Point Accelerators vs. General-Purpose Machines for Electronic CAD

Parallel CAD algorithms and machines span the range from the academically interesting to the commercially interesting, with a broad middle ground. The architectures include point accelerators, general-purpose CAD accelerators, and general-purpose parallel machines. A researcher in a university will evaluate these tools employing a different set of criteria than a user or supplier. In academia, one has to get funds and do interesting research; in a company, building, selling, and making a profit are the objectives.

In this discussion, representatives of academia and industry evaluate the various tools and predict their futures. Participants in the roundtable, which was held last November at the International Conference on Computer-Aided Design in Santa Clara, included Stephen E. Coit of venture capital firm Merrill, Pickard, Anderson & Eyre, Rob A. Rutenbar of Carnegie Mellon University, Alberto Sangiovanni-Vincentelli of the University of California, Tim Saxe of Silicon Solutions, and Robert J. Smith, II, of MCC, the Microelectronics and Computer Technology Corporation. Tom Blank of Stanford’s Center for Integrated Systems was D&T’s spokesman.

D&T thanks editor-in-chief Vishwani Agrawal, roundtable editor Charles Radke, and conferences editor Jerry Werner for their support.

D&T: Incorporating parallel CAD algorithms in machines obviously makes sense. There really does seem to be a consensus on that. The first question on which there might be some disagreement involves the value of point accelerators. Do they make sense for anything but simulation? And when I say simulation I’m referring to logic simulation, fault simulation, and even SPICE.

Saxe: Right now I’m not sure they make sense for much else. Simulation is one of the few unbounded tasks you run into. It is the central part of the design process. For bounded tasks, a point accelerator is nice but the tremendous amount of performance is not so interesting because you only run a big design-rule check once and you only run a big routing once. So I’m not sure there’s much interest in point accelerators today outside of simulation. I can imagine a day coming when you’ll interact with a placement algorithm or a routing algorithm if you have enough horsepower behind it so a person won’t have to wait 10 hours to see what happens.

D&T: Alberto, what do you think about point accelerators and things beyond simulation?

Sangiovanni-Vincentelli: It’s all a matter of price/performance and the kind of task you have to run. The issue of point accelerators vs. parallel general-purpose computers is very much a function of how much it costs to design a point accelerator vs. how much it costs to program an existing parallel machine. The big issue here is how much support we get with parallel machines. We have to ask the makers of parallel machines to give us the tools to program effectively—the lack of such tools is, I think, the greatest weakness of parallel general-purpose machines.

D&T: Let’s put off talking about general-purpose machines and address them specifically later. It sounds like you’re saying that except for simulation you don’t really see any benefit from point accelerators.

Sangiovanni-Vincentelli: What I’m saying is that one has to evaluate price/performance very carefully. I’m not even sure some of the parallel simulation is done best on a point accelerator. Look at circuit simulation, for example. We are very carefully evaluating how much it costs to design a point accelerator vs. how much
it costs to develop a circuit simulator on a parallel machine.

D&T: Be a gambler. Just shoot straight from the hip. What would you do right now?

Sangiovanni-Vincentelli: Right now I would not build a point accelerator.

D&T: For anything?

Sangiovanni-Vincentelli: Assume I have a new application that needs acceleration. First I would try to use an existing general-purpose parallel machine. And if I needed more performance I would add a special-purpose coprocessor to it.

D&T: More performance for what? How about a specific example?

Sangiovanni-Vincentelli: Three-dimensional process and device simulation.

Smith: For electronic CAD I'd say try to develop a multistage microprogrammable pipeline board that you can plug into one of your garden-variety commercial parallel machines. I think that when you talk price/performance you should really be talking about incremental cost and not the purchase price, because you'll use a general-purpose machine for a lot of different things. If you tell me I need to buy a point accelerator, I'll say my cost per seat goes up and I'll have to amortize the cost of the machine over the number of users. I don't have to do that with a piece of software I put on my Unix engine.

D&T: Can you expand on what you said your point accelerator would be?

Smith: If I were going to build something today to make a profit and I could not find a commercially available parallel machine—a general-purpose machine that could be OEM'd—I'd think about adding in something that would be a multistage horizontal microcoded pipeline of some sort. It would be software reconfigurable and yet exploit the general parallel machine environment.

D&T: Does anyone else have any opinions about whether point accelerators make sense for anything beyond simulation?

