Oral History of Robert K. “Bob” Brayton

Interviewed by:
Douglas Fairbairn

Recorded: March 20, 2014
Mountain View, California

CHM Reference number: X7128.2014

Douglas Fairbairn: OK. So we're at the Computer History Museum. It is March 20, 2014. And I'm Doug Fairbairn. I'm here with Robert Brayton, who is professor emeritus at UC Berkeley, and previously employed for at least a couple of decades at IBM. And we're going to learn about his life, his career, and the contents of some of those 450 technical papers and 10 books, which he has had a role in the authorship of.

Robert Brayton: I don't talk about any of the contents.

Fairbairn: So we won't quiz you on every one, but we'll make sure that you actually had some part in some of those. So I understand you were born and early years were spent in Ames, Iowa. It must have been a farming town?

Brayton: Well, I was born in Iowa. And when I was in seventh grade, I came to Ames. And Ames is not really a farming town, because there's a huge university there, Iowa State University. And I lived about seven blocks away from the university. So it wasn't typical Iowa like that.

Fairbairn: So before seventh grade, where were you?

Brayton: Well, a little town of Humboldt, Iowa. Ever heard of it? Well, you've heard of Humboldt.

Fairbairn: What year were you born?

Brayton: 1933.

Fairbairn: Right in the middle of the Depression.

Brayton: Depression yeah. Fortunately, my dad was a teacher, so those jobs continued. But I think they cut the pay by half or something during the years.

Fairbairn: Interesting. So there was an educational bent in the family from the very beginning? Did you have brothers and--

Brayton: Well, not really. My dad grew up on a farm. He was in agriculture. And even though we lived in Ames, there was no connection.

Fairbairn: But he was a teacher, anyway.

Brayton: He actually was a high school teacher.

Fairbairn: So there was some value of education, or recognition of the value of it.

Brayton: Yeah, I guess so. Actually, the biggest influence was in the neighborhood that I lived in Ames. There were a lot of professors living there, and children of the professors. And that probably swayed me a little bit.

Fairbairn: University town.

Brayton: University town.

Fairbairn: So you went to high school there. When did you discover your interest in technology, mathematics, computers? What was it like in high school? Did you think about going to-- was it Iowa State.

Brayton: No. Well, I didn't think about it. But most of my friends went there. And it was so easy to go to, down the street.

Fairbairn: I'm sorry, was it University of Iowa, or Iowa--

Brayton: Iowa State University.

Fairbairn: So that was sort of a foregone conclusion, that if you were going--

Brayton: I didn't even think about it. I didn't take any SATs. I didn't apply anyplace else. And I think in August, I sent in a letter saying, I want to attend. And that was it.

[LAUGHTER]

There was no pressure in those days.

Fairbairn: No entry hurdles to get through. In high school, did you know what you--

Brayton: No.

Fairbairn: Did you develop an interest in any subject?

Brayton: Basketball.

Fairbairn: Basketball. So you're a tall guy.

Brayton: Well, back then. I was 6 foot. And I think we had a center on our basketball team that was 6' 2". That was my main interest.

Fairbairn: So you were fully qualified as a basketball player at your height.

Brayton: Yeah. Yeah, but I think the thing about basketball was the Iowa State gymnasium was right down the street. So every Saturday we could go in and get some scratch games up. We played every Saturday. So that was my main passion in high school.

Fairbairn: Now, did you have siblings?

Brayton: Yeah. One brother older, and one brother younger. And they both went into the ag business.

Fairbairn: OK. So you went to Iowa State. Tell me about your time there and how you got steered in the direction of computing and mathematics.

Brayton: Yeah. Well, in high school I got B's and C's in everything else, but in math I got A's. So when I started at university, I thought, well, I want to be an engineer. I didn't want to be a mathematician, per se. I wanted to earn some money.

So I said, OK, I'll do engineering. And engineering was a big thing at Iowa State, because it was originally a tech school. And they had a good engineering program. So I started in aeronautical engineering--

Fairbairn: It was big at that time.

Brayton: --and learned that it wasn't going to be as interesting as it sounded. So I switched to mechanical engineering.

Fairbairn: This is the early '50s.

Brayton: I started in '51, so '51, '52. I was switching around the engineering disciplines. And then I took my first electrical engineering course, which was eye-opening, the fact that you can write down Kirchoff's laws and equations and then solve them and so forth. It was very impressive to me. It was very mathematical, which I liked.

So I took that as a double major [electrical and mechanical] for a while. So as a double major, I had to take five years. But in my fifth year, I decided I wanted to do more math and physics. So I dropped the ME and just did math and physics and electrical.

Fairbairn: So in your electrical engineering work, was it mainly equipment, motors and generators? Was it tubes?

Brayton: Well, transistors were just coming in. About '53, they started teaching about transistors.

Fairbairn: Wow, that's pretty early.

Brayton: Yeah. When were transistors invented?

Fairbairn: '48, '49.

Brayton: Yeah, so it began getting in the textbooks around '53, '54. And so that was great and eye-opening. And at the beginning, I was kind of leery about getting into electrical engineering, because my view of it was you had to know how to solder. And you had all these ham radio guys who knew much more than I did.

[LAUGHTER]

But then I got into the circuits course, and I found that was great fun. So that was good. But we did have motors and generators and a big lab with huge machines in it for generating things. I don't think they're there anymore.

Fairbairn: Now, did you have any professors that were really mentors that steered you in one direction or another?

Brayton: Not so much. I had good professors and everything. I remember one professor who was a mechanical engineering professor. And I don't think he was a mentor, but he was such a good professor. His name was Harold Black, I think. They have a building named after him now. But I took thermodynamics from him, and that was great.

Fairbairn: If they have a building named after him, he must have impressed somebody more than you.

Brayton: Yeah. So thermodynamics. And actually, if you go back into thermodynamics, there's something called the Brayton heat cycle. And that's like a 10th cousin, five times removed.

Fairbairn: Some other Brayton.

Brayton: Well, same tree. But it branched off about 10 generations ago.

Fairbairn: You had some of that in your blood, anyway.

Brayton: Yeah, something. My background at Iowa State was getting exposed to all these engineering disciplines. So it wasn't a waste of time to take mechanical and aeronautical.

Fairbairn: So at that time, what did you envision as far as a job or career? Did you assume you would just go off and work in one of these engineering fields?

Brayton: At some point in time, I thought, well, maybe I'll go to graduate school. But I was in ROTC, so I had to go in the army for six months. Fortunately, at the time the Korean War was ending, so ROTC obligation was cut from two years to six months, and then seven years reserve.

So I went to work for Sperry Rand UNIVAC. And I got using computers there.

Fairbairn: You mean after your ROTC commitment?

Brayton: No, before. I was at Sperry for about six months. And then ROTC army for another six months. And then in the middle of that, I started applying to graduate schools and got admitted with money at MIT. So I chose MIT.

Fairbairn: So at Sperry, were they building computers at this time?

Brayton: Yeah, I was on a team that was building an onboard computer for an ICBM. And our little group was designing the memory. So the memories back then were these iron cores.

Fairbairn: Core memory.

Brayton: Core memory, yeah. And they had the rows and columns. And then they had the sense wire going through it. We had to build a good amplifier for the sense wire that would sense the switching of the magnetic positions in the cores. And so that was good.

Fairbairn: And you're using transistors at this point?

Brayton: Yes, definitely. Yeah. And fortunately, back then, when you designed something and you wanted to get it right, instead of simulating like we do nowadays, we would build it on a board, and then we'd put our finger on something to add a little capacity here and there and so forth to see what we got.

Fairbairn: Fine-tune it.

[LAUGHTER]

So do you remember where those transistors came from? There weren't very many companies building transistors back then.

Brayton: I don't know. They came from the supplier, as far as I was concerned.

Fairbairn: All right. So you had some introduction, at least, to electronic design and so forth. When was your first introduction to computers?

Brayton: Well, let's see. At Iowa State, we didn't have any. But I began to get interested in them, I guess, because they were just coming out and being used. And so when I took the job at Sperry, they had computers, because they made the UNIVAC computer back then.

I was trying to think last night, what were we computing on? Because I remember writing some programs and putting them on punched paper tape and then running them through the computer, and it would do

something. But I don't know what we were programming at that time. Can't remember. I don't think we were simulating circuits.

Fairbairn: So downstairs in the museum, we have I don't know how much of a piece of the Sperry computer. Because at UNIVAC, it was used in the prediction of the outcome of the 1952 election.

Brayton: Oh. Yeah, well, that's probably one of the ones, or maybe a generation later.

Fairbairn: Yeah, so we ought to take a look down there and see.

Brayton: In '50-- I think it was '56 I started the six-month duty for Sperry. And I remember these big rolls of paper tapes.

Fairbairn: It might have been '56, but that would have been Eisenhower's second. We'll have to take a look. So is there any memory you have of how you chose MIT, other than they gave you money? What were the other schools you applied to or were considering?

Brayton: I wanted to go to Boston, so I applied to Harvard and MIT. And Harvard admitted me, but they weren't giving me any money, at least at the admission. So I went to MIT. And I had a background as an engineer.

Now, the difference was that I had decided that what was limiting what I learned about was my background in mathematics. And they only had the engineering mathematics. So I applied to MIT and Harvard in mathematics.

So I got admitted to the math department. Of course, they didn't have any computer science departments back then. But even at that time, I was interested in computers. But I just enrolled in general mathematics.

Fairbairn: So in looking at your future work, it looks like that played a critical role in the work that came later, as it was heavily mathematics based.

Brayton: Yeah it was a good choice, because I had the EE background enough to master what was going on there. So if you look at my work over the years, it's been kind of mathematical based, a little bit, although applied to computer or electrical things.

Fairbairn: So you entered MIT in the '56, '57?

Brayton: '57, fall of '57, in the math department. And I was coming in as an engineering undergraduate and starting math, and so I was a little bit behind in the math. So the first year I was there at graduate school, I was basically trying to catch up with all the senior mathematics that was taken by the math graduates.

Fairbairn: That was the same time Sputnik was launched, right, when you were--

Brayton: Well, Sputnik, when did they come in? I think it was more like '59 or '60.

Fairbairn: No, it was '57.

Brayton: You might be right, actually.

Fairbairn: Anyway, so I was wondering if there was any particular memory you had of that. So you entered the math department. And you had graduated with a double major, and then were going to pursue a PhD, I presume. That was your goal. And a PhD in mathematics, was that the path you were on?

Brayton: Yeah. Well, I came in, and I guess I told them that I was interested in computers and computing and all that. And so at the time, there were people in the math department that were in logic and recursive function theory.

Well, let's see. Oh, a guy named Hartley Rogers, which was in logic and so forth. So I came in there and was interested in that. And the research assistantship they offered me was partly EE and partly mathematics. And it was in the group that was run by John McCarthy. And-- I forget his name now. Mike, Michael Minsky.

Fairbairn: Oh, Mike Minsky, yeah.

Brayton: Marvin, Marvin Minsky.

Fairbairn: Marvin Minsky.

Brayton: And so I was in their research group. And there were about seven or eight of us graduate students. And McCarthy and Minsky and a few were senior guys. And then the year after, I guess the fall of '58, McCarthy had come back and had this idea about the Lisp programming language. So our group started working on that and implementing it. And so for the next three years, that was my research.

Fairbairn: Is that what you did your PhD on?

Brayton: No.

Fairbairn: That was just your research work leading up to that.

Brayton: This was called AI [Artificial Intelligence] back then.

Fairbairn: Yeah, I actually worked in the Stanford AI lab as an undergraduate, just doing technician work and so forth that McCarthy was running after he'd come out to Stanford. Anyway, so you were--?

Brayton: So I worked on that. But I was in the math department. And I took some courses in math that I got very interested in. And I thought well, AI doesn't have much of a future, I don't want to put my career there. So I just did analysis, and I worked for a professor, Norman Levinson, who was a student of Norbert Wiener. You know Wiener?

Fairbairn: I don't.

Brayton: He's famous around MIT. There's a whole story. But anyway, that was in pure mathematics analysis, basically. Differential equations was what I specialized in. So I did that as a thesis, but my research assistantship was in the Lisp.

Fairbairn: I see. So you got a little bit of both. You got into the computing world and the--

Brayton: Yeah, and it was great fun. And my job in Lisp was to write their compiler.

Fairbairn: And on what machine was that being done?

Brayton: That was an IBM-- let's see. It was before 704, I think, so I'm not sure what they had before that. But I remember it was in a big--

Fairbairn: Computer room.

Brayton: Computer room. Cool, air conditioned and everything. And you'd bring your cards and card deck in, put it there, and wait overnight and come back, and you'd read the results.

Fairbairn: Find out where your bugs were.

Brayton: Where you forgot a comma here. And then you'd go off and do it again. So actually, it trained you to be very careful about writing your program and double-checking and everything, because you didn't want to spend night after night after night fixing stupid things.

Yeah, so that was a great job. Great fun. And a lot of the work I've done since I got out had to do-- you could say it's had to do with compiling something. And so this is the first taste of that kind of--

Fairbairn: Interesting. So any special memories of your time at MIT? People that had the most impact, or aha moments, or something that steered you in a particular direction-- is there anything?

Brayton: No, it was just a good time.

Fairbairn: Just a good time, you learned a lot.

Brayton: I enjoyed all the courses. And MIT was a great place. There was Norbert Wiener. Oh, there was also a young professor there who was-- you know this book, A Beautiful Mind? What's his name?

Fairbairn: Yeah, I don't remember the name.

Brayton: John Nash. He was a young instructor there.

Fairbairn: No kidding.

Brayton: And we would see him around the hallway. I never even talked to him. But the rumor was, he's a pretty bright guy. And then he just suddenly disappeared. And of course, you know the story of him. He was just a great mind. He got a Nobel Prize in Economics, but he solved two or three major mathematical problems that nobody else had and that had been around for awhile.

But anyway, he was there, I remember. After I read that book I knew exactly who that was.

Fairbairn: Oh, I know that person. Interesting. So came time you're wrapping things up. When did you finish up at MIT?

Brayton: '61. So in summer of '60, I was able to get a summer job at IBM. And one of my professors at MIT, name was Jurgen Moser, was consulting at IBM in the summer, in Yorktown, New York. And so partly I got that summer job-- I guess I got the summer job separately. But he happened to be there. And so we worked on something together that summer.

And then when I graduated, I knew that was a good place, so I applied there. I got a job at Yorktown, IBM research at Yorktown.

Fairbairn: Is that because you were familiar with the place? Did you have a clear idea about what you wanted to pursue in terms of a career at that point?

Brayton: Yeah, again, it's mathematics and computers. IBM is a computer company. But I did see-- I went into their mathematics department, both in the summer and later. And I saw that they gave you really complete freedom almost at that time to do whatever you wanted to do in research. You picked out your areas.

So it was a great combination. I knew the environment and everything. So it was just the obvious place to go to next.

Fairbairn: So what did you dive into? You started there in '61, '62?

Brayton: Well, I was in the math department. And so they said, do good mathematics.

Fairbairn: [LAUGHS] Do good work.

Brayton: Do good work. And so I finished up work that I'd started with Jurgen Moser one summer. And we put it together, polished it, made a big paper called "A Theory of Nonlinear Networks." And it was published in the journal Quarterly of Applied Mathematics, I think in two separate volumes. It was a big, long paper.

So I did that. That had to do with writing the equations for electrical circuits in a different form and proving things like stability of the transient solutions and things like that one. So it was a pretty major piece of work. And then I continued looking at differential equations of electrical circuits and writing, doing papers and research on behavior of circuits,

Fairbairn: So was there a particular sort of problems that engineers or whatever were facing for which these were solutions? Were these helping people just to understand the behavior of circuits better? How did you get linked into--

Brayton: Well, in the beginning, there was no particular direct application to anything. It was just these were interesting problems. And the fact that you could rewrite the circuit equations-- these are the transient equations-- in this form was completely new and useful for proving properties. But I don't know that anybody had a problem where they wanted to prove stability or something like that directly. But it was more a mathematical thing.

But that activity led into the next phase.

Fairbairn: You can do formal verification before-- at a circuit level, right?

Brayton: That was the simulation, yeah.

Fairbairn: You could verify that it was stable or not stable?

Brayton: Well, you could simulate it. Even back then, the simulators were not very good. The state of being able to simulate anything was just pretty bad. And I remember around '65, '4, '5, or '6, there were problems where people at IBM could not solve a three transistor circuit, because it had a special property that people didn't understand.

And so the state of simulators back then was just pretty primitive. They could do linear circuits, but nonlinear was a problem. So that got into the next phase.

Fairbairn: Yeah, so after this first activity, what was the next? And when did you get tied into some practical--

Brayton: Well, I think about '64, we got involved in solving linear equations, and those linear equations came from circuits. And so the first thing was to be able to solve very, very large systems of linear

equations. But they were very sparse. They were coming from circuits, so the connectivity was just kind of locally at the nodes of them.

So this was when we did some initial work on solving sparse matrices, linear systems of equations. And that led us into simulating electrical circuits. And so at around '66, we started putting together a new program, which would-- sort of like SPICE, but we were several years earlier than that. It became a program called ASTAP, which was a circuit simulator, and was used for maybe 30 years [in IBM].

Fairbairn: So did somebody come to you or your department with this problem? Did you go out discover it? Is it just things you stumbled upon as you were doing interesting mathematical things?

Brayton: I remember talking to a group in Poughkeepsie. And they had this problem of simulating a small circuit. And they came to us and said, what's going on here? We can't do it. And so we analyzed it and found out that there were eigenvalues of the problem that were large or small, and so the normal numerical methods didn't work on those things.

I was, at that time, a manager. And I had about three other people. One of them, Warner Linear, was an expert in numerical methods for solving differential equations. And so our group got into analyzing what was wrong and told them what to do. And they did it, and that got us into this field after that. So you start small, and then you take on more and more and more.

Fairbairn: So did your group actually build this circuit simulator? Or did others take the ideas and fundamental algorithms and build it?

Brayton: We consulted with a group in Poughkeepsie that actually took it and made the program. But they made the program on our architecture. So we told them how to solve sparse matrices, what numerical methods should be seen, how to formulate the equations for computer solving and things like that, and mapped it out. The success of that program was very much due to that group being able to go out into IBM and holding hands as people tried to use it.

Fairbairn: Whole new different kind of tools than they were used to.

Brayton: Yeah, completely new kind of tool, how to use it. Of course, it didn't work exactly right. So you had to modify things and tune it a little bit. But that thing was a success in IBM, because of this other group. The guy named Al Jimenez, who I remember, was just an enthusiast. And he would program this thing and then go out to the users and hold their hands and guide them through using it. In the end, everybody was using it at IBM. And it was very, very successful.

Fairbairn: So here, you're, you say '67 or so? Maybe five years into IBM?

Brayton: We started that kind of stuff about five years in. And the development continued through the early '70s. We just kept working on it, and there was more. It opened up a box of interesting things to do. And we developed other things along those lines.

Fairbairn: So this became known within IBM. Did it get published?

Brayton: Yeah.

Fairbairn: Did that work then get adapted quickly outside?

Brayton: Yeah, we published. There was no slowing down. We could publish right away, and so we did. SPICE came on maybe a year after we had published. So we had already published the sparse matrix stuff and the numerical methods and the equation formulation.

So I always think if I hadn't been at IBM and was giving it away, it might have been ASTAP instead of SPICE.

Fairbairn: So your ideas were incorporated into the initial versions of SPICE. Is that right?

Brayton: Well, people were publishing about sparse matrices. And so I don't know who got what ideas. We had our own ideas. And we developed the things and published. And I'm not sure what the timeframe was between the development of SPICE. Maybe they were doing the same thing at a similar time.

Fairbairn: Yeah, we had a 40th anniversary celebration of SPICE here at the museum, when there was an IEEE plaque created as a major milestone, IEEE milestone. That must have been last year or the year before. So it was '72, '73 timeframe when SPICE was first released, I guess.

So how long was this simulation a major focus for your work? And what was the transition to your next major project?

Brayton: Well, that's interesting. So while this was going on, I became second-level manager. So I had a bunch of groups under me. And one of the people [Bob Risch] working, quote, "for me--" but these were PhDs, and they all went off and did their own thing. He was traveling around the company and looking at different groups. And he was more of a pure mathematician, but he wanted to do something useful.

And so he found a group in Los Gatos. IBM had a lab at one time in Los Gatos, a very nice little lab. And he found a group that was trying to build a computer out of some exotic cells, exotic circuits, so the circuits would be able to implement a larger logic function than the normal "and" and "or", things like that. So you'd put a lot of logic function into one of these circuits. It was called a cascoded emitter coupled logic circuit [CECL], where you had an ECLcircuit on top of another one, so it's cascoded. And you could implement logic functions of six, seven, eight variables or inputs coming in.

And so at the time we started talking to them, they had basically built a circuit, a computer from these by hand. No computer--

Fairbairn: No simulation or anything like that?

Brayton: There was no logic synthesis at all, in fact. So he [Bob] looked at their circuit. And he came up with an algorithm that would decide when a logic function could be implemented in one of these circuits. And so as his manager, or second-level manager, I had to understand what he was doing.

And so I listened to him. And I said, well, I'll just program that for you, because he couldn't program. And so I became his programmer.

Fairbairn: Reversal of roles.

Brayton: Yeah. And at the time, we were using a language called APL so we could program things pretty fast as a kind of prototype code. I started doing that, and that got me into where you had this problem of given a logic function, can it be mapped into something? How do you tell if a circuit of a certain type can implement that [function]?

Fairbairn: And what year was this, then?

Brayton: This was middle '70s, I think.

Fairbairn: So did this also get you out to California, to Los Gatos?

Brayton: Once. Once or twice. So the program I wrote took in a logic equation and then analyzed it and said whether it [CECL] could do it or not. And so this notion of taking in a logic function and then doing something with it became the start of something that grew bigger.

So at the beginning, I had a little program that would take up to eight variables. And they were called A, B, C, D, E, F, G. And it was written in APL. And then we said, oh, we need more variables, because things were getting bigger and so forth. And so then we got up to 26 variables, A to Z. And then it expanded more.

And so as we were creating this capability, other groups in IBM were thinking about their own exotic circuit and how it could be faster and smaller if we could pack more logic functionality into a single circuit, rather than breaking them into smaller ones. And so we started talking to Boca Raton and Poughkeepsie, Fishkill, and so forth [Burlington], different groups that were experimenting with how to make computers faster and smaller and so forth.

And so this capability which became 26 variables now blew up into many variables. We wanted to do a whole computer. And so this became what I called the Yorktown Logic Editor. And it was our first step into logic synthesis. And so that's how we got into it.

Fairbairn: Was there a milestone? Did you take these concepts, develop a program, which then became widely used within IBM?

Brayton: No, it was more of a demonstration. And in fact, there was another group in IBM led by John Derringer. They also started doing logic synthesis. And they were more oriented than we were to interacting with the mainframe guys. And so they had their own logic synthesis program. They were doing it in a different way, so we had a different way of doing things.

But they made more of an impact than we did inside IBM, in terms of actually getting logic synthesis going. Well, we published everything. We made more of an impact on the outside.

Fairbairn: Is that right? [LAUGHS] So they were on a path in parallel with your work? Is that right? You published.

Brayton: We had our own--

Fairbairn: Was there a rivalry?

Brayton: There was a little rivalry, I think. My feeling was that they were more oriented in a peep-hole optimization. You have a sea-of-gates. And then you look at a certain area. And you try to make a transformation. And you walk around and do the peep-hole optimization.

