes, all of a sudden you're confronted with a command line interface (CLI) that looks like a lot like DEC's command line formats: command, switches, a whole bunch of switches and then more switches. You wonder: "What happened to the WYSIWYG editor?" There was no GUI editor… Bill Joy had authored a simple line editor program called vi, Visual Editor, but it was modal. It was like TECO, DEC's line editor, where all you did was change individual character strings. I was astounded, "Man, this startup company is based on this?" But luckily, FrameMaker developed a Star-like application that ran under UNIX on the Sun workstation the next year, called FrameMaker, so that made me feel a little bit better.

Ogus: And they had Ethernet on their machines too, so.

Garner: Yes! Well, Andy really believed in the Ethernet. He was a convert to the entire PARC vision: bitmap graphics, Ethernet, client-servers, and so on. Strong networking performance really helped drive Sun, explemified by their cattchy slogan "The Network Is The Computer."

So at my very first day at Sun with Bill Joy -- I had gone there on the expectation that we would be designing a RISC, Bill surprised me by saying, "Well, that's not really what the plan is at the moment." His boss, Bernie Lacroute's plan

was that the next bread-and-butter product of Sun would be a Motorola 68020-based system with one or two RISC coprocessors on the side.

So Bill and I debated the pros and cons of RISC as a coprocessor. I can see in my notebook several approaches: perhaps an ‘020 and a RISC would multiplex memory, so they could share the same cache. Surprisingly, at the time, Sun was dead last in floating-point performance based on Jack Dongarra’s LINPACK benchmark. Sun had a floating-point accelerator card they were developing, but just based on the 68K, its floating-point performance was bad. So Bernie had directed Bill: "Let's design a floating-point coprocessor.” His response was: "Well, if we’re doing that, the coporcessor should be the fastest type of processor, a RISC." Well, I adamantly told Bill: “I did **not** come to Sun just to design RISC as a coprocessor!”

Ogus: RISC coprocessor, right.

Garner: Yes, I had not signed up there to design RISC as a coprocessor to something else; my expectation was to design it as the main processor. And Dave Patterson felt the same way. Another person that felt the same way was Dave Goldberg, who had been at the DEC Western Research Lab (DECWRL). His advice was that "There's no need for a 68020 anywhere.” and “Every product shouldn't try to do everything." We spent six months getting Bill and Bernie to accept the idea of making RISC the main processor.

So the way we pulled that off is… In development at the time when I arrived was a 68020-based high-end server that Bill Van Loo and John Watkins were designing, 68020-based with a cache memory system. So we said: “We'll just design a plug-in replacement for that board” -- same backplane, same memory subsystem, same I/O. That way if RISC falters for some reason we still have a product. So we got the go ahead. The FCS goal, First Customer Ship goal, was changed from mid-'85 to mid-'86 at that point and we began working on it in earnest.

Ogus: So at that point what was it thought would be used for the RISC processor? I know that we're going to get into a big story about SPARC.

Garner: Well, we had no idea!

Ogus: But it wasn't then yet started.

Garner: We had no idea. So I was the group lead for defining the RISC architecture. Dave Patterson was our consultant, in one day a week. By then the Berkeley RISC I and RISC II were well known and Katevenis's PhD thesis was published. My position was: "Okay, let's look at what we know about all the RISC machines." So I looked at what we knew about the IBM 801, John Cocke’s pioneering machine from IBM, from the paper, and a little bit of inside information, the Berkeley RISC-I obviously. And DEC had a RISC called Titan at DECWRL that Dave Goldberg knew about. The Acorn people had their ARM stuff. Fairchild was doing work.

HP was doing a RISC called Spectrum, later renamed Precision. We knew a little bit about that. Ruby Lee was the architect and she was the husband of Howard Lee who was soon my manager, so there was a little bit of... But they were very tight lipped, though. Steve Muchnick from HP wouldn't share anything about what they had done. Pyramid was doing a RISC and George Taylor had been involved.

So I looked at all these and felt that they were all pretty similar. So I thought “Hell, let's just stick with the -- Instead of inventing something new -- let's mainly stick with the Berkeley RISC I. It's in the public domain so we won't get sued. And they've addressed a lot of problems, so let's mainly follow it. We can make some changes." We needed floating point, primarily. I initially thought of just some straightforward co-processor interface to the Weitek floating point chips, but Bill set me straight that we really needed a proper floating-point instruction set.

Ogus: But the idea still was to design your own processor.

Garner: Yes.

Ogus: A custom microprocessor.

Garner: And we didn't know how best to do that. So we started calling technology firms to see how we were going to implement it. I knew the Xerox Mead-Conway approach wasn't going to work. John Doerr -- who had co-funded Sun as a VC and also Silicon Compilers -- and I visited Silicon Compilers. My reaction was "This stuff isn't going to fly for us. We're not going to be able to design with a short enough cycle time." The right register technology wasn't there. You really needed master-slave, fast flip-flops and not these kind of dynamic CMOS pass-transistor charge-storage latches.

The companies that we talked with, looking at my list, were: CTI (CMOS Technology Inc.); BRI (Barvon Research Inc.); LSI, still exists; WSI (Wafer Scale Integration). How's that for acronym soup? Cypress, TI (Texas Instruments); Alphatron; Toshiba; and then Fujitsu.

So, we looked at ten potential firms and they were all standard cell-based. I still didn’t have a very good feeling, like “How are we going to pull this off?” There were only like three of us. But luckily, we hired Anant Agrawal from STC, Storage Technology Corporation, who were trying to build an IBM-compatible clone -- and they were familiar with Fujitsu's gate array technonlogy.

Luckily, right about then, Fujitsu announced a 20,000-gate gate array. We looked at each other and said, "20,000 gates? I think we could do a RISC in 20,000 gates." So we went with Fujitsu. 20,000 gates, and also a quarter of the gate array could be SRAM.

Garner: And I have a picture of it, a gate array. <Showing photo of Fujitsu SPARC V7 Integer Unit gate array, Appendix Photo 1-G> We implemented the RISC Integer Unit in 20,000 raw gates [12,000 actual gates utilized] with a quarter of the die the register file for Berkeley's register windows.

Ogus: So the idea was to use a standard cell?

Garner: Well, the gate array--

Ogus: Or gate array, actually, .

Garner: No, that was a gate array.

Ogus: But not full custom.

Garner: Not full custom. And Fujitsu did not have the greatest tools so we -- the team -- I was not part of the chip design team. Anant and Masood Namjoo and Don Jackson on floating point did that. They used Daisy workstations. Vinod Khosla had been at Daisy before, so they used that as the schematic editor.

Thye didn't have good timing tools. The Daisy net lists would be written to magnetic tape, flown to Fujitsu Japan where they ran on a mainframe for the place and route and timing analysis. So there was a long delay between schematic creation and timing feedback. It made designing the gate array very challenging.

Anyway, before I jump ahead, there was one really big event in April, 1984 that got our attention. So we knew that IBM had the 801. Everyone knew they were proceding with various successor projects based on it. In fact, I later learned when I later went to IBM that my first manager there, Richard Freitas, actually worked with John Cocke on the 801 and designed its I/O controller board. And after that, he actually designed, worked on an 801 system as a coprocessor in the IBM PC! So at the very time that I proclaimed at Sun that "No, we're not going to do RISC as a coprocessor," inside IBM they were implementing RISC as a coprocessor! But anyway, we knew that IBM was doing something but we didn't know what. In April, 1984, we heard that IBM was going to announce their RISC product at Stanford's EE, what is it, 380 --

Ogus: Oh, yes, the 380 seminar.

Garner: 380 seminar. So anybody who was -- everyone who was anybody doing RISC designs -- was in that room. Hennessey, the MIPS guys who hadn't started yet, we were there. We were all holding our breath: "What is IBM going to announce?" Because we knew if they got it right we were doomed. Well, they got it wrong. They announced a chip called ROMP and it was a chip set actually designed for typewriters. No cache. It had direct access to main memory. Five-cycle loads and stores and taken branches. Five cycles! As bad as a CISC. So the cycles per instruction averaged around 3. It was pathetic. So everyone just breathed this huge sigh of relief. MIPS was probably started that day. We went back and said, "Holy shit, IBM blew it! You know, we have an opening. Let's go!"

Ogus: So that really started the whole SPARC track.

Garner: Yes, yes.

Ogus: I mean, I know SPARC's name came later.

Garner: Yes, so--

Ogus: Maybe we could move on to that whole story there.

Garner: Okay, okay. Yes, so we knew we had the Berkeley design to leverage off of but we knew that we needed to make changes: We needed an MMU, we needed a floating-point unit, we needed to tweak it. So I established what I called the SPARC Architecture Committee, so--

Ogus: Which probably wasn't called SPARC actually at the time.

Garner: No. Soon it was. Yea, the way we came up with the name, SPARC, was one of the guys on the team who wrote the instruction set simulator [Will Brown] started with a set of common buzzwords and wrote a program that printed out five factorial combinations of them. One of the combinations that emerged was "Sun's Processor Architecture for RISC Computing," or SPARC. So that's how we came up with the name!

Anyway, I led the committee. We met every week. The idea was to get lots of input from the compiler people and the OS people. Make it a not a slow process but to get everyone's input to design a decent instruction set. And designing an instruction set, it's like designing a house: You want to have the rooms in the right place; they all want to be functional; you don't want doors to nowhere. But everyone wants to put the kitchen sink in every room. It's “everyone wants everything” because they know it’ll be long lasting. Egos get ramped up. Dave Patterson and I helped moderate the egos.

I was also the owner of the SPARC architecture manual, so I had control of what was cast in stone every week. In fact, the version numbers of SPARC were just the incrementing version numbers of my documents, 1, 2, 3, 4, 5, 6, 7. Seven was announced. The more conventional 0.1, 0.2, … version numbers I thought were silly. There are integers afterall, let's just count.

Some of the tweaks that came from the team were simple. Folks like Richard Tuck and the compiler group proposed breaking out from the CALL instruction the advancement of the register window so that the compiler didn’t have to move the register window pointer if it didn't want to and therefore do things like tail recursion elimination better. You can actually not even use the register windows if you don’t want to. So that was a good idea. The compiler folks also insisted on sign extension, load byte with sign extension. And other RISCs, like the DEC Alpha architecture, did not do that and that was an impediment. People really wanted – for the C language -- to see sign extension. We were worried that it might affect the cycle time, so it was not a simple decision there at that low level.

I felt that in terms of the memory management unit (MMU), in the Xerox Dandelion the MMU was tables in main memory. Andy had designed a very clever MMU for the Sun-1 and the Sun-2: It would take low-order memory address bits and use them as memory DRAM RAS address bit and then an SRAM translated higher-order virtual address bits to physical address and then sent them main memory [as the DRAM CAS address bits, which can arrive much later than the RAS bits] . So, a two-stage MMU built out of SRAM. Kind of hacky, but it was brilliant and worked. It gave Sun “zero-cycle” access time to main memory for its 68K-based systems, a material differentiator]. And TLBs would be hard to implement in a gate array, so I said, hell, let's just stick with Andy's MMU. Also, I decided not make the MMU part of the SPARC architecture, because it would be changing in the future and it would likely be different with every kernel and every implementation. Since it's not visible to the user application software, it really doesn't need to be specified in the architecture. [Doing so would make it less flexible. Later Fujitsu came out with a more traditional MMU with TLB chip.]

HP made the opposite decision on Precision. They definated an entire I/O architecture which constrains you, to whatever decisions you make then. For SPARC, we defined memory-mapped I/O, gave it a separate address space identifier (ASI), lots of them, and moved I/O completely out of the architecture. That was something I really pushed

and the OS people supported it. The OS people also noticed that the 68000 had a high interrupt latency and SPARC addresssed that challenge very nicely by giving the kernel a new set of registers on kernel entry.

Designing an instruction set architecture is like designing a home. I had to guard the fort against everyone's crazy ideas. The most vocal person with crazy ideas was Bill Joy who was talking to the LISP folks who wanted a tagged architecture. So Bill actually tried horse trading: "I'll stop asking for features X and Y if you give me Z." I would get so frustrated with him that I -- you'd probably get reprimanded for this today -- but in one of the architecture meetings I actually whipped out a water pistol and hit him in the face just to get him to quiet down. <laughs> It was that crazy!

We did end up implementing a Tagged Add (TADD) and Tagged Subtract (TSUB) instructions for the LISP guys. Later the LSIP users at Texas Instruments (TI) noticed a bug in them. I flew out there with my tail between my legs realizing that we'd screwed it up because the tag bits when added together could overflow into the non-tag bits. So my heart wasn't into them I suppose. The Tagged instructions were later deprecated. In the end, the LISP people confessed: "All we really want is a fast clock rate and a large cache." So that just shows you how you can get too wrapped up in designing a new instruction set.

Ogus: So what about floating point? How was that brought into the architecture?

Garner: Yes, well, we did define a quality set of floating-point instructions. We had David Hough on the SPARC architecture committee who was a student of Professor Kahan at Berkeley who was the primary architect of the IEEE 754-1985 floating-point standard. Essentially, it defines single precision, double precision and extended precision. The Motorola 68881 designers were implementing 80-bit extended precision at the time. David was adamant he wanted extended precision in SPARC, that you really needed it to enhance the integrity of floating-point algorithms, so you wouldn't have to think so hard about what to do when values fell out of range. So we did put them in the architecture, but they were never implemented in hardware. I actually ran into David yesterday and he agreed it was probably a bad idea defining them in the architecture because it was never implemented in hardware.

One thing I did is have the floating-point coprocessor be essentially asynchronous. You would issue the floating-point commands and the answer would come back sometime later with register interlocking. I also decided at the same time to define a coprocessor interface where people could find a home for all their crazy ideas, vector units or graphics or whatever. I mirrored the floating-point interface in the coprocessor interface which put to bed -- gave folks a home for the enhancements that they wanted see later on -- putting many of the “kitchen sink” proposals to bed, with less pressure on the core instruction set.

I also sort of, like run-length encoding in information theory, defined a very small opcode space. I allocated a quarter of the opcode space for the CALL instruction, which meant you could branch or call branch anywhere in memory in just one instruction. This kept the opcode encoding really compact to discourage people from inserting more stuff. Folks eventually used the co-processor interface to add functionality, but that was one approach to keep the design simple.

Ogus: So, let's see, at this point, the SPARC was a design only – there was no implementation done yet, right?

Garner: Yea, we were starting to look at Fujitsu’s gate array technology.

Ogus: Okay, so again, the gate array was their approach?

Garner: Yes, it was, so we were starting to look at it and we realized right off the bat that there wasn't room for an integer multiply instruction because there weren't enough gates in the 20K gate array. 20K total, but usable was probably 15,000. There are more gates in an I/O pad today than there were in our gate array! <laughs> So there just wasn't room for an integer multiply instruction. So I defined a multiply step instruction instead, where you just shift one bit at a time, shift and add. Compiler people were happy with that. No divide instruction. [Integer multiply and divide were later added to SPARC V8.]

One question a lot of people ask is "Why did we stick with register windows? You know, they don't exist in other RISCs today. All you need is a lot of registers so that the compiler can take advantage of parallelism." Well, it was a very simple decision. I called a meeting one day, and in the room there was Steve Muchnick on my right. No, he was on my left, and Dave Patterson was on my right. And I asked, "Guys, are we going to do register windows or not?" And Muchnick responded first, based on his experience at HP, which he was very tight lipped about: "Well, if we're striving for high performance we need a coloring algorithm for register allocation and I'm going to need to hire two more people." And Dave Patterson followed with, "Well, windows gets you most of that performance for free, dynamically."

So I reasoned: "Okay, easy decision, we're sticking with register windows." And that's the way it went. It was nice having a lot of registers. Later Bill Joy came up with a clever implementation idea: since we had a current window pointer untethered from the CALL instruction, an implementation could pre-address a smaller group of 16 registers in advance and not be slowed down by the large 300-register file.

Ogus: So let's see, so you gave an example of the multiply and that you couldn't fit it in so you made a compromise. Did you actually implement the gate array version of the SPARC?

Garner: Yes, that was the first-- this was the--

Ogus: That was the first implementation..

Garner: This was the first one, yes, so--

Ogus: Okay, so--

Garner: I'll talk about the schedule in a minute.

Ogus: Yes, okay.

Garner: This was the first. <Showing a photo of the SPARC V7 Fujitsu gate aray, Appendix Photo 1-G>

Ogus: You're right, that's a chip, yes.

Garner: And I'll bring the board up and we can look at it. Yes. So, one thing that happened during this time is I got this call from the "No Such Agency," the NSA. They told me: "Meet us at so-and-so parking lot in Mountain View." And I said, "Well, how will I know who you are?" They said, "Oh, you'll know." So I show up in this Mountain View parking lot and there are these individuals who showed me their NSA badges and during our conversation they proposed, "We'd like you to consider adding a Population Count instruction to SPARC."

Ogus: Population count, okay.

Garner: Yes, a POPC instrucction. It was in the Cray computers and in other high-end machines. And I said, "I don't want to do that. It will affect the cycle time." I knew it was used for decoding–decrypting transmissions. I responded, "Look, you can do population count in just a few instructions. Cache misses and memory effects will be more important." Also, one of the homework assignments I had as an undergraduate, was to write the smallest program to count the number of bits in a word. My assembly code was the shortest of anyones. I wrote it smaller than the instructor had ever seen it. So I knew how few instructions you could do population count in, like, just seven or eight instructions. And so I refused to add it to the SPARC architecture and, well, they twisted my arm by flying me to Bethesda to the NSA headquarters. That was pretty eye-opening: "Stand by me at all times. If you ever walk too far away, the security lights go off." And I said: "Well, what if I have to go to the bathroom?" " That's where I saw a large office floor filled with Xerox Star Workstations. The NSA could afford them. There was lots of fan noise. And I'll never forget, on the way out of the NSA headquarters, the guard asked to see my papers – we didn't have laptops back then – because he wanted to count them. And I said, "But you didn't count them on the way in." So that was really strange. [POPC was later included in SPARC V9.]

Ogus: So they had heard about what Sun was working on?

Garner: Yes. Everyone knew that we were doing a RISC.

Ogus: Okay, they–

Garner: And I guess that included the NSA. And firms would come to us pitching their machine. I don't know if I listed some.

Ogus: Yes, you mentioned some of them. I mean, Motorola was doing their own research and–

Garner: Oh, yes. I need to talk about that. Yes. There was–

Ogus: Well, Motorola had–

Garner: Well, well let me--. So right in the middle of this work – even though we're starting on the gate array and I have a memo, it's 1986 – management was concerned whether we were doing the right thing. Afterall, Sun's matra was: "We're OPEN, They're CLOSED." And we were based on open BSD UNIX and VME bus standards. We told customers that if Sun ever disappeared you could take your application software to another company since this UNIX will run there. Of course, that really wasn't true. We made a few OS changes that made it Sun-hardware specific, but it was a good story.

So people were concerned. People would say to me, in fact I think this was John Gilmore: “If RISC is such a great idea, where is the RISC startup company?” MIPS hadn’t been founded yet, but coincidentally, they were founded just a week later. People would proclaim: “We’ll need the entire systems group, three people just to port the software to it.” And then another concern was: “Since everyone was doing RISC we might get scooped – someone could come up with a better one.” MIPS was in full gear at the time. So management hedged their bet by having all the firms developign RISCs come visit us. MIPS was engaged more at a higher level with Bill; but AT&T came by; Fairchild came by – they were doing a VLIW RISC called Clipper; INMOS Transputer came by; Bell Labs came by; and Motorola came by.

We nevertheless spurned all the offers saying, “Thank you, but we’re doing our own RISC.” Eric Schmidt once related to me, “I’m not sure if RISC is a good idea, but we’re able to hire a lot of good people because we’re doing a RISC.” So it was justified, from his perspective, as a good hiring attractant.

Motorola’s RISC development was under Roger Ross and the architect Mitch Alsup. Mitch had been inspired by Denelcor’s HEP machine, which was an early threaded architecture with three independent streams running so that when you went to main memory, since you otherwise had a long time, it executed other instructions from another instruction stream. Really way ahead of its time. Mitch was architecting Motorola’s 88000 RISC. But Sun was not into multi-threaded, multi-processor products, we all had uniprocessor applications. There was no way–so the cycle time was slower. Just supporting three streams running concurrently was too much complexity. I said this isn’t going to fly and Mitch wasn’t willing to simplify his design. The 88K design did change laster and was produced. During that time, we also invited Motorola management in to see our RISC first hand. So, let me see, it was Tom Gunther and Roger Ross. I guess that was a little later, so I’ll talk more about that later.

So we had Fujitsu’s attention but they also concerned, “Who are you guys? A little scrappy startup? You want to do your own microprocessor and take on these big companies like Motorola with design teams of hundreds of people? Who do you think you are?” My new boss, Howard Lee helped convince them that we could do it. Andy Bechtolsheim himself took a wait and see attitude. He, again, since the timeline for the high-end, expensive RISC server was years out, he didn’t really follow it closely. He wasn’t against it but it wasn’t his cup of tea.

Propituously, Bill Van Loo had already decided on a virtually addressed cache, so that made a real lever for us. With a virtual address cache, I didn’t have to go translate addresses through a TLB. So I designed the integer unit such that the address bits came out of the gate array unregistered, just raw computed bits. Then I latched the bits in an 74AS374 Advanced Schottky register that drove the cache chips. So, the CMOS gate array chip didn’t have to directly drive the cache array which gave us more time to access the cache SRAM chips, because I had an early clock to the external register. So, this enabled us to build a 16-megahertz, single-cycle cache, [something the Motorola 68K chips couldn’t or didn’t do], which I patented, the only single-cycle cache patent.

So, by giving us the flexibility to design our own RISC chip and system, really then, and all the years later, was the greatest leverage for SPARC. It wasn’t so much the instruction set itself and the details about it, it was more that we had complete freedom to design our own I/O interfaces. Later it was multiprocessor shared-memory coherency interfaces; e.g., Andy’s M-Bus or memory bus. So that was the true benefit we got from doing our own chips – we could design our own systems faster than others, including Intel, could.

Ogus: Right. Your full flexibility.

Garner: Full flexibility to design the system.

Ogus: Yes.

Garner: And here's one example of that. The virtual address cache presented some really interesting conundrums to the OS people. With virtual memory you can have two virtual addresses mapped to the same physical address. Processes might map them that way. So we dictated, "Well, in that case the addresses have to be equal modulo the 128-KB cache size" because they need to resolve to the same physical location. So on a cache miss, we compared the physical address stored in the cache tages against the one in the MMU to see if they were pointing to the same location. That was the kind of trick we needed to employ a virtually addressed cache. Bill Van Loo, who was co-architecting the 68020-based server, had already established that. So that was some groundwork that made it easier for us.

The SPARC chip project was code named Sunrise, a nice, appropriate name for Sun. The SPARC V7 chip set comprised two gate arrays: the Integer Unit (IU), like I said, designed by Anant Agarwal and Masood Namjoo. And the Floating-Point Control (FPC), that controlled the Weitek floating-point chips, the 1164 and 1165, was designed by Don Jackson. And then I had responsibility for designing the CPU part of the card which I have here. I'll have to show the Xerox cards later, which I forgot to show earlier.

So this was the SPARC CPU card in all its glory. <Showing the Sun-4/200 CPU card, Appendix Photo 1-H> This is the Integer Unit (IU) chip gate array, a 20K-gate array, and this is the Floating-Point Control (FPC) chip and those are the Weitek 1164 multiplier and 1165 arithmetic floating-point chips. And this is the cache and this is Andy's MMU. And the 128-kilobyte data cache in 16 chips. And this is the actual serial #1 prototype board, so it's got some red wires, the changes we had to make. I actually used this board for many years at Sun for my desktop workstation. Bill Joy teased me a few years ago about the red wires, however we didn't have a logic simulator, really, to simulate all the mixed signals on different types of chips. Yes, 22V10 PLDs -- I don't know if you remember them -- made this board doable… Earlier we had to use small PROMs back in the Xerox days. So, this is the SPARC CPU section, and everything above the center line here was the I/O stuff that Ed Kelley designed: the boot PROMs, the Ethernet interface [Intel's 82586], the time-of-day clock, the DMA interface, etc.. I designed the main memory interface here.

Ogus: So, this board actually became part of a Sun product machine?

Garner: Yes, so, with this board, you could take a Sun 3/200, which used the 68K-based processor card, unplug it and plug this board in its place, and get a Sun 4/200. That was our lower risk way to bring SPARC to market --

Ogus: What version of SPARC was that? V7?

Garner: V7, yes.

Ogus: Okay, because that was the first public product version.

Garner: Yes. So, I owned the SPARC architecture manual, which was reprinted years later, but I also had responsibility for engineering the Sun-4/200 CPU card. So, I was kind of busy working. I used to work… There was one period of time I recall -- I know this might sound strange-- when I worked every day, including Saturdays and Sundays, past 3:00 a.m. for six months. And I'll never forget, I'd be upset if I ever arrived home after the morning newspaper arrived, which was 5:00 a.m. I figured *that* was late. And I remember once walking up to the front door of our house, like at 4:00 a.m., and I pulled out my key card to open the front door! That just shows you how tired I was. <laughs>

Ogus: Didn't know where you were!

Garner: In fact, everyone at Sun was working like that. No one had families. I don't think that was intentional. Also, people's schedules start shifting. It's like we're used to more than a 24-hour day, so people would start coming in later each day and rotating around the clock. It was really, it was really weird.

Ogus: So, that was the first Sun machine that had--

Garner: Oh, but one thing I forgot to mention is in April '84, just a month after I joined Sun -- this is what really convinced me that we were on the right track. In my lab notebook I did a simple analysis assuming a generic instruction mix of 100 instructions, 10% writes, 40% reads, and 20% branch instructions. I estimated what I thought the Motorola 69020's cycles per instruction [CPI] would be based on our knowledge of the 68020, assuming certain I-cache miss rates and zero and one-wait state main memory. Its CPI came out to 5.2 cycles per instruction. So, 5.2 clock cycles is the average instruction execution time for a generic program, whereas RISC, assuming a 5% cache miss rate [and 10-cycle miss latency] and write-through, and some other assumptions, came out to a CPI of only 1.6.

Ogus: Not bad.

Garner: I knew that RISC compilers didn't issue that many more instructions than CISC, so just the ratio was already a factor of three to four. I had initially projected a 70-nanosecond cycle time for SPARC [later it dropped to 60nS], so my estimate was 8.5 – 8.8 MIPS for RISC versus 2.3 – 2.7 MIPS for the 68020, a speedup of 3.2x! My heart raced and I thought "Holy shit, Motorola is toast!" That night when I did this calculation, I knew Motorola was doomed. Motorola dropped too far behind with their 68K line, all because they had a microcode-based CISC. [Appendix Image 1-I shows my notebook page with an estimated 68K vs. RISC relative performance calculation.]

Ogus: So, that's again-- You're thinking of the big picture. There was this actual Sun product that had SPARC in it. What did you call it? The 4/200, is that what it was?

Garner: Four two hundred, yes.

Ogus: Sorry, 4/200.

Garner: Yes.

Ogus: And so, what was the next step? I know there was a big discussion about multiprocessors, and then the feeling was you didn't need multiprocessors, right?

Garner: Yes. Well, before-- before I get into that, I was just looking through my papers that I brought, and one thing that I had fun with, with the architecture, is I had fallen in love with the DEC PDP-10 instruction set designed by Gordon Bell. In an earlier DEC book, he had illustrated the entire balanced PDP-10 instruction set on a single piece of paper: such as Move Halfword Right/Left to Left/Right and Extend Sign, and so on. So, when I wroteup a paper on the SPARC architecture, in honor of that, I just had to show the SPARC instruction set on a single page. <Appendix Image 1-L shows diagrams of the PDP-10 and SPARC instruction sets>

Ogus: And you managed it on even a sparser piece of paper!

Garner: A sparser piece of paper, right. And I showed this to Gordon later, and chuckled about it.

Ogus: So, did you want to get into multi-processor development? You've told me that--

Garner: Oh, I was going to do the schedule next.

Ogus: Oh, okay.

Garner: So, just like the project at Xerox had progressed pretty fast… We were accused for not moving that fast, but we… I don't know, I thought it was not too bad. [Perhaps management was particularly patient with us?] The instruction set architecture was in place up by August of '84. SPARC V1 specification was released by December. We started the gate array and board design in May of '85. We powered up the CPU board without the Integer Unit chip in February of '86, and the very first IU chip came back in March of '86, but unfortunately it harbored a stuck register in it, although it was partially working. The 2nd IU version came back from Fujitsu in April. We booted multi-user Unix in June '86. The floating-point control (FPC) chip came back in July. The second version of the chips booted Unix in December '86 and we shipped the first alpha Sun-4/200 workstation in March of '87 to Lucid, one year after the first chip, basically. Benchmarking showed that we were running integer apps 3X faster than a 68020-based Sun 3/200, so we met my 1984 performance speedup prediction.

SPARC and the Sun-4/200 were announced on 7/7/87, at the Hayden Planetarium in New York City. The marketing hype was right on: “The most powerful Sun the world has seen!” at a Planetarium no less. They wanted to roll it out in New York City to get Wall Street’s attention. They also had many ISVs present at the announcement, with their applications all running on several Sun-4/200s. All their programs were written in C for SunOS, so all the ISVs had to do was recompile them, and they bascially just ran. That was really a strong positive endorsement.

Fujitsu reported a demand of 5,000 chips per month by the end of '87 and 15,000 chips per month by the end of '88. Oh, one thing that I had forgotten… Okay, so there were two things before the announcement that happened -- three things, actually. So, one thing that we did is we had a roughly cycle-accurate simulator that Will Brown wrote. Since there was a rivalry between us and MIPS, we [were constantly comparing our benchmark performance figures]. The only benchmark in play at the time was Dhrystone, which is a small synthetic benchmark, and Whetstones for floating point. We would run Dhrystone on the simulator and get a Dhrystones per second figure.

One day, from John Mashey or someone at MIPS, we got wind of this really impressive Dhyrstone figure from MIPS. We thought, "How is that possible?" And we realized… the compiler people realized that you could basically take some of the compiler generated code and outright remove it since it produced a constant, and instead just print the constant. In this case, the MIPS compiler had not done the real work, but instead had just generated the answer instead. That made me realize that Dhrystones were a shitty benchmark. Not only that, its code fit in the cache, so it was [not representative of the type of applications that customers were running on our workstation products].

There was pressure on us not do SPARC because MIPS was ahead of us and showing better Dhrystone figures. There was a period there in the middle where it looked like SPARC could get cancelled, so I authored a memo on why SPARC was able of achieving 10 MIPS and be three to four times better than the 68020. Steve Muchnick, whose team was constantly improving the portable C compiler, was a strong advocate for SPARC, and he helped rescue it by the skin of our teeth.

Nevertheless, Wayne Rosing invited the 68K and 88K development execs from Motorola in, Tom Gunther and Roger Ross, into our lab to see our Sun-4 SPARC board running at 10 MIPS. I showed it to them running on the bench, but they still didn't think it was real. The 68K guys had the 68030 queued up next and they knew it was going to be faster so they just chose to ignore our SPARC-based Sun-4, thereafter to their peril.

Ogus: So, they thought it was smoke and mirrors?

Garner: They thought it was smoke and mirrors. Well, we showed SPARC running in the lab, so anyway… One last thing that happened right before the SPARC announcement on July ‘87… You always announce things in advance and say they'll ship later, right? So, the SPARC machines were shipped later in '87, although one went to Lucid early in '87. Manufacturing was doing burn-in testing, where they put the entire Sun-4/200 workstations in burn-in ovens. In fact, I didn't tell this story… During the first month at Sun after I arrived, I was following a fire truck into the parking lot only to be shocked seeing smoke pouring out from under my office. I thought, "Hmm. I left something powered on last night!?" Sun had a small burn-in machine on the ground floor under my office, that had turned into a burn-up machine, frying Sun's board inventory for that month and almost tanked the company!

So, continuing, we had several Sun-4/200s being tested in an oven and they were getting random memory corruption errors. Main memory was getting corrupted! [Management even suggested they might delay the July 8th announcement of SPARC in New York City if we couldn’t find the problem.] So, on the evening of July 4th, 1987, with the Blue Angels screaming overhead at Moffett Field, there we were in the lab trying to figure out why main memory was getting corrupted. Obviously, your first thought was that it's gotta be a hardware problem: a problem with DRAM refresh in main memory. Or perhaps a cache coherency problem. Ed Kelley -- my friend and colleague who designed the other half of the Sun-4/200 CPU card and later the lower-cost SPARC Sun-4/110 workstation -- looked at me and proclaimed, "It's a software problem!" He had brilliantly deduced that because all these systems were diskless, a noteworthy difference between them and a normal system, there must be some problem associated with the diskless operating system. That proved to be true and a huge sigh of relief.

Oh, and then the other thing that happened is right after the Sun-4/200 went into manufacturing, there was this new company called Synopsys, which was doing gate array synthesis, and they had a static timing analysis tool, and they wanted a real live netlist to test their timing tool with. So, we gave them my Sun-4/200 board schematics, and I'm

thinking "Holy shit!", I'm going to find out I have timing violations in this board!" There were no static timing analysis tools generally available in the open market. Xerox had one, IBM had one-- Xerox had one, but Chuck didn't use it. IBM had one. But here was Synopsys with one, and I'm like worried sick. The next day their analysis came back showing only one timing violation, which was the main cache path, due to the capacitive loading of the 16 SRAMs. It was five nanoseconds off. But luckily, just then, an SRAM came out that was five nanoseconds faster, so that got slipped in during manufacturing at the last minute.

