Design Automation Conference Roundtable:

No doubt about it, logic synthesis has arrived, or so concludes our panel in this frank discussion of the strengths and weaknesses of actual systems. The roundtable begins with a brief look at the motivation for logic synthesis, continues with commentary on synthesizing control logic versus data flow logic, and then examines current systems and why they succeed.

Panelists include Robert Brayton and William Joyner of IBM T.J. Watson Research Center in Yorktown Heights, New York; Alberto Sangiovanni-Vincentelli of the University of California, Berkeley; David Gregory of GE Calma in Research Triangle Park, North Carolina; and Osamu Karatsu of NTT in Japan. Aart de Geus, also of GE Calma, organized the panel and represented D&T. D&T Roundtable Editor Charles Radke of IBM in Hopewell Junction, New York, coordinated the roundtable. D&T Editor-in-Chief Vishwani Agrawal of AT&T Bell Laboratories in Murray Hill, New Jersey, also participated.

The Design Automation Conference is co-sponsored by the IEEE Design Automation Technical Committee (DATC) and ACM Special Interest Group on Design Automation (SIGDA). The roundtable was done in cooperation with these two groups.

D&T: Let's start by looking at the motivation for logic synthesis and optimization. How do you see it?

Brayton: The motivation seems to be from people needing to be more productive. Designers are beginning to see that they can be more productive using logic synthesis or, perhaps, will be in the future. Logic synthesis has been around for some time, but it's on the verge of taking off at this point.

Gregory: There's also motivation from necessity. As designs become more complex, tools are needed to design large portions of the circuit automatically. Logic synthesis certainly offers a means for this.

Joyner: I agree, especially about motivation for greater productivity. Correct designs, smaller designs, faster circuits, faster turnaround— all of these are goals really, but the key one is productivity. I think the objective is to design a circuit or a machine in a shorter time with fewer people.

Karatsu: I mainly agree, but also I'd like to point out that productivity balance among several design phases or design steps is very important in the VLSI era. Ten years ago, layout design productivity was very poor and it seemed to be a major bottleneck in the overall design. Many organizations developed automatic placement and routing systems to reduce layout time, and after several advances in this area, layout design productivity greatly improved. Now, people are noticing that logic design consumes a large part of VLSI design, and they have identified it as the next major bottleneck.

Sangiovanni-Vincentelli: Major changes in the electronics industry have spurred great interest in application-specific ICs. This change implies that system designers not familiar with IC design will be exposed to silicon. Logic synthesis will certainly be an important tool in allowing system designers to effectively use silicon. This situation motivates the development of robust logic synthesis tools.

William Joyner: "...without active participation by logic designers... we would not have been successful with synthesis down to the technology-specific levels."

D&T: What is logic synthesis being used for specifically?

Joyner: At IBM Research, we are trying to use a synthesis system to reduce the time it takes from the concept, the beginning, of design—whether it starts from a previous design or from scratch—to the point at which a machine is ready to be released. This comes back to productivity. You can think of productivity narrowly in terms of how long it takes to get a particular design done, or more widely, in terms of whether there are errors that have to be corrected, requiring you to go back through this process. I tend to see the broader definition. If machines can be produced with fewer errors in fewer passes, productivity has been improved.

D&T: Do you mean that logic synthesis should strive to achieve "correctness" in design?

Joyner: Certainly. Correctness is a chief goal of logic synthesis, but checking tools are still used, as they were in the manual process, to determine the basic correctness of the output as related to the input. What synthesis tools do is allow us fewer iterations along the path of getting correct design input. We hope that the first design is correct.

D&T: I guess we all agree that correctness is not a sufficient condition. Anyone care to comment?

Gregory: At GE Calma, we found that indeed a number of other conditions had to be met to make synthesized circuits acceptable. Circuits have to be built out of components that a user has in his technology, and they have to be small and fast.

We have to remember that the people who will use this technology are circuit designers—those who now do this by hand—so synthesis programs have to be able to compete with the designers before the designers are going to accept our technology. Certainly, we have to design correct circuits and be able to guarantee this correctness. But we can do a lot more. We can let designers design from a higher level. This allows them to ignore a lot of the details that they used to have to know to design logic circuits. They no longer need to worry about layout rules, design rules, and the particular gates available in the technology.

