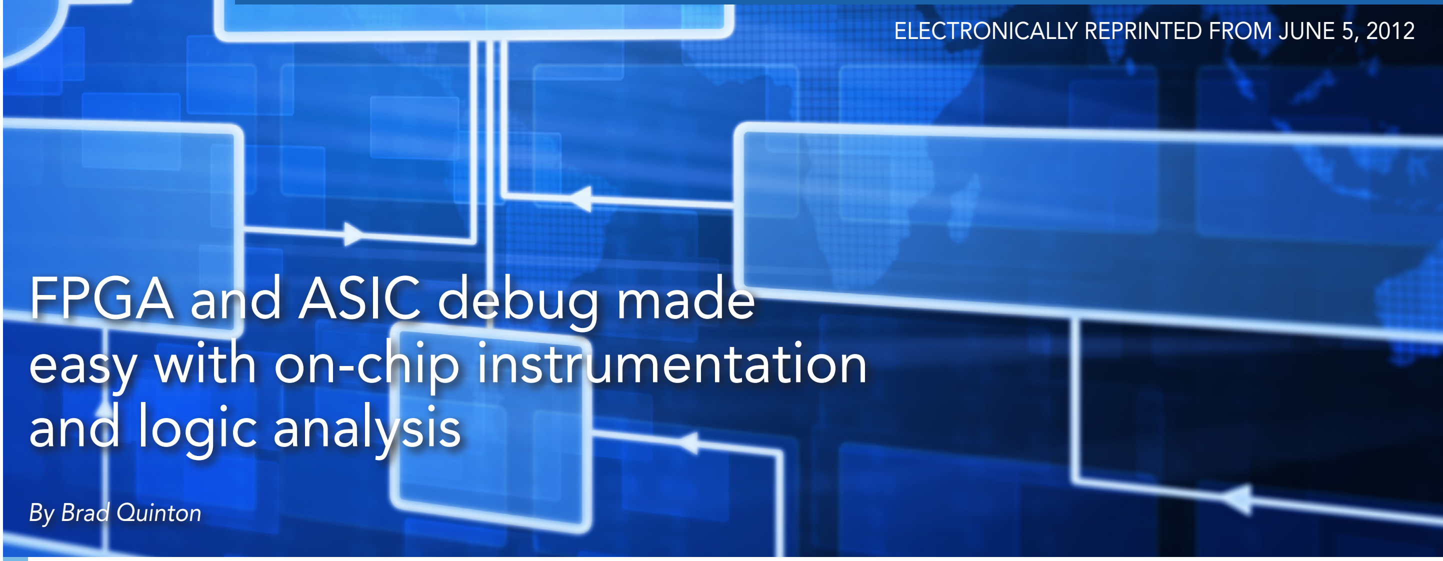
Embedded COMPUTING DESIGN
www.embedded-computing.com
ELECTRONICALLY REPRINTED FROM JUNE 5, 2012

Software | Electronic design automation



FPGA and ASIC debug made easy with on-chip instrumentation and logic analysis

By Brad Quinton

Modern ASICs and FPGAs are tedious and time-consuming to verify and validate. Adding small, highly efficient on-chip capture infrastructure to the design makes this job much easier by providing 10x the visibility of instrumentation points for a given area. In addition, by using compression algorithms, logic analysis capture stations can capture data for 10x or more capture depth.

ASICs and FPGAs have become massively complex, particularly for System-on-Chip (SoC) designs involving multiple cores. With this complexity comes longer and more tedious debug and validation cycles. Unfortunately, when something fails or goes wrong, gaining access to test points in highly integrated designs is next to impossible. Unless you want to spend weeks shooting in the dark at random errors while running through multiple prototypes, on-chip instrumentation is no longer optional; it's a critical must-have. Figure 1 shows an overview of the debug process using on-chip instrumentation.

While there are a number of ways to add instrumentation to FPGAs, a distributed approach using an instrument network is emerging as the preferred method, as it maximizes the number of potential observation points while minimizing silicon area or look-up table utilization requirements. Also critical to efficient debug is deep trace capture to see how the various parts of a system interact over time. Finally, designers must be able to observe the interactions of multiple devices and clock domains, both on- and off-chip, all fully time correlated for a true system-level perspective.

Taken together, innovations including flexible and complete access to observation points, deep trace captures, and system-level views have the potential to change the game for FPGA and ASIC debug from long and arduous to fast and efficient.

Debug challenges

Before reviewing different approaches to implementing embedded instrumentation, it's helpful to understand why instrumentation is necessary in the first place. The biggest reason is simply the ever-growing functionality in each system.

