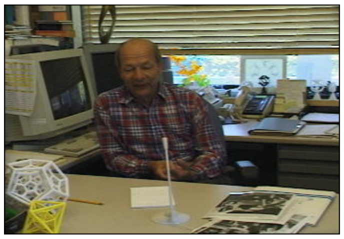


Séquin: I liked the job at Berkeley here in the stimulating environment and the work with my colleagues Dave Patterson and Al Despain and the nice weather. So I said, "Yes," and during that summer or at the end of that summer, we moved from New Jersey to Berkeley. Again it was kind of a dramatic move. This time not by boat of course, but it was almost like the old settlers. We had our Volkswagen Square Back, which seven years earlier we had freshly bought in Germany and then moved with us to France by boat here to this continent. We now loaded this Square Back up with our important belongings.

Actually, sorry, that’s the year before. The big move was when we when we came to Berkeley in 1976, where we had our belongings for a year that we needed and our two children in that car. We took like two weeks to travel through the country. We’re camping along the way and stopped in all the major roadside parks and our daughter, Evelyn, who was three at the time was rating these various parks by their quality of the swings and other facilities. Sorry, that was ’76. In ’77 basically we had everything here. We just needed to go back for a couple of weeks and pack up our household and put everything in a big van and send them on the way. Then fly out here and wait for its arrival.

When we moved into our house, and now it suddenly was empty of furniture, it didn’t look so nice anymore because all the walls had cracks in it. We realized that the house was like 70 years old and wasn’t as close to the quality that we had in New Jersey. And here we pay so much more for it than we had actually thought, so that’s suddenly a little bit of a shocking insight. On the other hand, it had this great window in the living room overlooking the bay. In the beginning, we had no furniture so we just had our sleeping bags. We essentially had our sleeping bags right up against this window. I think the first night, we probably must have been awake almost half the night just looking out the window and down to Berkeley and over to the Golden Gate and to the lights and the starry skies and all of that.

Then we forgot about the cracks. As soon our belongings arrived we did what the former people had done, we hung a lot of pictures on the walls and covered up all the cracks and then forgot about them and didn’t bother us too much. But the weather too in ’77, it was a very rainy year and, some of the houses slid down in the mud and there were long periods of very heavy rain. I almost felt like it was breach of contract. Here I had this gorgeous ’76, it was a drought and everything was wonderful. It’s under those conditions that I had signed up and now this was just wet and miserable ’77, and so that wasn’t quite as glorious as I had thought. Then I found that now I was teaching a full load, not just a half load I had as a visitor. In addition, there were all this comedy work and stuff and all these other things that I was shielded as a visitor. And I had to start raising money, which I didn't have to worry about the year before. Besides, Manny Blum who was then Chair would stick his head into my office every other week or so and say "Are you publishing yet? Because if you don't publish, you can't survive." So it was kind of scary. It was it was rough, some of those things.

Klemmer: Now did you come in with tenure or you came in on a tenure track?

Séquin: No, I did come in with tenure.

Klemmer: So that weight it was off your shoulders at least.

Séquin: Basically. That load was not on my shoulders, but still I didn't want to come here and embarrass myself and basically *he said that I'm kind of on the fringes and just hanging on there. I would say for several months, every day at least once I asked myself, "Why did I do that? Why didn't I stay at Bell Labs?" Then gradually it got less and less. I think after about a year, my worries had gone away and I just knew this was right. Then actually three years later, I must have conspired enough confidence here, they made me CS Chair. At that time I absolutely knew this was the right decision. I was so glad I did it. But also, there were many other exciting things out here.

When Tom Everhart had hired me, he said, "Look, you're going to be in the CS Division and for your survival, you have to make friends with the CS guys. They're your future. They're going to write your promotion cases. They're going to judge how well you're doing. But you have these various friends on the EE side. And you should keep those friendships. As a matter of fact, you're in a unique position of doing something very important and that is bring EE and CS more closely together. Make them not two separate boring factions, but make them one department and really bring the two halves together."

Klemmer: In 1977, was it one department or was it two departments?

Séquin: It was one department. But there was not much talking. As a matter of fact, there were some animosities, particularly among the old CS faculty whose thought came out of mathematics and originally had thought they would make a new Computer Science Department, which they would then dominate. Then the merger with EE was discussed they thought that it would be CS dominating and EE would be self−absorbed. But the CS side would be the dominant one, of course. That's not how it turned out. EE was there. It was established. It was much stronger. And CS was the fledgling new twin. So if anything,. CS was always realizing that the kind of the minority and all of that. That was hard for some of the people to take here.