And then, after announcement, SPARC and the SUN-4 received Fortune magazine's Product of the Year Award in November '87. I may be the only co-developer hardware designer to receive Fortune magazine's Product of the Year Award twice!

Ogus: Oh! Very impressive.

Garner: Yes, which is pretty funny. So, after all these late nights of nonstop debugging-- Oh, and I should mention that I married Robin in '84, and in '87 SPARC was announced and won the Fortune Magazine award, and that's also the year our son was born. So, that was another gratifying event that year.

So, after hours of all this nonstop debugging and production ramp-up and presentations, I was pretty burned out. Wayne Rosing had just started Sun Labs, the research arm of Sun at that time, and Ed Kelley and I became its first members, and Dave Ditzel soon arrived from Bell Labs. I spent this time to author a paper on SPARC for the Spring COMCON Conference in 1988. And I attended a RISC panel session of the 14th Asilomar Microcomputer Workshop, and I went back to PARC and gave a talk on SPARC, and it turns out that soon thereafter, Xerox adopted SPARC to run their Viewpoint Star software. So, I thought that was pretty cool.

Also, during this time, in September '87, Stan Baker of EE Times closed his restaurant for the night and invited people, techies from Apollo, HP, and MIPS, and Sun (myself) to work on a new performance benchmark suite for workstations.

Ogus: That's right. I was going to get there.

Garner: And I suggested we call it the “Standard Performance Evaluation Co-operative" or SPEC, and the lawyers changed the "Co-operative” -- because that's a legal term -- to "Corporation," and SPEC was born. So, we accumulated a bunch of standard integer programs that didn't fit in caches, and vetted them, and that started the whole SPEC process. That was music to everyone's ears because then we had at least a half-realistic set of benchmarks instead of Dhrystones and Whetstones to gauge performance.

And then SPARC International was formed in… Well, right before that, I guess, Sun announced a set of SPARC licensees to open it up: Cypress Semiconductor, LSI, and Bipolar Integrated Technology (BIT). LSI and Cypress embarked on a 25 megahertz SPARC integer unit chip -- full custom for Cypress, and standard cell for LSI. The LSI chip came back first. Eventually, 20 SPARC licensees were announced. The Cypress project was interesting because Joan Pendleton, who was part of our SPARC team [and had proposed a vector instruction set for SPARC], really wanted to architect the Cypress chip, but the Cypress management [T.J. Rodgers] did not go along with that. And then Fujitsu came out with a 25-megahertz SPARC chip for the SPARCStation1.

So, that was promising. Earlier, Andy had basically thought that SPARC was too far out, too expensive. The Sun 4/200 cost around $30,000, or about $80,000 in today's dollars. But he started to think about a low-end system. Andy badly wanted to do a low-end workstation, but Bernie and management did not think it was a good idea for Sun to be in the lost-cost market. So Andy actually spun out of Sun for a couple weeks [forming Unisun]. Management soon came to their senses that it wasn't exactly good PR to have one of your founders depart.

So management gave Andy his own division called the Education Division. At the time, Steve Jobs was targetting the education market, so Andy was in a way trying to compete with Steve, trying to formulate a lower-cost entry point. Working with a small team, they designed gate arrays with LSI Logic and designed a new I/O bus called the SBus. "S" stood for "school," as in "School Bus." Their low-cost workstation was called Campus.

Andy had a choice of processors: the upcoming Motorola 68040 or SPARC. Ed Kelley basically met with him daily and contended: "SPARC is here now. Higher performance than the future 68040, you should go with SPARC!" and over two months convinced Andy to chose SPARC. Soon, Sun's competition unexpectantly announced less costly workstations. Andy and his Education Division were suddently heros and Campus was announced as the SPARCstation-1 with LSI's 20-MHz SPARC chip. That's what really put SPARC on the map with its high volumes and a pizza-sized box that sat nicely on your desktop under the monitor.

Also at abot that time, Sun received a thick envelope in the mail, about five inches thick. IBM threw a stack of patents at us saying, "You, little startup, might want to take a look at these." They included patents for the IBM 801 RISC processor, and things like channel controllers for a 7090. If you had a Direct Memory Access (DMA) chip in your computer, IBM argued that you needed a cross license because their 1960s channel controllers had basically implemented the concept of DMA.

And IBM also had a patent on backspace -- if you had backspace in your user interface [who didn't!], since their selectric typewriter had a backspace key. The patents included complete mechanical drawings of a selectric typewriter and the schematic diagrams of channel controllers. But we didn't infringe on any of the IBM RISC patents because the 801 had separate instruction and data caches. Nevertheless, Sun's lawyers decided to cross-licensed with IBM anyway to cover for future potential IBM patents.

And then in '88, we started development of ECL SPARC chip with BIT and the high-end server.

Ogus: And then next was SuperSPARC, wasn't it?

Garner: Oh, yes. So, okay. Here we are. So, people were saying, "If SPARC is such a good idea, where is the single-chip implementation?"

Ogus: Because you had two gate arrays, then?

Garner: Yes. We had gate arrays and an external cache, so we're--

Ogus: I guess you probably had a lot of stuff on the board.

Garner: Yes, we had a lot of stuff. So, “Where's the single SPARC chip?” Well, so, they hired the 386 design team managers from Intel: Jim Slager and Jan Prak, who managed the 386 at Intel. Many people worked on it, but they were the lead managers there. This was going to be Sun's first full-custom SPARC chip. Its codename was Viking. SPARC International was formed by that time, so the control of the SPARC architecture was transferred there. Dave Ditzel had come from Bell Labs, and he was Sun's representative there and became really the SPARC cheerleader.

We had an integer multiply and divide at that point, and that became SPARC V8. Texas Instruments was selected as the chip vendor and the Viking team selected their BiCMOS technology -- bipolar transistors intermixed with CMOS transistors. And the pipeline design they were trying.. We, myself and others, thought they were trying to accomplish too much in a single cycle. They had flow-through latches and a very aggressive pipeline, and were having a hard time achieving the initial cycle time goal of 33 megahertz. They essentially created a separate deisgn culture inside Sun that was foreign to the rest of us. The management culture was also alien: closely guarded, circle the wagons, authoritarian. Jim Slager aggressively asserted full ownership of all CMOS design projects. Anant Agarwal, who would--

Ogus: He was working in ECL at that point?

Garner: Yes, ECL, and later spearheaded the entire Sun Semiconductor Division in its future chip successes. There was a serious falling out between Jim and Anant. Jim didn't communicate with Anant in good faith, treated him like a nobody, so animosity also grew between the teams. In the end, Viking ended up being two years late, and it almost sunk Sun because it caused us to fall behind in the RISC system marketplace. Luckily, people were buying Sun not so much for our RISC but also for SunOS, by then called Solaris. Customers really wanted Solaris. It was extremely stable and reliable, the most stable UNIX out there. We used to have contests to see how long you could have a machine run before you had to reboot it. My work station never rebooted in a year, and we had people who ran servers for over a year and a half never being rebooted. SunOS/Solaris was so, so stable.

Viking was two years late. Scott McNealy later felt that it was one of his two big missteps, the other being combining Solaris with AT&T's UNIX. I even considered leaving Sun at that time in the early 90s. Dado Banatao, who had done an Ethernet startup, started S3, wanted to do a chip combining X86 and SPARC. He got a well-known architect from NexGen, Greg Favor. However, I felt that a combined Intel & SPARC chip was a bad idea, and Dado didn't end up doing that and instead did a graphics chip set.

So, there we were with Viking late, but all of a sudden, in 1989 -- In the spring of '89, I was in one of the Sun buildings, and suddenly all these guys from Xerox showed up in the hallways. There was Jean-Marc Frailong and Pradeep Sindhu --

Ogus: Who you had worked with before at PARC?

Garner: -- who I'd worked with at PARC on Dragon! Stunned, I wondered, "What are you guys doing here!?" They had come to the realization that they couldn't get the Dragon project done on their own at PARC. I had left partly for that reason. So they decided to link up with us at Sun and codevelop the Dragon chip set with us. The project was renamed SunDragon. For me, personally, it was like, "Okay, this is kind of cool: Xerox, thank you for adopting SPARC, but now I see that you're going to hold my feet to the fire to finish the Dragon I/O chip that I had left

unfinished at PARC?" So, returning the favor, I joined the SunDragon proejct as manager of the I/O ASIC chip group, my first management position.

So, we did two 150,000-gate arrays with LSI -- that was their high-end gate array. We were responsible for the interface between the DragonBus and the Sbus. With the DragonBus you send out a request for a block for memory and got it back later. Packet-based, very high throughput, but also very high latency. The overall system (codenamed XX) employed two DragonBusses. So my team designed the two chips that interfaced between those two Dragon busses and the SBus, which was Andy's--

Ogus: School bus, yes.

Garner: School bus, which was like the PCIe bus. Intel announced a bus very similar to it called the PCI-e a little bit later. Bill and Andy and I debated how the system board should be configured, and while I wasn't involved in designing the system board, I later acquired one.

This was the massive SPARCCenter-2000 board or SunDragon system. <Showing the SPARCcenter-2000 card, Appendix Photo 1-M> It had the two DragonBus interfaces, and it had slots for four SBus cards. And these were the ASICs that my group did. Well, let's see-- So, it was a multi-layered board. You can see it's stacked, the DRAM DIMMs, and the two Viking processor chips. The principle behind the SPARCCenter-2000 was straightforward performance scaling: as you plugged in up to ten of these boards you would get twenty processors. Each board had memory, processor and IO -- they all scaled together as a unit.

Of course, the problem was that the system was very expensive, and my concern was there were no benchmarks that really showed its worth because, although it had high throughput, the memory latency was so high that individual single-thread performance was poor. Well, it turned out years later-- which I'll talk about the details— the follow-on Sun Enterprise 10000 became the foundation for Sun's business in the dot-com era. It was an excelelnt NFS and Web page server, propelling Sun’s growth during the dot-com era. But at the time, it wasn't clear it was a great idea, but it worked out in the end.

Ogus: Yes, so, that was a multiprocessor. You had told me that the move to multiprocessor wasn't just a smooth thing--

Garner: Yes. No, it wasn't. Bill and I argued against building multiprocessors for a long time because with processor performance doubling every 18 months, it was easier to design the next 2X processor than to design a multiprocessor machine [whose performance gain didn’t scale in proportion to the number of processors]. So why would you waste your time designing a multiprocessor? The rapid speed of evolution of uniprocessors at that time was so fast. It's just hard to imagine. So, finally, though, in one of our lunch debates I remember -- kind of reminded me of Butler Lampson's debates in the PARC cafeteria. Bill Joy would reign over these debates at lunchtime -- we realized that a multiprocessor could save on the cost of power supplies. A multiprocessor could leverage a common power supply instead of separate supplies for each server. Somewhat of an adhoc decision.

Ogus: Funny reason!

Garner: And that opened us up to working with Xerox on the SunDragon. Pradeep Sindhu authored a very intricate asynchronous memory model since on the Dragon bus you got memory back on the DragonBus out of order from issue times. This so-called relaxed memory order became part of SPARC V9, but I'm not sure any applications ever used it. Intel offered a similar relaxed memory model several years later as well.

So, I was saying what happened with this, history-wise. A company called Floating Point Systems (FPS) in Southern California decided to build a four-DragonBus -- or XD bus, as it was called-- system. And they marketed that. Sun had licensed the Dragon bus to them, and they marketed that as part of their own machine. Later, Cray bought Floating Point Systems, and marketed it as the Cray CS-6400. And then Silicon Graphics bought Cray in the mid-90s. So, now, Silicon Graphics featured both MIPS and SPARC-based machines, so they elected to sell their FPS division back to Sun, and that became the Sun Enterprise 10000/Starfire server right at the beginning of the dot-com era, propelling Sun's growth during the dot-com era.

This design that had started at PARC in 1983 or so, went to Sun, then became the Sun SPARC Center 2000, went to Floating Point Systems, to Cray, and then returned to Sun and propelled them during the dot-com era. Pradeep Sindhu -- again, I was disappointed with the long latency in the throughput nature of the design -- but Pradeep Sindhu took the design ideas and became the founding member of Juniper Networks, so he took the ideas of the XD Dragon bus and built switches out of them and became the founder and eventually -- or soon thereafter, the CEO of Juniper. So, that's the SunDragon story.

Garner: So, Anant's system, the ECL system, got canceled in late-1990, and people were-- engineers were kind of discouraged. SPARC had started with great fanfare, we were really high performance. Viking was late. SunDragon has high throughput, but not great for a workstation. People were looking for a silver bullet, so several engineers put forward, "Let's build SPARC – a scalable architecture afterall – in a new technology. Let's build a SPARC chip in gallium arsenide" -- what Cray had attempted with his Cray-4. So, we we formed a team of about 20 or 30 engineers and linked up with Vitesse Semiconductor as our gallium arsenide chip vendor. Anant asked me to manage the team to design the gallium arsenide SPARC processor with 200 megahertz as our clock goal.

Ogus: So, that was the Brute Project?

Garner: Yes. That was called Brute, and from September to 1990 to the June of 1991, I worked on this. Every Monday, I would go into work thinking: "This is going to be so cool. We're going to build a 200-megahertz gallium arsenide chip," but then by the end of the week there'd be so many problems. So, it had to be multiple chips: cache, integer units, etc.. So, we needed a multi-chip module -- made it expensive right there. Gallium arsenide has asymmetric transistors, it turns out: P-channel devices are much slower than N-channel devices. There's also no intrinsic SRAM memory cell, so we needed to interface with ECL memory chips. So, the number of problems and the challenges just kept rising and rising, and the power dissipation estimate got hotter and hotter, all along while Viking's late delivery was impacting Sun's revenue and prestige.

Something had to give, so Bill and Andy stepped in and said, "This is crazy." So, they redirected the gallium arsenide team, Anant's ECL team, basically, and said, "You guys are going to do the next CMOS processor, and it's going to run in parallel with Viking." So, there would be two leapfrogging microprocessor design programs-- like what Intel did.

Intel has a design team up north in Hillsboro and they have another design team down here in Santa Clara, leapfrogging each other. Jim was not happy about that, but hey, that's the way forward.

But Anant was tickled pink. We get to do a real, full-custom CMOS design project. But now the question is: “You're about a year and a half to two years behind the market. How do you attract a design team when you're that far behind?” Anant asked me to be the second line manager for what became known as UltraSPARC, a 64-bit SPARC V9 microprocessor, following the specs of SPARC International. I guess I should comment on the 32 to 64 bit transision.

It was pretty easy to extend SPARC from a 32-bit architecture to 64 bits because I had formulated such a simple instruction set. We basically just simply extended memory and registers to 64 bits wide. Bill Joy came in and realized that there should be a whole separate set of registers for the operating system separate from the register windows. So, he proposed three separate sets of trap registers, so even the kernel could easily handle multiple exceptions within itself. He would reign over SPARC International meetings over the phone because he had such an incredibly strong persona.

So, we began the UltraSPARC project in a hiring mode. We needed to hire over 120 people quickly, so we were interviewing over a dozen people per week. We held joint hiring meetings on every Friday on how to best build out the team. And we did find people who wanted to help save the ship, to help pull Sun up from behind.

A key hire was our lead architect Les Cohen, who came from Intel where he had designed the Intel i860. I was responsible for the front-end design team, the architecture, logic design and verification. My team was divided into the pre-fetch and instruction dispatch unit, the Integer Unit, the Floating Point Graphics Unit, the Load-Store unit, and the External Cache Unit. We were responsible for the architecture, performance analysis, simulation, compilers, specs, logic design, diagnostics, circuit and full-chip static timing analysis, and hardware emulation. I grew the team from 12 to 80 engineers and had six first-level managers reporting to me.

We designed a 16-kilobyte on-chip instruction cache; a 16-kilobyte data cache (both physically addressed); and a large 128-kilobyte external cache. And again, this allowed us to have full control over the system interfaces, designing the UltraSPARC Port Architecture (UPA), packet switched, nonblocking interconnect -- so that we could just connect multiple UltraSPARC chips together to form a multiprocessor --16-bytes wide, 600 megabytes per second bandwidth.

One thing that happened, there was a bitter residual feeling about the cancellation of the gallium arsenide SPARC, and the team wasn't so sure that going off and doing a CMOS chip would get us back into a leadership position. An example of that was what to name the project. What you call a project is important. One of the proposed names was Bob, which was clearly a reference to you know who. And the other proposed name was Spitfire that Nigel Ross from Great Britain had proposed, the famous fighting machine during World War II. When I stepped into this meeting the team was stalemated: equallly divided between both Bob and Spitfire. I cast the deciding vote. That just shows you how disillusioned some of the team members were.

Ogus: For Spitfire?