Robert Brayton: "...if you see your CAD support people not moving toward a logic synthesis system, get on them and say, 'This is the way of the future. Let's get with it.'"

Cont'd. on page 64

IEEE DESIGN & TEST
Continued from page 6

Brayton: I think those are good points. Also, our results will get better and better as we improve things.

Another goal is to have a system that allows for transparency so that if the user really wants to sit down and design circuits, he can. He should be able to say to the system, don't do anything with this except verify that all the circuits used are in the acceptable range.

One more goal is the control of the objectives. You should be able to control whether you want minimum area or minimum time or a trade-off between the two. Somehow that control should be reflected in the final results.

Karatsu: About the correctness of the design, the higher the description level of design, the easier it is for the designer to understand the behavior of circuits. This makes writing and debugging procedures simple. With a logic synthesis system, the designer can use a higher level description like a register-level-transfer language. So he can get design correctness much more easily.

Sangiovanni-Vincentelli: Shortening design time at the expense of design quality has been opposed by the IC community in the past, in particular, with respect to the quality offered by first-generation silicon compilers. I think that logic synthesis tools can be developed that offer excellent quality outputs. What is your assessment of the capability of logic synthesis tools today?

Joyner: Efficiency is one of the measures of whether automatic synthesis can compete with manual design. We have done some comparisons with manual designs and found that we can get designs from the synthesis system that are comparable in terms of speed, area, and efficient use of primitives.

I think more work needs to be done, especially to get more kinds of test cases and inputs to see how this comparison is holding up. The efficiency of the design is certainly something that cannot be sacrificed for just the speed of the design. When you talk about more complex custom microprocessors and special-purpose small designs, there is even more of a challenge for an automatic tool to come up with something that's done efficiently by a custom designer.

Gregory: I think the clear win on synthesis is to be able to design circuits faster than humans. As far as quality goes, designers have criteria for how small and fast a design must be. If a system can synthesize a circuit that meets these criteria, then it will always make sense to use the system, since it will be faster than a human designer. If a synthesis system can meet the constraint for only 50 percent of designs, it will only be able to speed up the design process for those 50 percent. Of course, the faster a program runs, the more it will improve design times.

Brayton: The synthesis program plays a role in two places. One is early in the design for estimation purposes, as in floor planning. For estimation you don't really have to be that good, but you should turn around a design fairly fast. Later on, though, you want really good efficiency. I think we are already approaching good efficiency. We've demonstrated that logic synthesis can produce designs that are just as good or better than manual ones on many kinds of logic.

Take PLAs. A logic synthesis system can do better than a logic designer, and I think that we'll also see that happening for other methods of implementation later on as better synthesis techniques evolve. Also, you can control the efficiency by the input, just as you do in programming. If you have an algorithm that does good sorting, you input that by actually spelling out the algorithm. You should also be allowed that with a synthesis system. A good synthesis system should not destroy a good design. I think we can control the efficiency perfectly well right now by controlling the input, by having the designer put a little thought into the input if he wants it to be efficient.

Karatsu: Many problems still remain in achieving a quality design efficiently, and much more work should be done to find solutions. When you use a currently implemented logic synthesis system, efficiency is affected by the structure of the logic to be designed or the skill of those preparing the design descriptions. But even for random logic, the logic synthesis program can output better designs than are possible manually—in complex decoders, for example.

At NTT, when we used logic synthesis with actual VLSIs that contained decoder circuits with more than 100 gates, we achieved a 50 percent reduction in gate counts over what was possible manually. This case was special because the development time was essentially infinite. In a real design environment, the development time is limited. But the point is that even manually a design cannot be optimized completely.

Joyner: At IBM Research, we are using specified target sets of building blocks and are trying to meet requirements of size, speed, or other criteria placed on a design. These synthesis approaches have been able to work within that constrained environment and achieve the stated goals. In that sense, the designs are acceptable and therefore by definition as good as manual design because they meet these requirements.

There are going to be additional challenges to the synthesis systems in the future. If we stay with the standard cell or gate array, we will need to influence the kinds of primitives, choosing the ones that are the easiest for synthesis systems to manipulate, rather than just having them handed to us by those designing physical patterns. If we go toward more customized designs, the challenge may be to design gradually less restricted arrangements of transistors and their shapes and sizes to meet design requirements. This is a harder problem, and something that is being looked at.

Sangiovanni-Vincentelli: Most of the current problems for logic synthesis systems originate at the interface with physical design. I am convinced that physical design tools such as module generators should be intimately connected with logic synthesis tools. What is your opinion?

Gregory: The system we've developed at GE Calma is designed to work with components that have already been designed. We had in mind to use gate-array or standard cell components. We take in descriptions of these components so that we can simulate them with technology-specific information. This allows us to look at the circuit and see how big it is and how fast it is, just as a designer would.

After you design a standard cell or gate array you still have to lay it out, and simulation results change according to the way this is done. There are two ways to account for this. You can try to integrate the layout with your synthesis program so that the program can lay out a design and
get simulation information directly. Or, you can build a tool that will look at the circuit and approximate what a layout tool would do with your design. We chose the second alternative because it is a lot faster and a lot less complex.

D&T: Aren't you working on tying layout and synthesis closer together?

Brayton: Yes, but first I'd like to point out that logic synthesis can be divided into PLA and random logic or multilevel logic. Up to now PLA synthesis has been available and quite good. Naturally, then, people think about PLA versus a data-flow-path logic organization. They try to group all the PLA logic together and say, "Here's my PLA," or they divide it up. When random or multilevel logic synthesis becomes available, as it is already in some places, I think we'll see a similar influence on how to describe logic initially and to partition it.

Another influence is that some cells you can build with seem to be more efficient and easier to use in logic synthesis. I'm thinking of the complex cell, complex CMOS gates, or complex domino gates, where you can really get a lot of logic power in a single gate. In the past, these gates have been more difficult to design manually because they are harder to manipulate mentally. A synthesis system really doesn't have much problem with that. Sometimes it's even a bit easier to do it. So we might see the influence of logic synthesis on which gates or cells are used in the future.

D&T: Tying synthesis tools to the layout alleviates the designer from needing to know exactly how to do the layout. Thus automation puts every designer in the position of being a very good designer, even those without expertise and experience. Automatic synthesis gives a lot of power to people who previously would not do logic design or who lacked the time to do a careful job.

Brayton: What's the interaction between logic and floor planning?

Sangiovanni-Vincentelli: Floor planning includes a set of tasks that a designer has to carry out early in the design cycle: power, delay, and area estimation are the most important aspects. The inputs to a floor-planning tool should be an estimate of power, delay, and area for each of the modules or components of the design. Logic synthesis tools could be used to provide an initial estimate of some of these parameters. The outputs of the floor planner are power, delay, and area estimation for the entire chip, and guidelines for the logic synthesis system and for the placement and routing tools.

For example, the floor planner can set constraints on the pin positions and the aspect ratios of the blocks forming the design as well as the timing constraints on the interconnection network. Of course, these guidelines are critical when a macro-cell design style is followed, less critical for standard cells and gate-arrays. Floor planning is an iterative and often interactive process working in close contact with logic synthesis, timing analyzers, and placement and routing tools. It is indispensable in a complete synthesis system. At Berkeley, we are building such a system, in which interaction between floor planning and logic synthesis is of great importance.

D&T: We've heard a lot of requirements, some of which might still be on the "wish list." Where are we today? What is available?

Karatsu: We can generate a logic network from a high-level behavioral description, and we can optimize it by selecting certain functions and by converting the gate primitives to the fabrication technology.

Behavioral descriptions can support asynchronous or synchronous circuits or a mixture; single- or multiple-phase clocking; and a typical set of RTL models. When you use automation, for example, each state is represented by a state flip-flop. For the asynchronous case, latches and primitive gates are used to build a network. The synthesis program then optimizes the network for the least amount of layout area. Typical functions for optimization are the number of gates, nets, and fan-ins. Also the number of gate levels between registers should be optimized. If there is some conflict between desired functions, the designer chooses which is more critical, and the technology is adjusted.

