Oral history of Robert P. Colwell (1954-)

Interviewed by Paul N. Edwards, Assoc. Prof., University of Michigan School of Information, at Colwell's home near Portland, Oregon, on August 24-25, 2009

Robert P. Colwell is an electrical engineer. His PhD work at CMU (1980-1985) demonstrated the theoretical superiority of CISC architectures over RISC. From 1985-1990 he worked at Multiflow, an innovative developer of "minisupercomputers" using a VLIW (very long instruction word) architecture. In 1990 he joined Intel as a microchip designer, serving as the chief architect of Intel's IA32 line from 1992-2000. He left Intel in 2001. Colwell recounted his experiences at Intel in his book The Pentium Chronicles: The People, Passion, and Politics Behind Intel's Landmark Chips (WIley-IEEE, 2005). Since retirement, he has been a prolific columnist and a consultant on legal and patent issues.

This oral history generally follows the chronology of Colwell's life and career (see table of contents), with occasional digressions. The interviewer chose to avoid the areas discussed in Colwell's book, which is semi-autobiographical.

Some of Robert P. Colwell's major achievements:

- Senior CPU Architect, Intel Corporation, Hillsboro OR, 1990-1992
- Chief IA-32 Architect, Intel Corporation, Hillsboro OR, 1992-2001
    - Lead IA32 architect, responsible for all of Intel's Pentium CPU architecture efforts
    - Initiated and led Intel's Pentium 4 CPU development
    - One of three senior architects responsible for conceiving Intel's P6 microarchitecture, the core of the company's Pentium II, Pentium III, Celeron, Xeon, and Centrino families
- Intel Fellow - 1997
- 2005 Eckert-Mauchly Award from ACM: for "outstanding achievements in the design and implementation of industry-changing microarchitectures, and for significant contributions to the RISC/CISC architecture debate"
- 2006 IEEE Fellow and the National Academy of Engineering for "contributions to turning novel computer architecture concepts into viable, cutting-edge commercial processors"

Interview contents

Part 1 (1954-1980): Overview, Early Life, Higher Education, Bell Labs

Overview

0:00:01 Paul Edwards: This is Paul Edwards interviewing Bob Colwell... Robert P. Colwell at this home in Oregon on August 24, 2009. And Bob, just to say a few words about your past: among many other honors and achievements, you were an Intel fellow in 1997. In 2005, you received the Eckert-Mauchly award from ACM for outstanding achievements in the design and implementation of industry changing microarchitectures, and for significant contributions to the RISC/CISC architecture debate. And in 2006, you were IEEE fellow, and also a fellow of the National Academy of Engineering for a contributions for turning novel computer architecture concepts into viable cutting edge commercial processors. You were the chief architect of the IA32 processor line at Intel from 1992-2001 and also worked at Intel from 1990-1992. Say a little bit for yourself of what you think of as your major achievements.

0:01:38 Bob Colwell: Technical achievements because I think there's other things. I would say, I would think, if I had to point one single thing, it'd probably be the P6 microarchitecture that we did at Intel. Because we did that with an eye towards a long term contribution to the company as opposed to “let's do a chip and after that let's do another chip”. You might think there wasn't much like that at Intel because they had this long history of X86 chips, like the 8086 and the 80186, 286, 386, 486 and then, they did the Pentium and you might think, "Oh, that's evidence of a nice long term view of where they're going." That's probably more misleading than correct. It was very much one chip at a time, expecting that each new chip might be the end. It's kind of how those guys viewed it.

You mentioned the RISC/CISC debate a few minutes ago. That's related to this because Intel was the canonical CISC place, and they weren't doing heck a lot of research, they were just turning out commercially successful microprocessors. But they were aware of what's happening in the field at Intel back then and they were seeing that one heck of a lot of publications were saying RISC is great and CISC is going to die. And if your product line is heavily CISC based, you have to pay attention to talk like that. So there was a fair amount of consternation in the late 80s bordering on almost panic at times wondering, what's Intel going to do. I joined Intel in '90 and I just kind of walk right into that and listened to these people and sort of frightening themselves to death about what X86 is going to go. How can we not die? The RISC guys are going to kill us and so on.

So when we did the original P6, we did that with an eye towards finding a microarchitecture that can serve as a basis for a long time, not just a single chip in 1995 and then we start over. And in doing that, we had to do things for example, like boosting floating point. The computers back then were measured basically on how good are they at integer performance and how good are they at floating point? And X86 was recently good at integer, but it was lousy at floating point, and it was partly because of a legacy decision made in the early 80s. It was pretty twisted up. It made it very hard to do fast floating point in X86, but we purposely chose to boost that, expecting that if we were successful at the microarchitecture then, that balance would come in handy. And if we failed to do that, we would end up with a chip that couldn't go anywhere because it wasn't good enough to sort of hit all the bases that needed to hit. They're still designing chips with big pieces of P6 lore embedded in them, so I think that's pretty cool.

0:04:55 PE: I was struck by this, reading through your papers and your book that the RISC/CISC did they, it carried right through your entire career as one of your first published papers, was about the RISC/CISC debate. I'm wondering if you think that it could be said that you basically settled that issue.

0:05:15 BC: Well, every time I would start to think that, I would run across somebody from the other side who is by no means a convert to my side. So no, there's still plenty of room to, sort of, have different opinions in that general area. A lot of significant work got done, and so anytime, you take an extreme position like the CISC one and RISC lost, it just devalues a lot of really important work. I've come to sort of see that, after all these years, although I have to say, back when the battle was raging in the 80s it was pretty much you're with me or you're against me, and conferences were pretty exciting to go to. I mean, this step was held very strongly by people. People that I didn't even know, once they read my name badge would turn and walk away. They're just very angry that I wasn't signing up for the RISC orthodoxy, or I was questioning anything associated with it and it got pretty personal. One saving grace I think, at least from my point of view was, after people like Dave Patterson, who was one of the chief people on the RISC side, I think after he got over his initial annoyance that there were people out there questioning anything to do with RISC, he was pretty much a gentleman. He was not the problem. He's been very friendly to me all over the years and I appreciated that. We could disagree professionally and still be cordial about it. Not everyone could do that. I gave a talk at Berkeley in about 1996 or so and the gist of the talk was here's what we did, here's what I think it means, here's what I think it means in terms of RISC and CISC and so on. And essentially the elevator pitch summary of that talk was along the lines of, if I can, you might say that CISC only stayed viable because Intel was able to throw a lot of money and people at it, and die size, bigger chips and so on.

In that sense, RISC still was better, which is what was claimed all along. And I said you know, there's point to be made there. I agree with you that Intel had more to do to stay competitive. They were starting a race from far behind the start line. But if you can throw money at a problem then, it's not really so fundamental technologically, is it? We look for more deep things than that, so if all the RISC/CISC thing amounted to was, you had a slight advantage economically, well, that's not as profound as it seemed back in the 80s was it? Dave Patterson was great about that and he said the word, "Are you saying that RISC made no contributions?" And I said, "No, nothing like that. There were definitely contributions." But the one that I most value is that before RISC came along, the entire field was on what I called a religious basis. People were publishing whole books about capability machines or object orientation where they would say, here's what you should do, follow these precepts and surely goodness and light will follow you the rest of your days. And the RISC guys came along and said, "No, you design it, then you measure it, and if it's fast, then it's good. If you don't measure it, you don't know if it's fast and if you measure it being slow, it still is not good." I think they get a lot of credit for turning the field back to a numerical basis on which we could do real engineering and leave this religious stuff behind. And I told Dave that, and he said he could live with that legacy, but one of his grad students couldn't. There's a guy up in the back of the auditorium and he was really mad. In fact, I wondered if he's the one that stole my P6 demo chip that I've passed around the audience and it never came back. I was wondering where it went.

Early life

0:08:52 PE: Okay. Well, we're going to start off with, in a sort of chronological sequence and just begin my talking about your origins, your early life. So where were you born, where did you grow up, what was your life like when you were a child?

0:09:11 BC: Well, let's see Pittsburgh is where I was born and raised. I lived there until about age 23, same house the whole time, so pretty stable. I had both parents. It was a great family. My parents were wonderful people. They're still around. They're still in Pittsburgh. My dad was a milkman and I don't know what he got paid but it couldn't be much and he worked really hard all the time. My mother was pretty much staying home, we had six kids in the family so she had a lot to do.

0:09:40 PE: Which one are you in the order?

0:09:42 BC: Number two and then, it went girl, boy, girl, boy, girl, boy. And the first four of us were all within four years, so we had lots of little kids running around when I was growing up. My older sister is 11 months older than I am and so, there's a period like a month every year that we're the same age in which I was kind of made big fun of her for that. But what it also meant was she was in first grade, I would have been in the same grade as she was, and my mother went through some decision and said, "No. No. I'll hold back a year," which meant I should have gone to kindergarten, but I was much too big and too old and they sent me home. They wouldn't let me be in kindergarten. So traumatic, but it meant that I got to spend that year playing with my Erector set at home, which I learned a lot from. You learn manual dexterity. You can sort of conceive things. I highly recommend it and I'm surprised that the Erector sets have kind of fallen out of fashion but you'd be amazed how many engineers have that as part of their background. I’ve also built Heathkits, another common occurrence among engineers of my age. So that was the beginning of it. We, I had one other advantage which was I had an uncle who was an electronics technician at Westinghouse in Pittsburgh and he was...

0:11:04 PE: Your father or your mother?

0:11:07 BC: Mother's side. My mother came from a family of eight kids and so, he was one of the uncles, one of her siblings. And he was very forthcoming with parts and saving me when I was trying to fix something when I couldn't figure out what was wrong with it. I fixed a lot of TVs and stereos. There was just kind of fun to see if I could do it, and a challenge, but I knew if ever I got stuck, he could definitely do it. And boy, that changes everything when you're not all by yourself out there. And I also learned to think. I really think in a fundamental sense, I learned to think from that guy, my uncle Joe.

0:11:37 PE: Did he take you into the shop and show you things and...

0:11:41 BC: Yeah. And he wouldn't just say, ”give me that thing; I'll fix it for you" and give it back fixed. He would say "what would you try to do. What's the malfunction that you think you're seeing, and what do you think might cause that?" He'd walk me through it, but what he was really showing me was how he would approach the problem. And, looking back at it, what he was also showing me was, if some human mind created something then your human mind can understand it. You should always assume that, because if you assume it, it throws away all the doubts that are otherwise going to bother you, and it’s going to free you to just concentrate on “what am I seeing, how is it working, what should I do about it, what am I trying to learn from this”. Never, ever think you’re not smart enough; that's all nonsense. That's where I learned it, it was watching him. He's still better at it than I'll ever be. I mean, I watched him do what I consider near miracles, like walking up to a refrigerator and telling you that the bearings and the compressor are bad, walking up to a car, telling you that there's something wrong with the carburetor just by listening to it. I mean, I have no idea how one does that. I watched him do it for years and years, but that was a real benefit for me growing up having him around.

0:12:54 PE: So were those experiences your first ones with electronics or were the heath kits, or where was it?

0:13:02 BC: I think the first ones were my uncle messing around with stuff that he was showing me or, fixing electronic stuff, radios. Radios were so much simpler back then with tubes. First of all, if you don't know anything about electronics, back then you would take the tubes to a drugstore and test them in the tester and sometimes they would tell you the truth about which tube was bad. But if you're really lucky some neighbor would say her radio quit working, and you just look at it and say, well there's five tubes and one of them is dark. Well, there's a clue, right. So you can sort of zoom in pretty fast with the easy cases. Yeah, I think it was mostly fixing things that had broken that got me started. And I still remember an incident when I was13. My mother had begun work in the evenings. My dad would take Monday evenings off and go bowling and we would be at home, my older sister was in charge, and the rest of us were supposed to do whatever she said. Well, the furnace broke, smoke was filling the house because the motor had basically fried. It was winter in Pittsburgh real winter. My dad had been saving a motor to make it table saw at some point. So I went down and looked at the motor and I said, "Well, it's not quite the same size.

0:14:22 PE: Was the furnace burning? Was it...

0:14:24 BC: No, it just, the motor had sort of overheated and had burnt up and it quit working and so, there was no more forced air so the heat had stopped. So I took the old motor out. You could see how it was clamped in there and there's just a belt around the pulley. So I took the new one, it had the same pulley on it and it didn't fit in the mount but I found an old rubber bicycle tire, to use a sort of a shock absorber, and I put up a clamp around it. It ran that way for probably the next, I don't know 15 years. I reminded him of this incident recently, and he didn't remember it at all. He came back and the furnace was working right. He never tried to build a saw either so he wouldn't have known that his motor was missing.

0:15:06 PE: Did you tell him?

0:15:06 BC: I did yeah, I did but maybe he thought I was making it up. I mean there was this dead motor sitting there. In general we kind of had to make do, you know, we weren't flush with money so if something broke you pretty much had to fix it. Whether it was a car, a TV or anything. So that was kind of the attitude I grew up in and fortunately my Uncle could show me how to fix things because my Dad wasn't very good at it. He was more mechanically challenged.

0:15:36 PE: Do you remember your first Heathkit? What was this question about?]?

0:15:38 BC: I think I do, actually. I think I mention it in the Pentium Chronicles book, because I still have this vivid recollection of watching this guy walk up. We had a long flight of stairs down to the street, like three different sets of stairs and they were long. So I'm watching this guy get out of a UPS truck with this huge kind of cardboard box, coming up the stairs and I'm thinking what could that possibly be and it turned out it was a guitar amplifier that my Uncle had bought for me and I still have it incidentally. It still works. It’s upstairs; it’s like 45 years old or something. In fact I used it I used to play in a band I used to be in.

0:16:12 PE: Was it a Heathkit? I built those also.

0:16:15 BC: Oh did you?

0:16:17 PE: Yes, it might even be the same one.

0:16:18 BC: Yeah they were great. Yeah, it was the one with the silver screen and two speakers in it.

0:16:21 PE: Yup, yup.

0:16:22 BC: It was a combo.

0:16:23 PE: I think that was it.

0:16:24 BC: The only thing they didn't get right in that design was the reverb. Oh, no, sorry the tremolo. The reverb wasn't their fault but the tremolo was. There was a little module in there and that module would fry. It would just die and then they'd send you a new one. I bet that that's the fourth or fifth one in that thing upstairs. But anyway, it was since you built it as you probably know, it was kind of cool to go from a box full of just parts to something that actually does something useful.

0:16:49 PE: They were fantastic.

0:16:49 BC: It's a blast and it sort of subliminally informs you that that this can be done. You can take a bunch of useless components in isolation and combine in certain useful ways if how and then make a useful artifact of it and that's what designers do I built TVs, stereos, radios.

0:17:12 PE: They also had something that actually taught electrical theory did you ever?

0:17:15 BC: I never played with that. Yes I saw those but by the time I could afford to do it I didn't need it anymore. Heath went into robotics for a while. They even went into woodworking for a while. They had a roll top desk kit. I never built it but they were kind of towards the end, they were kind of desperate to find a business model that still worked. The technology kind of walked on past them. But I still think that that in terms of informing the intuitions of generations of engineers that was a heck of a good way to start.

0:17:43 PE: Yes it was.

0:17:45 BC: if something went wrong, you had great documentation to go looking for what was wrong. Did you build color TVs.? If you built their color TVs, they sent you a volt ohm meter as well. They designed the thing in such a way that you could calibrate the important pieces with just that meter. Normally with a TV you'd need an oscilloscope and tester and a generator and all kinds of stuff. But those guys were careful. They knew what you were going to have at home because they supplied it to you and so that was tremendously useful if anything bad would happen or if anything broke, you could fix it. Not like today's electronics.

0:18:19 PE: No — probably some of it you designed.

0:18:23 BC: Yeah. So here's one of the things we did on the P6 and they're still doing it today on Intel's parts. There's a thing in there called a micro code patch facility. And that came from a proposal that I made about1993, I said "it's a real shame from our point of view at Intel that when Microsoft tests something they want to change about an existing product in the field they issue an upgrade, they just send you the bits and presto. They didn't do a recall, they didn't have to incur any expense, they just send you the bits. You might grumble but you might also realize hey these things happen it's to get it fixed". But Intel when they had the problem with the Pentium in 1995, they had to recall the silicon and that's really expensive. That had not happened yet when I started arguing for this but it was evidence that it was good. So I said "Well why don't we do the same thing that the mini computer guys did prior to micro processors, which was they had a writeable control store in there and they could rewrite it if they wanted in a way that’s roughly equivalent to our micro code in an X86 chip. So why don't we fashion a way to overwrite some piece of that micro code and we'll be able to fix errors after we've discovered them when the product's in the field? Of course we'll have to have substantial security mechanisms in there to keep the bad guys from rewriting it to what they want because micro code is all powerful, you can do anything. If you can reach the micro code, you can do anything you want and that's bad if you're a bad guy". So we worked pretty hard on that piece. But once a secure mechanism is in place it seemed pretty. There are some subtleties associated with it because every time you find and fix a bug you're looking to the future you don't know if any more bugs are coming. Since each one of them uses a piece of a finite resource, sooner or later it's going to be full and then what do you do, you recall the silicon anyway.

0:20:15 PE: Yeah, yeah, yeah. That's not good.

0:20:19 BC: So you have to husband that resource carefully and we had people that did that. But all of that came about because it's nice to be able to fix things rather than replace them. Intel to this day as far as I know is still putting that facility in their chips. Of course they're always getting, there's always going to be bugs so I think that the need for such a facility will continue.

0:20:39 PE: Yeah, yeah. Well so where did you go to school?

0:20:44 BC: For grade school I went to a Catholic grade school called St Norberts. Grades 1 to 8. And then in high school I went to a place called South Hills Catholic, which is now called Seton Le Salle. That was an all boy’s high school. And from there I went to Pitt and then Carnegie Mellon.

0:21:02 PE: And what about school, how was that for you? Was it a good school?

0:21:07 BC: Grade school?

0:21:08 PE: Yeah.

0:21:08 BC: I was bored out of my head most of the time. I remember just a few things from grade school that stick out. One was being bored a whole lot. But every once in a while a spectacular teacher appeared that understood what was happening and went way out of her way to do something about it. In 5th grade, for example, we had a nun there that realized that she had a set of kids that were reading far beyond the grade that we were supposed to be in. And she would take us in the hallway and we would do Shakespeare instead. I can't say I loved Shakespeare but it was a lot more fun than reading this 5th grade stuff. Yeah oh my gosh, and the math was the same it was just not interesting. It was way too simple. was a really good speller, I think there's a spelling gene that you just kind of either see a word and remember it or you don't. I have the gene and least two of my kids do.

0:22:12 PE: Yeah.

0:22:13 BC: It's not an effort, you don't have to work at it, it's just you see the word and it's either right or wrong. Well so I would be in spelling class because I could spell far better than the nuns that were teaching it and they did not want to be corrected and I was sitting there and I remember thinking, she's put a word up there that's misspelled. She's teaching 30 kids here the wrong way to spell a word. Now if I say something she's mad at me and if I don't they learn the wrong thing: which is more important? You shouldn't ask a 6th grader to wrestle with conundrums. But yeah that's the main thing I remember and I remember being in the choir, singing in the choir there. I started taking guitar lessons when I was 6 and I took those until I was 23.

0:22:54 PE: That's quite young for guitar lessons because your hands are still small as a 6 year old.

0:22:58 BC: Well my parents, and my Dad's theory was "Look if you're going to go to college, you're on your own because I don't have any money and with this many kids in the family we're not going to be able to help. But what we can do is give you a skill that you could turn into money if you need to". Music was one of those. He said every one of you has to learn to play a musical instrument. First I started on piano at age 4 and at 6 I switched to guitar because it was novel and I stuck with it. My sister plays flute and piano, my brother became a professional musician. So when I had kids I did the same thing and I said you have to learn at least piano and if you want to play something else or join choir you can do that as well and I'll pay for any lessons you want to take. So, they've gotten really good. All three kids play piano. My oldest daughter plays flute, my youngest plays French horn, they were all in the choir. it's been kind of neat when they come home and the start playing on the piano because they’re quite good. I don't know, I mean I read the same thing you do probably that says musicians tend to be good at math, that sort of thing. There's some kind of connection. There might be but that's not why I do it. I just think music's a really cool thing and at least in my case, I really enjoy doing it right before I go to bed. I spend the last hour of each day. I don't want to say it's practicing but sometimes it's practice but sometimes you're just goofing around because it's just fun to play. If you want to learn something new you really should do it right and practice, and sometimes I do.

0:24:25 PE: That's great. Did either of your parents help you with math?

0:24:28 BC: No. My Dad can add the longest column of numbers I've ever seen because he's a milkman. He would bring these sales lists home, and in a time before computers or calculators he would add the numbers up in his head. He was fast at that. Both my parents are quite intelligent. But they didn't go to college for mostly financial reasons. And so they had no basis to help me with anything beyond just fairly rudimentary stuff that I didn't need help with anyway. By the way I don't particularly think I'm gifted in math. I'm an engineer. I learn what I need to learn to get what I want to do but I can't ever remember doing it because I thought it was fun in itself. So, I took all of the hard classes because I knew that I was going to need the techniques that were in there. But the teacher I liked the best in high school used to be a nuclear reactor engineer and when he was teaching calculus every one of his examples had to do with heat flow and I loved that.

I mean as soon as you give me the problem couched in physical terms it’s like "oh now I know why I need to do this". I know this, I know the likely range of answers, I'm a lot more comfortable with that. Because what I found difficult was classes like, I don't know, biology or something, where it felt to me a lot of memorization was required. I just thought, you know, I have a good memory but I don't value it of itself; it's the insights I like. It's the connections, and I wasn't getting those enough so I really like them in physics and I wasn't seeing them so much in biology. Chemistry it took me years to figure out how it worked. I wrestled with chemistry, I really did. I understood it at an atomic level, like where the electrons go and the shells and all that. I don't know why the rules are the way they are, but they explain an awful lot of things. But when you start talking about huge numbers of atoms all interacting as a mass, I didn't like that. I wasn't as interested.

0:26:37 PE: Yeah, yeah, it's still very hard to get from the chemical structure of a molecule to its material properties.

0:26:46 BC: Yeah and there's some differences even. Because if you take a single molecule of water you can sort of see what the angles are of the two hydrogens and so on. But if you put a bunch of them together it changes. They affect each other and in fact that's, part of that is why ice acts in a strange way relative to just liquid water because of what those angles do. My intuition does not tell me what those things would have done. That's why that's not my field. If you ask me how a machine works in computing land, I'm a lot more comfortable because somehow my head is wired more towards that. Chemistry, I was always impressed with people who could just go “well of course if you stick two moles of sulfur and you add it to this other thing this is what you'll get”. Well, I'm glad you can see it but I don't.0:27:34 PE: So still grade school, high school, what else did you do anything like sports or other sorts of talents, interests?

0:27:45 BC: Yeah I was so clumsy when I was a little kid that my Dad feared for my safety, and he was probably right. Like I've had plastic surgery on this ear because I fell once in the house and crushed it on a toy box. One night we went to catch baseball down on the street and I managed to fall and hit my chin. They had to take me to the hospital. I fell walking to school once and got blood poisoning in my leg. I mean I was complete klutz. I was big for my age so when I played baseball, it turned out I was the pitcher because I could throw fast and I could generally put it where I wanted it. But if I had to run, that was not a pretty sight. I used to get ridiculed for my running style. And my Dad was really worried about it and so he eventually got me a jump rope and made me go use. I don't know if it helped. The one thing that I was good at was I had very fast reflexes. I still do to a large extent so that helps in certain sports like badminton, racquet sports. I really like racquet sports because that's what that's good for. Volleyball too; I've always enjoyed volleyball a lot.

I played little league baseball. My father was my coach for a few years, and if you ask him what were the hallmarks of me playing little league he'd say "he struck out a lot". But I was often in the all star teams because I was just big and I was a pitcher and I guess they needed me for something and if you threw me the ball I'd catch it. But, yeah that was it, it was mostly little league in the summers. I also played basketball. Okay here's another one. When I was in grade school I played basketball at my school for three years and I was tall, so I wasn't totally hopeless. But I was not good and at the end of that year I distinctly remember that he lined up all the 8th graders on the edge a platform and he went down one by one and he said "Jim, you need to work on your dribbling skills, but your shot is great and this guy here, your leadership on the court is great, keep it up, it'll help you play in high school". He gets to me and goes "you need to find a different sport". [Laughter]. I went great, thanks coach. [Laughter]. I mean my self esteem was taking a beating here, but he was pretty much right, I wasn't a basketball player. But I was a pretty good volleyball player, so when I went to high school we started a team and that was fun, and I've played a lot ever since so. And then racquetball, and I got into that in high school and mostly in college.

0:30:29 PE: Okay, that's quite a bit really.

0:30:32 BC: Yeah, well there's also, let's see, when I got to high school we started skiing, and I met people through my guitar playing. I was playing in a group, a folk group, with kids aged 15 and older, and somebody I met there was really into hiking and canoeing and skiing, and so I started doing all that stuff, all of those things with him, and so now we still do them to this day. We like to go out to the woods and hike and all that stuff, that's where it came from, at least for me. I still like to ski.

0:31:10 PE: Yeah. So when you were a kid, what did you think you'd do when you got older?

0:31:15 BC: Well I wanted to design electronic stuff. I always knew that.

0:31:19 PE: It came on really early.

0:31:19 BC: Yeah, I just got a big kick out of it. I had the chemistry sets, but I didn't necessarily understand it ; it was just, do these things and see what happens, make smoke, make things turn colors or whatever. That part was just fun by itself, but it usually meant that you had to wait a little bit. Mix them up, wait three minutes or whatever. Electronics: flick the switch, and it works, or it doesn't. It's instant gratification and I was just really intrigued by the scale, even as a little kid, I could see that the TV works because of electricity, the lights in the house are electricity, outside you see huge utility towers, that's all electricity. A summer of thunderstorms, which we have a lot of in Pittsburgh, that’s electricity too. My daughter was 8 before she experienced lightning and thunder but, yeah, it was the sheer magic of what electricity seemed to be capable of just really struck me. And because I was fixing things too, I thought, I want to know how these people do this and, what else can we do with this. It was fun. At the time, this was in the 60s I guess, computers were kind of invisible, no one ever talked about or saw them, we knew they existed, but it was only until I got to high school when two things happened. One was somebody got a teletype, I don't know why but they got this teletype, and I was standing near it when it started running and tch, tch tch (making a sound like a printer) and things were printing.

0:32:49 PE: Was that at school or...?

0:32:50 BC: Yeah, in my high school. I still don't know why, but that machine was there, and I watched it one day thinking how does it know what to print, I mean what is happening here. And it had the same affect when I went to visit a friend at college, who was just couple years older, and standing near a line printer and it fired up and started running and I thought, "oh I've got to figure out how this works", I want to know why that thing's doing what it's doing, who's controlling it, how are they controlling it. Do you know, it was just sort of wondering what's going on behind the covers that made me wonder, and also there was also this attitude that I've mentioned, that my uncle kind imbued me with, that you should never worry that you can't figure something out, you can, it's just a question of whether you want to or not and whether you have access to what you need to make progress on some question. So as soon as I saw the line printer start running it didn't occur to me that maybe I can't figure it out or I won't understand it; it was more "hey what's going on here", I need to find this out. When I drive a car, I don't need to be a car designer to understand enough about internal combustion to feel comfortable with the process that's happening. I know that you stick gas in the tank and it gets converted to kinetic energy and heat and I'm okay with that. I don't know what the exact budgets are for those things but that's how the process works and I can manage my time and use the machine that way. What I don't understand are people who don't have a foggy clue how the technology works for what they're using, and they don't even seem to care. That's when I thought "man, definitely like walking through a world or a magical universe where, you know, wizards are waving their wands at you and changing your life and you don't even know why, or how, or who these people are". I mean how can you live like that? (chuckles) I would think that's scary.

0:34:29 PE: Yeah, I understand that. I feel the same way (Laughter). Yeah. So at what point did you start to turn that into a reality? I mean you were working with your uncle a little bit, was there anything like a summer job, or any sort of more formal training while you were still at high school?

0:34:50 BC: We had a TV studio in my high school, a black and white TV studio, and it was an honest to God studio with cameras and audio. I started running the audio in my freshman year, just because I understood that better. But when things broke, as they always do, cables would fray or whatever, someone would have to fix those, and since I knew how to solder from my Heathkit experiences that fell to me. So it was just stuff like that, you know, just mostly fixing things and there were a couple of things like when I needed a, I wanted to build a new stereo amplifier for my parents, and we couldn't buy one it was too expensive, so I mentioned it to my uncle and he started accumulating parts and at some point, he had enough parts and he gave me a bag. Now true confessions, he gave me the bag, I built the stereo and I turned it on, and it fried. The smoke came out and he and I had to spend a few hours trying to resurrect it and I don't think we ever quite did. I don't recall ever getting that working properly, but the experience of building it and realizing that, you could do it, but if you screw it up it won't work, it's pretty vital too. You need a certain amount of paranoia to be able to tackle complicated things, you can’t just assume it's going to work.

In fact I think, one of the things I said in the book was, I always felt that you have to be able to hold competing ideas simultaneously in your head in the engineering effort. One part of your head has to say, "I'm going to consider every possible thing that can go wrong when I design this thing, so that it will live forever, never fail, never disappoint anybody, it's going to have no bugs, for the first time in history, this design will have no bugs and I'm going to work like crazy to make that happen". The other side of your brain is going "yeah, right, it's never been done, you won't be the first, no matter what you do, something is going to go wrong. There'll be intrinsic errors, there'll be bugs, or there'll just be users using it in a way that you didn't anticipate, they're going to think that's a bug, you might protest, wait a minute, I never promised you whatever". But if the end user who paid the money for the part isn't happy with what you gave him, there's a problem, and I think the best designers just internalize that and say I'm going to get you next time then. I won't make that mistake again. Anyway I think you have to hold both of those in your head at the same time. I used to have problems with my management on that exact issue, because I would enunciate that theory and my boss would translate it as "you're planning to fail?" In fact, remember the micro code patch facility, I had a lot of fights over that of that exact nature because the executive VPs would say, "You want me to allocate chip real estate on a facility to make up for the mistakes that you think you will have made? Why do you not just make no mistakes and then we don't have to deal with them on the chip". And I said "If you can find somebody who can actually deliver you perfect designs of any sort, you really should hire them and if they can do it, you should fire me and put them in my place, but I don't know how to do it, so until such day as you show me the magic..."

0:37:49 PE: Well maybe we'll come back to this later, but in the 80s during the strategic defense initiative the software debate was all about that problem, and those guys, the SDI people, would say exactly that kind of thing: "We prevent bugs by design. We're just not going to have any bugs in our code".

0:38:08 BC: Yeah (laughter), or they would say, you know, I'm going to do formal verification.

0:38:11 PE: Yes, formal verification, that was the holy grail for quite a while.

0:38:14 BC: And you've ever heard, I think I stuck this in my book about the Pentium FDIV error. It is to an extent due to a formal verification error.

0:38:23 PE: Yes, I remember that.

0:38:25 BC: You can make errors, people are good at making errors, they're going to make them somewhere. I think that there's maybe a quota that you have to meet, you know, you have to have at least this many errors. But yeah, the other thing about SDI is, is you can't test it, and my theory is if you don't test it, it doesn't work. I learned that from somebody at Multiflow actually, and I think he had it dead right and, in fact, I keep thinking, I've been collecting information to write a book, and I was going to use that as the title. I might still, I don't know but we'll see if I can get around to it. I think that's the fundamental thing that's wrong with SDI, if you can't test it isn't going to work. Guaranteed, I can't tell you just exactly how it will fail, I just know it will.

0:39:00 PE: The worst parts are the possible ways it can fail. Alright so, let's see. In one of your talks, that you sent me, you talked about this discovering the six transistor radio?

0:39:26 BC: Oh yeah. [laughter]

0:39:27 PE: Want to tell that story, since I asked?

0:39:29 BC: Yeah, that was back in around 1962 or 3 somewhere around there and I made some, what were for me, fundamental realizations. So remind me of that story because I want to tell you another one and I'll forget if you don't.

I was in my uncle's, same uncle, and I was in his backyard and he was playing with these two new walkie talkies he had just bought. I'd never heard of a walkie talkie. Back in 1962 transistors were only three years old, so these were brand new things and so we were playing with them, and I would grab one and my sister would grab one, and we'd be talking to each other and then I realized, as soon as she was far enough away to where I couldn't hear her without the walkie talkie, I couldn't hear her with it either. The range was such that they were the same range.

So I thought, well that's not fair, this things not doing anything, it's a sham. This walkie talkie doesn't work, and yet somebody sold it, right, so I'm thinking well how can that be, how can you go into business selling things that don't actually do what they're supposed to do, and as a little kid I was just shocked by that, like is this possible. So the next thing I think, like for first communion, (Catholic church, first communion), people come over and they give you presents. And 1962, I think, is when I had my first communion and I got a walkie talkie, I got a radio, and I forget how many transistors it was, but all my friends at school, well they were doing the same thing. They got radios and we compared notes, and one of us had an 8 transistor and one had a 10, one had 12, one had a 6 and I thought, well that's weird, they don't sound any different, they sound the same, it’s not like one's higher quality that I could tell, so I started wondering "well, what's the difference? What are those transistors doing if they don't result in any palpable improvement?"

So I took the backs off and I wanted to see if I could spot the difference. I counted the transistors (I knew that they were the little silver cans), they were easy to spot, and so I verified that they had as many as they said that they would have, but I also noticed the printed circuit wiring that was on the side of the board that was facing me when I took the back off, because I could see it and I looked at it and I realized those transistors didn't go anywhere. Only six of them were connected to the circuit.

0:41:36 PE: They were just stuck into the board with, by soldering.

0:41:37 BC: Yeah they were just stuck in. And I was puzzling over this for the longest time thinking, my understanding limited though that I knew it was, was that those wires on the board were carrying electrical currents of some sort and that's how these transistors work, so if there's no wires, how could they do anything, and I finally asked my uncle, "I give up I don't know what's happening here, why would they put transistors in there, how are they participating in the radio's function without those wires?" He says "they're not" and I said "well what are they doing in the circuit?" He said "they're not in the circuit, they're just in the product", "What?". He said, "in fact, I bet you that if we test them we'll discover that they don't even work, they're probably the broken ones that were on the floor and they just stuck them in there, so that they could call it a, you know, more of a transistor radio so that people would pay more for it".

I said "that can't be legal." He goes "sure it is", and I suddenly realize my gosh the world is a stranger place than I ever thought. That was that incident and that I've seen morally equivalent things all through my life, on a grander scale. Crazy stuff, and by the way, I became more sympathetic not less to the people who did it that way. One classic example that everybody has in folk lore and I've seen it live and so I know that this happened. IBM had printers and they would have a big circuit card inside and you would pay one amount of money for a printer of a certain speed, and a lot more money for a printer of a higher speed. If you wanted to upgrade you could do that. If you upgraded they would send the field guy, he would open the back, flip a switch and close the back. Every engineer is going to look at that and say "that's sleazy". But there was an incident at Intel where we had been caught in the late 90s with no competitive part to a cheap, a really low end X86 chip, by a competitor and so we quickly got together and said "what are we going to do?" We need to react to this threat, I mean there's obviously a market there and we have no product, and so if we design a new product that would be the most fun thing to do but it would take two years to get the thing out there. In the meantime — and we should do that, so we will — but in the meantime what do we do?

We realized we could just take a slow Pentium III and we'll turn off half the cache, just turn it off. The marketing guys proposed this, and I remember sitting there going "oh has it really come to this, I've joined the dark side, you know, this is really bad". But a really savvy marketing guy, who I have a lot of respective for, a guy named Lou Paceley, he kind of took me aside and he said "look, think of it this way. What we are all about is trying to offer products to the customer and if the customer buys them it means the customer saw something in our product that met a need, they hope to meet some need with what we are selling. Now if we sell them a single product at a single price point they either can afford it or they can't. If we offer them two, at different price points, now they have a choice and they can try to decide which one best matches what they're trying to do. What we're doing here is offering that choice." He said "I can't do it any other way than this, if I turn the cache on, suddenly all of my Pentium IIIs are going to drop down a bit. Then we'd have to sell them all at this lower price point because no one's going to pay more for the same thing; that doesn't make any sense. So this is an expedient."

Later on, another marketer told me a story about Wendy's hamburgers. Have you heard that one? I have no independent means to verify the story, but it makes sense, I hope it is true.

They said there was a time when Wendy's started out making three different hamburgers, the triple, double and single. And the triple of course is three singles stacked on top of one another and it was a mountain of hamburger, three quarter of a pound. And he said they had that for their line up for the longest time and they were selling a lot of doubles, he said, not so many triples. So eventually somebody at Wendy's said well what are we doing the triples for, it costs us extra money, no one's buying them, so let’s get rid of those. So they got rid of them and suddenly the double sales dropped to zero and people were only buying the singles. He said why do you think that is? I don't know. He said because people need choice, he said they are not always sure how hungry am I, is the extra amount of food worth the extra amount of money? I don't know, not really sure. So he said if you give them three choices they’re going to gravitate towards the middle and then decide whether to go up or down and he said most of them will stick in the middle because they’re just not sure and it keeps them there. But if you take away the top and now there are only two choices, they’re going to gravitate to the first one and realize they don't need the second one so he said the lesson is when possible you should always have three choices in your product line.

And we didn't at the time. We had two. That’s why he raised it, we had the Pentium 3 and we had this cheap thing. This is not enough. So we started thinking, I think did mention this in the book about blue crystals, he said we need the equivalent of laundry detergent blue crystals to fix this. And that's when I really got this is to much high rolling, I don't think in terms of blue crystals, let’s offer real product, real value added, not just fake. On a expedient basis temporary thing I can see you do what you got to do to stay in business, but long term it's got to be real you know. Sooner or later your competitors are going to come up with something real and kick you in the butt with it. So yeah that was marketing 101 at Intel.

Higher education

0:46:59 PE: Alright. So you knew you wanted to be a electrical engineer in high school. You worked in the TV studio. You got to college directly from high school, or what was the...

0:47:12 BC: Yeah.

0:47:13 PE: And did you apply directly to Electrical Engineering or how did that go?

0:47:17 BC: I did. Yes, I applied to the EE department. Although I had one other little learning thing was that I was a national merit scholar, so they had these little get together where all the colleges would come and try recruit you, and I remember a professor from the University of Chicago, I was wearing my suit and everything. He came up and he said I am so and so I am teach at University of Chicago, where you thinking about going, where you're going to study? And I said well I am not entirely sure where I am going to go yet all though I was leaning through something in Pittsburgh just because it would be cheaper, but I said I definitely want to study Engineering, Electronics Engineering is what I really want to do.

And I don't know what I was expecting but I did not expect what he did: his face fell and he said I can't believe you would waste your time on a subject like that. And I thought “waste your time”? I didn't anticipate it being a waste of time, I expected it to be challenging and interesting and that was not what he was saying. What he was really saying that engineering is a professional thing that you do, but if you want a education, a real education you have got to do liberal arts. After that if you want to do this ridiculous crap, engineering, and I never seen that attitude before until that one incident. I just remember going home shaking my head thinking do people really think that. It didn't change my mind at all. I still wanted to design things, but I didn't realize that there where people out there that felt like that. Later on...

0:48:35 PE: The University of Chicago is sort of the epitome of...

0:48:37 BC: Exactly. They are the top, I mean I kind of realized that later on too, I didn't know that up front. But I’ve run into that more times then before. I mean for example just this year my son graduated Cornell with a engineering degree and at his commencement the president of the university, not once, but five separate times, because I counted them in the speech to the entire graduating class, said the equivalent of you know, your parent should be really glad you sent your kids here because we don't just don't teach them engineering, we give them a real education with broad exposure to all these other things. And I thought if you want to say that once I am okay with it. I do think that it is easy to get sort of cloistered in an engineering field and forget all the social sciences and English, but five times? Come on! I paid you a lot of money for my son's education and you are telling me it is not any good, come on!

But it was even worse in Colorado. When we started there like four years ago they had a an intro session for the parents and the Dean. At the time my daughter had signed up for liberal arts, then she switched to engineering after about a semester, as did my son by the way. I remember sitting in this auditorium with all the arts people and this lady did the same thing, she stood up and she said I am the Dean here and you should be congratulating yourself that you didn't send your kid here for something like Engineering or Law or something because those are just professional activities, that is not what education is all about. And I told my wife it's all I could do not to put my hand up and say back away from the microphone, you are not worthy to use, it is a electrical device, somebody designed it, see that PA system, they designed that, see those lights somebody designed that too. All those people are doing things you don't seem to understand. I think that there is a fair amount of that. But I don't want to push too hard. As I said I have come to realize that a broader education actually is a cool thing, even for engineers. I don't have any issue with that. I just want there to be a balance.

0:50:44 PE: Okay. So you applied to Pitt because of... So it says Bob went to the University of Pittsburgh and did his MSc in, you went there in, I'm guessing 1973?

0:50:57 BC: Yes. And it was for my bachelor's. My master’s is actually from Carnegie Mellon. But yeah as a senior I applied there, Michigan State, MIT, a few other places. And I got accepted at all them, which is particularly ironic because MIT accepted me for that, and when I finished with my bachelor's I went to Bell Labs and was looking for grad schools. Oh, when I finish my master's and I was applying for PhD programs, MIT no longer wanted me. I guess I must have gotten stupider in the last few years, I don't know [chuckle]. I thought that was pretty funny.

0:51:35 PE: What probably happened you fit their diversity criteria [0:51:40] _____.

0:51:41 BC: [chuckle]. Quite possibly.

0:51:42 PE: At the PhD level they don't do that so.

0:51:43 BC: After watching my kids go through this I think it is a random process now anyway. But yeah I went to Pitt, it just seemed to be cheapest and at the time I had no idea how we were going to do anything else, I didn't have a car, my parents had one car that barely worked. To be honest the thing that intrigued me about Michigan State was they had a Electronic Music Synthesis studio. I wanted to use the synthesizer. I just loved them. They were the border line between music and Electrical Engineering, and I wanted to go there. Then it turned out that Pitt had one too. So that settled it, I am going to Pitt. It was nice was and cheap. I didn't get a great education there, in terms of Electrical Engineering; they were struggling as a department back in the 70's. Since then they have done a whole lot better, the school's actually quite respectable now. But it was a struggle back then. We knew at the time that we were being short changed. For Communications or Control Theory in particular, we just weren't getting instruction that made sense to us. And we were good students. So when I went to Carnegie Mellon the first thing I did was take Communications Theory again, even though I wasn't required to. Because I wanted to understand it and sure enough they had a gifted teacher there and I finally got most of what I wanted out of the class. But yeah it was... I mostly went to it because it was handy. That's really the main reason.

0:53:23 PE: You were still living at home?

0:53:24 BC: No I was living in the dorms but because it was in the same city every weekend or every other weekend I would take all my dirty laundry to my mom. Then she would feed me then I would watch the Steelers game that afternoon.

0:53:37 PE: You must have great parents.

0:53:39 BC: Oh yes they were great. They were wonderful people, yeah. Yeah, I don't know how the heck they raised six kids on no money. But the one thing they did was they instilled is an attitude: my dad's favorite saying is the world doesn't owe you a living. And I thought man that's right. If you want to get somewhere you can't just sit around and hope they will knock on your door. You have to go do the work. I think I have offended people on this subject because I say things like I don't like the American Idol attitude. I don't watch that show, but I have seen enough trailers, to where these people are going, I think I am going to win this and I think it's is going to change my life. I think you should change your own life. Your odds of winning this are not that good. They are about as good as a golfer or a professional sports team. Try to become a pro linebacker in football. If you make it you are going to make money, but most people can't do it. It's very, very hard. So you better have a plan that actually makes some sense. But it's just the attitude that I should just wait around till the world blesses me with fame and money. I think that whole attitude is garbage. We are never going to compete on a worldwide basis that way. I think you have to put a lot more work in to it than that.

0:54:50 PE: Lottery mentality.

0:54:51 BC: Yeah. Exactly. So I think my parents did a great job of instilling the right attitude in their kids.

0:54:59 PE: What became of your siblings? You mention that one is a professional musician. What about the rest of them?

0:55:04 BC: My older daughter... I keep saying daughter. I am so not used to talking about my family. My older sister got a nursing degree and then she raised her own fairly large family. And that's what she's been sticking with. She likes to write, and plays the flute for her church. We just saw those guys last week, when we were traveling to Pittsburgh. My sister Kathy is vice president of a health insurance company in Pittsburgh. My brother Denis, is the professional musician. He teaches in Carnegie Mellon's music department and he just recently resigned as the conductor of a group called the River City Brass Band, which is a professional brass band that has been playing in that city for, I don't know how long, for 25 years or so. It is quite fun watching him conduct, he is really good at it. Not that I would ever tell him that live, sibling rivalry being what it is. My sister Susan is an administrative law judge in Pennsylvania. And my brother Marty is a sales man at a mint, where they make commemorative coins..

0:56:22 PE: That's great. That great. Alright, let's see, back to Pitt for a little bit. So you felt like the Electrical Engineering department was not that great for you?

0:57:13 BC: That was the conclusion I reached after being like three years of being there. Didn't know that right away.

0:57:18 PE: Were most of your courses in EE? Or did they have distribution requirements and things that made you...

0:57:24 BC: They had some. Yeah they had some. For example we took some English courses just because we found them so interesting and fun, we were going to read anyway so, we figured we might as well get credit for it. So we would take those courses that were so easy that it was annoying. It was like is this what the rest of the school is like. I mean all you have to do is read and talk. And if what you say is in any way coherent, you win! You show up at the discussion classes and discover that most of the people didn't even read the book. They're not prepared at all. So it was disillusioning in terms of what some of the other majors must really be like, although those people probably were not representative, but that was the conclusion we where reaching. I took music synthesizer classes twice, just because I wanted access to the studio.

0:58:17 PE: ____ playing?

0:58:18 BC: Yeah. They had big synthesizers, big tape machines. Yeah it was a blast. I was the only non-music-major. And the other students generally didn't know much about what those analog synthesizers were really doing.

