

DESPITE GOOD TOOLS, SUCCESSFUL TIMING CLOSURE IS STILL A MATTER OF METHODOLOGY AND ATTENTION.

sage—that, with best practices, continuous attention, and a focus on timing that begins at project inception, closure can become an almost-routine, predictable part of the design schedule.

THE NEW ISSUES

It's not that closure is getting any easier. "As we have moved into 65 nm, we have been having more challenges with timing closure," says Behrooz Abdi, senior vice president and general manager at Qualcomm. "I think this was probably simply due to the increased complexity of our designs at 65 nm, but we haven't gone back to separate out individual causes."

Even without the growth in complexity, plenty of new issues have added to the problem. An obvious one is higher speeds. But a less obvious facet of that issue is that the high speeds don't always occur in the core of the chip. "One of the most difficult problems we encounter is timing closure on high-speed I/O pins," relates Hao Nham, vice president and general manager of design services at eSilicon. "Today's place-and-route tools just won't do a good job on these high-speed signals without very careful scripting."

Another issue that most design teams must contend with today is the use of third-party IP. Setting aside the whole issue of whether third-party IP arrives in working condition, and it usually doesn't, achieving timing closure within IP blocks is a serious challenge. Documentation may be insufficient for you to even understand the structure of the block without reverse-engineering it. "Often, you have to negotiate hard with the IP vendor just to get enough information to use the block successfully," states Richard Tobias, who at the time of the interview was vice president of engineering at Pixelworks. "You do code reviews and as much due diligence as you can. Then, you go in and fix the IP." Even if repair isn't an issue, the vendor may have based the timing information it provides on switch settings or assumptions about the application entirely different from those they used in this design.

Other kinds of IP have a way of intruding into the timing process, as well.

Timing is everything in SOC design

BY RON WILSON • EXECUTIVE EDITOR

It is a recurring nightmare. The SOC (system-on-chip) design has gone smoothly from day one. RTL (register-transfer level) for each block came in on schedule. Synthesis proceeded with hardly a glitch. Functional verification even ran smoothly. Placement and routing went without incident. And now, the postextraction-timing report is back—with 75,000 failed nets. You sit up in bed, cold sweat on your brow.

For too many designers, this scenario isn't a dream; it is reliving yesterday afternoon. Despite a proliferation of timing-analysis tools, despite the profound assurances of all the IP (intellectual-property) vendors, and despite margins so large they threatened the target operating frequency, what should have been the end of a design project breaks into a frenzy of iterations, Band-Aids, and compromises to resolve massive numbers of last-minute timing faults.

Are the tools to blame? Are the timing models wrong? Why, in an industry that is taking on far more complex challenges—such as optical-proximity correction, design for manufacturing, and process-variation compensation—are we still having trouble just achieving timing closure? And why does achiev-

ing it so often cost us too much in area, power, and schedule delay?

EDN sought answers from a number of high-volume ASIC-design shops and some specialized design teams. We heard a number of reasons that timing closure is still hard—and getting harder. But we also heard a reassuring mes-

The most common of these is scan, or its relative on steroids, BIST (built-in self-test). Tools that automatically insert scan chains are generally respectful of timing requirements, but BIST blocks are much larger and more complex, and, necessarily, they have the potential to more greatly disrupt timing. They not only insert devices into signal paths, but also require their own real estate, altering the floorplan and the routing distances between blocks.

“It’s not unusual now for an SOC to have 2000 or 3000 instances of RAMs,” observes R Chandramouli, product-marketing manager for physical IP at ARM’s Artisan Components group. “With their BIST circuitry, self-repair circuitry, and controllers, all those RAMs can cause big problems with circuit complexity and routing congestion. Timing closure becomes a big issue.”

But one of the biggest headaches for timing closure may be just emerging as a design trend: the aggressive pursuit of power management. By employing independent voltage islands, dynamically variable supply voltages, dynamic threshold control, and other such techniques, power-management engineers are multiplying—sometimes exponentially—the number of corners that require inspection in the timing analysis. Just identifying the real corner cases in an SOC that has a dozen voltage islands, each of which can operate at any of four voltages, is the stuff of chronic insomnia. Running all the cases is a job for a Google-sized server ranch.

BEST PRACTICES

Yes, timing closure is a hard job. And yes, new design concepts are making it harder. But experienced design managers say that you can tame timing closure by applying best practices in a continuously iterative process that begins at the project inception and continues until sign-off. Far from being a discrete step in the design flow or, worse yet, an afterthought, timing closure has become the heartbeat of a design cycle (**Figure 1**).

