10/00

CARVER MEAD INTERVIEW
By: Gene Youngblood
(Document 1 of 4 of Mead Interview)

GENE: The progression toward a speed/power inversion has been happening continuously since 1959 when the integrated circuit was invented and yet today we still can't build a supercomputer that doesn't require acres of freon tubes for cooling.

CARVER: Oh yes you can. It's just they're not doing it. You see the problem has been twofold. One, as the technology developed the big computer people -- the Amdhahls and the Crays -- stayed with the older bipolar gate array technology. They haven't participated in the revolution. Now the semiconductor guys don't know much about computers so they've copied a bunch of ancient architectures. So what we have in the personal computers is creaky old architectures with this wonderful technology. So the people driving the technology don't know anything about systems and the people who have built traditional large computer systems haven't participated in the technology revolution. Supercomputers are an extremely inefficient use of power and space. They just brute-forced it. They said we're not going to use any cleverness at all, we're just going to pour the coal to it and see how fast we can make it run. So they just turn up the knob and go for it. And you end up with these giant steam engines that blow off all this heat. It doesn't make any sense at all from any point of view. But, you see, the situation is actually much better than that. And that's the part nobody counts. I mean if you take today's techno- logy and use it to do a really novel architecture you can get a factor of 10,000 right now. Today. But you don't do it by running the clock that fast. You do it through parallelism. There's a lot more to be gained through architecture than there is in clock speed. You can get 10 GHz out of a parallel array of 10 MHz clocks.

GENE: But aren't there serious problems with software for parallel architectures? There seems to be controversy whether schemes like dataflow and functional programming can actually overcome the communication problems.

CARVER: As long as you're talking software you're missing the point. Because you're thinking of something programmable. And those schemes will never work out in terms of enormous parallelism. That's why I think this super-computer thing will turn out to be a net loss to our country. Because they're still clinging to the belief that we're going to make programmable highly parallel machines. I worked on that for a long time and finally came to the conclusion that it just wasn't going to make it. You were going to get factors of ten or a hundred and that was the end of it. Well that's not enough. We need eight or nine orders of magnitude to do the kinds of things we want to do with computers today. There's factors of a million there if you do it right. But for that you can't separate the architecture from the algorithm. You have to build that algorithm in silicon, not program it somehow. I think they'd better be facing straight into the fact that there are dedicated architectures for these enormous tasks.

GENE: What do you call the kind of architecture you're talking about?

CARVER: I call them silicon algorithms. You just map the application right into silicon. And the technology for doing that we understand pretty well. I don't mean we've work- ed it all out. But we've developed the technology for taking high-level descriptions and putting them on silicon. It's called silicon compilation, which is the obvious next step. You need to compile things right down into the chip instead of compiling them into some code that you feed to this giant mass of chips that grinds away on it trying to interpret it.

GENE: So the silicon compiler solves not only the complexity problem but also the problem of massively parallel architectures by allowing us to experiment with the dedicated hardware that alone can realize the full potential of that architecture.

CARVER: You bet. Otherwise there's no hope. Because then you're stuck. If it takes five years to design a chip nobody's going to experiment with algorithms in silicon, right? Because it takes years to get anyplace. And that's in series with your learning curve. So you have to have a technology for experimenting with designs in silicon more rapidly than that.

GENE: Then microelectronics and computer science have become synonymous.

CARVER: They have to. You see, the thing that allowed people to separate the machinery of computing from the practice of computing -- hardware from software, if you like -- was the idea that what computers did was a sequential process. The old Turing or Von Neumann idea. Once you start to do things in a highly concurrent way you can't separate the algorithm from the architecture any more. You're confronted with the necessity of mapping classes of applications directly into silicon. There's no such thing any more as the general purpose programmable machine when it comes to really high bandwidth computing -- which is the only reason you need concurrent processing in the first place. So that doesn't mean Von Neumann machines will go away; they'll be used as general purpose controllers for these special purpose instruments. The user has to talk to something that's coherent, that has a language and an operating system and so forth. But instead of that poor little thing trying to execute all those instructions, you're going to have special-purpose engines it calls upon to do these enormous tasks. So you can think of the dedicated chips as extremely capable instructions that you call from your personal computer.

GENE: Would a personal computer based on one of the new 32-bit microprocessors be powerful enough to control a such a high speed engine?

CARVER: Oh sure. The very first one of those on the market is the Silicon Graphics Iris machine with the Geometry Engine in it that Jim Clark designed. It's got a 68000 that controls an enormous graphics pipeline with at least 100 times more computation capability, which nevertheless is a slave to the microprocessor. The reason, of course, is that one small instruction can cause an enormous amount of work to get done. So the microprocessor is used to update the picture.