There wasn't too much nice collaborative work going on or not too much talking because it hadn't quite become clear where the contact points would be or what that boundary might be. The explicit charge that was given to me by Tom Everhart is to do something about that and bring ’em together. Obviously, me being in the VLSI field which relied heavily on integrated circuits but really wanted to build systems and computers and even some of the chips got so complicated you needed the computers to help build ’em, it looks like there’s ample contact points. That was my mission. And I said, "I’ll see what I can do." It sometimes felt like here I was holding onto two supertankers who were slowly drifting apart. It was my arm trying to prevent them from drifting apart.

It took a little bit of doing, but then through the VLSI classes and through the work in IC−CAD, I think we have really been able to build a bridge and glue the two divisions together to the point where it actually became a model. I think several other departments, MIT and others, they’ve looked towards what’s happening at Berkeley and have tried to take that as a model. I would say some of the most exciting things in the first five years or so was the work on the RISC, SHAPE and SOAR and the like. Then the Berkeley Synthesis Project really was only possible because we had this close collaborations between the faculty on the CS side and on the EE side. And the work in computer−aided design was definitely the best work done anywhere in academia. Again, because we had critical mass on both sides, the EE side and the CS side, and they’re both required to make really good CAD tools.

So that became, I think, a very successful venture in the end and got us a lot of visibility. When we built Soda Hall, or now most recently when Richard Newton tried to go and raise money for CITRUS, a lot of money came basically because of all the CAD companies that at some point had started and felt loyalty to Berkeley and realized that their roots go back to Berkeley and they were willing to help us when we needed their help. Now the whole thing, of course, had started in earlier days. In the early ’70s it was Don Pederson creating the program SPICE. And actually creating an integrated circuit labs at Berkeley and that was something very unique.

When Dick Fateman and I went to Japan to raise money for Soda Hall, often it was pointed out that it was SPICE that really was what they were grateful for. They said that it had saved them, so many tens or hundreds millions of dollars by essentially making sure their chips would work the first time and that they were really grateful something like that was out there and in the public domain and coming out of Berkeley. Soda Hall was basically cashing in on the pioneering work that Don Pederson had been doing with SPICE and with the integrated circuit things.

Klemmer: SPICE was Berkeley’s first CAD tool?

Séquin: Yeah or the one dominant, most important first CAD tool. Not sure whether there are other ones or smaller ones before that, but definitely that's the one that created all the visibility. So how did I get into the VLSI business here? Obviously part of it was because of my background from Bell Labs, but a lot of credit also I have to give to Lynn Conway at Xerox PARC because as soon as I was no longer a Bell Labs employee−−so in my second year here when I was now formally officially employed−− she hired me as a consultant. For many years, I forgot how many one day every week I was spending at Xerox PARC. It was always a day I was looking forward to. And I didn't have any particular fixed assignment.

It was mostly just be down there and have good and stimulating discussions with people like Lynn Conway and Alan Bell, Chuck Thacker. Carver Mead would often show up there. We were talking about designing integrated circuits. That's what I'd been doing at Bell Labs for six years. And Carver Mead and Lynn Conway had the firm conviction that laying out integrated circuit is something that could be formally taught. Now at Bell Labs I never took a course in that. It's something you learn by osmosis. You watched how the senior hacks would do it. Watched over their shoulders and picked up some tricks and used your own tricks and common sense.

Eventually you knew how to lay out integrated circuits. But it was very much like a secret craft. I think it was the conviction of Carver Mead and Lynn Conway who said, "There must be logic behind it. And if there's logic, we can capture the logic and then we can teach it." And that was their push for developing this program for VLSI design. And the pioneering book by you know, Lynn Conway and Carver Mead, on VLSI design, that, I think, was published in 2000. It really revolutionized the field.

Klemmer: So that's 23 years after?

Séquin: Sure. Wait a sec. No, no, no. Sorry. I'm 20 years off. 1980. 1980. All right. It was a round number, but not that round. So suddenly they convinced a whole lot of people in academia, and particularly in computer science that VLSI design is fun and that it's actually something that they could be doing at universities, and can be taught, and might be successful. Even though industry, I would say Intel and Motorola and the likes, were kind of skeptical because they felt, "No, no, that stuff doesn't work and particularly not trying to automate it because even the best program you guys have are only half as good as our best designers." That was true as long as you tried to automate the layout of a program logic array that had self write like 10 midterms and 10 inputs and 10 outputs.

