Oral History of John Darringer

Interviewed by: Doug Fairbairn

Recorded: November 10, 2010
Mountain View, California

CHM Reference number: X5970.2011

Doug Fairbairn: Okay. Today, we'll be talking with John Darringer, who is the manager of System Level Design at IBM Research, Yorktown Heights. I'm Doug Fairbairn with the Computer History Museum. We're going to be talking to John about his extensive career at IBM and other places involving mainly research in the area of electronic design automation. So, John, welcome. Glad to have you with us today.

John Darringer: Thank you.

Fairbairn: As part of this interview, we'd actually like to go back to your early days, forget IBM and so forth. We want to talk about where you grew up, what your family life was like and how you found your way into the world of technology. So you just take us back to where were you born, what was the family environment that you grew up in, tell us about how that influenced your path into technology.

Darringer: Sure. I was born in Zanesville, Ohio. Zanesville was about 50,000 people at the time. It's in the center of Ohio. Famous for its Y bridge. We grew up thinking that we had the only Y bridge in the world where two rivers had come together. My father worked in a steel mill, Armco Steel, which was the biggest employer, I think, in Zanesville. He had a group that was responsible for all the electrical equipment. My mother was assistant to Mr. Weller and Weller was the head of Weller Pottery, which was another big industry in the area. I had an older brother, much older brother, who turned out later to be influential in my career. I was a very good student, especially in math and science. I remember my sixth grade teacher was just so excited about science and it was infectious. She kept encouraging me. “You know, you have to be a scientist." Then, when I went to high school, we had great teachers, I thought, in geometry and biology and physics was very good so I really loved those subjects.

Fairbairn: Those early teachers can really be tremendously impactful.

Darringer: Looking back, you know, it's just- one or two teachers really made such a big impression on you that...

Fairbairn: Even as far back as sixth grade?

Darringer: Yeah, that's right. Miss Hinely. She was something. I had gotten interested in electronics. My brother, my father was involved in electronics, I mean, electrical stuff so I started making crystal sets and then I taught myself how to make amplifiers and oscillators, first with tubes, because that was the thing, and then transistors were just starting to come in so you could buy a CK722 or a 2N107 and make little amplifiers and oscillators. That was a lot of fun. Then I got a part-time job working at Les Sherwood's Radio Repair Shop. So, I’d go there and fix people's tube radios and get a discount on my parts. That kind of got me interested in the area but I think it was really my brother, who was very emphatic that I should be an engineer. He had, as I said, he was older so, when I was very young, he was in the navy. He came back from the Navy, worked in the steel mill for awhile, decided that wasn't what he wanted to do and decided to go to college and got accepted at UCLA. So he and his wife took off to go to study electrical engineering at UCLA. Now, that was a pretty bold move at that time, in the '50s, to go from Zanesville clear across the country to go to school. My father didn't understand why he was doing that but he did. Even though he was remote, he kept in touch with me, kept encouraging me to study and kept insisting that I was going to be an engineer. So, when it came time to think about college, I applied to engineering schools and Carnegie Mellon, I thought, Carnegie Tech at the time, I thought was a very challenging and interesting school so I applied there and got in. I remember my father, again, was

a little concerned about the cost of that but I managed to get a scholarship so it was okay, I got to go. It turned out, I loved Carnegie Tech and Carnegie Mellon. It was great.

Fairbairn: So you went there to study electrical engineering?

Darringer: Electrical engineering. I'm going to be an electrical engineer for sure. I got there and I realized that I wasn't so fascinated with circuit design but I was fascinated with engineering. The first engineering course I had, again, was Don Scharfetter was the guy. He was, I think, a recent graduate from Carnegie Mellon, gave this...

Fairbairn: I remember Don Scharfetter.

Darringer: Is that right?

Fairbairn: Yeah.

Darringer: Well, he made a big impression on me because I had learned how to solve problems where there are techniques but Don would teach us how to solve problems where there is no- you don't know how to solve it and you just have to reason with basic principles and make outrageous, simplifying assumptions and I really got fascinated with this. I thought, boy, this engineering is much better than math and physics. I really like this. So decided- I continued to be an electrical engineer. I was in the honors program and, because of that, we didn't have to learn about motors and generators. Instead, we had to study semiconductor theory, about holes in electrons and I had an information, a graduate course in information theory that, again, I just was so fascinated with that. Somehow, out of that and other classes, I had said control theory. I want to go into control theory. So, when I graduated with my Bachelor's degree, I decided to leave Carnegie Mellon and go somewhere else to study control theory. But Carnegie Mellon had made, I think it was still Carnegie Tech, made an offer, a better offer so I ended up staying at Carnegie Tech and studying control theory. Then, in that year, I get that master's the next year, we studied computer controls and I found out about computers and I thought, "Oh, this is really interesting. I really like that." My brother had been getting me some very interesting summer jobs in California working with aerospace companies and I got a chance to program a computer at Autonetics and so I thought, "I really like this." So that's what pulled me into-- computer science wasn't a discipline at Carnegie Tech at that time. It was just starting to emerge. So, after my master's, I decided to stay at Carnegie Tech and study what was going to become computer science. They had a program there, interdisciplinary program, and they had Alan Newell and Herb Simon from the Graduate School of Industrial Administration and Alan Perlis was there. They had recruited Gordon Bell to come from DEC and become a professor there. He brought a whole new, I don't know, direction or a new influence at Carnegie Tech, very strong in theory and he brought in this practical experience of how to build computers and just told one story after another about what computers were like and how they were built. Fascinating. He gave these great seminar series so he was a big influence. He was one of my co-advisors.

Fairbairn: So what year did you enter Carnegie Tech?

Darringer: 1960 was when I graduated from high school. Then I got my bachelor's in '64 and my master's in '65 and then started on the Ph.D. program.

Fairbairn: All through Carnegie?

Darringer: All through Carnegie. Well, I kept saying to myself, "You know, you really should go to another school," but it just turned out that, each time, it seemed like the right place to go. Looking back, it was great. I think Carnegie, became Carnegie Mellon, was a great school. What I like about them is they sort of- I felt like we were solving real, practical problems, right? I liked that. Gordon Bell was...

Fairbairn: Gordon Bell had a major influence on you.

Darringer: I think so. He came with his own PDP8 computer. We hadn't even seen this, you know? He brought one, had one in his office. So he was a great influence. Dave Parnas was my advisor. Dave was sort of interested in software design methodology, software development methodology and we talked and we decided that I could work on hardware specification. He was interested in software specification, I would work on hardware specification. So he and Gordon Bell were sort of my co-advisors. Parnas was my primary advisor. So what I ended up doing-- Parnas suggested I read a paper that turned out that I really, really liked. It was by Niklaus Wirth and Helmut Weber. It was called EULER [1]. Nik Worth was proposing EULER as a replacement for ALGOL and he wrote this paper which specified the language in two pages, as I recall. I mean, maybe it was three. Syntax and semantics of a very elegant, really unique language in such a compact space. The paper also described a LRK parser generator. I read that and I got fascinated, built the LRK parser generator, implemented the EULER language and sort of got myself into the language creation business, which...

Fairbairn: On what machine were you doing this work, do you remember?

Darringer: I think it was a Control Data machine. I remember, when I finished my thesis, I'd wrote the program in ALGOL on that machine and they were phasing that machine out. I and one other person were the last people allowed to run on that machine and, as soon as we were graduated, they were going to get rid of the machine. So there was no maintenance. We had to start the machine ourselves, maintain it and the hydraulics, the disks were controlled hydraulically and the hydraulic system was leaking so there was oil on the floor. It was quite an experience. Because the disks were sort of fading, we stored everything three times, you know, because we knew that it was going to get lost. Anyway, interesting times. So EULER, I got into the business of creating languages. What I did for my thesis was I came up with a minimum extension to ALGOL for describing hardware, just added three features, a register type so you had fixed bit length objects and then a bracket for saying that this would take one unit of time and then another way of saying that these statements would run in parallel. So just those three constructs added to ALGOL. We described a little sort of instructional computer in this language and could simulate it. Then I started trying to figure out if I could generate a hardware implementation from that automatically. Did do that but, looking back, it wasn't a very good hardware implementation. I would say it was off by at least a factor of two, maybe more, but...

Fairbairn: In terms of...

Darringer: Gate count, the gate count was the kind of thing we were looking at, at the time, yeah. So at least I became familiar with the problem of logic synthesis. Another thing that, at the time, there was a problem that students were working on their Ph.D. for one, two, sometimes as much as four years and then being told that their thesis topics were not acceptable so they were really surprised and they would have to sort of start over. In some cases, they quit. In other cases, they did start over. This was

unnerving for a lot of graduate students, thinking, oh, my gosh, you know, because, when you're working on your thesis, you don't really think it's that special, right, because you understand what you're doing. I happened to have my office, we shared with other grad students, was called Porter Hall 13 and the legend was that no one had ever graduated from Porter Hall 13. So it turns out...

Fairbairn: 13 was the bad number?

Darringer: I did graduate. I was the first to graduate from Porter Hall 13 and Parnas' first graduate student. The day that I turned my thesis into the library, there was a session in the auditorium where students were complaining about this lack of predictability of their program. Alan Newell, you know, listened to them and they did make some changes to have more periodic reviews to make sure that students wouldn't be surprised. So, anyway, it was with some relief that I managed to graduate.