Garner: For Spitfire, Yes. That was a good symbol for the project. It was a fighter. So, we hired Les Cohen, I mentioned, and he had already, for the Intel i860, had introduced some graphics instructions, so for us he designed SPARC's multimedia V-I-S, VIS instruction set, to speed up media applications. We were the first microprocessor of note to actually come out with an extended instruction set for multimedia, and Intel did later follow the example.

And while we were struggling to ramp up, right in the middle of that, I was shocked to see that Fujitsu was funding a startup to do a SPARC microprocessor. I went, "Holy shit. We could hardly get funding and enough people to save SPARC, and here's Fujitsu starting a company to do their own SPARC chip to out-compete us?"

Ogus: Was it literally a licensed SPARC chip?

Garner: It was a licensed -- Yes, we licensed the technology, so they got a license, and the startup was named HaL. The name was kind of odd. It was reported to be named after Andy Heller, who was the founder at IBM who pushed the RS/6000 through IBM, and Bernie Lacroute: "Heller and Lacroute." But Bernie decided not to join afterall, so the L ended up being spurious. HaL's objective was to outdo both IBM and Sun and build workstations and servers to outperform both. Fujitsu footed the bill to the tune of hundreds of millions of dollars.

Luckily for us, HaL succumbed to the second system syndrome. They designed a chip set that was too complex, too expensive, and too much hair. But I got to know Winfried Wilcke, who was head of engineering there, on that project, with whom I later linked up with. And Bob Montoya was a brilliant IBMer who was known for his screaming floating point circuits. Actually, Bob first introduced me to HaL at a Hot Chips conference. He walked up to me with a pencil that said IBM and SPARC on it and proclaimed, "These are the two companies we're going to kill!" <laughter>

Ogus: Oh, that we're going to kill!

Garner: By using SPARC. So… <sighs> back to UltraSPARC, what's the best way to develop a whole new pipeline from scratch? So you have these alternatives of doing a very complex pipeline that tries to execute a lot of instructions in parallel, versus a deep, high-clock rate pipeline with very simple controls and less concurreny. DEC was an example of the latter and other firms were an example of more complex approaches. So we built a microarchitectural pipeline simulator called Shade so we could test the efficacy of various pipeline tradeoffs and just simulate them, running applications. We would run traces of the SPECInt bechmarks through the simulator to find out how they were doing and noticed that out-of-order execution didn't by us very much. We estimated that it only bought us about a 30% performance gain. Well, if performance is doubling every 18 months -- that's equivalent to about 4% per month or 1% per week [=2^(1/78 wks)]. So if you're only going to get 30% and it's obviously going to take more than two months to get that implemented, so nah, we shouldn't bother. My rule of thumb was if an architect had a proposal for a new pipeline feature that could be shown to gain 1% performance, he or she had to prove to me that it could defined, modeled, implemented by the design team and debugged with diagnostics in under a week. Since that didn't happen--

Ogus: That was the criteria--

Garner: That was the criteria.

Ogus: Moving forward--

Garner: For moving forward, any new architecture proposal. Right?

Ogus: Okay.

Garner: And so, then, you could really cut out a lot of odd proposals that way and remain focused on a simple pipeline and faster clock speeds with process technology, on which Ron Melanson pushed our CMOS vendor Texas Instruments on. This was before TI became good at designing fast Digital Signal Processing (DSP) chips. We basically pushed their technology to the its limits. Ron kept up the pressure: “We need a faster transistor, we need a faster SRAM cell.” And that way, [particularly by insisting on scalable circuit layout design rules], if our design schedule slipped, we could compensate by pushing TI a little harder to speed up the clock rate.

The analogy I like to make for designing a large microprocessor is that it's like designing a large city. Let's take, say, San Jose. So imagine, Roy, you're told to design San Jose from the ground up. And really, I think the size of the effort is comparable. The success criteria is that everyone has to get from all their homes to their offices within say one hour. So you have to design all of the highways, freeways, buildings, offices – everything -- to move everyone from home to office within one hour, equivalent to one processor clock tick. If anyone is late, the design doesn't work.

So you also have to get a team of 120 - 130 people to all work together on this common goal of meeting the cycle time plus with correct functionality. So the cycle time, where it trades off against functionality, you make enhancements to improve the cycle time, which then makes the design more complex, which introduces bugs. And once you launch the chip at tape out, it's like a rocket. You can't really change it afterwards.

So we developed-- Guillermo Maturana decided to develop a cycle-accurate simulator in C++, called INCAS which we ran against the netlist, which was 300K-lines of Verilog. INCAS became the de facto spec, versus the actual written spec for UltraSPARC at 800 pages of English. You could misinterpret, perhaps, what the written spec is trying to say. So we decided that the cycle-accurate simulator was the most accurate spec. If the simulation of traces against the Verilog ever miscompared, we could ask, well, “Is it a problem in the simulator, the Enblish spec, or a bug in the design or implementation? And most of the time, it was a bug in the design. So we had this sanity check.

And because INCAS was cycle accurate, we could detect timing bugs in the design. So for instance, the UltraSPARC’s TLB had branch prediction bits to speed up branching. Well, if they’re incorrect, you'll just take more time to do the branch. But we realized through a timing simulation miscompare that one of the benchmarks was suddendly running 30% slower than before because a bug had been introduced into the branch mis-prediction logic. The whole chip could have run 30 percent slower if we hadn't had the cycle-accurate simulator. So that was--

Ogus: That was really useful?

Garner: INCAS ran 6,000 instructions per second, 600 times faster than the Verilog. So you could run a lot of cycles. Guillermo found 65 bugs using INCAS. Oh, and we actually booted Solaris and Open Windows on the cycle-accurate simulator over 5 billion cycles and found 65 bugs: 30 functional, and 35 performance-related bugs. We also got SPEC92 estimates by sampling a third of a percent of the traces. So we actually knew our SPEC numbers before the chip taped out, which is kind of stressful because you're getting wind of figures from other companies at the same time.

My main soapbox point was that the diagnostic team was just as important as the architecture and RTL design teams. They wrote 1,500 diagnostic tests, 5 million diagnostic source lines. Each diagnostic, on average, uncovered one bug in the design. So I maintained a chart that showed the number of diagnostic tests planned, the number implemented, and I expected that each one would uncover about one bug. So, I keep pushing to get more diags written. If you just stopped writing diags, you'd likely fool yourself and tapeout too soon. Following this methodolgy, I could reasonably predict when tape-out should be; that is, when we finished all the diagnostic tests we could think of, the last bug would be caught right before tape-out. Later, someone told me that, at Microsoft, they would send all the software diagnostic testers to a movie two days before a release so they won't report any more bugs at the last minute. <laughs>

The diagnostic team also wrote a 100,000-line pseudo-random diagnostic test generator. It produced 1 billion simulation cycles, just random collections of instructions, and then, we'd compare the results against the cycle-accurate simulator to spot any bugs. Oh, we also simulated up to four UltraSPARCs running concurrently so that we knew the MP stuff would work right out the chute.

Ogus: This is all using the cycle-accurate simulator?

Garner: Yes. We also had a huge simulation server farm. We had a farm that numbered 225+ SPARCstations, or 500 processors total. Two terabytes of storage. We had our own resource allocation manager called DReAM [Distributed Resources Allocation Manager], which had policy rules for running simulations and diagnostics. We had to develop 100,000-line CAD flows combing great tools from different EDA vendors: Mentor Graphics, Synopsys, Cadence, Chronologic, Parsec, Metasoftware, and in-house development of a Central Database Managemen System (CDMS). We had 150 concurrent users of the server farm. We also had to do design for manufacturability, which was also part of the gate array. How do you implement serial test in the chip to make sure the chips basically work when they come back?

And one of the disciplined things we did is we had a synchronous or a lock-step design release cycle, where every week, we integrated the entire netlist, ran the diagnostics, and reported the bugs to the team on Friday night. So on Saturday, they knew what needed fixing. Some folks didn't go home on Saturdays or Sundays. So we integrated on Friday, ran the diagnostics overnight, the failing units were reported on Saturday, and hopefully, by Monday or Tuesday, the bugs were fixed.

Ogus: Fixed, yes. 24/7.

Garner: It was a continuous staggered release cycle. Now, the other thing we did is we did hardware emulation of the netlist. We got -- let's see what was that company -- it'll come to me in a minute [Quickturn Design Systems]. We did ten complete integrations and virtual tape-outs for the emulator before tapeout. We also employed static timing analysis tools: We had Pearl, which was a Parsec Software tool; RC delays were a big deal, Piyush Patel ran that. Initially, 8 percent of the paths were 2X over timing target. So imagine you're actually running at half speed initially! So you can imagine all the work to fix the timing. My analogy was like cramming steel wool into a box: You push it down here and some of it pops up over there. So you'd start to worry whether the box is too small, or that there's too much steel wool. You just don't know which when you're designing the chip.

Oh, Quickturn. Quickturn was the name of the hardware emulator firm. So the CAD team had to take the Verilog netlist, convert it into the software for the Quickturn emulator, which mapped it into PGAs, which then ran the netlist at hardware emulation speed. So that process forced a complete chip release cycle. Some folks kind of hated the process but the need to do it perfectly fostered a lot of discipline. And the Quickturn emulation did find one bug.

Ogus: Well, that's fine --

Garner: Yes. We had that one bug in the gate netlist. So there were two sets of teams. I had the front-end team and then, there was a circuit design team, circuits and libraries, and Hem Hingar from National Semiconductor ran that. This was like two engineering cultures, where Hem's group had all the nose-to-the-grindstone circuit designers and we were the imagined frou-frou architects. So Hem and I -- our job was to try to get these two teams to work together as one.

This is where I learned that just listening to people was really important. There were a lot of arguments between people and personality conflicts and disagreements about the approaches we were taking. So we had gripe sessions every Friday. You'd get 20 or 30 people in a room and we'd let them ask us anything they had on their mind: What were they unhappy about? Me, Anant, Hem, people, schedule? And we would break down the main points and we'd break down the recommended changes. And we couldn't always implement the changes people wanted, but just the fact that we listened to them really made a difference. People would say, "Well, you guys really-- as a management team, you care, because you're listening to us." And that's one thing we learned. We also provided free meals -- this was a common phenomenon that the chip companies did at the time. Anant ruled, though, you had to work at least an hour after the free meal in order to partake.

So in Hem's group, Lavi Lev, Andy Charnas led a disciplined circuit design methodology team. Lavi had a 200-entry checklist for the circuit designers, 200 items he had to sign off on. Two hundred per circuit block. There were typically four design reviews per block, 13 simulation corners, and HSPICE verification. He drove for less than 5% clock skew across the chip. Power grid analysis: They placed a 100-nanofarad capacitor on the chip. Ron Melanson established a co-management team with Texas Instruments on the process and technology rules and, as I said, pushed the design rules to meet our faster clock rate needs.

I mean, it was truly a lot of work, to get 150 people to all work together to get something of this magnitude to work flawlessly. [Appendix Photo 1-J shows the UltraSPARC design team] Intel had even bigger chip design teams. I heard that they gave each young engineer a little piece of the chip and said, "You're responsible for these several square microns." And they'd burn out after two years.

So we had this tension between what we called "tall thin designers," expertise in logic, circuit and CAD, versus non-siloed "horizontal engineers." We tried to break down those stereotypes. That tension was there. It started with Mead-Conway, because I don't think Lynn Conway really understand how hard it is to design circuits that truly work across all process corners and meet the speed goals. And so, via the CAD tool flows and our various backgrounds, we were attempting to combine the two approaches together as a new team. It was very difficult.

On the schedule: Our first chip integration was in 1993, 1st silicon/chips was September '94. Unix booted the same month. First silicon came back and Unix booted, because we had simulated it all. The UltraSPARC based Ultra-1 workstation First Customer Ship (FCS) was one year later, in November '95.

The UltraSPAC-1 chip, I should have it here somewhere. The chip, we were all given these. So now, you can hold 5.3 million transistors in your pocket, and I think I have one in my pocket. Yea, 5.3 million transistors using TI's half micron process. So what used to fit on the board, it now fits in your pocket! Of course we're now up to a billion transistors per chip, but the die sizes have remained about the same. So you can see all the parts <Showing an UltraSPARC-I chip, Appendix Photo 1-N>: the caches, four-way super scaler ALU, nine functional units, half-micron technology, four metal layers, 5.4 million transistors, 167-MHz. 1st-silicon September 1994. We were all very proud of this chip!

Ogus: yes. Because--

Garner: We had gone from gate arrays to the late full-custom chip project (SuperSPARC) to a successful full-custom chip project (UltraSPARC), which put Sun back on par with the other RISC vendors at that point. And we had systems that people wanted, that ran Solaris. There were 300,000 lines of Verilog. So now, you've kind of lost touch with gate design, because you just write Verilog statements and all the gate design is done by the Verilog compiler. Five million lines of diagnostic code written. Two billion simulation cycles. We estimated it cost $60 million and 300-person years to do the chip. And the initial chip had no circuit bugs and it met the speed goals! So we were really very proud of UltraSPARC.

And so, in December '94, our MarCom person, Marge Breya, organized a public event called the "The Making of *Ultra*SPARC" for press analysts and industry watchers. One thing that I noticed was that if you look at-- if you looked at this chart-- <Showing a chart of Peak MIPS vs year> I did this internally, I guess, in December '94, to start with. If you did a plot, the SPECint rate and the year of various chips. No, this graph was for the Making of *Ultra*SPARC, the public talk. You had a 6-MIPS chip and a 10-MIPS -- the first SPARC chip and then, 22 MIPS and you drew a straight line rising at 55% average growth per year. And a year later, in October 1995, here I filled in the names of the systems that showed performance rising at 60% per year <Showing a chart of Peak MIPS vs year, Appendix Chart 1-P>: MIPS M/500, Sun-4/200, Sun-4/470, IBM RS/6000, SGI Crimson R4000, HP 9000 PA, DEC 10000 Alpha, IBM 9000 Power2, DEC 3000 21064A, DEC 250 21164, and DEC 5/300 21164 [with clock frequencies increasing from 8 MHz to 300 Mhz and IPCs varying from 0.6 to 1.8 instructions executed per cycle].

The SPECInt92 line can be seen rising at 60% per year and the question was: "Would it keep doing that?" Does the performance explosion go on forever? What I noticed was that there was no single technique underlying this growth rate: It wasn't just pure clock rate and it wasn't pure pipeline design. Breaking down the 60% per year speedup: 21% was due to improvements in the microarchitecture and compilers; 32% was due to faster clock rate – of that, 20% was faster process and 10% was faster circuits. Most folks thought that the annual 60% speedup was due just to Moore's Law. Well it's *not* just due to Moore's Law. Moore's Law stated that the number of transistors double on a chip every two years. However, that phenomenon just gives you a raw clock rate increase of ~32% per year – correpsonding to linear gate shrinkage from the doubling of transistor density every two years -- the square root of two [=1.41] every two years, or corresponding to ~1.32x linear shrinkage every year [=1.41^0.75]. To stay on this 60% per year growth line you also have to improve your pipeline and improve your compilers, or you fall off the line.

So the question was: “How, looking forward from 1996 to stay on the 60% growth line?” Would your instructions per cycle (IPC) need to go from 1.2 instructions per cycle in 1996 to 2.6 by the year 2002, to 12 instructions per cycle by the year 2008, to stay on the 60% annual performance growth line? <Appendix Chart 1-Q shows my Projected Future RISC Performance>

Ogus: But are those actual numbers?

Garner: No. These are projected futures. This was done in 1995. I was saying that to stay on the line--

Ogus: Yes. That's what you had to do?

Garner: You had to have an incredible instruction level parallelism and you had to have a 12-gigahertz part by 2007. And that didn't seem likely to me. So the next year, 1996, at a VLSI Technology Seminar in Hawaii, I did more detailed mapping of CPI and SPECint. <Appendix Chart 1-R shows peak SPECInt and clock frequency vs year> This graph also shows the SPECint per MHz. It varies a lot, but the overall performance sticks to the 60% growth line. So there's no magic recipe between clock rate and pipeline complexity and compilers. Everyone seems to be stuck on this line. It's not Moore's Law. Maybe it's Gordon Bell's Law or Bill Joy's Law or my Law. Floating point performance also had similar characteristics. <Appendix Chart 1-S shows peak SPECfp and clock freuency vs year>

So I projected in 1996 at the Hawaii conference how processor performance scaling would likely slow down, because I felt that it couldn't continue rising at the same 60% per year rate. <Showing my projected future RISC performance, Appendix Chart 1-T> If the clock rate were to continue increasing at its historical Moore's Law rate [32% per year], starting with 4 gigahertz in 2005, by 2010 the CPU clock would need to be 16 gigahertz [=4*1.32^5]. Today, in 2018, it would need to be something like 150 gigahertz! [=4*1.32^13 yrs]. So where's our 150 gigahertz uniprocessor? That hasn't happened.