Whereas in the past there were plenty of probe points (external I/O on the devices) to choose from, it's no longer possible to observe what's going on since in most cases the key interfaces are now inside devices. Current-generation FPGAs have 100x the number of functions running in parallel compared to five years ago, yet the number of external outputs has stayed the same. From the perspective of a developer attempting to debug unexpected behavior, modern chips are nothing more than a big black box.

As if that weren't enough, while the power of simulators continues to improve in a linear fashion, every increase in parallel functionality adds an exponential increase in potential combinations. Since simulations run on one combination at a time, it's not possible to cover all the functionality in pre-silicon simulation runs.

This inability to adequately simulate all the possible permutations in pre-silicon has led to FPGA-based prototyping before design completion. Particularly at the prototype level, access to observation points is extremely helpful in debugging functional issues quickly and efficiently.

Another debug challenge is the emergence of embedded software on silicon. More and more FPGA and ASIC designs include one or more processor cores. Such systems can include a complex mix of software, firmware, embedded processors, GPUs, memory controllers, and other high-speed peripherals. This increased functional integration combined with faster internal clock speeds and complex, high-speed I/O is making it harder than ever for developers to deliver a functional and fully validated system.

On-chip signal capture

Back when systems involved multiple chips and components it was easy to move logic analyzer probes around to look at different signal combinations. Even with the move to on-chip instruments, the need to flexibly move virtual logic analyzer probes to different signal points remains a constant. Since the designer can't anticipate every variable or potential application for a given chip, the more signal capture points available, the better.

A traditional ASIC approach uses a mux network with shared select signals (one per mux level) and provides n/m different signal combinations, where n is the number of probe points and m is the number of signals viewed concurrently (debug bus width). This is the most restrictive but simplest option, as it leverages simple multiplexers. To be effective, this approach requires significant up-front time to create groups of signals that correspond to every possible debug scenario, and once the capture points are in play, designers can only look at signals that are in the same group. This process is demanding, time-consuming, and highly unlikely to capture all debug scenarios.

The other extreme is to create a full crossbar mux that gives complete signal flexibility, which requires m muxes of n:1 in size. This can get expensive relative to area very fast, making this approach impractical for all but the smallest cases.

The middle ground is to either increase the number of select signals inside the mux structure or create a number of duplicate groups with different signal ordering. The shared select mux and the mux with additional select signals are both implemented in many homegrown approaches. While shared select mux schemes can handle common and expected debug scenarios, they still fall short of the ideal complete coverage. Thus, they are ill-suited for unexpected problems and can often lead to inefficient implementations, as signals are repeatedly connected to multiple multiplexers.

It is possible to find a more elegant and efficient solution by leveraging multistage, unordered networks, often called *concentrator networks*. This new approach effectively creates an observation network and is becoming commercially available.

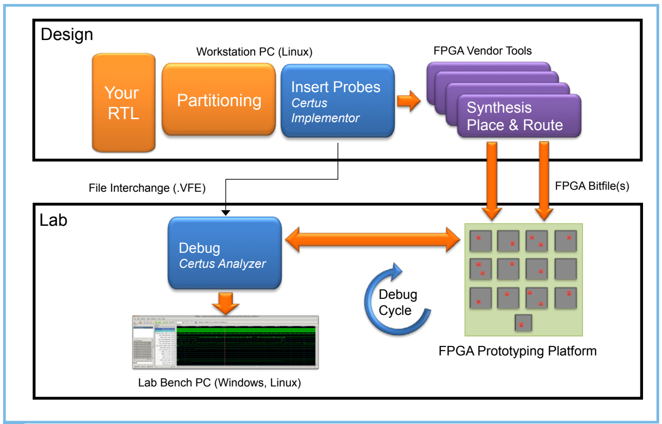


Figure 1 | The instrumentation and debug cycle is critical to detect errors in FPGA prototypes.

Signal capture approach	Signal visibility
Shared select mux	n/m
Enhanced mux	2^(m*log(n/m)/log2)
Full crossbar	2^n
Observation network	2^n

Table 1 | An observation network provides the same level of signal visibility as muxes while requiring similar or less die area.

Using a unique network architecture and complementary routing algorithms, an observation network provides the signal flexibility of a full crossbar mux while in most cases requiring no more die area than shared simple muxes. Table 1 shows a comparison of signal visibility calculated using the different approaches.

With an observation network, designers use automated tools to implement on-chip signal capture probes in the Register Transfer Language (RTL). At the design stage, there is no need to worry about different signal combinations or ordering since every combination will be available. The result is an observation network that grows linearly with the number of signals. This approach moves the complexity of determining routing off the silicon and into software. While producing a significant area/performance improvement, an observation network requires sophisticated algorithms to determine routing, making it difficult to use without commercial software to control signal selection.