Fairbairn: You'd pursued that path because of your own thoughts and ideas or were they heavily influenced by your advisors?

Darringer: Yeah, I think it was a combination. I was fascinated with- we had taken courses on assigning- Bob Floyd taught there and taught complexity theory and was one of the ones that built this first program verifier. So we were thinking about it specifying programs. Parnas was very much into how do you specify programs so I think I got interested in this business of specification and how do you specify hardware behavior and I liked implementing languages interpreters and so...

Fairbairn: So all of that was very cutting edge stuff at that time.

Darringer: At the time, yeah. I think-- well, and Parnas was a very demanding advisor, I think maybe because I was his first student. So I was reading a lot of what was going on all over the place. There were some other papers. Jim Dooley and Don Dittmeyer, I think, in Wisconsin, I think it was, did a thesis on synthesis as well [2]. That showed up just a few months before I was supposed to graduate so we had to explain how my work differentiated from that. So there was some other work in the area but not much. But it was a chance to look back at the previous techniques.

Fairbairn: So you graduated in, you got your Ph.D.?

Darringer: 1969.

Fairbairn: '69.

Darringer: At the time...

Fairbairn: What did the world look like at that point and your priorities?

Darringer: Well, what had happened is, as I was finishing up my Ph.D., Philips, from Holland, was visiting schools in the United States trying to recruit people to help them. They had just acquired Electro Logica, which is a company in Holland that made scientific computers, and they had decided they were going to go on, get into the mainframe business and compete with IBM. They were looking for people,

students and actually they actually recruited my advisor to come and help them with this program. By the time I graduated, they made me an offer and Dave Parnas an offer to come and work in Holland. Now, you know, I grew up in Ohio, I had never been to Europe, I didn't know anything about Holland so my wife and I decided, "Let's go," you know? I think, at the time, it was a pretty bold move but, looking back, it was just perfect. It was great.

Fairbairn: You got married somewhere during your...

Darringer: Right. I met my wife at Carnegie Mellon and we got married towards the end of my graduate school.

Fairbairn: She was equally willing to head off to Europe?

Darringer: Well, I think we both kind of crossed our fingers and said, "Okay, let's try it" and it was one year, possible two years so we thought, "Well, let's give it a try." I mean, it's just wonderful. The Dutch were very, very friendly and anxious to speak English and Philips was a great experience because I was with a design automation group there that was trying to build a design system to support the creation of large systems so we had to sort of start from scratch. How are we going to specify these machines and how are we going to store the data and how are we going to do simulation?

Fairbairn: So it was very early in their process?

Darringer: Right. They had some experienced people from Electro Logica but this was a much bigger effort. They had also a lot of expertise in electronics. Philips is very strong in that area. So it was interesting working with the different people there and I got to try some different things out and find that some things worked and some didn't work so I think I learned a lot. Plus my wife and I got to see, I think, practically all of Europe and Istanbul. We went everywhere, so it was just a tremendous experience for us. At the end of two years, we decided that I really needed to go back to the United States and start building a career there. We had sold the things in our house and had a moving van scheduled to come and this guy, Hans Ledeboer was the director at Philips, decided he wanted me to stay. So he had talked about it and I had told him, "No, I really am leaving," and so he invited me to his house one evening and persuaded me to stay. Somehow, he sort of gave me a blank sheet of paper and made me an offer I couldn't refuse so we decided, "Okay, we'll stay another year." So we had to buy back some of the stuff we had already sold and cancel the moving van and we ended up staying for a third year. At the end of...

Fairbairn: Did you continue the same work that you had been doing?

Darringer: Well, he allowed me to have my own project so I could kind of do what I wanted to do and travel a bit to some conferences where I hadn't done before so I had a little more freedom. So I was experimenting with a new language for describing hardware and software, pretty outrageous for '72, I think this was. At the end of the third year, though, we said, "No, we really do need to go back to the United States." So...

Fairbairn: How far ahead did Philips progress in the development of the mainframe computers?

Darringer: They had built this P series, I think there was the 1200, the 1400 and they had, I think, made a mistake and maybe they would agree, too, they had decided to make them almost compatible with IBM. So it was largely like an IBM machine except it had a few extra instructions which they felt would give it an advantage but the consequence of that, you had to recompile your code to run it on a machine so it wouldn't run without this recompilation and that turned out to be a problem. Plus competing in the mainframe business is a very competitive business. So they tried it for several years and then just sort of decided to get out of that business eventually. So I don't think they were that successful. I learned a lot, though, about what it means to build a machine in the industry and what are all the issues. I remember giving a talk on this approach for hardware specification for the whole machine, not just the chips and the logic but the racks and the cards and boards and, at the end of it, this gentleman said, sort of, "Sonny, come with me." We went down the hall and he showed me a room that was full of shelves full of drafting documents and the documents were for power supplies. He said, "You just missed the main, the most important part of your system is the power supplies." <laughter> He was right. I had completely forgot about that. Here's a whole room full of this handwritten documentation that we were just ignoring. You learn a lot when you go from university to industry. We came back to the United States. I didn't have a job because companies said they were interested but they were reluctant to have me fly from Holland to come for an interview. So I came home without a job and we lived with my wife's parents in the suburbs of Pittsburgh. That was a little uncomfortable, not having a job and being with your in-laws but I interviewed and eventually did get a job at...

Fairbairn: What were you looking for at that point? Were you looking for a research job? What kind of job?

Darringer: I was looking for a research position. I had decided that software was really the problem. I'd worked on hardware a lot and I could see that software development was really where I thought the problem was. So why can't we make some tools to support software development? I was lucky to get a position at IBM research in Yorktown to work on just that.

Fairbairn: What other companies- did you interview with other companies?

Darringer: Yeah. One of the first companies I interviewed, I think it was Control Data. I thought the interview went pretty well and, when I got a rejection letter, I was a little bit surprised and my feelings were hurt and so I ended up calling this guy, who I think was the head of engineering at Control Data, and I said, "I think you guys are making a mistake by not hiring me." Fortunately, he agreed. He said, "You know, you're right. Your skills would be very, very helpful here," he said, "But I just couldn't convince my management to allow me to grow the size of my group so," he said, "I'm sorry about that but you would help us a lot." So I felt a little better because that was my first rejection letter. So I felt like, okay, I'll keep trying. Fortunately, at Yorktown, they also were not hiring so they had to make a special exception and get the director of research to sign off on an offer but, fortunately, that worked and...

Fairbairn: So were they offering a specific job or project at that time or were they inviting you into to help define what direction you wanted to go?

Darringer: Research has a very flat structure. Almost everybody has a research staff member, which I liked, because, at Philips, there was a strong hierarchy and, at IBM research, you didn't see any hierarchy. Everybody was kind of all the same level. That's by design. I looked at different projects. There was a project trying to bring automation to the business process, trying to help people with small

business figure out how to automatically set up an accounting or process for your particular company. I thought that was intriguing because you had a language for specifying the business process and sort of a compiler that would generate the tools to support that. So that was tempting but I ended up joining a group that was focused on software development and trying to build tools to support that. That was exactly what I thought I wanted to do.

Fairbairn: Tools to support software development, not hardware development?

Darringer: Right. Software. So, for example...

Fairbairn: You had decided that software was the problem.

Darringer: Exactly. Hardware will take care of itself. Software was really where- and probably I was right. I remember, too, as a new person, I think, in order to get you to meet some other people, they assigned me to a task force and I learned what a task force was. This was to figure out how to control a high performance robot arm, they called it the manipulator. It's hydraulic driven, capable of tremendous forces and very precise movements and it was in a room where you had to wear a helmet if you went in there because it could really hurt you if it got out of control. So I was this language guy, right, so I created a language called Maple [3], which was interpretive, something like APL. It didn't look like APL, it looked like a Command language but you could type commands and the manipulator would operate or you could write programs and execute them. So that was a lot of fun. It was just a short assignment. What I remember was there were some very smart people there, some of whom became IBM fellows, who were trying to think of programs to put a screw into a hole so you have to turn it andadvance to make the screw fit in and also to locate an object on a table. Then we would work on these programs and then somebody would come in and say, "Well, we have a device to do that. There's an automatic screw insertion machine. The robot picks it up and just inserts the screw." So we were kind of deflated that we had written this program and it wasn't needed. Anyway, an interesting introduction to some of the work there. The first real thing I worked on was a symbolic execution system. Jim King was my manager and he had worked with Bob Floyd at Carnegie Mellon and he had built, I think, the world's first program verifier at Carnegie Mellon [4]. As part of that, he had written a package in Assembly language for putting formulas into canonical form, very efficient package. So I took that package and I built a, again, another interpreter based on PL/1, which was IBM's dominant language at the time [5]. So we could take a sub-set of PL/1 and you could run it just like normal, call it with some numbers and it would execute and produce a numeric result. You could also call it with symbolic inputs, you know? Put an X in quotes or any string in quotes for inputs and it would run and produce a formula out the back, using this package. You could put output assertions to check the results, you could put input assertions to sort of control the input space. Then, if there's a loop, you would get an infinite execution tree. You could add an invariant in for each loop and then it would collapse that execution tree into a single path. So we had one system where you could do normal program execution testing. You could do the symbolic execution so for, like, the data path, one path would cover a large number of normal tests and then you could, in fact, do complete formal verification by providing the appropriate inductive assertions. So I really liked that nice spectrum, you know? Because trying to get people to use program verification was kind of hard at the time. We thought, well, we could kind of ease them into this. That tool didn't- it was a great idea. We published it but it didn't really go a long way. Another thing we did, abstract data types were kind of a popular idea at the time and still are. I added the ability to specify abstract data types and then you could specify their behavior of their function of the operators with axioms. So you could say that, if you pushed X on the stack and then you popped if off, you'd get X back in a form of an axiomatic statement. Then this

symbolic execution program could reason about those kinds of programs. In that way, we could expand into more interesting programs. So that was fun.