Managers describe a simple iteration: Reduce the design to a lower level of abstraction, estimate timing as precisely as possible based on that level of abstraction, set margins to minimize failing nets, fix the outlying nets, and repeat.

“The object is to set your methodology to minimize the number of iterations and

AT A GLANCE

- Despite tools and years of practice, timing closure continues to be a major issue in chip design.
- Only a methodology that continuously refines timing estimates and deals with failing paths throughout the design flow can cope with the problem.
- The strategy is to make iterations as early as possible and to leverage the skill of experienced physical-design experts.
- New design techniques to enhance speed or reduce power also complicate timing closure.

to iterate as early in the design cycle as possible,” says eSilicon’s Nham. If you just set the margins wide enough, it would be possible to ensure at the netlist level that there would be no nets with negative slack in the postextraction analysis. That would save a lot of time. But it would also have dire implications for the overall design: You would be leaving a lot of either performance or energy efficiency on the cutting-room floor. So the iterative process begins even before architectural design, with predesign planning.

“We use a pyramid to describe our timing-closure strategy,” explains Jay Jayaprakash, ASIC-design manager at Open-Silicon. “Say we try to get closure on 90% of the paths by pessimism, 9% through analysis, and 1% through actively fixing paths. But if the circuit is very complex, 1% could be 10,000 paths. So, we look at increasing our level of pessimism and the energy we put into analysis until maybe we are fixing only 0.1% of paths. But that may cost too much power. Even in the best methodology, you have to learn to be efficient in dealing with that last fraction of the paths. You have to agree on a strategy with your customer.”

THE FIRST STEPS

“First, understand the design requirements,” Nham advises. In other words, have a clear agreement with the customer on the acceptable windows for die area, performance, active and standby power, choice of process technology, and design schedule. All of these things can trade off against each other. For instance, a design team can meet an impossible-looking

schedule if you relax all the other constraints, allowing virtually a push-the-buttons design flow. Or, the team can hit a very challenging speed target if it has unlimited area and power for dealing with slow paths and extra schedule for combing through them. The critical point is not to get locked into a set of requirements that will force tight margining on both timing and power and leave no resources to deal with the plethora of timing violations these choices will produce.

Once you’ve settled the design goals—always subject to later renegotiation, unfortunately—the focus of timing closure moves to architectural design. Choices that the chip architect makes can significantly influence the ease of achieving closure by determining the necessary operating frequency and hence the timing budget between stages.

Decisions such as whether to use a single fast CPU core or multiple slower cores, for example, can have a significant impact on operating frequency. In a recent mobile media-processing SOC that 3Dlabs designed, the choice of a pair of ARM926EJ cores over a single ARM11 meant that the cores could operate at a maximum of 200 MHz and meet all task deadlines. The fact that numerically intensive processing is done on a 24-element processing array meant that the array could run at no more than 100 MHz—a small fraction of the clock frequency that would have been necessary for a typical single-instruction, multiple-data coprocessor.

More subtle architectural decisions are important, as well. How deeply can the design pipeline processing units? Can you organize state machines so that the fastest state transitions are highly localized? Can you organize memory blocks to limit the physical distance between a client and the memory it must access? All of these decisions can remove—or generate—critical timing paths simply by changing the delay budget available for them.

Floorplanning also plays a vital role here. There are obvious issues, such as putting SERDES (serializer/deserializer) blocks close to high-speed I/O pins. And there are less obvious and more time-consuming steps, such as hand-placing smaller memory instances.

“It’s not uncommon to have a few hundred memory instances in a design,” eSilicon’s Nham says. “It’s best if skilled

designers place these instances in the floor map. That way, you can get pretty accurate estimates of the timing through the memories very early on.” These steps take place before RTL code even exists for much of the design, yet they can significantly impact closure later on.

BEGINNING WITH RTL

Detailed RTL coding of individual blocks is perhaps the place at which

designers begin to think seriously about timing issues. As we have seen, this point is much too late to start. But contributions to the timing-closure process get more frequent and more obvious from here on out.

The literature on RTL styles, best practices, and tricks to influence timing closure is large and too detailed to review here. Physical-synthesis tools have subsumed many of these details, so that

the designer’s control over the RTL optimization for timing is almost entirely through constraint files and synthesis directives. Interestingly, this approach puts a premium not on logic-design skill but on understanding the workings of the synthesis tool.