How significant an advantage does the network approach offer over a simple mux in terms of observation point visibility? Take this example, where 256 signals are probed (n) with 32 signals visible concurrently (m):

- **Simple mux:** Number of signal combinations (visibility) = 256/32 = 8
- **Observation network:** Number of signal combinations (visibility) = 2^256 = 1.2 x 10^77

The difference is 76 orders of magnitude. While the first approach is highly restrictive, the observation network approach provides any possible combination of signals. For roughly the same cost, the observation network provides a huge advantage with its increased flexibility.

Maximizing capture depth

For debug challenges that span hardware and software, the ability to capture long traces is critical to track down problems that show up over thousands or millions of clock cycles. Post-silicon and on FPGAs, deep capture is vital to see how the overall system works, as many of the bugs that escape verification take a long time to emerge. Furthermore, most software-driven functionality spans hundreds of thousands to millions of clock cycles.

Traditional instrumentation approaches capture the information as it is received from the observation probe using one entry in the internal RAM for each clock cycle of data captured. With this approach it is difficult or impossible to capture more than a few thousand clock cycles at a time without putting an unacceptable strain on internal memory resources. For that reason, compression techniques are now starting to be used to boost capture depth.

However, most well-known compression algorithms are poorly suited to trace compression, having been developed for visual media and communications applications. Specialized trace compression layers that use multiple compression techniques together, each specifically tailored to common trace data patterns, are now commercially available. For most real-life applications this provides 10-1,000x more depth with no loss in resolution.

Efficient system-wide debug

The last piece of the puzzle to more efficient FPGA and ASIC debug is a time-correlated, system-wide view that spans multiple clock domains running in parallel. When problems require correlation across multiple instrumented areas, the designer is looking at a time-consuming process of obtaining individual traces and then correlating events manually. For instance, an average ASIC prototype on an FPGA-based prototyping platform consists of two to three clock domains per FPGA across four to eight FPGAs. This means the designer will need to debug anywhere from eight to 24 clock domains individually. Tracing each of these 24 domains one at a time and manually piecing together the results is time-consuming and error-prone.

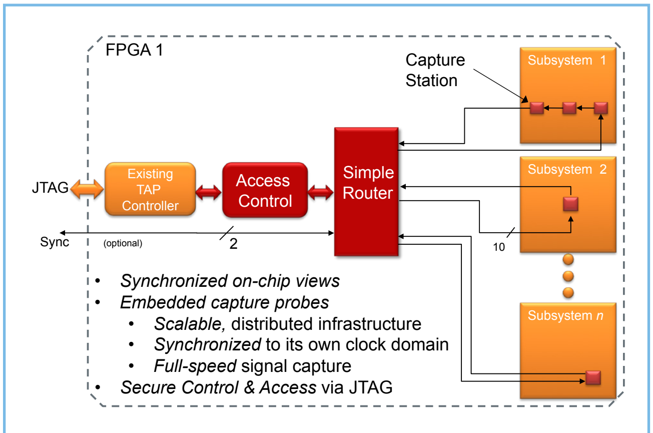

FPGA 1
Capture Station
Subsystem 1
Subsystem 2
Subsystem n
JTAG
Existing TAP Controller
Access Control
Simple Router
Sync
(optional)
2
10
• Synchronized on-chip views
• Embedded capture probes
• Scalable, distributed infrastructure
• Synchronized to its own clock domain
• Full-speed signal capture
• Secure Control & Access via JTAG

Figure 2 | Time-correlated views speed system-level debug.

A much more efficient approach is to use logic analyzer software to produce a time-correlated view from independent instruments operating in multiple clock domains and across multiple devices, as shown in Figure 2. Specialized debug software can collect data from each instrumented area of the chip, reverse the compression algorithms, and then align the captured data to produce a system-wide, time-correlated view. This leads to a single trace capture and debug scenario, both saving time and providing simultaneous hardware debug of many functional units and clock domains. This process often reveals emergent system behaviors that were never considered when the device was architected.

Innovations handle the unexpected

With increasing complexity and limited access to probe points, ASIC and FPGA validation and debug has become tedious and time-consuming. As more and more functionality is integrated into each chip, physical access to probe points has become impossible. The challenge then is to incorporate enough on-chip observation points to not only handle expected debug scenarios, but unexpected ones as well.

A key innovation that enables faster and more efficient validation and debug of even the most complex designs is an observation network. Compared to traditional shared select mux approaches for observing signals, the observation network delivers significantly more signal combinations with similar die area requirements.

Other innovations supporting more efficient debug scenarios include the use of advanced compression algorithms to boost on-chip memory capture depth and the emergence of logic analyzer software that produces a time-correlated, system-wide view that spans multiple devices and off-chip instruments. ECD

Brad Quinton *is the chief architect for the Tektronix Embedded Instrumentation Group.*

Tektronix
brad.quinton@tektronix.com
www.tektronix.com

Follow: