



One of the responsibilities of board-level designers is to ensure that verification and failure-analysis engineers have adequate access to signals without resorting to drills, bed-of-nails testers, or focused ion beams. This requirement used to be simple when pads were farther apart, chips were less complex, and signals were more robust. Today, however, design for debugging is a major concern for creators of board-level products. Some signals—especially in high-speed serial interfaces—are virtually unprobable. Some are inaccessible in the fine-pitch tangle beneath high-pin-count chips. SOCs (systems on chips) lock within themselves some vital information. And some nets that used to require only occasional glances—power and clock networks, for example—have transformed under the pressure of advanced power-management techniques into signal nets that require functional verification. Today, design for debugging requires a plan and a systematic design approach from the beginning of the project.

# DESIGNING AN **ACCESSIBLE BOARD**

DESIGN IN ACCESS TO VERIFICATION AND DEBUGGING DURING—NOT AFTER—THE DEVELOPMENT OF A BOARD-LEVEL PRODUCT.

BY RON WILSON  
EXECUTIVE  
EDITOR



This undertaking is not just another version of design for test. Test engineers want the quickest possible answer to one question: Can I ship this board? Verification and failure-analysis engineers have a different problem. They must be able to put the board through a sequence of states while observing its behavior, and, if the behavior is wrong, they must trace the problem back to its source. This requirement is far more demanding.

Reference designs, because they must be both accessible to the customer and nearly production-ready, present excellent studies for understanding how designers approach this problem. Video-processing vendor Stretch Inc, for example, is acutely aware of the challenge—and opportunity—of reference-design boards. “The reference design is more like an end product,” says Ashish Thanawala, director of systems engineering at the company. “We share it with our customers, who want to take it to market as quickly as possible, and so they care about the size of the board. For that reason, we don’t put in test points.”

On the other hand, access to the refer-

### AT A GLANCE

- Modern boards sometimes don't give verification and debugging engineers the access they need.
- Designers must early and systematically address the problem.
- Different kinds of circuits require different access approaches.
- Software is the key weapon in the verification and debugging battle.

ence design can be so well-planned that, even in its streamlined, near-production-ready form, it can be an adequate vehicle even for silicon debugging. “In this generation, we pretty much got away with using the reference design as a bring-up board,” says Stretch’s chief executive officer, Craig Lytle. “We are talking about whether to attempt that [approach] again with our next-generation design.”



### A SYSTEMATIC APPROACH

A successful compromise between manufacturing-ready compactness and

high accessibility requires a lot of work, most of it in the early stages of the design. “This level of design requires an understanding of what the end customer needs the board to do,” says Ken Havens, tool-marketing manager for ARM North America. “Often, you start out by exploring the features of the system SOC and understanding how the final system will use those features.”

As is often the case, the first step in a complicated problem may be to partition it into many smaller complicated problems. One way to do this partitioning is to divide the board into regions of control and visibility. Categorize each group of nets on the board into one of the following categories: power grids, clock trees, analog signal or control nets, digital nets that you can observe and control through a processor, digital nets accessible to an FPGA, and digital nets not in either of these last two categories. You can now work out an access strategy for each group of nets based on which category it is in.

Debugging engineers’ questions about

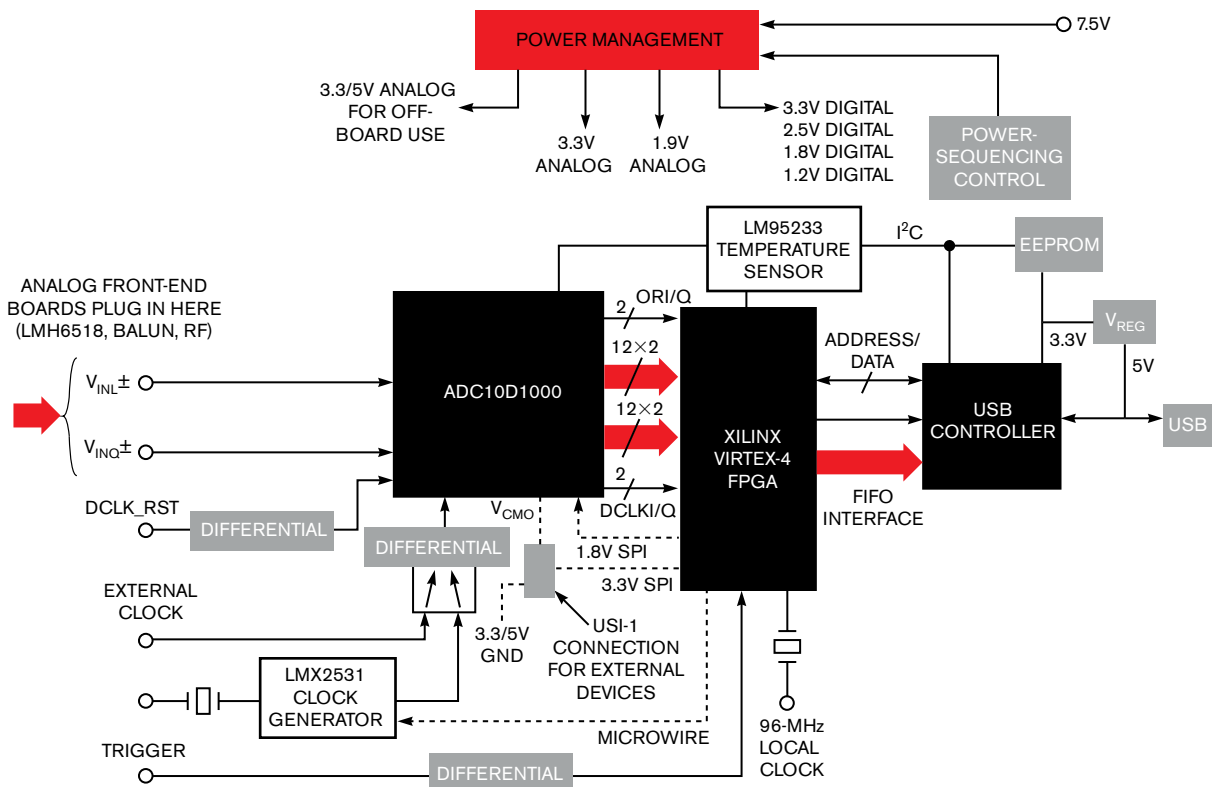
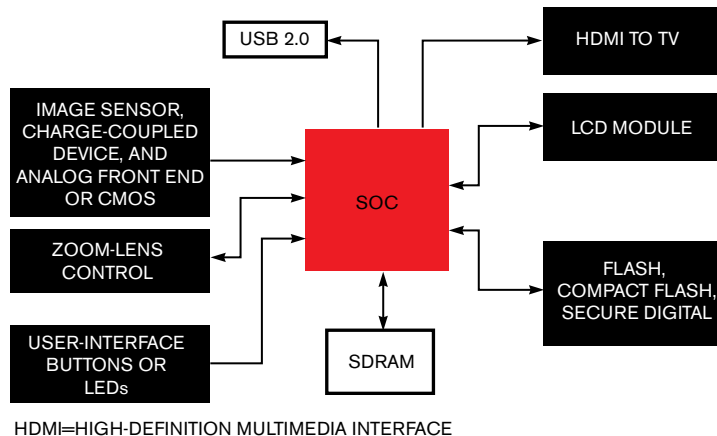


Figure 1 A reference-design board may have multiple supply voltages that must sequence properly on start-up.

power and clock signals used to be simple: Were they there, and were they within spec? The engineers needed an exposed trace, preferably near the largest load, for each supply voltage; a series resistor for current measurements; and an exposed trace on each clock. Nobody distributed high-speed clocks on a board, so they didn't have to worry about skew. Still, even on simple boards, engineers couldn't just probe around mindlessly on a power grid. "On any board, ground is a low-frequency noise source," warns Richard Fellner, a test engineer at IDT (Integrated Device Technology). Before you attach a probe and its ground somewhere, you need to think through what you might be coupling into the ground plane.

And you cannot just casually probe any old clock signal. "There are some clocks you pretty obviously can't mess with," observes Robbie Shergill, director of applications at National Semiconductor. "For instance, you can't probe clocks going into high-speed data converters. If you are going to need to see that clock, you will have to provide a replica."

With the advent of clock gating, dynamic loads, and power gating—all tools of power management—everything has become more complex. In effect, ag-



**Figure 2** An SOC at the heart of a board may also be the best place to observe and control the outlying digital blocks.

provide enough access to verify that the supplies and clocks are all coming up in the right sequence (**Figure 1**). It sounds simple, but getting this much access can be a serious problem if, for example, the traces between a local regulator and the core-voltage pins on an FPGA are all buried—as they likely will be if you tell the layout person only to keep the regulator right next to the FPGA.

The following principle will keep showing up: Understand the functions the board must perform and plan for the measurements necessary to verify

will approach the path. And it means understanding measurement techniques that could include anything from high-voltage probes to RF network analyzers. "Many times, the choice of signals is pretty obvious," says National's Shergill. "It's a signal path. Which aspects of the signal do you need to observe—at which points?" It's always helpful to have access to the pins, Shergill adds, but the need to have the reference design as close as possible to manufacturing-ready means that you sometimes are just not going to get that level of access. "Fortunately, there's often other access to the same signal somewhere else," Shergill says. Sometimes, physical accessibility isn't the only issue. Some nodes are simply too delicate to probe. "We've had designs in which just the leakage in a chip is enough to disturb a high-impedance node," he warns. You don't want to be sticking a scope probe into this area.