Several kinds of fabrication technology are available: CMOS, NMOS, ECL, TTL, HI, among others. Each requires different gate primitives to implement logic. A technology conversion program can convert the network into the specific technology.

Brayton: From a global point of view, we're really at a point of takeoff. We've demonstrated that logic synthesis does work. In comparison with PLAs, random logic is probably two or three years behind. But we're hearing about initial offerings from some of the CAD companies. Larger corporations are using logic synthesis tools, and they're working very well. We're also beginning to see university offerings in this area, which will reflect some of the research that's going on.

You have to understand that when you compare PLAs and random logic, random logic is a lot harder. It's more flexible; there's less of a scientific base for it. So even though we're only two or three years behind, I think there's a lot of work to be done and we'll have to see how it goes.

Gregory: With PLAs you have a very regular or structured environment for building logic and I think that is what we are ahead of logic synthesis. In logic synthesis we use two-level logic minimization, which was originally developed for PLAs. We also use multilevel logic optimization. There is still work to be done in multilevel logic optimization, but today's algorithms are quite good compared to what people can do by hand with random logic.

With logic synthesis, there is also the problem of designing circuits out of irregular components. Here again, we have been able to demonstrate that our technology is very good. We've been able to design circuits with components that designers use now, and in many cases we've been able to design circuits that are smaller and faster than those designed by hand. So, while there is still a long way to go, we are at a point where we can compete with PLAs.

Sangiovanni-Vincentelli: To the best of my knowledge, IBM's Logic Synthesis System was the first tool of its kind used in industry. How was this tool developed, and how many chips have been designed with its help?

Joyner: The development of the project began mostly from the effort at IBM Research Division in Yorktown Heights back
about six or seven years ago and with the revival of some ideas about applying techniques used in program language compilers to logic synthesis as an alternative to types of Boolean minimization. The real reason, I think, for its success on actual parts is the close cooperation with designers who were able to experiment with such a tool in meeting their design requirements. I am certain that without active participation by logic designers trying to build real parts we would not have been successful with synthesis down to technology-specific levels.

The IBM 3090 was the first successful use of the chips synthesized by this system. In the 3090, one part of the channel control element consists of bipolar gate array chips. Something like 101 of 108 chips were successfully synthesized, and a substantial number of other parts in the processor were also synthesized. The system is now being used experimentally in a variety of other technologies, including CMOS.

D&T: A popular comment is, "Logic synthesis is dandy, but it addresses only control logic and control logic is only a very tiny part of the actual design." Is that comment justified?

Joyner: For us, the number of gates being designed represented quite a large amount of logic. Certainly the numbers were not a small part. When we attempted to use the system for data flow chips, the resulting designs were not as good as manual ones. Later we added some features to the LSS system and were able to compete with manual design, in some cases. Even if we think of random logic as being the glue that holds other parts of the system together, it does consume a lot of manual design. There are more regular and structured tools for laying out data flow.

Sangiovanni-Vincentelli: Don Gajski of the University of Illinois at Urbana asked a similar question during the conference. His opinion is that in microprocessor design, control logic is just a small part of the design because most of the area is occupied by regular arrays implementing the data path.

The agreement among microprocessor designers is that even though regular arrays take 80% of the chip area, they require a short time to design. On the other hand, control logic takes 80% of the design time. Thus, it is crucial to develop tools that can produce control logic in a short time without compromising the quality of the design. In addition, I believe that logic synthesis tools may be effectively used even for components of the design other than control logic. What is your opinion?

Brayton: Of course we can think of data flow as being logic, too, and as such we should be able to synthesize it. Our experience is that we can do reasonable data flow logic if we have some reasonable starting point. I don't mean require the user to tell me how to make the logic, but if you give logic synthesis some sense of the initial structure of the data flow, you can synthesize data flow fairly well.

If you describe the structure and, say, give an adder an ALU in PLA form, then in my experience, it is hard to recover a good ALU design from that block description. That is an interesting problem to work on, but we can still ask the user to give us some starting point and go from there.

When you say logic synthesis, let's not focus just on control logic. We can do data flow, and we can also mix control and data flow as one, for more flexibility in floor planning. If you don't have to put control in one part and data flow in the other, you'll have fewer constraints and possibly a better design in the long run.

I have a question for Bill Joyner. You mentioned that 101 blocks or chips out of 108 in the control channel were synthesized automatically. That means seven were not. How would you characterize those? Are they data flow with lots of structure the designers know about, and is that why they were not done?

Joyner: I'm probably not familiar enough with those exact seven parts to characterize them. I know from discussions with designers of the machine and some of those chips, that some did contain structures that logic designers worked very hard to optimize in terms of area and speed. They wanted to get out of the synthesis system the same thing that they designed manually.

Our synthesis system now would do a better job on those. In fact, in running some of those chips we got acceptable results.

D&T: We are leading naturally into the next question, which concerns design entry through functional descriptions. How do you see present languages like VHDL, ISP, and EDIF satisfying the needs of logic synthesis?

Karatsu: We are currently using an HSL-FX language, which we have developed as a logic-synthesis-oriented RTL. Formerly, we used a simulation-oriented conventional RTL as input to the logic synthesis system. Several experiments showed that inadequate description caused problems in gate-level implementation—for example, redundant facility allocation for similar but not identical circuits and inefficient implementation of bus structure. So we have added more functional and structural information in an RTL context and developed this new language.

A language should not restrict design style to any specific type, whether it is synchronous, asynchronous, hard-wired logic, microprogram control, pipelined control, top-down, or bottom-up.

Gregory: Of the characteristics needed in a language for logic synthesis, the first and most important is to capture the functionality of a design. The second is to be able to capture an algorithm for implementing that function. Synthesis programs can design their own algorithms, but not as well as humans can. If an engineer has an algorithm in mind he should be able to specify it.

A third goal is to capture the structure. In a computer language there is only one structure that a program or algorithm can be mapped on. A circuit synthesis program is more flexible. It has to determine how many components like adders and ALUs to use. A language needs to allow designers to specify this if they want to.

Another desirable characteristic is to allow designers to specify required physical
characteristics of their designs. The designers should be able to say how small or fast they want their circuit to be.

Brayton: We have used two, very different languages to describe logic to a synthesis system. The first started from really wanting to describe something to be synthesized. It was created from scratch with synthesis in mind and simulation as a secondary effect. The second language was an existing register-transfer-level simulation language, from which we extracted logic.

Both languages are adequate. In the first, since we describe only for synthesis, it is easier to extract the logic. But both are doable. As long as a language is adequate—a reasonable modern programming language—the rest is not very important. The translation technique that gets the logic and transforms it is what's important, not so much the description.

Joyner: The language we began with is still used as one of our inputs to the logic synthesis system and was not designed as a synthesis language. It has many deficiencies that can be remedied to make it better suited to logic synthesis. The language was designed for simulation, and the methodology and uses of the language must be kept in mind.

Different tools must work off the same language to maintain an audit trail from what you started with to what you end up with. You want to simulate the language you used as input to the synthesis process to really ensure the final functional correctness. Because of differing requirements, compromises will always be made in what is desired for a synthesis language so that other parts of the methodology will work.

Sangiovanni-Vincentelli: Since logic synthesis is a relatively new tool, there is an important issue to resolve: how to integrate logic synthesis into an existing design system. What is your experience and opinion?

Karatsuba: The layout program is a bridge between the logic structure environment and the mask pattern. The logic synthesis program is a bridge between the behavioral logic environment and the logic structure. So I see logic synthesis as sitting on top of the design system.

Brayton: A logic interchange format, LIF, was developed for this conference that describes logic and allows you to synthesize from that. It effectively puts out results in the format you want, so it gives you a logic synthesis module that can be easily coupled into other systems. To couple this to another system, you translate the initial logic to LIF, synthesize, and then output again in LIF.

However, LIF does not solve the more difficult part of communicating with other parts of the system in a feedback path. After synthesis you may want to derive some information about timing constraints and then feed that back into the input to the synthesis system. You also need to allow sideways paths to input secondary information. A good integrated database system should solve these problems.

Gregory: I also see logic synthesis sitting on top of existing CAD and CAE systems. We allow designers to start from a behavioral description, like Boolean equations, and synthesize circuits at the net-list level. These circuits can then be entered on any CAE or schematic capture systems.

If you're going to build a system around logic synthesis, you can also use feedback from that system. When a layout is determined, information can be fed back on how fast and how large the circuit actually is. The logic synthesis system can take this information into account to come up with an even better design.

Agrawal: We have gone through this discussion without once using the term "silicon compilers." Why are logic synthesis and silicon compilation being treated as totally disjoint terms? Where does logic synthesis actually fit into the design? Silicon compilation is starting from problem specification, down to the layout of a chip. Where does logic synthesis actually fit in?

Joyner: One thing that seems to be lacking from most of what are called silicon compilers is the type of optimization and minimization we are discussing. I think the term compilation as it relates to languages, applies more to this kind of logic synthesis than it does to silicon compilers.

What's needed in commercially available silicon compiler systems are the simplification techniques discussed here, most of which involve tools to manipulate designs and take them down to the logic level.

D&T: So far we haven't talked about test at all, and after all, this is a Design & Test Roundtable. Can logic be synthesized to be testable?

Brayton: When you say testable do you mean the percentage of stuck faults covered? If we synthesize a piece of logic, I see no reason to expect less coverage than we might otherwise have had. In fact, some of the ideas from minimization and some of the work being done now are to discover redundancies and remove them. Synthesis can improve testing, not the other way around.

D&T: Synthesis programs create redundancies; is that true?

Brayton: It shouldn't be. Optimization means getting as few redundancies as possible. If we can optimize, and I think we can, we can expect fewer redundancies.

Agrawal: That is certainly true for PLAs, where you have fewer redundancies, but for random logic, where you are trying to fit into standard cells or standard building blocks, you do have some extra redundancies.

Brayton: From an automatic synthesis system or from manual design?

Agrawal: From automatic synthesis.

Gregory: In limited experiments we have found that every circuit we synthesized was 100% testable. We have also found that we can remove redundancies from hand designs. But, we still need to do more work to be able to guarantee that automatically synthesized circuits are 100% testable.

Joyner: Synthesis systems do need to address testability. Often, there are stringent requirements on the number of faults that need to be detected. There is no getting around the difficulties of removing redundancies—whether by optimization, which by its nature enforces the lack of redundancies, or by working to eliminate redundancies, as we did in our group. We're hoping that after synthesis, regardless of the approach used, we'll have fewer redundancies and more testable logic.

Cont'd on page 71
Continued from page 8

March 1987

VLSI Test Workshop, March 24-25, Atlantic City, N.J. Contact Wesley Radcliff, IBM, E. Fishkill Bldg. 321-SEI, Dept. 277, Hopewell Junction, NY 12533; (914) 894-4346.

Second Int'l Conf. on Robotics and Factories of the Future, March 24-27, Salt Lake City, Utah. Contact R. Radharaman, Dept. of Mechanical and Industr. Engnr., College of Engnr., Univ. of Utah, Salt Lake City, UT 84112; (801) 581-2241.

IEEE Int'l Conf. on Robotics and Automation, March 30-April 2, Raleigh, N.C. Contact Harry Hayman, 738 Whitaker Terr., Silver Spring, MD 20901; (301) 434-1900.

April 1987

Fourth Int'l Workshop on Software Specification and Design, April 3-4, Monterey, Calif. Contact M.T. Harandi, Dept. of CS, Univ. of Ill. at Champaign-Urbana, 1304 W. Springfield, Urbana, IL 61801; (217) 333-4666.

Second Tech. Workshop: New Directions for IC Testing, April 8-10, Winnipeg, Canada. Contact D.M. Miller, Dept. of CS, Univ. of Manitoba, Winnipeg, MB R3T 2N2; (204) 261-4528.

IEEE Microelectronics Conf. and Exhibition (Miconex '87), April 21-23, Winnipeg, Canada. Contact E. Gonzalez, Infotech Manitoba, 1970 Ness Ave., Winnipeg, Canada R3J 0Y9; (204) 896-2222.