True. Something like that can be done by human designer and they can throw in any tricks they can possibly think of and take shortcuts the computer hasn’t been told about and make a small, more efficient PLA. But by the time you have 50 inputs and 100 outputs and 200 midterms, human design cannot even track anymore−− keep track anymore and will probably make mistakes and certainly not have these hand−crafted little tricks on different lines. The gain is clear on the side of the computer. But it took a few years to convince the die−hards and− −and these established design houses that that’s the way it was going. Carver Mead had this firm belief and so I was pushing ahead and it was very, very exciting to work with those people.

Klemmer: It seems like you’ve made two really powerful generalizations here. The first being that you have cowboy computer scientists building a bunch of stuff. How do you take that and make it a more formal discipline? And you’ve seemed to have pushed the front on both education is one way to make that a more formal discipline. And good tools is another way to make that a more formal discipline. It’s really interesting to me that you push forward on both of those.

Séquin: Right. In a way that’s also an interwoven network of good coincidences. Through the stimulus of working and helping Lynn and Carver with their book I became acutely aware of the educational potential of that field. My first course that I introduced to Berkeley for its graduate course was in VLSI design and was based on the notes that were emerging on this collaboration with Carver and with Lynn. And I used partial notes of the book that they were writing in my courses and debugged it and gave the feedback back to them. At the same time I was working very closely with Dave Patterson, and out of that essentially came the RISC chip.

the RISC chip, of course, was then a first powerful test vehicle to see whether that methodology really works. Because all the students that worked on that chip was going through kind of the same VLSI design methodology and they used the kind of tools that were being collaboratively developed between Xerox PARC and MIT and Berkeley, primarily these were kind of the three leading places. And we had to build on those tools. So building the chip reinforced the educational methodology and then the educational methodology gave us enough students that knew about it and could really work on those chips. And it all fed on each other and at the same time it became now clear what my mechanism was by which I could bring EE and CS closer together.

It was exactly in that domain because we couldn’t really build chips without some of the help of our faculty from the EE side. Particularly, David Hodges and also Alberto Sangiovanni, Richard Newton later, and then the field became more CAD dominated. They took the leadership and they run things like the Berkeley

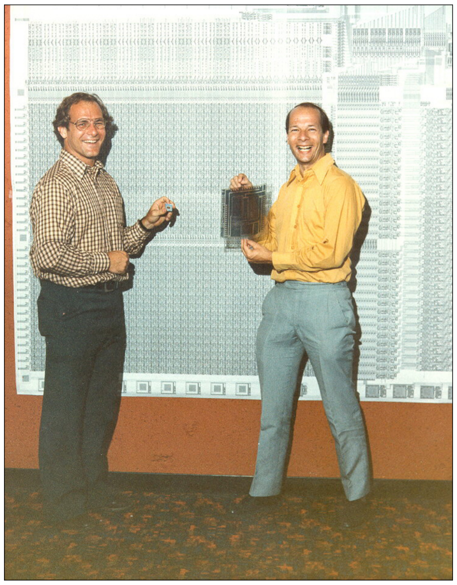
Synthesis Project and further strengthened all those ties. That was very fortuitous. Very quickly I realized that this was a good thing being here at Berkeley and it was far more exciting than just working on one thing at Bell Labs. It was much more multidimensional. And yes, at first confusing and almost overwhelming, but at the same time was tremendously exciting and was absolutely wonderful to be here. And that's when I stopped worrying about did I make the right decision. Because I knew it would have been the right decision, I would have been very sorry and sad had I not done it.

During those years, the Bell empire started to crumble and practically all the good people had left within a short time after that from the area of devices that I was coming from in Bell Labs. It was only a shadow if its former self. So the excitement had really gone out of that particular area at Bell Labs. So it definitely was a good timing of jumping ship and coming to Berkeley.

Klemmer: Could you talk a little bit about what it was like to be a professor and a consultant at the same time? And whether that was supported or discouraged, or how that played out at Berkeley?

Séquin: In general I think it was supported and the experience was 100 percent positive. It was absolutely a wonderful thing to have this in a way, one day off or at least one day different. It just gave sort of a rhythm. One day was clearly different and it helps recharging your batteries. It gave me new insight from the outside, which I immediately used in my classes. It would have been harder to get the same kind of impulses from the literature or whatever. It was really vital to keep me current. At the same time, the feedback from the classes to the book was equally important to Lynn and Carver.

This whole community that was building up with sponsorship



Sequin worked very closely with David Patterson in creatingthe RISC chip.

of DARPA, who early on, I guess, was convinced of creating this design environment but also a whole new generation of designers, which would clearly be needed for the computer chips that were very rapidly growing in complexity. We just didn't have nearly enough old−fashioned designers to keep up with the demand of designing all those chips. So we had to have a pool that's at least 10 times bigger. This effort here spearheaded from PARC and then tried out, first at Berkeley and at MIT, and then very quickly it was picked up by many other schools, Stanford, the University of Washington and other places, which then spread quickly.

What was really vital in some way for the quick evolution of the electronics industry, particularly as far as VLSI digital logic chips were concerned is computer chips. In Berkeley we had our own excitement in that respect. I vividly remember one day Dave Patterson and I were driving down to a microprocessor conference in Sylmar. While we were riding down 280, which is kind of easy driving and a little boring, we're talking about plan for the next academic year. Patterson wanted to do a three−term sequence where students would take maybe the VLSI course first and then an architectural course to evaluate new architectures. Then an implementation course where they would actually put everything together and build a chip. We were talking about potential forcing functions and he suggested building a microprocessor. And my instinct reaction was, "Oh no. That's way to complex. No way that we can handle that because I'd kind of seen what it takes to build an integrated circuit and how big an effort it was and I had a feeling that with nothing but amateurs and students, that would be would be a difficult task." But fortunately I had the good sense of not making any noises in that direction because I know the most important thing to get something going is to believe that it can be done.

So while I was often a little careful and timid about what I was setting out to do, I didn't want to spoil this so I didn't make any such noises and said, "Oh yeah, let's see what we can do." But I made a point that we should keep this chip as simple as possible and only put the functions on there that we absolutely have to have. He agreed to that. It made good sense to him. We argued back and forth about what might be on the chip and what not. Dave Patterson made it clear that maybe that would be the first study in this architectural class that he would offer where we actually would look at how bad does a computer chip get if we leave out multiplication and leave out all kind of other things, and only have maybe less registers or whatever complexity we could strip out.

How bad would the computer chip be and would it still be accessible? So at that point I pointed out almost a joke that "That's going to be risky business. Let's make it a RISC computer. For the reduced instruction set computer and see just how little instruction set we can get away with and have a decent computer." So that's what Patterson did. He did essentially run this class. They analyzed what a RISC instruction set would really do to performance. The amazing thing was it seemed like not be that negative. As a matter of fact, it seemed like if you did things right you could actually better performance out of it because by leaving out the really complicated stuff, the machine cycle became so simple that you can now run the clock faster.

Maybe because the cycle is so simple you can run the clock twice as fast. Every now and then you hit the instruction like multiplication that you don't have in your instruction set and you need to decompose that now into smaller operations, shifts and adds. It may take 20 instructions to run this multiplication. And you may think, "Oh, that's dreadful." But the analysis that they did in those classes to find out how often do multiplication instructions show up across the board in a whole lot of programs that they're running, they're showing up less than one percent. So if they show up less than one percent, then they're even 20 times slower, you're still ahead. For many of the complexities that we cut out, we realized they really didn't hurt that much at all.

The overall simplicity we gained in the chip really allowed to run the chip maybe twice as fast or more than it would otherwise and that started to look really, really attractive. I think there were some papers that came out of that which were published. But they were just paper studies. Like so many other paper studies, everyone says, "Yeah, yeah." Nobody paid any attention to it. So it was important that Patterson insisted on having this implementation course where we would actually build the chip. When we finally had the working RISC chip, it actually worked and we had real clock data and real performance data. We were allowed to publish in the very competitive international and solid−state circuits conference, bang, suddenly everybody paid attention. We actually have a working chip.

Now everybody clambered to read the papers and look at all that. And caused the revolution that it actually has caused. Also Dave Patterson did a fantastic job. He really became a missionary. He traveled the country. He must have given 20 or more talks in one year about RISC computers and RISC versus CISC, that's his version of the complex instruction set computers and the like. So the word got around very quickly and before he knew it, companies clambered to call their computer RISC just because it was now the in−word, even though many of them were not RISC at all because they still had like 200 or 300 instructions. But they still tried to find some way of how they could justify calling it RISC because everybody wanted now RISC computers because it was the cool thing to have. So that was very exciting.

Now in order to build the RISC computer, we actually had to have some tools to make those layouts. There really wasn't much good stuff around and we didn't have any of the then current Alma, or I'm sorry, Calma or Applicon Systems. Our students complained because all we had was Caesar, which was a little home−brewed system that John Ousterhout had essentially single−handedly put together. It was just an editor that allowed to place and connect rectangles in different levels and that would be then the mass pattern to make that particular chip. It was very straightforward, just doing the simple operations that needed to have. There were not even 45−degree lines. You could only go 90 degrees, so−called Manhattan geometry. And the students felt that while it worked, it seemed like a rather kludgey and dull, the great things we could do if we had a Calma System or an Applicon System and all that. On one of our fieldtrips, we took our students down to Intel or Hewlett Packard, I forgot, one of those places.