Fairbairn: How long did this phase of your work go on?

Darringer: This was sort of the mid-'70s. So, towards the end of the '70s...

Fairbairn: So you joined IBM in...

Darringer: '72. So this was probably around '76, something like that.

Fairbairn: So you were involved in that for several years?

Darringer: Yeah. Then...

Fairbairn: Then you got steered more back towards hardware?

Darringer: Well, yeah. What happened was I tried this on programs and I realized that, to prove properties- reason about programs, you needed a really rich theory. I thought, well, you know, hardware should be a lot easier because it's mostly Boolean operators. Why can't I apply this in that space? So I wrote a paper on how to apply program verification techniques to hardware verification [6]. I just gave a little example and published that paper in, I think, '79. That was included in this 25 years of DAC as one of the interesting papers. I felt good about that. That got me to looking at hardware design again in IBM and I realized, especially looking back, that this was a very special time in IBM. They had a machine called the 3081 and the guy that was responsible for the design methodology for the 3081, Mike Monachino, I think made some really high risk decisions [7]. One was to use this level sensitive scan design. I think that was the first machine to use that completely. So it's a synchronous machine, all the latches are accessible. So Mike made a decision to use level sensitive scan design and that decision had such a big impact.

Fairbairn: Just to be clear, describe briefly what...

Darringer: Yeah. LSSD [8] was an idea at the time that you would- one thing is, you would be able to externally access all of the latches in the system and you could scan in values to any set of latches, execute the machine and read out the values in the other latches. Another thing was that it was two sets of clocks. So there would be a clock to release the data and then another clock to capture the data at the end. So it was a very simple, systematic system to design. The consequence of that was that you really could, and IBM did, separate timing analysis from functional correctness. Everybody, even after that, was doing sort of an event circuit simulation and you would simulate this circuit and see what the timing behavior was and say, "That would check timing" and you were also checking the function at the same time. So what IBM did was they said, "No, we're going to check the timing with the static timing analysis [9]," which looks at all paths, not just the ones you happen to simulate, and they were able to do that for this synchronous design. Then the other thing they could do is, because now that I know the timing is correct, I can just collapse all these levels. As long as I execute the gates in the correct order, I don't have to worry about time. So the simulation cycle, they called it cycle simulation [10], ran at least two

orders of magnitude faster than the event-driven simulators so big speed up. So that was a big hit and they feel like had to do that in order to simulate these very large machines.

Fairbairn: So that was in the late '70s as well?

Darringer: Yeah, this was late '70s, right. The 3081, I think, shipped in the early '80s, maybe '81 or something like that. Another key thing that was a part of this methodology was a behavioral specification. So they had come up with a flow chart language so there were if/then/else decision box, GoTo branches, loops, no, I don't think there were loops, there were GoTos. And registers and so you would lay out on a piece of paper or they had a graphics interface where you could describe these flow charts and that would describe the behavior. So you would typically say, if this register has this value and I get this signal and then I do a certain computation, then I will set this register and transmit these signals. And they would branch off to other pages. So that was the behavioral specification. That allowed them to get that up and running before they did the hardware implementation and figure out if the machine was going to work correctly or not. Then there was a structural language, sort of much like it but, I mean, a corresponding language for describing the connection of the gates. Designers would complain that I have to design the machine twice. I have to write the flow charts to specify the machine and then I turn around and have to figure out how to connect the gates together to implement the machine. Then there was this fellow, Ralph Bonson, who had come up with a SAS, Static Analysis System [11], I think it was, for actually mathematically proving that the gates and the behavioral specification were equivalent. So this was Boolean equivalence checking in the late '70s.

Fairbairn: In fact, it sounds like this whole set of tools, the static timing verifier, the clock based simulation, were all unique in the industry at that time.

Darringer: I think so. And had a big impact, I mean, tremendous. They stepped through the methodology I think was a big step forward. So, especially looking back, you could say the stage was set for synthesis because there were chips where there was a formal specification in this flow chart language and there was an implementation in this and, because of SAS, I knew they were going to be equivalent, so the question was, well, why can't I generate the structure from the behavioral? Now, people who had tried- I don't think they had tried BDL/CS [10] and BDL/S [10], the languages, but they had tried that so everybody knew that it couldn't be done.

Fairbairn: Describe what those are, those languages.

Darringer: Yeah. BDL/CS was the Basic Design Language for Cycle Simulation so that was the list language for the behavioral specification. Then they had BDL/S for the structure. That was for the net list language. Those were the...

Fairbairn: These were internal, IBM languages?

Darringer: Internal IBM languages, right. Right.

Fairbairn: So IBM was not doing schematic diagrams, they were using text-based design?

Darringer: Right. I think another step that IBM was early with was Master Slice [12], what people called gate arrays. So they did not design at the transistor level, they designed at the gate level and used gates and flip flops and then they had theis Master Slice. Now, the chips, at the time, like the 3081, had 704 circuits, this is a very- IBM always had circuits, not gates. It was a collection of transistors in each of the array sites and those could be configured by the metallization to deliver a flip flop or maybe a four-way input NAND with two outputs or maybe two smaller NANDs or that kind of thing. So designers really did design at the gate level and the library was robust enough that it could deal with that. Fortunately, at the time, bipolar circuits were so strong that wire load was not a concern. You really just looked at the other devices of the other circuits you were driving. So approximate timing could be made by calculating the stages and...

Fairbairn: Adding up...

Darringer: ...adding up the stages, right. At least we started that way. So, let's see, what happened was, I went to Poughkeepsie, Poughkeepsie is where the mainframes were designed at IBM, I went there and we started talking to designers and we found a designer who had designed what you would say is a sort of simple chip, one of the more simpler chips and we got from him his flowchart specification and we got his structural specification. We brought them back to Yorktown and we tried to think about how could you generate one from the other.

Fairbairn: This is on your own? This is just something you decided...

Darringer: Yeah, that's right. It's exploring, right? It's not an official project. It was just I got curious and so I went and looked into this. Because I had built so many interpreters and compilers and we had spent time on rewrite systems, production systems, for handling these axioms, I kind of felt like there ought to be a series of transformations that could get from one to the other. So I tried to do it manually because I didn't have any tool to do it at the time so I just manually took the flowcharts and tried to build a graph in a very straightforward way. If it said plus, I put plus. If it said if then else, I would convert that to some logic formula. I built that. Then would look for opportunities to simplify it. Then, when I couldn't find any, then I would translate it to the next level, sort of a technology independent NAND level, look for opportunities to simplify it. Translate it down to the specific technology elements that were in the library, look for opportunities to simplify it. It was encouraged. I didn't have it but it looked, I think we can do this, you know? So I tried to make a project proposal and, again, there were many wise managers who knew this couldn't be done. Many people had tried and so why waste your time on this? But I managed to get a couple managers interested in this and they sort of appreciated the new approach.

Fairbairn: I was going to ask, so it had been tried before. What was the key-- you realized that you had a different way of approaching that?

Darringer: There had been a lot of work on Boolean minimization and there was a program, it turns out IBM, I think, invented the PLA, Arnold Weinberger invented this PLA and there was a MINI [13] program which was a heuristic program for minimizing logic to implement in PLAs. So people had tools for doing two level logic minimization and, of course, the textbooks had that stuff. So the problem is that, if you look at the bipolar circuits that were being built at the time, you could find paths that were 30 levels deep, maybe 20 might be more typical. But, if you take logic that's 20 or 30 levels deep and try to compress it to two-level logic to sort of implement it, it will explode. Just number of terms will just blow up. So applying two-level minimization, I think several people had tried it, it just doesn't work. It's just not- we

couldn't find a way to apply it. So what I instead, this was more like looking for opportunities, looking over the whole design, looking for local optimization opportunities. So giving up on global optimum but looking for local optimization and writing equivalence preserving transformations. The fact that, at least manually, it seemed to work on at least a simple chip, that plus it was a new approach, I managed to get some people, including the director of our computer science department, thought, "Okay, let's give it a shot." Let's do this. Let's start a project. Bill Joyner was, I think, the first person to join and he had been working on this proving micro-code correct so had a good background about hardware design and formal methods, very, very helpful. Together, we plowed through this chip and started building a little tool that could read in this BDL/CS language, map it to an internal representation, started writing transformations. Then there were transformations and then there was a scenario, we invented the term scenario, where you would apply them in sequence and explore with different sequences and different conditions. We started to think, well, actually, we were able to generate this one chip and so then we went to look for another chip and another chip and we did three chips. In one of the cases, we actually could beat, but just by one circuit, we could beat the manual design. I can remember, you know, we went to the designer with that chip and showed him this. He was very skeptical. I mean, "How could that possibly be?" So he went through every- this is maybe 20 or 25 pages of logic diagrams that we generated. He went through every page and, first of all, they complained that the logic diagrams we wrote were unreadable because we had generated names, not manually produce names, and they were laid out in a left to right fashion, we thought, in a logical way. That was not the way he liked them. So, anyway, we eventually found out that, yes, he agreed. It was one circuit better but, you know, we had spent a lot more time on it than he had and that was okay. So, anyway, we felt good. We were able to produce production chips. I need to mention that circuit minimization was just one of maybe a dozen constraints that had to be met. Choosing the correct latches was another important. There were lots of different latch types and we had to choose. There's this level sensitive scan design, had certain clocking constraints that had to be wired up. You had this chip connect together the scan chain according to certain rules. There were fan in/fan out constraints. The I/O circuits, drivers, sometimes had function in them so you had to take advantage of that. There were just a number of rules. And then getting ALDs, Automated Logic Diagrams, that people could read took a lot of time. So Bill and I started it and then Louise Trevillyan joined. Louise Trevillyan had been working with Fran Allen on PL/1s optimizing compiler so she had this experience about optimizing compiler techniques. She joined us. She brought with her a package that Bill Harrison had built that was for data models, creating a data model, so we replaced my homemade little storage system with this thing and that was really great for allowing us to much more efficiently represent these designs and apply transformations. Louise spent a lot of time working with designers on this layout of these diagrams, trying to- it was partly figuring out how to do it and the other part was working with the designers, getting them to realize that they didn't have a consistent standard. One guy wanted them one way, another guy wanted-- so we would get them together and get them to kind of agree on some common standard and then we would implement it and it was fun. I mean, this was a period where we were in Poughkeepsie every week. We were on the phone with him almost every day and there was a fellow, John Gerbi, was running chips for us up in Poughkeepsie. Jim Eadie was working with us to make sure, get support for us up there. So a great period where you're sort of in the battle, right? You're working with the design team.

Fairbairn: When you made this sort of breakthrough, you actually got some chips that were the equivalent of one circuit fewer. What year was that?

Darringer: So it would be in-- I think we published it in '80 so I think it was done in '78/'79 is when we were getting these results.

Fairbairn: This is years before the industry was...

Darringer: Yeah. I think so. Well, there was no EDA industry at the time, right? I mean, not even...

Fairbairn: Yeah, that's right. The big three at the time didn't even start until the early '80s.

Darringer: Right. There was no Mentor, no Daisy, not at that time. This was...

Fairbairn: The only thing going on was layout stuff, those sorts of things, and a few ERC verification companies.

Darringer: Right.

Fairbairn: But no design related ones.

Darringer: Right. Yeah. So we did publicize this. We wrote a paper, a new look at logic synthesis [14], and...

Fairbairn: Was that controversial? Was IBM happy to publish that or did they consider it proprietary or what?

Darringer: I think it was okay. This was early so I don't think IBM really appreciated the impact yet. When we published it, we only had three chips. But research, usually, you know, reserves the right to publish stuff and we didn't reveal any information about the technology. So we published that paper and then I went around thinking, you know, what do other people think about this? What are other people doing? I remember giving a talk, I think it was Berkeley, we went to the compiler people there that Fran Allen knew and they said, "Wow, I'm glad to see that the hardware people finally appreciate the benefits of optimizing compiler methods." So they saw it as….. They talked about people optimization and dataflow analysis and so they saw it as compiler techniques being applied. We went to Stanford and they saw it as an A/I effort so they said, "This is their A/I techniques being used." I kind of agreed with all of them, "Yeah, I think we're mixing a lot of this." Carnegie, I think, saw it, they were into production systems and they saw it in rewrite systems being applied. So it was familiar techniques but I kind of knew that to the synthesis space this was not Boolean minimization, right? This was sort of local transformations.

Fairbairn: So logic synthesis turned out to be sort of major development, had a major impact on the industry eventually. Was there anybody that you, having described this work, really took it and tried to...

Darringer: I organized a logic synthesis workshop on Catalina Island in the early '80s, I think, very early. Still, today, I think it's called the IWLS, the International Workshop on Logic Synthesis, started from that. I think a lot of people there were very impressed with what we were doing and decided that they could do that. I honestly don't know how the synthesis started. I know Synopsys started from GE but I don't know what the connection there is.

Fairbairn: So you were doing this in the late '70s, early '80s. When did it become a major production tool within IBM?

Darringer: Well, what happened was we did three chips and then we did five chips and that got people's attention and there happened to be an offload engine, a TCM, Thermal Conduction Module, with 100 bipolar chips on it, that was going to be designed to be an offload engine for handling, I think, the channels on the mainframe.

Fairbairn: It would offload a certain set of...

Darringer: Right. IBM, the mainframes have the computation and then have these channels for dealing with disks and things and I think the offload engine was to implement those channels. So this was 100 chip project that had gotten started late, staffed, hadn't really been staffed completely so I remember the manager was in a bit of a panic. "Where am I going to get the people to do this?" He heard about this logic synthesis and he spent a lot of time looking at what we'd done and he decided, "Okay, we're going to do it. We're going to try to use synthesis on these 100 chips." It ended up, after a lot of work, we did 96 of 100 chips automatic, completely automatically.

Fairbairn: So is this an iterative thing with the designers? They were using your software, it was still relatively new, and you were constantly improving it? Or was it pretty solid by this time?

Darringer: No, no, no. This was constant improvement. This was this period where we were up there every week. Now this is serious, these are serious product chips, right? So this is not funny, this is “Darringer, What's wrong here? Why is your program generating this junk?" and then we had...

Fairbairn: No longer friendly...

Darringer: They would buy us donuts from time to time but that was about it, yeah. <laughter>

Fairbairn: What year was this?

Darringer: This was early '80s. '82/'83, I would guess, something like that [15]. Yeah. Maybe '81.

Fairbairn: So that timeframe in the external world, Daisy and Mentor and Valid were just getting started with schematic entry and those sorts of things?

Darringer: Right.

Fairbairn: They weren't even thinking about this sort of thing in the commercial world?

Darringer: No. I don't think there was- the only thing that was going on was PLA synthesis, right? I mean, people were doing Boolean minimization and generating PLAs. That, I think, was it. There was talk. The math department, Bob Brayton was in the math department and had invited Gary Hacthel had come to visit and Alberto Sangiovanni-Vincentelli had come to visit and they were trying to build- they did

build, I think called it Yorktown Logic Editor [16], a set of tools for manipulating Boolean functions. That or at least that group later developed the Espresso [17] program for Boolean minimization. But, in terms of building real hardware, I wasn't aware of anybody really using those tools and techniques except for PLA because PLAs matched that optimization technique very closely.

Fairbairn: So you were working on this project. It was successful. They got this offload engine cranking out.

Darringer: Yeah. That kind of...

Fairbairn: ...that must have generated a lot of attention...

Darringer: Exactly. They saw that and they said, "Well, that's it. This is where we're going." So Poughkeepsie immediately spread to I think all the chips in Poughkeepsie, not that we did- we would always, occasionally, a chip where a human would have to go in and tinker a little bit but it was heavily used in Poughkeepsie.

Fairbairn: How did tool development work within IBM in that kind of situation? In research, you're developing it. You're working interactively with designers, hopefully it gets to a point where it's working and, all of a sudden, now it's in production.

Darringer: Very good question.

Fairbairn: How does this all happen?

Darringer: It's a perpetual puzzle or a quandary in research. So what happened was, the idea is that, when it gets to that stage, you should be able to transfer the tool to another group and we did have- IBM does have a very large central- it was called EDS, Engineering Design System, group at the time. So the logical thing would be for them to pick it up. But what happened was they saw this and they saw the potential and they decided they were going to re-implement it themselves and use PL/S, which was sort of a syntactic front end to Assembly language. They thought that this would give them more performance. They had their own idea about how to represent the data instead of this data model that we had borrowed from the compiler world. So they went off and built their own thing called Logic Transformation System. It didn't really do synthesis. It was more built to map one existing design into a new technology. One of the things that, while we were doing synthesis, we did discover that, in Rochester, there was a Jim Gilkinson had built a Technology Mapping System, TMS he called it [18]. It was specifically because Rochester seemed to...

Fairbairn: This was IBM in Rochester?

Darringer: IBM in Rochester tended to take their design and then really do a remap into new technologies without completely redesigning it whereas Poughkeepsie would tend to make bigger changes. So Poughkeepsie would almost never do a remap. Rochester would often do a remap. So they had built this Technology Mapping System, just helped the designer systematically replace this latch with this latch, replace this gate with this gate and, in the process, do some optimization when the circuits would allow you to do different functions. So Jim's program, I think, actually was used in Rochester to

map systems and the central group worked closely with Rochester and they said, "Okay, we're going to build this new logic transformation system that will replace TMS, replace LSS," our system was called Logic Synthesis System, and they spent, I don't know, a couple of years on that. It may have been used in Rochester on some mapping problems but it really just never could compete and eventually faded away. Then, eventually, Poughkeepsie insisted that the central group pick up the LSS tool and so they did. That went well. They were very helpful. Synthesis then started spreading at IBM and it wasn't just bipolar circuits for mainframes. We did mid-range computers and we were starting to do the workstation, the RS-6000 workstation microprocessor. We had a PC division doing PC chips. Most of those were in CMOS. So we had to learn how to do CMOS, which was different.

Fairbairn: But all this came back to research.

Darringer: Yeah, yeah. It was being driven out of research with really some very good, valuable help from the design group. Each development site has their own tools, people as well so they would work with this. So it started spreading, not completely smoothly. I mean, there would be places like in Germany, I remember, they got very upset and decided synthesis would not work so I had to go and visit them and spend the week there trying to explain to them how to use it a little differently. It started. But, overall, so I would say that, in the mid-'80s, towards the end of the '80s, all chips, all IBM chips were using LSS. They would try it and then there could be cases where they would have to go in and fix things up because of schedules or something. Synthesis kept evolving, getting smarter and having more clever scenarios, trying to put in more global optimization for redundancy removal. Dan Brand wrote a program that would really do global removal of redundancy [19].

Fairbairn: So was this program, from the beginning, were there issues in terms of capacity, computer cycles required? Was efficiency an issue, given the resources typically available to design teams? Was the computing power...

Darringer: It turned out to be a good fit. When we would go to Poughkeepsie, they would run the chips overnight so it would take several minutes, I think it was minutes, to run a chip. We would run a chip and we would count stages. So that was our timing model. Then we would run static timing analysis and usually, if we were off by a little bit, we would go back and change the assertions on the asserted arrival times and required departure times, rerun synthesis. So, in the beginning, it may be two iterations, maybe three. Two iterations would probably be enough to converge so that was fine. What happened was, as we started doing CMOS, the wire length became more significant and so we would find that it wasn't converging after two iterations. Then this very smart guy, still one of the smartest people, David Hathaway, came up with incremental timing analysis [20]. So, instead of doing that, why can't I do timing analysis off the same graph that we're doing the synthesis on? And, in addition, when I make a change to a circuit, I don't have to recalculate the whole of the timing, I'll just figure out what changed, what's the new timing number. So that allowed us to do an experiment. We could say, we think it's better to put these two gates in instead of this one. Try it. Timing, oops, not so good. Let's back that out. That was a major breakthrough. That really allowed us to not only run faster but also to deal with wire delays and it kind of got us into the CMOS, helped us with CMOS. So then IBM finally had a workstation about that time, the R6000. I think in 1989, there was a decision, "Let's rewrite LSS." This thing was running on a mainframe, right? Written in PL/1. Let's move to the workstation, write it in C code, and really rethink it, sort of a clean sheet of paper approach. Louise and some of the people from Rochester were involved in that and really did a great job. They came up with a much better system and go back and had a chance to really have the benefits of local transforms and some global optimization. So that became known as Bulldozer [21]. Up until then, every tool had a three-letter acronym and now this was Bulldozer. I didn't

understand why they would call it Bulldozer but everybody loved the name and so Bulldozer is still around. It's still working. When wire length became so important, as we got to finer lithography, that placement had to be incorporated. So I think, again, IBM was the first to do Placement-Driven Synthesis, PDS [22] it was called. That was built on Bulldozer. Then what's going on now is that IBM went through a period of custom design and we're now sort of going back now and Ruchir Puri is leading the effort and seeing how synthesis can actually compete with custom designers in a high performance server, very high frequency servers [23]. So synthesis is still a very active and, I think, cutting edge research project in tight cooperation with the central design group. So they're doing remarkably well, very, very high degree of coverage and building five gigahertz plus parts for high performance servers. It's really quite impressive. So it's still going on.

Fairbairn: Okay. Let's take a break.

[audio off then on]

Fairbairn: Okay. So we've been exploring this area of logic synthesis and the impact that it had within IBM in combination with some other important tools, static timing verification and cycle-based simulation. Tell me again what year that sort of all came together. How did your career track that and then what did you move onto next?

Darringer: So the 3081, which was in the late '70s, was the first machine that brought together these static timing analysis, cycle-based simulation, Level Sensitive Scan design, Boolean equivalence checking, flowchart specification, all of that was sitting there. Then we added synthesis in '78/'79 into the early '80s. So that showed up in the next machines, I think the ES9000 I think was a machine that was heavily synthesized. We did parts for the 3090, which had 612 circuits. Then, after that, I would say, as you go to the-- by the time we get to the late '80s, '86/'87, something like that, I think every chip was going through LSS.

Fairbairn: So how long were you personally involved in this software?

Darringer: So I had managed the project for four years. Then, after four years, I think I was asked to manage a bigger group. I picked up the RISC microprocessor development. We had started the RISC design, the 801. That group was there so I was asked to pick that group up and I think, looking back, it was probably because I had figured out a way to work with the development divisions and get research skills and talent connected so they thought, "That's a good thing, let's let this guy have a little more responsibility." Then the next thing that happened, fairly quickly-- so I did that for about two years.

Fairbairn: What years?

Darringer: So '78 was when I started the Logic Synthesis Project so until '82. So, from '82 to '84, I managed these two groups. Then, I think, in '84...

Fairbairn: The two groups being?

Darringer: Design automation and research’s part of design automation and research's work on the RISC microprocessor.

Fairbairn: Okay.

Darringer: Then Irving Wladawsky-Berger was our higher level manager. I don't know if he was a director or not but he was a high level manager. He asked me to take on large systems because he was getting promoted. So he was asking me to manage the large systems. Now, large systems research, what that means, this is a group of some IBM fellows, very smart people who think about the next generation mainframe computers and they actually have a laboratory, a very large, they probably have the largest IBM machine, current machine fully configured where they would run experiments, capture traces, do experiments and sort of plan for what's the next machine going to be like and then try to explore alternatives and work with the Poughkeepsie guys on that. So, to me, it was a brand new area. I knew something about the design of large systems but I hadn't paid much attention to architecture. So I was a little apprehensive about taking on this job but Irving is a very persuasive guy and he assured me that this would be fine for me and that I could handle it. So I listened to Irving and I took the job. It was just a great experience because, you know, it's just moving into a whole new area, learning all kinds of new things and meeting new people and learning more and more about how large systems are designed. So I really liked that job. I did that for two years. Then they said...

Fairbairn: Any major new software developments during that time?

Darringer: Well, one thing I did was we had a very good connection with the divisions on the mainframe but the midrange computer that was built in Rochester, the AS400, that sort of machine, they didn't see the benefit of research. They didn't think they needed any help from research. So one of the things I was able to do was show them that, by using similar techniques, similar approaches that we were using for large systems and we looked at the midrange computers. We could help them. And they were skeptical at the beginning but, eventually, I won them over. So we set up what was called a joint program where they fund things, we fund some stuff and we work on stuff together and so that was a big step because now research had a joint program in the midrange. At that point, I think they said, "Well, this guy seems to know how to do this stuff" so they moved me to director of large systems and midrange and microprocessors and the design tools. That was about 200 people at the time, researchers.

Fairbairn: Was there much commonality in design tools across this?

Darringer: The way it was at that time was there was a lot of- each location, so Poughkeepsie kind of had their own design methodology, their own tools, supported by the central group. Rochester had a different methodology supported by the central group. The PC group had their own way of doing business. The workstation guys definitely had their own way of doing philosophy. So there were a lot of dotted lines bringing it back to one central design automation group but there was a lot of separate development aimed at the specific processor and different methodologies, different contexts so you had to work with each of these different groups. At that point, so this was 1989, I think, I was, again, in large systems and worked with large systems and what you're doing there is not so much managing individual projects. You're more trying to make sure you've got the people with their skills matched to the projects and trying to get the projects matched with the needs of the company. So you're really thinking about it that way. So I spent a lot of time on IBM strategy, research's division strategy and Ralph Gomory was the head of research at the time. Ralph was an amazing person and had a very disciplined approach. He had been in that job for 19 years so he had a very disciplined approach for managing research and a specific philosophy for doing it. We kind of knew that and I liked it. I thought it was great. So, in 1989, Ralph decided to retire and John Armstrong was going to become the new director of research. John

Armstrong is a physicist so not from the systems background. My boss was in computer science, was Abe Peled was the head of computer science, he was a little nervous about the new guy in charge of research not knowing much about systems so he suggested, "Why don't we have Darringer go over and be the head of staff and work with John and make sure he understands things." John liked this idea so I took this job, it was called Director of Technical Planning. This was a staff position. You have staff people reporting to you, one from each of the major areas, including Zurich and Almaden. So this was probably the neatest job because you now get visibility to all of the research projects going on. There was about 3,000 people spread around the world and we would go and visit them and review their projects. We were responsible, the staff and I, were responsible for a 10-year outlook, which I should explain to you, the technical plan, which was a written English document that described every project. Ralph insisted on that and read it in front of you. So we would prepare that for him and work with the directors. There were no vice-presidents.

Fairbairn: How many such plans were there?

Darringer: Every year, we'd have a plan. The plan would be of the order of less than 200 pages, more than 100 pages, something like that. It would describe all the work of the 3,000 researchers. Ralph had a very strict syntax. There was an environment section where you would describe the competitive landscape that you were operating in, what's going on. Then there would be a strategy section where you would say, "What is it? What's special? What's really your thing that you're going to bring to this problem?" Then there was the plan, "Who's going to do what?" It had to be, you know, a project could be just a half a page. It was very small, very compact. We spent a lot of time training managers to separate strategy from plan and environment from-- but I found it- most managers, I think, found it tedious. "Why do we have to do this?" Ralph felt it was the way he really made sure he understood what was going on. And, for the staff, he helped develop this so you really had visibility to all of the plans. It was just super. Another probably even one of the neat things we did was the ten-year outlook. Again, one of Ralph's ideas. Stretch the researchers and make them try to think what things are going to happen in ten years. Most people said, "Well, come on, I can't predict one year, how would I do ten?" But Ralph would encourage you to do that. For technology trends, it was a little easier, you know? Semiconductor is probably going to go on for another ten years, disk drives are going to get smaller, things like that. But he would push us into other areas. "What's going to happen in the business areas?" "What's going to happen with displays?" Things like that.

Fairbairn: What level would that get pushed down to? Did senior managers just try to crank this out or did they get the bright visionary researchers to come up with it?

Darringer: It was a mixture of that. There was sort of a- you would sort of look at last year's and try to update it but they were always searching new topics. What's the new things that we hadn't even thought about? Let's get that in. So that was kind of a bottoms-up and it was an invitation for people to propose ideas that would make it into the ten-year outlook. The ten-year outlook continues today. It's called the Global Technology Outlook and it's much more, today, we're going to focus on business opportunities. It still includes technology. When I was doing it, it was very heavily focused on where is technology going and somewhat on the business opportunities. So it's still a very, very important part of the IBM planning process. This is reviewed by the CEO and the Corporate Management Committee. They review that. Ralph presented that to them every year.

Fairbairn: Was there any time when something came up in that process that really changed the direction or got IBM going in a certain way? I'm sure, over time, it influenced things but was there any sort of major ah-hah out of that entity?

Darringer: Yeah. I think-- since I was only involved in it directly for a couple years, but I would think recognizing that certain businesses are going to become commodity businesses, I think, displays. It was very early that we saw that, wait a minute, I can keep putting more and more pixels on the display but where is that leading us? So that thinking led to getting rid of the display research group in research, which surprised a lot of people. They just said, well, "It's not going," we're not going to do that any more. So, at the time, we had the highest resolution display in the world, you know? That stopped. Same with disk drives. I think we kind of predicted what was going to happen with the disk drive business, the head disk assembly and IBM recognized that's becoming a commodity business and decided to get out of it. So those are kind of consequences, I think. Externally, it seemed like people were surprised by disks getting smaller and were fighting that, saying, "Oh, no, these little three inch disks can't possibly compete with..." and IBM saw that coming and predicted it and planned for it. So I think it helped.

Fairbairn: I'm sure it did. I was just curious...

Darringer: Yeah, yeah. That's a good...

Fairbairn: ...little anecdotes or...

Darringer: That's a good question, good question. So, anyway, it was a lot of fun to do that. At the end of that, that was about two years as director of planning. Then, at that time, so I'm looking for something new to do and Bob Corrigan was the vice-president in the technology area and I think president of his division and was looking for someone to help him restructure the design automation group. Since I had worked in design automation, they thought, "Well, maybe this is a guy" so I accepted that job as director of electronic design automation. I changed it from electronic design system to electronic design automation, made a little name change. At the time I took over the job, it was about roughly 600 people, something like that, but almost all dotted lines. As I mentioned earlier every hardware group had their own philosophy, their own set of tools so there wasn't much synergy. We had five static timing analyst tools, we had seven logic simulators, we had three completely different set of physical design flows. This was just not very efficient and so Bob wanted it fixed. So that was my job. That wasn't easy because these groups all reported to, I think, six different executives in four or five different divisions and so I'm telling them to give me their tools group and let me manage them, some guy from research.

Fairbairn: ...eliminating...

Darringer: <laughter> But I managed to convince them and we developed a strategy. Another thing that was going on, at that time, all the tools were written on mainframes. At that time, Daisy...

Fairbairn: Mentor.

Darringer: Mentor and those Valid tools, workstations were popping up in IBM in research even and were incompatible. You couldn't send data from one to the other. They couldn't scale. IBM builds very large things and these things just couldn't seem to handle the big designs. So, "Darringer, what are you going to do about it? We need to get our tools over on workstations." Fortunately, we had the RS6000. It was at least coming. So that was another second thing. Consolidation, move to the workstation and also converge the methodologies. Now, we didn't think we could have one methodology but what I did was I tried to integrate the tools into little benches. So we had a logic bench for the front end work, we had a chip bench for doing the back end work, test bench for doing testing, circuit bench, you know? So the idea was that these benches could be put together for different people. That was the philosophy. So we worked on that for quite awhile and did manage to get almost everybody together in one solid group and become much more efficient. There's a great synergy between mainframe design and ASIC design. ASIC, automation is, of course, vital and performance is not the most important thing, although IBM builds very large and high performance ASICs. In the mainframe, performance is everything and automation is not the most important thing. So, by combining the two, we could bring some of the automation techniques from ASICs into the mainframe and some of the high performance techniques over to the ASICs. I liked that. I liked that, trying to balance those two. IBM was, as I think I maybe mentioned earlier, IBM had done master slice. Then we did CMOS, still gate array CMOS. Then they decided that, really, to be competitive, they had to start doing custom circuit design so they brought in from outside a whole bunch of people who know how to do custom layout by hand.

Fairbairn: You're talking custom physical layout?

Darringer: Right. Guys who would sit down with workstations and layout physical...

Fairbairn: Geometric layout?

Darringer: Yeah, geometric layout. By hand. Give them a schematic and they would lay it out. We had a lot of people brought in to do that hopefully to raise frequencies and make things more efficient. So that was a little bit of a setback for automation, right?

Fairbairn: What year was that?

Darringer: This would be in the mid-90s.

Fairbairn: By then, everybody's going the other direction...

Darringer: Yeah, I think...

Fairbairn: Automation was everything, right?

Darringer: I think, outside, yeah, automation was kind of-- well, we had ASIC automation, we had master slice automation. What we wanted to be sure was we weren't leaving any performance on the table. So these circuit guys came in and taught us a lot. Now, I mean, if I jump to today, what's happening is automation is coming back. So now mainframe custom design is going down dramatically and being replaced with, we think, equal or better design that's being generated automatically with very extensive synthesis that involves placement, clocking, wiring, congestion avoidance, everything.

Fairbairn: Too complex to manage by hand and...

Darringer: Exactly. There's so many issues that it's tough for a human being to deal with it. The tools are doing well. So back to the large systems area- I mean the EDA, yeah, that was...

Fairbairn: So you integrated and optimized and made it much more efficient in terms of numbers...

Darringer: Right.

Fairbairn: ...of people and projects.

Darringer: And trying to figure out how to work with some of the outside vendors. Why is it that we're doing everything? Can't we look outside? So we tried that. I don't think that was so successful. We ended up sort of, even today, we largely still stick with internal tools. I think it's that maybe IBM tries to be on the leading edge of technology, tries to build these very large systems and, by having their own internal team, they can really focus those guys on that problem. So, so far, that's still the way we're going at it. It's always an ongoing thing. We're constantly reevaluating that and maybe we should switch to the vendor tool for this or that.

Fairbairn: So this is the mid-'90s. You're doing...

Darringer: Right. Mid-'90s and so that went on until sort of 2000 and then I came back to research. So, at that time, I think the system technology group, the processor designer, the system design group

wanted to have more influence on design automation. Design automation has been- there's the technology guys and there's the system guys, right? There's kind of a tug of war there. Design automation usually had been with the technology guys and they felt maybe it should be closer to the processor guys. So the fellow that took my place, Jim Dickerson, was from the processor area and then I sort of helped him in transition a little bit and then I decided to go back to research and what I wanted to do was I thought, if you look at it, IBM is very strong from register transfer level to masks. I mean, the design animation flow is very good there. It's very good there. It's very challenging, very demanding, constantly being involved and innovated but the place where I thought we could do a lot more is above RTL, at the systems level, even for servers. I know outside IBM, there's a lot of system level design tools and methodologies developed to support system on a chip design. But I was thinking of, even for server design, there's a place where maybe research could help. So I've been leading projects there in that area, trying to figure out can we do high level synthesis for servers? Can we do automated performance analysis for new configurations? One of the things that makes that difficult is that IBM has, like a lot of companies, has experienced people, know system design and they've been doing it for decades. So, when I go to them and say, I want to help, tools to offer you, they kind think, "Well, I'm not sure I need any help here. I think we're doing fine here, young man." What's helped, though, is the move to three dimensions, 3D design. So IBM is looking very hard at 3D technology and that changes the playing field, right? Interconnects now can be much, much shorter. There are thermal issues and the design team, the architects, don't have this decade of experience dealing with that so there's a lot of questions.