CALL FOR PAPERS

Second Int'l Conf. on Supercomputing, May 4-7, 1987, San Francisco. Submit four copies of manuscripts for receipt by November 1 to Lana P. Kartashve and Steven I. Kartashve, 3000 34th St. S., Suite B-309, St. Petersburg, FL 33711.

Design Automation Workshop, January 21-23, 1987, Gold Canyon Ranch, Apache Junction, Ariz. Send short summaries of your interests and activities as they pertain to the role of artificial intelligence in design automation. Submit information for receipt by November 1. If you have suggestions for session themes or would like to organize a session, contact Gary Leive, GE Calma, PO Box 13049, Research Triangle Park, NC 27709; (919) 549-3613.

Design Automation Conf., June 28-July 1, 1987, Miami Beach, Fla. Submit papers for receipt by November 14 to Donald Thomas, MP Assoc., 7366 Old Mill Tr., Suite 101, Boulder CO 80301; (303) 530-4333. Topic proposals for panels and tutorials are due December 12.

IEEE Transactions on Software Engineering. Papers are solicited for a special issue with a focus on software tools and techniques relating to the development, analysis, maintenance, implementation, and testing of computer communication systems. Submit six copies by November 15, to either Sudhir Aggarwal, Room 2C-180, AT&T Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974; (201) 582-6095, or B. Gopinath, MRE 2K-306, Bell Comm. Research, 435 South St., Morristown, NJ 07960; (201) 829-4252.

Custom Integrated Circuits Conf., May 4-7, 1987, Portland, Oreg. Submit 60 copies of a 250-word summary, including a 35-word abstract that describes the original, unpublished work and the author's name, affiliation, complete address, and telephone number for receipt by November 15 to Roberta Kaspar, CICC '87, 20 Ledgewood Dr., Rochester, NY 14615; (716) 865-7164. Topics of interest include gate arrays, standard cell, and full custom ICs; physical design techniques; circuit design and simulation; DSP applications; analog circuit techniques; fabrication technologies; custom interfaces and packaging; and testing and reliability.

IEEE Microelectronics Conference and Exhibition (Miconex '87), April 21-23, Winnipeg, Canada. Contact E. Gonzalez, Infotech Manitoba, 1970 Ness Ave., Winnipeg, Canada R3J 0Y9; (204) 896-2222.

D&T ROUNDTABLE

Continued from page 67

Agrawal: I hope that as you are trying to make correct circuits, you will also make testable ones, which means you will include design for testability in your synthesis.

Karatse: I think redundancy in a combinatorial circuit depends mainly on the variety of behavioral level input and the logical primitives that you can use in the synthesis system. We can eliminate redundancies through optimization. We can add scan-in, scan-out features in each register to make it testable.

D&T: Imagine for a moment that you are the Ann Landers of logic design. What advice would you give designers? What should they watch out for?

Karatse: Over 20 years ago, the automatic camera with an automatic exposing system was introduced. At the time many professionals said it would be of no use because photography required experience that cannot be built into a camera. Nowadays anyone can use a camera with automated features because we learned how to use the technology. I think logic synthesis is similar. Today's designers have no experience with logic synthesis features. When they start to use the synthesis programs, they will develop their own techniques. After all, they are the ones who will be doing the designs. They will learn how to make the best use of logic synthesis.

Brayton: My advice is that the techniques are here, they work, and if you see your CAD support people not moving toward a logic synthesis system, get on them and say, "This is the way of the future. Let's get with it."

Gregory: My advice is along the same lines. The Design Automation Conference clearly shows that logic synthesis is here. We can synthesize circuits from Boolean equations that are as small and as fast as those optimized by hand.

Joynor: I think designers have to begin using a synthesis tool. They'll see its benefits, that it allows them to concentrate on doing the initial design and supplies them with other needed data, such as timing information. But they have first to use the tool.

Pursuing the analogy of the automatic camera, the camera's evolving features have made it appropriate to a wider range of applications. This will also happen with logic synthesis, although manual design will continue to play a role. Logic design will not meet 100% of all design needs, but it will address an increasing proportion of them.

D&T: I thank all of you for your participation. I certainly learned from this discussion, and I think our readers will also.