Klemmer: This was an undergraduate or a graduate class?

Séquin: It's mostly a graduate class. But there were undergraduates in it as in most our graduate classes.

Klemmer: And how big was it? How many students?

Séquin: Oh, I don't remember, but I would say a guess around 20.

Klemmer: Mostly in computer science?

Séquin: Yeah, almost exclusively. Maybe a few EE students, but mostly coming from the CS side. So we took them to down to one of those places where they built integrated circuits for profit. As part of the tour we actually got to see their layout room, which may have had about eight desks because I think it was Applicon computers, and we noticed that on the other side of the room, there were about eight drafting tables with green Mylar on them. All the engineers were bent over the Mylar with pencil and erasers and were drawing and marking up on there. And there were some clerks and some girls sitting at the Applicon tables and busily entering things. Looking at Mylars and then entering it to the computer. We felt that was strange. Why would you first draw it on paper and then enter it to the computer? So we watched a little bit closely.

Then we realized there was one woman and she tried to make a row of like soft pen contacts on one of those computer chips. It was a very sophisticated system so it could make something like an alpha or something that would signify to the computer, "I want to enter a contact." So she would scribble this sign, then wait a few seconds. And nothing happened so you would scribble it again. And maybe a third time computer goes beep and now computer understood she wants to make a contact.

Klemmer: So it was a pen−based computer?

Séquin: Yeah. Had a tablet.

Klemmer: Back in 1980?

Séquin: Yeah, not on the screen. It was on the side tablet. Now she would go and mark across on the actual layout screen of where that contact should go. The computer was very sophisticated, so as it placed a contact it would check the design rules to make sure that no two pulley features were too close and the contact window didn't cut anything else. That took about eight or 10 seconds. So she needed to place, say five of these contacts. There she finally had convinced the computer, "I want to make a contact." She placed the first one and then it goes, one, two, three, four, five, six, seven, eight, nine, beep. Then she could make a second cross. Then it would go one, two, three, four. And you can see I don't go through all 50 seconds because by the third contact, it has driven us bananas and we realized this is ridiculous. We can't work like that. And that's exactly so.

That's why the engineers, they go zip, zip, zip, zip, mark five crosses in about five seconds and that's that, and then they can go and scratch their head and what next? But they're in command, they're thinking at their rhythm and they're putting information down at their rate and not at the computer's rhythm. Of course, the system didn't have to be that slow. It was just a very bad design. A few years later, John Ousterhout had created the Magic System which had automatic design rule checking. But learning from that experience, we knew you never ever take control from the user. The user always has priority. So what should have happened is you go and say contact here, here, here, here and here. And the system immediately acknowledges that and shows you a sign.

Then in the background, while you scratch your head and now what do I do next, it starts checking. If you come back with new things to do before it's done checking, it just suspends the background process and watches what you're doing and gives you the feedback and then he goes back. Then when you go for a cup of coffee, in the background it cleans up all the details. There are so many things wrong with the user interface of those systems that in spite of them costing tens of thousands of dollars, having very kludgey, big apparatuses, they were really hard to use and people wouldn't want to use. Whereas the little bit of scaffolding that John Ousterhout threw together in a way was more powerful and certainly more fun to use, even though it was very primitive.

So we have this contract with John Ousterhout that for every chip that we would build, he would build a new tool that would address the critical need. Now, he built Caesar so we could do the layout for RISC−I. That worked and RISC−I actually was operational. But it was four times slower than what we had predicted because we overlooked one long, winding control pass where the signal had to ripple through and around the check then through a PLA and another PLA and this and that. That one slowed everything down. We had to slow the clock down by a factor of four to make sure that signal could complete its path before we could hit the next machine cycle. We said, "Why couldn't we see that before? We needed a tool because even though the chip was very simple in some sense, it was still hard to see everything that's going on at once. So the next tool that John Ousterhout built for us was a timing verifier.

It was a tool that would analyze essentially the change as it happened in each clock cycle, see how long it would take to ripple through the various paths, and mark along the cycles. Says, "Look this one takes that long." Says, "Oh, no, no." Then you can do something and break it into two cycles or put the buffer in to speed up the signal or whatever it takes. Bring it down to the allocated timeframe. With that timing verifier, we then built the RISC−II chip, which was essentially the same architecture but slightly different layout and we took care of all those problems. That one ran to spec and that's the one we could then publish in the IEEE Conference on Solid State Circuits.