Fairbairn: Can you describe what you mean by 3D technology? What exactly...

Darringer: Right. 3D covers a lot of space. There's people who've been stacking chips for a long time. The thing that IBM- and IBM has been stacking chips but the thing that I'm talking about is where you have layers, silicon layers that are manufactured separately and then put together and connected through silicon vias that go all the way through the silicon. You can stack two, three, four, five of those up. In technology demonstrations, we've shown four layers stacked together, connected together. The question is, what am I going to do with that? It's sort of obvious that the interconnect changes dramatically so that's an opportunity. Then the power you need to drive, since the inter connector is shorter, you can save a lot of power by not having to drive it so that's an opportunity. There's a thermal concern that, if I put active elements on top of active elements, maybe I'm going to have a problem I can't deal with. How do I cool it? Also, power delivery. If we're already at the limits of delivering current to the chip, and now we're going to try to stack up, how in the world are we going to get current into that collection of chips? So I think a lot of architects are wondering, what is this? Where is the real advantage here? How can I really get some advantage out of this? What I'm trying to do is build some tools to help them explore that space by giving them much more rapid analysis of possibilities and also much more detailed results so they can see some consequences they would miss if they just did it on a spreadsheet or something [23]. That's a lot of fun.

Fairbairn: So that's what you're involved in now, is that right?

Darringer: Right. Well, I mean, I could mention that's one of the projects. We have another project called Beam [24]. Beam is a static analysis tool for software. It just happens, because I'm interested in it, this project was started in 1996, a fellow, Dan Brand, again, a friend of mine, was writing C++ programs and was making errors. So he said, "Well, maybe I could write a program to help me find my errors" and so he wrote a program to look for problems. This program became popular among his friends, who were also writing C++ programs so he had a little following. But he didn't really go anywhere so, when I came back to research, I saw this and I thought, "Oh, I really like this program." One of our product groups was having some quality problems with their microcode and I thought maybe we could apply this tool to that microcode and it required we had to build a C front end to do it. We did that and got one of the product managers just extremely excited because we immediately found 100 serious errors in his microcode. He just, "Oh, this is the greatest thing in the world," you know? It would have taken me ages to find those errors. You found them in just one run. He was an evangelist for our tool and he used it. Word spread and all of a sudden we were doing microcode and microcode and microcode. So, today, all microcode in IBM is run through this tool, all the operating systems, much of their system software is run through it. It runs...

Fairbairn: All the O/S code?

Darringer: Yeah. Well, everything that's written in C, C++. There may be still some stuff in Assembly language somewhere that we don't deal with. Microcode is written in proprietary, you know, internal language and we process those languages. So this is another tool that's had a big impact in IBM and I kind of help get it connected to the product groups and help to promote it. Dan is the guy that really makes it work and we have some people in the central EDA group who support this. So it's interesting because here we're getting back to tools to support software. I was probably right, software is really very, very important.

Fairbairn: Software is the problem.

Darringer: Right. So we're looking at security, checks for security. It's a lot of fun. So it's another project in our area. We have a guy doing formal methods for protocol verification. We have a couple people working on the design of a very, very low power core for exascale computing. So that's really- you're really stretching everything we know how to get the power down on these cores and trying to see if high level synthesis might play some role in there.

Fairbairn: So speaking of cores, people have now started creating even single chips with four, eight, 16 cores and then computers are becoming more parallel and more massively parallel yet the software remains the problem, it seems like, in terms of being able to effectively-- do you have any comments, insight, interesting research?

Darringer: I mean, I think everybody realizes this is a big problem. So a couple of things. We have, again, a few people in research. It only takes a couple of people so we have a few people in research who are looking at restructuring EDA applications for parallel computing. One or two of those people have really some good ideas so we don't have a complete process yet but we're, like a lot of people, working on how do we do that.

Fairbairn: Looking at individual applications, how to get this application on parallel.

Darringer: Trying to learn, take specific applications, let's see what we can learn from that and let's look at another one and then let's look at another one and see if we can generalize but we're focusing on EDA applications. We just had a workshop at IC CAD on parallel computing, how do we move EDA applications into this environment? But it's a problem I think the whole community is working on. It's important.

Fairbairn: So your career has spanned this whole, huge number of things from detailed work and program verification and so forth to general issues about what is IBM as a major, worldwide corporation do. I'd like to talk a little bit about the future of some of those spaces and where you think the interesting challenges are for new people coming into the business or for companies to further explore or money to be made or whatever. Let's just focus on, since you just came from ICCAD, talking to people, doing research in this space, where do you think the interesting work and challenges are going on in that space today? Then we'll talk about some of the other areas. Let's focus on ICCAD. What do you find the most interesting? Where do you find the biggest challenge? What's the biggest roadblock?

Darringer: It's truly, it's probably the software area. There wasn't much of that at ICCAD so, if I think about...

Fairbairn: That's all right. Let's...

Darringer: No, if I think about what I saw at ICCAD that I thought was really interesting, it's 3D. Maybe it's because we're currently working on that but the thing I think about 3D is it's an opportunity to bring design automation to the early concept phase and we call it HLD, High Level Design Phase. I know what's going on in the SoC (System-on-a Chip) space and that's fine but I'm talking about server design. I think though that, in that space, there's an opportunity to bring modeling, more abstract modeling to the early phases and help these architects think about what machines could do with more rapid turnaround, more detail. So I'm excited about that. Now, that then drags you into current analysis so we're doing--here we are, we don't have any implementation but we're trying to predict how much current is going to be flowing through these C4s to feed this chip and I think it's really neat. People would know it's a problem but what can I do to let you think about it and reason about it very, very early in the process? So we have to make assumptions back to this. So, anyway, I like that area and I think that's a good area for research,

tied to 3D. There are a lot of university people working on exploring this space. On the software side, one of the things that I see as heterogenous multi-core systems, right, accelerators so we're seeing it already. There are companies making appliances that you can attach to a mainframe or a system and offload a part of the business to that, much more efficient, much more rapid. So where does that lead? How many accelerators do I have? Where is the best to put them? How do I program the accelerator? Is it an FPGA or is it somehow programmable logic or what is that? I think that's going to be a neat area. IBM just announced a mainframe called the Enterprise Mainframe, I think, but it's a hybrid system. It has a mainframe and a rack where you can plug in Intel processors and Power processors...

Fairbairn: Blades.

Darringer: Blades. The idea is that so, yes, we're going to let the mainframe manage the data but offload applications to Blades for certain applications and bring the data back. Then provide you an environment to manage the whole thing rather than have a loose collection. I think that paradigm is going to, in the future, I think that's going to be-- how do I decide how to offload what application and what do I put it on? I think that's a great opportunity for EDA and it's not just hardware, it's hardware/software and that's a thing I think EDA should get into is this hardware/software co-design. So that would be an area I would want to look at and we are looking at.

Fairbairn: So if you go back to your alma matter, Carnegie Mellon and talk to the computer science students who are trying to decide on PCs and so forth, are those some of the areas that you have-- what would you tell them are the exciting areas in...

Darringer: I think, when I do talk to people, one of things I really do get excited about is IBM has this smart planet strategy. People may, at the beginning, say, "Well, this is just a marketing campaign" but if you really look beyond that, I don't think it is. I think they've really come on with something and the thought is so how do I take computation and apply it to some of the world's biggest problems? How do I make better transportation systems for cities and countries? How do I distribute power? How do I make businesses run more efficiently? There's a revenue opportunity, there's business opportunity there for IBM so, of course, but it's also a good thing and I think it's really exciting. So what's happening in research is you see researchers working directly with countries and cities and banks and they're all over the world. A friend of mine just came back from Brazil where he was thinking about how can they help that country improve their support or their infrastructure? I think there's just so much opportunity there. Even in our group, this power grid, how do we make more efficient power grids? So there is a connection maybe with-- we know how to do very efficient power grids in a chip and there's a couple guys who think there's a connection there, that we think some of the simulation modeling work we do there could apply to big power grids. So we're actually working on that and we have a proposal to go and develop that. So I think this is really exciting because there's just unbounded opportunity. It's serious, it's big revenue opportunities for IBM and other companies and it's fun, satisfying to work on. So I would think that's a great place to go work.

Fairbairn: So you've had an incredible career working in many different levels of technology, as well as within the company of IBM. Is there any particular one you look back on or a couple you look back on and say, "Boy, that had the biggest impact or the thing I had most fun at?" What were the things that, in your career, really stick out to you as being the things you're proudest of or happiest about or whatever description you want to use.