For an example, I went to the final exam which wasn't a exam, it was performing music, so we go to the final event. First the teacher performs a electronic music piece. And what it is, is a classical piece where he took a bunch of photographic cartridges and mounted them to the table, like this table, a wooden one, and used them as pickups. And then the music, I think there were maybe like four or five performers, and each of them had a score, and it said at this measure take your car keys and throw them on the table at this approximate area. But I got there 10 minutes earlier waiting to see this performance, and there was this buzz coming from the teacher's amplifier, and he was scratching his head what to do. Of course as a guitar player I know what that buzz is. Do you play the guitar?

0:59:35 PE: Yes.

0:59:37 BC: Yeah. exactly where this is going. So I went up to him and said your amplifier's buzzing. He said I know, I don't know what to do about it. I said if I get rid of the buzz do I get an A? He said yeah. So I went to the wall and just flipped the plug around. The buzz went away. He said how did you do that. I said come on every guitar player knows this one. The electronics are such that they are not symmetrical inside there and you have to get that AC plug orientation right. So, anyways it's kind of fun and I really enjoyed the ability to make sounds and music out of electronic gear. I built a Theremin. what a Theremin is?

1:00:23 PE: Yes I do.

1:00:26 BC: I built my first one when I was little. Yeah, they're fun. In fact I've often used a Theremin if I'm talking to high schoolers about engineering and science in general. Theremins are a really good demonstration vehicle, because for one they're kind of weird and spooky, so they get the kids' attention. And the principle on which they're working can be explained in 10 mins to anyone who has had trigonometry. It's just multiplying sine waves and the outcome does exactly what the math says it will do. It generates the sum and difference frequencies. And you just filter out the sum frequencies because they're way too high and the difference frequency is the audio. And the kids go "What!" In fact the reaction I get a lot of times is they'll say "I didn't know there was a connection between math and the real world".

1:01:11 PE: Guess what...

1:01:12 BC: We're missing something in our educational process, but you know.

1:01:15 PE: Right. You ever see the movie about Theremin?

1:01:18 BC: I don't think I have.

1:01:19 PE: Oh, you've got to get this! [laughter] It is fascinating!

1:01:22 BC: [laughter] With Clara Rockwell or whatever her name is?

1:01:23 PE: No. Maybe. But Theremin's life, which is mostly what the movie is about, was incredibly weird. He was... Oh gosh, I can't remember exactly how it worked, but he either was or was thought to be a KGB agent. [laughter] He disappeared for quite a while, for 20 years or something. He left the States and went back to Russia and nobody knew what happened to him. And finally, after the Soviet Union collapsed the records came out. Anyway it's an amazing movie. You'll have to....

1:01:59 BC: I'll have to see that. I used to try to play the theremin, like musically. It's too hard. [laughter]

1:02:04 PE: Yeah, well, another thing you sort of see in this sort of period footage of these classical performances with a theremin at the head of the orchestra.

1:02:15 BC: Oh. You can be really good at it. But when I first built one, I was like 11 years old? I was messing with it, because I wanted to see how far you can extend the range. And by carefully tweaking all the different components, I eventually got this thing where it could sense when you were within ten or twelve feet of it. A Theremin's pitch goes up as you get closer to it. If you move farther away the pitch goes down. And when you get far enough away it's not making any noise at all. Just before it stops making any noise it sounds like it's growling, especially if you turn your amp up really loud and emphasize the bass. So I did that and I put it inside the Christmas tree. If my mother would get close enough to it... It would start to growl at her. Eventually she realized that this has got to be my son who did this to me.

1:03:06 PE: That was in the bio piece in your alumni magazine.

1:03:10 BC: Oh, okay.

1:03:11 PE: But it didn't mention the Theremin part.

1:03:12 BC: No. Yeah I did another one.

1:03:13 PE: It just said that it growled.

1:03:15 BC: Well there was another thing. I did another thing with the Christmas tree once. I recorded a purported message from aliens, and I ran the tape at half speed. So when I ran it at normal it was twice as high. I started it once you got close enough to the tree, so it sounded like the tree was talking. She must have just rolled her eyes – The other thing I did, what the Theremin's really good at, is you can generate any frequency you want in the audio range. So I just remember sitting in my parents' basement having just read that everything has a resonant frequency. And I thought, "Well, I could find out if that’s true for the furnace". So I turned the Theremin up real loud and wanted to see what the furnace's resonant frequency was. It was a big furnace, one of the really old-style big, metal tubes. And, yep, I could find the resonant frequency: the entire furnace was going RRRRM RMMM. It was really loud. And all through the house my siblings were completely panicking, because it sounded like everything was vibrating, they didn't know what was happening. They couldn't really hear the Theremin but they could hear the furnace and it sounded like it was trying to climb up through the house. I had to apologize for panicking everybody in the house. But sure enough the resonant frequency thing worked just like they said. You can find the window glass resonances just like with the wine glasses when you do that. When you hit the resonant frequency they change character.

1:04:34 PE: Cool.

1:04:35 BC: So I never made music with it, but I got some science out of it.

1:04:39 PE: Alright, so anything else about anybody, any teachers you remember in particular that were important, or courses or other students?

1:04:49 BC: Well besides Ken Gabriel, who is a good friend and has been ever since, one of my high school buddies was also in EE. And he and Ken and I... In fact it was the three of us in this department in Senior year. But yeah, we spent a lot of time together. One of the things...

1:05:06 PE: What was his name?

1:05:08 BC: Ed. Ed Balzer. He's been an aerospace engineer for 30 years or something at this point for Boeing. And I think it was Lockheed before that One of the things I discovered as an undergraduate was that for myself, it was way more effective to study in a group than it was to study in isolation. Because each of us was good at different things. The reason I became aware of Ken Gabriel for example was they would collect our homeworks and we'd pass them down the row. And he'd always use these green ruled engineering pads. This exact pad, I love these pads. He would use them and his printing looked like it came off of a printer. It was perfectly spaced and there were no errors and you'd look at this and you'd go "Oh my, this looks like the solution sheet for this homework". And it was page after page of this. The first couple of years of engineering, I call it boot camp, they teach you large systems of linear equations and so on and you do a whole lot of math. And I discovered for myself that I can do that math very accurately and successfully only if I do every single line, don’t skip steps and don't leave anything out. If I try to make sudden leaps I'll screw it up. Ken on the other hand can do it either way.

I just remember they'd pass it over and I'd look at his sheet and I'd go "Who IS this guy? How does he do that?" We'd start studying together and sure enough that's how his brain is. It's very logical, he knows exactly what's being asked, he knows exactly how to do it and all he needs to do it is capture it on the page. And whereas I would do it and go through it and think okay I understood this and this and I have no clue what that is all about, and he'd explain it and I'd think “Ah! Right.” Then I realized that when teachers teach, I think teachers have a preferred way of viewing any given thing. That's the way they look at it, this makes sense to. But really gifted ones can notice the case where they explain the concept that way and 90 percent of the kids get it and 10 of them didn't, and then they can try a second way. It may not be natural for them but they understand it so well they can try a different angle. And now half of the remaining kids get it.

I occasionally thought whenever I wasn't getting something I just wasn't smart enough. And then I thought well wait a minute, I'm plenty smart enough. The issue is whether I'm wired the same way the guy trying to explain it is. So he'd explain it and if it made sense I'm done and if it didn't I'd ask Ken. And he'd have some other side. Well, try this a different way. And suddenly I was much more effective than before a s student. And Ken's the one who showed me how to do that. So that was cool and it's been a useful skill ever since.

I told my boss at Intel this once: I used to think that if I go to a presentation and I don't think I really got it, I'd feel bad about that, like it's some kind of shortcoming of mine or I didn't pay enough attention. But after listening to enough presentations by people I realized that most of the time if I didn't get it it's *their* fault. They did a lousy job of presenting. And since then I've stopped feeling bad about it. Doesn't mean all the time but most of the time. So that's yeah, that's where I learned those things was at Pittsburgh. I also got to play a lot of racquetball and that was good. It was a good way to relieve stress.

1:08:35 PE: Any teachers in particular?

1:08:36 BC: Yeah. Well there was a guy named Ron Hoelzeman, who's still there, and I still see him occasionally.

1:08:47 BC: Yeah. And he still teaches there. I run into him a lot because we're both on a Computer magazine's board or editor staff. And Ron stood out for me because first of all, he taught Digital Logic and I needed that. So I paid a lot of attention in that class and it made a lot of sense pretty quickly to me. But I also remember going to his office once, he was my advisor for sophomore year or something. He doesn't remember this by the way. I went into his office and said "I want to design something. I just want to see what would happen. I want to see if I know enough to pull that off." He just kind of looked funny[laughter] "Why would you do that?" he said. "Well this is what I intend to do as a career and he said "Why do you want to do it now?". I just said "I think it would be fun. I never tried it. I have the urge. I want to see." So he gave some suggestions as to well, think about this or that, I don't remember what it is anymore. But I thought it was cool he was willing to indulge me even though he thought I was off the wall. But he was a very good teacher for Digital Logic stuff. And that was a particularly good thing for me to get good at.

1:09:56 PE: [chuckle] Well, I was just going to ask you. So, at what point did you become... Did you start thinking about computers at all? And what point did you start to think you might want to work in that area?

1:10:07 BC: Yeah, that's actually a good question.

1:10:09 PE: Electrical Engineering, you know, that's a lot to do.

1:10:11 BC: Exactly. Well at the time I was just thinking I could design TV's and stereos, that might be fun. But in 1974 we took a class where there were some microprocessors. They gave us a little board with the micro on it, showed us how to program it and so on. Now, we didn't get along with the teacher at all in that class. He just got it into his head in the beginning of the year that we were a bunch of big babies who didn't work hard enough and expected to get everything handed to us on a plate. None of that was true, but he told us that. "When I see you guys, that's what I think of, you guys were born with silver spoons in your mouths". That was a phrase he actually used. So guess what we did? We went to the cafeteria the next day and borrowed spoons to put on our desks. It's like "ok, buddy, you really want to have this fight?" It was crazy. But we did learn some things in the class, and one of them was how to program this single board computer. I thought that was incredibly fun. It was this seemingly inert block of junk you know, it looked like just a brick, could actually do things and you could control what it was that it would do.

1:11:12 PE: Was it an Intel chip?

1:11:13 BC: No, it was... I think it was an RCA chip, or a MOSTECH. It was one of those 6502, it might have been a 6502. It was one of those chips that was doing really well there for a while and then kind of fell over. Got replaced by other stuff. But yeah, that was my first introduction. And as I mentioned in high school I'd seen the teletype and the line printers, and I thought that was mysterious and interesting, but I didn't know how to follow up on it. But at least in this class...

1:11:39 PE: So nobody in high school... Some high schools by that time were having basic classes things like that, would you have ever gotten into that?

1:11:47 BC: We didn't have that. At the time I was concentrating on physics. And I took all the physics that my high school had and then the teacher said whoa... The physics curriculum was modular. You had to take as many modules, if you took 10 physic modules you got one grade, 12 modules was a higher grade. I took all there was and the course was only one quarter over. I didn't have any problem with the physics stuff in that class. And so the teacher said " Why don't you make some new ones?" so I did and I spent the rest of the time designing new ones. So I was mostly involved with physics at that time. And I still think physics is really cool. I just think it's neat how things work. They work the way I expect them to most of the time. We didn't have any computer-related stuff until basically until you got to college.

1:12:34 PE: And so you had this microprocessor class. Anything else? Any programming or anything?

1:12:41 BC: Well you had to program it. Yeah. I did it. And then we were taking programming classes and I thought that was interesting.

1:12:45 PE: What languages.....?

1:12:48 BC: Well, Fortran was the first one I ever saw. And then BASIC and I quickly decided that Fortran was useful, but BASIC was dumb. I didn't like BASIC at all. I thoroughly distrusted a language that forced you to live at a really low level of abstraction and then gave you a matrix inversion operator that’s at an arbitrarily higher level. Basic seemed to be just a random collection of junk... Who decided what was in here and what wasn't? I just distrusted it after that, it just felt like it wasn't well thought out. So then I took a class that was on programming languages and I saw Pascal and Algol, SNOBOL, all kinds of crazy languages. I particularly liked Algol, I just thought it was a great language and it made perfect sense to me. Admittedly, IO was a problem with Algol. I mean, many languages I struggled with, including C, which is the most famous language of all time, had odd I/O rules. But Algol and Pascal, you could look at those language and you could see the mind of a mathematician behind the design. And then you could see they pasted all this weird stuff on to it to do I/O.

[laughter]

1:13:50 BC: It just felt like that and I think it was that way. Once I took some programming languages classes I felt like all this computing stuff is really cool. But I have to say I didn't, even though people think I'm a computer architect, that's been my background, but I had taken quite a bit of stuff and done a lot of programming before I ever took assembly language. And I swear it was only when I took an assembly language course that I began to understand what computer architecture was about. Until then it was like there's some abstract thing happening back there; if I write the program the right way it will work. That was enough at the time. But when I took assembly I went "Oh my gosh this thing is dumb!"

1:14:26 PE: So did the programming languages didn't talk about registers and memory structures, things like that at all?

1:14:33 BC: Not really. I mean like the Algol class didn't. They didn't mention registers at all in that class. If you took...

1:14:38 PE: If you took a math problem they'd turn it into a program?

1:14:41 BC: Yeah. And then BASIC. It completely hid machine details. You didn't learn anything about them. But when you finally got to write an assembly program, you realized the interrupts have to be done a certain way and to save the state and put it back and oh man this thing is operating at a lower level than I ever imagined. I had no idea you actually do have to generate a memory pointer and output this data and later you have to bring it back in and oh man, and that's when I start thinking well, wait, for you to do that, you have to transfer the data back and forth over some shared path and if that path is slow, it's going to slow you down and well duh, it's if you tie enough of such observations together, you're a performance analyst. If you're an architect, you just take the mind of a performance analyst, complain about the result and think of ways to fix it, right? That's of course subject to constraints from marketing and economics and so on. But just fundamentally, you just see what doesn't work right and fix it.

1:15:36 PE: So, did you get much hands-on time with computers at that point, or ... it was probably a batch world and you...?

1:15:45 BC: Yeah, freshman year is batch world. Freshman year was card decks with punch cards. It was sophomore year or junior when time sharing systems were starting to become available and I think it was senior year, 1976 to 1977 this thing called UNIX had just come out and some friend of mine had an account and he said you should play with this, this is really interesting. I sat there and I said, How do you do a DIR? I want to see what the files are? He said, Well, that's not DIR, you do 'LS'. So, I did LS and I said, "Well, look at that. Why do they call it LS then?" I still don't know the answer to that. Then it got interesting. He said, "You can do things like you can take the output of what ls is going to present and dump it into a file or push into the input of some other program." I said, What? You can't do that on DEC mainframes, you know, like the PDP-10s." My inner curious hacker immediately appeared, and I said, "Well, if I could put the output into a file, what prevents me from taking that same file that I'm writing and saying tack that on to the end and what would happen, ah, let's find out" [laughter].

1:15:49 PE: Yeah, yeah, yeah.

1:15:51 BC: In the early UNIX systems, the answer was that they filled the file system and died. That's what they would do. So, I tried that a few times before I realized the machine keeps crashing because of what I'm doing and all the people sitting around are going, Damn, it's down again and I finally realized I was the reason. UNIX has become much more forgiving since then.

1:17:14 PE: So let's see we've talked about programming classes and digital logic and synthesizers, English. Anything else from the Pitt years?

1:17:28 BC: I had programming job for a couple of years as an undergraduate. There was a professor at Pitt in metallurgical engineering, material science, whose name was Shirl Breitling and he had been basically a ski bum at the University of Utah. Apparently on one particular run, he ran into a tree and smashed his leg to smithereens.

So, he was down and out for many months and in that period of time, he joined Pitt's faculty. He was trying to write a computerized implementation of a mathematical structure he had created to describe crack propagation, cracks in materials, which is particularly important in a place like Pittsburgh with all the bridges because there had been cases where the tug boat captain goes under the bridge and says, "There's a huge crack in one of those I-beams." In the I-beam, that can't be good and they want to know things like when there's a small gap, how long will it take to become a big one and so on. He came with a really complicated mathematical model to describe such crack propagation. So, he wanted to program it and see how realistic his crack propagation model was and he wanted someone to help him do that.

So, he personally did the original implementation of this code in Fortran and then after that, he basically gave it to me and said, "Here, clean this up, add a few features and so on". I looked at his code and it was just inscrutable. He had not ever had a programming class in his life. He was a just a smart guy and he was incredibly dedicated and he got this program working but he didn't know the first thing about coding style. Variable names, for example.

He would take these complicated partial differential equations that were like seven miles long. He would just chop them into pieces and he would arbitrarily assign the intermediate names to these things like “QCXY1” that random set of letters would appear somewhere randomly in the next line. It was a nightmare.

1:19:23 PE: No documentation at all?

1:19:25 BC: He didn't know you needed that, no comments in the code, no nothing. There was another problem where he had some kind of surface, I always thought of it as kind of like Velcro. There was a surface where he could leave molecules in a chain kind of a fashion and the number of times that there's a connection on the top of that surface, it would generate some physical property that he was interested in. He wanted more or less of those and he was trying to model all this as to how many of those links occurred, what's the average length of a chain and how often would it stop and a new one would start and so on? We wrote a piece of code to do that and I wrote the inner loop of that solver by just basically sitting and thinking really hard for about half a day once.

That taught me something because I had first taken an attempt at trying to code my way through the problem, but the complexity stumped me. That's a way of programming that no one should ever do and somehow some kids fall into it and some never get out: you kind of know roughly what you want to do, so you refine your thinking as your fingers spit out source code. You start typing in source code and you modify it as you go and it means you've never thought deeply enough and you're going to get hammered by one bug after another and eventually you're going to have to give up because the fundamental structure of what you're trying to do is completely wrong headed. I had attempted to do that and kind of shied away from it only when I realized things were not going well. I need to think harder before I do this. I spent an entire Saturday morning at my parents house, I can still picture sitting at the dining room table with my eyes closed because I thought if I can think carefully enough about how this has to work, I'll be able to capture it in code but if I can't, the code's never going to show me the answer. It isn't going to happen that way, I've got to think of it first and sure enough, after thinking about it hard enough, I came up with an algorithm that would accomplish the goal. Capturing that algorithm in the code was simple after that and it worked and I thought man, I felt really good about that partly because I'd solved his problem but mostly, because I'd realized something fundamental that I felt I was going to use for the rest of my career. That turns out to have been true.

1:21:36 PE: I guess so.

1:21:37 BC: And that's where I learned it.

1:21:44 PE: But talking about jobs, reminds me to go back a step and ask whether you had jobs in high school?

1:21:59 BC: I taught guitar. Every Saturday I'd go downtown and I had a bunch of students that would come in on a half hour basis. I hated that by the way, I really didn't enjoy it. that the guitar part's easy but the student part, I don't know how people do it, don't know how do teachers do it. If you don't practice, you don't get better, just by dint of getting older, it does not make you better. It was really hard for me... you would see lots of kids whose parents put them up to this guitar practicing and you could also see that the only time that they have the guitar in their hands was in front of you on Saturday afternoon. To make progress at that glacial pace, it's just not going to happen.

1:22:43 PE: Right.

1:22:44 BC: And yet, you don't want to push too hard the other way because they're not trying to be professionals and maybe they're busy, I lacked any context back then. I just knew that it was frustrating for both of us.

One particular guy was a parking lot attendant, boy did he make progress, he did nothing but practice. It was a breeze teaching him, I thought I must be the greatest teacher in the world, this guy's making such progress. But most of the students, especially younger kids, would just not practice, so it was just not going to happen. The other part of the problem was coming in, in the morning and my heart would just sink when I'd see this long day in which all the day is chopped into half hour increments and I'm on this path and I can't get off of it, I’m stuck here all day long. I didn't really like that and of course it had its counterpart when I got to Intel in my later management days and I got to the office in the morning only to realize I've got meetings all day long, so I'm not going to make any progress in anything today. [laughter].

1:23:46 PE: I know exactly what you're talking about. It's what my life is like now.

1:23:50 BC: Yeah, I really didn't like that.

1:23:55 PE: Any other jobs in college?

1:23:57 BC: In college, I played in a band. No, I mean high school was just informal stuff, just I was teaching guitar and playing and in college I was playing in a band.

1:24:07 PE: Yeah?

1:24:08 BC: I did that for a couple of years.

1:24:10 PE: What kind of band?

1:24:11 BC: Pop, top 40, rock 'n' roll, you know, we were playing lounges, we played lots of weddings, parties, all kind of crazy stuff and by the way the drummer for that band ended up playing for Michael Macdonald.

1:24:24 PE: Oh, yeah?

1:24:25 BC: He's a really good drummer and he's been a professional ever since but... Yeah, I kind of knew he was pretty good.

1:24:32 PE: Yeah, yeah.

1:24:33 BC: But yeah, on our good days, we were not too bad. On our bad days, we sucked but hey.

1:24:42 PE: Okay, so you graduated from Pitt in 1977?

1:24:46 BC: Yeah.

1:24:47 PE: Correct?

1:24:47 BC: Yeah.

1:24:48 PE: And sounds like you must have gone to, on for your MSE almost immediately.

Bell Labs

1:24:55 BC: Immediately, yeah, but that's because it was, that was Bell labs. I worked for Bell labs, now let's see when was it? That summer, 1977.

1:25:04 PE: The summer after college?

1:25:05 BC: Yeah.

1:25:05 PE: In 1977?

1:25:06 BC: It was a Bell Labs program called OYOC, one year on campus, a lot of people went through this. You graduate, you work for them for the summer, they send you away to one of their pre-approved schools and literally half the class went to Stanford. About six of us went to Carnegie Mellon which is where I went because my then fiancé was still in Pittsburgh.

1:25:30 PE: Okay.

1:25:32 BC: And then, after that one year of grad school at Carnegie Mellon, I went back to Bell Labs. And worked there for a couple of years.

1:25:43 PE: I see. Is this Bell Labs in New Jersey?

1:25:45 BC: Yeah, Holmdale.

1:25:46 PE: Let's hear about Ellen. You mentioned that you were engaged at that point. How did you meet?

1:25:56 BC: Yeah, yeah, I met Ellen at a Halloween party in 1975. A mutual friend was having a party and we met there. She was a student at a community college then and then a couple of years later, she switched to Pitt and got a degree in computer science, and she was also an OYOC student for Bell Labs and went to Princeton for her masters.

1:26:23 PE: So you guys were apart during that period?

1:26:26 BC: Well, we've, let's see, I think we have spent three summers apart so far just because of work related stuff, the summer in 84 when I was a grad student on my PhD, I was here in Oregon at Intel getting the information I needed from the heads of the designers and she was still in Pittsburgh at the time. Then there was the summer she was an OYOC student at Rutgers.

1:26:50 PE: Explain that acronym, just for the...

1:26:53 BC: One year on campus.

1:26:54 PE: One year on campus, so that's?

1:26:55 BC: O Y O C.

1:26:55 PE: O Y O C, thank you for that.

1:26:57 BC: They say it so often it becomes a word for people that go through it. It was a good deal, they gave you stipend, they paid for your tuition, your books, you had a job when you came out. It was cool.

1:27:10 PE: And how did you get into this? Because you must have had to apply for it while you were a senior.

1:27:14 BC: Correct. Well at the time, Bell Labs was this legendary industrial research lab and all the smart people had gone there and tremendous innovations came out of there. All the seniors would talk to each other about where they were going after graduating, and several of them were talking about if I can get into Bell Labs, that's where I'm going to go. I thought, well, it must be a good place. So, I didn't really know a whole lot about it at that time. It was either that or the space program, I always wanted to be an astronaut too and go to NASA but they never picked me. I actually applied to be a mission specialist for probably 15 years straight back in the 80s.

1:27:57 PE: Really?

1:27:58 BC: Yeah and I even know that during one particular, year they actually checked on my references and called all the people that I'd provided as references. So, it got that far but I don't think they needed a computer designer on the space shuttles.

1:28:10 PE: Okay, so what was Bell Labs like?

1:28:15 BC: Well my first year, my first six months of it was pretty much hellish. I had joined a systems engineering group, which as it turned out was a terrible fit for me. It's hard for me to remember what it was like, it's hard to explain what actually happened but here's as best as I recall. When I was a senior in college, I interviewed at Bell Labs in New Jersey and that was the first airplane flight I'd ever taken. You get off the plane in New York and somebody takes you down to the Molly Pitcher Inn in Red Bank, New Jersey and you spend the day interviewing. In fact, you spend as I recall a couple of days and then you talk to different groups to do different things and they do their best to tell you this is what our department does and if you came here, here's what you'd do. But it didn't really sink in for me. I mean it's very hard to make sense of it, there's no context, I knew I wanted to design things but I didn't know how to translate that into what they, to see was doing stuff like that. And in the groups that you've talked to, will give you different levels of feedback, like "we really want you to come and work for us" or "we're going to make you a job offer" and as a senior you go well these guys really want me there must be a good connection there and these guys may be lukewarm, so, in the end I ended up taking a job with this systems engineering group. My first boss there, I shouldn't say his name, because we didn't get along, here's what he was good at, there were a bunch of people back then that could go up to a pay phone and instead of putting a coin in, they could whistle...

1:28:51 PE: Yeah, sure.

1:28:53 BC: And get free phone calls.

1:28:54 PE: Like Captain Crunch?

1:29:55 BC: Yeah, and they had blue boxes and black boxes and all kind of boxes and his job was to make the design of those phones in such a way as to make that harder, or to make that whole exploit impossible. Now they didn't tell me this time, but that problem was about to go away on its own. The reason you could do that back then was that the signaling was carried on the same wires as the phone conversations. So the whistling sounded like control signals. Within 5 years of all this happening they switched to an entirely different system where the signaling was not on the same wires, so you can't do that anymore. But meanwhile my boss was spending all of his time trying to recapture the lost revenue. And he had been at Bell Labs forever. So he knew all the cool stuff about what is inside the handsets and that sort of thing. That part was fun.

But they were doing economic studies. And the one economic study that I ended up working on was if you moved a certain amount of functionality from one telephone branch office to the central office, how much money would the Bell System save. I was unprepared to do that on my own. But they paired me up with an experienced systems engineer and between the two of us we ended up with a useful estimate. Now the estimate, I still remember the number, the estimate was, you save at least 20 million dollars and as much as 120 million dollars each year if you do this for as many years as the situation stays. They said it’s too small, we are not interested in that. And at that moment I realized, I don't want to be in this field. If you are not interested in numbers like that, then, what am I going to do that's going to have an impact enough for you to know it is.

1:31:34 PE: You can't save at least billion dollars.

1:31:36 BC: Pretty much. And the Bell system back then was just a different place. So here is the thing. I came back to my boss after a few months and I said, that I am a newly wed, I am living in an apartment with no furniture, I don't like this job, I don't see it getting better, I can't sleep, I can hardly eat, I am not enjoying this, something has to change. I have joined the wrong group, I need to fix the problem. I want to fix the mistake that I made. And he did not say, I understand, let's find a good match for you, that Bell Labs wants you, if this isn't the right job. He did not say that. What he said was if you are so stupid as to try to get out of my group in that way I am going to make life hell for you. I will personally make sure that you never get another job anywhere, no one is going to trust you. He just went on this flaming rant. He was just basically shouting at me. And it was funny because if he'd taken a reasonable approach he might have gotten me sticking around for another year. I wasn't threatening to quit. Bell paid for my masters, I am going to try to pay Bell back and I am not going to walk out. But this job is killing me. Instead he was such a jerk about it. I went to see his boss. And that was illuminating, because his boss said, what, he has a problem. He just doesn't know how to deal with new hires, we've had a problem with his supervisory skills forever. Ignore him.

He said, What do you want to do? And then I said I want to be a designer. I want to design things. And he said like what, we do lot of designing, what kind of designing. I said maybe computers.

He said, well we have a microprocessor design group here. Let's get you to go and talk to those guys. I went over there and within 3 days I switched and man, it was night and day, I loved it over there. It wasn't like there were no problems. But they were the problems I understood. They were interesting problems. Life was so much better after that. That job was perfect for me.

To develop systems with microprocessors back then you had to unplug the CPU chip and plug in a cable that went to a box called the in-circuit emulator. That's how you would debug: your terminal would talk to the in-circuit emulator, it would pretend to be the micro processor and it would report back to you what was happening in the system. My new boss said we have dozens of these in-circuit emulators that are not working right.. He said we have been just collecting them, I don't know what to do with them. So I said, I'll come in and look at them.

1:34:03 PE: What was this?

1:34:06 BC: It had a chip in it.

[background talk]

1:34:20 BC: Picture a single board with a computer chip on it. And it is doing something but and you don't like what it is; maybe it has lights displaying that are not right. How do you figure out what's wrong? You can't reach in there and see what's in the register. But with the in-circuit emulator you could pretend to be that chip, run the code the same way he would, but if you ever hit address 46, take a break point, stop right there, show me what's in the registers. That's the kind of thing you could do with these in circuit emulators. And back then you had to have these emulators or you were not gonna sell chips to anybody. They were part of the deal.

1:35:00 PE: I don't remember debugging like that.

1:35:02 BC: Essentially he said we’ve got a bunch of these broken, I don't know what to do with them. See if you have got any idea. So I went in on a weekend and... And so I and a guy who designed them, we spent the basically the whole weekend reviving that bone pile. It was great fun, because it was debugging real hardware. Each one had a different thing wrong with it. But it was the kind of thinking process that I really found fun, hard to resist. Very challenging. We came in on a Monday and told my new boss you had these 30 broken and now you have only two. You got 28 working ones now. He was happy and seemed impressed. I knew I had a home here, this is good.

1:35:44 PE: Who was your boss?

1:35:45 BC: Which one, the new one? Al Hoffman was his name. He worked for a guy named Lee Thomas. They were running the micro processor design group. But not everything was great there. One problem was they designed a more complicated in-circuit emulator and started shipping these boxes, which were big and cost 60000 bucks or so. They were seriously expensive. You needed them to debug your microprocessor-based system, but they were really screwed up, very poorly designed. They would send me out to Indian Hill Chicago all the time to hold the hands of these guys trying to use these things. My first introduction was this: I'd walk into their lab and on the board were lot of gravestones. Each one had a name of a specific in-circuit emulator. They had them all piled up against the wall, none of them worked right. My group would send them a new one, and they'd just add its little gravestone on the wall with RIP whatever date they got. It was horrible.

These guys just thought my design group were morons, and they had a point. In some sense I can definitely see why they thought that. That was my introduction to trying to do field support on a complicated machine. Indian Hill, Chicago is where they designed all the really big telephone switches called ESSs. Electronic Switching System I think. There is a number one ESS and there is a number five ESS. The number one is the huge one that you find in the middle of a big city. These things are, I don't know how big it was, much bigger than the first floor of my house. That one machine was gigantic. It was filled with mechanical relays. So when you hit the reset switch, they would all clack at once, the din was unbelievable. And here was the problem. All of that was being controlled by our microprocessor. When something went wrong all this stuff would reset. You couldn't miss it. You could tell it was resetting from down the street.

And so they were saying your damn thing is resetting on us all the time, we don't know why. And so that was the problem I had to go out and debug. The first thing I noticed after a few days of struggling with this, taking careful notes, just being observant was... Luck played a role. It was an evening and I have been sitting here all day staring at the screen waiting for it to reset again thinking if I don't come up with an idea soon this is going to be bad. I'm gonna have to go back with the tail within my legs. And as I was watching this I realized it just reset and at the corner of my eye I could see somebody left the room and he had flipped the light switch off. I thought, no, couldn't be! But what the heck. We will try. I'll reboot the machine. I went over there and I flipped the light switch. And it rebooted. How in the heck can a light switch do this?

I think I asked one of their more experienced systems engineers, I mentioned this to him, and he said what, if your power supply does not have sufficient rejection of AC line noise it will pass that noise through to the digital logic chips inside your emulator. And if it does that your microprocessor is not going to like it. Sure enough, that's what it was. So we got that fixed within a few days, we replaced the power supply with a better one. It So I got that problem fixed. But it still wasn't doing everything the customers wanted. There were obviously other problems. The next thing I noticed is if you look at the back of this in-circuit emulator, there's a backplane, and in a backplane normally you would expect a big sheet of copper to be the ground plane and another sheet of copper to be the power plane. That would give you low inductance and low impedance, both of those are really important. And I had never looked into this because I had never designed these before but I didn't see any planes. How is the ground current getting from the power supply connector this end to the card that needs it at this end?

And the more I stared at that I realized I don't think there is one. They didn't put a plane in here. But then how was it connected at all. It turns they used a piece of copper about an inch wide on the back plane and that was the ground. I took my scope probes and I measured it. There was a voltage drop. At one end I saw 5V and at the other I was seeing 4V. You can't do that. And that was DC. I mean in terms in inductance there's a lot of switching, with sharp edges, causing voltages to be induced, there is no hope. The +5V was the same way. It also used a tiny strip of copper instead of a real plane. We redesigned the back plane, and it still didn't work right. At this point I have no faith left in the designers. I don't trust anything. So I took my scope and I just started poking around the signals and I realized they all look terrible. They don't even look like digital. It is just a mess. I checked one end of the 6 foot cable, then I go to the box end and the signals are not good there either. The long end of that story is when I finally finished debugging this I found they had taken a 6 foot flat ribbon cable and they had taken signals in a centrally randomly distributed them on the cable. They will go like data, data, address, clock. Didn't really matter. what cross talk is? Over six feet, you might as well tie the signals all together. They were talking to each other like crazy.

1:41:21 PE: Unshielded, yeah.

1:41:23 BC: Unshielded, not near enough ground, it was incredible. And of course it is attached to this gigantic switching machine. It's generating noise like crazy. So I redesigned the cable, kept all the signals apart, put a ground wrapper all way around it, I mean I over killed that sucker. And finally it worked.

1:41:41 PE: [Laughter]. Fascinating.

1:41:41 BC: That was my introduction to engineering at Bell Labs.

1:41:46 PE: Let me just tell you one story because your story makes me think about it. In mid-1975, my first job after high school, I was a computer operator. I was operating a Honeywell 64/20 which had those big old tape drives. And we had a glass window on the front which would come down very smoothly, like something from Star Trek, for you to mount and dismount the tapes. And we started to have this problem where these tape drives would just dismount randomly. There didn't seem to be any rhyme or reason to it and the field engineers would come out and sit there for 3-4 hours and it would never happen and then they'd leave and then it would happen.

1:42:31 BC: Then it would happen. [chuckle].

1:42:32 PE: When they came sometimes it would happen, but they couldn't figure it out. And one day one of these guy was sitting in there waiting for the dismount and just noticed that it happened, and it was about I think about 10 o' clock in the morning and what was going on was the sun was coming just into the position where it would hit one of the infrared sensors, and the sunlight was just at the right angle that it would cause this thing to happen. It took two months to figure that out.

1:43:12 BC: It's always about the assumptions that you don't realize you are making. Always.

1:43:18 PE: Okay. So, you were in the, what did they call it, the Micro Systems Design Group at Bell Labs. And you worked on these boxes and that lasted a year

1:43:41 BC: Probably more like 2 years. And meanwhile we were designing the next generation of 32 bit chips, trying to go directly from the 8 bit generation to 32 bits.

1:43:50 PE: Right. I saw that in your CV. That was interesting.

1:43:54 BC: Yeah.

1:43:54 PE: Just went right over the 16 bit phase.

1:43:56 BC: They did that very intentionally. I was at the meeting when it was decided. They said we are Bell Labs, we are not going to follow others. There is already 16 bit out there. So there is no point in doing 16, let’s do 32. There were some dissenting voices along the lines of well, you can maybe fit a 32 bit micro processor on a chip but it may not be very fast. Because you won’t be able to do any performance enhancing functionality. You are just going to use bare bones. But they said it still probably better to do that. And we will catch up because the technology will get better. That part was probably true.

This was a classic meeting, I was the most junior guy there. I was standing against the wall watching this, you know, all the big brain guys coming in and they say okay our job here today is to design the instruction set that was for the new 32 bit chip. Now we don't even have to be compatible with anything. We have a blank sheet. It was kind of early in the microprocessor business, like 1980, 1979. Microprocessors had only been around about 9 years at that point. But they knew that that was cool because if they'd had to be compatible with their previous 8 bitter it would have been a different game. But instead they specifically said look we're just going to recompile and we'll just pay the price and we're just going to move on. And we want to be 32 bits. And I remember thinking this is going to be great. I'm so lucky to be here on day one of a new design because I want to see how it's done and learn from the Bell Labs guys, because they've got to be really good. And then what happened next was dismaying. They took a book on the VAX 11 assembly language, literally threw it on the table, and said “what we're going to do is really go through this book and pick and choose which instructions we like”. And I thought "wow, what did they do? What did the VAX guys do? This can't be the right way to design a new instruction set". I was just aghast.

The VAX had, in many ways, a useful instruction set. But it had some extra junk. It had some complicated instructions that probably weren't paying their own way. And this was at the very beginning of the RISC/CISC wars. There was already the sentiment that you could get carried away with your instruction set. That was already out there. We knew that people were wondering about it. And it could be that's what he was thinking of. I don't know. All I know is that I'm standing there thinking this is just not off to a good start. Where's the analysis? Where's the understanding of why we should do this, not that? I mean, you can't just say that the VAX guys must be smart, let's do what they did. That was not cool. But it made me think, okay they don't know how to do this. There must be a better way. And that's what I was thinking when I left to go back to school a couple of years later.

That's when RISC and CISC debate took off. That's why I was kind of extra sensitive to what instruction sets were all about. I mentioned I don't like the religious attitude toward the design. I think that it's good because you measured it. It's not good just because you followed all of the secret rules. It just sort of felt to me like this was way too mystical. There had to be a better way. That's where these attitudes came from when I first saw the original RISC/CISC publications. I was somewhat sympathetic to the new RISC ideas, but I felt like the conclusions that were being drawn were grander than the data supported. I understand if you're trying to get something started in a new field or whatever, it's easy to get carried away. Even if you don't, your practitioners will. Your acolytes will. That's one reason why I got involved in it early on, because I wanted to make sure that the right lessons were being drawn and it's easy to get carried away and claim the moon and stars when you're not entitled to, you haven't proved that yet.

1:48:12 PE: But that's... I think you'd say in some way your papers, but it does fit your role with us in basic engineering principle, simple is better.

1:48:20 BC: That part of it does seem definitely true.

1:48:22 PE: It's easy to see how you would... Intuitively appealing idea. But maybe in practice it isn't like that.

1:48:29 BC: Well we're probably going to talk about this in depth later on. But the first thing that bothered me about it was there was what I considered a too-quick willingness to blur the distinction between micro architectures and architectures, in the early RISC work. I wasn't as convinced as I later became at Intel, but maintaining a single architecture is a really cool thing to do. I mean IBM had made tremendous amounts of money by staying compatible with System/360, 370, all along the way. You pick the architecture and you change what's underneath it. As long as the code can't see the difference you're in great shape. And yet the RISC camp came along and said as long as you're willing to make tradeoffs across that boundary all over the place, you can go faster. And I thought, so what? If you take the brakes out of your car you can go faster too, but it's not as useful a car. There's more to it than that.

So, that was the genesis as to why we became involved in that but mostly on the other side of the RISC/CISC wars: it was my feeling that too much was being claimed. And in claiming too much I was afraid of two things. One was losing the fundamental real contributions of what the RISC researchers were doing. And two was de-tuning peoples' intuitions about what's important, what should be researched and what conclusions should be drawn. So, I felt that the whole field was going to take a left turn and have to recover from it later. I don't know if I would characterize it as having done that at this point. I do think we trained an awful lot of people that technology is all there is and performance is all that matters. I don't think those are true. Economics is what matters. And there's a lot more to anything besides, including computers than just performance.

My attitude was that, what's the fastest car on earth? I don't know what it is. Let's pick a Lamborghini just for fun. Why doesn't everyone drive Lamborghini's then? Because not everyone can afford those; there is something else to this issue besides performance. But in the computer field if you're going to publish a paper it's easy to get something published that says if you do this your computer runs 20 percent faster. That gets people's attention. But it's much harder to say, if you do this you'll run 3 percent faster but it will be 5 percent cheaper and much more reliable. Good luck publishing that one! [chuckle]. And yet if you're in business like Intel you care about those things.

1:50:57 PE: Yeah, yeah. Okay, so to close out on Bell Labs, any other people that were influential with you or that you influenced or colleagues that you worked with later?

1:51:12 BC: Oddly enough, there's a guy I don't remember his name at the moment, since I was in Indian Hill, Chicago a lot, I remember being there one night and I'd only been married a few months and I hadn't thought about having kids at all yet. Suddenly the guy I was working with there he said, "I'm sorry but I have to go home. I've got to go home and see my kids. They are going to be going to bed and I just don't want to miss it. My kids are the highlight of my life". I remember thinking "that's interesting. Tell me about that, you know". Because I was just thinking about computers at the time. And I said, what's it like becoming a dad? He said it's the best thing on the planet because I love my kids they are the best thing that's ever happened to me. I went holy cow, I've never thought about that before. That was the first time I really started thinking, geez maybe I should have kids. [chuckle]. So that was a Bell Labs thing. [chuckle]. Who knew, right? But in terms of technical stuff, yeah it did. Here's another incident. The guy that was the chief architect of the 32 bit micro that I was telling you about, his name is Don Blahut.

1:52:14 PE: Spell it?

1:52:14 BC: B L A H U T. And I remember coming back in a car from the airport, from Newark airport. We'd gone to a meeting associated with the design of this new micro. I said, "Don, you're awfully quiet". And I didn't know him very well but I said you don't seem real happy about today's outcome. And he put his head in his hands and he said, you know, I'm in charge of this design and I've lost control of it. He said, when I laid it out I had blocks this function here, and that function there, and this is what they say to each other. But the designers have been borrowing back and forth and moving things around so much that I no longer know what's going on. It's one big pile of stuff now. When you've lost the structure of something complicated, you're screwed. You revert to making tiny little decisions the implications of which you can't even predict. As soon as you do that you're just waiting for the axe to fall. Something bad is going to happen to us, its gonna kill us, and I can't prevent it. My immediate reaction was, "oh crap! [laughter] The leader of my team just said we're going to die".

On a more long term basis I thought now there's a lesson to carry forward. You do not let the structure disappear because if you do you've lost your intellectual leverage on what you're doing. What Don was trying to do at the time was complicated. But what we did after that would blow it away. The stuff that we did at Intel was way more complicated than that. Sure enough there were times that I felt like we were close to the abyss but at least I knew there *was* an abyss. He had shown me that there is one and somebody can fall into it. So I give Don credit for giving me an honest answer to the question at that time that was going to do me some good later. It really did. I had many opportunities at Intel to go to my boss and say I've been asked to do this feature by marketing for perfectly good reasons, but after we evaluated this technically it's my conclusion that you're asking to inject too much risk into this project. Or the return on that investment isn't high enough to warrant that much risk. And so I'm going to argue against it and the marketing guys are going to scream and call us idiots and say you didn't understand what we said. I don't know how many times I prevailed on that, but at least I had the terms to couch it in.

Looking back at it I think we did a decent job overall of hitting those compromises. Those were the hardest ones. It's easy to go to your boss and say they ask for this, we looked at it and we'll put it in. And it isn't going to cost you anything. Who wouldn't love that? If you say, well, we're going to put it in but it's going to cost you 3 percent of the die, now all the sudden it's got a real price tag and the boss goes, "oh, I don't know. This hurts. Can you please make it 0?". Er, no. We often had those discussions. Who wouldn't want something for free? Anytime the discussion hinged around risk it got awkward really fast. Because engineers are quantifiers. Tell me a percentage then I can deal with it but if you just say I want to stick a feature in it that will make it this much faster, you can quantify that, it will cost this much in schedule, you can quantify that. It will use this much die size, I can quantify that. I get to sit and say, well wait a minute, it's going to be awfully risky in terms of complexity. There might be some bugs. I don't want to do it. You can quantify your answer. I can't quantify mine. Now how do I trade those off?

Those are very difficult conversations to have. I was lucky that people like Albert Yu, my executive VP at Intel, would sometimes say look, I don't completely understand the argument you're making, for good reasons because you're basing them on things that inside the chip that I can't possibly know, and you're comparing other things that I do know and I know the price tag of. But in a sense you're asking me to trust your intuition, because I can't do it myself. I can't do it based on what I know. I said, I agree with you. That's the right way to view this. But, if the day comes that you can't trust my intuition as your chief architect you need to fire me and find somebody else. Because I don't think there's another way to do this.

I don't think it's a shortcoming of mine that I can't make crystal clear to you every single thing that I think we should do or not do. I don't think that's a problem. It's not necessarily some lack of communication skill on my part. It means I can't give you all the details that I consider in making decisions. And by the way, I don't know all the details either. Because the people that are feeding me the information face the same situation. They know a lot more than they can tell me. I just need to figure out when he says it, take it to the bank. If this guy over here says it check into it a little bit further first. Because they don't mean the same thing by the same words. You get to that level of understanding with the people you work for and you become a more effective team. And so Albert, bless his heart, he would be willing to play this way but he wasn't completely comfortable all the time. I think that's one of the reasons why the team worked so well.

1:57:03 PE: That's Albert...

1:57:04 BC: Albert Yu. Y U. He was the executive VP in charge of all of Intel's micros in the 80's and most of the 90's.I used to come home if I was frustrated for one reason or another, I would come home and my wife, Ellen, would say okay, what did Albert do this time? So no, I wasn’t always really fair to Albert. But frustrations, you can always find them.

1:57:34 PE: Anybody else at Bell Labs, memorable? Colleagues? Anyone that you worked with later?

1:57:40 BC: There was a guy, whose name I’ve forgotten over the years, a guy that was many levels up in Bell Labs management. And after I did a few of the things they requested of me, they started thinking maybe I was competent and I started getting picked on for odd jobs. At one point this high level guy called me up to his office which was like God calling me on the phone, or it felt like it at the time. And we're sitting in his office and he calls somebody, gets on a conference call. And the guy says something like alright what are you going to do about our problem? Jack something was his name. And Jack says well, that's why we've got Colwell here. We're going to figure out how to solve your problem. I'm thinking, what problem does he have? And I'm to the point where the voice on the phone says well I sent you a description of the one of our engineers thinks we're up against. And he leans over and he reaches in the garbage can and he goes, yeah I got it right here. [laughter] He puts it on the table.