Coit: There are things that are technically feasible and things for which there is a need, and then there are things that make economic sense as an ongoing business. The intersection of the three is what works. It's not just the cost of the design, it's also the cost of programming, the cost of support, and the cost of selling—of convincing the customer that he should pay that much money to buy it.

I think market size is a serious issue for point accelerators. You're selling two things when you're selling a point accelerator—you're selling some hardware but you're also selling an application software package. All the economics of the application software package business applies. Application software companies have marketing and sales expenses of up to 40 or 50 percent and R&D expenses of 15 to 20 percent. They have very high gross margins and almost always have to make their packages portable because the hardware customers are running can become obsolete. In contrast, the hardware business typically runs on 60- to 70-percent gross margins and keeps sales and marketing expenses under 30 percent, ideally around 25 percent for an end-user business. R&D is only about 10 to 15 percent. So, the two businesses really don't match up very well—if you set out to build and sell point accelerators you can end up having the sales and marketing expenses of the applications package business but the gross margins of the hardware business. Consequently, you just can't reach the customer with the support he needs and you end up with a customer who feels gouged on the price or a customer who is frustrated because he is not getting what he needs.

Some of the ideas I've heard here—about more general-purpose plug-in boards and multistage microprogrammable pipeline boards—remind me of Floating Point Systems. Floating Point Systems began with a general-purpose floating-point accelerator that had a linkage to application programs but was not an application in and of itself. What we should remember about Floating Point Systems is that it took a long time to grow. The interfaces were always a pain in the neck. The applications always required conversions. From a venture capitalist's viewpoint, it was a venture that needed a long runway to take off.

D&T: We've talked a lot about marketing—how about if we look at things from an academic perspective.

Rutenbar: We've all been throwing around the word "viability" and we've all been assuming it's viability from the viewpoint of the guy who wants to get venture capital to start a company to build something like a point accelerator. There are other kinds of viability. There is viability as a research topic. There is viability as a marketable product. I think the point accelerator is viable as interesting research.

D&T: What would be the targets you'd shoot at?

Rutenbar: Electrical rule checking and design rule checking. But you have to be extremely careful. Another interesting thought about the viability of point accelerators is that they may not be viable from the viewpoint of someone who wants to build and sell them but they may...
be viable if the consumer of the CAD application is also the producer of the point accelerator. I'm thinking of the Yorktown Simulation Engine at IBM, of course. What may happen is that the point accelerator business may be totally taken over by large manufacturers who require extensive CAD. IBM or DEC may find it economically feasible to build five boxes to do design rule checking and electrical rule checking—five boxes to do 3D point process simulation—because they do not have to sell them in the open market and they do not have to make enough money on them to finance a second project.

D&E: What characteristics would a good candidate for a point accelerator have?

Rutenbar: It would follow the principle of least obnoxiousness. The obvious, the canonical complaint is that simulation engines accelerate simulation but leave flattening, netlist sorting, and preparation for simulation to be done. As a result, the end-to-end simulation time with a simulation engine may not be nearly as good as you may have been led to believe. That means people aren't spending enough time looking at what the simulation engine—the point accelerator—is going to plug into. Design rule checking and electrical rule checking currently means huge gigabyte files of flat-sorted polygon edges. So if you want to build a point accelerator for something like that, perhaps the right notion is an accelerator specially designed to do flat sorting of polygon edges, one that does so in a way that is least disruptive to the environments of the people who are using it.

D&E: Can you rank the characteristics a good candidate for a point accelerator would have? Which are most important?

Rutenbar: It should mesh easily with the customer's CAD methodology. It should offer terrific price/performance.

D&E: A stable algorithm would have to be included in there somewhere, too.

Smith: I disagree that a stable algorithm is important. My list says you need to have a system-level view that takes into account the entire user service cycle, which is the same as what Rob (Rutenbar) said. Another important characteristic a point

accelerator should have—down at the hardware architecture level—is an unusually high aggregate memory bandwidth.

D&E: Does the algorithm need to have that characteristic?

Smith: No. I'm saying the point accelerator must have that characteristic. In other words, the algorithms we want to accelerate are very data-structure-access intensive. I think a point accelerator needs to be software-reconfigurable by the vendor.

Sangiovanni-Vincentelli: I agree with Rob (Rutenbar). From a research point of view a point accelerator may be good. I think if you're working on algorithm acceleration you should build the machine to check whether the algorithms are actually working. There are so many assumptions you make. In the university I see a lot of algorithms in which certain assumptions have been made, assumptions that turn out to be false when the machine is built and the algorithms run on it.

Rutenbar: Let me address the issue of whether you should build the hardware. We all know what an incredible pain it is to actually build hardware. Sometimes you can derive insights without having to build it. For example, we had a project in which we were restructuring scan-line algorithms for some pieces of a design rule checking and electrical rule checking system. We restructuring them in the form of a deep pipeline. We did some paper design. We did some simulation. The numbers were interesting. We thought they were reasonable but we clearly recognized they were the results of simulation.

We would have liked to build the hardware but the interesting thing about doing this research—which was originally directed at the point accelerator, of course—was that it had some nice spin-offs. When we were done with this research we felt we had a nice understanding of the algorithms even though we didn't build any hardware. In particular we got some insight we didn't have before into how we can do multiprocessors.

We also got some insights into more general programmable pipeline environments. At Carnegie Mellon University there's a large project to build a machine called WARP, a one-dimensional systolic array intended primarily for vision and robotics applications. But from a distance WARP looks like a low-level, horizontally microcoded, programmable pipeline. We said, "Hey, we have this architecture, this idea for the restructuring of scan-line algorithms for a deep pipeline, and we have this programmable pipeline over at CMU." So perhaps the insight can result in something we can actually do.

It's nice to build the hardware but sometimes just thinking about restructuring the algorithms for the hardware gives you
some leverage to use in a different application.

Coit: I'll say something that shows the perverse way in which venture capitalists think. In markets like this perhaps the real opportunities are in components for point accelerators—perhaps compilers, perhaps specialty semiconductors used in do-it-yourself kits for accelerators.

Saxe: I'd like to expand on Rob Smith's earlier statement that a general-purpose machine used for simulation should represent only an incremental cost whereas a point accelerator should be considered as costing its full purchase price. If you go into a big company like Intel and walk around their VAXes and see what they do, you'll find some that run one program day in and day out. In such cases the customer has bought himself a point accelerator that just happens to be a general-purpose machine with someone's application package on it. The way you can get a point accelerator into that market is to walk in with a machine that exactly duplicates the operation of the VAX—at better price/performance.

Smith: That reinforces something I was going to point out. I think another attribute of a commercially successful point accelerator would be that it would accelerate a preexisting compute-intensive tool that already has an established market position. It's a tough sell to introduce a new tool based on a new accelerator.

D&T: One criticism of using specialized hardware for CAD is that it represents a "bigger hammer" approach. Is there a smarter technique that we can use instead of specialized hardware?

Rutenbar: Sometimes a bigger hammer is the right answer. There is the practicality vs. elegance argument. As an academic I'm probably not the right person to comment on the practicality side of it, but I can talk about elegance. I think that elegance is now beginning to be applied to the general approaches people are taking to put things such as special hardware on parallel machines. For example, you can do a very good "point accelerator" by building an instruction set processor with a microprogrammable chip set, making it horizontally microprogrammable, recasting your CAD tool in microcode, adding a large and fast memory, and getting rid of the virtual memory hierarchy. Such a machine may be very viable commercially but it is uninteresting in a fundamental sense. What can I learn from it that will surprise me? What can I get out of it that I can leverage in a radical new direction? Not a lot. So, whether elegance or practicality is desirable depends on your viewpoint.

D&T: Let's change paths now and look at parallel CAD algorithms and algorithm development. Richard Newton at UC Ber-

Smith: "There are three problems [in writing parallel CAD algorithms]—debugging, debugging, and debugging."

keley has said that "the low-hanging fruit has been picked" and that significant advances will now come much more slowly even though a lot of work still needs to be done. Do you agree with this?

Saxe: It takes a lot of engineering to take any application out to the user community and to get it to the point that users will accept it. So there really are no easily picked fruit from that point of view. But I think there are some fruit still relatively easy to reach in the form of parallel routing or parallel placement machines that the industry doesn't have yet.

Sangiovanni-Vincentelli: I think that now that we have so much power in the form of parallel machines we can attack those problems that we could not attack earlier because their computational cost was too high. A problem that comes to mind is design verification—design verification in terms of logic verification. Guaranteeing that two designs are functionally equivalent, instead of using simulation, has been so expensive that you could do only a limited amount. But now that we have the power of parallel processors we can rethink this application.

I think the higher-hanging fruit are the ones people earlier discovered were too expensive to pick, but now we can afford to pick them, especially since they offer such a high payoff.

Rutenbar: Let's go back to the low-hanging fruit—there is a very technical sense in which you can think about it. There are many problems that are still computationally expensive and for which there is an obvious solution—making them parallel, using a follow-your-nose, easy sort of parallelism. Design rule checking and electrical rule checking are made difficult and time-consuming in large part because you have a lot of different masks and a lot of different checks, each of which may take anywhere from several hours to a day. So if you have to do 50 of them, it takes a long time. But if you have 50 processors you can put one on each and immediately you win big. You feel warm and happy. It's easy fruit to pick.

But what do you do when you have 1000 processors or 15,000 processors? What do you do when your processors are one bit wide? Life is suddenly much more interesting. At some point you have to bite the bullet and have a lot of processors somehow cooperating on tasks, decomposing tasks that you liked to believe were nonpartitionable. That's what I think is interesting.

D&T: Some of you are writing parallel CAD algorithms. What problems are you encountering?

Smith: There are three problems—debugging, debugging, and debugging,
Sangiovanni-Vincentelli: For parallel machines, we need not only symbolic debuggers to verify that the code is correct but also ways to detect which processor is doing what at what time and what traffic is in the network. For example, we developed a circuit simulator on a parallel machine. Our experiments demonstrated that on 8, 10, and 16 processors, hardware utilization was excellent. However, on 100 processors, hardware utilization was rather poor. It took us a long time and a lot of pain to figure out the cause of this degradation.

On some existing parallel machines, you may have to stare at the lights under the processor boards to get an idea of what the processors are doing. If the light is red, the processor is computing. If the light is green, the processor has some free time. And if the light is yellow, the processor is stuck. This is what I would call a low-level debugging tool!

Smith: I'd call the lights built-in instrumentation. You can use them for debugging, for performance analysis, and for assertion checking; they give you some automated assistance and show the correctness of an output.

Something that should probably be number four on our list of problems encountered in writing parallel CAD algorithms is nondeterminism. It turns out that for massively parallel computers—meaning ones with more than two or three nodes—the answers computed are often not the same each time a program is run. It is quite challenging to devise a completely deterministic algorithm—one that is deterministic in terms of outcome and in terms of temporal behavior, where temporal behavior means the rate at which it produces its result. I think that we are going to find out a lot more about determinism and nondeterminism as we get into parallel algorithms.

Saxe: That's the problem we've run into in our parallel processors. On the purely practical side, nondeterministic behavior makes it difficult to see if the machines are working when you build them. You run them and you get two different answers—and then what? And usually the two answers are isomorphic. A human can sit down and say that he understands it, that the answers just came out in a different order. But nondeterministic behavior makes automated testing tough.

D&E: People who are trying to deliver parallel CAD tools seem to have special problems, ones such as how to convince a customer that different runs producing different answers is acceptable. Do you dream of supporting parallel CAD tools across a variety of machines?

Saxe: We don't dream of supporting tools across multiple machines. We support tools for the particular machine we've built. We have the one problem with nondeterministic behavior. I don't think the customer will accept different answers from different runs. I think, in fact, that what you need to do is go into his current single-flow algorithm and essentially replace it with your parallel algorithm and get the same answer he got with his old nonparallel algorithm.

Smith: The customer's acceptance is a matter of degree, I think. For some customers the machine's behavior will be close enough to the ideal to be satisfactory and acceptable. I think there is a spectrum of customer acceptance here.

D&E: Anyone have other insights about parallel machines?

Smith: I agree with what Alberto (Sangiovanni-Vincentelli) said about software support. As a matter of fact, I believe that the big breakthroughs in the commercialization of parallel machines are going to be a result of their becoming easier to program. In the late 70's and early 80's, a number of hardware-oriented companies had brilliantly conceived hardware that was very difficult to program. They found out that this hardware had difficulty getting accepted in the marketplace simply because people couldn't program fast enough to field the units in large quantities. This was part of what caused Floating Point Systems' growth to be slow.

Coit: In some way we've again minimized the value of the point accelerator. Somehow a good programming environment goes hand in hand with general-purposesness. I don't think you can take a single point accelerator that can do one algorithm terribly well and give it a rich programming environment. That doesn't make sense.

Rutenbar: An interesting question is how much performance you lose when you abstract the user far away from the lowest levels of hardware. That question brings you down to a very, very nitty-gritty engineering problem and there aren't any good answers yet. There are a lot of good ideas about how you insulate users. There is a consensus about message-passing layers—you put a message-passing view of the world in your software and run it on a message-passing machine.