GENE: Even with dedicated hardware there's a communication problem in any massively parallel architecture. So people start talking about wafer-scale integration. Is wafer scale technology necessary for highly concurrent machines?

CARVER: It's not essential but it would help a lot. There are many dimensions to the problem. But essentially you're right that every time you come off the chip and have to drive a big wire instead of the kind of wires you have on the chip, you pay for that in speed. So to the extent that you can put the wiring on a wafer instead of on a circuit board, the thing will run that much faster and be that much more cost effective. The problem is nobody has yet come up with a viable scheme for tolerating the faults that always occur. None of the redundancy schemes are viable yet in terms of efficient use of the technology. It has not attracted the magnitude of research effort that it's worth. I think it's a very exciting area. But as an alternative to wafer-scale integration you can arrange the chips as islands of computation where the links between them aren't too huge. It depends entirely on the nature of the algorithm. How tightly connected are the clumps of computation? You could, for example, arrange the islands to be synchronous and where they're not connected too tightly use an asynchronous protocol so you don't have to lock all the clocks together. You'd still have a fairly small box. So even if you have to go off chip you arrange it so everything that's on the chip is tightly connected. That's the whole game of making concurrent architectures whether you're on a wafer or not. It has to be based on locality. Otherwise the wiring mess just gets completely out of hand. Incidentally, the brain has the same problem. Your cortex is basically two-dimensional. It's only a millimeter thick. That's a lot thicker than a chip but it's by no means a fully connected three-dimensional volume. People think that because the brain is encased in this little round thing it's three-dimensional. It's not. It's two-dimensional, and it's wired in a two-dimensional way. In fact there are two layers. There's the gray matter on top of the cortex which is where the processing happens, and there's the white matter on the bottom half of the cortex which is where the wires are -- a solid mat of wires about half or one-third as thick as the cortex. That's exactly what we have in silicon except we don't have quite as many layers of thickness. But it doesn't qualitatively change the nature of the problem. And we're getting more layers faster than the brain is. So the whole idea of putting priority on locality is as true in the brain as in a chip.

GENE: Are silicon compilers based on artificial intelligence? Are they expert systems?

CARVER: You could think of them as expert systems, if you like, in the sense that they capture the expertise of the designers, but they don't do it by being artificial intelligensia; they do it because some really smart people worked the problem. Silicon compilers are based on ordinary, old-fashioned, good systems design. That's a whole different

game. In my opinion, artificial intelligence has done absolutely nothing that ever helped anyone to do a chip design. They may help in the future but so far they haven't.

GENE: Will the silicon compiler become the universal method of chip design?

CARVER: Yes, for highly concurrent architectures.

GENE: Regardless of the complexity of the chip?

CARVER: Well you simply can't implement most of these algorithms without the complexity. There has to be a certain scale of integration before it makes sense. But yes, you're going to need a silicon compiler to design even a Von Neumann-style chip at the complexity level of VLSI But once you have VLSI what's to stop you from doing something a whole lot more interesting than a Von Neumann computer? That's really the point. People talk about the complexity as though it were a problem. It's not; it's an opportunity. We have to think about it as an opportunity for new ideas, not as a problem that has to be overcome so we can make one more Von Neumann computer. That's a terrible waste of a very beautiful technology. And that's what we're seeing right now. The situation is analogous to what happened in nineteenth century England after Farraday demonstrated that you could make electric generators and motors. The factories of that day were long sheds with huge steam engines that drove a rotating shaft that ran along under the ridge pole of the roof with big pulleys on it and belts down to all the machines. If you wanted to stop a lathe you slid the belt from the drive pulley onto an idler. Well, when they built the first electric motors they built enormous motors and used them in place of the steam engine. You still had the shaft and pulleys and belts. They could not conceptualize that they really ought to have fractional horsepower motors distributed around -- what we now call the parallel processing approach. But the other thing was that of course they couldn't afford to change everything. We look at that today and say well that was really dumb. But if you were living back then you'd probably have a thousand arguments for why it was the only sensible thing to do. That's where we are today in computer science. We have this wonderful technology and we're building one monolithic computer. And if that turns out to be inefficient we'll make a big one on a chip. So they talk about micromainframes. It reminds me of a cartoon of the board of directors in Detroit and one of the directors is saying "Well, if we have to make a compact, we'll make the biggest compact in the industry!"

CARVER: The reason I got into the whole sordid affair fifteen years ago is that we as a culture were stalled. The really smart innovators, who are always the little guys, were unable to get at the technology. And that's criminal. And of course the big companies liked it that way. Because it's fat lazy comfortable, right? Which is why they don't like me. I'm rocking the boat. But the important thing that has happened is that the little guy like Jim Clark, one guy, can get at it and turn the whole market upside down. That Geometry Engine that Jim Clark built is a different architecture but the same idea that I proposed to Dave Evans [of Evans and Sutherland] in 1975 as precisely the right thing to do with the technology, namely to do the transformation and clipping stuff. Here was the company that was supposed to be leading the way in graphics unwilling to do anything about it. They could sell these megabuck boxes, why did they need a cheap one that did the same thing? Finally Jim Clark did it, six years later.