Some experienced designers recommend keeping constraints as simple and slack as possible. Doing so reduces the runtime for physical synthesis, of course.

STATISTICAL ANALYSIS TAKES ON TIMING VARIATIONS

Identifying corners for timing analysis has always taken process variability into account. But at 90 nm and below, process variation doesn’t contribute one variable to the timing problem; it contributes many. Variations may occur not only in the effective channel length, but also in channel mobility, threshold voltage, contact and via resistance, metal dimensions, and other areas. Including each of these important sources of variation in the selection of process corners for timing analysis would produce dozens or hundreds of corners.

Equally problematic, such analyses would be too conservative. Many of the variations are relatively uniform throughout a single die or a wafer. (These variations are die-to-die.) Others are random in nature and tend to average out, rather than accumulate, along a path. So, corner-based analysis may establish margins that are far larger than they need to be, because corner-based analysis assumes that all parameters are simultaneously at their worst-case corners—an extremely unlikely event in reality.

The literature has identified statistical-timing analysis as a way to attack this problem. But moving from identification to a production tool is another matter entirely. One vendor that has achieved this goal—albeit only with internal designs and a few technically very strong customers, is IBM (Figure A).

“The basic principle is to do the analysis from just one corner,” explains Leon Stok, director of EDA in the Systems and Technology Group at IBM. “We analyze variations from this one corner to see whether timing margins improve or get worse in each process-parameter dimension, such as voltage and metal tracking.”

“This statistical tool is applied primarily in sign-off,” Stok continues. “It is embedded in [IBM’s] EinsTimer [statistical-timing-analysis tool], so it becomes just another part of the sign-off process. In practice, the designers get a timing-slack report just as they ordinarily would. It shows the paths with negative slack, and also paths with positive slack that the tool still considered to be critical. The difference from conventional static-timing tools is that the designer also gets a statistical report on the sensitivity that each of these paths has to the specific factors we are tracking.

“Internally, we set margins so that the user doesn’t

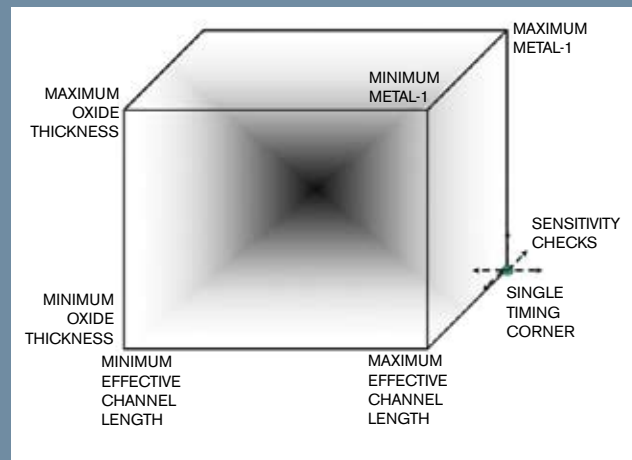


Figure A IBM’s statistical timing approach starts from one process point and computes sensitivities, rather than performing analysis at all possible corners.

have to deal with a ridiculous number of paths in the report. The margins, like the sensitivity analysis, are based on the statistical data we have collected from test wafers and from test patterns on production wafers. In addition to eliminating paths that are going to have positive slack, we provide a fix-up tool—an incremental routing, buffering, and sizing tool—that works with the statistical timer. That [tool] should reduce the typical report from maybe 10,000 initial paths to a few hundred that the designers will actually have to examine. And we provide training so that the designers will understand how to interpret the sensitivity data they are getting.

“Overall, the statistical approach removes a certain amount of unrealistic pessimism from the timing analysis. That means having to look at fewer nets and wasting less time, power, and space on overdesign to meet unnecessarily wide margins.”

Stok says that IBM’s flow provides timing information at each level of abstraction in the design flow. But the statistical tool comes in at the end, when there is enough physical-design data for it to discriminate between necessary guardbands and unnecessary pessimism.

But it also avoids complex iterations of the synthesis step, with modifications to the constraints on each iteration. The problems are not only the time consumed, but also the difficulty in capturing and tracking the changes in the constraints so that you can reproduce the design if it needs RTL changes later in the cycle. The synthesis tool will do whatever it can think of to meet timing constraints, often at a direct cost in area and power. Later changes in logic design, without closely examining the constraints, can produce interesting results—in the negative sense of the word.

Needless to say, a thorough look at the netlist-timing analysis is vital, even though the estimates are still not based on physical data. It is at this point at which Jayaprakash's pyramid sets its base. The pessimism that Open-Silicon counts on to remove 90% of the timing failures begins with the largest possible margins on netlist-timing estimates. "We ask our customers to give us a substantial timing margin at this point," Jayaprakash says. "The industry average is about 15 to 25% using zero-wire-load models." As always, there is a trade-off that you must base on the customer's design requirements. If the chip has to be near the physical-speed capabilities of the chosen process, a large margin at this point will simply cause lots of synthesis iterations, as the tool tries everything before giving up. But too small a margin may substantially increase the number of postplacement failed paths, as the discrepancies between the physical-synthesis tool's timing guesses and the placement-based timing guesses exceed the margins.

Designers will also be inserting test structures into the logic now—BIST structures during RTL development or after first synthesis, and scan structures after logical synthesis. It is important to remember that none of these structures are transparent to signal timing.

But there is another issue with BIST structures, as well. BIST will introduce into the design new operating modes that must have their own timing analysis. Often, they will involve both paths that are not used during normal operation and different frequencies. (This consideration will resurface with a vengeance after physical design.)

It's at this point, after synthesis but before detailed placement and routing,

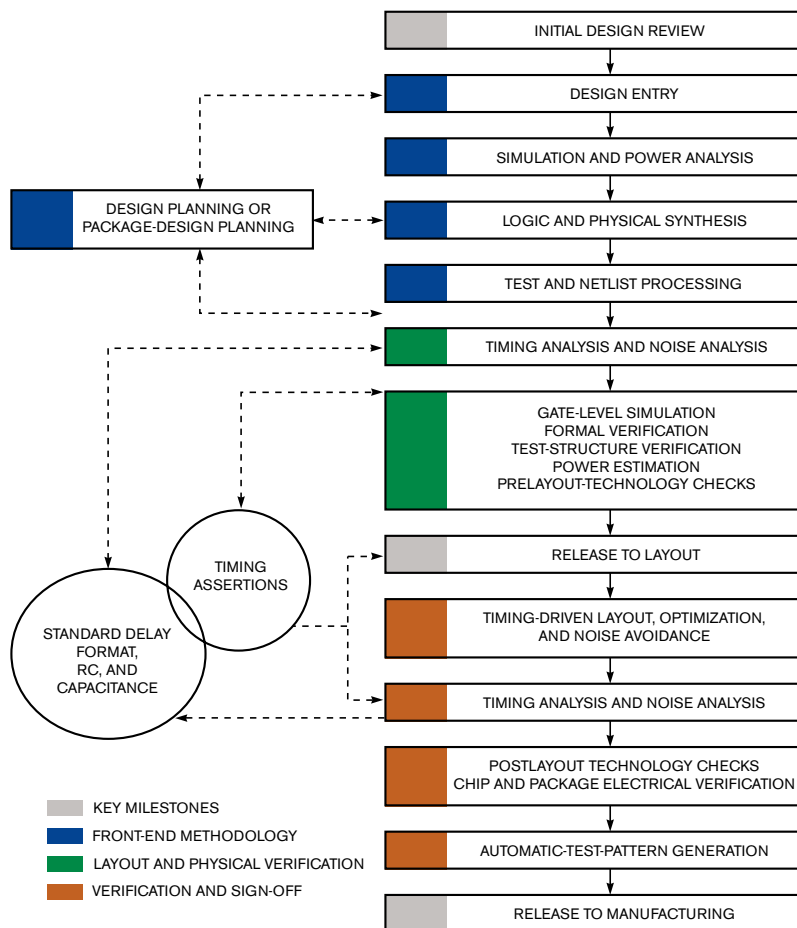


Figure 1 This example of an IBM ASIC flow shows multiple points for timing checks in both the front end and the sign-off process.

that experience may have the greatest leverage on successful timing closure. Both Jayaprakash and Nham emphasize the importance of having previously taken similar blocks through the full design cycle.

"You can't always even identify the critical paths at the netlist level," Nham says. "But you can intuitively spot problem design areas, see where constraints are missing, and do other reality checks on the netlist timing. It's a matter of interpreting the tool output based on your experience with previous similar blocks—categorizing the block design and setting synthesis parameters based on what has worked with similar designs in the past. If you get the right tool parameters, you'll get timing closure."

Jayaprakash agrees: "We match the style of a particular block against the other blocks we've done in our 50-some previous designs and adjust the timing

margins based on how much margin we have needed in the past on similar blocks." This approach, coupled with the ability of tools to give feedback on timing in a matter of hours for a whole region of the chip, makes iterations at the netlist level to cleanse timing paths a powerful method.

PLACEMENT AND ROUTING

During the placement-and-routing process, more detailed models built on increasing amounts of real physical data replace the rough fan-out-and-pattern-based models of the synthesis phase. As in previous steps, there is always the trade-off between accepting more failed paths and asking the tool to try harder. But, at this point, manual intervention becomes an important factor, as well.

In the most complex designs, says Open-Silicon's Jayaprakash, even if you have succeeded in getting 99.9% of the

nets to pass, you still have thousands of failed paths. The first response to this report might be to crank down on the constraints and run the tools again.

“Don’t just keep pushing the button,” Jayaprakash urges. “The first step should be to ask why the tools didn’t fix the timing violation in the first place. If you could look at each failed path, you might see obvious things. A flip-flop has violations on both input and output, so the tool fixed only one. Maybe all the paths into a memory are failing because the memory is just in the wrong place and you didn’t allow the tool to move it. Maybe there’s a problem with one widely used clock.”

Nham observes that hold-time violations are particular problems. “The tools don’t deal well with them today,” he says. “They tend to fix the setup violations and not the hold violations.”

Open-Silicon uses scripts that categorize the failed paths according to starting and ending points, clock sources, physical region, and a number of other parameters. The resulting spreadsheet allows designers to sort failures into maybe 15 buckets, for which all the paths in a bucket are likely to have similar causes of failure. “Now, 10,000 violations become a manageable 15

buckets,” Jayaprakash says. Experienced designers can then go through the buckets and identify the large-scale issues.

After you have completed this task, there will still be some number of paths, probably in the low hundreds, that are one-off issues. But this problem is manageable.

With simpler chips, the process is similar, but the scale is smaller. “If you have been eliminating violations from the netlist level on, you should be left with dozens, not hundreds, by physical-design time,” Nham says. “This is important, because, if you are trying to fix hundreds of violations all at once, the odds are that you will inject new violations as you fix the old ones. Even with a small number of violations, though, there is no tool that can substitute for an experienced layout engineer at this point.”

Two more analyses intrude here to complicate the problem: signal integrity and IR drop. After all the unpleasant timing surprises that the routing and extraction tools reveal, these two additional tools have their own bad news to bring. Signal-integrity tools may demand additional delay on victim nets to allow coupled transients to settle. And IR-drop analysis may tell you that your drive

strength is not what you thought it was.

“Once you get up to a few million gates, you have to do IR-drop analysis,” opines Jayaprakash. “It serves a number of purposes. It is necessary when design constraints are tight. It can be used in the final analysis pass to prove the existence of critical vias—which layout-versus-schematic analysis will not do, by the way.”

IR-drop analysis estimates the actual load at each node in the supply network and then estimates the resistive loss through the supply network and, hence, the actual supply voltage on the device driving each signal node. It requires an accurate model of the supply-trace resistances and—more critically—an accurate estimate of toggle rates in all the operating and test modes. Often, the design team will have to estimate the toggle rates and then feed the information back to architects for validation. The tool will report a problem as an additional delay on a path due to the lower drive strength.

But it gets worse. “At 90 nm, you need to do both static- and dynamic-IR analysis,” Jayaprakash warns. “The static analysis may look fine, but you can have V_{DD} drop by a factor of two for a tenth of a cycle. That’s a killer.” The

FOR DRAM, TIMING CLOSURE IS A DIFFERENT GAME

According to Qimonda principal engineer for DRAM-product development Thomas Vogelsang, timing closure for a new DRAM design is every bit the challenge that it would be for a new SOC (system on chip). But the process is profoundly different.

Three factors make the difference, Vogelsang suggests. First, key parts of a DRAM are asynchronous, and some of them are purely analog rather than digital. This situation brings a whole new set of tools into play. Second, DRAM development is an evolutionary process, so that each new design builds upon a long history, giving designers a lot of experience upon which to rely when making timing estimates early in the design cycle. Finally, timing closure in DRAM occurs from the bottom up, rather than the top down.

“The most critical part of timing analysis for DRAMs is parasitic estimation,” explains Qimonda director of DRAM-product development Michael Kleiner. Vogelsang adds that parasitics account for a large part of the total timing budget for a cycle. This situation means that final analysis starts at the most physical level, with patterns of polygons that occur in the memory array. Field solvers create parasitic models for these polygon patterns, which are then put into a table.