Sangiovanni-Vincentelli: I think we have to either spend more time parallelizing compilers, making them very, very smart, or spend more time developing an environment to support the writing of parallel programs.

Coit: Despite all the skepticism about both point accelerators and general-purpose parallel machines I've heard expressed here, I still see a lot of evidence that the experimentation market is alive and well. Most major companies still have laboratories and CAD development groups that are
Ritchm instead of SPICE. Thus there is an acceptance problem that has to be worked around.

Smith: Let me comment on that. Every time you issue a new software release for a commercial CAD tool you have the same problem. You have to go through a revalidation, a recertification, a reestablishment of credibility. That too is a characteristic of our market.

D&T: Let's close the discussion with this question: In five years, where in the parallel CAD market will the smart money be?

Saxe: Yes, I was going to say that there are a lot of different customers. There are the "buy one of anything" customers, who can delude you. You think sales are going well until you've sold to all five or so of them and then sales fall flat. There is the "back to the wall" customer. Fault simulation for a big chip is one of those problems he just can't do on a general-purpose computer in his lifetime. Your accelerator comes out and he buys one to see whether it can do the job. There is the "design cycle" customer. He wants to run 500,000 clock cycles of an IC through a standard logic simulator and he can't do it. So he buys your accelerator. And there is the customer who buys solely on the basis of cost. He may have an IBM 3094 totally dedicated to doing simulation and will buy your machine if it provides the same performance a little more cheaply. You'll always have this range of customers in the market. The question is what size your segment of the market will be.

D&T: If you were to advise a prospective buyer of parallel CAD tools—specialized or general-purpose—what would you tell him?

Coit: I'd tell him to make sure to get the highest leverage from the product for dollar spent.

Nothing sells a design system faster than when a buyer's competitor is gaining an advantage because he has a better CAD strategy. That gives the buyer a tremendous incentive. But so far that dynamic has not really helped us. The linkage between losing your shirt in the marketplace and lacking the right CAD tools is not as clear as it could be. When that linkage becomes clear then perhaps the market for the CAD tools we've been discussing will explode.

Saxe: I'll turn the question around and cite something customers have told me about parallel algorithms. They are worried about whether the algorithms are really the same as the ones they are used to. The classic example of this is in circuit simulation: SPICE vs. relaxation. They aren't sure about betting their next-generation memory chip, for example, on the new algorithm instead of SPICE. Thus there is an acceptance problem that has to be worked around.

Smith: Let me comment on that. Every time you issue a new software release for a commercial CAD tool you have the same problem. You have to go through a revalidation, a recertification, a reestablishment of credibility. That too is a characteristic of our market.

D&T: Let's close the discussion with this question: In five years, where in the parallel CAD market will the smart money be?

"How do I use all the technology that is available today, including parallel CAD tools, to good advantage in solving my problem?" and "Is this the right tool to use?" This will be a service business linked to a hardware or software business. It will be a significant opportunity because many customers won't have the ability to make sense of the tools that will exist or make them work for them.

Smith: In general I agree with Steve (Coit). But I also think the smart money will be on smart parallel CAD. I think that in five years CAD will pretty much be interactive graphics and parallel compute-intensive algorithms.

Saxe: From a user's point of view what we have today is inadequate. He can get his chips built, but just barely, and we in the CAD-tool industry are only just keeping up with the growth in ICs. This makes it pretty clear to me that uniprocessors won't grow fast enough to build the next generation of uniprocessors. Only by going parallel can we hope to get some bigger improvements in performance.

Sangiovanni-Vincentelli: In five years the smart money will be on massively parallel computing. Research on massively parallel algorithms and architectures will have to be done. This research will have to examine the trade-off between the computing power of each processor and the number of processors and their interconnection topology.

D&T: We've talked about hardware accelerators as commercial products and as research objectives; we've talked about their being implemented as point accelerators and on general-purpose parallel computers. There seems to be little disagreement about the commercial arena: Only point-simulation accelerators are economically viable now. In the research domain, the advances needed for the next generation of accelerators include more flexible hardware engines, better parallel programming environments, and the development of parallel CAD algorithms.

Blank (D&T): "...the advances needed for the next generation of accelerators include more flexible hardware engines, better parallel programming environments, and the development of parallel CAD algorithms."

Coit: Certainly there is a market for components, and by components I mean the technology nuggets that go into parallel devices. I would include in that compilers, algorithms, and busses.

But I think there may be bigger opportunities in applying the tools—that is, in providing the answers to the questions,