After that he built the Magic System, which took the layout one level up. Rather than laying out individual rectangles exactly where they go and to their exact dimension, you would only lay out symbolically the connective paths between the various elements and plop down maybe complete blocks symbolizing transistors the crossing essentially of a silicon gate over a diffusion channel. And create the overall arrangement that connectivity and verifier that works. The geometry would be automatically compacted by separate program that would run and move all the features and munch them together as close as they could go without violating the design layout rules. With that environment it was then possible to build the SOAR chip, which is a small talk on a RISC, and later the SPUR system, which actually required the construction of three separate chips. But by the time we got to SOAR, I essentially had been commandeered to become Chairman of the CS Division.

I was pretty much out of the loop and it was Dave Patterson who had to carry the torch alone at this point. He gladly went on and with his enthusiasm and made all of these projects into good learning experiences and good successes. Except SOAR was a typical second system. Where you now think you know what to do and so you want to do everything one better and you throw everything in and the kitchen sink. That chip really was too complex. It was actually not really a RISC.

Klemmer: What an ironic problem.

Séquin: Exactly. Exactly. So then you have to back up. Says, "Wait a minute. What was the spirit here? Oh, keep it simple, stupid was the real mission and spirit, and evaluate carefully what features actually contribute the most towards the final performance. Everything that's not among the critical ones are the ones you are throwing out." So that was forgotten in SOAR. That garbage collection of all kind of neat stuff in there, but there definitely wasn't RISC anymore in the true spirit of the word. I guess it was the SPUR system that a nice compromise was found. By that time, technology had advanced. You could do many more things on a single chip. We still created a three−chip system having a floating point unit separate and a cache separate from the CPU. Now the SPUR system was a very defining moment in I think EE/CS collaboration. Probably the highlight that I've seen here in all the years. Involved several people from the CS side and several people from the EE side. I think at peak it had six or seven faculty involved and up to about 30 students.

Klemmer: That's huge.

Séquin: That's huge.

Klemmer: The management of that must be−−

Séquin: Yeah. Again, I think a key manager was Dave Patterson. He is an incredible coach and with his charisma, but yeah, level headed sense, good sense of humor. He was able to keep this all together. At moment there was a friction and tensions but somehow, I think he was one of the key people that really kept it together. The key thing at that moment is that we developed tools and chips simultaneously. That was definitely unique and that was definitely something very special Berkeley EE/CS that nobody had done so far. It was done at the scale that nobody would have dared to attempt. There were several courses involved at a time. I forgot all the numbers, but some of them were EE courses, some of them CS courses. Didn't really matter at that time, we definitely had an audience that it was equally balanced between EE and CS

The Smalltalk on a RISC (SOAR) group included Bill Joy (backrow, third from the right) and Paul Hilfinger (back row, thirdfrom the left).

students and they both took both kind of courses.

We had a matrix organization. Every student involved had to be involved in one of three chip design efforts, either the CPU, or the SPUR cache, or the SPUR floating point unit. As part of that team, he would learn very intimately exactly what it takes to lay out the chip and would use many different tools to do that, to do the layout, to do the verification and do the compaction and various timing analyses. At the same time, we were pushing the various tools that we had. The layout tools, the compaction tools, and the verification tools, and timing tools. Every student also has to be a member of one of those tool development groups. And as part of that team, he would find out intimately what it takes to actually build a tool that could be used by others because all the others would then use his tool.

He at the same time, would got to use other tools as part of one of the chip design efforts. So this matrix organization I think was really truly brilliant and was at the heart of this effort and Richard Newton and Alberto Sangiovanni were key ingredients. They were basically pushing the CAD side through the Berkeley Synthesis Project. Then Patterson was more in charge of the architectural development. The fact that we're building tools and using them immediately to do a real world tasks, something that we really care for that we wanted to get done. Not something trivial to just show the tool can be used, but actually a task that stands on its own and is not defined by the tool. It defines the tool. I think that was different and that's why the Berkeley CAD tools were a class better than everybody else's CAD tools. Just did an academic exercise and built a tool to show, "Oh, I can do this," but didn't really build a real chip with it and prove that it can be done. So this, I think, made Berkeley a truly unique place.

Klemmer: Could we mention some of the professors and maybe a couple of the star students that were involved?