It turns out that they needed some handholding associated with these in-circuit emulators that we were sending them and so this Jack guy sends me out to Indian Hill that day with just the shirt on my back and he says when you get there, buy some clothes. You'll be there for a few days, as long as it takes to solve this issue. So whatever it was I did that and I solved whatever it was and I came back. He sent Ellen and me to the restaurant in New York’s Trade Towers (the ones that aren’t there any more post 9/11). Windows On The World, at the roof top. It was a cool thing. The lesson that I got from all of that was, there's a certain level of management that you apply your resources against what you're being told you need. And even if it's expensive or whatever, you have to see what's appropriate at his level and he was playing with much bigger toys than we were doing down at our level.

When I eventually reached a level in management where I was higher than he was and you get signature authorities that start to hit numbers like $50,000 for example and you start going okay if you want to drive this business as fast as you can forward there are times when that kind of a price is what you pay because you're balancing that against the billions that the products will bring in. This became more clear to me when we had issues late in a production just before we going to go on the serious production on certain chips. And I would get daily phone calls from people like Andy Grove. He say, "Hey, I just want to mention to you, you know, everyday you delay the thing it costs us about 7 million bucks." Great, thanks. Let me work smarter for awhile. He makes sure that you knew that there were serious consequences to what you were doing.

2:00:33 PE: But as you say, that also means that you can spend a fair a bit of money to stuff that lost.

2:00:38 BC: Yeah, absolutely. Another example: I said, "Okay Albert, when you start out an effort like this chip development, you have some expectations as to what the validation plan is going to be. I'm going to keep testing pre-silicon on the machines throughout the project against our test-plan." And as you get halfway through it, a whole lot more than you did when you set the plan. So now that there's a lot more work to do because it seldom happens that you initially plan too much.. The work only expands. So I went to Albert and I said, "So, if I don't grow the validation resources, one of two things will happen. Either we're going to delay the project or we'll put it out with much higher risk than I anticipated because there are things that needed to be tested that weren't. And so I what I would like to do is throw more people at this and I'm looking for like 10 people for the next several months". Albert said, "Can you prove to me that those extra 10 will work at the same efficiency as the ones you already have?" And I said, "No. In fact, I could probably prove to you that they’ll work less efficiency. It's going to take them some time to come up to speed, they are incremental resources on top of the installed base, so their targets are not going to be equally important". But that's not the important thing. The important thing is what are we getting for what are we paying, and the cost of 10 heads in the context of this project is tiny, you can't even see it with the project’s existing 850 people already on it. Those extra 10 heads are roundoff error. He said "Yes. But how do you know that adding the 10 will make it work? Maybe you'll end up pushing the schedule out instead."

And I said, "Yeah, okay." I've talked to Fred Brooks. I've read his book. I know what the point is about adding heads can sometimes make a late project later. And if that will going to happen, I'll be telling you that and I wouldn't be asking for the heads.” We both had a point. He did. I did. But at least we could have the discussion and we could choose whether or not to apply the resources. But yeah, the number just gets bigger. Especially at Intel, the numbers get really big, really fast. After a while you just stop thinking about it, actually. It is too frightening like when the FDIV errata episode happened. I spent a few nights thinking yikes, if we had a bug like FDIV in our new P6 chip, what will happen and of course, wish we had the patch facility so I can always help work around at somehow. But they only shipped 5 million parts and that cost Intel a half a billion dollars to recall those 5 million parts. Well, we're going to ship hundreds of millions within the first year or two. If we have to recall them at that price tag, we have a serious business problem. And as it's my team doing the validation, we're on the hook no matter what. But at some point you just have to think about that hard enough to get it, try to get it right, give your best shot. Then, you can put it away or it will drive you nuts. It will get nowhere to become paralyzed by the fear that something is going to happen.

Part 2 (1980-1990): Bell Labs continued, CMU (PhD), RISC, Multiflow

Bell Labs (continued)

0:00:00 Paul Edwards: Okay, okay, so this is part two of the interview with Bob Colwell on Monday, August 24, 2009. Let's see, we are now, we have been talking about your time at Bell Labs and let see you're CV says you left Bell in 1980.

0:00:26 Bob Colwell: Yeah.

0:00:26 PE: But by then you were already involved in your PhD degree. Correct?

0:00:33 BC: No, I do not know, let me think. I left Bell in August of 1980 and started school in September.

0:00:42 PE: Okay. So you were basically full time, the first year at Bell was this, unpronounceable acronym program.

0:00:53 BC: Oh yes, yeah.

0:00:54 PE: Where you got your masters degree and then the next two years were just full time work okay.

0:00:59 BC: For that microprocessor design group.

0:00:59 PE: Right, okay. Anything else you want to say about Bell before you move onto the PhD.

0:01:07 BC: Well, another thing that I noticed there, it may not be politic to mention this, but I learned something important from it. That was the first big company I had worked at and it sure felt to me, the mental image that I got from just working there for a couple of years was that there were some very good people at Bell but they were surrounded by many, many people who weren't at that high level. My mental image was like looking over the ocean and seeing islands of excellence here and there. You have got some leaders and you've got people to do the work it's all going to work. But those people that were on those islands, they were getting dragged into the muck by the people around them, they were getting held back, they weren't getting enabled, pushed forward, it was the other way around. I saw the bureaucracy working in such a way as to not entirely nullify but definitely degrade those peoples' ability to make a difference.

There are, of course some famous counterexamples to that: UNIX and C came from Bell Labs. The transistor came from Bell. Bell Labs had in the past done many marvelous things and even when I was there they were still doing some really cool stuff as a group, but just in terms of my local area I was dismayed at what it took to get anything novel done there. It just felt like they wanted to go down the center of the road and not get distracted one way or the other by much, so that was one learning I thought to myself I do not want to have that happen again, that is the one reason why I did startups after that for a while. Because at a startup the opposite happens and there is so much work to do, there is no room for bureaucracy. You just do your job and three other peoples' jobs too because otherwise there is no hope for your startup and I liked that.

For example, I really enjoyed being that at Multiflow (which we'll talk about later). I was the hardware documenter there, no else wanted to do it. My pattern tended to be that in the morning, I would think about architectural issues and make decisions that I only had to check with one or two people, and then I would write those decisions into a document, making them instantly the plan of record. Then in the afternoon I would actually design based on what I had decided to do in the morning and in the evening I would write tests to see if my design would work the way I said it would. This was a very effective and efficient loop: I had no communication overhead, because I was the same guy that did all three of those things. There were no gaps, and I just felt far more productive doing things that way than at a place like Bell or even occasionally at Intel where you have to convince a lot of people that your idea is the best one of the competing three hundred ideas out there and sometimes that was a pretty onerous burden.

The flip side of it is, is of course, that if you are in a startup and you have just a very small set of people making all the major decisions, if they mess one up then you're out there without a net so pick the right people, right? Let them go. The net isn't going to help you much if you happen to fall anyway. In a startup you don't get to do second tries very often. So you see a whole lot of responsibility in a hurry and you can do it or you don't. But back to Bell Labs: I just felt like there was too much inertia. I saw this one at that time when I was leaving Bell Labs, my boss asked if I was going to come back after getting my PhD and I said I do not think so. I explained to him that I was not happy with my performance review. I said, you rated me a superstar for the first years, but the last one you just gave me implied I was mediocre. I felt my last year was the most productive, so I do not understand why that happened. He said, alright, well to tell you the truth your officemate told us that you are going to leave before you told us and we knew that so when we gave out the raises, we gave them to people that we are going to still be here.

I said, well, excuse me, but what you just did was blackball me from your company, because if came back I'd be starting as a mediocre middle performer, and I'd have to dig my way out of a hole. I'm not going to do that for something that you did. He said, yeah I understand and what the Bell system designing microprocessors is kind of like the US Army going into the car business selling jeeps (laughs) and I said, no when you put it that way, really, I am not going to come back (laughs). Strange but I think he was right about that Jeep analogy by the way, it was pretty accurate. So there was another thing I learned was, you know, watch out for corporate politics.

0:06:01 PE: Interesting, okay.

0:06:02 BC: Man.

CMU PhD program — thesis on RISC vs. CISC

0:06:04 PE: Okay, alright so you went back for a PhD. When did you make that decision and why? Was it?

0:06:11 BC: Well, I thought that I was going to do a PhD all along, that was just, you know, just seemed like something I should do. I am unsure how generalizable that observation is for most people, but at least for me I thought I would have the stamina to get through the program. I knew that I could write and I was pretty confident that having a PhD would be better than not having it in general, so what's not to like. Then when I saw that episode where they were picking the instruction set, I thought there's got to be a better way to do that. My thinking back then was okay Bell I'm going to let you pay for my master's degree and then I am going to come back and work for you for a while sort of pay off the moral obligation. Then I asked them how long does it take for someone to come back from a masters and work here so that no one is mad at them when they quit. They said how about two years, okay cool, (laughs) so that is why, essentially that's why I picked two years and that is when I started coming back to Carnegie Mellon after that.

0:07:35 PE: Did you put elsewhere for a PhD.

0:07:39 BC: Yeah, I did, actually at MIT.

0:07:41 PE: Because at MIT right you mentioned that before.

0:07:43 BC: They said no. (laughs) And I said okay fine, we'll see.

0:07:48 PE: Carnegie Mellon is perfectly good.

0:07:49 BC: Yeah it is a good school. MIT wanted me last time, before and now they don't.

0:07:56 PE: So who was your advisor and tell me about showing up there.
0:08:00 BC: Doug Jensen was my advisor and when I first got there, I remember checking

around the faculty of both ECE and computer science. I was in ECE because I viewed myself then and I still do as an ECE.

0:08:14 PE: ECE stood for?

0:08:16 BC: Electrical and Computer Engineering, as compared to Computer Science for example. Carnegie Mellon is very famous and especially then in 1980 it was world class in computer science. ECE wasn’t number one or two like CMU has always have been in computer science, so it is like, and I could have chosen either one of those departments, I mean, I applied to ECE specifically, because I view myself as an engineer not a scientist. I still feel that way to this day, I like to make things. To know things is fun too don't get me wrong, but when it comes right down to it, if I have to choose I will chose to make something even if I do not completely understand what is behind it as opposed to understanding something that I cannot use, you know, both of the extremes but I know which one I want, so I picked ECE and then I found some people that were in both departments, like Dan Siewiorek.

0:09:11 PE: Spell Siewiorek

0:09:10 BC: Siewiorek, there is no hope of spelling that (laughs): , it is S I E W I OR E K, I think it's a Polish name and he ended up on my thesis committee eventually. Dan's a good guy but for whatever reason I think Doug might have had money and he was talking about being interested in designing things and Dan was at that time doing CAD.

0:09:36 PE: Assisted design,

0:09:38 BC: Design tools automation, computer automation. At the time Doug had some very ambitious plans, shall we say, this is 1980, at the sort of a ramp up to Star Wars and Reagan and money falling from the skies for anything military-related and Doug had good connections into that community so he could get lots of money. He worked hard to do that and he was constantly on the phone consulting with his sponsors and so on. If you had asked him in 1980, "what are you interested in?" his answer would go along these lines, he would say, "picture a swarm of nuclear rockets coming up over the horizon from the general vicinity of the Soviet Union and you have sort of ground-based interceptor missiles that each one of them can take out one incoming rocket, what is the method by which you assign one ground-based rocket to one incoming missile in such a way that no two ground based team up on the same guy and leave someone else to get through?"

His theory was that if we had perfect knowledge of these missiles coming into the horizon from the start, we could design those ground-based systems in such a way as to target them appropriately, but we do not, the Soviets will not tell us that, darn them, so instead what we really want to do, is we have to launch now, we cannot wait all day. We have to launch missiles without knowing where they're going yet and then have to figure it out on the way up. He would weave this story about how these interceptor missiles have to talk to each other, I am going to get this one, you go get that one, kind of like football players on a field. That scenario was used to justify doing a lot of related work that we all thought was interesting, so in my case it was computer architecture. In fact it got really specific in my thesis where I basically dissected the Intel 432 chip and if you want me to connect the dots between that and the missiles I do not know if I can (laughs) but that's where it started.

0:12:11 PE: Yeah, okay

0:12:12 BC: That got us some funding and then we spent it the way we thought we should.

0:12:16 PE: Okay.

0:12:17 BC: It was only later that we got involved the RISC/CISC debate, that was my influence, Doug wasn't into that at the beginning. He was into all kinds of other things, he actually started sponsoring things like Real Time prioritized scheduling, operating system work, he had interests all over the place, but always real time. Doug really wanted to talk about things that are happening now, not pre-computed or batch computing or databases.

0:12:49 PE: So he needed speed.

0:12:54 BC: Yeah, a certain kind of speed.

0:12:56 PE: Yeah, that feeds into the RISC/CISC question.

0:13:00 BC: I started seeing the RISC stuff getting published and I would come to school grumbling that they did some really neat work, but the RISC folks are claiming A when they really have only shown A prime and you know, and in the difference, there might be something to learn there. It seemed so unfair to me. Eventually Doug asked what would I do as a research project to straighten out the confusion. I said, I know exactly what I would do. The next thing I know, I am doing one of those things, Charlie Hitchcock's doing one, and Brinkley Sprunt is doing one. We actually went off and did the things I listed to him kind of on the spur of the moment, got some good publications out of them too and it was pretty neat work, but as I mentioned I was somewhat unprepared in a naïve kind of way for the vehemence that I was about to face, from some people, not everybody but some in that general field.

0:13:57 PE: No, you weren't the first people to do sort of experimental verification of these different tests, so the different modes that these things operated in and try to sort out which parts they were coming from, the instruction which parts they were coming from, implementation and.

0:14:18 BC: We probably weren't.

0:14:20 PE: In thinking more about the first paper, in the 1983 paper, you talk about some prior work that the RISC people must have done, but it does not seem like there as lot of other work like that out there?

0:14:35 BC: Yeah, well I'd have to go back and look at it to be sure and to sort of put myself back in that timeframe, I was

0:14:45 PE: I'll hand it to you.

0:14:47 BC: people like Cray and the people that did the CDC machines, they were pretty savvy about what the performance advantages of the various things that they were doing and I know that Lynn Conway was, for example.

0:15:12 PE: I have just tended about appearing through the risks of this fog

0:15:17 BC: I titled this by the way [laughter]. Dave Patterson said "Why don't you guys come on out here, we'll talk about these things" . I said "Fine" so Doug Jensen and I get on the airplane and fly all the way across the country. We went to Chez Panisse, this famous restaurant near Berkeley, but of course we were three hours ahead and so Dave Patterson thinks it's 9:00, but I think it's midnight. Doug was content to let Dave and me just debate. He didn't say much, he was just sitting there and Dave Patterson started the discussion saying "Sir, I read your fog paper," [chuckles] he said something like "I think you are in a fog." Well I said "Yeah, I think we are, but I think you are too, we're in the same fog buddy." So it was funny because we ended up in this impromptu debate, him versus me. And in the middle of it, I realized, I'm working really hard to hear what he is saying and give him a cogent answer back. But he does not appear to be working very hard. And I thought okay, either he's a lot smarter than me, which is possible, or there's something else happening and then it dawned on me what it was: This guy has debated this stuff with the smartest people in the industry. I have never talked about it with anybody but my adviser and now him. In effect, everything I said, he'd give me a numbered response back. I'd try to make a point and he'd mentally think "oh that deserves answer number 27"; he wasn't thinking, he seemed to be mostly just retrieving. I realized okay, what I need to do is nudge him off his game. When you play chess at a high level, you memorize a bunch of openings if you want to be really good. And then after that in the middle game, you're on your own. When you get to the end game, again you're back to the book because there are certain ways to solve it with just a queen or whatever.

And so I realized the game is if I'm going to debate, I'm going to go down in flames if I don't get this guy to start putting work into it the same way I'm doing. Eventually we got to where I think he was actually thinking in real-time and it got a lot more interesting and a lot more even. But that was my first experience debating this RISC stuff live and I took on the, you know, the biggest guy then. When we were on the plane on the way back, I was thinking, "What was I thinking?" I should have prepared for this. I mean, Dave Patterson's a formidable debater. Very smart man and you know, you don't go in unarmed to that competition. My paper stimulated that whole episode. At that pizza meeting where Doug asked what I thought we should do about it, I broke things down this way. We could take a really complicated instruction set and explore the heck out of it and show specifically what the various contributions are. And then compare them with what the RISC guys are predicting and then that will tell us something.

One of my hot buttons back then was the register set because a lot of things were being published from the RISC guys along the lines of RISC having register windows with which you do the procedure calls and you pass the parameters by putting them right in the. The callee inherits the registers for free. If you can put your parameters in these registers that overlap, you get the call overhead for free, and I thought it was not really free is it? You had to have a whole lot of registers to do that, which take a lot of space and can end up being slow. So if it cost you your clock rate to include those registers, then maybe the whole idea was a loss, or you know, if the size of the register file is too big, there are going to be some other techniques that you can't attempt because they're going to hold you back. Later on, Out of Order, for example, came along with Intel and a large register that would have killed us, it would have been really painful that. I was just thinking in terms of if you had to implement this thing, it's going to be hard.

So we said "Well heck, the RISC guys are being unclear about what the extent to which the claimed performance benefits actually come from the register set. Not from the reduced instruction set but from the size of the registers. Because after all, I could take them back and stick a lot of registers in there if I want and if I get the performance you're saying, then it's not the reduced instruction set, is it? It's just having lots of registers. So we simulated adding a bunch of registers to the VAX and sure enough, its performance went up same as the RISC guys said the RISC designs had. Which means the register windows were not a RISC effect, not an instruction set issue. The third leg of this proposed research was this military computer family effort, MCF. The military periodically tries to consolidate on one architecture. Because they know that it would be centralized, and they all agree to disagree again and never do achieve the consolidation. Occasionally they will agree on something like the Ada programming language and then realize they have all made the same mistake but in the case of the architecture, there was this thing called MCF. This was in the 1970's, and they were trying to figure out what is the one architecture that all the military branches can agree to use? And I said well, they put a lot of thought into what such an architecture ought to look like, what the characteristics would be. So let's apply a RISC analysis to that scenario and see what happens? We ended up doing all three of these proposed studies. I did the 432 dissection, I think Charlie did the register of VAX simulation, Chuck Kollar did the MCF analysis, and then we all wrote the paper.

0:21:00 PE: So is that, was the 432 part of that?

0:21:03 BC: Yeah.

0:21:06 PE: Okay. What is MCF?

0:21:06 BC: No, no, no. That was separate. It was part of the same research effort under Doug Jensen, but I was doing the leg that associated with the 432 and Charlie did the register set and I think Brinkley spearheaded some other piece. So, yeah. We ended up tackling them all but that was the paper that started it.

0:21:26 PE: Yeah. Well, I wonder, let's talk about your, that dissertation recording too a little

bit, but I want to show you something. I think you have seen this but, yeah. It's kind of, this is by a blog or quote from a guy named Brian Cantrell. This will interest you. Because you've seen this. This is a discussion of your 19, let's say 1988, 432 TOCS paper.
0:22:07 BC: Well he thinks it's brilliant, this guy must be smart. I like him already. I think I've seen this before but a long time ago.

0:22:17 PE: Yeah, in a second, I'll read some of it for the recording.

0:22:22 BC: Yeah. Every malfunction caught within context, which I think is incredibly promising. In fact I think he and I communicated at some point over the years. I seem to remember this last thing, because my intention with writing some of what I was doing then, was that we do a whole bunch of work, culminating in a product or a whole system, whatever it is, but then we move onto the next one project and we hardly ever capture what we just learned. This means that no one else benefits from our experience because we didn't write it down, right? One of my intentions with the 432 work was specifically to document all that because there were so many misconceptions about it. I mean it was appalling how wrongheaded some of the stuff that was being said about it was. The 432 deserved criticism, but it needed to be the right criticism.

0:23:22 PE: So this is a 2008 blog entry on revisiting the Intel 432 by Brian Cantrell and he is revisiting, he is reading Bob's paper. This is some of later paper of performance effects of architecture complexity in Intel 432 which he wrote with Edward Gehringer and Doug Jensen, which is quite long and must be based on your dissertation work. Right. And asks about, you know, what's the relevance of Colwell's paper now 20 years later. One of the principle problems that Colwell describes is a disconnect between innovation of the hardware and the software repair levels, this disconnect continues to be a theme and can be seen in current controversies in networking, in virtualization and most clearly in my opinion in hardware transactional memory and he goes through bits of the paper and talks about all the important things that you set down there are still true today. So, you want to say a little bit about the

0:24:28 BC: Yeah. I can give you a crystal clear example of this. Sure. That the summary of 1984, we were still living in Pittsburgh but I came up for the Summer to work at Oregon Jones Farm because I had access with the people that designed the 432. They were showing me their simulators and everything. It was really important stuff to write a PhD thesis, and I needed that kind of insight or info.

0:24:48 PE: How did you get that access and

0:24:50 BC: A guy named George Cox was on my thesis committee. George is still at Intel and he was travelling to CMU all the time because he was working on a project called iWarp, which was a chip that CMU and Intel were doing together. George was a good Intel soldier and he was always looking for grad students he could eventually recruit for Intel. So he started recruiting me early on and when we mutually realized that there was a way that it would benefit me tremendously to go out to Intel, he arranged it and he became my boss by the Summer. So yeah, that was George's doing.

0:25:22 PE: At that point of 432 is basically dead.

0:25:25 BC: It pretty much was.

0:25:25 PE: A product and sounds like it was kind of like a whipping boy for a lots of other, lots of papers about things that could go wrong in chip design but

0:25:34 BC: And if you're trying to use it, you could see, it was painful. [laughter]. Man, it was horribly slow. I started analyzing it and started asking questions and we started looking at the code which was ADA source code compiled into 432 assembly language. What happens at the assembly language level is what the machine is really doing. And so I started looking at that and thinking "Holy cow, this is terrible". If you could do a procedure call, a RISC machine will do it in a couple of clock cycles, and if you have to save the register set okay, it will take 30 more clock cycles. But the 432 was using a 400 or 500 clock cycles, a number so big that it was like a flashing neon sign saying "Researchers, come look here, something is wrong." I started looking at that. Now, it's not like I spend a lot of time in bars, even though when I was in a band, I did. But one particular night that summer, I was with some friends, and we went up to this bar where I struck up a conversation with some random guy I didn't know sitting next to me. He asked me what I was doing at Intel for the summer. I said "I'm a grad student doing some research on the 432." "Really? What kinds of things are you finding?" I said "Well, I'm finding that there's a huge disconnect between the software and the hardware. I mean the hardware should be capable of doing a procedure call much quicker than it's actually doing, for instance, it's taking many hundreds of clock cycles. And in other places, I'm seeing all kind of wastage from the compiler just setting variables that never get used, classic stuff you just don't do." And the guy said "Really? I mean tell me more about that part."

Eventually, I said "Okay, who are you? You know way too much about this technology." He tells me "I'm the leader of the compiler team." And I said "In that case I probably just fatally offended you." He said "No, not at all because I know we generate bad code and I don't care." He said "We don't like the 432 hardware team." And I thought "Oh my God, there is no hope that this project is going to work when you have the two main casts killing each other." He said "That hardware team never listened to us compiler folks At some point we decided that we'd live up to the letter of the contract but beyond that? No."

That alone might not have been enough to sink that project. But there was another aspect of it that combined with the warring camps: what I call the religious attitude. By religious attitude, I mean the following. The guys who did the 432 started with a really brilliant idea, that you can use the notions of encapsulation and object orientation in a holistic way across system boundaries where it had never before been tried. They didn't invent the ideas, but man, they went whole hog with them. Everything and I mean everything in a 432 computer system is an object. Objects have properties, they have rights access, they have certain fixed aspects to them that you have to obey. And if you obey them, everything is great; if you don't, the system will stop you in your tracks and say "You tried to access something you have no rights to" right *here*, at this line of code. It's like if you're trying to debug code, it drags you right to the line that's offending. In terms of security, we could use that kind of attitude today, with all the viruses and worms we contend with.

Unfortunately, in our zeal to get away from the 432's complexity, and because the field as a whole did not retain the right lessons from the 432, it is constantly being reinvented nowadays I'm on program review committees for conferences, and there are papers that are submitted that are proposing ideas to improve security with the same ideas, just different With the 432, even the hardware, the actual chip, was an object. To the software, it's an object with certain capabilities. The 432 developers approached everything that way and yielded a certain beauty in that once you have really adopted that point of view, the overall system makes sense in a way that no other computer ever did. There are also many problems with this whole scheme. One is the overhead involved in looking at the world that way. I don't think it's unmanageable overhead but if you don't pay attention to it, it will *become* unmanageable overhead, and so part of the problem with the 432 was these guys just weren't paying attention to performance.

They thought in some deep sense that they were on a mission from God. They had this great epiphany and they were going to follow it to the end and surely it wouldn't screw them up. Well somewhere along the way, they threw away a little bit too much performance, and if they had been paying attention to that aspect of it, I think that they could have fixed those pieces, and if they made peace with the compiler guys they could have fixed that part of it too. They could have actually got within shouting distance of a competitive part. But, as it was, there were so many mistakes that there was no way to recover from them all.

0:30:41 PE: Yeah, yeah, yeah I know.

0:30:44 BC: Gordon Bell came to Carnegie Mellon and he gave a talk about his personal conversion to RISC, because he had switched camps at that point. Prior that he designed the VAX which is a canonical CISC. In that talk, he told the audience, "of course the 432's a piece of junk: it's got seven levels of indirection in the addressing path". I was thinking well, yeah it does, but in terms of what's wrong with it, that's not even in the top 10 list. Afterwards I went up to him after the Q&A session was over, and I said "Gordon, you blamed the seven layers of addressing for something and" I said you know, "I've measured this machine extensively and they've got a cache that shorts out all seven layers almost all the time. It makes no difference, you could make it 20 layers and it won't make any difference" and he said, "no, no, no anyone can see just by looking at it", I said "you can't tell anything just by looking at it, that's the point of the RISC research, you have to measure it, and I've measured it and my conclusion is not the same as yours." He said "hey the audience got the point that's all that matters" and I thought wait a second, you can't tell a lie to an audience and walk away from it. He clearly did not view things the way I did so that was disturbing.

There were a lot of people who drew lessons about the 432s who were just crazy. Their lessons weren't based on anything but casual observations, on the level of a block diagram and, yeah there were things wrong but you couldn’t see them from there. I pointed out that if you're going to design a competitive machine and you're not going to have registers, which the 432 did not, you'd better have ready access to certain things like constants, you're going to need 0, you're going to need 1, you're going to need 4 (because you're going to be scaling pointers so you need to multiply by 4). Well the 432 guys gave you 0 and 1, no 4. So if you needed any other constant but 0 and 1 you had to open the constants object. Opening a constants object required the operating system to get involved, and that's why every procedure call was so nightmarishly slow.

0:32:41 PE: That's interesting. So there was a level below which they didn't do the object orientation?

0:32:47 BC: Where?

0:32:48 PE: 0s and 1s.

0:32:49 BC: Those numbers were embedded in the instruction stream, as an immediate constant, as part of the instruction. The compiler viewed it as, there is an instruction that sets your target to a 1 or a 0. Now had they said, "I've an instruction that sets a variable to any random thing" then you're back to the constant stuff. the 432 designers were being consistent with their object orientation philosophy, but there had to be a better mechanism for low-level considerations such as constants, you just can't throw this much performance off the boat and expect to survive. If they had somebody whose job it was to notice that the boat was sinking that would have helped but they didn't, they didn't think of doing that.

0:33:32 PE: Well I just want to go to the end of this paper and read a couple of lines here which is this is performance effects of architectural complexity in the Intel 432. When you say that this paper has shown that the 432 was at some 25 to 35 per cent of its potential throughput due to the poor quality of code emitted by its ADA compiler another 5 to 10 per cent is lost to implementation inefficiencies such as the 432’s lack of instruction stream literals and its instruction stream bit alignment [laugher].

0:34:04 BC: I forgot about that one.

0:34:04 PE: We found, and then skipping a few lines, we found that a combination of plausible modifications to the 432 such as wider buses and provision for local data registers increased performance by another 35 to 45 per cent. This is saying you think he could have improved by about 90 per cent.

0:34:25 BC: Yeah. 10X. And that's why I said you could get it within shouting distance, because I think somewhere else the RISC folks measured it as being about 1/20th the speed of a RISC chip and so if you give it a factor 10 you could get within a factor of 2 of the best and then at that point you could start saying well wait, I'm not as fast as him but look at all this other cool stuff you get. But if you're 20 times slower there's no cool stuff going to save you from there. It's kind of a shame, I mean I think that whole aspect of computer architecture died because the 432 was such a debacle and it didn't need to be, there were some good ideas in there, they just got lost.

0:34:57 PE: Well tell me a little bit more about your time at Intel, you know, when you were working on, when you were, first, you know, you talked about this conversation with the compiler guy. What else happened while you were out there, what sort of, what were you actually doing at Intel? Just finding out about how the chip worked?

0:35:16 BC: Yeah and running their simulator and then asking a guy named Konrad Lai to explain what it meant, because simulators...

0:35:21 PE: Konrad?

0:35:24 BC: Lai, L A I. And he's still there too, actually. But Konrad had written some of the 432 microcode and after I saw what he had done with that microcode, I was astounded, it's got to be some of the most clever microcoding I've ever seen in my life. He was doing recursive calls inside that microcode, and he's got no registers to do it with. If he had been any less smart than he was or if he had been on a team of other microcoders I think he would have failed; I think he could only get away with the tricks he did because he was the only guy he needed to be consistent with. He knew exactly what he was doing because he's brilliant. But anyway, there was a lot of cleverness buried inside of a fairly large mistake. Konrad was helping me understand what the simulator was telling me as it as running, and that was extremely useful.

There was a team, the team of Intel guys around me were working on what eventually became known as the i960. They were following up the 432 with a capability based attitude towards what became the i960 and they were working with a company called Siemens. It was supposed to be this big secret collaboration, so they were really worried about anyone finding out what they were doing. Was it called Orion, no, it had a code name and whatever the code name was, I remember walking past Kevin Kahn's office, and asking him "hey Kevin what is 'Orion' and he said "you can't know about that!" I said it's written in block letters above you on that sign above your office. [laughter]. I did get to see what this design team was doing to a first order, I could sort of see how they were interacting, and we were sharing a mini computer so I just became one more terminal attached to this minicomputer and any time I used too many cycles, someone would come in and yell at me "I don't know who you are but you're using up fifteen per cent of our computer and you're not allowed to do that", which is kind of funny because I remember the names of the people who came in and grouched at me and about, let's see that would have been 1984, ten years later I was in charge, I became their boss [laughter]. It's like be careful who you piss off [laughter]. I didn't take any retribution but I was thinking you know...

0:37:55 PE: So a question I haven't asked you yet, did you start using PCs by then? Were you?

0:38:02 BC: Yeah sort of, I got a PC for home and it started being used in this very room about 1992.

0:38:12 PE: Really. That's the first?

0:38:13 BC: That's my first, prior to that it was all workstations or mini computers. And it was all UNIX. And even then, at Intel we were using a version of UNIX called AIX that ran on IBM R6000s, so it was all workstations and UNIX and I knew very little about Windows until I started studying it because I knew we were going to have to run it.

0:38:33 PE: Did you have a workstation at home?

0:38:35 BC: No, at home from I'd say till about 1997 or so, Intel supplied us with Smart Terminals and an ISDN terminal. Yeah, it was a little bit better than dial up.

0:38:52 PE: About 128K I think.

0:38:54 BC: Exactly, yeah, in fact I guess it was a lot better than dial up at 14.4K but it wasn't nearly as good as we get nowadays. That was our access to our computing resources. I think I switched to PCs when laptops came out and became useful. They had them earlier but they were pretty slow. I got my first laptop probably around 1995, 96. Carried it on the road, got used to it, and then after I got used to Windows in general, so I just switched over to it.

0:39:23 PE: Okay. Alright back to 1984, you were at Intel. I guess about that time?

0:39:33 BC: 84 yeah, well I mean my wife was at the other end of the country.

0:39:36 PE: Okay that's one of those summers that you were separated? Did you have kids then?

0:39:43 BC: No but my eldest daughter was born in 1985. So we did not have kids at the point, but Ellen was working full time in Pittsburgh.

0:39:50 PE: What was she doing?

0:39:51 BC: She was working at Three Rivers Computer. It was a work station start up, a small company. I worked there too, part time, for a couple of years when I was at Carnegie Mellon on my PhD. Because at first, after the first year of PhD work, I had started in September, and when April or May rolled around and I'd done all the course work, I got the urge to design something. I'd heard that Three Rivers Computer were designing cool stuff, and saw a demo that they did that was eye popping for the time where they had a black and white screen with the equivalent of an 8 1/2 by 11 sheet of paper in its entirety. It was a black and white bitmap display so you could, not just a terminal, and you could do graphics with it. I remember seeing a demo that these guys had done that had windows sliding underneath each other and was intrigued and so, around that time I said, what, I'm going to take this summer off from school and I'm going to go and work for you guys, what would you like me to design? And by the way how did you do that demo, I want to see what's behind that?

They said well "oh, well, on that demo, we cheated". The demo was just bitmaps blasted to the screen one after another, in such a way that it looked as though they were sliding past one another. But we're going to make it work for real after the end of the summer", which they did. Beware demos! They showed me a list of things wanted to have done and a color graphics version of that same engine was on the list. I had always wanted to learn about graphics, so I dove in and started working on it and I designed them something over the next few years, got it working, and demonstrated it, just in time for the company to die.[Chuckles]. That's how it goes with startups.

Anyway, that's what Ellen was doing, she was working in, the compiler team for those guys at the time. I basically just collected as much 432 data as I could and at the end of the summer I knew I now had enough data to start writing a thesis and that was my aim at the beginning and so I was very happy with that. It's one of those things that when you're in a PhD program, and I don't know if it's happened to you, but it's like when you start out you're thinking, I hope that I find a topic that I can do and that is acceptable to the committee. I hope there is such a thing, because otherwise it’s going to feel really bad if there is not, or if I can’t find it. And then you start, you get the thesis topic and then you feel pretty good, but you're not sure if you can really do it, but there comes a day in the thesis course of action when you realize, I have everything I need, I'm going to get there, and I can see the path. The remaining ambiguity is no longer large enough to worry about and I just remember thinking that's the coolest feeling. It's almost as good as getting the degree, it's like a load’s been lifted off your shoulders . I remember that happening right about the end of the summer of 84 when I said I got the ammunition I require, I'm almost done here. Kelly was born May of 85.
0:43:01 PE: That must have been around the time you graduated.

0:43:04 BC: I remember more than once taking her to school and putting her sleeping little form in a box under my desk as I was typing my thesis, because Ellen was working. But I'm glad I don't have any pictures of that because it would have looked really bad parentally. There was an episode when I learned more about university politics than I ever wanted to know. Right about this time, at Carnegie Mellon, it turned out Doug Jenson was not sending enough of his research money to ECE (remember I said he had a joint appointment with ECE and CS) and the department head for ECE got really angry with him.

This dept head and Doug also seemed to disagree about whether Doug was taking seriously this ECE tenure effort -- Doug was supposed to be filling stuff out, and going through the motions for getting tenure. There was obvious tension there. And, there came a classic day when I came into work at the school and my office door key didn't work anymore, I couldn't get into my office, but I didn't know why. My office mate came along and opened it, but when try to log into the computer, I can’t, my account's been shut off. Okay, the door was one thing, but these two together can't be a coincidence.

So I went downstairs to my friends that ran the computer, you know, and I said "why is my computer account cut off?" and he said he was told to do that, and I said "well, if somebody tells you to turn it back on would you do it?" and he says "Yeah", I said "Turn it back on" [laughter], so he did. I asked who had told him to turn it off? He said, "the department head". So I went to see the Department Head. who informs me that he has had it with my advisor's lack of funding for me, and is consequently pulling all departmental support for me I said "hey, if you have a problem with a faculty member, why are you taking it out on a grad student, I'm within a month of graduating and I have nothing to do with the problem you're point to." I got no sympathy, and walked out of his office in a daze.

0:45:20 PE: Geez, wow, that's.

0:45:22 BC: So I went stumbling out of his office thinking now what do I do. I don't have the ability to fix this and I don't like being used as a baseball bat on somebody else.

0:45:29 PE: Yeah.

0:45:29 BC: But I had a brainstorm, I went to Dan Siewiorek. Dan is a guy who's been around CMU, he knows everybody, and he's got more clout than the next 50 professors. And I told him this sad story. He muttered "this is ridiculous", got on the phone and ten minutes later it's all put back together. He basically overruled the department head and just fixed it all. But boy I learned a lesson on that one.

0:45:49 PE: Right.

0:45:49 BC:I had no idea politics could get that bad especially with something I had no control over whatsoever and which really was not my fault.

0:45:56 PE: the saying about academic politics it's so vicious because the stakes are so low. [laughter].

0:45:59 BC: That could be the truth there yep.
0:46:10 PE: Anyway so you got to graduate.

0:46:14 BC: I did get to graduate, yeah.

0:46:16 PE: What happened to Doug?

0:46:18 BC: He stuck around a few more years and then I think he went to Honeywell which because he'd been there before. And actually if you Google for him you'll still find him doing real time research that's quasi military in nature. Every once in a while, he will ping me on something. I think it was a year ago, and he said, this is like some anniversary of one of our RISC papers; how about we write an update to it? I said, "how about we don't, let that sleeping dog lie man, I don't want to touch it." And I told Charlie Hitchcock that too and Charlie said, "oh good call, because I wasn't going to do it either", too many scars.

0:46:57 PE: Let's see okay, so anything else you want to say about your time at Carnegie Mellon? People, colleagues, other students that you worked with and maybe worked with later?

0:47:12 BC: I learned some very useful things at Carnegie Mellon. The one I'm thinking of now is being aware of what people are good at. People that don't necessarily make direct connections to you. For instance, I've written lots and lots of code in my life and I've also worked near people that were just geniuses at writing code, so I could tell that I'm not. I can see the difference; I wish I could do what they do but I can't. So when I write code it works when I'm done, and that's the only guarantee I'll ever make, it will work, it might not be pretty and certainly will be operable and it will work. And you look at their code and it's elegant, it's beautiful, even though it may have been done in a hurry. I wish I could do that but I wish I could play violin like Joshua Bell too, it just isn't so.

But the point is when I was a grad student I started noticing that guy over there is really good at networking issues and this guy over here can write utilities like a madman and maybe this guy is great at electronics. I typically didn't need help in that area but the point is everyone had their specialty. I started thinking about these people as a network of resources rather than just fellow students trying to graduate. I realized that many of them just because they're nice people will help you if you ask. But almost all of them will help you if you find some reason to motivate them to do it. I don't mean this necessarily as bribes but that's why I came up with the chocolate chip cookie scenario. What that is, is there are people that you realize maybe have done something really good for you. For instance, perhaps I would ask somebody to do what I think is a small favor, and he'll hand me back three hundred lines of code, having stayed up all night and to solve my problem. I didn't mean to ask you to do that much but, oh my gosh, now that you did, I must find a way to reward them by giving them a plate of chocolate chip cookies.

The idea is to give them something that they appreciate generally, that it surprises them so it has novelty value and they will remember it. If you come back and ask for help again later, they're going to realize that was that guy, he appreciates it when I do something for him. It's a reinforcing loop that works really well. After I caught on to this, that there's many ways to motivate people,, I started doing it on purpose. Even in cases where I think I can do this myself, if it's going to take me four days but if I get help from him it's going to take an hour, well, that's a good trade off. He's probably a busy guy but that's still a pretty good trade off, and maybe I can help him later. So I would purposely do that. Grad school's one thing because if you need more time you take more time. But when I got to Intel the schedules were tight, and asking for help (in the right way) turned out to be an extremely useful paradigm and I used it all the time.

0:49:48 PE: Okay.

0:49:48 BC: For people in different organizations it's good to be nice to them and appreciative. When you do something unusual it sets you apart because everyone else forgets to do that. If you do it makes you stand out in their minds and it means you can do it again the next time. The reverse of it also applies. There was a computer science professor there who shall go unnamed, who bought up the rights to a particular piece of software that we were all using. It had been written by computer science people, I don't want to give too many clues because someone out there will figure out who this is. Anyway prior to his having done that we had a user community online and if you need to learn something you just posted a question, and would get ten answers back. And most of them will be right because these guys know what they're doing. It's not like the internet where most of them wrong but may be one of them is right.

After this guy took over ownership of the software he posted, a friend of mine posted an answer to a question and this new owner put up a blistering attack on the guy, about how you shouldn't post information if you don't know what you're doing. A very personal attack in a public place. And we were really, really unhappy that he had done that to our friend and it seemed so unreasonable. One of us and I swear it wasn't me, decided that the proper remedy for this was to find a bunch of dead insects from the corridors and mail it to the guy. Why that somehow remedied this issue I've no idea but I know that he did this and somehow we all felt better after that like we had made our point. I'm not sure what point we just made but the idea that we were very unhappy with that previous transaction should have registered with the miscreant involved. The guy probably had a legal right to do what he was doing by saying, "look I now own this and I don't like the way you're supporting my product and I want you to shut up". But it was a completely ineffective way to go about ...

0:51:44 PE: Yeah.

0:51:44 BC: ...remedying this. Like you saw when like Sony had this robotic dog called, was it Aibo?

0:51:53 PE: Yeah, yeah, yeah.

0:51:53 BC: Or something like that and people started hacking that dog’s programming, to make it do crazy stuff. And at first Sony was going to go off and sue these people because they were not supposed to open up this dog. And then later I think someone there had a realization that if you just let them do this, this is a good thing, they're actually helping us, don't be afraid.

0:52:10 PE: It'll be on TV every night or on YouTube.

0:52:12 BC: And we're getting this for free you know, so what's to lose. That was good and the attitude there is to be careful about collateral damage when you're like asserting what you think are your legal rights, sometimes it's not the right thing to do even if you are formally entitled. That played out again when we put the microcode patch facility in the Intel chips, and we put in some other things associated with debug. We had this discussion about to what extent do we patent those things. And if we do patent it, then to what extent do we prosecute anything that looks like others are doing what we did. My recommendation was, this is kind of like seat belts: when Volvo invented seat belts they didn't go around to all the other car makers and say for an exorbitant fee I'll let you put them in your car too. They said, we have seat belts and if you don't people are going to die unnecessarily. That's an issue that just supersedes any monetary considerations and so my understanding is that Volvo basically gave the right to that out to anyone who wanted it.

0:53:11 PE: I never heard that story.

0:53:12 BC: Yeah.

0:53:12 PE: It's interesting.

0:53:13 BC: There was another case when we were at Multiflow. We were using a simulator called Endot, it was a language....

0:54:08 PE: ENDOT?

0:54:09 BC: I think it was ENDOT. I can't remember the details about why they came up with that name but it was a language in which you could specify the design of digital system. Kind of like Verilog but much earlier on. They were trying to achieve the same goal with a different language. We licensed their software and we had written a simulator for our next machine and we were using the Endot simulator. But every time you started the simulator it would take an hour of initialization before the simulator would start giving you results you actually care about. And that would just kill you; I mean how many lunches can you eat in a day?. You can forgive the first time because you can go to lunch but after that you just pull out your hair. Well a really brilliant guy named Dave Papworth, who is still at Intel, was on our Multiflow team and Dave decided not to just put up with this, and tackled the problem. What he discovered was, at the beginning of the simulation Endot had to initialize every variable, meaning every piece of state in the entire design.

They had some kind of list of such variables that had been constructed by walking over the entire design in some page by page manner. In other words it wasn't sorted in any way. When they were setting these variables to zero for initialization they were doing it alphabetically. So they would walk down the entire list, find the next variable, set it to zero, go back to the beginning of the list, so it was a ridiculous algorithm. Dave basically patched something into their binary that said jump to my binary instead and then go down the alphabetical list of variables, set each to zero and be done and pretend that you did your algorithm. It worked like a champ it took maybe a minute and a half. We thought this is great, these Endot people are going to love us. No, they threatened to sue us. They said, "don't you see right here you said you won't reverse engineer this." We said, "sorry we mentioned it. Did we say we fixed it, oh no, no we didn't do that". It was ridiculous, you just can't you know, you can't give things to people for free sometimes.

0:56:10 PE: Yeah.

0:56:11 BC: Anyway that's the first time I'd seen that. That kind of overzealous attitude can hurt you pretty badly.

0:56:17 PE: Alright, so we're at Carnegie Mellon, anything else on that?

0:56:30 BC: Yeah, we're doing the PhD right now. My masters project was a digital synthesizer, digital music synthesizer.

0:56:38 PE: Okay.

0:56:38 BC: And that was a pretty neat machine I still have it in my garage.

0:56:42 PE: Do you?

0:56:42 BC: I do, it's a 19 inch instrumental rack with these huge circuit cards and everything. Don't ask me why I still have it. [Chuckle]

0:56:48 PE: That's good.

0:56:50 BC: It's just sitting there.

0:56:52 PE: UM has one too.

0:56:53 BC: It ought to be in a museum, a synthesizer museum but I have yet to find one close enough because it weighs a ton. This thing's really big and heavy. Two of us designed it for our masters' project at Carnegie Mellon. And I learned a lot doing that project. We did the whole machine completely on the cheap. We couldn't even afford to fabricate PC boards especially ones that are literally two feet on a side. So we drilled every one of the holes with a drill.

0:57:24 PE: Wow.

0:57:24 BC: We made the edge connectors by putting masking tape on the copper and etching it away. And we couldn't afford a backplane so we wired the backplane with just wires and remembering my experience with Bell labs, it didn't work. I remember it was Christmas Eve and we were trying to get this thing working and we think well okay, I'm sure there's some bugs but we weren't able to get the cards to communicate the way we were expecting. After staring at this thing after we've been debugging all day and at some point I just put my hand around the wires and it started working. [Laughter] And I said "I kind of suspected that might happen". What we had to do was because we had no ground plane it ended up we couldn't just use wires in space; we had to use twisted pairs for the ground. And if we did that the signals were immune from each other's cross talking and other noise and that was all it took to make it work. It was a good Christmas Eve after that. Designing that machine was kind of cool. We thought we were going to be one of the first people on Earth to create a digital music synthesizer and I think it turned out we were the second. A guy at Bell Labs was working on a very similar design.

0:58:44 PE: Digital synthesizer?

0:58:45 BC: A digital synthesizer yeah, a music synthesizer.

0:58:47 PE: Yeah.

0:58:48 BC: And he beat us by a few months but his budget was enormously higher than ours was. So I didn't feel too bad.

0:58:54 PE: And did it have a keyboard?

0:58:55 BC: No you had to program it. I wrote a bunch of codes so that you could enter a file to drive it. The first piece I taught it to play was a symphony by C P E Bach. I've no idea why I chose that piece. The way set things up was that I'd type in like the next three measures worth of score into a file. I'd feed that file into the synthesizer and it would play it. Now the synthesizer had 32 channels of any wave form you wanted basically and that waveform had an real-time envelope computed by the synthesizer. Do about envelopes?
0:59:25 PE: A little bit.

0:59:26 BC: The canonical thing a synthesizer does is attack/decay/sustain/release. You initiate a note, then its amplitude climbs up to some sustaining level and then you release the key and the amplitude drops down. We wanted to do what we called N-segment envelopes. Not just four segments but any number of segments you want. And we did all of that with a bit-slice processor, and just as we needed it a guy named John Snow at MIT I think it was published a thing he called a lexicon of analyzed tones. And he would show you what all 32 partials of a violin needs to be to make it sound like a violin. We just took it straight off his pages and said okay, there's the segments here's what it should look like, and sure enough it sounded like a violin. And the fun part was noticing that a violin has a certain range but if you emulated a violin with this synthesizer, you could play above or below that range to see what would it sound like It was a blast. It was a fun project, we got the thing working in about July of 1978 then, we did the demo for our masters demo in August and then I got married a week later.

1:00:34 PE: That sounds like a nice time.

1:00:35 BC: It was a blast. It was one of those things in 1978, I had never heard digital music before and it was shocking. We're all used to digital sound now; CDs for example. But prior to that if you heard music recorded it was an LP or it was a cassette, and there was always background noise. So you start playing an LP and you hear this little crackle in the background, and then the music starts and subliminally you get used to expecting that. Tapes are the same way. They were just hiss and noise and you could hear that. But with CDs or with digital it's like nothing, nothing, nothing bam music and it was shocking when you first heard it. I was just so struck by that once in the lab when we got this thing working and I thought this is the way that music is going to be in the future. I wasn't anticipating MP3s, maybe I should have. But yeah, you could do a lot with it, it was fun. But so far no museum so it's still in my garage.

1:01:28 PE: So shall we talk about Multiflow?

1:01:34 BC: Okay.

1:01:49 BC: So I mentioned this guy named Paul Rodman. So in 78 he went off to...

1:01:55 PE: RODMAN?

1:01:57 BC: Yeah. In 78 and he is at Google now. But he was at Silicon Graphics for a long time designing chips, designing lots of stuff, Paul's a real good designer. So in 1978 I went off to Bell Labs and Paul went off to Prime Computer and worked with Dave Papworth whom I already mentioned.

1:02:15 PE: Right.

1:02:15 BC: At Prime.

1:02:16 PE: Who then worked with you at Intel?

1:02:17 BC: Yes, but the common thread is in 1984 Paul and Dave went to Connecticut and did this Multiflow start up. I didn't show up until 85 and I showed up because Paul called me and he said, "We're doing a really cool start up and you should come and help us". I had listened to Josh Fisher's pitch on this I don't know if you saw this. (Noise). So here's, this is Josh's reminiscing, I wrote, there's an article in there I wrote about my reminiscences of Multiflow. You can keep that if you like, I've got copies, this is a copy of my IEEE circuits and magazines spring 2009. It's got a cover story on the VLIW architecture of Joseph A. Fisher. Part I.

1:03:06 PE: It sounds like there could be more so.

1:03:07 BC: Yeah, yeah, I heard there's going to be a second one. So Paul somehow got involved in that and I don't remember how he got connected to Josh. But Josh had come to Carnegie Mellon in 1982 and made the rounds of grad students and I was one of them. I spent an hour with him.

1:03:21 PE: Was he already thinking about this whole construction work?

1:03:23 BC: He was in fact I talk about that in this article and I said, he was talking, from my point of view, Josh was going bananas. He was saying (laughter) we're going to reorganize the code. I mean when you first think about where the parallelism in code is you're going to reach the same point where the guys in about 1972 did. There's a famous paper by to Tjaden and Flynn, I don't know how to pronounce the first name, T J A D E N. And those guys had asked the question "how much parallelism is inherent in a sequential stream of code for a uni-processor?" What they were suggesting was there's an instruction and it will produce a result instruction. That instruction plus 1 probably needs that result and he's going to use it to produce another result and so on. And there'll be a chain down through the instructions but it does it in every instruction. Some instructions are independent of other ones. And if that happens, you can run it at the same time in principle if you have enough hardware to do it. And so we...

1:04:24 PE: Then you can sort out which ones are which?

1:04:25 BC: Yeah, yeah. So we're just going to give you that for free.

1:04:28 PE: Right.

1:04:28 BC: And the only question they were asking was, to what extent do true data dependencies limit the parallelism in the code? Until Josh came along, the answer was it limits you pretty badly. Because after all a basic block of code is like 5 instructions, 6 maybe and then there's a conditional branch. And that conditional branch is a problem because if you're just doing a static look at the code you don't know which way it's going to go; it could go over here or it could go straight through. The job of a branch predictor is to help you decide it's much more likely to go over here than it is to do this like in a loop.

1:05:03 PE: Yeah.

1:05:04 BC: Well that conditional branch at the end of the loop is trying to send you up here and the loop's probably going to be taken a lot of the time.

1:05:08 PE: Yeah.

1:05:08 BC: Maybe you go up a 100 times or 99 times and fall through once kind of thing.

1:05:13 PE: Yeah.

1:05:13 BC: Well in that case predict that it goes up and it will do pretty well.

1:05:16 PE: Yeah, yeah.

1:05:17 BC: But the common wisdom was okay, these branches are a problem we only have 5 instructions to work with. There's not much parallelism in there, let's not spend too much this is a wasted effort. Well Josh came along and he said, "well what if we get clever about those predicted conditional branches by ignoring them. If we just ignore them what would happen?" Now he didn't cast it this way, this is my spin on it. You might come down this sequence of instructions fall through the branch because you don't know what it is yet, execute a bunch more stuff at some point you'll figure that you should have taken the branch and you didn't. Oh that's a bummer. But what if when you realize you've done that you insert special code to undo the bad effects of the instructions it shouldn't have executed and did. And that special code will undo all that and then will go ahead and do what the branch should have done.

So this is Josh's pitch. I said "Josh that's ridiculous, I mean first of all how do that you can always undo the effects? And second it's not just undoing the effects, if you predict the wrong way there are things you should have done that you didn't. Now you can inject code and do them but this is, the bookkeeping alone would be a nightmare. You may not get any performance if you go down this path". And he said, " I've reason to think I would though". He said, "I think I can get a lot of parallelism this way". With Multiflow's products, the widest Trace was 28 units wide, and the minimum version was 7 units wide.

1:06:50 PE: A unit is just ...

1:06:51 BC: Like an add, it would be one unit floating point operator, a branch, a move.

1:06:56 PE: Okay.

1:06:57 BC: Load. So he said "suddenly I have a source of parallelism that could keep that machine really busy. And granted some fraction of the work I'm doing is fixing up this goofy stuff that I cheated on." But he said, "My experiments show there is surprisingly low overhead". And I went okay well alright, well good luck on that Josh. That was in 1982. So three years later, Rodman gives me a call and he says, you're graduating, so come on up and join us, we're going to build Fisher's crazy machine. [Laughter] I went, oh Geez, do I really want to bet my career that there's something here. But I at least wanted to check it out because I knew Paul was a smart guy and if he said there's something there then there's something there.

And so I went out there but not only him but Dave Papworth was there. He's a really smart guy and I didn't know him at the time. But after I talked to him for a while I thought okay this guy knows what he's doing. And he's got reason to think it's going to work. Plus they were fixing some of the fundamental weirdness's that Josh's original paper had, where all the functional units were arranged in a circle and were cross connected each to all the others. [Laughter]That makes a pretty picture but if you actually had to build it think how many wires that is. There's a picture of a rubber mallet in my article that we used to call "the persuader". You'd need the mallet because the amount of force needed to seat these big circuit cards into the backplane was ridiculous was and otherwise you'd kill your thumbs trying to do it.

1:08:25 PE: Oh yeah, yeah.

1:08:25 BC: We had a rubber mallet and we'd just get it close and then whack and it would go right in. So Josh's thing would have even more edge connectors, even more insertion pressure, so oh man it would have been a nightmare. As it was we were using jumpers across the other edge because we didn't have enough connections. But anyway that's how I got involved with Multiflow because Paul went there first and he said come on up.

1:08:47 PE: Oh what did Doug think about it, your advisor. Doug Jensen is it?

1:08:51 BC: Doug Jensen yeah, I don't have any idea what he thought of it.

1:08:54 PE: You didn't meet...

1:08:55 BC: I don't think we ever talked about. I'm pretty sure he didn't talk to Josh when he was on campus and yeah I don't think we, in fact to this day I don't recall talking to him about it. We talked about the Intel stuff to some extent but I don't remember ever doing the Multiflow stuff. Good question. [Laughing]. Well I wish I knew the answer.

1:09:18 PE: Okay. So he called you up and you went up and checked it out, it seemed like they were there and you went.

1:09:29 BC: Yeah. Good move too. The company died, I mean we were in business for about five years. But we shipped products, we shipped about 110 systems.

1:09:39 PE: Yeah.

1:09:39 BC: And I learned a heck of a lot. Because in a start up, if the work gets done it's because you did it. (Chuckle) That's one thing that did disturb me sometimes, when I was at Multiflow debugging which I spent a lot of time in the lab with the scope probe in my hand trying to figure why things were not working the way I wanted them or whatever. And I had this visceral sense that when I put the scope probe down and went home and slept the world had stopped; no further progress was being made in my absence. As soon as I put it down we were falling behind the competition. The first machine we designed at Multiflow, we had seven people. Paul was one, I was one, Dave Papworth was one. There were very few people to design basically a super computer. And we pulled it off but it was a lot of work and...

1:10:27 PE: Who were the other four on the team?

1:10:29 BC: Well a guy named Chandra Joshi was one, he designed the memory subsystem.

1:10:33 PE: J O S H I?

1:10:34 BC: Yeah. And he eventually ended up at a start up that got bought by Cisco at which point I hope he retired rich, at least I hope so. A guy named Rich Lethin was one of them: the floating point board designer.

1:10:47 PE: L E T H I N?

1:10:49 BC: Yeah, he's still a good friend he runs a consulting business in Manhattan. And those guys worked on lots of interesting projects, compiler stuff, hardware stuff. Rich teaches VLSI at Yale, he's a real good guy, he's a PhD from MIT. There was a guy named Mauro Lupero who did the IO design.

1:11:12 PE: That's L E P E R O?

1:11:13 BC: L U P E R O.

1:11:15 PE: L U P E R O okay.

1:11:18 BC: And let's see who else do we have, is that up to seven?

1:11:24 PE: There's one more so.

1:11:25 BC: I think I'm missing one.

1:11:27 PE: So let's see, Dave, Paul, Mario Lupero, Chandra, yourself and you did mention one other. Paul and who were the first two?

1:11:47 BC: Paul and Dave Papworth.

1:11:49 PE: Dave?

1:11:50 BC: They were the first two. Yeah.

1:11:52 PE: That's six so anyway.

1:11:53 BC: I'm sure I'm forgetting somebody because I'm pretty sure the answer was 7. Yeah.

1:11:59 PE: I'll try to fill that in later because it's probably in here somewhere.

1:12:01 BC: Yeah, it might be but if it isn't I know where to look to figure it out. Anyway that was the team that did the first version of the machine that we sent out. And then we did a second version of the machine a couple of years later. We means Rich mostly because we didn't change a whole lot of machine for that version, just the floating point. The real big re-spin was in going from TTL implementation technology to ECL gate arrays. And we weren't finished with that when the company ran out of money.

1:12:30 PE: Okay TTL?

1:12:32 BC: TTL is Transistor Logic.

1:12:34 PE: Okay.

1:12:35 BC: And that's a family of chips that were circa 1965 or so and to this day you can still buy those chips. But they were one of the first families you could get them as simple as four NAND gates in a package or as complicated as you want. They ran on five volt power supplies, they had certain signaling conventions that meant you could plug and play them together. They were a fairly easy sort of a family friendly kind of logic family. Unless you pushed it as hard as we did in which case it was nasty. Oh boy, we had what are called ground bounce problems. If you take a wire and you got a current running through it, that's one thing, DC current let's suppose. Now let's suppose that you change the amount of current by flipping a switch. There was a current flowing, now there isn't. That wire had a certain inductance, every wire does. Well the inductance means you've built up a magnet field that goes commensurate with that current and when you turn the current off the field collapses. As it collapses it induces a current in the wire. That's how transformers work.

1:13:42 PE: Yeah, yeah, yeah.

1:13:42 BC: So the point is digital systems switch currents all the time and every time they do they induce a voltage across the wire. Every chip does, every micro processor has this inside. Well the problem was that the TTL family in the 1980s had just recently switched to a much faster technology called Advanced Schottky, or what Fairchild called FAST. It was still Advanced Schottky, the point was that the switching edges had very fast rise and fall times,.

1:14:11 PE: Is the second word there Advanced?

1:14:12 BC: S c h o t t k y, Schottky.

1:14:17 PE: Is this a person's name or ?

1:14:18 BC: Yeah some guy named Schottky invented something associated with this. But in a practical level what it meant was to go from a 0 to a 1 or a 1 to a 0. It used to take a certain amount of time but now it took much less time. And so dI/dT, the change of current versus time, was at a much bigger number than it used to be and that's directly proportional to the voltage induced. So suddenly we're getting these huge voltage spikes that were upsetting the internal logic. If you had a flip flop it could easily clock itself again and it would get the wrong answer. It was just a nightmare. I designed a floating point square root divider unit for that machine and went into it completely naive, not knowing that this ground bounce stuff meant it wasn't going to work. Octal buffers were susceptible because they were high current devices and could switch all at the same time, in the same direction. For those, it wouldn't surprise you that you would get a big inductive spike. But I was designing with four-bit slices in tiny little chips; why would four bits have this problem? But it did. I still have scope photos, I have notebooks, that is how I do engineering, right? I bet I have 13 of these over the years, showing the ugly signals I saw in the lab.

1:15:33 PE: _____.

Multiflow

1:15:33 BC: These are not the good ones. You should see the ones upstairs for the Multiflow.

1:15:38 PE: Let me describe it a little bit. This is a brown notebook, it’s a computation notebook. On the outside it has got pieces of paper cut out and pasted in and then lots of notes in it in kind of a web book fashion, all around it. Thick.

1:15:56 BC: Yeah. So I've got these notebooks all the way back to the seventies. That's just how I organize my head and I can find things that I want. I have lots of pictures from my battles with this problem at Multiflow tracking this down: it has got to be this chip, it is not my design, it has got to be the chip. Eventually, Rich Lethin and I made a pilgrimage down to TI in Richardson, Texas and we said as best as well can tell many of your chips don't work properly. And does this come as a surprise to you? I half expected them to say, "what you are out of your mind! You've done something wrong. Come on, you don't know what you’re doing. Go use somebody else’s chips." But no, they said "Yeah we know, let me see your list." And they looked at the list and said "here is some more that you don't know about."

[laughter]

1:16:39 BC: And we went "Thank you very much. This is what we needed." At lunch I asked them "how did this happen?" and by the way it wasn't just TI. Their parts were no worse than anyone else. Motorola's were no good, Fairchild's were no good, they all had this problem. And so I asked TI, "how did the entire industry fall on its face at the same time?" We are killing ourselves trying to work around the shortcomings in your silicon. And the guy said "the first generation of TTL was done by the old gray beard guys that really know what they are doing. The new generation was done by kids who are straight out of school who didn't know to ask what the change in packaging would do to inductive spikes." He pointed out that what had happened was we had gone from 14 pin DIPs, which are small, to 24 pin DIPs which are really long and he said the dies right in the middle of that package and the power and ground have long leads, from the package out to the corner pins. Now if that had used middle pins they would have been short. But at the corners they got really long and therefore power/ground current inductance got really bad. I said "that is just wonderful because we picked TTL specifically to try to avoid any such problems with the implementation technology." We said to ourselves, Multiflow machines are going to be such a risk at an architectural level and should pay off so well that we can back off the aggressiveness of the silicon technology, we will make up with it for the compiler. We will take no risks here. Instead we signed up for this TTL family, and inadvertently just shot ourselves in the foot with it. It took us about a year to work around all the different ways in which it was failing. It was incredible. You learn a lot that way.

1:18:21 PE: So know if I remember this right from what I read, that this brief overview, the company opened in 1984 and the first product shipped in 1987?

1:18:32 BC; Yes, I think that is right.

1:18:34 PE: And these were mini super computers? So that is that is kind of a special niche. Who was buying these things?

1:18:41 BC: Yes. Well not too many apparently. [chuckle]. I used to keep a list on my desk and it said the top 60 mini super computer vendors. This field was considered to be a burgeoning opportunity in the 80's. It turned out to be not burgeoning. [chuckle]. But no one really noticed that right away.

1:19:04 PE: It was a mini version.

1:19:04 BC: Yeah. It was a... Kind of a small explosion I guess. I think the problem was they were expensive. I think that was one of the big problems. Our smallest machine was like $300K. And the biggest one was about a million bucks. That is another thing I learned at Multiflow was, in about 1988 or so I guess, we visited a company, it might have been Delco. Normally at these visits you do an architecture pitch and then the marketing sales guys do the talking, I am there to answer technically questions. Then at the end of it they say "so can we sign you up? You guys see how good this technology is.”

Normally at that point the guy you’re pitching to says "this is great, I really appreciate you having come in here today. I learned so much. You guys are doing such great stuff. I will get back to you." What he is really saying is "Please get out. I am not going to buy your machine. " But this guy, bless his soul, said "well I like your pitch, I think there is a lot of really good ideas in this machine, but I am not going to buy one." I said "well I am really glad you told me that. I am not glad you feel that way, and I want to make sure you got the essence of our pitch. I am telling you that we are trying to sell you a machine that compared to a VAX, is 5 times faster and 1/3 the price." I said, "Not in my entire life do I expect to have that kind of a winning advantage over a competitive machine again, and yet you don't want it. I have got to know why, because your answer may change my career here."

And the guy says “sure I will tell you why. My job is to make sure that my company has the computing cycles needed to get our jobs done. I know that I can do that with VAX's because that is what I have proven. That is what I have got in house. Now you are offering to come in and replace what I have already got with another machine isn't directly compatible. I would be taking a chance on that front and it could be a problem. You also want to charge a lot of money, on the order of half a million bucks. And I could come up with it, after all the VAX’s cost even more, but I am going to have to require a signature attention all the way to the top of the company. If anything bad happens after that, I am gone. They will fire me. That number is too big. He said “I am not incentivized to take a chance and I am dis-incentivized from the point of view of my own career." That night we were on the plane on our way home and I told Josh, "if this guy is representative of the industry, then we have a real problem here. We are not going to sell machines at this price range because he knows we are a start up, we could go away next week and he would be stuck with this half million dollar piece of hardware and no one to support it. We need to find another plan." I said this business plan is starting to worry me a lot.

Incidentally when you start out designing anything reasonably novel you can easily sell between 10 and say 20 of them. The CIA will want one or two, the NSA will want one or two, the navy will take two, I mean the question is not can you sell any machines the question is can you sustain the sales after you have sold some. And that is what we were having trouble doing. So that is why this anecdote was so telling for me was, here's a guy telling me you have problem with your business plan, and it is a really big one. At that point Multiflow approached Digital and got some money, and for a while there they were going to be a partner. I don't know that the deal was but it didn't work out.

And then at the end Multiflow ran out of money and just pulled the plug on the place. In the course of doing that we were talking to people like Motorola showing them our really cool compiler technology, offering to map it to their hardware. The idea was that if we got together maybe there's something between their microprocessors and our great big box that we can make and be good target for this compiler and if we had such a thing, boy you would have something that no one else does. They took it seriously. But at the time, the 1080's, the packaging technology didn't provide enough I/O's. You could not do enough pins and since you had to limit the number of pins you need lots of bits to come in and off that chip fast. There is no point in putting seven units on this chip if you can only do one thing through the external bus. It is a bottle neck that you can't work around. So that was the beginning of the end. That was when I first got this inkling that these machines were too expensive.

1:23:29 PE: Now one of the stories you told the [1:23:31] _____ chronicles about Multiflow or about your time there was the story about being at the copier with a plans for a house, building a house. I take it was a house that you built or were...

1:23:44 BC: Yeah we did end up building a house.

1:23:47 PE: Yeah. And the vice president I guess coming in and giving you a distraught look.

1:23:52 BC: Yes, his eyes got real wide when he realized what we were doing, we were committing to the area in a really fundamental way. Yeah.

1:24:01 PE: Yet, he is saying to you in a non verbally, maybe that is not a good idea. [chuckle]

1:24:03 BC: Exactly. You could see it in the way that he walked away; I realized he couldn't really say "are you sure that's wise". That would convey a message he doesn't intend, that we are in big trouble here. We did that house in 1985. It actually worked out just fine. But he was privy to problems that I didn't have to worry about. I was trying to make the technology work. It was his job to keep the company floating.

1:24:39 PE: But that must have been exciting? The whole experience.

1:24:41 BC: Yeah, it was great. It was really fun. I mentioned earlier about my feeling about Bell Labs that they were islands of really good people surrounded by expanses of people that weren't. I didn't feel that way at Multiflow. It seemed to me that pretty much everybody was a star. If you got an opinion from somebody you could probably take it to the bank, because they are probably right. That was a spectacular running curve. Trying to keep up with people like that causes you have to be on your game all the time. The other thing was , I have always wrestled with this question, how do you throw yourself wholeheartedly into a technical effort while you have a spouse and kids at home? It is really hard to balance those properly. There is no good answer but there are some bad answers. For instance, when I was running the project with Intel I felt very bad when I heard somebody got divorced. With a team of 800+ people, that statistic was going to happen no matter what. I didn't want it to be the case because all I had said was you got to be here this weekend and that was her birthday or something. I just didn't want to do that. But yet at the same time if you don't give above normal devotion to your project you're probably not going to put out a world class product. So striking the balance is really tough.

1:25:52 PE: What was that like for you? Because those years at Multiflow must have been when your kids were very young. Because your first child had just been born.

1:26:00 BC: Yes in 1985.

1:26:01 PE: Just born and then your other 2 were probably born during that time?

1:26:04 BC: Well Ken was born in 87 and Kristen came in 90. So she was post Multiflow but the first two were during it. When we were trying to get the last machine running, we didn't finish before the company died. That was the hardest because the kids were in existence and they were really little, so you don't want to miss that, but we also knew that the company, this incident with Delco or whoever it was had already happened and I was already starting to worry about the need to find another path quickly. Well, getting a new machine out might have been a good thing. The thought was this machine was going to be way faster than the previous one. Maybe it will put some distance between us and the competitors and we will start selling things; who knows. So we wanted to finish it, we were highly motivated. I was designing one of the key gate arrays that would go in there. So I worked like crazy, to the point that I was there night and day. The tools of the day were such that I had a terminal at home next to the bed and we were running on a VAX, running VMS and I couldn't figure out how to do a shell script or a batch file. So when part of the CAD tool chain finished, I had to literally type something in and hit a carriage return. So I would kick of the first part of the tool chain at work and I would go home, go to sleep and the beep from the thing would wake me up and I would type in the next tool, hit carriage return, and go back to sleep. We did that for days. On a lot of nights, Ellen would bring the babies to work and give me dinner, that went on for about 6 months.

1:27:41 EMC: I remember that.

1:27:42 BC: Yeah I remember that too

1:27:43 EMC: Good morning

1:27:44 PE: What was she doing during that time?

1:27:45 BC: Raising the kids. She was working part time with Multiflow too.

1:27:48 PE: Oh really.

1:27:49 BC: In the compiler group. So it was quite the busy time. But the way I was viewing it was that ultimately it was your job to get the balance right but some of the harbingers that you have to watch for are, if you find yourself on the critical path of a project wide effort, everybody takes their turn in the hot seat sooner or later. When it is your butt in the hot seat, work on work. Get off the critical path as soon as possible. Work as much as it takes to get back out of the hot seat and give some one else a turn and when that happens go see your wife and kids. Try to make sure to redress the balance. At the end of it if everybody is equally unhappy with you, then you probably have done about as well as you can.
1:28:34 PE: [laughter]. Yeah.

1:28:36 BC: It sounds bleak but that is how it feels.

1:28:38 PE: I understand. Yeah.

1:28:39 BC: I The guy who gave the keynote address at Colorado this year is Barrack Obama's strategist for his 2008 presidential campaign and obviously he did a brilliant job of that.

1:28:55 PE: David [1:28:55] ____.

1:28:55 BC: David... Is that his name?

1:28:56 PE: Yes, he's Obama's main advisor.

1:29:01 BC: He is writing books that are coming out real soon. Here is the message that he gave to the graduating class at Boulder. He steps up to the microphone and says “I didn't graduate from college. I goofed around a couple of years and quit. But I've done okay.” [laughter]. I think to myself, hello, I am a parent, could you please shut up. That is not the message that I want you to give the kids. He also said if you want to accomplish anything useful in life you are going to have to yourself into it and when you do it is real easy to forget to call your parents or to forget to give the attention to your relationships you need to do to keep them alive. He said he did not do that very well, but that looking back on it, he was not sure he would change anything.

1:30:02 PE: Yeah.

1:30:02 BC: So yeah, that's what I remember from Multiflow.

1:30:02 PE: Well, so what was it like working with Josh.

1:30:10 BC: Yeah, Josh.

1:30:10 PE: Josh Fisher.

1:30:12 BC: Josh is cool. Josh is really bright and he's very pragmatic.

1:30:17 PE: So he was a Yale Professor right? And he must have quit to serve Multiflow?

1:30:21 BC: Yeah. I don't know if he took a leave of absence or what but he was full time at Multiflow and he brought a guy named John O'Donnell with him. John was also ex-Yale, but I don't think he was a Professor.

1:30:31 PE: How about the guy?

1:30:31 BC: Well, that was Ruttenberg.

1:30:32 PE: That was Ruttenberg.

1:30:32 BC: Yeah. John Ruttenberg was the compiler guy and John O’Donnell was a hardware systems type. I'm actually not sure exactly what role John O’Donnell played at Yale.

1:30:42 PE: Okay. He was the guy who led the extremely long instruction... Enormously long instruction project that turned into Multiflow.

1:30:54 BC: Yeah. So there's small set of folks but they were really good. They were really smart. There are some very colorful guys there too, such as Tom Karzes.

1:31:01 PE: Spell it.

1:31:03 BC: Karzes.

1:31:06 BC: Yeah. Tom was absolute manic about bugs and code. He detested bugs with every fiber of his being, and to tell you the truth, I never saw him have a bug in his code, but it wasn't his code that was the issue. It was everyone else's. Tom would have just as much heartburn over someone else's code that had a bug in it. And he would have no compunctions about sharing that belief with you. For instance, some guy came into my office once, whom I didn't recognize. He said, "you don't know me. I've only worked here a week but, what I do with this Karzes guy?" He said "He came into my office this morning and said, Look if you had any ability at writing code whatsoever, I would try to help you, I would try to fix what you're doing wrong but you don't, you need to get out of this field. Your code sucks and there's no hope of ever fixing it." I said "Don't take it personally, he says that to everybody." So Tom was rough. He was writing some of the most difficult code in the compiler and I think he felt the pressure. There were some colorful characters, and sometimes they were hard to work with but man, they got things done. They were amazing about what they could accomplish if you just overlooked the social interface part.

1:32:21 S3: Well, it seems like the compiler might have been the most successful piece of the whole project. It sounds like that survived and is still in use and you see any assimilator. And Intel may have...

1:32:35 BC: The Intel iTanium project started with the Multiflow compiler. They gradually rewrote pieces of it to make it go faster.

1:32:39 S3: Yeah. I'm sure

1:32:40 BC: But some of the fundamental algorithms are still in there. It was a brilliant piece of work. When That was one of my misgivings when I started at Multiflow. It seemed plausible to me that the more complicated the pieces, the more buggy something's going to be. And I already know that people using optimization switches on normal compilers complain a lot about the code being buggy, so they turn them off again. What Multiflow’s compiler team was going to try to do was light years beyond that, so I worried they would fall on their faces, and the complexities would eat them alive. That's not what happened due to this maniacal attention to detail that the team applied to the correctness of what they were doing. The speed of what they were doing was not all that great. Sometimes compilations would run for three days. That was a serious issue, but when it was finished you could hang your hat on the code that it generated. It's going to be right.

There were other things we learned. The performance analysis team would often show us code that had been running on Crays for years and the sales opportunity was that Crays were getting old and if a customer could run this faster on a much cheaper machine, perhaps they would be interested. And so our job was to make sure that their code would run faster on our machine. We'd look at their code and see that it was littered with assertions, programmer inserted assertions that would say things like "i is never greater than j in this inner loop", and we'd look at this inner loop and say "yes it is." The assertion is wrong. This happened all the time. If you saw people's fingerprints on the code, that’s where the bugs would be. Now, the nature of the data set that they were running into that model presumably meant that they got a useful answer and they were okay. But a general data set could have failed miserably on that code. We saw this over and over. But our compiler if generated a code it's probably right.

So those compiler folks, I take my hat off to them, they achieved something amazing with that thing despite facing unprecedented problems. We would sometimes tease Josh and say you sold this whole paradigm as being "You feed me any junky spaghetti-like code and I will run my magical compiler on it and extract all that wonderful parallel performance. Then I would run it lickety split and you'll be so happy. But the truth appears to be that we're good at vector codes just like vector machines are because the parallelism is obviously out there and the spaghetti code just doesn't speed up very much." We had these buttons with the word vectors with a red slash through it and we would say we don't do vectors, we do far better than vectors. But then, we discovered really there's two types of codes out there in the mini super computer world. There's vector code and there's junk that nobody cares about and we can't speed up anyway. So yeah, we're pretty good at vectors but here we are with buttons that said no vectors.

We had to back off at that marketing message or we wouldn't have sold any machines. There were things that the compiler had to do about register management. Paul Rodman used to say spills, spills, spills, spills, spills, spills. What he was referring to was you can get the compiler into a kind of mode where it doesn't have enough registers anymore, so it takes a set of registers it's always allocated and it spills them to memory. That's the spill part and then it starts using those the reverse happens and so spills, spills, spill. And when you get into that syndrome, the machine is running very slowly. You don't want to go there.

So we had lots of occasions where the hardware guys or the performance analysis guys would get together with the compiler team and say "you know, this policy you have in place about this kind of code right here is leading to this kind of behavior that's really ugly", and then we'd work out some compromise that would keep it out of that particular troublesome neighborhood. If, to this day, you actually look at the performance numbers that we posted at Multiflow, the rules for certain official benchmarks are that you have to tell all the compilers switches that you used, you know, just to keep you honest.. Our compilers switches were a very long string. There was so many that was incredible because we were tuning the compiler. It was very programmable so you could match with pretty much anything you could think of and we did. So it was quite a riot. I learned a lot. The other thing was this also useful when I went to Intel. When I first got an office, I showed up on a Monday and...

1:37:16 S3: At Multiflow?

1:37:16 BC: My first Multiflow office was small. It was about the size of where you're sitting to the doors behind you and there were two of us. There were two of us. The other person was a compiler guy named Stefan Freudenburger and he was sitting on that chair looking the other way. But when I first got to Multiflow, John O'Donnell shows me my desk, and he said "This is the seat that the albatross used to have." And I said "who?" and he said, "The guy who used to sit here was nicknamed the albatross because he wasn't getting anything done. We fired him last Friday and now we're going to get you design what he failed to do." Great. Just ducky. What he had failed to design was a floating point unit and I'm thinking to myself "I hate floating point and don't know all that much about it".

I climbed the floating point learning curve, and I was sitting in the office with Stefan. He was a compiler guy and so if I had a question, all I had to say it out loud and from one meter away another head would say the answer. I even managed to contribute something to him once:, I had been looking for a first order indicator of how good compiled code was so I would know if I should recompile it with different switches I wrote a little utility that would just count how many floating point operations there were in a loop and the higher the better. I wrote this utility in about half and hour, and Stefan looked over my shoulder at one point and said "what is that? Where did you get that tool?" I said "I wrote it and it's trivial." He says "I need that. Give me that."

There were several occasions where that kind of unexpected synergy happened just because we were so close together. At Intel, I looked for a similar synergy, and noticed that the cubicles we sat in were fungible; you can move people around. At some point, we were going to do a massive reorganization and had to decide who's going to go where. I suggested that we lay out the cubicles the same way the chip floor plan was going to look, since neighboring units on a chip implies communicate between those units. And anytime that happens the people who design those units had better be communicating back and forth to the same degree. We tried this and it worked out really well for about six months, and then because we kept adding people to the various units and there was nowhere from them to go because they were in the middle of the chip, the scheme eventually came apart. But for the time being it worked really well.

1:39:48 PE: I remember the story from the Pentium Chronicles and I thought that was incredibly interesting. I'll tell you some things later but there's... Some of the people at my school have done a lot of research into distance collaboration versus stuff and then the finding is typically the face to face interaction is just orders of magnitude, faster and deeper and better than anything you can do at the distance using any technology, that we have now.

1:40:18 BC: Sad but true.

1:40:18 PE: Yeah. And I thought that was a great example of how to maximize efficiency of a thing like...

1:40:25 BC: Well, it worked. When I first started on the P6 project, for some reason they stuck the marketing guys more or less in the center of the design team. Normally, they would be exiled to the edges because the match just isn't really there, but because they were there, we saw them a whole lot more often and that had a beneficial effect. If you don't see people on a routine basis, an awkward dynamic creeps in, and you can't stop it. It's just going to happen. You start thinking us versus them, if a questions shows up and you're not sure why they did it, by default you think it was a bad reason instead of a good reason. But if you see them everyday, you're going to cut them slack. I don't know why that happens but it does. The relationship between marketing and engineers is often testy anyway. So anything that you can do to lower that friction, it's a good thing. Each of those groups has something important to offer the other one.

1:41:30 PE: There's also this phenomenon that just people you see everyday, you constantly reminded of whatever task it is you're doing with them. So you're motivated to continue to work on it. If you only talk to somebody once a week then, you're driven by the conference call it's coming, and don't think about it until Friday morning before the Friday afternoon call and you do all the work then, [1:41:53] ___ do it

1:41:55 BC: It was a Packard or Hewlett. One of those two guys was famous for management by walking around and I saw the value of that when I was at Intel. It's like you walk into a subordinate office and hey, "how is it going?". They'll often tell you something that otherwise you'll have no clue about it, like they're hopelessly stuck. Or that guy over there owed me something he didn't deliver, and you think I can fix that.
1:42:24 S3: Yeah. Tell me about the end of Multiflow. When did you start to realize the ship is going down?
1:42:32 BC: Well, actually I didn't... We didn't fully realize until the day that it happened. We were called in to a meeting in March of 1990 and even up to that day, we were working hard. It wasn't one of those things where the rumors were hot and heavy so why sweat it. No, we're working our buns off and he called us into this meeting and he said, "Hey, pulling the plug. Sorry." And further...

1:42:58 PE: It sounds like it was a deal possibility of DEC?

1:43:01 BC: Correct.

1:43:02 PE: Kind of right up to the last minute and then...

1:43:03 BC: Because we had had previous deals with them too. They already invested more than a million bucks in us. And so yeah, there were some beliefs that they might come out. Here's what we were being told. It was possible that they were going to commit to Multiflow's technology for a future version of the VAX or something like they were, or Alpha. And actually that might have been a good idea but it didn't happen. There were internal factions inside of DEC that didn't want to go in that direction. So yes, that didn't work but here was the part that got me. My wife, Ellen, was part of the company too at the time. Multiflow was self-insured. She was pregnant. So as soon as they pulled the plug, we were uninsured with a pre-existing condition. That mattered, because I was looking for a job. Well, I'd been invited several times by Jim Aylor at UVA to consider becoming a professor. I always thought it would be fun to be a professor somewhere and I visited them and had a good time talking. I knew a lot of faculty and so on. So I went out there for an interview, for a potential job there and...

1:44:07 PE: This is in 1990?

1:44:08 BC: 1990.

1:44:09 PE: 1990, yeah.

1:44:10 BC: Anita Jones was the department head at the time for computer science. I spent the entire afternoon in her office and she called one person after another and trying to find a way to work around to the problem of adding a new faculty member to an existing insurance base if they have a pre-existing condition. At the end of the day, she was very apologetic. She said "I can't believe it comes to this, but we can't take you under our insurance policy" and I said "I understand it's just how it works, but I hope you understand that I can't take the job under these conditions because I'll be risking everything I own that the pregnancy would go with no complications."

1:44:55 S3: That's a story for the healthcare debate.

1:44:57 BC: Yeah. Isn't that weird? I went to Intel and I said, by the way I've got to tell you this before we waste much time, I've got this pre-existing condition. He just looked at me and he says, I don't care, we'll take her under our insurance policy no problem.
1:45:16 PE: So were those the only two possibilities you considered?

1:45:19 BC: I don't remember, yeah there were other ones but I didn't take them very seriously as I recall it. One of them, connection machine, Thinking Machines was the company. And a fair number of Mutliflow people went to that. Because it was the cool new technology, it was in the same general area and so on. But I personally didn't...

1:45:40 PE: Was the same idea?

1:45:41 BC: Yes it was in the same area anyway. But I wasn't that convinced that it was a good way to design things. I mean Danny Hillis was the star and he was getting lots of press and had lots of funding. But I just didn't think that machine was going to go anywhere, because I felt it would turn out to be unprogrammable. I didn't think that they were going to find enough apps that could fit that kind of architecture. Some Multiflow people went there, some people went to a place called Convex, which was one of our competitors in Texas. Convex had really ticked me off at one point, because they were doing things like calling our prospective sales and telling them, “call the bank, Multiflow is going to die”, about three years before we actually did. They were just throwing mud anywhere they could, they were street fighters, man they were tough. And I just thought some of it was below the belt.

At some point some Convex recruiter called me and said sorry about Multiflow dying would you like to come to Texas and keep designing mini supercomputers. And I said, "I'm really glad that you called because I wanted to say the following thing to somebody from Convex. I would rather flip burgers than ever come and work for you". And he laughed because wow, you must be really be upset about this. I said, "I'm not upset because we went out of business; I'm upset about the way you ran your business and that is a big difference". He said "okay, I don't know anything about that", and I said "I believe you, and no hard feelings dude but I'm not going to work for you".

Part 3 (1990-2001): Intel

1:47:04 PE: So how did the Intel contact happen, did you call them up or did they call you or?

1:47:10 BC: I'm not exactly sure, but it was an obvious thing to try. At this point in 1990 we had tried to stay in the Northeast because Pittsburgh is where our families are and although it's a long drive, we can go there for holidays. But if I want to stay in the computer business, Boston's clearly going down, nothing's taking off in Pittsburgh in any real way associated with computers. Santa Clara, California is the obvious place to go but I don't want to raise my kids there. And Austin, Texas might take off, or maybe it won't but I know that Oregon has an Intel site. I wonder what's going on there. I knew they were designing some chips I just didn't know what. And then fortuitously I got a phone call from a guy named Randy Young who was the general manager of the new x86 division in Portland. And he said "I've been told that there's a bunch of people available from Multiflow and we could pick up some really good talent and someone said we should call you". I asked him what he had in mind. He said, "We're going to design an x86 chip to beat California, which is where the team is now that does all the x86 chips. I think we can beat them".

I said "that sounds like fun." At this point I'm thinking I've just finished five years of blood, sweat and tears designing a machine that all of the people in the field said was really cool but few customers bought. I figured why don't I go to a place where they might make fun of the machines but customers will buy a zillion of them: and that would be Intel.
So I sounded him out on what kind of a team he was going to put together, how he viewed what he was doing and all that sort of thing. And he said the Pentium is going to have 3 million transistors but the P6, they already named the project, it's going to have about 10. I thought with 10 million transistors you could do some really cool stuff. So I went out and interviewed, talked to the guys and then I found out Dave Papworth was going to be part of this team. And I thought man if he's going to be on the team, I'm in.

1:49:09 PE: So he had already decided, he'd already been?

1:49:12 BC: He was strongly leaning to it.

1:49:13 PE: Yeah.

1:49:14 BC: I don't he'd formally accepted yet, but he told me "I think I'm going to go there". And I said, "That's cool, I think I'm going to go there too, we'll have a great time". Because I like Dave -- he's great to work with. That pretty much made up my mind when I realized what a good team nucleus that was to start with. So I ended up in Oregon in June and Dave came in August and it was great. Randy Young only lasted a few months, though.

1:49:39 PE: Really.

1:49:40 BC: He did because...

1:49:41 PE: Hired you and then disappeared.

1:49:42 BC: He was on the other side of the cubicle wall from me, I mean I could hear his half of every conversation, I couldn't *not* hear it. He was on a staff meeting once a week for all afternoon and he had tremendous fights with his management in California. They seemed to want him to be the figurehead general manager but they wanted to pull all the strings from Santa Clara. Randy Young knew that if you want to build a division in Oregon that's going to really achieve big goals, you cannot control it from there, it has to be done from here. Eventually he just got tired of the whole game and punted it. Then they put Pat Gelsinger in charge, and he was there for a while for a couple of years and then they replaced him with somebody else. By that point we'd established the division, we'd established the culture. I was the first guy in the division and Randy Steck came in pretty soon after me.

1:50:32. PE: Steck? S T E C K?

1:50:33 BC: Yes. Randy ended up as a vice president there. He was a really good general manager, I mentioned him in the Pentium Chronicles lots of places. He and I agreed on the kind of culture we were looking for, which was straight talking, tell the truth. If it's bad news, tell the truth anyway, don't hide it, don't over promise, don't try to divert our attention, just face up to it. I think there were too many politics at Santa Clara. They developed a strange culture of the shell game. Anything that didn't quite go well, if you don't like A let's talk about B. Well that didn't fix A. We also went out of our way to attract talent in specific areas when we needed it, and I never saw them do that in Santa Clara. For example with P6 when we realized it was going to be a multi-processor design. As soon as we decided that, a bell went off in my head which said, every other company that has tried to do this has screwed it up on the first try because it's very hard. Caches have issues, they're complicated and subtle, it's easy to get them wrong. And so people like Sun had tried this but remember this is back in 1990, 1991 and 1992. MIPS had done it, DEC had done it, and they all required multiple spins to get it right, sometimes lots and lots of spins. So I hired a team specifically picked for expertise in this exact area, and told them, "Your job is to keep us from re-spinning this chip based on cache coherence multi-processor issues". And that focus worked really well. It was a result of squarely facing a weakness and saying we're not good at this, so we need to do something special, not just assume it's going to fix itself. A certain amount of paranoia is a good thing.

1:52:21 PE: Yeah.

1:52:22 BC: I think it was that kind of brutal honesty about one's own capacity that kind of put the Oregon team a little bit ahead of what Santa Clara was able to do at the time. Plus I think it was helpful to have the corporate headquarters 500 miles away. [laughter].

1:52:38 BC: The out of sight out of mind thing.

1:52:39 BC: Yeah.

1:52:40 PE: Can help you sometimes.

1:52:41 BC: Absolutely, please don't help me so much; stay away.

1:52:44 PE: Well this seems like a good point to stop for today and we can continue on the Intel story tomorrow.

1:52:50 BC: Okay, cool.

0:00:00 Paul Edwards: It's Tuesday morning August 25, 2009 and it's Paul Edwards and Bob Colwell in part 3 of our oral history interview. So yesterday when we ended we were talking about how you got to Intel and you were speaking of Randy Young hiring you and how that happened. It occurred to me after I went back last night to ask you, I can see the pieces of your experience that made you a desirable person from Intel's point of view. You worked in this microprocessor design group at the Bell Labs. You spent 5 years working on mini super computer, all the RISC and CISC and you had come down on the side of CISC in a certain way, which was their technology. And you studied the Intel 432 very extensively, so about Intel and how it worked. And you'd been there and so on. But the interesting thing what happened to you at Intel is you went very quickly from being part of small team of architects to leading this really colossal number of engineers. I think it was up to more than 800 at the time?

0:01:25 Bob Colwell: Yeah, but I wasn't indirectly responsible for those. Randy Steck had to manage the bulk of those people. The most I ever had under my command was about 150.

0:01:33 PE: That is still quite a few.

0:01:34 BC: Yes it was.

0:01:37 PE: Orders of magnitude more than working with a small group of 5. And you were hired to lead this project so they must have realized that you would soon be managing.

0:01:45 BC: I don't think so. I don't actually think that's what they had in their mind at all. I think when they hired me, they knew that needed some sort of high level leadership. I wasn't the only one. They hired Dave Papworth, and you've got Glen Hinton from the 960 group shortly after that.

0:02:01 PE: What's the 960 group?

0:02:04 BC: The 960 was the chip that the remnants of the 432 team went on to design. Glenn was the leader of that and then right around 1990, there was high level decision made in the company, that Oregon would no longer do 960s. They would move that effort down to

Arizona and the team that was left behind would become the core of this new x86 development group. So we inherited Glenn that way and he was a very senior chip designer. I think essentially they thought the three of us could probably drive this thing forward. And even then to tell you the truth, in the first year so, the exact power structure was not clear. My boss was then Fred Pollack who later became an Intel fellow and a very well known guy inside the company but when I first started as architect on the P6 chip, within a couple of weeks, he came in as the architect manager. I think he lacked faith that the three of us could pull this off. So he contacted a group called Metaflow. Not to be confused with Multiflow, no connection. Metaflow was a San Diego group startup.

0:03:15 PE: M e t a flow?

0:03:16 BC: Correct. They were trying to design an out of order microarchitecture for chips. Fred thought what the heck, we can just license theirs and remove lot of risk from our project. But we looked at them, we talked to their guys, we used their simulator for a while, but eventually we became convinced that there were some fundamental design decisions that Metaflow had made that we thought would ultimately limit what we could do with Intel silicon. Maybe not in the first chip but definitely on follow on chips. Plus we had complete confidence that we could do what we set out to do, we had more than enough talent. So I think it took out for a while for Fred to sort come around to that. Of course, it's his career on the line, too. He probably was looking for ways to reduce risk and there's nothing wrong with that. All I'm saying is, I think there was probably some serendipity in the whole thing as well. By the time summer ‘92 came along, I think Fred felt that he had to choose among me, David and Glenn as to who is going run the group management while Fred was on sabbatical.

I can't really guess how he determined that he should pick me. The way I view it, those guys are so smart that they spend a fair amount of their time trying to down-translate what they were thinking to terms everyone else can understand. I wasn't needed in that kind of role. I would have got it done but these guys were spectacularly good.

So I just found another roles, other day to day kind of driving the thing forward, keep track of what was decided, keeping upper management apprised of important developments and so on. The nuts and bolts of the architecture development, so what I tended to do was look for places that weren't getting done. And those are the areas that don't always occur to people that are as ridiculously smart as Dave and Glenn. Because they don't need it, it's the team that needs it. The big brain guys can just do whatever they need. I think I sort of backed into a sort of quasi-management role just because that's what was needed.

0:05:19 PE: Kind of fell into it organically?

0:05:21 BC: Yeah, and I think I do a reasonable job at down-translating better than those other two guys because I don't have as far to go [laughter]

0:05:33 PE: That's a very modest thing to say.

0:05:34 BC: Who knows, who really knows.

0:05:41 PE: Just a little bit more about how you ended up at Intel, did anyone in particular had recommended you as a choice?

0:05:52 BC: I don't know.

0:05:55 PE: You know, the way you described it yesterday was that Randy Young had just heard Multiflow was collapsing and that there was talent available and he kind of called around.

0:06:06 BC: You see the thing is... I mentioned George Cox was my mentor for the 432 research that I did as a grad student. I don't recall if he had any role to play in making Randy Young aware that we Multifloids were on the market, me in particular. I just don't remember that very well. I just remember getting phone calls, I was also interviewed at Santa Clara into the Santa Clara team as was Dave Papworth. That was kind of weird because what I remember most about that interview was, we were talking all morning with various people, got in the car to go to lunch and I get in the backseat and there was a guy in right front who ultimately became a friend of mine. But what he said as soon as we turned out was, “Colwell were you at Multiflow?”

And I said yeah, and he replied, "what the heck were you thinking when you did the following design decision", referring to some specific issue in the Multiflow products. It turned out he was working at a rival mini supercomputer start up called, I think it was Convex. No it's Cydrome. So I didn't know who this guy was and I said, “well this is why we did it, and you could question whether it was right. But you said you worked at Cydrome, right? What the heck were you thinking when you put your floating point on this huge PC board, when all you needed to do was buy a chip?” It was kind of fun; I think he enjoyed getting shot back at. That was the Santa Clara team.

The other classic thing was a question I was asked at that interview. You'll think I'm making this up and I'm not. The question was, "you're the commander of a garrison in the desert with sand all around for many miles. Suddenly a submarine surfaces in the desert, and starts firing on your garrison. What do you do?" I'm thinking this has got to be the dumbest question ever. So, okay you can ask me a dumb question, and I am going to give you a dumb answer. I said, “I'd send my destroyer out there and sink that thing.” He said good answer, and he wrote that down. I thought “I don't want to work here [Laughter] if this is what you think constitutes a good answer”. You never know. Maybe he read the book *How to Move Mount Fuji*. Did you see that one? It's a book about how Microsoft is reputed to interview people and there are many screwy questions like that.

Leading the IA32 design team

0:08:17 PE: Oh. So this gets into things that you probably learnt about from other people rather than first hand. So the 486 was introduced in 1989 that's from Santa Clara, and the second design team was established in 1990 to do the next generation. The P6.

0:08:52 BC: The next generation after the Pentium. 486 came out in 89, and then that team had already been on to the P5 project and that became Pentium. The Pentium chip came out about 1993 or so, and we were to follow that one with the P6 chip.

0:09:03 PE: And that was a kind of typical Intel MO, right, that they would have sort of effort.

0:09:20 BC: It became a standard Intel method, but prior to P6, they only had one x86 design team. So they would get a chip out of that team, start another one, then start another. It was just when we Oregonians came along that they were making enough money to be able to support two different teams. Then they noticed there was a certain advantage to doing this overlapped execution development model. There probably should have been more advantages to tell you the truth. For my taste there wasn't quite enough synergy between those two teams. And that wasn't because of the Santa Clara team, they were very nice. I went down there a lot to ask them questions about x86 and how they did validation. The kind of tribal lore that you needed to know. And not once did I get pushed back on by any of those guys for being from a rival team. Never happened, but still at least at a platform level for instance, the Pentium guys had inherited the interconnection between their chip and the world. Also known as the front side bus.

They didn't tweak it much and it would have limited us terribly with our P6 chip. So we just ignored it and launched off in a different direction with our frontside bus design. I recounted that in the book that this eventually became an issue that we'd gone off and done that. So there were platform things that we did not share very much between Santa Clara and Oregon. I also think that became a worse problem later on between P6 and the Itanium developments. We tried reasonably hard to pass along what we had been learned about electronics and fast-circuits and out of order designs and so on. The Santa Clara team just did not want to go there. So we ended up really at odds on a lot of things between those two projects. It was much worse than between P6 and Pentium.

0:11:03 PE: Tell us bit about what Intel what like when you first got there in terms of their corporate culture. Some of the famous stories about Intel have to do with Noyce being a very anticompetitive guy and everybody is going to work in a cubicle. Then I guess he had probably retired by the time you gone and Gordon Moore was the chairman at that point. Did you have any contact with him?

0:11:29 BC: Yeah, I did and it was always fun, I get a charge out of Gordon Moore. I didn't see him a lot, maybe 5 or 6 times over my entire tenure there. But they were memorable times, that was the funny part. One that sticks out in my mind is when the Itanium effort was just getting going. We Oregonians were being systematically and purposely excluded from that effort. HP was worried that HP had brought some kind of intellectual property to the table for Itanium, and they did not want to see that IP appear in a competing line of x86 processors. So at the time they were being very careful to separate these two apart. We therefore knew very little about what Itanium was, we just heard rumors that it's kind of like a VLIW, which is what we did at Multiflow. Papworth and I would look at each other and think, as far as we know there are only two people in this company who know anything about VLIWs and that's the two of us.

Anyway, for some reason, there was an organizational meaning at which Albert Yu could not appear. He designated Fred Pollack, but Fred could not appear, so Fred designated me, and I showed up. So first of all I am two organizational levels down from who is supposed to be sitting there and I ended up sitting next to Gordon Moore. This was probably about 1994 or so. The presenter happened to be the same guy who was in the front of the car from when I interviewed with the Santa Clara design team; same guy. He's presenting and he's predicting some performance numbers that looked astronomically too high to me. I did not know anything about how they expected to get there, I just knew what I thought was reasonable, what would be an aggressive boost forward and what would be just wishful thinking. The predictions being shown were in the ludicrous camp as far as I could tell. So I'm sitting and staring at this presentation, wondering what are they doing, how is it humanly possible to get what he's promising. And if it is, is it possible for this particular design team to do it. I was intensely thinking about what's happening here. Finally I just couldn't stand it anymore and I put my hand up. There was some discussion, but you have to realize none of these people were really chip designers or computer architects, with the exception of Gelsinger and Dadi Perlmutter.

0:13:53 PE: Sorry Dadi

0:13:54 BC: Dadi Perlmutter, he's one of the executive VPs in charge of all the micros right now.

0:13:58 PE: D A D I

0:14:00 BC: Yeah, his real name is David, he’s an Israeli. Everybody calls him Dadi. And then Pat Gelsinger who was the chip architect, designer in 386 and 486. But most of those guys at this presentation haven't designed anything themselves, they know how to manage complicated large expensive efforts, which is a different animal. Anyway this chip architect guy is standing up in front of this group promising the moon and stars. And I finally put my hand up and said I just could not see how you're proposing to get to those kind of performance levels. And he said well we've got a simulation, and I thought Ah, ok. That shut me up for a little bit, but then something occurred to me and I interrupted him again. I said, wait I am sorry to derail this meeting. But how would you use a simulator if you don't have a compiler? He said, well that's true we don't have a compiler yet, so I hand assembled my simulations. I asked "How did you do thousands of line of code that way?" He said “No, I did 30 lines of code”. Flabbergasted, I said, "You're predicting the entire future of this architecture on 30 lines of hand generated code?" [chuckle], I said it just like that, I did not mean to be insulting but I was just thunderstruck. Andy Grove piped up and said "we are not here right now to reconsider the future of this effort, so let’s move on". I said "Okay, it's your money, if that's what you want."

Suddenly this came up again later in another guise but again Andy shut me off, he said "we're not here to discuss it". Gordon Moore is sitting next to me and hasn't said a word, he looks to all intents and purposes like he's asleep. He's got his eyes closed most of the time, you think okay, the guy's tired, he's old. But no, 20 minutes into this, he suddenly opens his eyes and he points to me and he asks, "did you get ever get an answer to your question?" and I said, "actually no, none that I can understand". Gordon looked around and says, "how are we planning to move ahead with this, if the answers don't make sense?" and this time Andy Grove said to him "We're not here to discuss that, Gordon". [laughter]. Gordon really impressed me with his ability to cut right to the heart of the matter. I saw him do that more than once, you think he's not paying attention, but when he pipes up, what he's about to say is dead on; he's not only been following the discussion but he sees where the real issue is, he points at it. That guy was impressive.

The other thing I remember, Gordon and I, I had lunch with him once at some corporate event. I was again sitting next to him and in the background, they had a bunch of Intel random facts playing on a display screen. They would ask a question about Intel and they'd give you multiple choice answers. And I said, "Hey Gordon, this is great, you must know the answers to all of these". The first question to come up is how many times has the Intel stock doubled since the beginning? Gordon being an incredibly rich man, I said "come on you must know this?" and he said "actually I have no idea". He says I don't pay any attention to that stuff" [chuckle]. I had seen a book in which Gordon was quoted as saying "'we model the Intel stock plan after IBM's.

Apparently T J Watson Junior had noticed that all the upper executives at IBM in the 1950's and 1960's were getting very rich and everyone else was not and he didn't think that that was a good idea for the company long-term. So, he put in this stock plan only to find out the people were using it in a way he hadn't planned. What they were doing was, when the day came that they could buy the stock at 85% of the real price, instead of sitting on the stock, they would buy and sell it that day and pocket the 15%. Watson said "no, no, no, folks you don't get it. You have to buy the stock and sit on it and it is going to appreciate and you will make more money that way". He saw it as an education issue, in that people who have never played at this level, just didn't get it. And Gordon saw the same thing. So, Gordon specifically said "my idea of investing, buy stock and then forget you have it".

0:18:00 PE: Yes.

0:18:00 BC: All during the 90s that was a brilliant scheme: buy the stock and sit on it, but, that was Gordon. I had nothing but good experiences with him multiple times. Very smart guy.

0:18:15 PE: Did you ever meet Noyce?

0:18:18 BC: No, I think we are only overlapped our Intel employment by about two weeks. I joined the company and then he was gone and, then he died shortly thereafter so I never met him. I read his biography, 'The Man Behind the Microchip'. He was an amazing guy.

0:18:34 PE: And you talked a little in the book about some experiences with Andy Grove and you just mentioned one other, anything else you want to talk about?

0:18:41 BC: Yes, the incident I discussed in my book was the most negative one, actually. I respect Andy Grove. I've read all his books, and I've watched him in action on numerous occasions. I think he is what CEOs ought to be. Admittedly, he sometimes drove the marketing folks crazy, because for some reason it just didn't look very hard to him and I think he always felt like anyone can do that and he would override marketing decisions on the spot. The poor marketing guys would labor for weeks coming up with names for products, positioning stories and all that and then they'd go into one meeting with Andy and he'd just make up something new on the spot, make them do it, and he knew that what everyone was telling him was wrong but he couldn't prove it. But for designers we could deal with him.

There were several times when we took him information, like we're going to have to grow the chip’s die size, or we're going to move the schedule a little or whatever it was, it wasn't always positive, but he would listen. He would make you explain why you wanted to do what you were doing, but he wouldn't necessarily ding you for it if your plan was reasonable. If you said " some surprises have come into our project and if we're going to hit the performance target we're going to have to move the schedule out a little bit and the die is going to get a little bigger.” Grove might say "okay, how much bigger" and we'd tell him and then he’d ask "well what's that going to do the yield, what's that going to do to the schedule, and what about power dissipation?" He'd make you go through your story and make sure it held together but at the end of it if he wouldn't say I really hate you guys because you're incompetent. He'd say, "'let's move ahead and it sounds like we're making progress, so let's keep going".

0:20:04 PE: Yes.

0:20:05 BC: There were, he wasn't one when these people that would shoot you for bad news. I thought that was excellent role modeling.

0:20:12 PE: Yes.

0:20:12 BC: Everybody gets reviewed at Intel once a year. At one of the times, my boss pointed something out to me that I hadn’t realized. He said, "You know, I have noticed how you deal with bad news: somebody comes to you and says boss there's a bad thing that's happened, you wanted me to do 'A' but I can't and I had to do 'B' instead. Your response is that that's a bummer; what are you going to do about it?" [chuckle] That's just my natural reaction, it didn’t come out of a book I read somewhere. It seems like the right thing to tell them. You don't want to adopt their problem as your own. The person that brings it to you needs to bring their solution to it as well, and then your role is to evaluate it and see if it's okay and so on but your job is not the bail the guy out that's what employees are supposed to do. Andy was good at this. We'd been working on the P6 for about three and half, going on four years, which is a long slog. These chips take five years, a lot of long term focus by the design team. After a few years, people are getting fatigued: they're tired the kids are getting older; the grass is not getting cut, it grinds on you. We thought, maybe we can get Andy Grove to come up and fire up the troops, just make a showing tell us he cares about what we're doing and so on.

Andy came up to Oregon and he gave a short presentation about the company and his vision, where we're going, why, where he wants our chip to fit and so on. Then somebody in the QA session, one of the engineers, put his hand up and said "Andy so what do you think about our organization here, how do you think we are doing?" As a manager I wanted Andy to say something like "I hear nothing but good things about you guys, I think you're going to set the world on fire with this chip", I wanted him to fire them up. That's not what he said. He said "As far I'm concerned you guys haven't come out of the closet yet. As soon as you make a chip I can see and test then I'll know how good you are." [chuckle]. Well it doesn't really fire you up but it's honest!
0:22:12 PE: Rose to the challenge?

0:22:12 BC: He isn't going to baby people.

0:22:14 PE: Come on you guys.

0:22:16 BC: I think he took...

0:22:15 PE: You've got to show me.

0:22:17 BC: Yeah exactly. And there's a certain honesty to that, that fit our culture in the Oregon team very well.

0:22:23 PE: Yes.

0:22:22 BC: Andy maybe took it a little too far in other ways. He was legendary for incidents such as people asking to ride their bicycles to work, but there were no showers. Andy would tell them this is not a country club, I don't owe you a shower, I owe you a cubicle, lights and bathroom, so get to work. [laughter] So yeah, he was kind of fun. Later in my Intel career, I became convinced that the executives had overblown expectations of the Itanium processor family all stemming from that original over-promises we talked about earlier. I just felt that the executives were continually being fed misinformation, maybe not deliberately but in effect, they were being told that product line was going to be capable of things I believed it would not. I also felt that the Oregon team, at this point, had a certain amount of credibility, and we were predicting what our designs were going to do much more conservatively, I think, resulting in a gap that looked very....

I went out of my way for probably the last couple of years of my career there to go to each executive and say "I don't what you're being told about Itanium, but this is what I think. You don't have to adopt my point of view wholesale but I at least want you to know that there's some concern out there that things are not as they seem and you should maybe dig around a little bit if you feel threatened by that". In the discussion I had with Andy, I gave him all my concerns, and I said "this thing is not going to deliver the performance you're expecting and besides there is a long list of technical being made that don't look right to me". No one of the execs could have just fixed the problem, I was wanted them to know they had challenges they weren't being informed of, which would cause the Itanium product line to not pay off the way they were expecting, so the company's plans ought to take that into in account.

Grove said, "well but wait a minute, why, if things are as bad as you think they might be, am I getting some reports that some of our customers like that product? How do you account for that?" I said "fair enough; I'm talking at the architectural level, not chip by chip and, so, if you still care, if you want to compare architectures, let's do it carefully. The chip that you sent out there went into a platform with 16 gigabytes of DRAM in it, and you're comparing its resulting system performance to the highest I can do on an x86 which as is four gigabytes. That difference in DRAM makes a huge difference in system performance for databases. Moreover, you let Itanium max out the die size, it's as big as it can be, at its theoretical limit: you can't literally make a bigger one. If you let me grow my x86 chip to that same size, I am going to boost my performance by 25%, more than enough to swamp what Itanium achieved. So at an architectural level if you keep doing this game and you match everything up, my old crufty x86 architecture beats your shiny new one, and that should scare you to death.

0:25:13 PE: Yes.

0:25:14 BC: I should not able to catch up to Itanium on an even basis. And then finally, he just kind of gave up and said "well why are you bringing this to me, I don't even run the company anymore". I said "you're just the last man in line, [chuckle], I tried these arguments on everybody else". He said "I wish I could help". It wasn't so much that I wanted to walk in and prevail, it wasn't like I wanted anyone to bow down and say Bob said so, let's go do it. I wanted to feel listened. I felt that as the technical guy my job was to the decision-makers the facts. If you're the executive, you get paid the big bucks to make hard decisions, so go ahead and make them, but make sure you make them based on the facts, not just on the half the facts., With Andy you would always feel that he had listened to you. After you make your pitch to him, Andy would repeat back to you what you said, and he'd do it very accurately. Then he might tell you what he was going to do as a result of having heard you. Half the time he might say he'd heard what you said, but because of other reasons he was not going to make any changes, but he would still thank you for bringing him that input. Other times he'd say, "I never considered what you just said, and I'm going to have to think about that some more". Either way I felt like I had done my job here.

0:26:17 PE: Yes.

0:26:20 BC: I added some real value to this.

0:26:20 PE: Yes.

0:26:20 BC: Feel good about it.

0:26:20 PE: Yes.

0:26:21 BC: But that was lost after Andy left. That was lost, that part of the culture went away.

0:26:27 PE: Who succeeded him?

0:26:28 BC: Craig Barrett.

0:26:29 PE: Right. Were you still there when that happened, when did Grove leave?

0:26:34 BC: Grove stopped being the president in January 1998.

0:26:40 PE: Yes.

0:26:41 BC: And that's when Craig Barrett took over.

0:26:43 PE: And what changed at point?

0:26:46 BC: Well Craig's not Andy, I mean he had a different way of thinking and doing things, Craig, I don't want it to sound cynical but I always sound cynical when I talk about him because I had such a bumpy relationship with him. I don't think he felt like he needed anything I could tell him, and it wasn't just me, I wasn't taking this personally. I never once got the same feeling I got with Andy that my inputs were being seriously and politely considered, and then a decision would be made that included my inputs.

0:27:21 PE: Yes.

0:27:22 BC: That never happened. Instead, for example five Intel fellows including me went to visit Craig Barrett in June of 98 with the same Itanium story, that Itanium was not going to be able to deliver what was being promised. The positioning of Itanium relative to the x86 line is wrong, because x86 is going to better than you think and Itanium is going to be worse and they're going to meet in the middle. We're being forced to put a gap in the product lines between Itanium and x86 to try to boost the prospects for Itanium. There's a gap there now that AMD is going to drive a truck through, they're going to, what do you think they're going to hit, they're going to go right after that hole" which in fact they did. It didn't take any deep insight to see all of these things, but Craig essentially got really mad at us, kicked us out of his office and said (and this is a direct quote) "I don't pay you to bring me bad news, I pay you to go make my plans work out".

0:28:22 PE: Gee.

0:28:25 BC: So.

0:28:25 PE: Yeah he's polar opposite.

0:28:26 BC: So and he, and at that point he stood up and walked out and to back of his head. I said, "Well that's just great Craig. You ignored the message and shot the messengers. I'll never be back no matter how strong of a message I've got that you need to hear, I'll never bring it to you now."

0:28:38 PE: Yeah.

0:28:40 BC: It's not rewardable behavior. It was sad, a culture change in the company that was not a good one and there was no way I could fix it. If it had been Andy doing something that I thought was dumb, I'd go and see him and say "Andy what you're doing is dumb", and maybe I'd convince, maybe I wouldn't. But as soon as you close that door, it is a dictatorship. You can't vote the guy out of office anymore, you can't reach him. There's no communication channel.

0:29:09 PE: Just talking a little bit more about that corporate culture at Intel. I am interested in some of the things you talked about in the book, about the aspects of the culture that you tried to establish or enhance, especially the data driven culture that's been obviously a feature of your work all the way along, testing and validation and sort of realistic assessment of how things are doing, where they should go.

0:29:35 BC: You heard my rant against religious designs like yesterday but there are books in this room that date from that era that when I was just starting of myself as maybe I could be a computer designer. I'd read these books and they would recommend that you think this way and then do that and do this and keep this view point, and by implication your designs would be good. None of it was quantitative, it was all high level ethereal rambling and while it wasn't crazy nor irrational, it wasn't the right way to proceed. It might even have been workable had you followed their prescriptions accurately and *also* paid attention to the performance. You'd actually end up with a good design. But the aspect of measuring as you go, just wasn't there. It's kind of you have the faith that God wants you to do it this way and then, as soon as you've adopted that view, you don't have to question it anymore. But as an engineer, I think you *have* to question what you're doing because things are changing out from under you. You have to -- if you start ignoring something, then it's going to become a raging fire. You going to pay for that some time later -- either the product would suffer, the schedule would suffer, or you'd have a bug. Who knows? In no case will it turn out in your favor. I think you have to continually question what you're doing and one way to do that is quantitatively. So you do validation that way although that's a, you have to do that judiciously. That's a sidetrack. You want to talk about that later?

0:31:01 PE: Yeah, let's come back to that.

0:31:03 BC: Let s come back to that. But, so, in the beginning with the P6 effort, you have to picture a small set of people in the closet.

0:31:13 PE: Right, the storage room.

0:31:13 BC: I mention it in the book that we literally met in a storage room doing the early P6 architectural design, because we couldn't find anywhere else that we could routinely meet. We only got into that storage room because Dave Papworth was so creative with jimmying the door lock with his employee ID. This turned out to be a really good thing that we kind of stumbled into. Each day we'd try to reconstruct where we had gotten to the previous day. If the fundamental engine works kind of like this then, what are the challenges, oh well how do we do this and this, and this. And each of us would say, "Ah, well, here are a few ideas that might work. We'd write all those down and keep going until we're too tired. Then the next day we'd come back and each of us was expected to have taken his piece and thought about it deeply.

Typically what you would do is come to that next day's meeting and report, "Okay, I was supposed to look into how we're going to do this function. I wrote a little program to shed light on this issue and here's the data I found. The data was somewhat surprising so then, I went off and I did this research and look what I found." You feed this all to your fellow designers and they all think about it. What happens next is the crucial part. It's not so much that you're trying to convince them that you're right because you're not sure you're right yourself. But you're throwing ideas out there in the space where all you really have are your intuitions. Your intuition is telling you that you came up with five alternatives and this fourth one looks like probably the best one. So you may have a little bit of data to suggest why the fourth one is the right one but typically you don't have a whole lot and the data that you do have may seem more compelling to you than anyone else in the room. So at this point you can go one of two ways (and we tried them both), which is why I became so convinced data driven is best. If you go the wrong way, what happens next is architect A picks alternative number four, but Architect B says his intuition is firmly convinced that alternative 3 is far better.

0:32:59 PE: Right.

0:33:00 BC: This is where it can get messy. A says "Oh, really. Well, I designed a machine like this and it had this in it and I know more about it than you do". B does not take this lying down, so now, it's an ego clash, which quickly makes everyone completely forget the technology issue. What you're trying to prove is you're smarter than I am. Meanwhile, I think my dad can take your dad. You have to stay out of ego clashes. They tear the team apart. They stop forward progress. You end up with a poor design. There's just nothing good about them.

So what we discovered was the best way to proceed is that any time this problem began to rear its ugly head, just before you take the ego step, the team has to recognize this is as far as we can go without stepping into ego land. So now, what we have to do is either go collect real data that convinces all of us that number four is right or we all agreed that there cannot be data that's going to make this decision. We will need an intuition to decide and we're going to resolve it by designating somebody like David or Glenn because they were extremely good at this. So it typically it wasn't too hard to defer to their judgment if yours differed from theirs. I think that was the first part that was good. This, all the things I just described to you came about in about 1991. Prior to that Dave Papworth and I had written a program called DFA which I mention in the book.

0:34:21 PE: Do you want to talk about that?

0:34:22 BC: In that book or that?

0:34:24 PE: The data flow analyzer.

0:34:27 BC: Data flow analyzer, right. I viewed DFA as kind of telescope that you could look around the universe with because you want to know what's in the corners and you've never been there before. You don't know what would happen if you went there. You could use this tool to answer or to at least suggest answers to a lot of questions; otherwise you'd just be lost at sea and guessing, totally guessing. The proof was that we wrote this tool and started using it and then, as I mentioned at the book, typically what would happen is Dave would come in early in the morning saying, "I was up all night playing with DFA. What do you suppose will happen if you had 30 ALUs but only two load store units?”

He'd give you this long scenario. You'd realize right away, he's not asking because he thinks the answer and he wishes he did. He is asking because he was surprised by the answer and he wants to see if you'll be surprised too. (Plus, it’s a fun game.) So of course, with context as a cue, you think well, if I picked the obvious answer I'm going to get nailed, so I'm going to pick something outlandish and hope that I hit it by accident. The point of all this was that we were retuning our intuitions. After you design machines for awhile, you start getting comfortable with the way that machine fundamentally works and that's true of any machine, airplanes, cars, refrigerators, you name it. Whoever designed it will have developed a certain amount of experience with it, and will start feeling pretty comfortable about it. If you asked them "what if you tweak this?" they'll give you a reasonably competent answer. They didn't simulate, they didn't study it but they know how the machine works. So, we had all designed machines before this and so, we were reasonably comfortable with some kinds of machine design.

But this out of order thing was so different that we just weren't sure which of our baggage should be carried over and which of it was just going to be wrong. DFA was instrumental in retuning us and that was the method that Dave used. But again, it was a quantitative “let's measure the answer”; not “let's just not stare at our navels and see who...”

0:36:23 PE: Yeah,

0:36:24 BC: Who has the biggest ego thing.

0:36:24 PE: Yeah.

0:36:26 BC: And that got passed along to the team. Did I say this in the book? I think I might have, that when we had looked at this long enough and we thought that we had a pretty credible story for how the P6 architecture was going to basically be organized. That was almost, that was about the same time when the 960 team arrived and needed something to do. So Fred was still running the show at the time and he and Randy were running things together. And they said, "Well what, why, how can we get...

0:36:57 PE: Randy?

0:36:57 BC: Randy Steck.

0:36:57 PE: Randy Steck.

0:36:59 BC: Because Randy Young, he only lasted six months in 1990 and he was gone. And so Randy Steck and Fred said, "How can we get this big crop of people involved in this project? I know you architect guys aren't done yet." It's not like you have an architecture spec and you're just sitting around, waiting for someone to design a chip out of it. It wasn't like that. We were running like crazy. But the fundamental thrust of it, the fact that we'd be out of order for example; we had decided on that by then. So I said, "Okay. I know what we'll do. It's not like we can have a big document we can give people. We don't have that yet. But I'm happy to sort of do a chalk talk and they could sit in the audience and ask questions and so on and maybe it's a good way to get people going". And this became memorable to me because I spoke for, I don't know probably an hour with my back to the audience, drawing on the board, diagrams of this is what we think and here are some questions that we have.

Basically, we don't know how to do this yet and blah blah blah and finally, I thought okay this is a good starting point and see if there any questions and I kind of turn around and the looks on people's faces were just amazing. It was not like, "Oh my God. You guys are brilliant." It was like, "Holy cow, what did we sign up for, these people don't know anything. You call yourself an architect, but all you have are questions." That's the kind of questions they were asking. “This is not a design you don't know what you're doing yet”. And I said, "Well, we're not finished. That's for sure but we were hoping that since you guys are smart you can help us figure this out." They said they weren’t architects. I said, "Well yes, you are -- that name doesn't mean anything we're just designing a chip here." Historically at Intel, architects and designers were considered separate but if you remember Multiflow we had seven guys. You don't have time to put labels like designer or architect on your forehead.

0:38:34 PE: Yeah.

0:38:34 BC: You just get in there and do what needs doing and if it’s...

0:38:36 PE: Yeah.

0:38:37 BC: ...an architecture job or a validation job or documentation. It's all the same.

0:38:41 PE: So the Intel distinction was between people who sort of like were building design and draw up the blue print for the general plan and then, the designers we're going to implement the pieces of the plan and so get into the details.

0:38:56 BC: Yeah and in fact, I think we still teach kids that way in school. At least we give them the impression that that's the split of responsibilities. But I learned way better than that back at Bell Labs when we were working on the Bellmac32, their 32-bit micro. I had only been working on it for a few months before I left but in those few months, I learned this one key thing. I was supposed to design some piece of that chip and I was struggling with it because I had constraints about schedule and die size and so on and my unit had to do some particular function and it just wasn't working out. I could not find a way to make it do all the things that I thought it needed to do in the space that I had. And then finally, I had this inspiration that the very thing that was finding most difficult about this function Might not even be necessary. I think architecturally it's not possible to invoke the troublesome part.

I figured I'd ask the chip architects about it, and if they confirmed that it's not possible, I'm done. My unit is done. That's the only thing I'm wrestling with. So I went to see the architect thinking this is going to go great. They're going to think I'm brilliant after they see what I'm suggesting. So I tell this guy my story and he does not say, "Hey, you're right. You don't have to do that." He says, "Get out of my office. I am an architect. You're only a chip designer. I don't need to talk to you. I've got other things to do." He honestly said that. And I remember walking out and thinking we are not going to get a good design out of a team that feels this way.

0:40:17 PE: Yeah.

0:40:17 BC: No matter what else. He was so insistent on the split of responsibilities that I thought, man if I've offended him somehow maybe I could understand the hostility but I don't even know this guy. We had never had dealings before. When I got to Multiflow, this was definitely not the culture: you did everything, you are chief cook and bottle washer. Dave Papworth is the same way. When the two of us got to Intel we said, "what, they're going to call us architects but we're just designers like everybody else. It's just that we got started before everyone else did and our responsibilities are in a different area. In the past on Intel's 386, 486 and Pentium, as soon as the architects were "finished" with their part, they were expected to move on to a new design. I made it very clear to my boss at the time that I was not going to do that. Architects are the ones that were there at the very beginning. We're the ones that know where all the bones are buried. If you want to change something in a future derivative chip (which you always will want to do), we're going to be the first guys to know what is safe to do and what isn't. It's a bad way to discover that it wasn't safe by designing a chip, putting it out in the world and then discovering the breaks.

What you really want are people that were there at the beginning who can safely guide you through those kind of thickets. And also because there may be bugs in the chip and we're the folks that know the most about what to do about them, where they might be, and how best to handle them when they are found. We architects also put the microchip patch facility in there, which requires considerable discretion as to when do you play that card, to what extent do you play it. Some bugs you should just suffer with. Because otherwise, you could be chewing up a big fraction of the patch resources for a potentially trivial problem. For all these reasons I kept the architecture team engaged on P6 through production. We did not walk away two years before silicon. I also felt that it would inform the architecture team, and make them better engineers. To my mind, there's nothing more dangerous than having an architecture team that makes all the high level decisions and then walks away from the consequences. They're going to diverge more and more for reality.

0:42:20 PE: Yeah.

0:42:22 BC: I've seen that happen and I didn't want to be any part of it. I thought you get much stronger products by people who stay engaged with it and suffer the consequences of their decisions.

0:42:30 PE: Yeah.

0:42:30 BC: And so that's what we did with P6 but it was, it all came from this original data driven idea that it keeps you out of the ego land and keeps you honest.

0:42:40 PE: Well let's talk about data flow analyzer a little bit. Well actually, no, let's backup a little more. I mean, you talked some about it yesterday and there's quite a bit about it in the book but talking about the decision to use an out of order instruction.

0:42:59 BC: Engine. [laughter]

0:43:02 PE: Right.

0:43:04 BC: Yeah. Well, there's two ways you speed up a chip. Very early in my tenure at Intel, my boss came in and said your job is to beat the Pentium. You have to design an engine that beats the Pentium because if you don't there's no point to doing your chip. We have a Pentium. So I said "Okay, yeah that's how the computer world works but there's two ways to achieve that." Well, there's more than two but here are the two major ways: Do a faster clock, keeping the engine you've already got, but crank the clock up, and boost the compiler so that it generates better code. Especially back then, you could get big wins from the compiler. However, the compiler can be tricky because some of the things that you improve in the compiler will also benefit Pentium. So all the boats rise, and the differential doesn't necessarily help you but still you want it because there might still be a differential relative to your competitors outside the company. So it's something you push. So way #1 was boosting the clock rate along with the compiler, and way #2 was parallelism, how many things the engine is doing per unit time? Arguably, you could do all of these at the same time. And that's what we did but you use different techniques entirely for each of those differently areas. How did I get started, what was the question you asked?

0:44:30 PE: So the decision to make an out of order engines.

0:44:34 BC: Right, okay. The Pentium ended up being two way superscalar, in order. Now, superscalar just means they had two pipelines in which you could do operations. Suppose you had a sequence of add operations you want to do, and each instruction was add, add, add. Further suppose they were independent of one another; no instruction cared what the previous one did. That might happen for example if you had a bitmap, a frame buffer that is going on the screen and you want to increment every pixel. All you are going to do is load, load, load, load, add, add, add, add, store, store, store and they're all independent, so if you had a machine that could do two things at once, you're going to go twice as fast. It's a simple idea right and the fact that an engine is in order does not matter in that case. Pentium had already played the first card of how to go parallel, and 486 had already played a little bit of the clock game. They'd pushed on the clock and gotten good performance out of it having done that. So those two cards were already known, it is pretty obvious to everyone in the field where the knobs were on how to go faster that way. How to do out of order or how to do parallelism best was not nearly so clear. We could see the kind of code that was being generated for the Pentium and the performance that they were getting from it. The problem with being in order superscalar is that you do not have any options. If the next instruction you want to execute shows up in the first pipe, it's easy. Then the next instruction after him wants to go into the second pipe, but what if there is a dependency?

0:46:07 PE: Yeah.

0:46:07 BC: The first one produces something the second one needs, so #2 cannot execute, he has to wait any way and now you've gotten no benefit from your second pipe. That was one aspect of the other things that we mentioned yesterday where you might have a sequence of five or six instructions and then a conditional branch. You cannot employ Josh Fisher's trickiness, borrowing parallelism below that branch to move up here above it. That does not help you here because you do not know whether you should have done it or not, so that is the in-order part of it. They did do branch prediction with the Pentium, so they did manage to say, well look I think this branch is going to go here and there is an instruction there and let's grab that and maybe it can run in parallel. If that did not work, you know, if the branch was mispredicted they had to have a way of recovering from that, undoing the instruction that they borrowed and so on, but it is not out of order in the sense that I can grab that instruction and execute it above one of these other instructions that otherwise would have been done. If you go superscalar in order there are pretty severe limits on how much that's going to buy you, and even then you have to recompile to get usable parallelism typically because you want your compiler to move things apart. If there's that data dependency, you may just need to move instructions apart, and often that can speed you up, because now the otherwise adjacent, dependent instructions can actually run in parallel with this one as opposed to stalling the whole machine and then getting back in order again. An out-of-order engine naturally spaces the instruction executions better and finds other things to do while the functional unit latencies are elapsing. So we looked at what Pentium was doing at the beginning of P6, we wasted no time considering in order three way superscalar or four way superscalar. We talked ourselves out of that prospect in about ten nanoseconds, because there just would not be enough parallelism to make that worthwhile. Also I had spent the summer, one of the tasks I had that summer at Fred Pollack's request was to write up…

0:47:56 PE: In 1990?

0:47:57 PE: 1990, summer of 90, was to do an annotated bibliography of papers and books that I thought were relevant to what we should design next and a lot of the papers being done back then were out of order. They were predicting that there is a lot of parallelism if you can make a machine that can dig the parallelism out. There were papers from Illinois and other places suggesting some possible ways to do out of order so we looked at them and thought okay, one of those might play out. We were very hopeful that out of order was going to be a viable strategy and we committed to it reasonably early, as of September of 1990, that that's what we are going to end up doing. Another question was the right way to map an out of order schema onto an x86 platform because as I mentioned there's all this research going on but none of it was x86, they were all assuming RISC instruction set architectures, so we had an open question as to whether there anything about the x86 instruction set architecture that would prevent those research results from being relevant. Your research might predict a 3x speed up on important code but it's RISC code and if I did it in x86 would if I find out that I will only get 1.2x because of some horrible thing associated with the architecture of the x86?

None of us were x86 experts back then, not me, not Dave, not Glenn, in fact we had all studiously avoided it because it is ugly, it's not a pretty architecture at all. Suddenly we had to climb up a new learning curve for x86 as well as for out of order. After a discussion one day, Dave and I realized that if we had a tool that could track data dependency chains through assembly language, we would be able to characterize x86 dependency patterns and statistics. Once who depends on whom, you can effectively gauge the efficacy of taking later instructions and pretending that they have been executed at the same time as the previous one. That is of course leaving out all the implementation details of how would you do that, what kind of hardware would it take and so on, but we were just asking a question, if you could build such an x86 engine, conceptually how fast could it possible go.

Writing such a code analysis tool turned out to be easy to do: it took Dave and me about two weeks to write this code and then Glen later added a graphical interface, so it looked better, you can make better sense of its results. In effect Dave wrote an entire x86 microcode suite in a just a few days, and while it did not have to be totally accurate, it had to be pretty close and there is nobody better doing something like that than he was, he knocked that thing out so fast. Anyway we got this tool working and then we could do things with it. For instance, we wanted to know, if you had an infinite number of ALUs but nothing else changes in the machine, how fast would the code run. I call that exploring the rim of the universe, you can tell the tool all of the machine assumptions you wanted to make and you would find that you will bottleneck for some other reason. What that tool was teaching us was that if you remove this bottleneck then a new, preferably smaller bottleneck suddenly pops up behind that you didn't see until now, and then you'd look and ask what if I get rid of him too?

Pretty soon you start getting a feel for what's important in this machine for the type of code you're analyzing. Then you start backing away from the universe's rim and tell the tool I do not really have a billion ALUs, let's suppose I have 6, or suppose I have 4, numbers that you can almost actually hit, nowadays of course you can do far better than this but back then it was like you can do 2, Pentium did that, but we think, we can do 3, maybe we can do 4, you know, numbers like that, so you could intensively pick on possible machine architectures using real code. The DFA analyzer would tell you if you do not screw this up in the implementation this is the kind of parallelism you'll get out of it, and so the cool thing was within a few months we were pretty confident that all that stuff that the RISC researchers were saying would also apply to us. There didn't seem to be anything in the x86 architecture that would prevent it. And that was really valuable information to us.

And so after that we were committed to this out of order idea. By the way it was still in the back of my head that while we had not seen a show stopper on this out of order X86 thing, it did not mean there was not one. The possibility still existed that we may not see the showstopper problem until another two years when it is far too late to gracefully react to it. Until we actually got silicon, and ran the operating system, I was determined to keep watching for such a horrible eventuality.

0:52:35 PE: I'm glad you brought up the research literature, because I meant to ask you about that yesterday and I forgot about it. I can see from your CV you know, you've had some contact with professional societies all along and you must have been going to conferences and so on, but, how much did the academic research and computer science influence what you did as a practice in computer architect?

0:53:00 BC: Well, I should first point out that the answer that is laden with dangers. I have been asked to do recommendation letters for people looking for tenure or whatever, for years and years and at this point, I have noticed, I bet I have fifty or sixty of these that I have done. Three weeks ago, I had to get deposed because Wisconsin is suing Intel over a patent that they said Intel should have licensed. Wisconsin accuses Intel of having taken Wisconsin's IP and stuck it in their chips and now Intel owes Wisconsin a whole bunch of money. Wisconsin is citing as evidence, exhibit A my recommendation letters for that professor who has his name on the patent. I did the deposition prep and the deposition for free because I was so angry that they had misused my recommendation letters in that way. and now yesterday they called me after you left and said the trials going to start on October 5th and they are thinking of calling me as a witness and would I be willing to come out to Wisconsin and do that and I said, "Yep and you have to cover my expenses, but when they ask me what I am getting paid on the stand, it is going to be zero because I am really mad about this, it ruined the entire system".

0:54:13 PE: Yes.

0:54:16 BC: Why would I ever write another recommendation letter? when I wrote those letters they told me they would be private, that the professor would never see it and so on and how those things work, you've probably done them: you have to be flowery but not inaccurate.

0:54:27 PE: Right.

0:54:29 BC: But when you see those words up on a screen it looks like you are saying he invented everything ever known to man.

0:54:36 PE: There is a very strict protocol you know, especially the best institutions we call them walk on water letters.

0:54:41 BC: Yeah exactly.

0:54:42 PE: You have to say that this person is perfect and.

0:54:44 BC: Because if you don't, you are saying they suck in the secret code of recommendations. But that is not what a jury is going to see so these guys are basically going to use this to try to extort a lot of money out of Intel and I was just really mad about that and still am.

0:54:55 PE: Right.

0:54:56 BC: Anyway how did I get onto that?

0:54:58 PE: So, I was asking about how you used this literature?

0:54:59 BC: Oh yeah.

0:55:00 PE: And how much contact do you have with it.

0:55:02 BC: The problem is, this has happened more than once where I have been questioned very carefully about "what do you think the contribution of professor XYZ was to Intel's bottom line?" and so on, and I'd say, you know, it is extremely hard to answer that. On the one hand these people are my friends, these people work for research at schools, but there is subtlety that is very hard for most people to grasp. It is that once the first country has designed an atomic bomb and set one off, it is far easier for the next country to do it, even if they do not have any spies or anything else. It just how it works, if that you can be made to work...

0:55:37 PE: It can be done so.

0:55:39 BC: You're way ahead of the game. And in that sense the people that did out of order research in the eighties, they helped us a lot just indirectly by knowing there appears to be a way to make this work. Now, knowing that let us concentrate on the details rather than constantly be in this fear of oh I do not think it is going to work at all. Without that, then you split your head in half and you do not have enough neurons working on the second problem. So I readily grabbed that part of it because the out of order researchers were there ahead of us. We should also say that the 360/91 from IBM in the 1960s was also out of order, it was the first one and it was not academic, that was a real machine. Incidentally that is one of the reasons that we picked certain terms that we used for the insides of the P6, like the reservation station that came straight out of the 360/91.

At one point NCR was a little panicky about whether they should really commit to Intel because what this P6 chip might not work. They hired Mike Flynn from Stanford to come and spend a day with us.I was presenting to him and suddenly Mike Flynn got really agitated and he stopped and he said, I do not like the fact that you guys have started with terms, that you do not even mean the same thing by them.These people were here ahead of you and this isn't fair, you are not showing respect for your ancestors.I said, well Mike, we named it specifically that to try to give honor where it's due, we know we did not invent this topic, that is the first time I ever saw it was the 360/91, so I used its name even though I know it is not quite exactly the same. He said, Oh, okay (laughs).He really felt strongly about it. Now that the shoe is on the other foot, because it has been like 19 years since we did all this and when people use the same terms we did, I get a kick out of it because they should do that.

0:57:29 PE: Yes, okay.

0:57:30 BC: Anyway I got diverted again. Here is a specific example of where it gets very tricky. And this is a real example, there was a paper submitted to Micro in about 91 or 92, I was the reviewer of this paper and the paper was on branch prediction. I read this paper and its authors called their creation two level adaptive branch prediction. And no one had ever done this two level adaptor thing before that I knew of. I looked at it and thought it was cool. Because at that time that we had gotten to the point where we knew we are going to go out of order, we knew that the pipeline was going to be deep, and if you've got a deep pipeline, you'd better branch predict accurately. Otherwise you are throwing away that whole pipeline every time you do mispredict.

So I read this paper and thought there might be something in here that we could try to use to make a better branch predictor. We started working on that right away and the other thing we did was to hire the grad student co-author as a summer employee.Things got weird after this because what happened was this paper was for an in order machine and so if you think about what that means with a branch predictor, you hit a branch, you predict it, if you mispredicted it, you want to change what's in that table because it told you to do the wrong thing.If you keep doing it wrong you want to flip it the other way and so there's mechanisms in there to handle that, but with a speculative machine, for all you know, you could be executing data, because a speculative machine can be executing complete garbage due to the front end of the machine fetching instructions from wherever it thinks is right yet could be totally wrong. It could be pulling instructions right out of the garbage can and when it does that some of them are going to be branches. When some of those branches access your branch prediction tables, if you allow those branches to change what is in there only to later find that it was a bogus branch and shouldn't have never been hit, now you have lost the state that should have been in the branch predictor, so it is quite tricky to figure out, what's the right way to handle speculative branches in an out of order machine. We started with the basic ideas that were in that paper and then we modified them very heavily to account for, what an out of order machine has to do.

Unfortunately the grad student that was with us, I think we taught him a lot more about how to do this than he taught us about publishing his paper. Yet the next thing we see is a paper coming from his school with our stuff in it, which drives industry people nuts because they feel this is their professional career and their idea, and it might be the best one they ever had and someone else's name is one it. So it's quite frustrating, and we kind of bounced back and forth with these guys for a while about what we were going to do. It led to some bad feelings for several years. I think it's indicative on how difficult it is to merge successful academic research with the needs of a product line.

And then there's other things. I accidentally offended a whole bunch of academics once when I was on a committee in Princeton, looking into I don't even know what it was anymore, and at some point one of them asked me, point blank, "To what extent is the research that we've been doing for the last few years directly affecting your product line", and I said, "frankly, not very much. The designers do not read those conference proceedings. They invent their way around problems. You know, as well as I do, for every problem that you face there may two or three reasonable ways to answer it. Not a thousand. There's a thousand bad ways but just two or three really decent ways. So, if you picked one in your paper and we picked one and it happens to be the same, then the odds are pretty good that it was coincidental. It doesn't mean we stole your idea; it doesn't mean we wouldn't have been able to do it if you hadn't come along; it's just the way it is. I said that the architects tend to read the conferences for future direction kinds of ideas, not to solve specific problems. They really hated that answer. They felt I just dissed their whole field. [chuckle].

There are times when I think we would have been better off paying more attention on things that have been done in the literature, but in general, I don't know. We did license a branch predictor, we did take a private license for one of the things, to one individual somewhere who had come to us with the proposal and was no longer working at whatever big company he used to work at and he said, "I got this cool idea that I've had since I left that place and it might benefit you and here's the price" and we looked at it and said, "hey that's not bad". We paid his price and put it in. But it wasn't one of those, you know, conference proceeding kind of things; that just doesn't happen very often. At least not directly.

1:03:04 PE: So would you say, how often did you go to professional conferences? What were the main professional societies that you would go?

1:03:13 BC: Well, I've only ever been a member of IEEE. I've never joined ACM, because my wife's a member of ACM. So, she gets all the proceedings so I just read hers. I've gone to maybe two conferences a year for 30 years typically, sometimes more. If I get invited to be on a panel session or do a keynote, I'd go to those. And I've been to conferences that seem kind of far afield; for instance, I gave a key note at an Asynchronous Circuits Conference once, and I'm not an expert on Asynchronous Circuits.

I gave one on formal verification, because we had done a fair amount of that in Pentium 4, I don't know much about it, but I could say as the project manager, here's what I wanted to accomplish, here's what I had spent, here's what I think I got back for it. It was kind of fun. One of the ways I try to stay current is by accepting program committee roles, because then you have to read the papers. It's a hell of a lot of work, as you know, to stay current, the field's not sitting still, it's moving. So, if you want to keep your finger on the pulse of where people are heading and what it means, that's one way to do it. I just did that this year, I was on an ISCA and Micro committee and then I had to travel for both of those to go to the meeting.

1:04:38 PE: ISCA is International Society of Computer Architects?

1:04:42 BC: Close. International Symposium of Computer Architecture. ISCA and Micro are the premier architecture conferences. Typically most of the same people come to both of those anyway.

1:05:02 PE: Okay, so we talked about the out of order engine, then I wanted to hear a little bit more about the data flow analyzer and exactly what it did. It's a simple program, it's basically tracing what happens when you have instruction with the output of that, where does it go next and there's quite a bit going on in these things.

1:05:28 BC: Yeah, there is. But, that's the trick, to ask the right question of the analyzer. If you couch it the right way, it will give you a useful answer. If you don't, it'll give you garbage, it's up to you to notice the difference. In fact, that became a real issue We had such a good time with DFA on the P6.When I write software it works, but I make no bones about it being pretty. In some cases, people who are really good at software will look at my code and just laugh. That literally happened with DFA because I had written all of the command line parsing routines, to handle the invocation switches. I wrote all the codes on how to find those switches and what they meant.

This one guy, whom I really respect a lot for software, he looked at my code, and he said "you do realize there are existing libraries to handle those string manipulations", and I said, "I do now". Dave and I got this thing working, we used it for a few years, and many architects modified it along the way. They wanted to ask a question and DFA wasn't quite suited for it by default, so they'd go in and change the DFA source and then run it. We all got real good at that and, for some reason, I think the way Dave and I wrote the code made sense to the other architects in a way that a professional engineer writing software like that might not have; they might have had to climb a learning curve to see what he or she was doing. This became relevant because after we had such a good experience with DFA on P6, we tried to modify it to use on the next chip, the Pentium 4.

This time around, I figured I'd better get myself a real software engineer to maintain DFA and extend it, put some real professionalism into it. And the guy that we got to do that went off and did exactly that, and he made crude comments to us about our original code; he was just kidding, but it was partly true. But, the problem is that he took it over, when we started using it on the Pentium 4.We called the new project Willamette at the time, it was the code name for the Pentium 4.

1:07:45 PE: Willamette, like the, this is just for the recording, Willamette like the town in Oregon?

1:07:51 BC: Yeah, the river downtown. That's the one, the river, you were in the Marriott right? It's the Willamette River in front of you.

1:07:57 PE: Yeah, yeah, right.

1:07:59 BC: So, we started applying this software engineer's version of DFA to Pentium 4 and at that point, we could no longer just grab the source code and modify it if we felt like it, that's not how the pros do software. If you're a professional, you check it out of a revision control database, and there's other things you do. He had put a lot of structure to the code that, I'm sure, corresponds to what you're taught in school, how to write good code, but as an architect, you got to look at it and think “I don't know what this is in here for. I don't know why this is here”.It started injecting a lot of doubt into our minds. And then we realized, after enough attempts that the new DFA was not giving us useful information about the Willamette project nearly to the same extent that it had done on P6. There were just too many differences in the microarchitectures, there were dozens of areas where the DFA approach just can't be used, it doesn't match the way the machine is going to work, and there's nothing you can do about it. Those areas were becoming more and more important.

1:08:55 PE: Because it was a different machine or because of the way the DFA was built?

1:09:00 BC: It's both. The DFA knew from the instruction's trace which way a branch should have gone. But because it was executing or it was using the trace as the input, when it hit a branch, it would only know the way it was supposed to go, which of course is the way that it actually did go. It didn't know what lay down the other path. So, we had some heuristics in there that said, "Well suppose a branch mis-prediction had happened there; we'll take the following default performance hit". It did not, in fact, happen in the actual trace when it was collected, but suppose it had because the Willamette engine was such that it would have messed that one up, we would have gone over here, what's over here? What's that code? That code in a P6 would have done a certain thing and then gotten flushed. It wouldn't have lived very long and then it would have gotten detected and flushed. But in a Pentium 4, the mispredicted bogus code could actually make quite a mess because there's a much deeper pipeline, a lot more parallelism. We needed to know what that bogus code was and it was very awkward to find out what it was. To find it out, you would have to take the actual binary, combine it with a trace and go look up in the binary what the code would have been and it was very awkward. So, instead, we kept living with the heuristics, but we learned to become very skeptical about certain results. After awhile you could tell if the trace had certain characteristics that would make DFA results unreliable. You just had to know that. So, in effect, our earlier success kind of set us up for a later failure. We probably should have junked DFA, started over and done something else entirely.

1:10:37 PE: If I understand it correctly, the DFA was the first tool of that type that the Intel had ever had.

1:10:43 BC: Right.

1:10:44 PE: And so this is part of the data driven culture concept, because what you're looking at with the outputs of the Micro code, it seemed what it really does so you can correct it. But you guys mainly just used it within your group and nobody had a, it didn't travel beyond it.

1:11:07 BC: Actually, I think we did send it, I know I gave a talk on it to the Santa Clara guys because I got feedback from my boss later that they had really enjoyed it. I mean, they'd have to go out of their way to tell him, "Colwell came down and gave us a pitch, we loved it". DFA is a great tool. So, those guys all had access to it, but it was really a fairly P6-specific tool, so the Itanium team had no use for it. It was mostly the proliferation teams, like the one in Folsom, California or the Israelis, they loved it.They weren't designing any P6s at the time, but they wanted to. So they were setting themselves up for that. So, it did get some usage in other places, I just don't know what extent or what they got out of it. There are two other aspects I wanted to bring up; one is that you have a lot of the more interesting questions or decisions that one has to make as a design team.

You have 20 different things you have to consider, all of which influence the decision, no one of which is overriding the other ones, but taken as a whole you have to consider all of them. And maybe 16 of them are quantifiable, schedule impact, die size, performance; you can quantify those pretty accurately. But some of them aren't so measurable, like risk, complexity, and whether the team likes an idea or not, because if they don't like it, they're going to fight you, they're not going to put their heart into it and things are going to go south after that.The tricky part for me always was you want to avoid the syndrome that says I got 20 things but I'm going to only base it on the 16 I can measure, I'm going to forget the other ones because I can't quantify those. That's one syndrome you definitely want to avoid. And then the other one is related to that: you want to avoid the follow on design that says I'm not only going to base it on the 16, I'm going to set up an arbitrary numerical threshold in such a way that the decision makes itself once I've measured all the 16. My boss' boss really wanted that badly.

He really wanted it because he said, "Bob, how do when it's time to tape out the chip?" And I said, "I don't. I'll just tell you when it’s ready". He said, "Not good enough, I want you to setup an objective criteria early in the project that says when all of these things have been numerically measured and have exceeded their required threshold, boom, tape the part out". And I said, "I will not sign up for that. Not now, not ever. And anyone who tells you that you should do it that way, you should fire them, they're stupid”. Because it doesn't work like that. As the project goes along, you learn a lot and to pretend that on day one enough to accurately predict something that complicated, is bananas. Instead, what you have to do is take your best shot and make a flexible decision, because you're going to change it.Validation is the classic case. We mentioned that yesterday, we start out with an idea, when I've done all these thousands of tests, surely that's enough to prove it's ready to go. But instead, what happens is you don't get all the thousand done because some of them turn out to be way harder than you thought. In running some of them, you realize there are whole areas of the chip you forgot in the test plan, so you have to add on many thousands more.

Sometimes you get wrong answers. Sometimes the validation test itself is flawed, one way or the other. You better take that into account. And then you realize, not all the different things in the chip are equally important. For example, you want to make sure that the integer adder adds 2 and 2 and gets 4. But what, if it doesn't, you don't need a validation test, because nothing else will work. If you can't add, everything fails and you'll find that, there's no threat there. The threats are in the complicated corner cases where there's state involved like cache coherence issues.Processor A writes to memory which affects processor B's cache. Processor B unfortunately had a bug and he modifies cache in a slightly wrong way. Now, 10 million clock cycles later B gets around to trying to use that data and it feeds into something else that then fails. Unroll that and figure out what the causality chain was.

It's nightmarish and it's all because there's deeply hidden state that can stick around for a long time. And then when you go to debug it you have to have a tool that is going to have all 10 million cycles in it so that you can access what you need to understand the problem.

1:15:38 PE: Yeah., yeah.

1:15:41 BC: It's quite tricky. So those are where the threats are and you realize where those corner cases are better and better as you go down the development change during the project.

1:15:49 PE: This brings up that there are a couple of terms that you've been using just now that I noticed also in the book that I want to ask you about just whether the origins of them just because I'm curious. What is tape out? So you, something that you do you when you send it production, I assume.

1:16:05 BC: Yeah.

1:16:07 PE: But why is it called taping out?

1:16:05 BC: Picture a bad B grade science fiction movie with a computer in it. They've also got the big mag tapes.

1:16:14 PE: Yeah, yeah, the tape drives.

1:16:15 BC: Exactly. In fact, you were telling me a story yesterday about tape drives. Well, those tape drives used to be the *only* physical medium that could handle the entire database for a chip.

1:16:26 PE: I see.

1:16:26 BC: So you literally stuck everything that the Fab needed to know on that tape and you carried that tape over to them and gave it to them. There's no network. There's no email. There was no nothing. You give them a tape.

1:16:35 PE: Yeah.

1:16:37 BC: So yeah, that term still sticking around even though the kids nowadays designing chips don't know what a tape is.

1:16:42 PE: Yeah.

1:16:43 BC: They've never seen one.

1:16:43 PE: Yeah. Well, this is another term that's like that but still maybe still around I don't know this might be gone now, it's core dump which of course kind of came from dumping the magnetic cores.

1:16:54 BC: Magnetic cores, right.

1:16:56 PE: Which are, long, long gone.

1:16:55 BC: Yeah.

1:16:57 PE: But when people still talk about it they think about as meaning like the center of the computer, the core of a computer in a different way.

1:17:03 BC: Yeah.

1:17:05 PE: Still makes sense but it's really the original of the term is different than people realize.

1:17:09 BC: Yep. I just read an the account of this the other day in the ACM journal of some sort where one of the old fogies was saying that one of the new guys was stuck and he had a real problem, a system problem where his program kept crashing. He didn't know what to do and old fogy said just do a core dump and see what the bones will tell you and the kid said "huh what does that mean?. How do you do it?"

1:17:29 PE: Yeah.

1:17:29 BC: So he taught him how you could do this to get the machine to show you what the memory looked like...

1:17:35 PE: Yeah.

1:17:35 BC...just before the mistake and the kid was suitably impressed.

1:17:39 PE: Yeah, yeah. The other term is corner case.

1:17:42 BC: Oh yeah. That's common usage. It's not unique to me. Where it comes from is if I sell you a chip, I give you a datasheet for it. And the datasheet will tell you things like stay in this temperature range, stay in this voltage range, stay in this frequency range...

1:18:00 PE: Okay.

1:18:01 BC: If you stay in all of those ranges that I tell you, I guarantee you the chip will behave as predicted by the spreadsheet. If all of those values are nominal you can just take one of them and move it back and forth, up or down to its max or min. It needs to still work but that's actually fairly easy to do. What's hard to do is when you take one to the max and a different one to its min. Picture it in N-dimensional space and you're wondering if the...

1:18:26 PE: ______

1:18:26 BC: That's it if the corners are those spaces. That's where the horrors lay.

1:18:32 PE: Okay. So one more key decision that you made in P6 project was to stop supporting the 16bit code.

1:18:49 BC: Well, not quite. I mean, we just still ran 16 bit code....

1:18:53 PE: Right.

1:18:53 BC: Successfully.

1:18:53 PE: Yeah, right.

1:18:54 BC: But we chose not to make it go faster. It wasn't our emphasis. We purposely chose to make 32bit code our target and 16bit would do what it did.

1:19:04 PE: It sounds like that was controversial within the company.

1:19:08 BC: Sort of but after the fact. I mean when we first made the decision, it didn't seem, no one was real upset about it. Maybe they weren't paying attention. It was only later when the chip came back and we were running it, and selling it that anybody brought it up. Keep in mind of course, the first P6 was intended as a server part.

1:19:29 PE: Right.

1:19:29 BC: It was supposed to be workstations and servers, with two die in a package. Too expensive to put on a desktop or a mobile; those were not what we had in mind. We wanted the first one to be suitable for servers and work stations. Guess what does not get run in servers and workstations: 16-bit code. You can recompile and you run these high level apps and they don't care about the old legacy stuff. No one is going to run an old game on a server so it all made sense to us and we figured well, for the first generation we won't emphasize 16-bit code. And for the second generation, the Pentium 2, we can easily throw some stuff in. We know what to do. It's not a mystery how to speed it up. It's just that on our first P6 generation we didn't have the time and we didn't have the transistors. But when certain people became aware that we had not emphasized 16-bit code and they realized that it meant that our 200 megahertz part was running the 16bit code at the same speed as a Pentium that was running at 100, they took that as evidence the design team was, negligent or something.

And some people got really exercised about it both inside the company and outside. I don't know if I mentioned this in the book but one of the industry analysts, I won't say who it was, came to Intel ostensibly to interview me about P6 but I don't think that's what he had in mind. I think he just wanted to yell at me, because he stood up and started marching around the room in a circle around me, shrieking. He was yelling how could you be so incompetent as a chip architect to not know that you have to speed up 16-bit code. You have screwed up the entire industry. We are all going to go down in flames all because you are too damned stupid to do your job. He went on and on like this. I gave him a few minutes and I finally said, "Hey, are we done? You're obviously not looking for input from me and you're not telling me anything I find interesting so let's just break this off," and he just did a couple of harrumphs and sat down.

1:21:29 PE: Wow.

1:21:29 BC: And that was the end of it. At a dinner talk for Michael Slater, the first time P6 was ever mentioned in public, I did the talk. That same guy in Q&A session puts his hand up and says, "Do you really want us to believe that you have three independent instruction decoders on that chip?" And I said, "Are you suggesting I would lie about that? I showed you my floor plan. I showed you where the decoders are; what do you want, a note from my mom?" This was shocking to me. This guy is just a piece of work.People get exercised about the craziest stuff and of course, this problem disappeared after a few years, when Pentium II nailed it. And 16-bit code did exactly what we said, and fell out of favor. Who cares? But to have crippled the future of the product line for old legacy code just would have made no sense and I was not going to do it. And heck, any decision you make, someone is going to be mad at it. It's just inevitable. No one's always happy. I just finished reading a book about Lincoln that the, "Team of Rivals." Have you read that book?

1:22:34 PE: No.

1:22:34 BC: Oh, my gosh, I could not believe it. Lincoln was getting pestered constantly and no matter what he did, there were people saying this is the stupidest man on the planet. They literally published opinions that he was an ape, a primate. I mean they said incredible things about him, but the reality was he was a people and management genius. Unbelievable how effective he was at dealing with people. And one of his classic things was even people that were purposely trying to screw him over, he would forgive that. He would just forgive it. He was very magnanimous. It was called "Team of Rivals" because when he put his cabinet together, he picked those people even though all of them were his political rivals.

1:23:10 PE: Yeah.

1:23:11 BC: And everyone of them initially thought he was stupid.

1:23:13 PE: Yeah.

1:23:14 BC: That everyone thought they could control the presidency through him because they had the misperception that he was limp and weak. After he'd been running the country for awhile everyone of them understood Oh man, this guy is way beyond us, but that the fact that he had the greatness of character to put a cabinet together composed of his rivals was unbelievable. I look at that and think if I tried to do that inside of Intel, oh God life is too short….

1:23:39 PE: Yeah, that's a risky strategy. It works for some people but...

1:23:44 BC: It worked for him. I mean there are people that are truly evil but he was picking people because they were really good at something important. Like his work secretary Stanton, that guy had to basically get the generals all to do the right thing and the North started out with terrible generals.

1:24:01 PE: Yeah.

1:24:01 BC: Just terrible.

1:24:02 PE: Yeah.

1:24:02 BC: They were incredibly bad and it was only until later that Ulysses Grant showed up that the North started doing better.

1:24:09 PE: Yeah

1:24:10 BC: But Stanton thought Lincoln was a complete joke at the beginning, big joke. And only later on he became one of his best friends because he realized Lincoln was incredible. So, all the same personal dynamics and animosities, they all play out in companies too.

1:24:26 PE: This brings up another thing. I just detected this in a lot of your writing that you must read quite a bit of [1:24:32]___.

1:24:33 BC: I've read all of his books and I have upstairs another set just like this.

1:24:37 PE: And I've seen your office with two walls of bookcases and four shelves high on each wall, four book shelves high on each wall.

1:24:46 BC: I'm just interested in everything. Some of them, the ones that I think are most relevant to designing computers are over here and the stuff that's like the war in Iraq or in Afghanistan, or woodworking or guitar and so on is on the other side. So that's the fun stuff and this is the necessary stuff.

1:25:05 PE: I see that I have quite a few of these books myself.

1:25:07 BC: These are the classics. The ones that I don't like so much are upstairs and if I really don't like them I get rid of them.

1:25:14 PE: One of the ones I guess but you really didn't say so exactly is the Diane Vaughan book on the "Challenger Launch Decision."

1:25:20 BC: The Challenger launch decision. It was enthralling.

1:25:24 PE: The kind of thing that you were doing.

1:25:26 BC: Yeah, exactly.

1:25:26 PE: Being complicated organization and...

1:25:28 BC: Decision making process when a lot's at stake, yeah.

1:25:29 PE: Under lots of pressure for time and schedule, and yeah

1:25:34 BC: Yeah, I found that book amazing.

1:25:37 PE: It is amazing.

1:25:37 BC: Scary too but.

1:25:39 PE: Yeah.

1:25:41 BC: we faced similar questions (except the life and death part). Let's suppose you find a bug in a chip that is in production. The first question is how serious is this bug? Who does it affect, how bad is it, do I have to recall the silicon, or can I work around it? Who do I have to notify? And then, once you were through all those, you must consider the implications of this bug, are there other bugs like it, where should we take our validation engine now and refocus it on this thing to hit the area all around this bug and make sure there wasn't some nest of bugs like it that that we didn't think of. It kicks a huge effort into motion and lot of it is people having to make decisions with only partial knowledge.

1:26:25 PE: Yeah.

1:26:25 BC: I've come to think, and at the time I thought, this is really awkward. I wish I didn't have to do this. I think there's no one out there that has perfect knowledge ever.

1:26:31 PE: Yeah.

1:26:33 BC: That's all we ever do is make decisions with partial knowledge. The question is, how do you have *enough* knowledge? This is a cultural thing that I also saw at Intel that I did not like. In the early days, it seemed to me that Intel was much better at saying we need an answer to the following question. We don't need to make it right this minute if in fact we'll get a much better answer after three days of thinking and checking and asking the right people. So we're going to wait those three days. Later on, towards the end of the 90s and I don't know maybe today, it became, of all the people in the room who has the strongest opinion on this.

1:27:07 PE: Yeah.

1:27:07 BC: Whoever that is we're going to do what you say.

1:27:09 PE: Yeah.

1:27:09 BC: And it became an article of machismo that I don't need to check with my technical folks by golly I'm going to make the call right now. I sat in more than one such meeting thinking wait a minute, the question you just asked is in my area, I know way more about it than you do, and I don't think I can make the call right now. I think the right thing to do is ask the right people and two days from now make the call.Meanwhile while I'm thinking this, someone else already answered and we've moved on.

1:27:36 PE: Yeah.

1:27:38 BC: I think that's ridiculous and worse, this is going to sound like a joke but it's true. I swear it's true. I sat in on a staff meeting more than once where a technical issue would come up like what do we do about this bug or what about do we put this feature on the chip.

And you look around the room and you see 20, 25 people in it, three of whom are HR, two are lawyers, there are people that just have no technical connection at all and the meeting leader asked for a vote. I swear he really did.

1:28:03 PE: Right, right.

1:28:04 BC: He asked for a vote. The hands are going up and I'm thinking my vote is just as good as the HR person's. Well how did this happen? I don't think he literally said "Well, 15 aye's and 7 no's and the motions carried, but it sure felt like that. And I just thought this has got to be the worst decision process anyone ever thought of. If it's a technical issue then let the technical people deal with it, come on. Decision processes I find fascinating just because of this partial knowledge aspect.

1:28:34 PE: Yeah.

1:28:35 BC: How do you have enough information?

1:28:37 PE: Well so let's talk now about your path kind of from the small initial core group of architects to getting more and more people involved in the design process. What was that like for you and how did you I mean you talked already about you kind of fell into the role of being the person who glues everything together and the rest of the process? But there's another step from doing that with the small group to kind of managing and organizing a much larger group.

1:29:08 BC: In terms of the management of that large group of people, it was Randy Steck who did a great job of figuring out what needed to be done, handling the people in the right way so that they stayed engaged with the project and didn't give up, or disenfranchised. I mostly ran my part of P6 with on the order of 40, 50 people. Only towards the end I inherited some architecture and validation folks in Folsom who had been working on Pentium III that the number ballooned up to the skies.

At least under my command, I insisted that validation be part of the architecture group, not part of the design group because I wanted there to be a creative tension between the people testing and the people designing. One of the syndromes that I think you want to avoid (and people fall into it all the time) is, if you embed the validators into the team with the designers, it's but a short step from there to having the validators become extra designers, and then nobody does the validation. The following thing is going to happen and I guarantee it. The designer is going to say or the validator will say, when are you going to have this design done because I'm suppose to start testing it or I am going to start falling behind my schedule. And the designers are going to say, what pal I don't have a problem, that's your problem. I've got to design them first before you can test anything and I am behind the gun. Hey why don't you help me, you know, if you design this part of it, I will do this part of it, you do that part, we will get it done. And the validators going, oh this is an honor, this is great, I think I can do this, I am going to show my stripes. So they will sign up and do it. Now of course once the units been designed they own a part of it.

1:30:40 PE: All right.

1:30:40 BC: They are not going to test it as well.

1:30:42 PE: Yeah.

1:30:42 BC: It is very hard to attack your own babies, and that's what you got to feel like you are doing when you are validating. You have to feel threatened to your very core that there is bug in there and you are going to find it. If you are the designer then you created it. I have met very few people capable of pulling off right sort of emotional distancing needed. So I think it is much healthier to have a separate team that doesn't have an emotional stake in the design side and attacks vigorously. If it survives that attack the product's in much better shape. Doesn't feel good as a designer but I think that's the best way to get to a good product. Anyway that why we put the validators on my side and the point was they became a separate unit of 20 or 30 people with a self sustaining mission.

I made sure they had the connections they needed to succeed. But they didn't need a heck of a lot of management from me or supervision once they were on the right path. At the beginning, they did need some input from me. When I first took over from Fred, the very first thing I had to face within a week was that the validation crew was not getting the job done. They were not cranking the cycles we expected, and they weren't finding bugs like I expected. Their plan was to wait to until the entire chip was designed and then wield their battery of tests they would have by then developed. I told them that plan was not going to work, we need incremental testing as we go along so we can incrementally tell the design team what they are doing well, and what they need to fix. If you find something serious, I need to know that now, I don't want to find out three months from now. Their answer was, that's just not our plan. I said well, it's your *new* plan. [Chuckles]

And then I started noticing that a lot of the people on the team just weren't capable of doing that, they weren't capable of operating that way. So I jumped in and discovered that the guy who was running the validation team had a wrong attitude of what his task was. He was hiring people to meet his hiring quota. You know, he had 15 hiring requisitions ("reqs") so he hired 15 people. It was not in his lexicon to say well I filled 13 of them but I don't like the candidates for the other two, I am just going to hold onto those reqs until someone good comes along. So we ended up with a dysfunctional validation team.

1:32:51 PE: Okay.

1:32:52 BC: So first I canned that supervisor and then I had to can some of the people in his organization, and start over.

1:32:57 PE: Okay.

1:32:58 BC: That was a strange introduction to management [chuckles]. I wasn't sure I was doing the right thing, but I really didn’t see an alternative. And in retrospect I was, but I didn't know that, I had never done management before.

1:33:09 PE: Yeah.

1:33:10 BC: I just knew I didn't like the answers I was getting. So, again it's partial knowledge but as for running a large team...I used to also say that Randy Steck knew how to manage, I know how to lead. I didn't call what I did ‘management’, I call it leadership.

1:33:25 PE: Interesting.

1:33:25 BC: I think it's a lot easier to lead than to manage in some ways. Although I guess if you don't have the gene for leadership you can't...I don't know if you can develop it. You can develop management skills by studying it or being mentored by an expert like Randy Steck, but the distinction I drew was, when I had like staff meeting, it was not my style to say "look guys, I thought about this, I want you guys do this, you do this, you do this, you do this, any questions? No, let's go." That wasn't my style. Instead I’d say "guys, this is what we are up against, this is where we are, this is where we are trying to go. These are the problems that we are facing right now, what do you think we should do?" and I’d collect input from these guys, compare my thoughts with theirs, and go from there. I wanted them to think things through, because they often came up with ideas I hadn't come up with. I never wanted the case where I would say " go do this” and have them think to themselves, “I thought my idea was better, but okay, you are the boss" no, I hate that. If these people are smart enough to get the job done, they are smart enough do it in the way that makes sense to them. It meant my management overhead was a lot less because those guys were good. They knew what they were doing, so what I did was mostly stay out of the way, and run air cover for them sometimes. So, P6 was blessed. Pentium 4 was much more difficult for me because the team had gotten way bigger and...

1:34:40 PE: How big was it?

1:34:41 BC: The architecture team, when we inherited the Pentium III guys it hit 100+. Just in the architecture effort alone for the Pentium 4 we had probably 30 guys as opposed to the handful who provided that basic microarchitecture of the P6. And that was mostly a reflection of the complexity of the design, because Pentium 4 was way more complicated than P6. I mentioned air cover; I only had to run air cover occasionally for P6, but I had to run it all the time on Pentium 4 because we were just constantly having friction with other corporate groups, especially Itanium. For example, we were being hounded to make our Pentium 4 x86 platform front side bus interconnections compatible with the upcoming Itanium chips. They thought that that would ease the impedance match of the Itanium into the product field.

1:35:32 PE: So Itanium could put into plus or the, you know...?

1:35:36 BC: Into our board, and the bus that's on the board. My reaction was that if Pentium 4 and Itanium were plug-compatible, then one or both of our design teams has made a grievous error. Because the Itanium is supposed to be a high-end server part where performance is everything and cost is secondary, and I know for a fact that my part is constrained by cost so it can not have as much bandwidth with the bus as I otherwise would have taken. Therefore we can't do this compatibility idea,...one size won't fit both. I fought them and fought then. At the time I prevailed, even though I got beaten up repeatedly over that issue especially by the Itanium guys who felt that I was just being vindictive about their product. It was still that way when I left. Nowadays both of those product lines are on the same bus. As soon as I left they went ahead and did it. Now I don't know what the impact was, I don't if my dire predictions came to pass. I just know that eventually the resistance collapsed and they went and did it. My style is technical leadership and I think architects responded to that whole lot better. One classic example, at one point in the late 90s I forget it was Craig's initiatives or Andy's or whose it might just been, no, it was Albert. Word came down from on high that somebody high up was not happy with the amount of effort being expended by the company as a whole. They felt that we had strayed from Intel values. And what they actually meant was if you show up at 8 o'clock there needs to be enough cars in the parking lot.

1:37:08 PE: I've heard the story [chuckles] it is interesting.

1:37:11 BC: And so the word came down, that we are supposed to tell our troops that you got to be here by 8am and you have to be here through 5pm, and before that or after that it is up to you what you do. But if you can't manage those we need to have a written explanation why that's going to be. And when I was in the staff meeting where I was told we had to pass that message on, I said I refuse to do that. That's a bad idea, and I am not going to do it. My upper management informed me that I didn’t actually have choice: if you work here as a manager you are expected to reflect the messages that you are asked to by your management team. I said yes, and that’s why I am not a manager, I am an Intel fellow and I am a leader and I reject that. And by the way I am counseling you, upper management, that that's a bad idea too. Albert said, I am on the management team too and I have to say it, he said I don't get to turn the message down so neither do you.

I said well okay, tell you what, what you are asking me to do is going to cause harm to my product and harm to my team that's why I am refusing. I want to turn out world-class products and I am telling you, you’re threatening it. He tried a few more times and I said okay look, I will tell the team that I was required to tell them what you are telling me. He said okay, but he didn't understand what I was really saying there. So actually I did, in front of the team I said I am required to tell that if you aren't here by 8 in the morning you are supposed to write a note. There was dead silence in the room, 50 or 60 people looking at each other quizzically. We are here all night, we are here on holidays, we are here on weekends and you want us to do what? I said I hear you I made the same objection.

One of them said, if I write you a note what you are going to do with it? I said I am going to put in that round can right there, I am not going to read it, I don't care what it says, because this is a dumb thing. Keep getting your jobs done, meet your commitments, and how you do it is up to you, you are adults. Go make it happen. So that's kind of where it went, it kind of died there, but if I were a true manager interested in a manager career at Intel, I could not have handled it like that. Managers have to go the direction they tell you to go. And that's one of the reasons I never aspired to be manager at Intel; there's too many games. I can't tell you how many times I was in a staff meetings, high level ones, with all the top brass of the company, and questions would get asked and I would realize wow there are games being played here, interesting games. When that guy answers a question he gives you a straight honest technical answer but when those two guys over there do it, they're posturing because the big boss is here. You can see it, you can hear it in their words, you can see it. They will say things like, boss that was a great idea. [Chuckles]Where did you get that, out of a comic strip? It's like, give me a break, do you really think no one sees you do that?

1:39:39 PE: Yeah.

1:39:39 BC: With Andy Grove, this stuff didn't work. He just didn't react to crap like that. But later on, when he was no longer running things, it started working, that was rewardable behavior and it got really embarrassing. I didn't even want to be in the room watching these guys do these little dances. Because I thought man we should be wrestling with the difficult questions, the whole should be greater than the sum of the parts but not like this it ain't. Turn out great products and we will be rewarded that's all there is.

1:40:11 PE: Yeah. Well... I guess... I don't want to jump ahead too much, but then since you just mentioned this at the end of Pentium chronicles, you say they left Intel because you were frustrated and I think this is part of the story?
1:40:26 BC: Yeah, that's a piece of it, but it was more of the other thing I was talking about where I felt it was my job to bring messages to the boss. I can't always prove them what I am saying is correct.

Perhaps I have reached a frightening conclusion about our plans, then I should have enough confidence to think that the boss will consider my vision; I want to make sure he is considering it, that's it. What he should say is, thank you for bringing this to me, I realize that it is hard, I realize you are going around your chain of management and that is sometimes awkward and painful and takes guts and I am glad you did it; I may not agree with you, but that is ok. That's a perfect message but I was not getting that. Instead I was getting the message that my inputs were not welcome, they were not interested. I was told I was only doing this because I expected to benefit by attacking other projects. And so on. Mr. Executive, I don't want to hear your crackpot theories as to why I am bringing these issues to you; I know very well why I'm doing it. What the executives should be assuming is that I'm there in good faith, I am doing it for good reasons.

I profoundly believe that designing good products is all anyone has to do and if you do that you will rewarded and if you don't then you have failed. It's that simple, I don't think you need to play all these other games, personal glory seeking and all that stuff, it turns me off. It's a distraction. One of things that we wrestled with early in the P6 effort was how to accurately reflect credit on a team where 5 people are brainstorming every day. That's why when you look at the patents, my name is on 40 patents and I think 35 are from Intel. And almost all of them have the same five names on them and that is because when all was said and done we all participated in the process that led us to that to that outcome.

One of the five, who shall here go unnamed, had a somewhat juvenile tendency to try to get the last word in and then claim sole credit for the idea. I had to take him aside on multiple occasions and say look, first of all you are a junior guy and you are in a company of three very senior guys who really appreciate and value your inputs, which is why you are in that meeting. You will notice that we are not playing ego games with you, we are not saying we are the big boss you're a little peon, so give me your ideas for which I will take credit.We are not doing that. But by the same token you have to not run off with it either, just because we're being fair about this thing. You're going to rip my team apart if you do. So we had that problem lots of times and it took him a while to come around; to this day occasionally there is a still little bit of flavor of that with this particular individual but it is okay. I mean he was really smart, he was really talented, he had lots of great ideas, it was just that his personal skills were missing that one piece.

1:43:16 PE: [Chuckles] Not that uncommon with computer people.

1:43:20 BC: Yeah, exactly. When I saw that, yet another corner where that needed a little extra attention. Other guys might not have noticed.

1:43:27 PE: Yeah.

1:43:28 BC: So I took it took on myself to say we will be a better team if our behaviors are a little bit different.

1:43:33 PE: Yeah, yeah.

1:43:35 BC: It look a lot of handling. I think on the whole it was pretty affective. I've heard since then that without that hand holding this particular guy had much more difficult time fitting into teams and contributing. I feel bad about that because he is really a talented person, but it is just something you cannot take for granted, it doesn't just happen, you actually have to intentionally make teams the way they need to be.

1:43:55 PE: Interesting. Okay, so we talked about the data flow analyzer I want to go also ask you about later on you moved to...you built the behavioral model in between the data flow analyzer and the SRTL?

1:44:15 BC: Yeah.

1:44:16 PE: And then it sounded like there was a probably unbeknownst to you, there was another parallel effort going at the company level that they...

1:44:27 BC: Yeah.

1:44:27 PE: They also produced a simulator. Which you did not name, but again gave your own acronym of... Something like the simulator that these other people produced, or something.

1:44:39 BC: [Chuckles] A simulator I don't want.

1:44:40 PE: Yes.

1:44:41 BC: Yeah. If you consider the root cause of that problem from Albert Yu's level, he was being told irreconcilable visions of what the performance would be of future chips. Culture differences existed between the Israelis, the Oregon team, and the Santa Clara team. We all made our own decisions as to how aggressive to be. Do you over promise and under deliver or the other way around. There was no Intel stamp that said you will do it this way. So poor Albert is trying to put together the roadmap on which all these parts will eventually place. And he needs to position them relative to each other, and it has to make sense because he's going to have to explain it to customers.

Well, he was getting numbers that weren't reconcilable with each other. The Oregon team would be conservative. We would tell him some number that we are pretty darn sure we could hit, and we were hoping for a good upside from that. But from one of the other teams he'd get some wildly optimistic number that if all of the planets align, you might get close to it and things never ever work that way. The assumptions never line up in your favor, all of them at once. A few of them maybe; but all of them, never. Can I prove that? No. I just know that it's true and if I'm going to make a roadmap that's the basis on which I'm going to make it. But the other teams might say "Well, if this goes well and that works the way we hope, and this compiler does that and blah, blah, blah, I'm going to end up with this really cool number and that's the one I want Albert to hear". At one point over a beer, I was taking my cohorts to task on this very issue and I said, "You must know that you and I aren't playing by the same rules and it's messing him up." He said, "Yup, I know."

You must know that that all the essentials don't line up like you would need, so you are at much greater risk of missing your promised numbers than I am. He agreed. And I said, "Well then, doesn't it make sense that we should agree eye to eye on this as so that poor Albert doesn't have to apply different fudge factors to all what we say?" He said, "Well, I don't know the answer of that question but I can tell you why we do what we do." He said the Israelis keep trying to really make a dent in this company (which, by the way, they now have. They made a huge dent. The Israelis did a great job on Core 2 for example. It's one of Intel's flagships right now.) But prior to that, they kept trying to make a dent and they weren't really getting there. And so he said, "We were always in danger of getting you know, just chopped off because we were you know, we just weren't one of the teams they thought of. So in order to get attention, we would get aggressive on our proposals as to what we think we could do". And he said, "Besides, there's also a decent chance that Albert will quit his job and go somewhere else or the project will get reassigned and we would not have to deliver on our promise." This problem was enough on Albert's mind to where somebody sold, in fact who it was, the same guy that was sitting in the front seat that took me at task on Multiflow. He made a proposal that if we had one common simulator in this company and everyone had to use it, it would take a lot of this uncertainty out and it would make us all play by the same rules. He went on a mission to spread this gospel through the whole company and made a lot of headway because Albert loved it. And so he went to Folsom, and they bought in, and Santa Clara also bought in. T

hey came to Oregon and I said "Get lost. I have got my own simulator. I'm not going to switch horses in the middle of the stream. I have reason to think your simulator is not as good as the one I've already got. So what's in it for me?" He said, well you might have to take one for the team because the whole company has got to be on the same footing." It was an awkward question, actually. He had some good points and so did Albert. I don't deny that their motivation was good and if it could have been made to work, I would have probably gone along with it. But for the same reasons that DFA was good for P6 and not for Willamette (Pentium 4), this simulator had some aspects like that that would not have helped us and might have hurt. Where it would have been really good was for the machines like the Itanium for example and it was not so hot for the seriously out of order machines that we were building.

I actually put a really talented guy on evaluating this new simulator, to find out if we could even live with their simulator if we had to. His conclusion was yeah, we probably could but it's not worth the effort because we're already halfway to the goal with the simulator we got, and we're fine. Switching is just overhead, it's not going to help us, and it might hurt. So I went back to Albert and said, "Well, Albert to what extent do you want to jeopardize your main product line for the future possible benefit of other products that clearly aren't as valuable?" And he said, " I don't know anything about that but I wish we had just one simulator." In the end I stalled it long enough to finish the project and after that I don't even know what happened because I wasn't in charge anymore. That's just an example of a corporate initiative. Some of which are pretty good. Some of them are disasters. As a project manager, I think you have to know when to lie down on the tracks and when to cut your losses, and when to say thank you, this is good. Because sometimes there are initiatives that are valuable.

1:50:03 PE: Yeah.

1:50:04 BC: When I started the P6, Albert said "I want modular design. I want you to design units once so I can use them in multiple projects after that." And there were some examples that worked that way, like the floating point divider. If the Pentium guys had not touched it, since it was the same divider in the 486, the FDIV recall problem would have never happened. But no, they opened it up and messed with it and broke it. We had the same divider and we didn't mess with it. We left it alone and it was okay. So that was an example of a function unit that over time should have been improved but for other reasons we didn't think we need it to be improved, so we left it alone. There were other units like the APIC, the Advanced Programmable Interrupt Controller. Albert said, "Some people have designed this new thing called the APIC and you're just going to use it, okay?" And I said, "Well, who designed it and how do I know it works?"

I'm a skeptic. I'm a paranoid engineer. And so, I put some of my validation guys on it and asked them, "What does this thing do? Does it work?" They took a quick look at it, didn't like it, gave it to Dave Papworth. He took a look at it and proved it could not work. He pointed out that the APIC spec required that the inputs must be both edge sensitive and level sensitive. You can't be both at once. You have to pick one or the other. So we went back to the APIC guys and pointed that out and they said, "Okay, yeah. Well then, just turn one mode off and use the other one." And I said, "No, no, no, our confidence in your design is quite low at this point -- we don't trust you." It took a lot of effort to retrofit quality into that design because it just wasn't production ready. So any effort by the company to sort of centralize that process by which one group creates a unit and proselytizes it to the company, I profoundly distrust that way of doing things. You have to pack your own parachute.

Part 4 (1990-2001): Intel continued

0:00:04 Paul Edwards: Okay so, this is part four of the Tuesday session with Bob Colwell, Paul Edwards interviewing, August 25, 2009. Let's see... so we're still talking about your time at Intel. We were talking a little bit about the behavioral model and the simulator. I think these tools are very interesting because they're application of information technology to its own design, its been a feature of this industry that's been really kind of unique. You don't see that so much anything else except maybe machine tools which are used to make other machine tools. But as the chips get more and more complicated there's you talk about this on the book but once you've heard the three million, six million transistors and many hundreds of thousands or millions of clock cycles to do some operation just figuring out what the thing is doing becomes incredibly complicated. And these tools you've built were part of that process. Did more things happen in that stream while you were at Intel?

0:01:19 Bob Colwell: While we were at Multiflow somebody, I don't think it was me, had the idea to put what we call debug hooks into the actual hardware. And that would turn out to be an incredibly useful facility because at Multiflow the machines were built on big circuit cards that were like twenty inches on the side and there were seven to ten of them, and some of the cards had been modified with wires because there were design changes or errors; the point was that every so often the machine would malfunction because something had gone wrong: a wire popped off or there was a bug we hadn't thought of and the machine would leave the rails. It would work for a while then something bad would happen and it would go off into the weeds and we had to figure out what happened. If you don't have some mechanical help then you're reduced to putting things like logic analyzer connections into the machine to track what the program counter (PC) values had been. And that's problematical too because one of the leads could pop off and now you're not getting the right word anymore and it's quite difficult to operate that way, you have to be highly motivated to attack that problem at that level. It can be efficacious to do it but it's painful, you have to really want to do it. So instead we said "what, without too much trouble, we could put a queue in there that and every time there's a PC value it goes into the queue." You don't use it for anything it's just in there. And then there's a trigger mechanism that says stop collecting PC values just keep what's in there and then via another means we can dump that queue to a screen or to a file and we can analyze it. This facility turned out to be incredibly useful. Just outrageously useful and there were a bunch of other things we could also do with the same kind of basic facility. I actually wrote a paper...

0:03:05 PE: Isn't it exactly like a software debugger.

0:03:07 BC: Yeah just, well if you remember my days from Bell Labs when I was developing the in-circuit emulator, I didn't invent those, but I knew what the facilities were inside them and this is similar. Boy when that worked it was really useful. We wanted to build it into the big computer for the same basic reasons. When I got to Intel and I think one of the things we realized, was the same thing that was happening all the way back to Bell Labs days: when you have a microprocessor-based machine it does what it does. You can't see what it's doing, you have no direct visibility into it and if you didn't design some way of getting visibility into it, it’s very difficult to infer anything from the outside characteristics. You’d like to put wires into this thing, but you can’t. We need a million things we just can't reach. That's one other reason why RTL simulators are so valuable, because when you're in a simulator of the internals of a microprocessor you can essentially say, "I want to see what that signal does every single clock cycle" and the simulator will show you, but in real silicon, forget it. You can't reach every wire in there. You can't even, not even if you're super motivated and you take the lid off and you probe around with a probe you just can't do it.

All the way back at Bell Labs when the microprocessors were only a hundred thousand transistors, we already faced this issue. That seemed like a big number at the time. We could see now we're up to six million and the transistor count is going to keep going; we're up to a billion in a few years and we need some other way to handle this problem. So back in the P6 days we went on a campaign to stick in what we called debug hooks. It was things that we knew we would care about or, taken in conjunction with each other, would tell you interesting things about the state of the machine. We provided some ways to scan this information it out so you could reach it while the machine was still running and we put a fair amount of work into getting those in there and working. They made a big difference.

Later on we also added performance counters; a similar idea but those are in there in case you're not happy with the performance you're getting and you want to know why. Maybe it's taking too many cache misses, maybe it's going to memory more than you thought it would, who knows but we had a bunch of counters and you could apply those counters to various things and then run your program, look at the counter value and say "oh my gosh, it's way bigger than I expected. I wonder why.". It would give you some insight as to how your program was doing directly from the hardware. You might think well wait a minute you could run that software on a simulator, and in principle you could, but the speed difference was enormous, w the real hardware is running at hundreds of megahertz and the software would be running at ten hertz. You're not going to live long enough to see the end of the simulation. So there's limits to that. ... If you tell the design team it is your avowed intention to put in design hooks or debug hooks, or performance counters and having been a designer I knew this was going to happen, they're going to go "okay yeah I can see why you want those.

So after I've designed the basic functionality and gotten that right, if I have time, I'll put that stuff in too”. In other words it’s an afterthought. By being diligent as a manager on projects you can say "wait I noticed Jim you didn't put your debug hooks in there yet, when are you going to get around to that?" You can make sure they all eventually do it, but what you can't really do is make sure their heart's really in it because if they stick in there in slap dash it's not going to work, it's not going to tell you what you really wanted to know. An example is a performance counter where they weren't careful about stall cycles, like when the machine has to stall for various reasons and they ended up counting the stall cycles as well as the running cycles.

And it basically made the performance counter’s count a random number. And you needed to think more deeply what was being asked about the debug hook. So to deal with that I purposely put people in charge of both things; one guy's job was to do the debug hooks and make sure they came out right, make sure they got validated, make sure they worked the way we wanted. Another guy did the performance counters and his job was to do the same thing. This managerial stratagem didn't mean that there was no bugs in those areas but it did mean that basically they worked and they did what they're supposed to do; you could get real value out of them, they weren't a total waste of time. The only way I knew how to do that was to get a human being to constantly apply pressure because otherwise you were going to get this default "yeah I'll get to it when I get to it" and then they won't get to it. So yeah so we did that on both P6 and Willamette and got lots of good value out of both of those I think.

0:07:38 PE: Was that the first time that had been done?

0:07:40 BC: At Intel yeah, to my knowledge. I believe the Pentium did some performance counter. I'm not sure what it did, can't remember anymore. But nowadays the Intel will sell you an entire suite called V-tune that uses...

0:07:53 PE: It spell V T U N E?

0:07:54 BC: Correct. The last I saw, it was reasonably expensive actually but it's a suite of software tools that use those hooks inside the microprocessors to give you a coherent picture of what the machine was doing. When they first proposed doing V-tune they worried me, because there's a sneaky issue here. To do performance counters and to sort of see where all the cycles are going as your program runs, you may want to break it down to the actual instructions themselves, the x86 instructions. There, you might find out that this instruction is using 1 percent of all the cycles and this other one's using 5 percent and now you know who to pay attention to, but if VTUNE was going to be that accurate it would be at risk of exposing the microcode. The way P6 worked was that it would take the x86 instruction, turn it in to micro-ops, and the instruction hasn't finished until the last micro-op associated with that instruction has finished. So as a microcode writer I might know that to implement that instruction took me five micro-ops and here they are, but I don't want those outside of Intel to know that, that's microcode, some of Intel's family jewels. If you tell people the microcode of your machine they can design their own machine much more easily. There's a significant IP aspect of the microcode, so we had to try to find a happy balance between enough information to let the performance analyst succeed and not so much that we would expose intellectual property in our microcode. The compromise that I proposed was that if it's four micro-ops or less we show them the actual sequence, because for instructions that simple, there's no mystery to what it does anyhow. But for instructions like far call through a gate there's some really complicated x86 stuff; there might be 200 micro-ops. We don't show you those micro-ops; instead, we say the x86 instruction takes 200 micro-ops. So the performance guy knows it's 200, he doesn't know that they are and doesn't care. So little things like that abound and you have to address them. You have to think about carefully to see where the balance should be.

0:09:52 PE: Yeah. A couple of more things I wanted to ask about. One is this period in the mid 1990s. We talked about this a little bit yesterday but mostly the context of your sort of what kind of computers you had at home. When Intel decided to go to Windows NT from UNIX that sounds like that was quite a messy, messy transition and not good for your team anyway.

0:10:26 BC: It was horrible. For me personally, this was one of those cases where I felt generally bad that I'd somehow let my team down and I don't like that feeling. I mean it's one thing to feel like there's forces beyond my control that aren't the way I would like them to be. Get over it, that's life. But that particular one… I had always encouraged my whole team along these lines "if you want to call yourself a chip architect you must observe no boundaries ever." You remember that guy at Bell Labs that said "hey you're only a designer, I don't have to talk to you", well, that's a boundary. There must be no boundaries, it leads to bad design choices. So I said if you find a bug in UNIX for example and it's messing you up go find out where it is. Go into the source code, figure it out, work around it, don't let it stop you. I don't want to hear excuses; those are not going to help. Go nail it down. If it's in the compiler go nail it down. There are no boundaries. I learned that from Dave Papworth, too, by the way. He was just a wizard at sitting in front of a dead computer and following the symptoms wherever they might lead; nothing stopped him. There was no area where he'd say "Man I don't know enough about this to do this, so forget it, looks like quitting time." If he didn't know enough he would learn it that day; he would just keep on going. I thought he was a great role model for this. If an Architect starts thinking that way, they start realizing that there are trade offs that you can make that you could not have made if you'd stopped your investigation at some arbitrary boundary. You didn't realize, hey something over here could've helped me. So I was trying to get my troops to think in those terms. Well, here's the problem: I succeeded at getting a lot of the architects on my team to think that the operating system was fair game. Suddenly NT comes along, and we don't have source code. We're not allowed to have source code. We're not even allowed to cheat and look at the assembly code, we can't disassemble it. We are forbidden to mess with it.

0:12:17 PE: Right, you're talking about the source code of NT.

0:12:18 BC: Of NT. Yeah. We had a whole bunch of tools and scripts and things that were working really well with UNIX and these guys weren't Windows experts either, any more than I was. So they would take their first cut at solving a tools problem, well it looks like the way you do this in an NT universe is you take my script and you modify it the following way and you run it and then you observe the output. It looks like it’s okay so chalk that one up and move on to the next problem. But after a couple of months of this there was tremendous grumbling and this is why I said I felt really bad, because my chief software guy at that point came to me and said "you let us down, you should not have let this happen. This is a huge dent in our project for no good reason. This is exactly the kind of thing that you said that you would never let go and that you said that if we ever face it we will stand up to anyone that needs it. And yet here you just lay down and took it." I felt very bad about it, and suggested we try to make it work, let's see what's going to come out of this. I agreed that I had not prevailed on an issue that was important but I didn't know what to do about that at that moment. He kind of grumbled but basically he must have accepted my quasi apology because he went back to work instead of quitting like he threatened to do. Anyway, and he got the team going on the conversion. Next the validation chief came to me saying "I'm worried. We're not getting the cycles that we used to get. We used to crank through a certain number of validation cycles every night and now we have more machines with theoretically higher capacity and I'm getting far less useful output. So why is that? Is it that they're slow, not as efficient? Half the jobs I submit, literally half, never finish. They never succeed." I said "Is it a random thing?" He says "yeah as far as I know, I have no idea what the pattern is, I just know that every hundred jobs I submit, I get fifty back. And so it’s going to take me twice as many machines, for twice as long.

I went back to the software team and I said "guys there's a problem here, can you guys take a look at this? Maybe we'll find out what's wrong -- maybe there's an issue we didn't think of." They attacked it and then said "oh okay I think we can see there were a few bugs maybe in NT" and they patched those and ran it and it looked okay. So again they moved on but then two weeks later some new problem would show up and they would slay that dragon too but we were asymptotically heeling over. We weren't getting to the number we wanted and the jobs were still failing; not 50 percent but like now we're getting 70 percent through and it's still not enough. And finally the pattern became clear to us. They had never really solved the original bug because without the source code, they couldn't know what it was. They were guessing at the bug, guessing at the solution, and doing as much testing as they could. They passed that, thinking they were done but they weren't. They hadn't actually solved it,

0:15:15 PE: Could it be a corner case?

0:15:16 BC: Yes. It would just come back and bite us again.

0:15:18 PE: Right.

0:15:19 BC: That was one thing, that dawning realization that that’s what is really the root cause here. We don't have source code and we're shooting ourselves in the foot over it.

0:15:26 PE: Yeah.

0:15:27 BC: The other one was that the validation crew said "we're not going to finish the project this way. So we're going to take drastic action. How about we get a bunch of Linux boxes in here and we start using those." And so I mentioned that to Albert who was the decision maker here and he said, "No, over my dead body. I do not want that, you can't have that. You have got to make this Windows NT conversion; NT is the future." So I came back to the validation crew and I told them what he said.

They replied "Well, what we've already done is put a bunch of Linux boxes in the lab and they work just fine." We then decided to take the Linux boxes on an “extended field trial” for the next few months and if it works well, we'll tell Albert that we did it.

And this is the Dilbert moment at the end of all this. The Linux experiment worked beautifully. We got all the cycles we wanted, and it was as stable as a rock. We got our validation back on track. At the end of the project, the guys who did that guerilla action, they got an IAA award, the highest Intel technical award, from Albert for having taking the initiative.

0:16:34 PE: Good for them and good for him.

0:16:36 BC: Yeah. It's like whoa. Well...

0:16:38 PE: Yeah.

0:16:38 BC: On the one hand, he's reinforcing the idea that I subscribed to: get your job done; get this thing done no matter what it takes. On the other hand, he's also suddenly reinforcing something he doesn't intend which is you can't always listen to what management tells you.

0:16:53 PE: Yeah.

0:16:53 BC: It's like okay, you know.

0:16:54 PE: Yeah.

0:16:55 BC: But that was a truly Dilbert moment when they awarded this thing.

0:16:59 PE: It's too bad MAC aren't going over to Intel yet why, because its UNIX based and you could have just used them.

0:17:05 BC: Yeah. That would have been nice.

Interaction of work and family life in the Intel years

0:17:07 PE: Let's see. Oh yeah. So say a little bit about your interaction of work with your personal life at that time, your family and support here. I'm asking this question because historians are getting more and more interested in the sorts of invisible work the support people who are doing important things.

0:17:49 BC: It's very relevant to tell you that.

0:17:51 PE: Yeah. Talk about, there's some in the book about your, the family nights and stuff like that that you started to do for that.

0:17:59 BC: I think it makes a huge difference if your family, and your spouse in particular believe in the mission that you're on. If they want you to succeed in what you're doing, if they believe in it, that makes a huge difference to how effective you're going to be.

0:18:19 PE: Yeah.

0:18:20 BC: So at Multiflow, Ellen was part of Multiflow. She was bought in. We both wanted to make this thing a go. At Intel, she couldn't get in the building anymore without me escorting her. She wasn't at the meetings where they would tell us what we're trying to do and why and so on. She was necessarily a couple of steps removed in a way that she hadn't been prior to that. And it was also the first place where she and I weren't working at the same place at the same time. We both worked at Multiflow, we both worked at Three Rivers Computer. We both worked at Bell Labs. So we were just kind of use to car pooling and seeing each other at lunch or whatever.

0:18:52 PE: Yeah, yeah.

0:18:52 BC: And suddenly it was this estrangement effect and plus I was working my tail off at Intel and it wasn't necessarily always clear why to her.

0:19:02 PE: Your kids were still quite young?

0:19:04 BC: The kids were little...yeah. When I first, in '90, Kelly was five and Ken was three and Kristen was an infant.

0:19:09 PE: Yeah.

0:19:10 BC: So for the first few years, they were quite small.

0:19:12 PE: What did she do during the Intel period from 1990 to 2001?

0:19:19 BC: Who, Ellen?

0:19:19 PE: Yeah.

0:19:19 BC: She was mostly here raising the kids and during a couple of those years, she work part-time at some local technical start up. I forgot which years those were but she never worked at Intel.

But yes, she was mostly here raising the kids. So if you ask her, her memories, she's told this to me many times. Her memories of the early 90s are Bob's away on a trip or working at work and two of the kids are throwing up all night. And she says, enough of those nights, and you start feeling picked on.

0:19:51 PE: Yeah.

0:19:52 BC: Because you're the only one here. I think yeah, I sympathize. I agree. But that is why we project managers tried extra hard to reach out and have nights when the kids could come in. The food would be there so she didn't have to cook it. And we even have things like, when things got really hot and heavy towards the end of the project, we would actually pay for landscape services to go and cut the grass at people's houses.

0:20:16 PE: Right.

0:20:15 BC: So they'd feel like their house or their life isn't falling apart just because they're dedicated to their work. This actually became an issue. This was a subtlety that's only partly related so I'm going to get back to that in a minute but I'll forget if I don't say it. On the P6 chip, the architects kind of front-loaded the project. We spent the first year and half to two years with a very small set of people thinking hard about what this machine ought to be like. And then the rest of the time, the other three years had the big design team join us and they went off and did the details and created the silicon. But one of the lieutenants of that design effort watched this process and convinced himself that of a five year project, the first two years should not be spent with a team of five people. In his mind, that's a waste of time, and instead those people should do their jobs in like six months and then never change their minds because if they changed anything, it screws up the design team. They lose a lot of time and effort. It's not very efficient.

So he told me at the beginning, the guy I'm referring to became the design manager. He said, "Well, these are the ground rules. I'm going to give you guys six months, do anything you want in those six months. But after that, you can't change anything unless I agree with it, and I won't. And so, you guys basically have six months to get your act together." And I just laughed and I said you have got to be kidding me. We're not going to be able to do that. I guarantee we can't do it and I guarantee you that after six months, we will make major changes to the design and there's nothing I can do about it and nothing you can do about it. If you want to have a competitive part at the end, that's the way it's going to be. What he was trying to tell me was at the end of every project, you have a snowplow effect where you've accumulated more and more and more stuff to do and it all has to get done before you can tape it out and get going. Yet the schedule date doesn't move so that is the mental image everybody on the team has. You're driving a snow plow and you've got to get the road clear but there is just more and more stuff in front of you.

0:22:11 PE: The snow is accumulating in front of you and...

0:22:14 BC: The target is standing right there and laughing at you. It's not moving. So this guy was trying to prevent or at least cut down on project's traditional final six months' death march, because that distressed people's lives. I mean for the last six months of every one of these projects, you are at work all the time, you don't take a break, you don't take holidays, you don't anything. You're at work. He said I've got to cut down on that. But he believed the root cause was a function of the architects' laxity or lack of discipline or something.

0:22:38 PE: Yeah.

0:22:38 BC: So he and I had a lot of friction. He was correct about the problem that the snowplow effect does happen. I don't think he was correct that the solution would be to frontload the project even more, as though then there would be less snow. You're still going to have a snowplow, it's just going to be different stuff. I think that's nature's way. When you try to finish anything, you're going to discover, you get what you think is the bulk of the work done, and the remaining 10% takes another 50% of the time because there was so much more there that you didn't think of.

Welcome to the real world. To me, that's classic: "design anything you want, you're going to find the same effect". We ended up with a problem, although his heart was in the right place. He was trying to save his design team I don't want to say agony but the intense forced march at the end of a project because of the disruption that it causes to family life and so on. But I don't think his solution was going to work, ever, because part of it is you just learn so much as you go along. The mental model I use is that you start out on a journey and you think of all the things that are going to happen and there's going to be intermediate steps along the way. You throw tools and food in your backpacks so you'll be prepared for those things. If you're a good designer, you'll end up with a pretty good backpack but I guarantee you coming around the first bend that you're going to be surprised by something and you're going to have to react to it. If you don't react to it, you're going to end up with a worse product. If you do react with it successfully, you're going to do a little extra work but it's going to be a better product for it because after all, your competitors are following the same path. And if they don't react to it and you do, you beat them. Now, if you both react to it, you're still competing. This is how it is but you can't just refuse to do it because you do have competition. So I just think that the discovery aspect of complicated designs was not as clear to this guy as it was to the architects. We probably should have tried to find a compromise in between but instead we just fought like cats and dogs. It wasn't very pretty. But yeah, this tension with family life was tricky. It was really tricky. And I mentioned this yesterday you see it's one of those things where some, I had this happen more than once. Somebody would come in at my office, a guy who’s like 24 years old and he’d say "I've been married for a year. My spouse says she never sees me. We're starting to fight. What should I do?" And you go "Oh man. I don't want you to get a divorce and I would never tell you that work is more important than your spouse; never, ever. Because it isn't".

0:25:03 PE: Yeah.

0:25:04 BC: You're going to have your spouse for the rest of your life. This work, this project's going to come and go in a few years and if you do a good job you'll remember it fondly. But under no conditions would you say "Well, I've lost my wife but at least we turned out a good chip." That's not going to suit you. You're not going to like that. But by the same token, if we don't apply ourselves seriously to this thing and our competitor does, they're going to kick our butts. It's a competitive field; you've got to get it done. So my compromise was when you're the guy at the critical path, then work is it. But when you're not, find the right balance to make sure that you're not sacrificing something even more important than the job just because it's more urgent. It’s not much of a prescription but I think it's the best compromise I've ever come up with.

0:25:48 PE: Yeah, yeah.

0:25:50 BC: And then trying to keep the kids in the loop is tough. When I was working a lot in the evenings, thank God Ellen would bring the kids just to see me, because I couldn't get home so she would bring them the other way and that helped a lot too. Early in my career at Intel I didn't travel a whole lot the first three years. But after that I was on the road quite a bit and then it got really hard.

0:26:14 PE: What were you doing were you trading in Santa Clara and things like that...

0:26:18 BC: Santa Clara was...

0:26:18 PE: For Intel?

0:26:19 BC: Sacramento, visiting customers, going to conferences, giving talks. There were just lots of reasons to do it. It also seemed like good career stuff and in retrospect I think it was but it means you're away from home a lot again and it gets old.

0:26:39 PE: Yeah. I want to come back to the customer visits I just wanted to ask about it, just a little bit more on supporting characters in all this. What about people at Intel that wouldn't normally come up in a company, in a book like the one you wrote such as secretaries, other kinds of staff people who aren't necessarily engineers and who else like that?

0:27:07 BC: Yeah. Intel has an HR Department, Human Resources. I actually wasn't really aware of those folks for the first few years I worked there. I just didn't see them much. But when I took on this management role, like every year when you do the R&R, the ranking and rating and all the machinations of giving out reviews and meeting with people, raises and so on: all of that is under the watchful eye of HR and so I started seeing these people more often and I really liked them. At first, I was a little bit confused as to what their role was, because these guys were so helpful. People like Mary Killeen and Dawn Kulesa and there was a guy whose name I forget but there was...

0:27:46 PE: Could you spell those names of Killeen, K I L L E E N?

0:27:49 BC: Correct. And Kulesa was K-U-L-E-S-A and both of those ladies were unbelievably helpful to me. I mean they went way out of their way.

0:27:57 PE: Is this Dawn?

0:27:59 BC: Dawn, yeah D A W N. They went way out of their way to explain to me how the process worked what the ratifications were of getting it right and of getting it wrong. They were super helpful because this is my first time I ever did that stuff and I didn't know Intel's way. I was only a recipient of it before that. So yes, those folks were super helpful but it became clear that there was friction occurring, some conflict was hiding behind that. I later on discovered what it was.

Initially my attitude was oh thank God somebody at Intel had the wit to set up an HR Department to put a more personal human face on a big company that otherwise would seem as though it had no soul. It feels like that sometimes and these people can represent Intel and give it a human, warmth and attitude, that makes it so much more helpful or if you feeling lousy some day, they will try to pick you up and so on and I said that's a brilliant idea. But that whole interpretation is wrong. That's actually not what HR is for. Mary and Dawn were fantastic because they are great people, not because it was in their job descriptions. But really what their job description is, when it comes down to it, is that they want to keep Intel from getting sued by its own employees. If engineers leave on a disgruntled basis, and some do of course, when you let go of an employee, they're not going to be happy. They're not going to say thank you for firing me, I didn't it like here anyway. No, my experience is they walk out the door saying there might have been issues with my performance but they weren't my fault. They were <u>your</u> fault because you're a lousy manager and let me explain to you why. I got that reaction once, and I've learned since then that it's pretty typical of people. It's one of their coping mechanisms. HR was trying to help us, and once I understood that it's not really their role to help, they're just being great people, I realized that I had to be careful what I asked them to do or I can get them in trouble and they could lose their jobs. Those people were really great. Katie Abela-Gale was my admin for many years...

0:30:00 PE: Can you spell that name?

0:30:01 BC: A B E L A dash G A L E. She was unremittingly cheerful, which helped a lot. Some days just don't go very well, and you feel like kicking the doors and wondering why fate did this to you, and her cheerful outlook was really helpful.

This was on top of basic skills for setting up my travel and working on expense reports and all that kind of crazy stuff, she would keep track of, "oh by the way Bob next Wednesday you're committed to having this report done or whatever" and I'd thank her, because otherwise I'd have forgotten. She would remind me of official stuff that otherwise just wasn't part of my daily perspective. She was really good at picking up people's spirits, including mine, and when things weren't going so well. Yeah she was a lot of fun. We had a pretty good admin staff. There were of course management issues that I never saw coming. I don't know if I should have expected them or not, but a classic example for me was, I'd been promoted to this management job and two days later one of my top software guys says "hey Bob I want to talk to you can we go in the conference room?" I figure he's going to congratulate me on the new job. Nope that's not what he's going to do. We go in the room and he said "I just want you to know we've been working as peers now for a couple years now and now you're the boss and I just want you to know, I don't think you can do this job. I don't think you have the skill set, I think you're going to fall right on your face, and when you do, I don't want to be here to have to see it happen, so I'm just warning you, sooner or later we're probably going to split up". I said "Well gee, thanks for the vote of confidence." And now, the same guy about a year later, again grabs me in the morning and says "we need to talk". So we go into the conference room and this time he says "look I don't have a PhD, and I noticed that you just hired somebody, like last week, and he drove up here in a Porsche. I can't afford a Porsche, I've been working here for like 20 years, that isn't fair, I don't like that." This time I said "I'm not your Dad, if you want to advice on how to deal with the inequities of life go talk to your Dad. How do you know that the guy didn't inherit the dang car from his father? I mean come on it's nothing to do with PhDs or any other damn thing." I basically spanked him and sent him home, but I remember walking out and thinking, did I sign up for this? Where's that in my job description? People!

Women at Intel

0:32:27 PE: Yeah. One sort of side question is, none of the, unless I'm wrong, none of the people that you talked about in the Pentium Chronicles, in the engineering staff, were women. Were there any in your team?

0:32:40 BC: Very few. Yeah there were a couple.

0:32:43 PE: And it's of course very typical in this world but.

0:32:45 BC: Yeah there were very few. But I had a woman on the Microcode team for a while, and I had one in the Performance Analysts team for a while, and I had one on my

Compiler Liaison group for a while. But they were scarce, yeah, and I really liked having women on the team, it made the meetings much better. The boys behaved themselves better, it sometimes comes down to that really. I think the dynamic is improved when you have the right representation. And then some of the women like, there was a woman on the design side, two of them actually, they're now VPs, and they are tremendously effective. They were really good at their jobs.

0:33:24 PE: Who was that?

0:33:26 BC: Rani Borkar was one of them.

0:33:28 PE: Spell the name out.

0:33:30 BC: Rani is R A N I, Borkar is B O R KA R. And Eileen Riggs was another one. She was a cache designer and she was just so darn good that every time you gave her something she'd hit out of the park and say "what do you want me to do next?" I love employees like that. So yeah, we didn't have enough. And I also spent a fair amount of time recruiting, I was just trying to talk people into coming to work for us, visiting schools and it was reasonably tough to get the women, you know, to join a big all male team essentially because they knew what they'd be in for. They had to basically have a pretty burning desire to design something to be able to put up with that environment. And some of them did, but not all. It was a matter of some chagrin now and then because I could see a good candidate just couldn't quite see her way through to joining us, because she was a little bit put off by that atmosphere, probably rightfully so. My wife, Ellen, is a software engineer, and she put up with this at Bell Labs for instance. One of her examples is, she went into work one day and someone had taken a, like a kids toy, cut up doll, where you put the dresses on her and you can change her clothes, and of course they didn't have any clothes, they just pasted her up there, so Ellen chopped a piece of paper out in the shape of a dress and put it on there, to cover up this doll, and from then on it got really dirty from guys lifting up the dress. That's the environment she was in. And more than once she was...

0:34:56 PE: This is at Bell?

0:34:56 BC: At Bell Labs, and more than once she happened to be too close to the coffee machine and some engineer would walk by and say "hey get me a cup of coffee". She said "excuse me, I am not subordinate to you in any possible way". But you know, that's what women face, and they are often called the “girls” and not the women. Yeah, I got my sensitivity raised a lot by just listening to what she was putting up with. I never heard Intel do anything like "the girls" but maybe I just didn't see it. I did have to intervene one time, because there was an issue between an engineer who had become enamored of one of the female administrative people, and she couldn't figure out how to politely tell him she really wasn’t interested. And he didn't know how to take a clue. So I had to do it. I had to do one of those Daddy things and say "okay folks here's what you missed in high school”.

Customer visits and marketing

0:35:52 PE: Right, oh boy. Okay so I was going to ask about customer visits. You talk a little bit about this in the book, you know about Compaq and Microsoft and another company, oh Novell. And one of the things you say about them and I mean the part, that section is partly about most companies not being able to look very far ahead, so in one or two years but not much more than that and your time horizon is a bit longer. What else happened on these customer visits?

0:36:27 BC: Well some of things I learned...

0:36:30 PE: What were you doing, was it one thing were you just telling them about the new chips and what...

0:36:33 BC: It depended.

0:36:34 PE: Were you able to expect?

0:36:35 BC: First of all the dynamic isn't quite what you might think, and I was surprised by it when I first encountered it. There are people whose full time job it is, at Intel, to maintain good liaisons, good relationships with important customers. They could be field sales guys or field service app guys. They *own* that relationship and if it goes south they get dinged for it, so they care about it a lot. They called us “factory guys”, I really don't know why, I don't work at a factory, but they called us factory guys, and they would only call in a factory guy if they knew you pretty well. Basically they didn't want you standing up in front of a customer and saying bad things, of any sort, because then they might not be able to fix it, and it would hurt and cost them money. And so they would invite us to come under certain conditions, like if the customer really wanted an engineer from the design team, they would carefully pick and choose who they would invite and then they would carefully school you: This is what's hot for these people, these customers, this is what they care about, this is their hot buttons. Don't talk about this, that or this, and watch and follow my cue, I'll let when the question is directed to you.

Some of them were very careful to manage that, so that surprised me at first. Because they structured it that way, I had some flexibility on what my role could be. For example I wasn't always the presenter, I was just there listening and answering questions if they arose. But I loved those because that meant I could sit in the back of the room, and the back of the room is where the action is. The front of the room is where the bosses are sitting: they're closed mouth, they don't tell you anything. The engineers at that company are sitting in the back and they murmur what they really think. They have to, they can't help themselves. The Intel field guy would say "we understand your last product, had these features in it and you haven't said, but maybe you're going to expand on that in the next version", and the front guy will say "ah we really can't talk too much about that, but let's continue". Meanwhile, their engineer in the back's muttering "man that was a disaster, we better never do that again". I started sitting in the back, because that's where the action was, and engineers have this subliminal urge to tell the truth, even if it's painful to them. That's why that joke is funny about the guillotine!

0:38:45 PE: [laughter] That was funny, I laughed out loud when I first heard it.

0:38:49 BC: Yeah, that's a famous joke because it's absolutely so true.

0:38:54 PE: Why don't you tell it, since we're...

0:38:54 BC: First you have to believe that if the guillotine blade gets stuck, you're exonerated, you're free to walk away. I don't know if that's even true, but it's part of the joke, it’s required. So the idea is three people are on death row and the first sticks his head in the guillotine and they release the rope and it doesn't fall because it's stuck, so he walks away. The second guy does the same thing, it's stuck. Then they put the engineer's head in it and he looks up and he says "hey, I can fix that".

0:39:25 BC: Yeah, "I can fix that, I think". That has just enough truth in it. That's why I sat in the back and listened to these people. I'd get the real truth and then I'd come back, write it up and give it to the management and the field guys, and say well they might be officially saying this but they're unofficially saying the following, and that's where I think the true communication is happening.

To be fair, it's probable that I told them stuff that they would otherwise not have heard either, because I'd talk to these guys at break and they would ask me really blunt questions, and sometimes it got awkward. Like there was time when I was in front of the customers with an Intel representative of the Itanium product family and, you remember how I talked earlier about my lack of faith in the numbers they were projecting. And the Intel representative was waxing at length about the marvels of Itanium and I was sitting there squirming, fully believing that there was no way that they could possibly hit those numbers, and finally I'd heard enough, and I grabbed the Intel guy at the break and said "you have to stop what you're saying.

If you believe that it's true then you're just wrong, and if you don't believe it, you have no right to say it to a customer, because sooner or later, Intel as a company will be held accountable by these people. They're going to base a product line on what you just said, and if we're going to let them down, I don't want it to be on my watch". I take more pride in my company than that, and we had a fight, he didn't accept that input very gracefully. He reminded me it's none of my damn business for example. So we had a minor squabble over it but the point was that it's not a good place to discover you have basic disagreements, in front of a customer, it's just not the way you want to do that.

0:41:02 PE: What about Microsoft?

0:41:07 BC: Those guys were interesting. Microsoft, we visited them several times, and they know as much about the x86 architecture as Intel does. To a large extent because they have to live with all the squirrelly architecture cases, you know, that we get right once per project and then forget about. I think their overall technology vision had a slightly longer time line than most, so it felt like they were thinking between three and four years out rather than just eighteen months like most people.

I assumed then and I still think now that it's probably partly a function of how long it takes them to do a new operating system, it's five years, sometimes longer. So maybe they're forced into having a longer view point than many other companies, but I also found that if they gave you inputs it was generally dead on, they had thought hard about what things they would tell you.

They also had some tendencies you needed to understand, and we ran afoul of this twice. If you are contemplating say, a physical address extension facility in the chip, you can think of several different ways to do that, which way you prefer, but they're the customer, so you want to see if they agree or they spot something wrong with what you are thinking of. You want to get it right. But if that's all you've got and you go to visit Microsoft, my experience is, they will give you a polite hearing and they will give you their off the cuff reaction to it, but that's as far as it will go, they don't take you seriously unless you give them a chip. But if you give them a real system and say here's something new, what do you make of this, they take that very seriously and they'll hammer on it and they'll get back to you with very clear feedback about good and bad. But if it's just Powerpoint, well, they see lots of Powerpoint. They engage it but not deep enough and...

0:43:00 PE: They'll be saying that's our product, we know all about Powerpoint.

0:43:03 BC: Yeah you don't trust anything that's in Powerpoint. [laughter] This does feel like that actually, but so yeah Microsoft, so those were pretty interesting meetings and a lot of times you sent some very senior people to those meetings, so it was kind of fun to watch them in action, Ballmer for example. Yeah I did, at one meeting…

0:43:23 PE: Steve Ballmer, B A L L M E R.

0:43:26 BC: Yeah and Bill Gates. I saw him in action once. He was reputed to always zero in on technical folks and demand to know "are you technical?" and then if you said no, he had no use for you. But if you said yes, he was going to hammer you to prove that he was smarter than you. That was, he was reputed to do that. I didn't see him do that in the meeting I was at, so I don't know.

0:43:50 PE: He probably mellowed as he got older.

0:43:53 BC: Yeah maybe, he also had this habit of sitting there like this.... Constantly looking like had some kind of condition.

0:44:01 PE: It's no secret that a lot of computer people are sort of borderline Asperger's Syndrome. [laughter].

0:44:07 BC: Anyway Microsoft was among the more sane and useful customer visits. Novell had to be the low point.

0:44:18 PE: Why?

0:44:18 BC: That was the low point. This is in 1992 or so, anyone that says "I have no use for faster chips, I'm happy with the one I've got", it means you've signed off you've gotten off the train. It means you're going to attract a competitor that is going to spank you thoroughly. Novell could have become Cisco; in the early 1990's Novell was well ahead of Cisco, they had the same basic technologies, the same original vision from networking, but somewhere along the line Novell just fell off the tracks. They seemed to just lose the urge to take over more, or whatever that ambition is that drives the big companies, like Google or EBay or whatever, those guys did not seem to have that gene, they just didn't. So when the Novell guy said the existing chips were fast enough, I said "we're here to tell you about the details of a new chip. It'll be a faster computer, and we want to hear from you what you'll do with it to make sure that we're designing it the right way." And the answer came back, "We don't care what it is, doesn't have to go faster. We're satisfied with the 486." And I thought okay, you're doomed. I don't know how, I don't know why, but I know you're doomed. I mean I know that I can trade faster computers for good stuff. That's all you need to know to realize your business plan is broken. But they didn't seem to know that, so that was a particular spooky meeting. I still wonder what the heck they were thinking.

0:45:38 PE: Let's see. Tell me a little bit about the end times of your career at Intel. So you left in 2001? What lead up to that decision?

0:45:51 BC: The last year I was there, I was not in the microprocessor group anymore, because I had parted company due of my disagreements and vision with the top brass, and I did not want to work on the follow-up to the Pentium 4 which eventually became Prescott. I didn't think it was going to be a great product, and going further down a path that I thought was technologically finished, a dead end. I wasn't willing to devote more years in my life to a cause I didn't believe in. So I bagged that, and I was reaching the conclusion that it was time to leave microprocessor group. I started thinking about more general topics, like what is Intel's future as a company? I thought there's a tendency, a trend happening where companies like Sony used to do, well, they have been in the consumer electronic business for a long time. Panasonic has, Sanyo. All those guys are selling complicated electronic gear into a market where that brand name means something. Now, Intel has being doing the same thing where their branding means something but it isn't the same consumer market. Intel has been reaching some different audience than that, and it wasn't clear to me then, that the Intel brand was necessarily going to translate into the general population the way that Sony and others had already achieved.

0:47:16 PE: Well, there was that whole Intel inside campaign. I think it started right around the time you went to Intel.

0:47:21 BC: Yeah, correct.

0:47:22 PE: Very successful in making people aware which computers would run on Intel chips and which would not, but that's probably as far as it went.

0:47:28 BC: Yeah. And it was considered.

0:47:30 PE: What do you think, is the number of the chip [0:47:32] ___

0:47:33 BC: Exactly.

0:47:34 PE: Except it's made by Intel and it may not have sold any computers. It's just, it was the brand name.

0:47:41 BC: It probably did. Intel believes that was a hugely successful campaign and that's important because they spent a lot of money on it. A huge amount of money. They think it was a success because they were trying to avoid a commoditization effect where you don't actually care whose chip is inside the box. You personally probably don't care here what's in here as long as it does what it supposed to do. Intel saw that coming and said we want them to care. We want them to think that's it better if it has our chip and not someone else's. And so, we had to draw their attention to the issue. That's what the campaign did. But it was clear that over the long run, the trend was lower prices, more performance, lower prices. That's a path towards commoditization. And at that point, you don't care, no one cares. It's just like a cell phone today, no one knows what the hell's in here. Well, I do. But normal users, they don't care and nor should they. There's no "ARM Inside" on the outside of that cell phone.

Final year at Intel; Consumer Products Division; parting ways

0:49:12 PE: Okay. So we were talking about your last year at Intel. Decided to leave the microprocessor group, and where did you go?

0:49:23 BC: I was thinking that Intel's job was to avoid commoditization, while simultaneously preparing for the day when we had to compete on the open market, with people like Sony for example. Because it just seemed to me that that battle was looming. That hasn't actually played out quite that way, although Intel has started competing with people like Qualcomm and ARM, which are the cellphone guys. But anyway, I thought sooner or later if you want to maintain value, you have to be known, your name has to be known to the consumer, the average consumer and that was the essential tenet of the Intel inside campaign. I didn't invent that, I was just extending it. I thought okay then, how is Intel going to climb this learning curve, because right now, we don't know jack about how to reach the actual end users, other than for computers. And we've already got products in that market with some other division. I then became aware that there was an Intel group called the Consumer Products Division, that was only a year old. So far, they had designed an MP3 player, wireless keyboard, wireless mouse, a computer microscope and a couple of other things I'm forgetting right now. They were all consumer branded as Intel and so it seemed that there must be someone else thinking the same way I am. Maybe I should join that group; maybe this is Intel's foray into consumer electronics base so they can learn how to do it, learn what the rules are, in preparation for the day when we actually have to compete here for real. I joined that group as their CTO for the last year I was there.

0:50:59 PE: Chief technology officer?

0:51:02 BC: Yeah. And I quickly became disabused of the idea that they were in this doing exploration of the consumer base. That wasn't it. They didn't know *what* they were doing. They didn't have a mission. They didn't know what their goals were. I kept trying to pin them down. Are we here to make money? Are we here as the advance guard of a consumer spearheading effort? Are we here to sell computer chips via, yet another demand drive, driven thing? No. None of the above. We don't know. For example, they designed a really nice computer microscope. I think it was a nice piece of gear.

0:51:36 PE: What was that?

0:51:37 BC: It was a plastic microscope that could do 30x and 200x and some other number and it had a USB cable to your computer and you could take pictures. Whatever that thing was seeing, you could take a picture, put on the computer and it took pretty high quality pictures. It was a damn nice design, and they charged about $149 for it. They went out of their way to make like an experiment booklet. Here kids, if you want to have fun with this, do these experiments. It was done with in conjunction with Mattel, a partner at the time. But the product was a failure, and I think that what happened was very few kids are geeky enough to think playing with a microscope is fun for very long, no matter how the good booklet is. Because what you need them to do is go through the booklet and say "this is incredible. I need more things to look at." Most kids don't and so, it's not a cheap toy at $149. When I got there, the first thing I said was "Okay, so maybe as a toy, it's not going to set the world on fire but classrooms everywhere need these. This would be wonderful in a Science classroom for K through 12." And they said, "Yeah. We haven't been having very much success with that because they only have Apple Macs in the classroom and Macs don't do USB." And I said, "Well, you're just one driver away. Get somebody write the proper driver and you're there”. And they said, "Oh, but we don't want to encourage the competition." And I said, "Look, are we in to this to make money or what is our mission. Tell me the mission and I'll know how to answer."

0:53:18 PE: Besides, then look what happened.

0:53:19 BC: Yeah, exactly. It's like you're not going to stop Macs from classrooms just because you didn't give them the microscopes. They just had this fundamental confusion as to what their mission was, so they would do contradictory things that made no sense. They designed this great microscope, but they didn't know what to do with it. Then they designed a fairly poor wireless mouse and keyboard. Poor in the sense that if you turn the mouse’s laser on all the time, the battery is going to die, but if you turn it off too much the laser's going to be off when you grab the mouse. So you have to do something clever so that it can tell when you you're touching it, moving it, and turn on the laser only when necessary.

Then you’d get the best of both worlds and it works well. They didn't do that level of engineering. They got the thing barely working and said it's done and both the keyboard and the mouse had that kind of half-baked characteristic. It was expensive. It didn't work very well. It killed the batteries quickly. That wasn't going to work. We also had a MP3 player, with 128 MB of music storage in it. At that time 128MB was huge. It was twice as much as anyone else could do, because Intel owned the fabs that made the flash. So they had this built in advantage for awhile, and figured that when they went into the MP3 business they could make the Intel MP3 player have twice the capacity of anyone else, and surely that would be a large enough commercial advantage. But my first day at work as CTO I thought, well I'm going to prepare myself on this because I don't know much about consumer world. So I went down to Fry's Electronics, do Fry's? So, I went down to Fry's there in Wilsonville and I looked at the competitive offerings. Kids these days think there were no MP3 players prior to the iPod; not true whatsoever. There were dozens and dozens of them. So I looked down and I saw Sony had approximately 10 different models. Panasonic had 12 and Sanyo had a bunch and I looked at it and said we have just one. If these guys are my role models and they know how to compete in the consumer market, it must follow that they think you have to have a lot of different offerings to form a product line. We start with one. How the heck are we going to prevail with one?

0:55:17 PE: Sort of like Wendy's thing from yesterday

0:55:18 BC: That's true.

0:55:19 PE: That's three kinds of everything.

0:55:21 BC: That's a good point.

0:55:22 PE: And then, go to two.

0:55:23 BC: So, they got this Intel MP3 player working but it couldn't even compete with the current crop. Let alone what came later when iPod took over the world, because Apple understood that most people are not ready to go ripping all their CDs. Most people won't do that. They need a source for buying the music that fits on to the player so they don't have to mess with it. And then, if you give them nice category or catalog of songs and artists and albums, they'll even fork over money for that, so you can create an infrastructure and environment that the iPod user lives inside. No one had done that before. These other MP3 guys were just saying "I'll sell you a player. Good luck to you after that. I don't know what you're going to play on it, but…" That business proposition did not work in the consumer space. Anyway, the point was I reviewed the list of the consumer divisions' products and I thought, "That doesn't work. This doesn't work. That doesn't work. What are we doing here?" I spent the rest of the year trying to inject more sanity into the proceedings, trying to get everybody to adopt the same mission. "Are we here to make money? Are we here to sell x86's, or what are we here to do?" I never did get a clear answer, not from anybody, not from the bosses, not from my immediate boss, not from any workers. Nobody knew. So at the end of like several months of this, I said "okay, this is ridiculous. This division is going to die. It's got to". I don't want to be part of it. The heck with it, time to go.

As I was leaving, this is college courses management-101 right here, I told my boss "Look, it's time for me to go do something else outside the company. I'm not mad at Intel and I don't want anyone here to think poorly of me. I'm not leaving because I got mad at the engineers or the management. I'm still in a position to do Intel a lot of good and you're in the position to do me some good.

Here's what we should do. You've already announced you're going to lay off 8,000 people. Make it 8,001 and then I'll get a severance package out of it. We'll part as friends. Everything is going to be great. I still go to college campuses all the time. I'm going to tell them good stuff about Intel because I don't have any bad feelings about the company". My boss says "well I don't think so. We don't encourage Intel fellows to leave, so I don't think it's going to work, but I'll talk to my boss, Paul Otellini. He's currently the guy running the place. And he came back and he said, "No. I don't think it's going to work."

0:57:50 PE: Otellini.

0:57:52 BC: Correct, yeah.

0:57:53 PE: Paul Otellini.

0:57:55 BC: Currently the CEO. No, I never had any dealings with Paul directly on this so I really don't know. This is just inference as to whether he was involved in this. But the messages started getting more and more strident coming back. "Hell no. No, we're not going to do that". The time was approaching when I said I'd going to walk out the door. And so finally I asked one last time , "Is that the final word? I'm just going to walk out on the street with nothing." And my then GM boss says "Look, you want to quit, quit. You think you're special. You're not special. We can replace you. You know where the door is, it's over there. You want to leave, leave." I said, "Tell you what, I'm going to pretend you didn't just say that. I've managed far more people than you have. What you just did was a mistake. Now, I'm going to let you get away with it because I'm obviously aggravating you in some fundamental way and I don't mean to. But you don't want me leaving on bad terms. You just don't. It's not good for you. It's not good for me. I'm trying to just get you to see that." And he just shrugged and walked away and that's the last I ever saw of him

0:58:54 PE: Amazing.

0:58:55 BC: So, yeah it was like that. After I quit Intel, guess who calls me the most to offer me consulting jobs? The guys who want to sue Intel because they figured this Colwell guy knows stuff we can use. And I have turned every one of those jobs down because, on the one hand, I think it's morally wrongly to use things that I learned there against that company, and two, I've got a lot of friends there. I'm not going to work against them. It just didn't feel right. It would be a conflict of interest, I'm not going to, my policy is I'm not going to work adverse to Intel, but Intel is so paranoid. They tried for years to get me to sign something that would say I will never work adverse to Intel. And I said, "No guys, you don't want that. You want me to not *do* it. You don't want a piece of paper. What if I signed your stupid paper and then did it anyway, you're going to sue me. What good is that going to do you? It's going to cost you more money than you'll get out of it. You're out of your minds. What you want is my goodwill and I already offered that to you so quit throwing it in my face." It was just weird. They only seem to understand adversarial relationships. Any other kind of relationship, they're not really sure. They can't really trust it. It's crazy.

1:00:03 PE: That sounds like a really disappointing note to leave the company on.

1:00:06 BC: It was. Yeah, it felt like a direct slap in the head. I thought, it's not that I needed to get full credit for everything. But I did make major contributions to a product line that's yielded over $30 billion a year for 10-year straight. That's not nothing and now, it's 15 years actually.

1:00:23 PE: Yeah.

1:00:24 BC: But no, they don't - it's like what have you done for me lately. This is just one guy who was the creepy one. I think he gave Paul some kind of distorted message, "Colwell is threatening to blackmail us if we don't give him money". It's not what I said. I just wanted them to put me in the retirement pool with everyone else and I'd benefit like they do. But no, they couldn't do that. Here's some irony: I mentioned Wisconsin is suing Intel now for a lot of money and they asked me would you please work for us and not get paid because your credibility will be higher that way. I said "Considering how I left the company and all the creepy crap you've been doing ever since I left, you're asking a lot. But because I'm so irked that these Wisconsin reprobates are doing this with my recommendation letters, yeah, I'll do it". So I did the deposition for free. I would have done the trial for free, too, had the case not settled. But yeah, that's what you get with big companies you know, some reasonable people but the machine as a whole is just a machine.

1:01:23 PE: Yeah. I had that experience myself. It's that institutions are not human beings.

1:01:28 BC: Yeah.

1:01:29 PE: We have loyalty in the long run the way a friend does. Well, anything else you want to say about your time at Intel? Any people we haven't talked about that were important?

1:01:51 BC: These teams were huge. So you know, I knew a heck a lot of people and some of them were a great deal of fun. There was a guy that was really into music and so every few weeks we would get together and compare notes on who he would be listening to and check out this CD or that one. That was fun. It’s little things like that really stick in your mind, or things that people do for you that they didn't have to and you are amazed when you realized what they just did for nothing and you realized wow, he'll remember you. And then there are others… I remember one night at 3:00 in the morning on a Sunday. I was debugging some RTL code and of course, remember the architecture attitude that nothing stops you? Well, I was walking through other people's code fearlessly and thinking that if the bug is in your code, well by golly, I'm going to track it down and nail it if I can. And so I went into this guy's code, I spotted the bug. I nailed it down but I wondered. He's sometimes fairly prickly about code ownership. I wondered if I should call him now or wait until Monday because by Monday this thing is going to be nailed in. There's no chance to change it then. It will be become part of the official records. I'd going to call him now and tell him something is messed up, so he can feel some ownership in the resolution. So, I did that and he wasn't at all angry that I called him at 3:00 in the morning on a Sunday.

And in retrospect, it was really good that I did because on the one hand he grilled me, "I know that code, you're injecting a new bug into my code". I said "No I don't think so, here's what I did, and so on". Says he "Oh okay, you're right. Send the code to me. I want to look at it". He looked at it right then and there. I don't mind you feeling a right of ownership because you're gonna get the job done. I can handle that; in fact, I prefer it. Yeah, there are a tremendous number of different personalities involved in these design efforts. To some extent blending them together into a team is part of the magic because it doesn't always just happen. For a lot of people, it does; I can't imagine a team that would be unable to get along with somebody like Glenn Hinton. Glenn is just a great guy to be around. I think David Papworth is too. For somebody at that level, to not have an ego problem is remarkable and those guys don't. It's not that they don't have an ego. They know they're good at it but they don't use that as a stick over anyone else's head. No “shut up, I'm the king". It doesn't happen. So it was really a blast to work with those guys.

And there were some managers too that I thought were quite talented, including Randy, but he wasn't the only one. I actually felt bad for Albert, Albert Yu. I mean he did a pretty good job through the 80’s and at least part of the 90’s. He made a heck of a lot of money for that company. It's just that I felt like he got overwhelmed gradually by the total numbers of things happening and the intrinsic complexity of things like Willamette and P6 and Itanium. I think decisions started to suffer from that. There was also one other incident that I don't remember if I stuck in the book or not. I won't say the guy's name, because it doesn't really matter. But I had gotten a PowerPoint presentation via email and it said," I've been requested to send this to you for checking, because I'm going to show it to the executives next week and try to convince them to do some course of action." And I read the PowerPoint and I replied "I don't agree with this, and as you and I both know you're only presenting one of the possible alternatives and there's four possibilities. You're hiding the fact from the executives that's there four choices here, not just one, and you're only showing them one you want, which happens not to be one I think best. I'm not going to bless this. You're crazy".

So the guy shows up in my office and says, "You are thoroughly deluded if you think that the right way to run a company is to have the technical guys present all of the possibilities and ask the executives to choose one number. You can't do that. What you can do and what you need to do is show them as much as they need to know to make the right decision and no more". And I said, "So in effect, you're delegating to yourself the decision making ability here without even telling them that you're doing it. That's what you're doing." I said "If I'm ever a CEO and someone does that, I'm going to fire them on the spot, because that's lying and it’s extremely dangerous to the company." He said, "No, no, no. You don't know how to be influential with these guys. You can't give them the ability to screw up because then they'll screw up." I was aghast at that whole way of looking at it and partially because he wasn't entirely wrong.

1:06:26 PE: He does understand how to have power in this situation.

1:06:30 BC: Oh, this particular fellow was very effective. I mean that was his best thing. In fact he used this as an example several weeks later to come to me and say, "I have to counsel you, and give you some advice, because you're not as effective as you should be at these levels for various reasons." And I said, "If you measure effectiveness as making them do what you think they should do, you're right because that's not my goal , I want to achieve something bigger that anyone of us can do alone and so, they have to have the facts at their disposal to help me with that. Now, if it takes a little extra effort so be it and if it exposes us to the risk that they'll think something that I don't like just because I didn't hide it from them; well so be it." I still think that's a better way to do things but this whole way of interacting with upper management had never dawned on me until he carefully laid it on me. There was also, you read about the bag checks thing probably?

1:07:32 PE: Yes.

1:07:32 BC: Yeah, I really hate institutionalized stupidity.

1:07:36 PE: You knew your bag would be examined at the exit to the company and your way out the door.

1:07:42 BC: Yeah.

1:07:45 PE: To keep anything from leaving.

1:07:46 BC: Yeah, well, that's the ostensible reason but the real reason is because that's just because they always did it and once it becomes institutionalized no one questions it anymore. And in fact, it has a huge over head and cost that no one thinks to question. But you know, I can't stand crap like that. It just grates on me. Stupidity grates on me.

1:08:02 PE: I think about this kind of thing every time I have to fill out a form somewhere that asks for all kinds of information about me that these people probably don't actually need from what I'm trying to do.

1:08:13 BC: What if they already have?

1:08:15 PE: If they already it, that's even worst.

1:08:18 BC: Yeah, I really detest it when I call the airline to get refunds or something and tell them my frequent flyer number and the flight and so on. All this detail. And then, they say "Oh, you need to talk to so and so." Click, they're gone. And the next one goes through the same procedure. Don't you have computers? Tell it to send you what I just said to him. Institutionalized stupidity.

1:08:39 PE: We were you know, I told you we were on vacation on France over the summer and my son had an ear infection so I took him to a French doctor. The first thing she did was to make him take off all his clothes.

1:08:53 BC: Just because she wasn't sure where his ears were?

1:08:55 PE: Well, just because that's what they do. French doctor's office and that's just part of the procedure. But it's like what does this have to do with an ear. The infection's not on his knee or something.

1:09:11 BC: Well, I don't know. We can, if something else occurs we can bring it up later.

1:09:15 PE: Okay. All right, so we'll stop here on Intel.

Part 5 (2001-2009): Recent activities: writing, consulting, legal work

00:03 Paul Edwards: Okay. So this is... I don't remember which number we are on. I think it is number 5, Part V of the interview with Bob Colwell, Paul Edwards interviewing and it is August 25th, 2009. We finished up with Intel and that brings us to what you have been doing since then, which is many different things. Writing columns, consulting, working on legal cases. What has been the most interesting thing you done since then?

Hobbies: analog synthesizer restoration, woodworking, others

00:38 Bob Colwell: I think the most interesting thing what I have not actually finished yet. There is a quite long story and may not be of interest to you. You can stop me if it is not.

Bob Moog was the analogue synthesizer guru during the 60's. Somebody who was a friend of his is a keyboard player here at Portland, back in the 80's. This guy's name is Gregory Kramer, he lives here in Portland, and in 80's he went to Bob Moog, his buddy, and said I am a keyboard player, I play piano and I am envious of the way violinists and guitar players can put some English on the note they are playing. You can get a little vibrato with your finger. And it is pretty important, for violinists for example, and the classical guitar players still do it. Gregory said I can play a key on the piano and wiggle my finger but nothing audible happens. I want something to happen. Can you please help me? Moog said, "Sure, I can design something that will do that."

So Moog designed a mechanism by which you put 100KHz radio into the key and it leaks up the four corners and when you touch a finger on it, it leaks out proportional to where you're touching the key's surface. The four corners each lose a different amount of the carrier and the closer you are to a corner the more that corner disproportionately loses, computed by simple algebra. Moog designed a scheme for doing this and built a prototype. That prototype's currently in my garage, built around 1983 or so.

And then 25 years went by and both of them got busy and did not get back to it. Then about five years go now Gregory Kramer got back to Moog and said how about that keyboard. I got the prototype but I want the full up thing and Moog said sure. So he started building the full up thing but he died, just a few years ago. He did not finish it. So, Gregory Kramer still wanted his keyboard. He approached some people here at Portland area and tried to get a team to get together and finish the job. And they first tried to get Portland State University to help with it but I think senior design students are not really up to the task. I had to redo some of the stuff that they had did and fix what was not working.

I got the prototype working again and I engineered a new interface, since Moog's original interface was to a 1983-era IBM PC. I did not want that interface anymore, even if could I get it working. I interfaced the prototype to a modern Linux box and got the prototype working again and then started scoping out what it would take to build the full up keyboard. That project is still going on now for five years. Here is an example of what stops you on a project like that.
[pause]

03:29 BC: What stops you is, in 1983, if you are going to design an analogue synthesizer, you likely will use a chip called Curtis Electronic Music CEM3340, 3360, they had a bunch of different chips and they performed functions like analogue multiplication. You could design modulators and so on and everybody used the same chips. Those chips have now been obsolete for 20 years and yet there is still high demand for fixing old keyboards that use those chips. You can still get them for $90 each on the black market (from people who have stripped them from junked keyboards). So as part of this effort I actually reached Doug Curtis himself and I had dinner with him with one night, and asked him what can we do here. And he said he no longer had access to those chips either. Did not have any, couldn't make more, can't help you. Unfortunately, Doug Curtis died shortly after that. For a while there was a pressure to redesign Bob Moog's circuits to use more modern techniques, more modern silicon, but somebody at Moog Music found a in one of his labs that was clearly associated with this project and in it were 90 of these chips. So the project is back on after a year of wondering what to do.

04:54 PE: Who's we?

04:55 BC: Me and a guy named Gordon Hoffmann, who is another local guy who does hardware, and is interested in the subject. Mark Gross helps with the software. Gregory Kramer is the instigator and a guy named Charles Sorgie has thought about how one would use a keyboard that gave you five degrees of freedom off of every single key in real time. Not obvious how you would do such a thing. So it's an interesting project. I love the intersection between the electronics...

05:24 PE: Sort of a combination of the piano and Theremin.

05:26 BC: Yeah. You get fully generalized data output, so you could make it do anything, change pitch, change vibrato, change instruments entirely. I think it is going to be an interesting experiment to find out what one does with a keyboard like this one when it is finished. I have a suspicion that what we'll discover is you don't need and cannot use five degrees of freedom out of every key and instead what we really should do is simplify the whole design and use MIDI as the output. And right now it is not MIDI, it's whatever I design as the interface. And you cannot stuff this many bits into a MIDI pipeline. So that's why I'm thinking I bet you can simplify this and not really lose anything. We will see. Anyway that's been one of the more interesting projects that I've worked on, even though, it is free, not a paid gig.

I am working on a project right now, which is also just for fun. The idea is helping water the yard. Every spring Ellen comes in and says she just put a bunch of hanging baskets in the yard, and if you don't start watering them they are going to die. What she is really saying is that every spring we turned on the sprinkler system and find that it is broken. So I have to get out and fix it. To do that, you go to the garage, you set the mechanical sprinkler controller to zone 1, turn it on, run to the backyard, realize zone 1 wasn't the right one, run back, turn it off.

When you do this over and over and over, it gets really old. And last summer, two years ago, as I was doing this I suddenly stopped and said what I really want is stand in the yard and do this: zone 1, zone 2, zone 3. Then I realized my iPod Touch can talk to my home internet, it is on my network, it is wireless and it has a browser. So if the sprinkler had a host for the browser and URL in it, a TCP/IP stack and so on, I'd be in business. So I designed an interface card with 16 relays on it, and a Rabbit Semiconductor module that knows how do the wi-fi and finished the project in 2010. I published this design in Nuts and Volts magazine, January 2011.

07:34 PE: I predict that this will not be a huge seller. But it is fantastic little project.

07:39 BC: It has been lot of fun. It’s been a hoot! So anyway that was, I'm working on that and that's part time. On my to do list is to record music too. I had been intending to record CDs with my guitar for a long time, and have never gotten around to it. This project I did, this took a long time to do all this wood working.

08:01 PE: This is the bookshelves and desk in this office.

08:05 BC: And therein lies another story. You see these raised-panel doors. Those particular two doors right there. When you go to build them, you realize there is a gap on the left, there is a vertical stile, there is a panel, another vertical stile, a gap, a vertical stile. You add up all these pieces together to figure how wide the gap is. The idea is how do you size the various pieces so that when you put them together it really fits. At first I tried to do that with fractions by hand and I was getting a different answer every time I added them up. I cannot add 13/16ths to 27.90 inches, it just was not working out. Then I thought “what are computers for, anyway?”. I’ll just use a computer. I got myself a spreadsheet and was very cheered to discover that it knew how to handle fractions, up to three digits in each numerator or denominator. I just added all those measurements from earlier and I made the spreadsheet tell me how big each piece needs to be. Then I went to the woodshop and I cut all those pieces, which takes a whole day for the cutting, the routing and everything. And then you glue it up and it has to be clamped for a couple of hours. I started in the morning but late that evening I had pair of doors built. I still don't know if they're going to fit till I mount them. I mount them on the cabinets, I close the doors the first time, and they overlapped each other by an inch and half. What did I do wrong? I looked at my spreadsheet, and there’s the bug, easy to spot now that I know what to look for. If it just been a half inch of overlap I could ripped a little bit off each side and fixed it. But this is too much and would look funny if I tried that. Next day, I go back to garage with my fixed spreadsheet and do the whole thing over again and that night I mount the two new doors, close them, and now they have a *gap* of about an inch and half. The neighbors could hear me swearing. Second bug in my spreadsheet. I was just about to tear my hair out. The doors you are looking at came from the third days' cutting.

09:53 PE: Oh, God. That must have been frustrating.

09:54 BC: Yeah, it was. But what I learned from that experience was to make the spreadsheet self checking. You can stick independent checks in the spreadsheet that says there is an invariant here. For example, the sum of all these pieces cannot exceed the size of the width of the cavity and that can't be a gap any smaller than some minimum. Eventually I got it right.

When I started this huge woodworking project, I could find books on how to remodel a kitchen or your bathroom, but not your study, no one's has written one. I thought I could do that. I don't think it would pay well but it might capture at least what I think I learned from this project. That was the other thing, I still consider myself an intermediate woodworker. I have seen people doing incredibly good woodworking and my hat is off to them, but places like Fine Woodworking magazine are geared to the best woodworkers in the industry, and a lot of times I can't use like that knowledge. I can't work that way. I do not have that tool, I do not have that background. I cannot sharpen my chisel to the seven angstroms that these guys do. So I was thinking, I bet there's actually a need for reaching the intermediate woodworker, I still want to do something. And I am thinking of pursing that at some point see if really a such thing. Anyway that was those. What else. There were other projects too now and then. Like Intel occasionally would hire me for doing contracting some specific...

11:55 PE: You work for them still?

11:57 BC: They are my friends. Intel does not do it. It's my friends who will call and say, we want you to do this, will you give us some time. I have done a bunch of reviews for books. I actually don't mind doing those. To me it is fairly easy job. It's the kind of thing where things just jump off the page without much effort. If it is a typo I will usually see it. And I figure if I am reading and I stumble over a sentence twice, that usually means there's something about that sentence that did not quite work for me. I think I'm a pretty representative reader, so I'll bring that to the editor's attention. Again, you can't pay the bills doing this, but it is easy, and kind of interesting. What else did I do? There is a whole bunch of little jobs here that I have done there for various people.

Legal work: expert witness appearances, patent system

12:53 PE: Let's talk about the legal work. I mean you said yesterday, of course you cannot talk about the details of specific cases, but just tell about you. I mean you must have been involved in some of these while you were at Intel.

12:53 BC: Yeah, I did.

13:08 PE: You've done various things in lawsuits, but you have also done expert witness work a number of times since then. What's that like?

13:20 BC: Well, I think design is one heck of a lot more fun. I'll just say that upfront. When you design something that people use and express satisfaction with, that's just as much fun as it gets for a design engineer I think. It validates your conception and your execution and if the customer is happy with it, then okay I did it right. With the express witness stuff, you end up doing a heck of a lot of work writing reports, getting deposed, and going to trial. Then the trial comes out the wrong way, and guess how you feel then. Pretty bad. There is no fun. That's happened to me twice now. I have been to trial three times, lost twice, and had the case settle once. There is no real job satisfaction associated with that. And then most cases don't go to trial, they instead settle.

14:15 PE: And you do not have to influence or not.

14:17 BC: Exactly. They do not tell you what the settlement terms are. Sometimes the lawyers will say things like, your work was directly instrumental to making our client reach a conclusion that they actually like, so they're really happy with you. But that only goes so far. Without putting numbers on it I do not how much to trust that scenario. Anyway yeah, I'd say the job satisfaction part is relatively low. The expert witness business actually is more instructional than I expected. I have done cases with PCI buses, or cache coherence issues or whatever. Halfway to the trial I'll know a lot more about these topics than I ever did at Intel. Because at Intel I would go only so far into it, to the part where I could delegate to people who would expedite it from that point on. But here I'm the only guy they've got and I've got to go all the way down and chase it to the end. I've learned quite a bit doing these cases but yeah job satisfaction is lacking. The other thing is I only take the case if I believe in my side. Lawyers don't actually have that luxury, it's an adversarial system, and someone's got to be on both sides of it. But as an expert I won't take a case if I think that the guy's a troll. I think patent trolls are killing the system.

15:36 PE: What's a patent troll? I actually have this but let's put it on the

15:40 BC: Oh an example of a patent troll is somebody that buys up a patent that's reaching the end of its design lifetime, , maybe it's got two years left. The usual reason nobody wants that patent is because it isn't very valuable. But if you're a patent troll you say "Well, what's the harm if I take this patent and just go to IBM and accuse them of infringing my patent". IBM will check, and probably conclude they don't infringe the troll's patent, and the truth is they don't, but the patent troll guy is going to just say pay me a few million bucks, and I'll go away. If IBM refuses to pay, then the case will go to trial, and end up in the hands of a jury. Well the jury is generally not prepared to second guess the patent examiner, so they generally won't find the patent invalid, and in terms of infringement they would have to understand both the technology of the patent and the technology of the accused device in order to make an intelligent comparison. In point of fact they understand neither one.

So I don't know on what basis the jury can make their decisions on but it can't be technical. Of course everything you say as an expert witness is going to be directly rebutted by the guy on the other side. They can always find somebody to say what they want them to say. Pretty much the only satisfaction there is, is if somebody gets sued over some bogus patent, someone has to represent them, in terms of the lawyer and as an expert witness. Someone has to help them and then I feel like it's useful and I'm going to give them the best effort I can. I'm certainly not just in it for the money. Wouldn't do it for free but it doesn't mean I'm going to work for the other side either. It pays the bills.

The one thing, if you had to fix just one thing in the patent system, for me it would be this, and it would put us in line with all of the other countries in the world. We're the only ones that do this thing where we say, "yes, when you file for a patent that day is the important date" but, if necessary, you can argue if you lose that, and some other guy says he's patented the same idea, you both patented the same idea and he did it before you, if necessary you can still say "ah but wait, I conceived of it before you did, so therefore your patent is invalid and I should have gotten it". We're the only ones on Earth who let that game be played. Everyone else just says "hey, you filed when you've filed, we're not going to get into this ‘my lab notebook is better than your lab notebook’". Two of the cases for which I went to trial hinged on this exact thing. Because it wasn't a case of who filed first. They were claiming "Well we conceived of it before you" and that's just a horrible grey zone. I don't know if we'll ever fix that but that would help a lot I think.

19:11 PE: Yeah there's a lot about the US legal system that could use some work. [chuckle]. Let's see, speaking of patents you've got 40 of them. And you were talking, you said earlier that about 35 of those were while you were at Intel, tell me a little bit about that process and what you think some of the more interesting patents were. And when I say the process, I mean, so did Intel just kind of do this on your behalf, was it something that you had to initiate, what was the patent application process like?

19:44 BC: It was probably, the story I'm about tell you is probably not representative, but here's how it actually happened for us. My division's IP lawyer was a guy named Richard Calderwood, a great guy, really cool, we really liked him.

20:00 PE: Calderwood?

20:00 BC: Yeah Calderwood.

20:01 PE: C A L D E R W O O D?

20:03 BC: Correct. And after he'd been there for a couple of years, just in passing one day he said, "Bob, for the amount of effort and work that you guys are doing and the amount of money Intel's investing in this design team you should have a lot more patent applications than you have. Why is that? Are you guys not as creative as you're supposed to be?" What he's really doing is he's goading us into doing something and I said, "Calderwood, we are the greatest design team you ever saw and if we get together for an afternoon we could inundate you with ideas for what should be filed as patents." He said "Oh yeah? Name that tune". So we actually met for an afternoon, identified about 275 ideas that would be patentable, and that's what became the bulk of the P6 patents.

Richard's little ploy worked brilliantly, so he employed an army of outside lawyers to come in and help us write the applications and so on. But then an unfortunate thing occurred, which is, it came to Richard's attention that we were offering to sell a simulator for the P6 micro engine, and it was a pretty accurate simulator, and Richard said "You know, some miscreant could potentially make the case that in offering to sell that simulator, we were exposing the IP buried therein, in a legal sense, in a sense that you had better file for a patent within 12 months of having offered it." We had first offered it about 7 months prior, which gave us 5 months to file all 275 patents. Now we ended up not doing 275 but it was like 240, still a big number, and we went into this panic, fire drill mode.

21:47 PE: What year was this?

21:48 BC: About 94. It just seemed to last forever, every day you'd come into work and on your desk would be packed a pile, 3 feet high, with the latest stack of patent applications, and of course, each of these had to be signed. The inventors tended to be 5 of us, so those patents had to make the rounds. First they would be on his desk and then mine and then yours, and Richard was babysitting the whole thing to make sure it didn't stall out or get stuck somewhere, so yeah that fire drill was intense. We spent many, many hours reading these things and signing them but that's how the P6 patents came about.

22:31 PE: And that, those patents belong to Intel and not you personally?

22:34 BC: Correct.

22:35 PE: Alright.

22:35 BC: Yeah, you sign an employee agreement to that effect when you sign on, which is kind of too bad in some ways. Oh well. You asked for favorite patents. I can't cite any of them by name, but the patent that I thought was the coolest one was the one that's on the idea of a micro op and it in a sense covers this basic idea that you can take a complicated pile of work to do, a series of x86 instructions, and then have hardware break it down into constituent things to do, and then launch them all independently and each of them will make his way through the world on his own schedule and eventually they'll all come back together at the end. Then they'll be put back in the right order, and when everyone of them has been done, the corresponding x86 instruction has been done. And that fundamental idea exposes parallelism, because now these 5 things can march off and all happen at the same time potentially.

23:34 PE: Is this kind of like packet switching?

23:36 BC: It kind of is yeah. Back when we came up with it I was viewing it as launching little paper boats in a creek and then they would all kind of float down the creek, and some would get stuck for a while, and then somebody at the other end was collecting all the red boats, I need all the red boats, oh here they are yeah, grab them all.

23:56 PE: Exactly like packet switching yeah.

23:58 BC: Yeah it kind of feels like that. That fundamental way of looking at computation was new to us at the time. When it first came up I wondered if this paradigm would reasonably handle all possible instruction flows, because it damn well better, because it's the only one we're going to have, but it's the first machine I'd ever seen that lacked a central controller. Most machines including the Pentium or 486/386, you could identify a central agent somewhere in that machine as a finite state machine of some sort who knew what the state of the machine was, knew what all the pipelines were doing, knew what the interrupts were being requested; it was that finite state machine that was running the whole show. But it was a centralized resource, and everyone had to report into it from all of the corners of the chip. P6 could not do that; in P6 we said "we're going to split these machines instructions up into constituent atoms (micro ops) and then we're going to watch them find their own way through the machine." The first thing that should jump to your mind is, wait a minute what if one gets lost? That would stall everything wouldn't it? The answer is yep, if you lose one you're toast.

We did have bugs like that when we first started simulating the machine and implementing it. But that was actually a good thing. Because what it meant was if you made a mistake, of any sort, the machine stopped right where the mistake was. So it basically led you by the nose and said "look right here, this is the one that's busted" which was a good thing. What that led to however, I'm kind of getting away from patents, but on both the P6 and Pentium IV there was the question "what are the odds that there is a remaining latent bug in the micro engine, such that some obscure combination of instructions and micro ops causes one of them to get stuck somewhere, you get a deadly embrace, or you live lock or something and the machine stops making forward progress, what are the odds?" And if you can't say that that's literally impossible, then what should we do about it? The answer might be to validate some more until you are 100 per cent sure, but if you become convinced that you can never be 100 per cent sure then you need another plan. And so our plan was to take advantage of the fact that there are no other paths through the machine longer than some maximum number of cycles, so if we had a facility in there that knew how many cycles it had been since the last successfully retired micro op had occurred, we'd have a dead man timer. It would be like "uh, oh we're stuck" and we'd proceed to flush the speculative state.

26:34 PE: _____.

26:36 BC: Yeah, exactly, this is not good and let's roll back the speculative state to the last point where the machine was definitely known to be on track. You already have to be able to do that, that's not a new requirement, because that's how you recover from mis-predicted branches. If we did this rollback stratagem, we could work around those bugs and sure enough, that facility is in all of the P6 and Pentium 4 chips and at least at the time I was there it occasionally engaged, occasionally it would do its job. I don't know what the statistics are anymore or what it still does whether they still have it. But I just remember thinking when I first, when we first started doing this I didn't like it, it smelled funny to me. It was like if you really knew what you were doing would you have to do this, are we just doing this to cover our own ignorance. But at the end of the day you have a schedule, you have to get the machine out, you do what it takes and that's what we did.

27:27 PE: Well they're deterministic machines but our ability to understand them is limited.

27:35 BC: Yep and heaven help you when you do the cross product of all the intrinsic complexity times power downs, and you start trying to save power and the unit's not even powered up when you needed it, oh man. And then of course, you start peppering it with interrupts, traps, break points, faults; it gets real exciting really fast. The complexity is enormous.

27:56 PE: Any other patents you want to talk about?

28:00 BC: That's the only one that really comes to mind. We did one other one that did, that made me wonder about the entire process of the patent office. At some other point Richard said, "We're going to have a blue sky session" and then

28:22 PE: This is Richard Calderwood?

28:23 BC: Richard Calderwood. Yeah and he said, and I don't really remember exactly what his motivation was, but his intention was to get together the people that had been filing the most patents and purposely generate some new ones. Spend an afternoon and generate some new ideas. It didn't even have to be part of the product space, just one person feeding off another one. I'm not sure what he was expecting to get, but he actually did get a fair number of new patent applications out of that, but one of them I made up on the fly. I observed that if you have a network, sometimes each machine wants to know what other machines are up, who else is on the network and who else is sane because you could have 10 machines on the network and then one of them goes nuts, and is no longer really there, and the other ones may need to know that to react to it successfully. What kind of protocol could you cons up to allow for that? We just invented something on the fly that might work.

Next thing I know I get the patent application on my desk that has all of that stuff in it. It wasn't like deeply thought out, it wasn't research paper level, it was just, vague, "hey I wonder if you sort of did something like this" but of course when you see it written as a patent it looks like ooh this is really official stuff this must be a product somewhere. But then, of course, since then I've run into things like the guy who got the patent on pushing his kid on a swing. Some lawyer patented...

29:57 PE: What?

29:57 BC: Yeah, when you push your kid with a little bit of a push little bit of each time and they go higher and higher. This guy had patent on that and he said...

30:07 PE: You going to have to pay him if I want to push my kid on a swing.

30:10 BC: His official story is, he did this to show something bad about the patent system. I don't know what. And so far, he hasn't asked for royalties but, and then Michael Jackson the singer, the entertainer. He's got a patent. Have you seen the Smooth Criminals - there's a video. He does this dancing, coat and tales, and at one point, he stops and he leans over impossibly far; he should have fallen on his head, but he doesn't. Then he stands back up and he start dancing again. And I had just assumed it's a computer generated thing of some sort, but no he did that live. And the way he did was via his patented idea. The bottoms of his shoes had a inset, like a hole with a slot in it and he would slide it into something made with it on the floor and you couldn't really see it because it was low. But he slide his feet into that then he could lean over and then, he would just back out of it to walk away.

So my daughter at one point, when she was in grade school had, the teacher said "Everybody go find a patent and tell everybody what it is." And I said, "I've got a patent for you. Show them the video for this, they're going to love it", so that's what she did. There are whole websites devoted to bad patents by the way. Pretty frightening stuff. We also patented some of the Multiflow ideas, three or four maybe but those were pretty good. Some real good thinking led into the ways that the Multiflow machine was put together, and we tried to capture some of that. Although it led to a weird case, as Multiflow guys, they could pump a lot of money into Multiflow. And I think they somehow got the patents rights to stuff that even hadn't been done yet.

After I went to Intel, I start getting calls by some attorney that said, "We want you to help us finish applying for this patent on Multiflow technology," and I said, "Well, who would own the rights? "Digital." I said "I don't work at Digital. Why is it in Intel's best interest for me to help Digital? “I can't answer that". And I said "I don't see why I should help you". So I ask the Intel legal guys and they said exactly the same thing. They didn't want me helping other companies get patents. That's crazy. Apparently, Digital’s lawyer thought I was going to cooperate, because hey I want to get another patent but no, not really.

32:20 PE: Just to have your name on it, even thought it would belong to somebody else?
32:23 BC: Yeah. Some of these guys, such as Glenn Hilton, are up to over a hundred. He's very prolific. Andy Glew has a lot too, and that's kind of impressive. So here's what happened, this is a dynamic of a big campaign.

32:41 PE: Andy Glew. That's G-L-E-W?

32:43 BC: Right. One of the dynamics you see is, you go organization meeting - division level - where you have 6 or 700 people in an auditorium. And then the general manager, he gets up and says a bunch of things about vision, direction, you guys are great. And then at some point, they give out some awards. Most of these awards are peer to peer recognition kinds of things, so and so did a great job on a tool, saved my bacon, I love this guy. But then they get to the patent part, and they start, they'll read the name, they'll read the inventors, you come up to the front and they give you a plaque.

And of course, most of the people getting the patents are the ones that have worked there for awhile. Never the new guys because they haven't been there long enough. So the new guys watch this and think "All the people I'm supposed to think of as role models are the ones getting these patents. Getting patents must be a really good thing to do. I think I'll get myself some". So now they're set up to do the wrong thing, which is any time you have a problem, innovate your way through it. I had this problem many times with new architects where I'd say "I don't want you to innovate unless it's the right thing to do. If there's an existing solution, for God sake, use it. I don't want you inventing something new that's going to inject risk into my project for no reason."

33:55 PE: This is what you talk about in the book as gratuitous innovation.

33:57 BC: Yeah, exactly. And it was quite an interesting problem, because it just hadn't dawned on the kids. These guys are getting patents. Yes, we like patents but it's in context. It's the context you're missing. So it surprised me the first few times it happened, till I thought about it, and see that it's natural if you looked at from their point of view.

34:15 PE: Strange incentive structure developing.

34:19 BC: Yeah. And it wasn't like they give you a lot of money. I don't know what it is now. I think it used to be one thousand bucks split equally among the inventors. It's not nearly enough to change your life at all. There were rumors I heard that people like IBM or Digital would sometimes give inventors a fraction of what the patent was worth to the company. And in some cases, that was an enormously big number but, no Intel isn't one of those. So it's nice but, poor Glenn, he's got 100's of these wooden plaques, and what are you going to do with them? It's not like you can put them all up on a reasonably sized wall, and it cost Intel 70 bucks each to make them. So at one point, I told Glenn just tell them, "knock it off with the plaques, tell them to give you the 70 bucks."

Writing: IEEE Computer columns, The Pentium Chronicles, other projects

34:58 PE: Yeah. So another thing you've been doing since you left Intel is writing, the Pentium Chronicles, you've been writing columns for IEEE computer, anything else?

35:12 BC: Well, I have got more projects in mind. I wrote an article for Josh Fisher, this thing that I gave you yesterday. I wrote an introduction for David Shaw's machine, the protein folding engine he built in New York City. I wrote the foreword to a book for some people I knew at Intel; random requests show up and they look cool I'll try to do it. Hennessy and Patterson are the authors who write the classic architecture books. Well, they had a problem a few years ago where today's kids have gotten very savvy about the internet and if you assign them a homework problem out of the book, they look up the solution online and they feed it back to you, no thinking involved. So Patterson suggested that we structure the problem set in a more parameterizable way and every year we'll change just enough of it so you can't just reuse last year's answers. The structure is still there but the numbers have changed and they have to actually think. But someone has to create those problems. And I foolishly signed up to do a chapter, and oh man, it took a lot of work. I was shocked how hard it was.

36:19 PE: Things are very hard. We used to do that for classes a lot. And there's internet cheating so I guess it became just, the proportions are scandalous, enormous numbers of people. They don't even view it as cheating. It's just that, I think they're getting help but they're literally sucking the answers off the internet. There was no thinking involved.

36:39 BC: That was to engage the brain. Yeah. So I started up work but I learned a lot. What I learned was they don't pay me enough to do this. It was very hard. It took a lot of time. The only thing I'm happy about was when I first turned the problem set in, I fully expected that within a few months I'd get a nasty email from some teachers, somebody that says "your problem answer, your suggested answer is way off. You're out of your mind. You've got this dead wrong." And I'd look at it, "Oh rats, he's right." That hasn't happened yet. So I'm feeling pretty good about that or maybe they're not even using my problems, that's also possible I guess. But yeah, that was, at least it was an interesting exercise just to find out this is not easy to do. I've taught twice, a computer architecture course for Oregon State, and I was again surprised at how difficult it is to come up with a decent test. A midterm or a final or homework assignments and stuff. It was like man, this is a lot of work.

37:31 PE: In a way, it shouldn't surprise you because it's exactly analogous to your experiences as engineering where solving the problem has its own difficulties. But figuring out what the problem is, in the first place, is the most important thing to do because the way you conceive the problem determines what kind of solutions you can apply. It's just the truth that we are.

37:53 BC: Yeah. Yeah. I believe you. I guess I just hadn't thought about it before I tried to do it. But anyway, so I did that. What else was there, a few other things like that. Oh, so IEEE Computer Society, I was on their Board of Governors for year, although I was spectacularly ineffective at the role I thought. I told them that too. So I mentioned the CSTB this group that's doing the book about sustaining Moore's law.

38:25 PE: What is that, just for the recording?

38:29 BC: CSTB is Communication Science and Technology Board (I think). It's an organized effort by a quasi governmental agency where you pull together a team of practitioners and researchers to attack some problem of national interest. Back in, I guess it was 2000, I was on one of these, where we were looking into embedded computing which was then thought to be on the verge of exploding. And it actually has. There were all kinds of interesting questions about security and privacy and technical issues about how to handle buses and networks and downloading and you name it. CSTB studies scope out how big of a problem something is, where are the promising answers, what kind of research should be done. It was fun. At the end of it, I really wasn't sure if I could picture a congressional staffer reading this book and really getting enough out of it to use it.

But, oddly enough, a couple of years after that, I was in Sweden where I was asked to be the external technical guide for a brand new effort that University of Halmstad was doing in embedded computing. At the first meeting they whipped that book out for guidance in how to do this. My jaw dropped and I said "Oh, wow. I had no idea people actually read the darn thing." Anyway, we're doing equivalent study now. We're finishing up the details on sustaining Moore's Law and sustaining the growth in computing performance. I learned a lot from that as we heard from various smart people like Bill Dally and Mark Horowitz and Dan Dobberpuhl. Really good people came in and briefed us about programming methods, including NVIDIA's environment for using their graphics processors to go fast. It's a good way to keep up with the field and hear what the practitioners think is the biggest problem. So I've been doing that. I've been on lots of conference committees. There's also a group called ISAT I was part of for a few years.

40:44 PE: ISAT?

40:48 BC: And I never remember what it stands for, Information Science and Technology, I don't know. It's related to DARPA, the department of defense guys that sponsor some of the Department of Defense's research. DARPA's mission is to keep the United States from being technologically surprised the way we were with Sputnik in 1957. ISAT's job is to keep DARPA from being surprised, so DARPA looks down the road five years, 10 years or whatever and ISAT tries to look even further and more broadly. ISAT is an effort of about 30 academics and industry practitioners. It has a three year rotation. We meet once in a summer, which I was just coming back from this last weekend in Wood's Hole, MA. You spend all night and day, everyday, talking about some particular technical opportunity. These 30 people are split up into three groups approximately, and you join the one that you're most interested in and then, you tackle some problem that you think might be a useful area for DARPA to invest money in, on behalf of soldiers or sailors or whomever you're trying to help. We get briefed by really interesting people. We had two Special Forces guys last year, come in and tell us what things are like. This year, we had a guy that just got off a rotation in Iraq, who teaches at Harvard, and he gave a lecture. Another guy, oddly enough, who gave us a lecture this week was a cartoonist. His name is Scott McCloud and he has written five or six books on how to think about cartooning, what you should look for. If you're a cartoonist how do you do it. It was quite interesting. The connection still wasn't entirely clear to me but it was fascinating anyway.

When I first tried to listen to him talk, I thought, okay this was a stretch. But here's an example of where he gets surprisingly interesting. At one point he said, as a cartoonist, you're attempting to tell a story with only visual pictures and some words to carry the message and there's assumed to be a chronological time sequence to these panels. You can't mess with that because the readers are assuming it. So just within that box, you have to tell a story and entertain them and maybe educate them at the same time. Let's see how you do that. It is tricky. There's a certain technique you use and but the essence of it is that it's communication. At one point he said he'd thought deeply about this, about how do you convey a message on the face of a human being, because that's what you often do in these things. And he came up with the theory about there are six basic emotions - joy, disgust, fear, anger, surprise, something like that. And somebody took his ideas and put them on a website and now there's a slider: more disgust, less disgust, more joy and the face that moves, the eyes move, everything moves.

43:39 PE: Yeah. Just lines. Just...

43:42 BC: No. Well, it's like a sketch. It's not photo realistic but it's realistic enough. It's better than a cartoon. The idea is that you could pull a slider over and see what that one emotion does in isolation. You could also do them in tandem. You could do how about half disgust and a lot of surprise, what would that look like?

He said he was surprised that someone had done this. He was even more surprised when he heard the following thing. He said Autism researchers got wind of this website and observed that the issue is with autistic children for instance, one of the things they most struggle with is that they can't tell what facial expressions mean. They just don't have the lexicon for it. They don't know how to recognize expressions. So the question is if you had a training site for this, can you use it to teach them?"

44:27 PE: I've read about this in the newspaper.

44:30 BC: Did you? Cool. So it's the kind of thing, like I didn't see this coming, but it was fascinating to listen to it. Bing West has written several books about our current military disasters, such as having to take Fallujah twice, once after the locals hung our contractors from a bridge. Bing West wrote a book on that and he came in and briefed us last year. That's where I saw Ray Kurzweil was actually at one of these ISAT meetings. So a lot of cool stuff happens, and you get to hang out for a week with really smart people at the top of their game, tackling problems that are just beyond reach.

45:01 PE: Yeah.

45:02 BC: And then all year long, you pick a new study. You join up with it and then the group meets three or four times during the year, and in the summer week tells everyone else what we found. Now, it's August. Next month, we're going to take the studies to DARPA and try to sell them at DARPA. And then they're going to evaluate them and say "That was a good one. We're going to put some money into this", or "Hey nice idea. What else have you got?" That has been taking a fair amount of time.

45:33 PE: You do not lack for things to do.

45:37 BC: Yeah, there's a lot. You're an author. The Pentium Chronicles sold 7,000 copies maybe and that's not enough. I mean that's not nearly enough to pay the bills. I've worked it out as a buck-70 an hour. So I didn't do it for the money. I wanted to reach an audience.

45:59 PE: That's actually pretty good.

46:00 BC: That's what Wiley said but I thought "Well it was good for you but..."

46:03 PE: Yeah. No, a best selling academic book is 3,000, 4,000 copies.

46:09 BC: Yeah. At $100 each, right?

46:11 PE: No, no. Even $35, $40 books, you know, paperbacks.

46:19 BC: So, you've written multiple of them. So what drives you to do that?

46:23 PE: Well, you know, partly it's career because you need to do it to promote it.

46:29 BC: Yeah.

46:30 PE: Partly it's interest. I wanted to learn about it. ... and reputation value. Because lots of people who don't read the whole thing will know about it or learn some of the key ideas and then you get the credit.

46:50 BC: That's true.

46:51 PE: In the ecology of academia that's the most important thing.

46:55 BC: Yeah, that's a good point.

46:57 PE: The science book is probably going to do better. My first book is considered an academic best seller in that category, of maybe 4,000 or 5,000 copies. And this next book is going to be a trade book so they're thinking maybe 5000 or 10,000 copies. I wouldn't expect much more than that. Too technical.

47:12 BC: And there have been multiple instances where grad students will tell me "I read your book and I treat it like the bible. It tells me things I don't know and cannot get anywhere else." I think, well that was my intention. I'm glad to hear that.

47:26 PE: It's an excellent book. It really is. I really enjoyed reading and I felt like I learned a lot. The scale of projects that I manage is a lot smaller than the ones you manage, but the lessons still apply.

47:39 BC: People are people, right?

47:41 PE: Yeah, sure

47:42 BC: Yeah. In fact that's one of my motivations in writing it down, was thinking that there's some universal things happening here. Intel gave me some negative feedback at the beginning of the project along the lines of to the extent that what you write down is useful, it's also useful to our competitors, and we don't like you helping them. And I said, "Yeah but it's differentially worth more to you. Because it's all from Intel's point of view, you guys just resonate immediately. The other guys are going to have to translate. So I really think it will help you a lot more and so on average or on balance, that's the right way to look at it." But I got a lot of negativity from Intel.

In fact at first I proposed writing it for Intel press when I was still there. I offered them to write a book and they said, "No. No. We don't want that. Here's the list of books that we want somebody write. Hey do you want to write one of these?" I said "No. Get someone else to write about USB? I mean it's great. It's valuable. But I don't add value to that." It's crazy. I eventually had to give Intel the preprint copy of it so they could calm themselves down. They really thought I was going to trash people or give away the store. I don't know what.

48:52 PE: Yeah.

48:53 BC: They calmed down after they saw the draft.

49:01 PE: Well, we have pretty much covered all the things that I wanted to talk about. Well, I guess there is one thing that we haven't really talked about. We covered a little bit. But could you just sort of larger political contacts, the various times and places you went through, it's so hard to go to backfill on the particular subject. But, you know, your early career was probably Carter administration, Reagan administration, that period into the cold war. Your adviser in graduate school was interested in military problems and probably getting money from DARPA.

49:47 BC: Definitely, yeah. And the army, the army paid for my PhD.

49:52 PE: Did you ever have any political involvement, or were you aware of those events?

49:57 BC: I was definitely aware of them. I've never had an inclination towards the public office. I think it would kill me. I view, like that's why I read about people like Abraham Lincoln. How in the heck did he do what he did? How does someone with high personal integrity manage to live in the public sector and accomplish anything? You know, without sacrificing that integrity. Because I read things in the paper about some prominent politician having really believed strongly in issue A but didn't care about B and then somebody else believes in B so they swap votes and both of them go "yuck."

50:32 PE: I know.

50:32 BC: It's really distasteful to me. To me, if it is right, it's right. You should view it in isolation and not confound things. But then they say, "Hey, you know, don't watch sausage being made if you like to eat it." Well, that's great: you make the sausage, and I'll just eat it. But I have paid a lot of attention to national and international events for example. Because I just have this innate belief, bias maybe, that things are pointing in a direction that can be very dangerous for humanity as a whole. And what I mean by that is there was a time even back in the 50s when I was born where, if we had needed to, we could have in principle backed away from a technological approach to living and survived anyway. We hadn't yet driven ourselves to mega farms using precision, agricultural techniques that use huge amounts of fossil fuels on behalf of huge populations. It was much simpler. I told my kids this. You have to think about it carefully to see it. But back then the population of the United States was, oh I don't know 200 million, 230 million. Now, we're 300 million.

51:52 PE: Less

51:52 BC: It was less?

51:53 PE: 100 million.

51:54 BC: It's like 2 to 1 right? Or 3 to 1. And we had a lot fewer cars per capita. Nowadays, if you're a driver on the roads, you're seeing three or four times the traffic that used to be on those roads back when I was growing up. And the trends are that things are going to get worse. If there was a major disruption to the food supply to New York City, what would happen? Within three days there's no food left. There's not enough food. You have to be able to get trucks and planes and trains into that city or those people are going to die. Did people realize how thin the safety margins are and if they did, would they do anything about it? As a technologist, I look at these questions and see that the population who is most at risk with issues like this are not going to see the threat and are not going to know how to fix it if they do. I think it therefore follows that the engineers will have to, when they design a system, they have to realize they're doing this on behalf of all these people who can't second guess them. The general public is not going to help you decide whether you got it right or wrong and yet they're going to suffer badly if you screw it up.

So I think that's something that we all have to take into account. That is not taught at schools. When you learn electrical engineering not once do they bring up issues like this. They just tell you the rules of nature. Here's Ohm’s law. Here's how electrons work and that's all valuable stuff. We all operate inside of a social environment that we as engineers have to take into account and we have to realize it's an asymmetrical process. We know things that the population does not know and will not know and they need us to get the ethical parts right as well as the scientific parts. Some of that stuff is not that hard. The part where it's going to get harder is, for instance, climate change issues. And frankly, I even felt to some extent at Intel when we were doing the Pentium 4. And I was thinking here we are jacking the clock way up as high as we can possibly make it go so that we can make sure we distance ourselves from competition. That much worked. But in so doing, we knew darn well that we were beyond the efficient part of the curve. We were burning much more power to achieve that extra performance then we ever had before. It seemed to me that if we played this out in a global scale, we'd be populating the world with these inefficient machines, burning irreplaceable fossil fuels like crazy.

54:18 PE: Yeah.

54:18 BC: Is that something that I like as a legacy? Of course not. I do like selling a lot of machines. But did I fully discharge my duty as an engineer who thinks in these terms? I wasn't convinced that under the circumstances, I could have done any better but I also wasn't convinced that we had hit the right trade-off. So, it was an unsettling thing to think about that way. I was much happier with thinking of the original P6 which I think was a really good compromise between performance and power. I think the guys that design the processors for cell phones, for example, do a nice job. Those are very power efficient because those are battery operated.

And that's why right before I left the company, I was begging Intel’s top brass to get really serious about battery operated processors because I think that's the future. Their answer was "Well, as a technologist you might be right. But you must realize that profit margin on those CPUs is so much lower than we get now that it would be ridiculous to think about trading our current business for that one." And I thought, well, he's got a point in terms of finances. He's absolutely right. But in terms of littering the world with machinery that is burning a heck a lot of power for not a really good end, I'm right but what I do about that. I don't claim to know the answer. What feels right to me, for now, is asking the questions. I think if we stay sensitive to this environment that we operate inside of, we will at least be positioned to ask the right questions at the right time and get many more of the answers right.

56:02 PE: It's probably a great place to end the interview. But it's just makes me more want to tell you that at AGU last December I saw a proposal.

56:10 BC: What's AGU?

56:11 PE: American Geophysical Union. So, I'm in a poster session and there were thousands of posters. And one of them was these guys from Berkley who had a plan to build a supercomputer from cellphone chips. And the reason was that it would be low power.

56:28 BC: Low power.

56:29 PE: A green supercompute. And it would be very cheap because these things are so inexpensive. How they're going to wire them, that seems like another problem.

56:38 BC: It is another problem. But if they were willing to take the basic core and put the stuff around it that would be needed to do the networking, that would make a damn good machine. No problem. Yeah, it would be fun.

56:53 PE: Okay. We'll stop there. I want to thank you very much for taking two days of your time to be interviewed for this purpose. I'm looking forward to seeing the final transcript and putting it up on the ACM website.

57:03 BC: It's cool. Yeah, you're welcome.