In such cases, it may be necessary to replicate the original signal at a test point, even at the expense of adding components. "We almost always end up putting additional components on the board to improve access to some signals," Fellner says. One advantage, he points out, is that if you are buffering a signal anyway, you can engineer the test point to be where you want it for debugging, not where it happens to have landed in the board layout. Fellner emphasizes the phrase "engineer access." Don't just drop in test points, he says. "Sometimes you can do a quick calculation to understand what you need. Sometimes you need to run a simulation."

Chan Lee, vice president of large-

## UNDERSTAND THE FUNCTIONS THE BOARD MUST PERFORM AND PLAN FOR THE MEASUREMENTS NECESSARY TO VERIFY THEM.

gressive power management has turned clock and power nets into analog signal nets. Engineers may need to verify, for instance, that the clock to a chip is arriving only when the clock gate is supposed to be conducting. They also need to verify that the transitions between clock-gating modes don't cause glitches. Alternatively, they may need to verify that a power-gating sequence brings down the supplies to an SOC in the right sequence and on the right ramps. Such requirements, in turn, imply a need for greater access to clocks and power connections at the point of use.

Almost everyone faces one instance of this issue in the form of power-up sequencing. With perhaps a dozen supply voltages on a board, designers must

them. "You must have the test defined, including the signal characteristics and the instrumentation," says IDT's Fellner. "That information will determine what kind of connection you have to provide on the board."

### ANALOG PATHS

In principle, planning for access to analog signal paths and control loops is similar to planning for power and clock grids. You must understand the functions the path performs, identify nodes the engineers will need to control and observe the path, and understand and provide for the measurements these activities require. But this simple statement can imply a lot of circuit analysis and an understanding of how your verification team

scale integration at Ambarella, says that designing accessibility is something you learn over time. “We mostly use back-of-the-envelope calculations to design test access,” Lee explains. “There are too many variables to do an accurate simulation.” Clearly, you must take into account the signal, the electrical environment at the node, the kinds of measurements debugging and verification engineers will have to make, and the equipment they will have to use.

Observing analog paths can be tricky, but controlling them can be even more challenging. If the signal originates off-board, you can use a signal generator with an appropriate matching network to drive the signal path. If the path begins at an onboard source, such as a thermal sensor in a critical chip, the problem may require more thought and, possibly, an analog switch to multiplex in an external reference. If the path begins at a DAC, it may be wise to provide an alternative way of jamming digital data into the converter.

### HIGH-SPEED I/O

The problem with high-speed serial I/O is the difficulty of probing the signals at all. “We typically have several different image-sensor interfaces on a board, including MIPI [mobile-industry-processor interface] and LVDS [low-voltage-differential signaling],” says Lee. “Some of these [interfaces] you can probe with a scope on a test point, and some you can’t.”

As speeds increase, the problem becomes critical. “Some of these buses are now differential signals at 3 GHz or more,” Fellner points out. “You can’t probe them anywhere. If you need access, you have to design in a differential repeater.” Even if you can capture the signals, though, they are difficult to interpret. “These signals are nondeterministic, so conventional automatic-test equipment has difficulties with them,” he says. You need a matching receiver to recover the data.

Working with these high-speed serial links is similar to working with other analog signal paths that originate in digital circuitry. It is a huge advantage to be able to control the digital ends of the link from software. “Typically, we put in more ways to control high-speed I/Os

## FPGAs HAVE ONE ADVANTAGE CPUs LACK: YOU CAN ALTER BOTH THEIR HARDWARE AND THEIR SOFTWARE.

from software on the host,” Lee says.

Software control usually includes a mechanism for forcing data patterns onto the bus and perhaps a loopback fixture so that the interface can check itself. Increasingly, however, these software techniques also exploit the fact that many high-speed interfaces offer programmable electrical characteristics. Even the simpler fast outputs may have programmable slew rate and drive strength. And more sophisticated transceivers have programmable pre-emphasis and equalization, sometimes with automated training sequences. These features exist so that the transceivers can adapt to the board. But you can also use this programmability to characterize the connection the transceiver sees for your own information or simply to retune the transceiver after you have attached a probe to the link.