Séquin: The best thing is if I looked that up off line. Pretty bad with names as it is and when I edit the final thing we can fit in some of those and maybe I can even consult with Patterson make sure I give the right credits to the right students. One thing I can say offhand, the RISC−I and II chips by Manolis Katevenis and Robert Sherburne. They both were really outstanding designers and then very meticulous and very careful designers. The fact that the chip really came to work and worked as well is primarily due to those two students' efforts. Manolis Katevenis went to Stanford afterwards became a professor there, but then went home to Greece and is a professor now in Herculean. Bob Sherburne is a little harder to track. He has been in many different places. The last thing I heard, I think he's now actually going to business management school because I guess he'd been frustrated that he had been in so many places where things weren't quite done the right way. But I have to look that up too. Every couple of years I got an email and he had another exciting job again. Both of them have really done well as they fully deserve.

Klemmer: Could you talk about the tension, or maybe it's not a tension, of building real systems in academia? One reason that academics often give for not building the fully debugged system is, "We just wanted to prove the research point. And going further than that would just be not really the role of the university." What do you think about that?

Séquin: That's certainly a real danger. I think a lot depends on the environments that you're in and what kind of support you have. I actually experienced that very acutely, so there is no doubt in my mind and most of my colleagues mind that Dave Patterson was a hero pulling through the RISC effort and doing something path−breaking. It happened during my stint as CS Chair that I had to write his tenure case. The first version of the case was rejected by the Budget Committee and they said they couldn't give him tenure because he had only two journal publications. Now, there was a problem. Of course he had fifteen or 20 conference publications. It took some careful writing and explaining to say that we now have entered a new era where systems building has become really important and that in Berkeley and CS, we have made the conscious decision, we wanted to actually build real systems.

First of all, it takes longer and often the payoff comes only many years later when you finally can write the definitive journal articles to summarize looking backwards of what had happened. In the meantime, you have to rely on the conference publications that show immediately what the results are that have been been achieved in every single step. But also, some of these conferences are really very competitive. To get into those conferences is maybe harder than getting into a journal because you get one shot and then you're either in or out. If you're out, that's it for that year and you can try again next year. Whereas in the journal, even if the first time it gets rejected, you can see all the complaints and take care of all the issues that the editor raised and fix up the paper and try again. It maybe comes back a second time for minor modification. You fix those and eventually you're in.

It may take two or three years, but eventually get your journal publication. Somehow getting into good conferences that are really competitive in a way is a bigger achievement than getting a journal article. So very carefully spelling this out to the Budget Committee, which had seen little of that before and had to get used to the idea of engineering systems being the important factors was at task. I got some supporting letters from the Chairman at the time at Stanford and at MIT. Interestingly enough the following year, I had to write similar supporting letters for them because by that time they had similar young faculty that had come through the system's end and faced exactly the same problem. I had to write what we did in Berkeley and that that's now the new time and that was accepted procedure and help them to get their cases gone through. Fortunately on the second attempt there was no problem and the Budget Committee understood.

Klemmer: Where was the Budget Committee? Is that engineering wide or university wide?

Séquin: No, that's university wide. That's the problem because they're seeing people that basically just write down ideas, and now you can write ideas much more quickly than you can actually realize them and measure them and demonstrate them and then write the paper about it. I think that education has taken a hold and now doing systems is understood as being measured somewhat differently and it's really measuring impact more often now than just measuring net number of papers. In that respect the whole RISC and the CAD work here in Berkeley is definitely outstanding. We have established that in Berkeley, the various Chairs including the Deans were supportive of system buildings and did whatever they had to do to support that effort and support the reward structure that would encourage people to do that.

We also had the infrastructure set up so that this was not too daunting. We had good staff, we had good money coming from DARPA a lot, but also from NSF and from other places, from industry support, which allowed us to have an infrastructure that makes it possible to really pull off projects like that. We had a faculty that was big enough, so we had enough people with the same interests and the same specialties, so we could actually form teams of faculty to deal with that, not just an individual faculty. That certainly makes it easier. The fact that they're only half as big and we had only a total of two or three that were somehow interested in VLSI and architecture and CAD tools, the SPUR effort could not have taken place. Having a total in the order of 70 or 80 faculty as we had at the time across EE/CS, the chances that you'll find six or seven that overlap that area was much better and that allowed us to build all that.

We had all the right environment in place to actually, in Berkeley, make that somewhat less daunting. It still takes guts and still it's a big effort. The other thing I think that made Berkeley unique, which I often point out when I go around the country and talk about what makes Berkeley special, is we had a very close integration between graduate instruction and research. We would give the students in our advanced courses projects that were taken right out of the research that professors at the moment are interested in. We would take the results from the research and feed 'em back to the graduate students in those classes. We would take the most advanced tools that we were building as part of our research effort and use them in those classes and have a whole class of non−voluntary testers on those particular tools.