So I predicted that the processor clock rate would likely slow and only hit 6 gigahertz in 2010 (instead of 150 gigahertz). And that's about what happened. Someone got a five or six-gigahertz clock, but really nothing faster since then. And the SPEC performance number, I said, would level off. It would flatten out, and that's what it's done. I've gone back and looked at these figures. [Also at the 1995 Hawaiian seminar I predicted that most transistors in future chips would be in memory and/or graphics computational arrays, which did occur with the rise of GPUs. However, I did *not* anticipate the rise of multi-core processors.]

The other thing that happened is that Intel began to catch up with RISC performance. In 1996, a 200-megahertz Pentium-I was gaining on our 200-megahertz UltraSPARC-I: the 200-megahertz Pentium was 5 SPECInt95 and UltraSPARC-I was 6.9. But by the end of '97, a 300-megahertz Pentium-II eclipsed our 300-megahertz UltraSPARC-II. It benchmarked at 11.7 SPECInt95 and SPARC was 10.5. So it was clear there was no fundamental intrinsic magic behind RISC. You could apply RISC ideas to CISC implementations, improve your compiler, perhaps do register renaming to effectively get more registers in hardware, do good branch prediction, and do hard-wired control with no microcode in sight. That's a key point, no microcode.

So the talk I gave at San Jose State in 2011 … at the time this was the only fundamental difference between CISC and RISC: CISC was generally microcoded and RISC is hard-wired control logic -- with lots of registers (or register renaming and good compilers. I think Dave Patterson has come around to this too.

Read-only-memory(ROM)-based microcode began with Maurice Wilkes in 1951 with the EDSAC. The MIT Whirlwind in 1947 had a regular diode array for control. You would have two matrices of diodes that asserted control signals at the proper times. [Chuck Thacker and the Berkeley BCC-500 designers adopted this approach.] Wilkes himself was against using SRAM for this – preferring that control circuits be a diode array or ROM, primarily to simplify the design process. What happened later is [that designers moved to more alterable ROM, such as CROS and TROS units in the S/360s and then to SRAM-based control store with its vast flexbility, allowing one design to support multiple instruction sets, as in the Alto, or add customer specific instructions, fix bugs in the field, etc..]

IBM's Mythical-Man-Month author Fred Brooks, strongly advocated for microcode store in the S/360 family. IBM had architected the 360 line to span a 10X performance range, which was great, but most of it was shifted down in performance. IBM didn't mention that. Speaking with Fred Brooks several times over the last several years, he related how he had instructed IBM's 360 model designers that if they could prove that a hard-wired control design was 30% faster than a microcoded design, they could go hard-wired. But the microcoded designs worked out to about the same performance as hard-wired control because performance was broadly limited by main memory speed since there were no memory caches. Later, high-end IBM modes 85 and 195 had caches. The model 195 had both hard-wired control and a fast cache. But the microcoded S/360s were limited by main memory, so it was advantagous to have microcode there because you were fundamentally limited by main memory speed. As soon as less costly memory caches came along, essentially taking the place of writable microstore, voilà, microcode became a hindrence to higher performance .

And so, if I may show the-- my Dandelion CPU card, the Star card. This is the CPU card for the Dandelion, the Xerox Star. <Showing the Dandelion/Xerox Star processor card, Appendix Photo 1-D> So here are the four AMD 2901 bit slices, and there's the 4K entry, 48-bit-wide control store. We saw the Sun board, right? <Showing the Sun-4/200 CPU card, Appendix Photo 1-H> It has the Fujitsu gate array instead of four 2901s and it's got a large single-cycle cache. The Dandelion CPU had a 137 nansecond cycle time, a 7.3-megahertz clock rate, which is not too bad. Five years later, the first SPARC chip ran at 16.7 megahertz.

Thus, broadly speaking, all we did in CPU evolution from Xerox to Sun was to take the Dandelion's writable microstore SRAM chips and turn them into a main memory cache. That is, to convert from CISC to RISC, just convert the microcode store into a memory cache! <laughs>

Ogus: Yes. That's a good way--

Garner: That's a gross difference between CISC and RISC -- that's it! Obviously, RISC instructions are simpler -- you don't need really wide microinstructions. Also, RISC greatly benefits from improved compilers.

Oh, by the way, I hadn't shown this. This is the Ethernet card for the Dandelion: <Showing the Dandelion/Xerox Star Ethernet card, Appendix Photo 1-D> These are the chips here to implement the small FIFO. And this is Ron Crane's analog phase-lock loop section right there with the 20-megahertz crystal oscillator that is used to Manchester encode

the bits on the wire and is then divided by two for everything else. The lower half of the board, not shown, was the low-speed laser printer interface (LSEP) -- I can't remember who did that [Pitts Jarvis and Ken Yamanaka].

Ogus: Yes. I don't remember.

Garner: Yes. But so, this explanation for RISC versus CISC, in my mind, it makes a lot of sense. And I started to think about these higher-level . Also, I got more involved with the Computer History Museum in Mountain View.

One thing that happened is in 1998-- we presented the Dandelion hardware, the final demo of the Xerox Star at Xerox PARC. I don't know if you remember that, at PARC. At about that time, I had heard that Dave Patterson had consulted with IBM in some way. As a young kid, I had always assumed that brilliant people got ideas that fell right out of the sky into their heads. But David had interacted with IBMers, so I got up the nerve in 1998 to call John Cocke, the architect of the IBM 801. I reached him at his home and I said: "John, I understand that Dave Patterson may have somehow consulted with IBM." He said, "Yes. I came out to California, and I met with Dave Patterson, Dave Ditzel, and George Taylor every week at the local pizza restaurant in Berkeley, and I just told them *everything!*" <laughs> I said, "That's interesting." So I called Dave Patterson. I said, "Dave, I just talked to John. He told me this." He goes, "Yeah, yeah, yeah."

And then, a little bit later-- because Dave had been closed-mouth about it before -- and later, in 2003, the CHM did a “25 years of Hennessey and Patterson” retrospective about Dave Patterson and John Hennessey, when John was then head of Stanford University. Dave jokingly confirmed that John Cocke had “met with him and told him everything.” John Hennessey’s reaction was, "Well, not *quite* everything!" <laughs> [since at Stanford Hennessey’s students had also contribited to improving compiler technology.]

What John Cocke realized back in the early 1980s was that IBM was not going to bring RISC to market, because it threatened their mainframes too much. And the best way for-- because John was in research, to bring it to the world would be to go out of IBM, go tell everybody about it, which then might inspire everyone else to do one, which then might pressure IBM to get its act together and do one itself. Andy Heller initially drove the RS/6000 efforts inside IBM and did a good job. IBMs various RISC implementations got better and better and better. So, John’s unusual approach payed out! I did an oral history with the 801 team two months ago and its project manager [Frank Carrubba] confirmed this story.

Ogus: Confirmed that, yes.

Garner: Yes. By the way, one comment, again, on the S/360 was from Fred Brooks. I told you he said he gave design teams the option to design a machine either way, hard-wired or control store. I learned that, the Model 60, aka the NPL 400, was actually first built as a hard-wired machine -- the machine Bob Evans was standing in front of when IBM announced the S/360 family in 1964. But it shipped as a microcoded machine. So IBM actually changed it from a hard-wired machine that was working to a microcoded machine. That makes it the only computer I know of, built in the same technology at the same time and designed both ways. I was able to figure out who the manager of the project was and I eventually was able to find him and contact him. He was Stu Tucker. He confirmed that the performance of the two models was identical, which makes sense because they were main memory limited. So that's the only computer I know of that was actually implemented both ways, microcoded and hard-wired control.

So my last comments on SPARC—oh, one thing that happened is in November of 1996, COMDEX held an event, the “25 Years of Industry Achievement” in Las Vegas, and the committee selected seven microprocessors for being, quote, "Seminal in nature and incredibly innovative for their time." And the ones they nominated were the MOS Technology 6502, the Zilog Z80, the Intel 8088, the Motorola 68,000, the Intel 386, Sun SPARC, and Intel Pentium.

Bill Joy didn't want to attend the ceremony, so I accepted the award for SPARC. I was a little intimidated standing alongside luminaries like Federico Faggin, Andy Grove, Masatoshi Shima, John Crawford and others. But it was a fun event. I was the only person-- I was only one of two people who didn't wear a tuxedo. <laughs> So I kind of stand out in the picture. I figured we at Sun were a rebel-rousers, so why would I wear a Tuxedo? But anyways. It was a fun event in Las Vegas, complete with go-go dancers at the end. <laughs>

Ogus: So, let's see. Where are we in terms of where you were at Sun? You were-- was that close to the point when you decided to leave Sun?

Garner: Not quite. I was on the program committee for the IEEE Hot Chips Conference. So we, Sun w/UltraSPARC, had caught back up on performance, as good as anybody. But what we were going to do next to move the needle? So I organized an off-site workshop at Asilomar Hotel and Conference Grounds, where we had Bill Joy and Michael Deering, who was the leadering graphics Distinguished Engineer at Sun. We wanted to figure out a way to give Sun a boost in hardware. And we came up with the idea of incorporating the graphics pipeline in the CPU chip. So, we went off and persued that. And we concluded that, since Java was here and features just-in-time (JIT) compilers that are able to compile into any instruction set, let's devise an entirely whole new microprocessor instruction set! It doesn't have to be SPARC-compatible. Let's be crazy and bold again.

So, we started-- after two days of discussions and walks on the beach, we decided on several new ideas: a new multi-core, multit-hreaded, simplified VLIW, speculative-execution architecture that would run just-in-time compilation of Java code. I wasn't the architect, but I was asked to be the director of the project. Andre Kowalczyk was the second-level manager. The team grew to 50 engineers and four managers doing this whole new chip. Its architect was Marc Tremblay.

It was called UltraJava internally, later renamed to MAJC, M-A-J-C. It was the first chip produced that really had a multi-core, multi-threaded architecture, so it could run-- the idea was if you're going off chip to cache – it’s now a second-level cache, far away at these high clock rates -- let's run something else under the external cache access. Thus multit-hreaded and multi-core.

After more careful analysis, the we couldn't find enough room for the graphics triangle rendering engine on the chip, so we kicked that out and just stuck with the front-end processing for a graphics pipeline. And then, I tried to find out who could use it. Well, Michael could-- Deering could use it in the graphics group. We visited Sega in Japan and several game console companies. We said: “We got this hot new high-performance triangle processing graphics chip. “You program it in Java. Do you want it?” They just looked at us and smiled. <laughs> Because their experience told them that Java wouldn't be a great environment for developing console gaming software. Nevertheless, the MAJC chip was designed, taped out, and worked. It was used in two of Sun's XVR workstation graphic accelerators, but never went anywhere else after that.

Ogus: So it actually made it into a product?

Garner: Yes. It actually made it into a product, but it was a big cost for not--

Ogus: Okay. It didn't go further. Uh-huh.

Garner: So after MAJC started, while MAJC was going on, actually, I was in a executive meeting, and Ed Zander, who was head of Sun at the time, started thinking how he'd capitalize on Java more. Bill asked me, "How big would the Xerox Mesa processor chip be if implemented in silicon?" I said, "Well, about two microns on a side," and he immediately raised his hand and proclaimed: "We're going to implement Java directly in silicon!" <laughs> And that's how the picoJava project got started.

The marketing crew made a press release by Monday on how small and beautiful picoJava would be and I thought, "We haven't even hired a design team!" So other people put together the team. I didn't like the idea because Java was meant to run on any instruction-set architecture, not just a tailored chip. The picoJava effort was derided by the analysts for having taken this approach, as I was even able to haul Bill Joy up in front of them once. I actually gave the presentation at the 1997 Microprocessor Forum on picoJava, just because I wanted to have the experience of speaking there.

And then, while MAJC was in mid-stride, Bill and Andy asked me to head a Java software project to give another boost to Java, called Jini. Bill had this kind of crazy dream-like idea that computers could be magically brought into being by having components finding each other on the network and exchanging code for what their interfaces needed to be, using Java's Remote Method Invocation (RMI) interface.

So device drivers-- today, in an operating system, you have a device driver for every type of I/O device you might ever connect and there has to be rigid interfaces-- these are the bits to pass between applications and the I/O device. His idea was that the environment sends the printing code to the computer. You don't have a device driver at the low-level interface. You just send code around, because Java can send code around.

Well, the defect with that vision is that eventually the code has to talk to hardware. So eventually, that code has to understand the hardware. But we had a big dazzling project. We built imitation Jini devices running diskless Suns and keyboards and monitors. Later I found that someone actually built a Jini device at IBM Almaden Research Center. But Jini was mainly hubris.

So during that time, Intel was catching up with SPARC in performance and we at Sun were getting eaten from below-- getting eaten badly from below by Intel and with Intel compatibles such as AMD. And firms were designing competitive multiprocessor systems. I decided it was time to leave Sun. So I left in 1998.

Ogus: 1998? Okay.

Garner: And that's where I think we can leave things for now. Again, I'd like to say that one of the main-- oh, and then, SPARC eventually got an IEEE Milestone Award, in the year 2015, sponsored by the guys at Oracle [Uday Kapoor]. I wrote the text for the plaque. That was kind of nice.

I don't know if I showed all my pictures. Let's see. So the original SPARC team-- oh, and I didn't show this picture. So this was the original Dandelion hardware design team: <Showing photo of Dandelion/Xerox Star hardware design team, Appendix Photo 1-E>

Ogus: Yes. <chuckles>

Garner: So that was the picture that Bob Belleville -- I guess he took it with a delayed shutter. That's me. That's Ron Crane. There's you, Roy. That was the guy from [England] --

Ogus: Dick Snow.

Garner: Dick Snow. There's Don Charnley.

Ogus: Don Charnley.

Garner: Jim Cucinitti and--

Ogus: Neil Hansen.

Garner: Neil Hansen with the two--

Ogus: And Nora Ogawa.

Garner: --technicians and Nora--

Ogus: Liddle.

Garner: Ogawa.

Ogus: Ogawa was her maiden name.

Garner: Yes. So that was our whole team.

Ogus: That's right. In 1979.

Garner: Yes. And then, this was the SPARC architecture team standing in and around Bill Joy's Ferrari. <Showing photo of Sun SPARC architecture team, Appendix Photo 1-J> So there's me, Dave Patterson, and Steve Muchnick (compiler group manager), Anant Agrawal. Will Brown (simulator), Faye Briggs (suggested SWAP instruction), Joan Pendleton, Don Jackson (floating point controller chp), Dave Goldberg helped me in the OS group, Sunil Joshi did the Integer Unit chip with Anant, Richard Tuck in the compiler group, and-- I forgot his name [Alex Wu, in the compiler group]. Dave Weaver (in the compiler group) eventually took over the SPARC architecture manual from me; and Wayne Rosing, who became the manager of the SPARC effort and then all of hardware engineering. Wayne, a great manager to work for. He had managed the Lisa program at Apple before he came to Sun and was at DEC before.

Ogus: That's right. I remember that.

Garner: So I worked for two managers at Sun who afterwards managed the Lisa program and the MacIntosh programs. Wayne was a great manager. He did quite well at Sun. Retired and then went to Google and did quite well at Google.

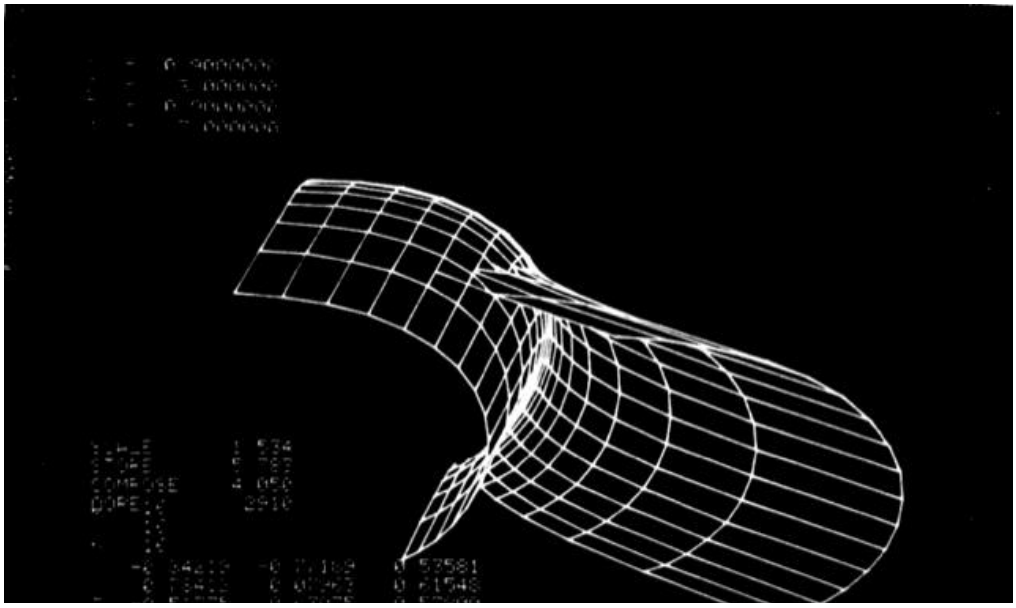
And next, if you include the rest of the SPARC team, you can see the whole team in this picture. <Showing photo of Sun SPARC architecture, hardware, and software team, Appendix Photo 1-K> The faces are a little small. It was taken later. This includes the rest of the software people. Still, a pretty small group.