Darringer: Yeah. It's a good question but it's a hard one to answer. I think I've been fortunate to have a series of really great jobs. Working on synthesis was a lot of fun because it had a big impact and it's still going on today and the team is just still having a tremendous impact so that was good. This director of technical planning, having visibility to all these different kinds of research in IBM, I really liked that. The challenge of trying to get this large organization for EDA together and operate more efficiently was very satisfying. And being back in research and having fun on a more individual basis working on these projects also is a lot of fun. Another thing we didn't mention at all was outside IBM. Let me just mention one thing. When I was working in EDA, one of the major themes was integration, especially around synthesis. So we initially figured out that, when I brought synthesis and incremental timing together, had a tremendous impact, maybe a 10X improvement by having incremental timing. Then the next thing was placement, we brought in placement together. Now you could do incremental placement and get the logic set up for final placement, big impact and less iteration in the design. Now we're bringing in wiring and avoiding congestion. So that theme, we knew that was going to happen and predicted it in the early '90s. If you would look at the strategy charts that I would present in the early '90s, that would be the main thing we're going to say is we're going to have a data model inside and IBM does have a data model inside. And we're going to integrate these tools there and make them operate incrementally. So we believed in that very much that that was the key. Now, we also said we don't really want to make all these tools ourselves, we want the vendors to participate in this. So we started campaigning to have an industry standard interface and we actually proposed our model as a standard but IBM really didn't want to let it go. This was an opportunity, Cadence came in and said, "Ah, we like the idea, we're building a new data, the Genesis data model, we'll contribute that, open source." So that was a big shift in the industry. So I was part of SI2 that kind of helped get that going and pushed on that really hard and got Intel, LSI Logic, HP, many, many companies saw the advantages of this approach and got on the bandwagon. Then I organized a set of workshops at DAC, interoperability workshops, and I remember the first workshop. We got up and we had Greg Spirakis was there from Intel and we had a guy from LSI Logic and HP was there and we got up and said, "The key to the future is integrating around this data model and you guys have got to have a standard." Nobody in the room knew what we were talking about. They just- you could just see the blank look on their face and, after the meeting, "What are you talking about?" Then the next- we did it the next year and the next year they said, "Okay, I understand what you're doing but there is no way this is going to happen. You're never going to get vendors to agree to an interface on their data. This is just not going to happen." So we kept this up and, for five years, the fifth year, I said, "This is it. This is the last interoperability workshop" because I could see that it's happening. It's just going to- you can't stop it now. It's going to happen. And then, sure enough, it's taken off. So, anyway, those are things that I got involved in outside IBM. I spent a lot of energy on that and I'm happy to see that it took off thanks to the work of a lot of other people. Thanks to Cadence for supporting it so much.

Fairbairn: So one final thing. We were talking earlier that this interview, several hundred are being collected by the Computer History Museum to try to capture the personal stories and the facts behind some of the major developments in computing and technology over the last 50 years or so. I'd like to get your- if you have some sort of personal feeling about that, the importance of that or what benefit that can play to the industry and to people looking at possible career opportunities and so forth. Do you have any comments on that?

Darringer: When you first approached me, I hadn't thought about it a lot but, in the process of thinking, getting ready for today, I did have a chance to think about it a little bit and I think it's a great thing you're doing. I really think this is going to turn out to have a very big impact. I think just the raw data, by itself, is not going to be so helpful to a large number of people but it's going to provide kind of the raw material for other people to come in and analyze and develop interesting conclusions and stories from so that I look forward to. And I look forward to looking at the material myself. So I think it's a great thing. Also, the museum itself. I haven't had a chance yet to visit it but, from what I've seen about it and heard about it, I think it's a tremendous thing. As I mentioned, I have grandchildren and my grandchildren are going to museums and are very impressed with what they see. My grandchildren haven't been to the Computer Museum. I would like them to see that and get them even more interested in science and engineering. I think the museum's doing a great thing and I think these histories are going to help us look back, see what some of the trends were that we didn't see at the time and also hear the stories where individuals made a difference with just some simple action. Sometimes when you see something happen, you think it must have been a very large set of people that did that. It turns out often it's just an individual, one person got it started. That can be inspirational to other young people to think, "I can attack these problems, I can make a difference." So I think that'll be great.

Fairbairn: Okay. I think that's a great point to end and I appreciate the time and energy you put into this. I think the career and scenarios that you described here are just tremendous additions to our databank of knowledge about the history. Thank you, John Darringer.

Darringer: Thank you, Doug, for inviting me. This has been fun.

Fairbairn: Okay.

END OF INTERVIEW

REFERENCES

[1] “EULER: a generalization of ALGOL and it formal definition: Part 1”, Niklaus Wirth, Helmut Weber, Communications of the ACM, Volume 9 Issue 1, Jan. 1966
[2] “A Digital System Design Language (DDL)”, Duley, J.R.; Dietmeyer, D.L.; IEEE Transactions on Computers, Volume C-17, Issue 9, September 1968.
[3] “MAPLE, A High Level Language for Research in Mechanical Assembly”, J. A. Darringer, M. W. Blasgen, IBM Research Report RC 5606, September 1975.
[4] “A program verifier”, J. C. King, PhD Dissertation, Carnegie Mellon University, Pittsburgh, PA, 1969.
[5] “Applications of Symbolic Execution to Program Testing”, J. A. Darringer, J. C. King, IEEE Computer, Vol. 11, No. 4, April 1978.
[6] “The Application of Program Verification Techniques to Hardware Verification”, J. A. Darringer, 16th IEEE-ACM Design Automation Conference, June 1979. Also reprinted in the book “25 Years of Electronic Design Automation” by A. Richard Newton and Bryan Preas, ACM Publications, 1998.
[7] “Design Verification System For Large-Scale LSI Designs”, M. Monachino, IBM J. Res. and Dev., Vol. 26, No.1, January 1982.
[8] “Method of Level-Sensitive Testing a Functional Logic System”, E.B. Eichelberger, U.S. Patent No. 3,761,695, September 25, 1973.
[9] “Timing Analysis of Computer Hardware”, R. B. Hitchcock, G.I. Smith, D.D. Cheng, IBM Journal RD, v26, No 1, January 1982.
[10] “Hardware design and description languages in IBM”, L. I. Maissel, H. Ofek, IBM Journal of Research and Development, Vol. 28, No. 5. September 1984.
[11] “Boolean Comparison of Hardware and Flowcharts”, G. L. Smith, R. J. Bahnsen, H. Halliwell, IBM J. Res. Develop., Vol. 26, January 1982.
[12] "Customized Metal Layers Vary Standard Gate- Array Chip," Pomeranz, R. Nijhuis and C. Vicary, Electronics, pp. 105-108, March 15, 1979.
[13] “MINI: A Heuristic Approach for Logic Minimization”, S.J. Hong, R.G. Cain, D.L. Ostapko, IBM Journal of Research and Development, vol 18, No 5, Sept 1974.
[14] “A New Approach to Logic Synthesis”, J. A. Darringer, W. H. Joyner, Proc. 17th Design Automation Conference, 1980. Also reprinted in “Selected Papers on Logic Synthesis for Integrated Circuit Design” by A. Richard Newton, IEEE Press, March 12, 1987. Also reprinted in the book “25 Years of Electronic Design Automation” by A. Richard Newton and Bryan Preas, ACM Publications, 1998.
[15] “LSS: A System for Production Logic Synthesis”, J. A. Darringer, D. Brand, W. H. Joyner, L. Trevillyan, J. V. Gerbi, ACM Computer Science Conference, 1985. Also republished in special retrospective issue: “Evolution of Information Technology 1957-1999”, IBM Journal R&D, Vol. 44, No. 1/2, Jan/Feb, 2000.
[16] “The Decomposition and Factorization of Boolean Expressions”, R.K. Brayton, C.T. McMullen, Proc ISCAS April, 1982.
[17] “Logic Minimization Algorithms for VLSI Synthesis”, Brayton, Robert King; Hachtel, Gary D.; McMullen, Curtis T.; Sangiovanni-Vincentelli, Alberto L;, Kluwer Academic Publishers, ISBN 0-89838-164-9(1984).
[18] Automated technology mapping, J. L. Gilkinson, S. D. Lewis, B. B. Winter, A. Hekmatpour, IBM Journal of Research and Development, Volume 28, Number 5, 1984.
[19] “Redundancy and Don’t Cares in Logic Synthesis”, D. Brand IEEE Transactions on Computers, Vol C-32, No. 10, October 1983.
[20], “Incremental Timing Analysis”, R.P. Abato, A.D. Drumm, and D.J. Hathaway, L.P.P.P. van Ginneken, U.S. patent 5,508,937, 16 April, 1996.
[21] “BooleDozer: Logic Synthesis for ASICs”, L. Stok, D.S. Kung, D. Brand, A.D. Drumm, A.J. Sullivan, L.N. Reddy, N. Hieter, D.J. Geiger, H.H. Chao, and P.J. Osler, IBM Journal of Research and Development, Vol. 40, No. 4, July 1996.

[22] “An integrated environment for technology closure of deep-submicron IC designs”, Trevillyan, L.; Kung, D.; Puri, R.; Reddy, L.N.; Kazda, Design & Test of Computers, IEEE Volume: 21, Issue: 1 2004.
[23] “Design methodology for the IBM POWER7 microprocessor”, J. Freidrich, et. al., IBM Journal of Research and Development, Vol. 55, No. 3, MAY/JUNE 2011. [24] "Early Chip Planning Cockpit", Jeonghee Shin, John Darringer, Guojie Luo, Alan J. Weger, and Charles L. Johnson, Design, Automation and Test in Europe, Grenoble, France, March, 2011.
[25] Evidence-based analysis and inferring preconditions for bug detection. D Brand, M Buss, V C Sreedhar, IEEE International Conference on Software Maintenance, 2007.