GENE: As early as 1972 you were saying we'd soon reach the limits of Von Neumann architecture. And yet even today several more orders of magnitude in performance can still be gotten out of that architecture through submicron scaling and increased compaction.

CARVER: But you see the Von Neumann machine doesn't take very good advantage of that. For one thing it treats all memory as if it were equidistant. It makes everything as hard to get to as everything else. They try to compensate with things like caches, where they put some memory closer to the processor, but that's after the fact. It's not built into the architecture. So as a result Von Neumann architecture actually works against increased compaction at the chip level. It doesn't scale well at all. It's not a way you'd ever design a computer if you had VLSI technology in mind. If you want to get the full benefit of this marvelous new technology -- this new medium, if you will -- you need a whole new way of thinking about computer architecture. Everybody viewed this technology as a cost reduction mechanism for traditional designs instead of as a new medium with which to realize new classes of architecture. I've spent the last fifteen years of my life trying to get people to think in a fresh new way. And they haven't been doing that. The solid fundamental work that's necessary even to understand what's sensible to do isn't being pursued in very many places. I mean the fundamental limitations work from an architectural and algorithmic standpoint, not from a technology point of view. We're in pretty good shape technologically. There's a lot of depth

there across the board. But it's not paralleled by similar depth in the leading-edge systems area. We don't have good people in every major department at every major university doing excellent work on the cutting edge of architecture as we do in device physics and materials research. Basically that's just starting to happen now. There have been some signs of life recently, but that's after fifteen years of beating on it.

GENE: You said in 1977 that there was an eight-order-of-magnitude possibility in compaction of integrated circuits -- potentially 100 million transistors on a chip. How do those early projections look to you today?

CARVER: The original analysis we did in 1972 of how small you could make devices -- one quarter of a micron -- is holding up remarkably well. The chip sizes have also been going up, although slowly. In that 1977 report we were thinking of a chip about a square centimeter, approximately what they are today. So if you scale the devices to a quarter-micron and enlarge the chip area a little more you can get a hundred-million transistors on a chip. That's with the technology that's known today. There's not a single step in there that hasn't been proven feasible. You'd use ion-beam dry etching and X-ray lithography. That will be done in the next decade. There should be people doing it now.

GENE: How much more computing power is available beyond today's technology just by scaling everything to the limits?

CARVER: It depends on what you count. You can count computation per unit power or unit area. I think the fairest thing is computation per dollar. What are you going to get for your money in terms of cycles per second of real computation? From now on that number is going to go roughly like the square of the scaling. Ideally, when you scale everything the speed/power product goes like the cube: switching speed increases linearly as you scale down and density goes up as the square. If it's two to one in dimension it's four to one in density -- if you reduce a transistor by a factor of two on each side, four little transistors now fit where the big one used to be. But in addition to that they're running twice as fast. So every time you go down a factor of two in size you get a factor of eight -- a factor of four in the number of gates and a factor of two in speed. That says gate cycles per unit area goes like the cube also. But as you approach physical limits you don't quite get that full cube law anymore, which is why I say that from now on it's going to go like the square. So if we scale from, say, 1.25 down to .25 micron, that's five to one, which is 125 times in gate cycles per unit area and also 125 times in how much computation you get per unit power; it's probably not quite that good due to approaching physical limits, but we'll certainly get a factor of 100. And remember we're not really at 1.25 micron yet. It's really a 1.5 process today, and that's only the leading edge; three microns is the standard production process in today's commercial world. So that means we're talking about chips a thousand times faster than the ones in today's personal computer once we reach all those limits. And that's not counting concurrent operation. A computer with a thousand of those chips running concurrently would be a million times faster than today's machine.

GENE: What about three-dimensional chips?

CARVER: These things are built up in layers and in principle you could keep going. Everybody knows how to do that. For example, SOI -- silicon on insulator -- is probably the most promising way to start stacking things. But there are other ways. The problem is that the yield -- the overall probability that the thing works -- goes down exponentially with the number of layers in the process, even if it's just an insulation layer or a layer of wiring. And exponentials are deadly things. After a while they really get you. And it's not just the yield problem. As you make things smaller they also become much more susceptible to things like cosmic rays and all kinds of other mechanisms of failure.

GENE: Compound materials like gallium arsenide are supposed to be less vulnerable to soft errors.