So in those various classes that formed the matrix of the SPUR effort and the Berkeley Synthesis Project for the CAD tools. In those classes they have to do certain homeworks on those tools in a given time and they were graded on that. So they all gave their best effort to make it possible, but boy, would they complain if the tool really had some flaws. And, of course, those complaints immediately got back to some of the tool developers, who were actually sitting in the same class. They heard the outcries from right and left. So they knew exactly what to fix and they fixed it typically within days in order not to have the wrath of their colleagues on them. And they fixed that and made it possible that the tool was good enough to actually do the homework assignment.

This cross−fertilization was very important; it pushed tools very quickly to be actually usable for real task and usable by many other people other than the designers themselves. Another frequent story here is the way we pushed the frontier, the one that I mentioned at the retreat where Alberto Sangiovanni was teaching one of those CAD courses. After a first somewhat mundane assignment dealing with some optimization of some procedure in this classical SPICE program, he got some complaints from the students that that was mundane and couldn't they have something a little bit more interesting and more challenging. He said, "Okay. I know what to do." And he gave them the assignment for the next three weeks to make a compact end level channel router.

Now a channel router is a program that connects a number of wiring pins, say A through Z, on one particular block, to a number of wiring pins, say 1 through 20 with potential branches and so on, which are maybe in a completely different order on a second block. Having those blocks to be as close to one another so that you can only have as few longitudinal rails running between those two blocks as you can get away with. The idea is to try to have as few of those longitudinal rails and so you need to make these dog legs and zigzags jogs and, of course, you need at least two levels in order to do a good job on that because you then run, say the horizontal lines in metal and the vertical lines in polysilicon. That way you can simply jog over certain distance, go to the other level, go in that direction, jog up to the first level again and go across.

The state of the art was that for these two levels of metalizations that were pretty good algorithms to make channels with the minimum number of these longitudinal channels and potentially minimum number of up and down via poles. Technology was moving ahead and was actually providing a third level of metalization in many chip manufacturing places. Also those blocks that have to be connected. They grow ever larger, from 20 panes they now wanted 50 or maybe 100 panes and much more complicated wiring pattern. So it was really crucial to use that third layer on the best possible and of course now it’s not so obvious anymore, which way should you zig and zag in just two layers. It was on the horizon becoming clear that before too long there might probably be a fourth level. Who knows, maybe one day we have a fifth level.

So Alberto boldly stated to the students, "Build an end level channel router, where end can be anything from two and actually higher than two. And design an algorithm that if I tell you you have seven levels, you will do the best possible thing, try to minimize the number of rails between those two blocks, and do the best job possible." What he didn’t tell them is that even three level was considered pretty daunting by most people in the field and people had tried and nobody had to come up with something really good yet. But students didn’t know that and they were happily going away with a neat looking problem that seemed challenging. After two and a half weeks they came back and said that seemed like a hard problem and could they possibly have an extension of another couple of weeks or so because this was really a hard problem. And Alberto says, "You’re kind of sissies, but okay, I’ll give you 10 days, but that’s it, not more than 10 days and that’s it. And you have to have your solution by then."

And they were happy to get at least 10 days and they went away and did some work. Now during that time Alberto got a phone call from a colleague that worked on some of the eastern universities, and they talked about research. This colleague of his said that he had just applied for a grant with NSF and he wanted to study end level channel routing over the next three years as part of his research program. Alberto pointed out and says, "Yes, that’s just a homework assignment I gave my class this past week." Then supposedly, he tells the story, it was silent for about one minute, disbelief on the other end. But I love this story because it points out of what you can actually achieve if you don’t know how hard things are. And the students didn’t know.

I should say, practically all the teams of three or so students per team, came up with some solution. Some were elegant, some were kludgey, but they all found some way of doing something. Two of the solutions were so good that immediately after they could be extended into papers that were submitted to the big annual conference on Computer Aided Design. One of those papers won Best Paper Award. Both of these efforts were then stepped up to Masterson, I think one even to a Ph.D. program, before too long became core of layout systems and really important functions. So this kind of integration between research and class work I think is something really, really wonderful. We have used that a lot at Berkeley and always with great success.

So under all that, it then becomes possible to actually build real systems. Because you have the culture, students know they have to deliver not just paper but something that works, they expect that. They expect you to use other people's work. They expect to build stuff that other people can use. I think with this sort of a background philosophy and then infrastructure that's supportive enough and you're higher ups essentially doing whatever needs to be done to give you the right reward structure. Then I think real systems building becomes a fulfilling job at the university. I think we're lucky and blessed in Berkeley that we can actually do that and we have such an environment. But it's not possible everywhere.