### THE DIGITAL NETS

The largest category of nets on most boards is circuits that you can control and observe from an external host processor, a local processor on the board, or an FPGA. This category is in the realm of software-assisted debugging and verification. “There is a significant role for software,” says Stretch’s Thanawala. “If you can detect problems in software, you don’t need to probe the board. So our chip-development team tries to anticipate what the board-verification people will need.”

This development is no surprise to ARM’s Havens. ARM has watched as the evolution of software kernels, in-circuit emulators, and external debugging tools for the ARM cores has accelerated over the years and spread—from being simply about debugging software,

to controlling and observing everything in an ARM-based SOC, to controlling and observing chips outside the SOC, as well. The tools were there, and verification and debugging engineers were quick to exploit them, often routing signals back into an SOC from other parts of the board to bring more of the design under the CPU’s sphere of control.

Empowering the software to control digital blocks may require no additional hardware effort. Software-accessible registers usually control the digital hardware peripheral to a CPU. You must mentally walk through the functions the verification and debugging engineers will examine to ensure that the registers can exercise all the operations the verification team will need to watch. Observability is another matter. Thanawala suggests that a peripheral or an interface design should include an error register so that the driver software can detect and at least begin to diagnose problems. In addition, you might need to make the control state and internal registers of the device visible to the software, even if that task means routing them back into the general-purpose I/Os of the chip containing the CPU.

In this way, a processor—whether a stand-alone microprocessor, a microcontroller, or a core in an SOC—can be the heart of the access strategy for a board (Figure 2). But there is an even more powerful alternative: an FPGA. “You can make an FPGA the center of your design for accessibility,” says Brent Przybus, director of platform marketing at Xilinx, a company that should know: It has not only counseled customers on using FPGAs as debugging controllers but also developed its own reference designs for its application-platform strategy. In part, the principle for using FPGAs as accessibility gateways is similar to the notion for CPUs: The chip by its nature sits at the junction of many key signals on the board and contains important state. This situation is likely to be true whether the FPGA serves as an SOC at the heart of the board, as a coprocessor,



➤ Go to [www.edn.com/091008cs](http://www.edn.com/091008cs) and click on Feedback Loop to post a comment on this article.

➤ For more technical articles, go to [www.edn.com/features](http://www.edn.com/features).

or as an interface. FPGAs have one advantage CPUs lack, however: You can alter both their hardware and their software. That flexibility gives you access to internal nodes in logic functions. For example, Przybus points out, implementing a DDR3 DRAM interface in an FPGA is one of the few ways to get enough access to troubleshoot that fast memory interface. With this approach, you not only see internal nodes in the controller but also can force data patterns through the interface, fire off transactions, and bring out a copy of the data that is traversing the virtually unprobable serial links.

You can insert—or the verification team can reprogram the FPGA to insert—a virtual logic analyzer into the FPGA fabric, so an external multiprocessor can trigger, trace, and control nodes in the chip. “It makes sense to route signals from other parts of the board into the FPGA instead of out to a test header,” Przybus says. If you don’t feel like designing your own logic analyzer, your FPGA vendor has one available—in Xilinx’s case, ChipScope. The next step in sophistication is to put the controlling CPU itself into the FPGA as a core. Both proprietary fast RISC cores and industry-standard microcontroller cores are now small enough to slip almost unnoticed into a design. The CPU core in the FPGA allows you to write both the control code for the logic analyzer and the verification or debugging routines for other parts of the board and to run the code without an external host computer.

These ideas are obviously valuable if you already have an FPGA in your design. You can simply reprogram it for verification and debugging mode, and

hardware additions to the board should be minimal. Some designers are finding that it’s worthwhile to move to a higher-pin-count FPGA to route more signals to or from the chip, extending the FPGA’s reach to more of the design. Some even argue that this capability justifies using a low-cost FPGA in a design in which a standard product or an ASIC could do the job.

In design for accessibility, it is vital to understand the verification and debugging engineers’ needs. Having a systematic approach and soliciting the views of other experienced designers can be a big help. It may be possible to prevent the oft-repeated question from evaluation, verification, and failure-analysis engineers: “How the heck am I supposed to get to that point?” **EDN**

## FOR MORE INFORMATION

**Ambarella**  
[www.ambarella.com](http://www.ambarella.com)

**ARM**  
[www.arm.com](http://www.arm.com)

**Integrated Device  
Technology**  
[www.idt.com](http://www.idt.com)

**Stretch Inc**  
[www.stretchinc.com](http://www.stretchinc.com)

**Xilinx**  
[www.xilinx.com](http://www.xilinx.com)

You can reach  
Executive Editor  
**Ron Wilson** at  
1-510-744-1263 and  
[ronald.wilson@  
reedbusiness.com](mailto:ronald.wilson@reedbusiness.com).