CARVER: Every technology has its own horrible problems and that includes gallium arsenide. There's no magic. The number of electrons created by ionizing particles isn't tremendously different between materials, not by huge orders of magnitude. It's really a size question, a straight-forward scaling issue -- how many electrons represent your signal versus how many electrons are created when you put an ionizing particle through the material? As you scale down there are fewer electrons per device, so it becomes easier for a random particle to switch the transistor for the same reason it's easier for you to switch it.

CARVER: I believe that the vision of the original founders of AI -- the Minskys and McCarthys -- was extremely correct. But when they got into it they discovered that what they were really headed for took eight or nine orders of magnitude more computation than you could get out of a regular computer. And so there came to be two groups: one went looking for that eight or nine orders of magnitude; the other just punted and faked it -- and that's the vast majority of the AI community today. I'm one of the people who's off trying to find that eight or nine orders of magnitude.

GENE: I've heard cellular automata used as a general term for massively parallel architectures.

CARVER: Cellular automata are an invention of Von Neumann, strangely enough. He described them back in the forties. It was the first ultra-concurrent architecture, a really good first step in what's now become a large class of things. For example, you can view Kung's systolic algorithms as an extension of cellular automata. And if you look at it that way, it's a very natural evolution. But in fact a cellular automaton is an extremely precise thing -- an interconnected set of finite automata. Traditional cellular automata have only been connected to neighbors, and as such they represent only very local computations. Any such local computation can be expressed as a cellular automaton but it's often not relevant to do so. Very few algorithms map well into that domain. So they've kind of boxed themselves out of a lot of the more interesting views of computation.

GENE: What about the demarcations between levels of integration, like SSI, MSI, and so forth?

CARVER: To me the differences aren't so much in transistor count as in the level of functionality, that is, how you have to think about it in terms of what it's doing. The first integrated circuit in 1959 had one gate or one flip-flop, so you could think of them as elementary logic elements. Medium-scale chips had things like counters and registers. Now people could start thinking at a higher level than just gates. That was in the middle sixties. The next real step happened when memories and microprocessors were integrated on the chip around 1971. That's when LSI begins for me, not because of the number of transistors but because now it was a complete system-level function. Those early micros only had about three thousand transistors but they were a processor nevertheless and they changed the way people thought. Now actually if you look at the Intel 286 today, it isn't really more capable -- or not very much -- than those early chips. It's got a lot more transistors but the architecture's pretty much the same; the instruction set's pretty much the same. So I don't put it in a different class from the 8008. Also, according to this view the gate-array business represents a throwback: gate arrays are the SSI level and standard cells are the MSI level.

GENE: If 32-bit microprocessors aren't qualitatively different from 8-bit versions, then when does VLSI begin?

CARVER: I'm looking forward to the machine that'll do ray-tracing in real time. It's not beyond our capability right now. Provided you compiled the algorithms into silicon in a massive array. I don't believe that polygonal representation for shapes is the right one. I think stuff like superquadrics is a whole lot more sensible -- maybe not exactly that representation, but something that has some nice mathematical properties when it comes to ray-tracing. So if you're going to use .CP10 a ray-tracing algorithm you'd better use something other than polygons. In fact, you have to invent the representation and the algorithm together. That's why nobody's going to do it unless they get their head around the whole problem.

CARVER: My own work is concerned with music. As you know, if you try to simulate even one musical instrument you're in for at least 10 MIPs worth of computation for a single voice -- one string on a guitar or whatever. In some cases you get a whole instrument like a flute, but if you try to do a piano you've got to have a huge number of processors. So you get into the gigaops very quickly to simulate any reasonable ensemble at all. A whole orchestra is a lot of gigaops. And it turns out you can do that. We're right now looking at about 10 MIPS per chip for doing music. A specific architecture for doing music, not good for anything else, and you can make some of the most beautiful voices you've ever heard in your life. We'll be able to simulate a very convincing orchestra. Now I'm not saying we're ever going to replace a real orchestra, but there's a set of things you can do if you have the ability to create your own orchestra. This is what the synthesizer guys ought to be doing and can't; and the reason they can't is that you need four or five more orders of computation than is available today. That's what prevents them from true orchestral simulation. A standard synthesizer sounds like hell and it doesn't have to, but you've got to find that four or four or five orders of magnitude -- in this case it isn't eight or nine, only four or five, but it's still a lot. And we can put it in a personal workstation for use by composers. I believe that's the way composition will be done. Composers

simply cannot afford to experiment. You can't afford to pay a hundred union musicians to fool around. So composers are really stuck, just be- cause they need that four or five orders of magnitude. We're still in the very early experimental stages but we can already mix amazing voices. Most people say it sounds like a real instrument, not like a computer. Except you can transform it and move it into different spaces. We can simulate a marimba bar that would have to be 27 feet long.

End of document 1