And these are the *Fortune* Magazine awards: <Appendix Photo 1-U is photo of Fortune Magazine's 1981 Product of the Year Award for the Xerox Star 8010 Workstation and Appendix Photo 1-V shows 1987 award for the Sun-4/200>

END OF THE INTERVIEW

Appendix: Photos, Images and Graphs

Image 1-A: Example images from 3D surface viewing program (SURV), computed on a PDP-10 and rendered on Tektronix 4010, Dr. Greg Nielson, Arizona State University, Mathematics Dept, using author's hidden line removal algorithm, 1975.



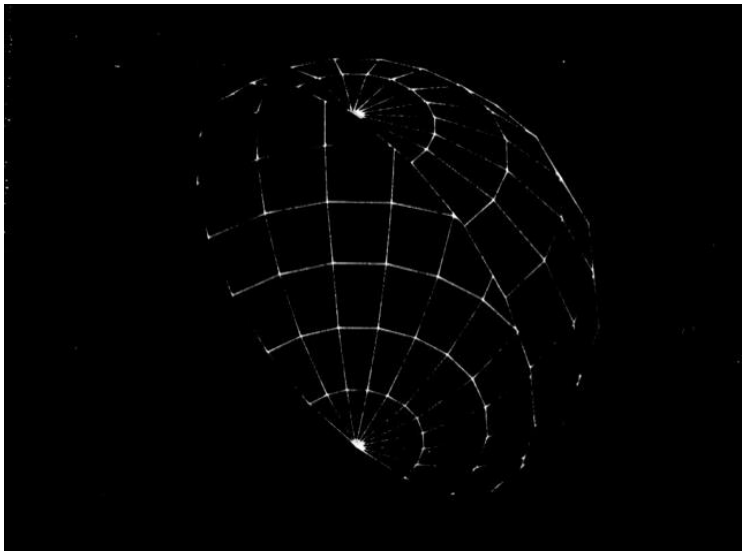


Image 1-B: Xerox System Development Department (SDD) Org Chart, 1978

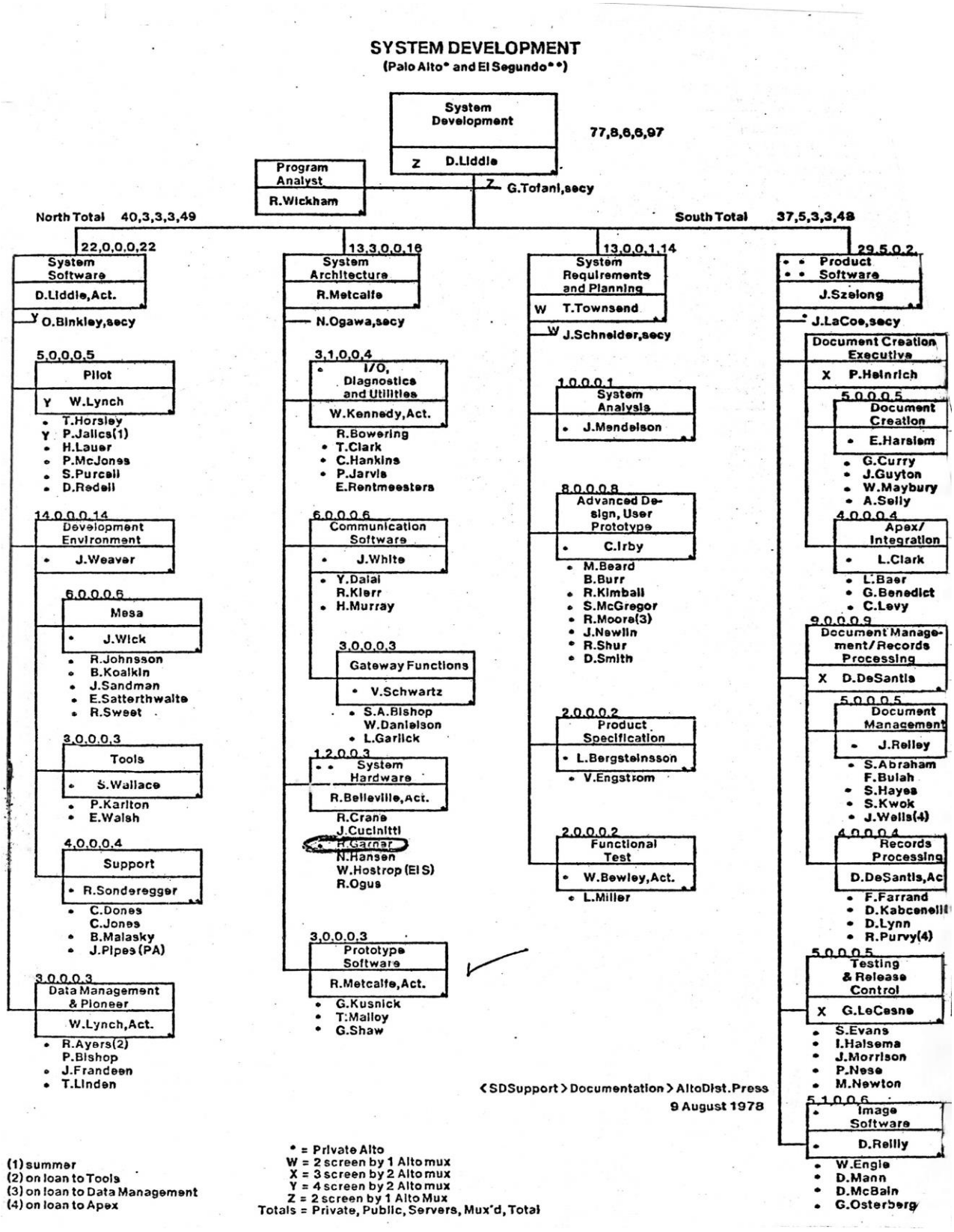


Photo 1-C: 3-Mbps Ethernet controller for Xerox D0 workstation showing stichweld wiring on back, 1978.

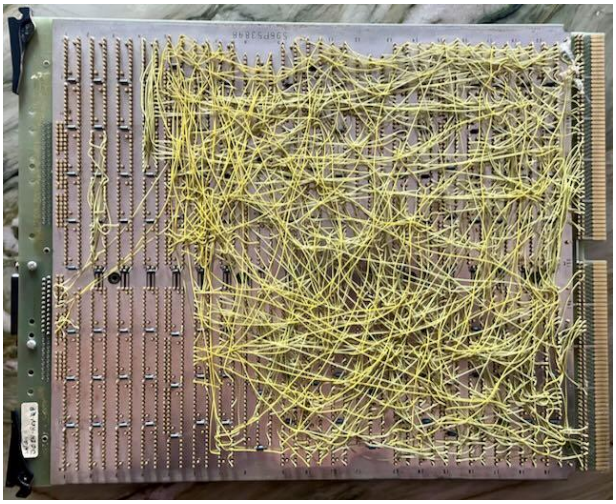


Photo 1-D: Xerox STAR/Dandelion workstation 10-Mbps half-duplex Ethernet controller (top) and CPU card (bottom), 1979

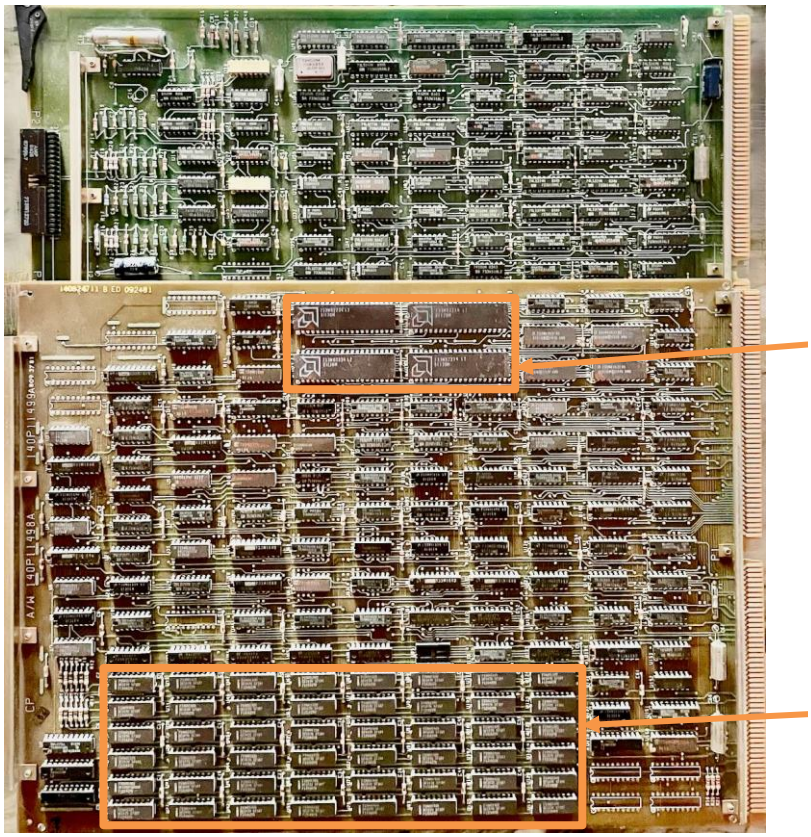


Photo 1-E: Dandelion/Xerox STAR hardware team

(Back L-to-R: Robert Garner, Nora Ogawa (secretary), Neal Hansen (tech), Ron Crane, Roy Ogus, Jim Cucinitti (tech), Don Charnley. Front, L-to-R: Bob Belleville (manger), Dick Snow (visiting from England). Not shown: Dan Davies.)

Image 1-F: Sun Microsystems Engineering Org Chart, hand drawn by Eric Schmidt, March 1984