And we had divided our thing into a-- the peep-hole optimization depended upon the standard cell gates that you were using at the time. And so you'd have a sea of logic gates, standard cells. And they are connected up in a certain way. And then you would make these transformations on a set of gates, and then move it around.

We took a different point of view. And we said, let's do some stuff first that was technology independent. So we don't know what the standards cells are that we're going to use, or where it's going to go. We just get a better compact view of the logic as logic equations, and then we would then map it into whatever standard cells that were provided.

And we were motivated that way because we didn't have one standard cell. We were working with these various groups. And they were cascoded emitter coupled logic, or cascode voltage logic and so forth. So we did the technology independent optimations, and then mapping later. We had a different approach.

Fairbairn: I did an oral history with John Darringer couple years ago. So you can go back, and it's posted online. You can see what he has to say on his side of it. I don't remember exactly what his comments were on the particular topic.

So what was the reception to that? And you said his work got more play within IBM. Describe the interaction you might have had with people outside. Did people start calling you up and wanting to learn more?

Brayton: Yeah, the people that we talked to had these more exotic circuits. So we had a lot of interaction with them. But we didn't get into the mainframe groups and the main bread and butter of IBM. We were acting more like a research group, trying to invent new ways of doing things, and then building algorithms and software that would implement these ideas.

As that effort got started, then we saw that it was building up to an interesting, broader area-- dealing with logic, how you manipulate logic and so forth. So we had what I call the summer of logic. There were four of us who were working together on this at the time. Alberto was at IBM for a year with us, and Gary Hachtel and Bob Risch and myself.

And we decided we'll organize the whole thing. And we will divide our group into four parts. And we would each have a graduate student come in and work with each one of the major guys. And we would parcel out the jobs.

So Alberto had the two-level logic minimization job. Gary had the general logic manipulation algorithms. I had the multi-level logic. And Bob Risch had decomposing logic for delay.

Fairbairn: And what year was this? Summer of--?

Brayton: I think it was '82. I'd have to look back. '81 or '82, something like that. So we had this grand plan for the summer. And we executed it. It was amazingly successful. Lots of new ideas came out. And of course, we were all working in logic, so we were sharing logic across all four areas.

And ultimately, one of the programs that came out was called Espresso, which was a two-level logic minimization. And we wrote a little book on it that described all the algorithms and detail and mathematical proofs and things like that.

Fairbairn: So the Design Automation Conference was going on at this time. Did you attend that? Was that one of the outlets for your work?

Brayton: I hadn't been attending that. And I forget the first time I went to one of those. It was around that time, though. I know we had taken the paper-- we quickly wrote up something on two-level logic minimization, sort of the first Espresso paper, sent it to Design Automation. And it got rejected.

So we didn't write it very well. That paper had a lot of good ideas in it. It just took another year to polish it up.

Fairbairn: And then it was published?

Brayton: Yeah, it I was published, and actually became a little book, in fact, too.

Fairbairn: So I'm familiar with the name Espresso. That became a commercial program? Or did that go out through Berkeley or something?

Brayton: We had a version in IBM. It was an APL version that came out of that summer. But that version became a kind of road map for the next version, which was done at Berkeley, written in C, and was distributed freely all over the world.

Fairbairn: I see. All right. So we're now '82 time frame. So your initial work focused around circuit simulation. And then you moved to logic synthesis as the major area-- a lot of peripheral things, I presume, along the way.

Brayton: So circuit simulation was continuous variables. And logic was discrete variables. And so it was a major shift there, but all dealing with circuits and computers and things.

Fairbairn: So you were at IBM until-- what, you started in '61, '62, and you were there for 20?

Brayton: Actually, 26 years.

Fairbairn: 26 years. So you came out '80?

Brayton: Seven.

Fairbairn: '87. OK, so we're in '82. So you have another five years at IBM. Tell me.

Brayton: Yeah, good question. So things kept building up. And so we added capability to our little logic editor. Capability was layout and placement.

So during that time, the guy who was doing layout and placement was Ralph Otten. And he was from the Netherlands. And he came over and worked for us for about six years. And his expertise was in layout. So we added some layout capability at the lower end, some timing optimization at the lower end.

So that was when Giovanni De Micheli came into the group.

Fairbairn: He came into the group?

Brayton: Well, he was not in our group. But he basically came into IBM research and was working there for-- I don't know how long. But basically, when he was there, he worked with us. And he'd also been a summer student. So he was a summer student at IBM as well.

Fairbairn: So was any of this work getting out to IBM designers?

Brayton: No.

Fairbairn: Or was this mainly all people outside of IBM were taking advantage of it?

Brayton: Well, we were publishing and we built up to finally something we called the Yorktown Silicon Compiler.

Fairbairn: OK. By combining the place and route stuff, and the logic synthesis?

Brayton: Place and route. We added a language. We added some high-level synthesis. So we had high-level synthesis, where you could describe things at a higher level. And the guy who did that was Raul Camposano.

So we had that language, logic synthesis, timing and layout. So we could start at the top. And we started a project where we'd take the-- IBM at the time had a RISC computer design.

Fairbairn: Is this the 801?

Brayton: 801. And so we said, we're going to compile that.

Fairbairn: So yesterday I was talking to one of the designers of the 801.

Brayton: Who?

Fairbairn: Rich Freitas.

Brayton: I know the name.

Fairbairn: He worked on the IO controller. He was a young kid at the time. But he's taken on the task of gathering some of the other designers and project manager to do a joint oral history on the design of the 801.

Brayton: That's interesting. I didn't know they were doing that.

Fairbairn: Yeah, well, that was just yesterday. We've been working around that. And finally we got him in here. He had a list of names of people he got somewhere that he's going try to track down. Several of them are here on the West Coast, because they migrated out here to HP or whatever.

And a couple of them are still on the East Coast. So he was going to see which ones he could entice to come together and do a joint oral history. So anyway, that's to come. And we'll let you know if that happens.

Brayton: So that's interesting, because when we're doing that, we decided to do the 801. Well, we had to figure out what the 801 was. And we had to sign out to check out these documents, which were super secret inside IBM. So we got the manual for that and read through it and figured out what things we had to implement, and then do it.

So we had to put in the information. Describe the 801. And then press the button. And it would compile. And we actually did it.

Fairbairn: And the target technology was like some IBM gate array technology?

Brayton: So we had our own target technology, which was I think a differential cascode voltage switch [DCVS], which again was one of these exotic circuits that you can implement a lot of functionality in a small area. And so part of the effort was to say, OK, now we have this function. We know how we can implement it in a single device. Then we had to lay out the device and figure out the actual layout of the polysilicon and so forth.

And Ralph Otten did all that. He figured out how to take the logic equation and lay it out and make a device that would implement that logic piece. And so that was part of that flow, too. So we would on-the-fly create the logic cell that would be used as we-- and I think we tiled in those. But it was not a gate array, it was more of a--

Fairbairn: Standard cell?

Brayton: Standard cell. But the cells we got were created on the fly as we did the logic synthesis.

Fairbairn: OK. So were you actually able to generate a full 801, or the CPU portion?

Brayton: We drew a picture.

Fairbairn: You drew a picture.

Brayton: We never used it to make anything and try it out. But we demonstrated--

Fairbairn: But you can handle that complexity.

Brayton: Yes.

Fairbairn: How many gates were we talking about?

Brayton: You know, I can't remember. You have to ask Ralph Otten. Because actually, in the middle that, I had gone to Berkeley for a year on a sabbatical. And I came back and then we finished it up.

Fairbairn: So this was in 1985.

Brayton: In '85 I went to Berkeley, came back for one year, '86. And then I went to Berkeley. So it was reasonable complexity at that time.

Fairbairn: Absolutely. For that time, being able to do that kind of thing is--

Brayton: And we did write it up. And it became a chapter in a book by edited by Gajsky. There was a whole bunch of things. There's one chapter called "The Yorktown Silicon Compiler." and I think there's a picture of our 801.

Fairbairn: Wow, that really was a pioneering piece of work at that time. So it sounds like you'd already-- Alberto had come to Yorktown there. And you then went and spent a year at Berkeley. The Yorktown Silicon Compiler was the next major-- is there anything else that you want to highlight before we talk about the transition to Berkeley and how that came about?

Brayton: No, I think we've pretty well covered that.

Fairbairn: OK. So in the midst of this, in this work on your silicon compiler, and in fact, logic synthesis before that, you had established some contact with Berkeley. How was it that Alberto Sangiovanni-Vincentelli came back to Yorktown, and you then later went to Berkeley for a year?

Brayton: OK. I can't really remember exactly when I first met Alberto, but it was a number of years before that. And at one time, we had an interchange between Berkeley and IBM research.

And we would have a meeting every year. And we would organize it like a little mini-symposium. And people would give papers and things like that. So that was going on for a while. So there was always this interaction.

And I forget-- I think Alberto came at the invitation of Gary Hachtel who was also one of the major people in this effort. Also, he was a major person in the ASTAP development. He worked in my group for maybe 15, 20 years. So he was one of my major, major collaborators over the years.

Anyway, I think he knew Alberto even before I did. Actually, Gary got his PhD from Berkeley under [Don] Peterson, so there was a connection there. And I can't remember exactly when Gary and Alberto knew each other.

But anyway, Gary invited him to IBM. And when he got there, he was going to be there for a year. And we were doing logic stuff. So we forced him to have to program in APL. [LAUGHTER]

And he learned it fine. And so we were all doing that. All that work was done completely in APL, even the layout.

Fairbairn: So what then led to your year in Berkeley? How did that come about?

Brayton: I guess just because of our connection with Alberto. And at the time, IBM research had this policy, almost like a university, of every seven years, you could have a sabbatical. And some people took advantage of it.

And I had one at MIT in the '60s. And I had another at Imperial College in London. And then Berkeley. So doing sabbaticals, and Berkeley was the natural place to go.

Fairbairn: Now, on these sabbaticals previous to Berkeley, did you do teaching at these? You went to a couple of other universities, it sounds like.

Brayton: At MIT, I did teach. This was in the '60s. So I taught a-- what did I teach?

Fairbairn: A long time ago.

Brayton: A long time ago. I taught a course that Roger Brockett had created on linear control theory, I think it was.

Fairbairn: Is that something that appealed to you at the time? You thought, could I imagine being a university professor? Or I'm going to do this job and go back to my nice job at IBM?

Brayton: It didn't occur that I wanted to do it. I did the teaching just because that was part of the sabbatical. But it didn't thrill me so much. And when I was there and I came back and went to other places, I never thought I would want to teach.

Fairbairn: Then you want to the Imperial College in London, you say.

Brayton: Imperial College.

Fairbairn: You taught there, as well?

Brayton: No, I didn't teach there. I was in a research group. And I interacted with a professor there [Bob Spence]. And we got together and wrote a book called Sensitivity and Optimization. So this was for circuits. So you compute how sensitive the circuit is, behavior to variation of parameters, like a gradient.

So he did half the book, I did half the book. And then that finally came out. That was the first book I actually wrote. So while I was at Imperial College, we worked together, basically, on this book.

Fairbairn: So did either of these-- I guess the first one, MIT, was much earlier. But did these sabbaticals serve to spread some of the ideas that you had been developing at IBM? Did you learn which way did the primary information transfer?

Brayton: Well, like anything, I guess both ways. So I didn't go there so much to teach courses. I went there to do research in a group and as part of a group. I knew things, they knew things. And so we exchanged ideas.

When I went to Berkeley, of course we had a whole agenda at the time on logic synthesis. We knew what we wanted to try to get done and so forth. And so when I went there, there was a big effort in '85-- Alberto [Sangiovanni-Vincentelli] and myself, Richard [Newton], and a bunch of students, maybe eight students, who started doing this stuff. And we had marvelously smart graduate students. You know Rick Rudell, did you know him? Or Albert Wang? These guys were working on this stuff in '85.

So that was eye-opening, because I could see the leverage you get out of having a good set of graduate students and working together and having ideas and getting them implemented and things like that.

Fairbairn: So what year were you at Berkeley?

Brayton: '85.

Fairbairn: '85, '86?

Brayton: '85, '86.

Fairbairn: So somewhere in here, speaking of logic synthesis, the company Synopsys got started. Some of the people you described ended up being part of that.

Brayton: So Aart [de Geus] started a company called-- something in North Carolina called Optimist or Optimal Solutions.

Fairbairn: Optimal Solutions, yeah.

Brayton: And I do remember going to a conference there. We had a conference at the-- there was a bunch of conferences that were-- it was in logic, so it might have been the early [IWLS, International Workshop on Logic Synthesis. That was there.

Anyway, Optimal Solutions was there. They had a little closet [of an office] where it seemed like there were four or five people there. And they were doing logic synthesis, so we visited them.

Fairbairn: Now, had they taken some of their ideas from the previous work you had done?

Brayton: No. Aart came in and had worked at GE, I think.

Fairbairn: GE, right.

Brayton: And then apparently they [GE] weren't interested in logic synthesis, and so he took ideas that had come from there and started working. It's a completely different set of ideas. And then at one point, they decided to move to the Valley, because of the connections around here and everything.

So they moved. And then when these graduate students-- at the time I got to Berkeley, we were working on a system called MIS, M-I-S. It didn't even stand for anything-- interactive synthesis or something.

So Rick and Albert put together MIS. And they programmed it in C. And it was pretty successful. And then they were hired by this company, which became Synopsys.

Fairbairn: Synopsys, right.

Brayton: I think when they [Synopsys] moved, they may have changed their name. But I remember after that meeting-- no, actually, the year before I remember meeting Aart at one of these meetings that was also at North Carolina. And he was in the airport waiting for an airplane. And so I talked to him.

And he said he was thinking about starting this company. And my advice was, don't do it. Don't do it, because you'll never get back to research. You start this company, you--

[LAUGHTER]

Fairbairn: That might be good.

Brayton: I don't know if he remembers that, but I do. My sage advice.

Fairbairn: So did Synopsys ever incorporate the technology which you had created then?

Brayton: Yes. So the MIS program was up and operational. And the guys who actually put it together and implemented everything and had many new, good ideas were Rick and Albert Wang. And they went and were one of the early people who joined Synopsys.

Fairbairn: Now, one of the other names you mentioned was Raul Camposano, who also ended up at Synopsys.

Brayton: Right. But much later.

Fairbairn: All right. So you had your year at Berkeley. And did the seeds start germinating about moving to Berkeley? Tell me about that process and how you came to a decision.

Brayton: Not at the time. It was a great experience and so forth. And I guess I learned what it would be like to be at the university, and especially working in the group that was there at the time. So there's a major EDA set of people who were doing EDA. And there was Don Petersen, Alberto, Richard.

Fairbairn: Alberto. Yeah. Right.

Brayton: So anyway, that was great. But I didn't think of leaving IBM. And then I got back '86. And in the year '86, '87, IBM started offering early retirement broadly across the company. And they gave some additional incentives. If you went to a university, they would do other things and so forth for you. So there was an incentive to actually leave.

And so I'd had the experience at Berkeley, and I thought, well, OK, I can handle that. But I didn't know where I would be able to get a job offer. And so I interviewed a whole bunch of places. And it was fun interviewing. But in the back of my mind, it was Berkeley I wanted to go to. But a job offer was not guaranteed. Not only that, but my wife's from the East Coast.

Fairbairn: That was my next question. How did your family-- you have children by this time, I presume.

Brayton: The children were already in college.

Fairbairn: They were already off, OK.

Brayton: The first two had already graduated college.

Fairbairn: So they all were born and grew up during your IBM career

Brayton: We were an empty nest. But my wife had the attachments to the East Coast, and so going way out here-- family and things like that. I wanted to go to Berkeley, but it wasn't sure.

So I got an offer from MIT and Princeton and Penn. And of course, my wife's from Boston. She wanted me to take the MIT offer. And probably I would have done that, but when I went to MIT that spring, I think it was maybe March or something, to give a talk, they had this major blizzard. And they almost closed the Institute there. And so that helped to persuade-- yeah.

Fairbairn: You must have been used to that. You were working in New York, and then coming from Iowa.

Brayton: And then we'd been out to California for a year or so.

Fairbairn: Oh, yeah. You had that for a year.

Brayton: We had that experience.

Fairbairn: Said, oh, we don't have blizzards here.

Brayton: So anyway, I got the offer from Berkeley, and then we came out in '87, summer of '87. So it was a big transition for me and also for Ruth.

Fairbairn: So you came out. You actually end up retiring from IBM after 26 years. Then you started a new career. Similar field and so forth.

Brayton: Well, I told my wife I was retiring, so I wouldn't work so hard.

[LAUGHTER]

Fairbairn: She bought that, huh?

Brayton: A little bit.

Fairbairn: So what did you think you were coming out to do? Did you come out to be a full professor, to teach? What was the agenda?

Brayton: Well, I was coming out as a full professor. But I had this half-time professorship, half-time research thing. So I didn't have to teach the full amount.

Fairbairn: Full load, yeah.

Brayton: And of course, the research part was contingent upon getting the money and supporting our research. But that wasn't a problem back then. So it was, I say, a major advantage to having a half-time teaching job, because I could continue the research I was doing.

So I came out here. Yeah, the first year was a lot of work. I think because the first year, Alberto says, well, I'm on sabbatical.

Fairbairn: So then did you have to teach one of his courses?

Brayton: I had to teach both his courses. No, I had to teach one of his courses. And then I made up a new course in logic synthesis, which was the first time it had been taught. Yeah, it was a lot of work, because I wasn't used to teaching. And it was a new experience.

Fairbairn: So it must have been a major cultural transition, also. Even though it's a research institution, the whole dynamics of Silicon Valley and startup companies and so forth was quite different from the stable world of IBM, I presume.

Brayton: It was, but back then we were at Berkeley doing our research. So as a research piece, it wasn't that much different. IBM gave you lots of freedom. And Berkeley, lots of freedom. So the difference was you did the teaching, you had more students around, and then figuring out how to make a Xerox copy or something was a little different.

Fairbairn: Do a little more of your own work, huh?

Brayton: Yeah. So getting used to that environment. But it was great.

Fairbairn: So then what did you dive into? What was the research work that you were doing at Berkeley when you first arrived?

Brayton: Well, when I first arrived, it was continue the logics. And there was always a piece here, a piece there and so forth.

Fairbairn: Always work to be done.

Brayton: And so we were doing research and publishing papers. That's how many of those 450 papers got there. We were just publishing all the time new ideas, new pieces, and so forth. And it was a very, very hot topic in those days. So that kept up for the next five years. And graduating lots of PhDs at the time.

Fairbairn: Were you consulting for Synopsys or Cadence?

Brayton: So I did some initial consulting for Synopsys. I gave some lectures on logic synthesis, basically a synopsis of the course I was teaching back then. Went down two or three times a month, something like that, and had some very eager students listening in. And that was that.

So even today, logic synthesis research, we're still doing it.

Fairbairn: Still working on it, huh?

Brayton: Yeah. You can always get better area, better speeds, mapping into different things.

Fairbairn: So when did that at least slow down. And did you then start to investigate other areas?

Brayton: Well, I got into verification.

Fairbairn: OK. So that's the next major step.

Brayton: That's my next and last thing. So I kept doing the logic synthesis. But there was a time when we had a talk at Berkeley by a guy from Bell Labs, Bob Kurshan. And he told us about his work on verification at just one of the lectures that happened during the week sometime.

And I remember coming out of that lecture saying, “this is it”. This is the next thing we should do, verify. And so we got into kind of an arrangement with Kurshan to come to Berkeley every once in a while and help us to do some things.

And at the same time, there were some developments about how do you represent logic on the computer. And there was BDDs. And Randy Bryant had worked on and developed BDDs around '86. They were coming in to the logic area.

Fairbairn: These are binary decision diagrams?

Brayton: Yeah. They were coming in. And so we started investigating them initially for synthesis to represent and manipulate the equations. And then with Kurshan, we saw where we could use those in verification. So we started putting together a verification system.

Fairbairn: Now, what were you trying to verify, and what were you comparing it against?

Brayton: Well, initially, we had some toy examples. Have you heard of the dining philosophers’ problem?

Fairbairn: I think so. I just don't remember the--

Brayton: You see you have a circle of philosophers. I don't know why they're philosophers. And they're dining. I look at it as they have chopsticks, but there was only one per person, and an extra one. So you could only dine when you had two chopsticks.

So then there was some protocol about when you start eating and your neighbor wanted to eat, then you had to pass him a chopstick. And we had to verify that the whole thing could not stall. If ever you wanted to dine, to eat, eventually you could.

This was a good problem, because you could parameterize it to "*n* dining philosophers." So we got it going. And I remember our joy at being able to prove six dining philosophers or something like that. And then we got it up to 12 or something.

And so we were using BDDs, which was a new thing to use in formal verification. And we were using our expertise that we had developed for the synthesis. So it was a natural transition for us. Then we started building systems that would do verification.

Fairbairn: So this is all formal verification?

Brayton: Formal.

Fairbairn: So you could prove certain properties about some logic network or whatever.

Brayton: Or you synthesize here, and you had the original golden model here. Prove they're equivalent.

Fairbairn: Right. Equivalence model, yeah.

Brayton: So that's a formal verification problem. In fact, it's not any easier than normal property verification. But we were motivated by wanting to do synthesis and then proving that we were getting the right answer and not making errors in the software.

Fairbairn: Now, were you consulting? Or were there companies that were trying to adopt this technology for products?

Brayton: Well, this was quite early in the verification, so there weren't any companies at that time.

Fairbairn: Hadn't evolved to a practical--

Brayton: As the capability got better and better, then people began to see that well, maybe this could be commercially useful.

Fairbairn: So this is in-- you arrived in Berkeley in '87.

Brayton: Yeah. So this is, like, from '90 until today that we've been working in verification systems, and seeing the capabilities improving orders and orders of magnitude over what was there before.

Fairbairn: Now, did that improvement come as a result of major breakthroughs in terms of algorithms, approaches? Can you describe a high level where the major steps or improvements were.

Brayton: Well, mainly [it was] through techniques and algorithms. BDDs were the first thing. They were new to the field. And so they just allowed you to do much more than you ever could. So they were the first breakthrough.

And I'd say the second breakthrough came when people started using SAT solvers, satisfiability. So you have a logic circuit. And it had maybe, let's say, a single output. And you wanted to see if there was an input pattern, and there might be thousands of inputs, that would make a one at the output of the circuit. This is called satisfiability. So either you can make a one, or it's never possible.

So SAT solvers started coming in to play in logic synthesis and verification, I guess, at the same time. But I think they're motivated at the-- well, I don't know. I was going to say motivated by the logic synthesis.

There were some people who were doing logic things [who] decided that the current SAT solvers were very limited, and they could do better. And so there was a series of developments that improved the SAT solvers, and then improved them and improved them, so forth. So they just became completely flooding the area with it.

So I'd say the next breakthrough was this capability of this orders of magnitude improvement in SAT solvers. And that actually still continues. And so that was one. And then about three years ago, there was another key breakthrough, which was invented by a guy who worked on it at Stanford in his PhD and then went to Colorado.

Aaron Bradley is his name. And he called it IC3. And we implemented his idea and kept developing it further. We called it PDR. So IC3, PDR, it's the same thing. And this was eye-opening. It was really a brand new way to prove things, Property Directed Reachability, and became almost a paradigm for new developments.

Fairbairn: Now, could you do new kinds of proofs? Or did it just handle much larger circuits? Or what were the major advantages?

Brayton: Yeah. It wouldn't solve all the circuits. But it almost displaced BDDs. BDDs have this problem of not being stable. So you give it a problem, and it just gets bigger and bigger and bigger, the BDD. And so you can't predict that. So it wasn't the most desirable thing, but on some problems, it could solve them.

And then PDR came on and displaced that almost completely. And one time I said to my students, no BDDs and wrote BDDs with a cross [through it]. But in the end, there are still problems that BDDs can do that other methods can't. But SAT solving was one of them.

Well, I should mention one other algorithm [interpolation]. So we had BDDs. And then we had interpolation. And SAT solvers were coming in about the same time. And then PDR. So we have this bunch of tools that have come in.

And so every year, there's what they call Model Checking Competition. Model checking is either property checking or equivalence checking. And there's been one every year [for about 7 or 8 years]. And you can see this progress over the last four or five years with the number of problems they can solve, the size of the problems they can solve, and so forth. It's just been growing maybe by a factor of five or 10.

Fairbairn: Each year.

Brayton: Yeah, each year. So that's been really important going on.

Fairbairn: Now, have there been companies that have spun out of these ideas.

Brayton: Yes.

Fairbairn: What was the--

Brayton: Well, let's see. There's a whole bunch of verification companies and also within the major commercial [CAD companies], like Synopsys has formal verification, Cadence, Mentor Graphics. Then there's individual companies, like Jasper and Real Intent and I should name all of them, but I can't do it right now.

Most of the startups have struggled a little bit, because of the downturn. It was when they started up. I think Jasper is doing pretty well right now.

Fairbairn: Yeah, I was involved in EDA during that period. And there was a lot of talk about formal verification for a long time.

Brayton: Sort of like AI.

Fairbairn: Yeah. When does it become real here?

Brayton: Well, it has become real. And there's big advantages in being able to formally verify, but it's a harder problem to actually use. Because if the problem's too big, you [have] got to cut it down somehow. You have to chop it up or make some assumptions and things like that. And that's an art.

And with synthesis, you got what you got. You pushed a button, you get an area and a delay. And maybe you can massage it to get a little bit more. But you don't have to get the right answer. You have to get an answer that's good enough. So in some sense, it's easier to use.

Fairbairn: Now, you've moved to Berkeley. Did you ever go back and do any collaboration with IBM?

Brayton: Yeah. In fact, in formal verification, we have a very close collaboration with people in IBM. They're located not at Yorktown. They're located Austin. And IBM has its own formal verification tool called SixthSense.

And we have that with those people for many years. Sometimes we have conference calls with them once a month, something like that. And they've been very open sharing information. And we share information. Since IBM doesn't sell a formal verification product, we can actually exchange good ideas. Whereas if you're talking to somebody who's selling a product in that area, it's a little bit more difficult.

Fairbairn: A little bit tighter.

Brayton: And also, IBM can give us examples of problems, whereas commercial companies can't give [away] their customers examples. So it's much harder.

Fairbairn: So you talked about sabbaticals at IBM. Have you taken sabbaticals while at Berkeley? Do they have a similar program, where you've gone off to other universities or research places?

Brayton: Yes. I took two sabbaticals while I was here. One was I went to Paris one time for not a very long time. But at the time I went, there was-- DEC had a research lab in Paris, outside of Paris. And one of my former students was working there.

Fairbairn: You got yourself an invitation.

Brayton: I got myself an invitational. Lived in Paris, and then took the train out to the suburbs every day. That was fun.

And I went to Delft. And that was, again, I got myself an invitation, because my friend who I worked with at IBM was a professor at Delft at the time. So I visited his group. And where else? Yeah, those were the two that I did.

Fairbairn: So how long did it take your wife to adapt to California? Or has she adapted, or still--

Brayton: She's been adapting.

[LAUGHTER]

Fairbairn: An iterative process, huh?

Brayton: We have children back in the East Coast. So we have a condo near where our daughter's family lives in Burlington, Vermont. So we can go back there and spend the summer there. And then another son works for Intel in Portland. So we go up there easily.

Fairbairn: So you're a little closer to him.

Brayton: And the third son is around here.

Fairbairn: Well, at least two out of three on the West Coast, huh?

Brayton: Yeah.

Fairbairn: So the work in formal verification. I was going to ask a specific question about that. Has formal verification made the step into the multiprocessor area?

Brayton: In using them?

Fairbairn: Yeah.

Brayton: Yeah. Definitely. Well, I guess multiprocessors have been more prevalent for the last two years. And so if you look at the Model Checking Competition, they now run the competition on servers that have four processors. And so all four are allowed to be used in solving the example problems that they run there.

And so the top competitors are all using multiprocessors. But they're using it not in the deep, parallel kinds of things, more at the top.

Fairbairn: You mean just segmenting the problem at the top, and them--

Brayton: Well, you don't segment it probably. So I mentioned you have BDDs and interpolation and other techniques, PDR. And so as you're attacking a problem, it's not just one thing that will solve it. Sometimes it could be solved by BDDs other times by PDR or interpolation

So you don't know which one to try. And you could say, I'll try this one first and give it so much time, and then try this one. With multiprocessors you can try them all at the same time. So this kind of parallelism is the parallelism of the algorithms at the top.

We haven't so far gotten into parallelizing the algorithms themselves. So there's been some effort in trying to do that, obviously. But it just hasn't panned out.

So one of my students, Sharad Malik at Princeton, I think he started a project to say, OK, let's try to investigate SAT solving and implement it as well as we can [just] to understand it, and then try to parallelize it. So they were not so successful in getting something parallelizable, but they were successful in improving the SAT algorithm so much on a single processor that this became one of the big steps and improved SAT solving.

Fairbairn: So how much of the problem as you look forward is making the current algorithms more efficient, or being able to take advantage of multiprocessors? Is it computationally limited, versus it doesn't matter how long you run it. There's a class of problems that it won't solve.

Brayton: It's still computational, limited, because in the category of complexity, these problems are PSPACE-complete. So it used to be that-- what's the complexity one, one level down? I'm coming a blank. We have to cut this out of the tape.

[LAUGHTER]

NP-complete. So SAT solving is NP-complete. And if you take it literally, you wouldn't try doing SAT, because they it's NP-complete. But there are a lot of practical problems you can do. But at the next level, if it's PSPACE-complete, you would also say, oh, this might be too hard. And yet that's what we do all the time in formal verification.

We know there are problems we can't solve. On the other hand, we know problems that we're trying to solve come from humans. And that's a very small, special category of the kinds of problems you see. It's not a random logic function. It's something. And so that helps.

And so we know there are problems we're not going to solve. And even though we run it in parallel and use all the algorithms, it's just not going to get there. Too much. But there are a lot of problems that are specialized that we can solve. We can't, say, predict what algorithm will solve it, but we're solving a lot of them.

And I'd say the examples that IBM gives us that are reasonably hard, 50-60% of those [are] in equivalence checking, where you do synthesis and then compare against the golden model. We can probably solve 85, 90% of those, even though those [types] are still PSPACE hard.

Fairbairn: Now what sort of complexity are those problems?

Brayton: PSPACE hard.

Fairbairn: The number of elements or gates--

Brayton: Oh, how big are they? Well, you can have 10,000 registers and 100,000 to a million gates. And we can still solve some of these. Others have 50 registers and a few hundred gates, and we can't.

Fairbairn: Still can't solve them.

Brayton: Yeah. So in the end, if you solve them and prove that the property holds or there's equivalence, you have to do a proof. It's a mathematic proof that says that no matter how many cycles you go, you'll never get an error on the output. So for [an] infinite number of cycles, you'll never get an error.

So you have to do that by some kind of proof, like induction or something. And so it's computer [based], but mathematical proof. That's what formal verification is. But it is surprising the size of problems that we can actually solve.

Fairbairn: So one last question on the technologies. What about the word behavioral synthesis? Where does that stand now? That's another area that we in the industry have talked about for a long time.

Brayton: Yeah, so that hyped.

Fairbairn: And I'm curious to your perspective on that.

Brayton: So that was hyped up quite a bit in maybe the '90s.

Fairbairn: Yeah, I helped in the '90s.

Brayton: And it never paid off. But it's one of these things where just like AI, there are parts of it that basically have spun off. And they've become very useful.

Fairbairn: Specialized narrow applications or whatever, right?

Brayton: I don't know exactly what and where it's successful, because I haven't been following it. But I know there are now companies and methods within companies that actually are quite successful in the high-level synthesis. Cadence has a very good effort I know, because there are a bunch of Berkeley people that have been developing that within Cadence.

There's been some startups that have been started and sold to-- I was thinking Cylex, but I'm not quite sure. But it apparently is one of these things that's now beginning to pay off. And people are getting into that.

Fairbairn: Finally getting some traction.

Brayton: The original idea that it should be a useful thing is there, because you describe something at a much higher level. But they couldn't compete with humans, but now there are things where you're putting more and more things together. So maybe it's a different ballgame nowadays.

Fairbairn: So in terms of your major technical contributions at Berkeley then, have we run the course? Are there other areas? When did you retire from active research?

Brayton: Well, I'm still doing it in a way. I'm what's called a Professor in the Graduate School. So that is a formal thing that allows you to get research contracts, have graduate students. You don't have to teach. You don't have to be on committees, things like that. But it's sort of going back to the old IBM days, where it's full-time research.

Fairbairn: You just get to do research, huh?

Brayton: But now I'm trying to slow down a little bit.

Fairbairn: What does that mean? Does that mean you only spend four or five days a week there?

Brayton: Well, I take off the whole summer. But I'm still in contact with my research group through Skype calls and things. And when I'm here, I have a weekly research meeting with my group. And then I actually do some programming in Python.

And trying out-- it's sort of orchestration of verification algorithms. How do you know what to do, when, and for how long? And putting that together into a Python script, and then using the concurrency, multi-processors to spawn off the individual algorithms on different parts.

Fairbairn: So you're still deeply involved. You are hands on.

Brayton: Yeah, I've always been a hands-on thing, even when I was at IBM and programming for Risch, who was one of my employees.

Fairbairn: So what do you do when you aren't doing this research at Berkeley? You have, obviously, your family, your kids and grandkids and so forth. Are there major hobbies that you've developed? Or not time for those things yet?

Brayton: Well, I've slowed down physically. I used to play a lot of tennis. And I used to ski a lot. Now, if I ski and fall over, it's hard to get up. And in tennis, my competitive spirit was there, but I couldn't reach the ball. And so I would pull my hamstrings. And so I had to give that up.

I ride a bike or walk or things like that, and read, do crossword puzzles, Sudoku.

Fairbairn: Looking back at the technology then, if you were starting anew, what do you think the area is ripest for impact or breakthroughs or whatever, at least in the general space? Or even different space.

Brayton: Well, personally, I would probably get into robots, robotics. It seems to be on the cusp.

Fairbairn: On the knee of the curve, right?

Brayton: So if I were younger and had studied it five years ago, I'd be really well placed. And I guess even today, that would be a good thing to study commercially. One of the things I like even better is astrophysics.

Fairbairn: That's had a huge Renaissance, hasn't it? It's just amazing.

Brayton: I mean, what they're discovering is just amazing, just fascinating. And every half a year, they have this major thing.

Fairbairn: Yeah, it's just stunning. I mean, for a while, it seemed like things were stagnating. And now it's just an explosion of new ideas and new discoveries.

Brayton: The other day, did you read about the inflation?

Fairbairn: Yes, yes.

Brayton: That's incredible!

Fairbairn: The sizes, and both small and large, and the periods of time, all of which are unimaginable in terms of what we normally come in contact with.

Brayton: I used to enjoy physics a lot. But I'm not going to start reading technical physics. I like these books that just are about--

Fairbairn: Skim the top.

Brayton: Skim the top. Have you ever read something called QED? It's a little book by Feynman on quantum electrodynamics. It sounds very imposing, but you read it and it's completely understandable. It's hardly any equations. So that's interesting stuff.

Fairbairn: So have we missed any major steps? Have we covered your major contributions, at least at a high level? We didn't dip down into the 450 papers and 10 books, but--

Brayton: Good thing.

Fairbairn: Could they all be categorized under the topics that we talked about at a high level, you think?

Brayton: Yeah, generally. There's the differential equations and then solving them and then logic and verification. And most of those are in the category. One paper I wrote was on how do you organize a tennis tournament of mixed doubles. And we called it Spouse Avoiding Round Robin Mixed Doubles.

Fairbairn: And it got published somewhere?

Brayton: Yes, in a mathematics journal, because it was tied to a very famous conjecture [self-orthogonal Latin squares] by Euler. You have n couples and there's a man and a wife. You never want to have the man and wife play with each other or against each other to avoid arguments.

And then the question is, given n, is there a tournament that behaves that way? And the answer is for the n not equal 2, 3, and 6, yes.

Fairbairn: So this sounds like something you could actually have practical application for. People putting together tournaments, right?

Brayton: It came when I was working at IBM. And I was in an indoor tennis club. And the manager of the club came up to me, and he said, I want to organize one of these tennis tournaments.

So I went back to IBM and talked to some really good people [Alan Hoffman, Don Coppersmith]. And we got together, and we slowly proved some of the hard cases, so for n equal to 10 and 15 and so forth, and built up that, until finally we could say after that, [given these cases] all the remaining cases can be solved by this other technique.

So it was an interesting paper. I was proud of that a little bit.

[LAUGHTER]

Fairbairn: Yeah, so I wanted to wrap up here in terms of you had said from your own personal view, you find the area of robotics particularly interesting. For somebody entering college or graduating and looking at the technology sphere, where do you-- obviously robotics is something that's interesting. What are the other areas that you think are ripe for exploration and explosion of opportunity and so forth?

Brayton: I guess anybody will say the biology part-- DNA and so forth, and then some of the synthetic biology. So that might be another field that is going to be here and growing and have a lot of commercial application.

Fairbairn: Does any of the algorithms that you've developed in logic synthesis and that sort of thing, is there any applicability to any of those in other fields?

Brayton: Sure. So a lot of synthetic stuff is digital in the way that they work. And so you could adapt some of these tools to mapping and to those kinds of things, little cells that do the computing. And there are new technologies that people are looking at.

And we don't know what's going to work. There's nanotubes, and there's spin electronics and so forth. So I don't know. I think that any technology that would come into play and be seen as moving ahead, you could then work on a logic synthesis system that would adapt, and get that as the hind end of doing mapping.

Fairbairn: And does Berkeley have a structure that allows the cross-pollination in these areas?

Brayton: Yeah.

Fairbairn: Are you involved in any of those?

Brayton: No, I'm not. I decided about two or three years ago not to look at anything else.

Fairbairn: Your plate was already full enough and time to wind down, not wind up.

Brayton: There's so many interesting things to do, even in both of these areas. And that's what my group does. And we have great problems and good ideas. So that keeps me busy now.

But Berkeley has this CITRIS building that is between Cory Hall and Soda Hall on the corner. And it's dedicated to getting people from different engineering departments together and working on different projects. And that's what this building is there. And CITRIS stands for [Center for Information Technology Research in the Interest of Society]**,**so societal problems.

That's been going on, and it seems to be very active. And many of the professors in my department have labs over there, collaborating with some of their colleagues in mechanical or civil or whatever. And I think the robotics people in our department have their projects over there with other engineering colleagues.

So, yeah, actually, Berkeley's quite a good place, where there's getting together and synthesizing different developments and ideas.

Fairbairn: All right. Well, I think that's a great place to wrap up. Thank you, Professor Brayton for--

Brayton: No, thank you for your questions.

Fairbairn: --taking the opportunity.

Brayton: You kept it going.

END OF INTERVIEW