Then, a pattern-matching tool scans the physical design of the array, matching patterns it encounters with patterns in the table and extracting the appropriate parasitic values. This approach leads to an overall Spice model of drive transistors and parasitics for the array. The Spice simulation has to cover known process corners, as well as IR drop, and follows signals into the array, through the memory cells, and out through the sense amplifiers. This method leads to timing values for the memory array itself, derived not from estimated delay figures but from simulation waveforms.

Designers then combine these values with traditional static-timing analysis and FastMOS simulation runs on the synchronous-digital-control and interface logic to produce an analysis of the overall chip. Finally, designers employ Monte Carlo analysis on the I/O pins to derive a full picture of the pin-to-pin timing of the device.

As DRAMs become more complex, the balance is gradually shifting toward static analysis, Vogelsang says. “DDR-3, for example, is a much more complex design, with many more clock cycles per operation. So, static-timing analysis becomes more important.” But nothing changes the fact that the heart of the chip, and the source of most of its delay, is an array of very analog circuits.

solution is usually using any available space in the interconnect stack to hide decoupling capacitors.

At this point in the design flow, careful analysis of test modes—especially for BIST structures—is vital. “The scan mode may use a much lower frequency, so at first glance it works fine,” Jayaprakash observes. “But it may have a 50% toggle ratio—far above anything that is reasonable in normal operation. You have to analyze it separately, with its own toggle rates.”

As architects and designers struggle with the challenges of extreme processes and power constraints, they are unintentionally compounding the problems of timing closure. One example is the growing issue of process variations. The less reliable early predictions of timing are, the wider the margins have to be. Add in process variations that can change interconnect-line widths or transistor-drive currents by a factor of two, and the margins have to be so large that many designs become impossible on paper. Statistical static-timing analysis is one way to attack this problem (see **sidebars** “Statistical analysis takes on timing variations” and “For DRAM, timing closure is a different game”).

But a seemingly more innocuous design trend may turn out to be a bigger issue. The use of independent voltage islands and dynamically adjustable supply voltages—along with the use of dynamic threshold voltages—threatens to become a huge issue.

Just defining the islands and correctly inserting the level shifters can be the first problem. “The tools still need improvement in dealing with voltage islands,” eSilicon’s Nham warns. “Insertion of the interface cells isn’t as fully automatic as it’s supposed to be. And errors in inserting them lead to soft errors in the chip.”

A more intractable problem is the sheer number of corners created by a design in which each of a half-dozen voltage islands may, at any given time, be operating at any of five voltages. “We employed a number of voltage islands in our portable media-processing architecture,” reports Nick Murphy, vice president of architecture at 3Dlabs. “Because of that [step], the number of timing corners grew enormously. The big problem turned out to be unexpected hold violations. It turns out that path delays vary unevenly with chang-

es in supply voltage, so it is not safe to make simplifying assumptions about how the timing relationships change when you turn the voltage down. We ended up developing our own sign-off procedure for the chip.”

Open-Silicon’s Jayaprakash recommends using the following strategy to try to beat this problem: “Running corners at 11 voltage modes is infeasible. So, we try to identify worst-case modes, high speed and low speed, and just use those corners.”

But, if the future brings new problems, it is also bringing new solutions. Several IP companies now offer versions of GALS (globally asynchronous, locally synchronous) interconnect technology. By making all of the paths between blocks self-timed, or, in some cases, timing them to a much slower master clock, the hope is to eliminate timing closure for the long wires that are the chronic sources of trouble in final timing closure. This step, in effect, pushes timing closure down in the hierarchy until it is necessary at only the block level.

Carrying this concept to its extreme, you can make critical paths within blocks self-timed, as well. In fact, this practice is already common in some companies that have the resources for custom-design work. And wholly self-timed chips (out to the external interfaces)—notably an ARM processor and a family of interconnect switches—are on the market today. This approach may, as process issues continue to grow, be the future: a time when timing closure will be a specialized method applying only to certain legacy-circuit designs. In some ways, it would be a relief. **EDN**

FOR MORE INFORMATION

Artisan Components Inc
www.artisan.com

eSilicon
www.esilicon.com

IBM
www.ibm.com

Open-Silicon
www.open-silicon.com

Pixelworks
www.pixelworks.com

Qimonda
www.qimonda.com

Qualcomm
www.qualcomm.com

3Dlabs
www.3dlabs.com

You can reach
Executive Editor
Ron Wilson at
1-408-345-4427 and
ronald.wilson@reedbusiness.com.