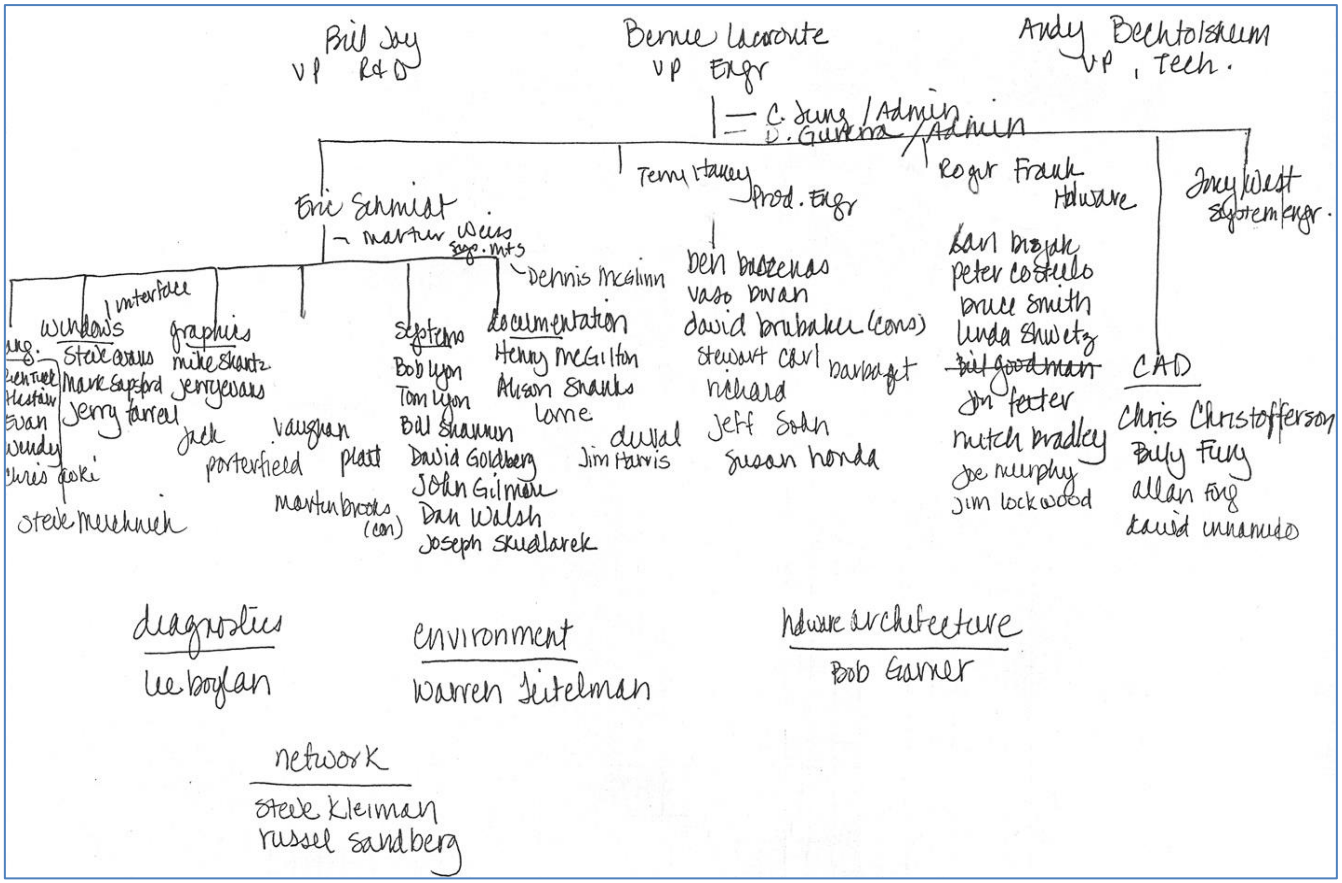


Photo 1-G: SPARC V7 Integer Unit chip, Fujitsu MB86900, 20k-gate array, 1986

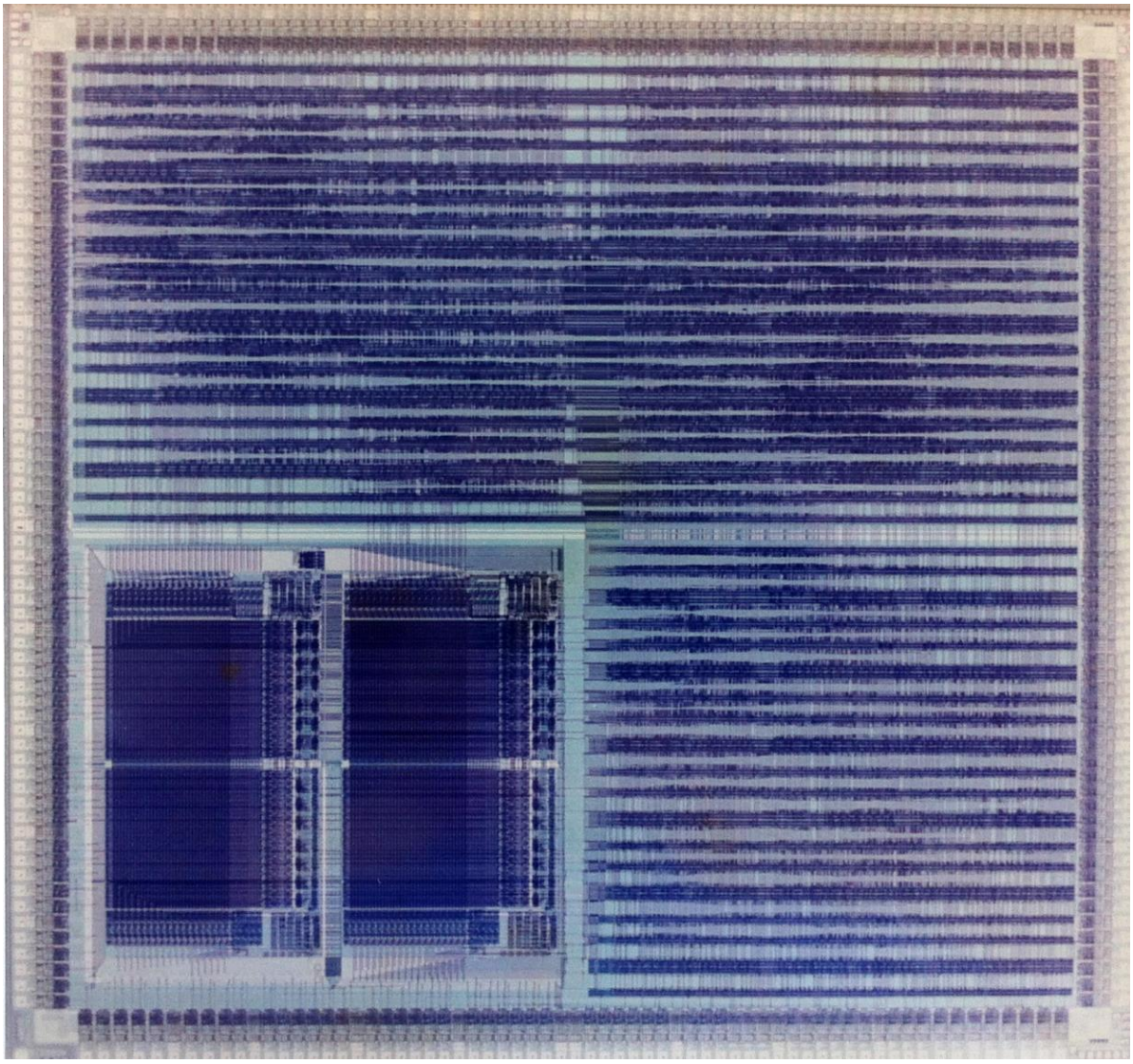
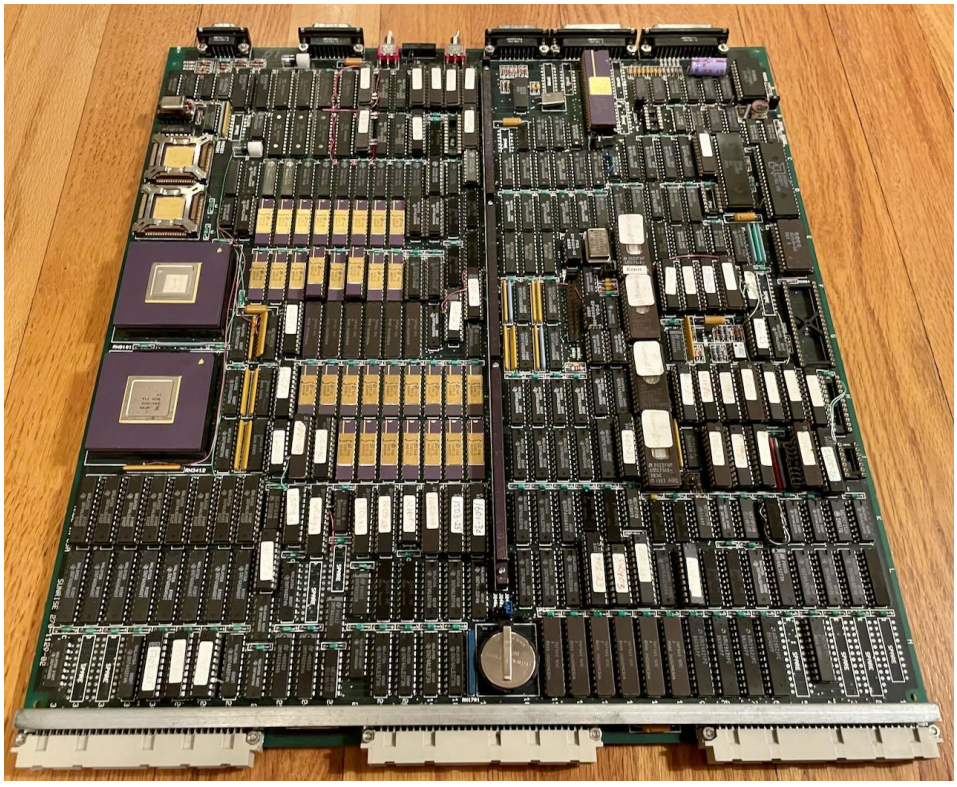


Photo 1-H: Sun-4/200 Processor card, 1987



Sun-4/200 SPARC central processing unit, annotated:

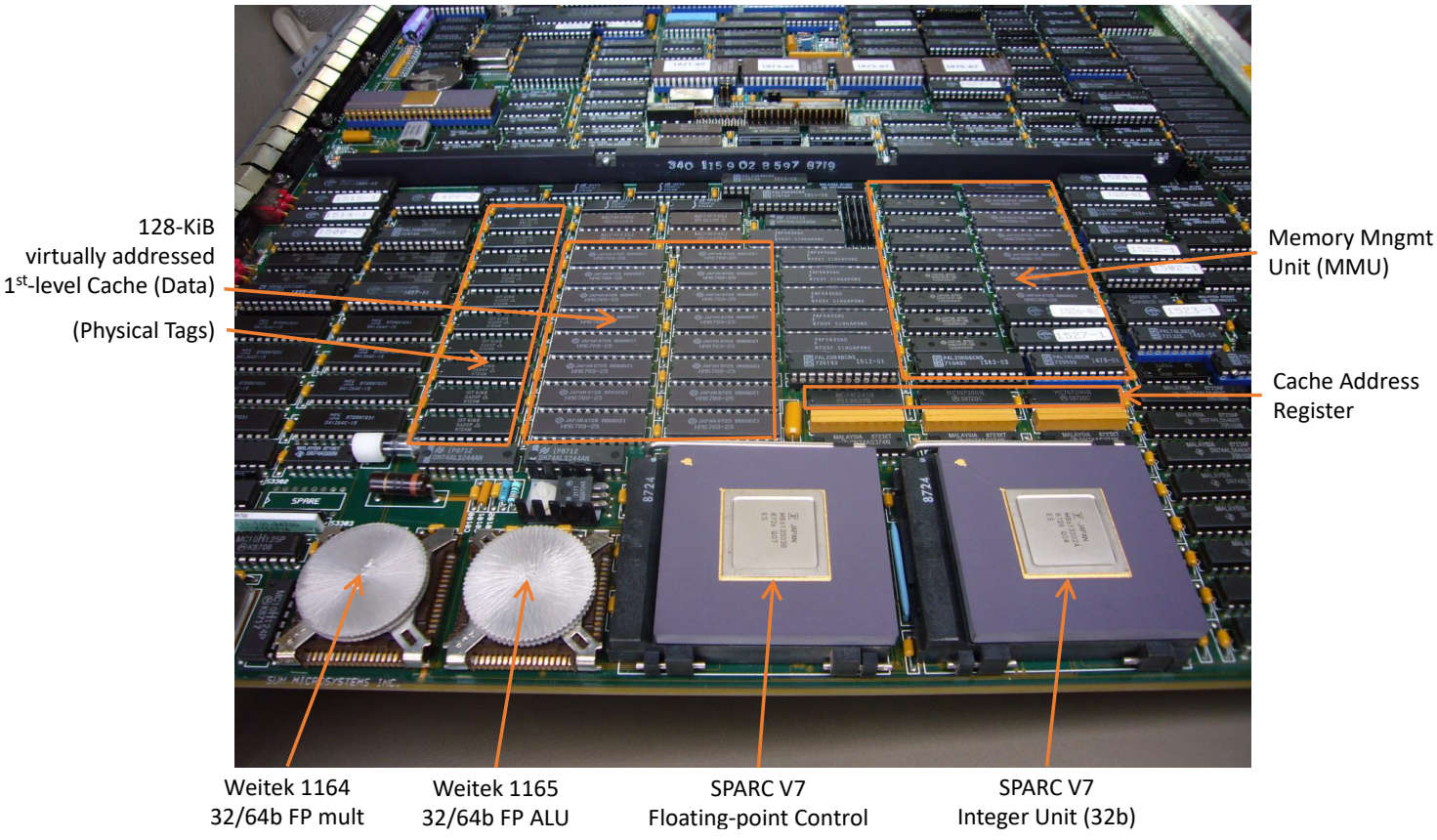


Illustration 1-I: Estimated Cycles per Instruction (CPI) for Motorola 68020 vs. RISC
(R. Garner's engineering notebook, April 26, 1984)

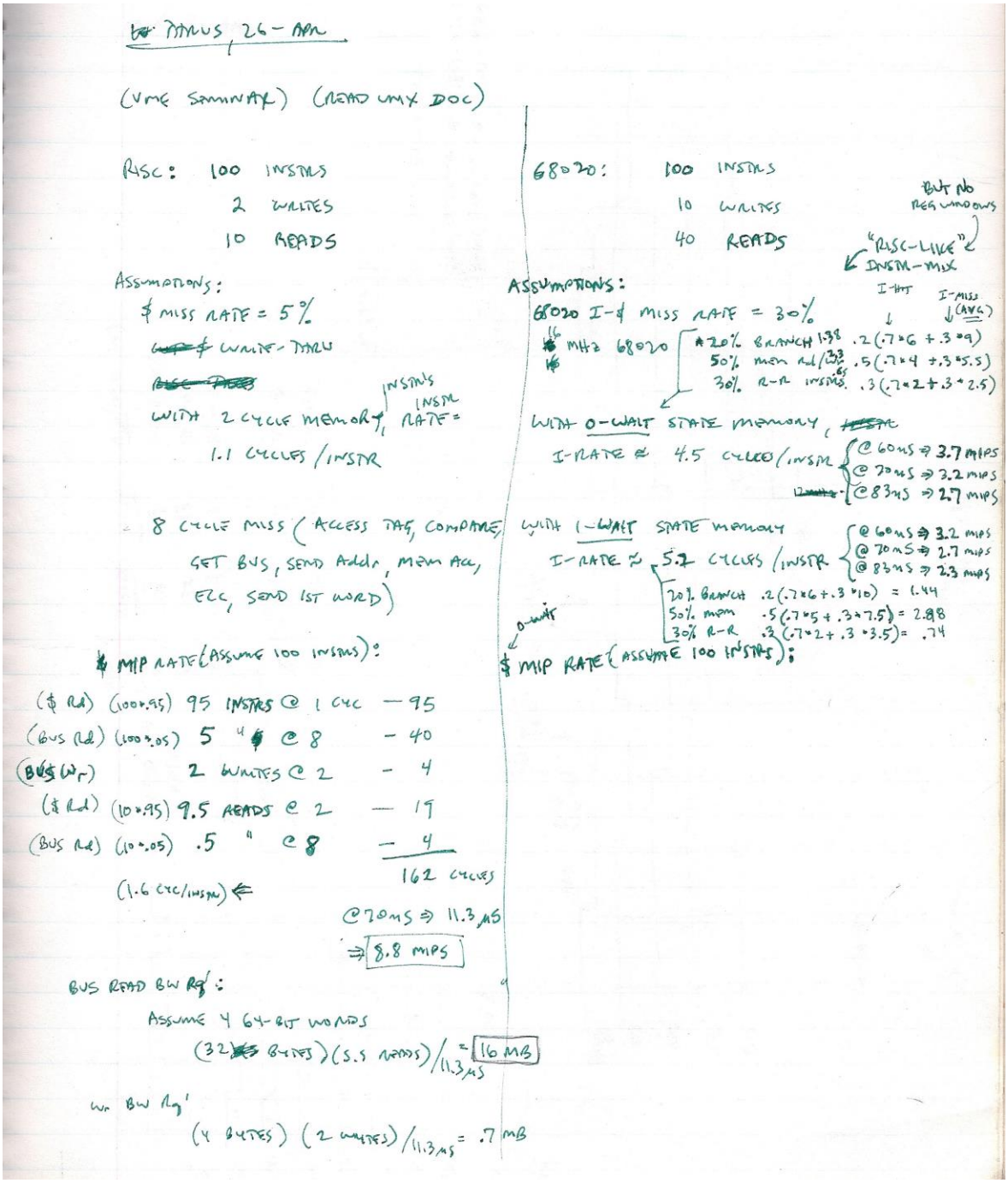


Photo 1-J: Sun SPARC Architecture Team, 1988

(L-to-R back: Dave Weaver, Alex Wu, Richard Tuck, Steve Muchnick, Dave Patterson, Robert Garner, Anant Agrawal, Will Brown, Faye Briggs, Joan Pendleton, Don Jackson. L-to-R front, kneeling: Wayne Rosing, Sunil Joshi, Dave Goldberg)

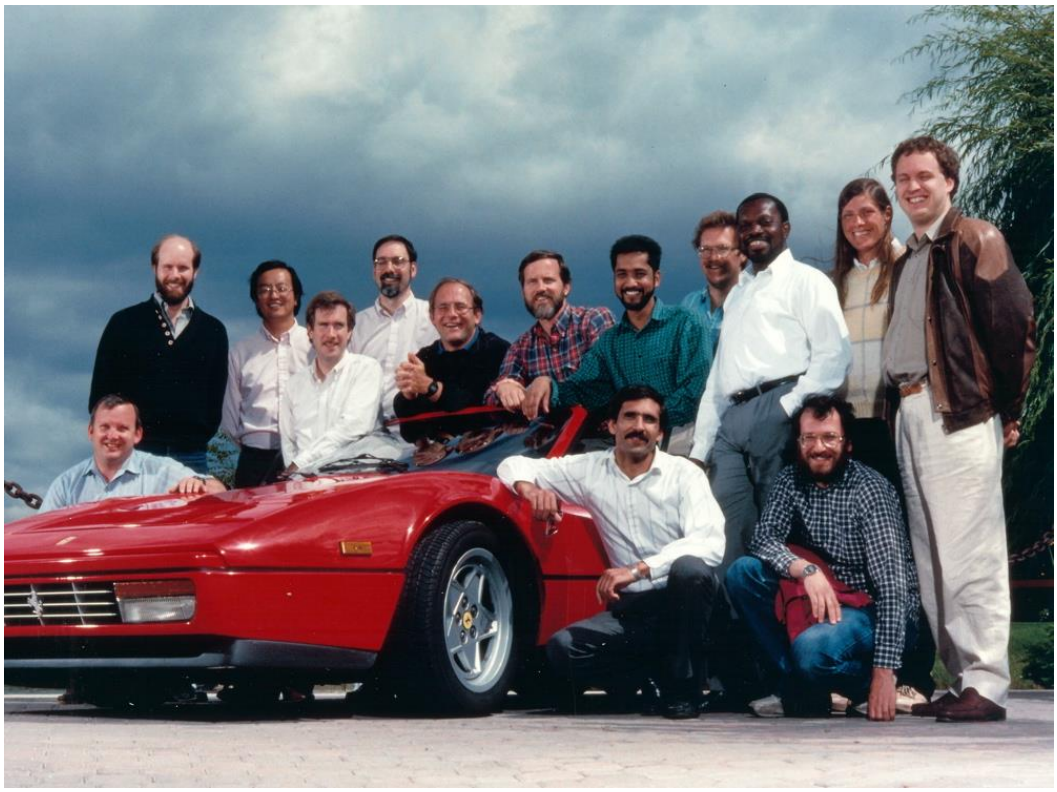
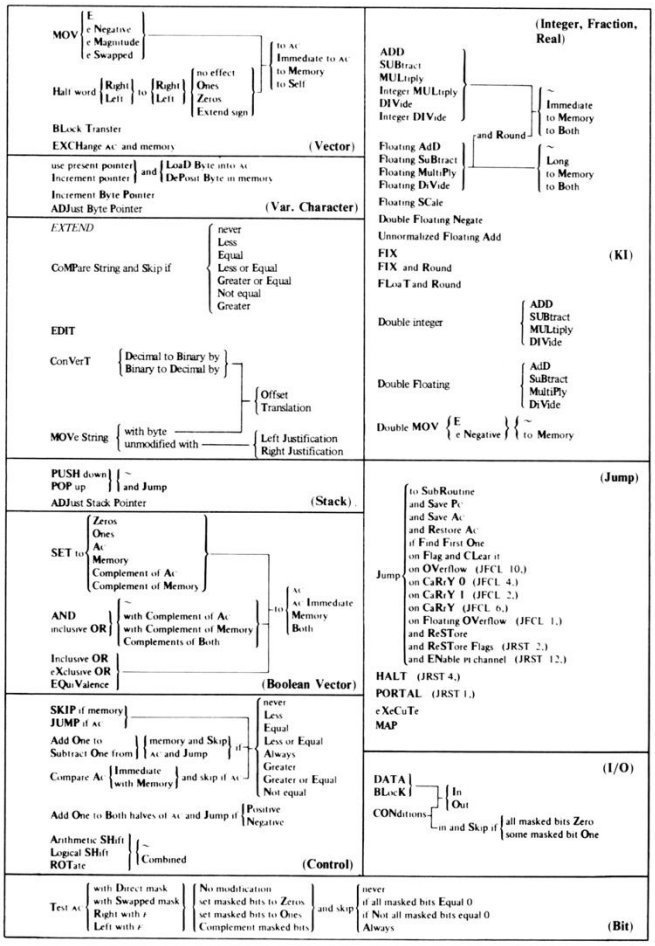


Photo 1-K: Sun-4/200 Hardware, Software, Manufacturing and Marketing Teams, 1988

Image 1-L: Diagrams of the DEC PDP-10 (left) and SPARC Instruction Sets (right)



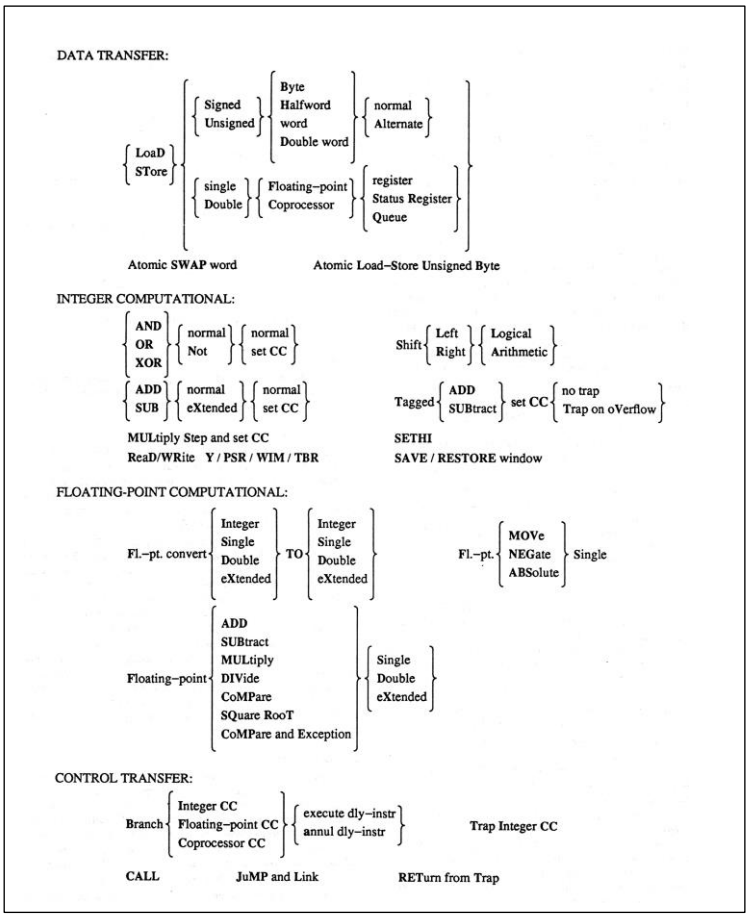


Photo 1-M: Sun SPARCcenter-2000 board, 1991

Two S-bus cards (left front), two empty S-bus slots (right front) showing two LSI I/O interface ASICs.

Photo 1-N: UltraSPARC-I microprocessor, 5.2 million transistors, 0.5-micron process, 1994

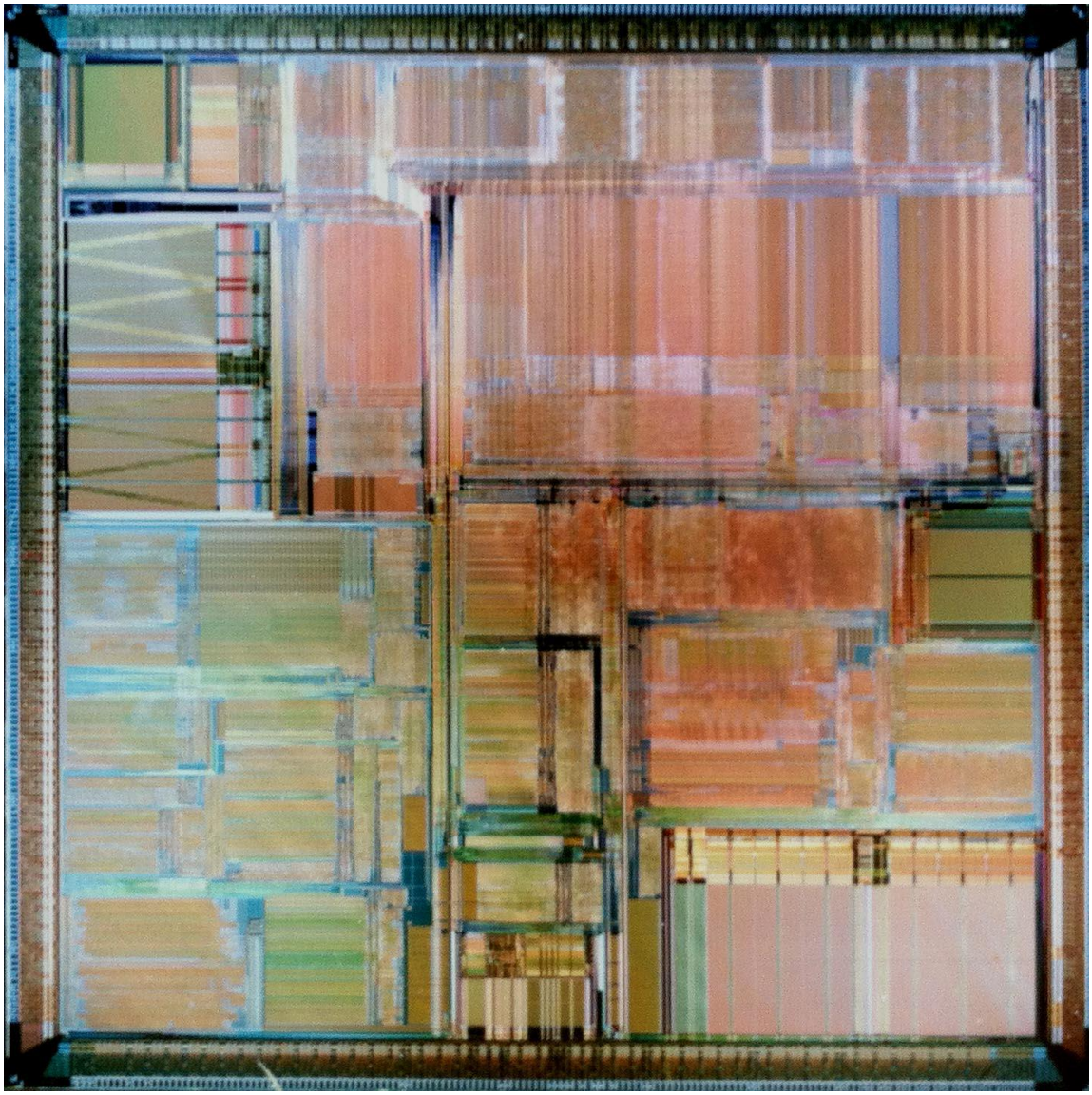


Photo 1-O: UltraSPARC Design Team, March, 1995

Chart 1-P: RISC Peak Performance, 1987- 1995
(Charts by R. Garner,1995)

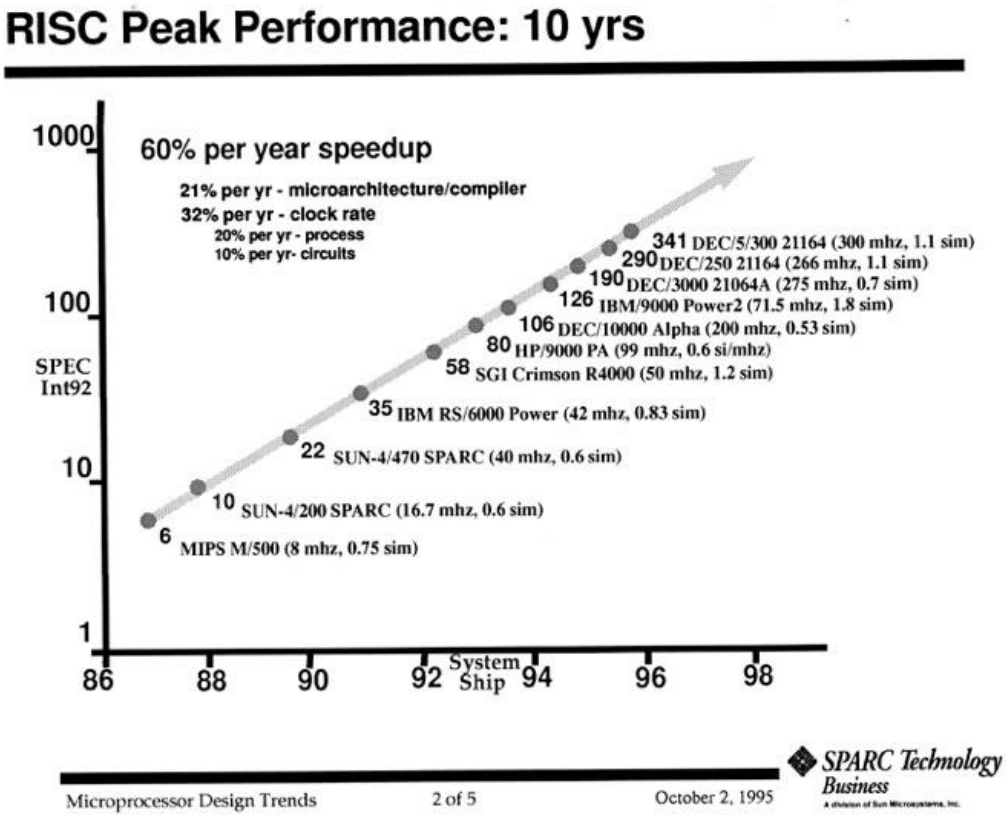


Chart 1-Q: Projected Future RISC Peak Performance, 1996 – 2008

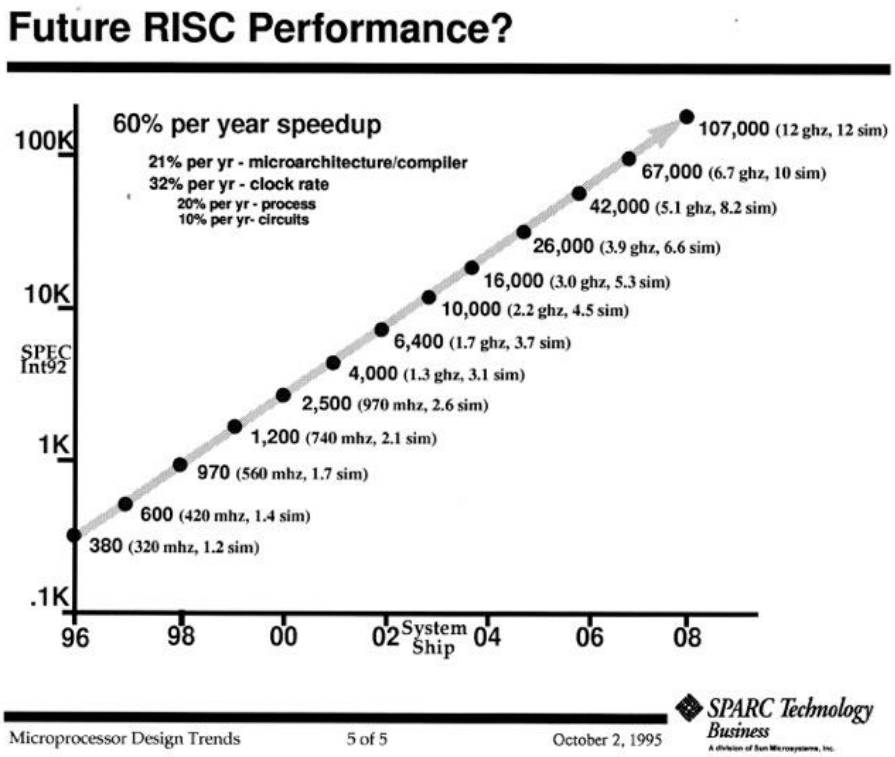


Chart 1-R: RISC Peak Integer Performance, 1987-1996
(Charts by R. Garner,1996)

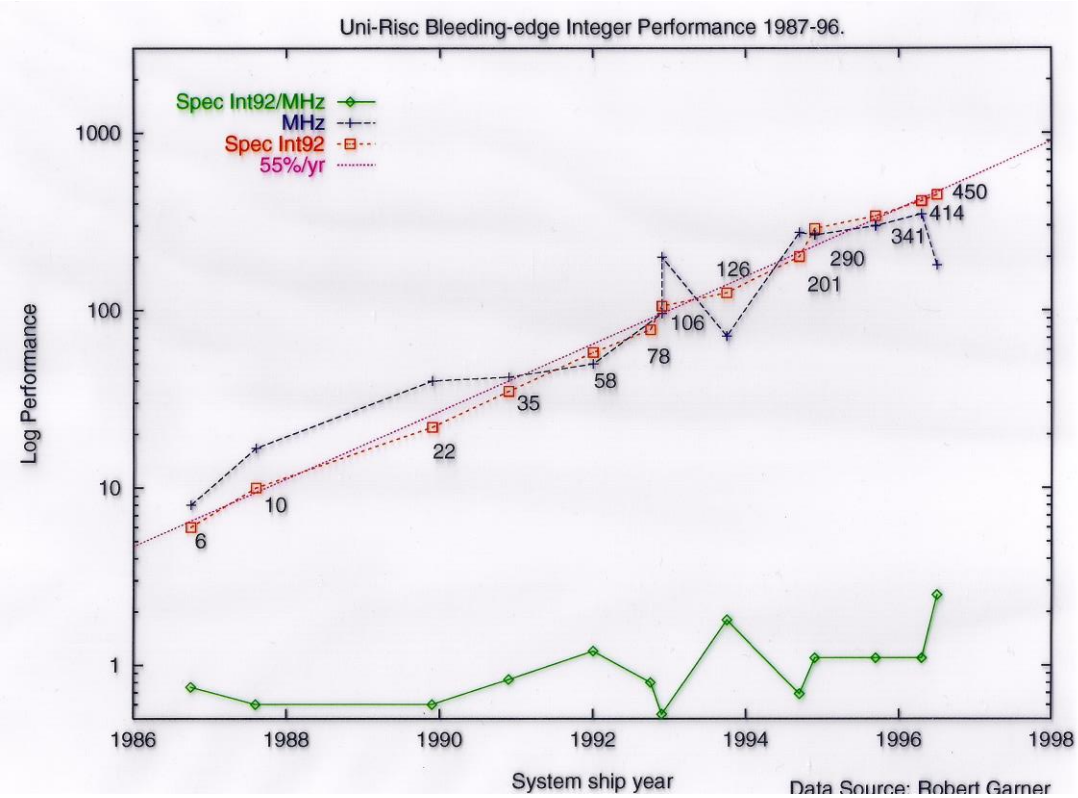


Chart 1-S: RISC Peak Floating-point Performance 1987-1996

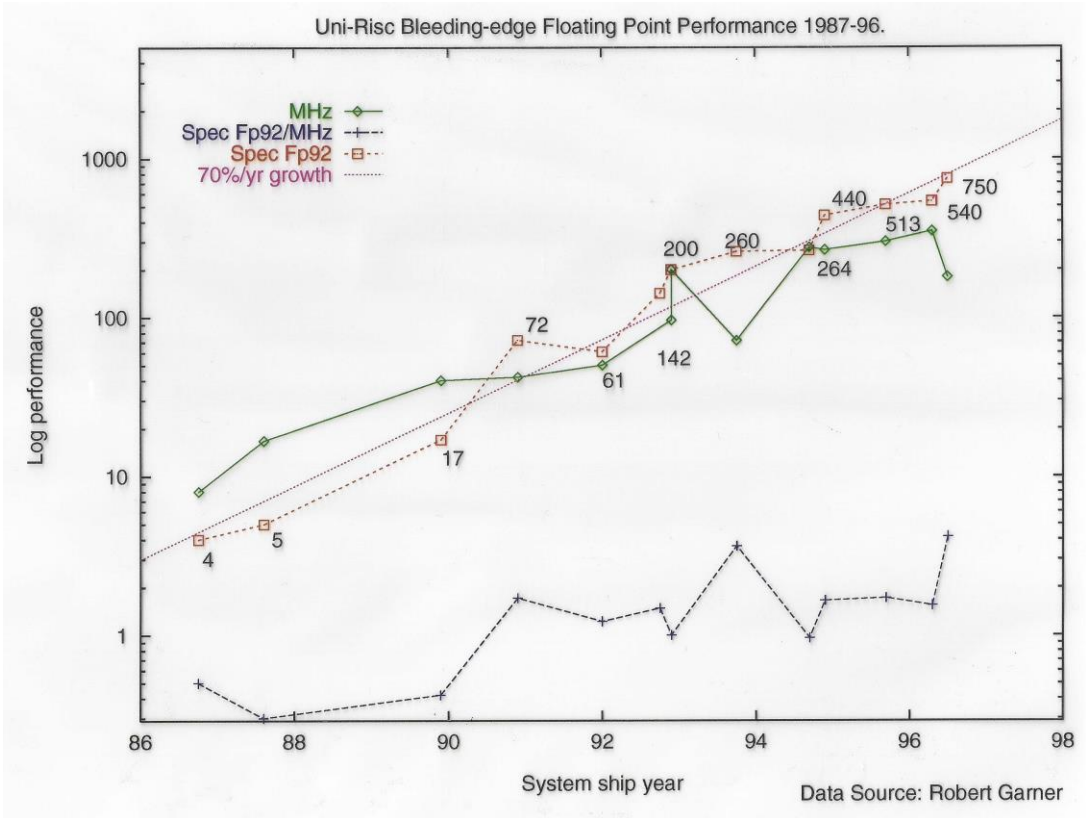


Chart 1-T: Predicted Future RISC Integer Performance, 1996 – 2010
(Chart by R. Garner,1996)