Boost FPGA Prototype Productivity by 10x

With On-chip Instrumentation and Logic Analysis

Introduction

Modern ASICs have become massively complex due in part to the growing adoption of system on chip (SoC) development methodologies. With this growing complexity comes inevitably longer and more tedious debug and validation cycles. This is especially true for FPGA-based ASIC prototypes or simply FPGA prototypes. Because re-spinning an ASIC is so expensive and time consuming, both the hardware and software in the ASIC must be fully evaluated in as close to real-world conditions as possible and all bugs need to be isolated and corrected before tapeout.

On-chip instrumentation eases RTL debugging tasks in FPGA prototypes by providing access to signals that otherwise would not be accessible. In the past teams often used the troubleshooting tools provided by FPGA or EDA vendors that made it possible to access a select number of signals and perform basic analysis, or to route internal signals to external instruments such as logic analyzers.

Unfortunately, FPGA vendor debug tools consume considerable chip area and block RAM, and have significant limitations. For simple designs with low utilization, plenty of available block RAM resources and relatively quick implementation runtimes, the FPGA-vendor supplied tools can be “good enough,” assuming all goes according to plan. However, for complex designs like FPGA prototypes, which are already resource constrained, multiple synthesis and/or place-and-route cycles may be required to iterate on signals you want to observe or to adjust capture depth to isolate on a particular issue. For difficult bugs, these limitations can add days or weeks to debug cycles with the issue compounded further by long or difficult implementation flows. For particularly daunting bugs, teams can be forced to develop workarounds and tapeout with known bugs.

Given the vital importance of FPGA prototyping in ensuring that ASIC design projects stay on schedule and meet expectations for features, performance and reliability, using “good enough” debug tools represent a false economy. Clearly, there is a need for embedded instrumentation in the FPGA prototype design process, but the tools need to be fully capable of pinpointing the root cause of erroneous behavior quickly and efficiently. This means they must provide flexible and comprehensive access to observation points and deep trace captures to support debugging both hardware and software across multiple clock domains and complete systems comprised of multiple FPGAs and boards. The debugging tools must also make the most of limited on-chip resources and must not place artificial restraints such as taking data off chip and thereby limiting clock rates.

Tektronix®

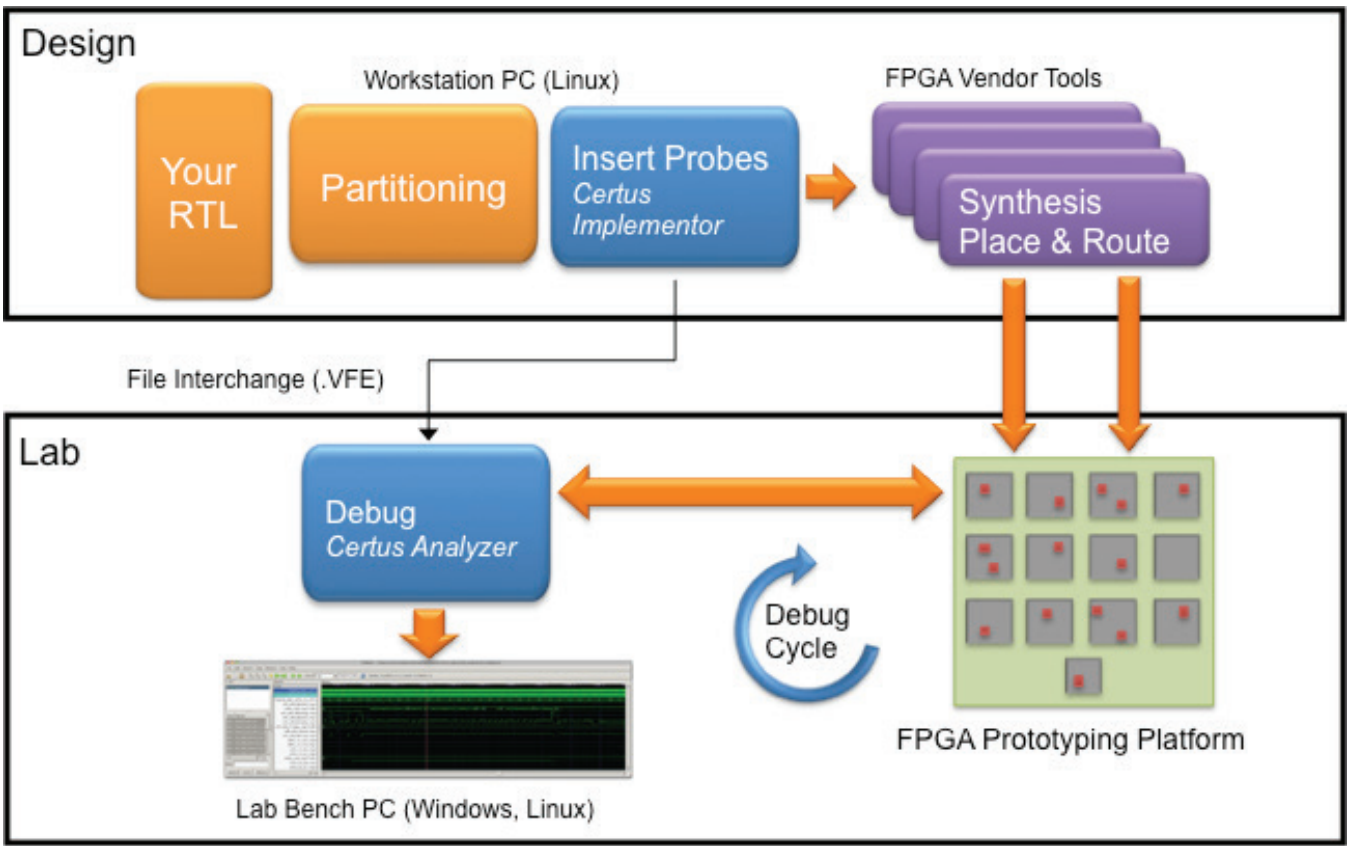

Figure 1. Overview of the instrumentation and debug cycle for FPGA prototypes.

As detailed in the following pages, the Tektronix Certus ASIC Multi-FPGA Debug Suite delivers a number of innovations that taken together can provide a 10x advantage for debugging productivity. These innovations include flexible and complete access to tens of thousands of observation points, deep trace captures and system-level views. Certus users typically observe between 1,000 and 50,000 signals simultaneously across 2-10 FPGAs and 5-12 clock domains while achieving capture depths upwards of 1M clock cycles. This represents orders of magnitude of more visibility than what's possible with the FPGA vendor tools. Figure 1 provides an overview of the instrumentation and debug cycle for FPGA prototypes using Certus.

While there are a number of approaches to adding instrumentation to FPGAs, a distributed approach using an instrument network as supported by Certus maximizes the number of potential observation points while minimizing silicon area and block RAM utilization requirements. Also critical to efficient debug is deep trace captures to see how the various parts of a system interact over time. Certus uses advanced compression techniques to significantly extend capture times. Finally, Certus allows designers 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.

Debug Challenges

Key Takeaway:

As ASICs have become more complex, complete system testing and debug using FPGA prototypes has become increasingly popular. Gaining access to test points for efficient debug requires specialized on-chip instrumentation.

FPGA-based prototyping is becoming the technology of choice for the design and verification of complex ASICs. The relatively low cost of FPGA-based prototypes means that multiple systems can be provided to the hardware and software development teams. The high performance of FPGA-based prototypes means they can be linked into to the rest of a real-world system without creating specialized subsystems.

A modern ASIC is rarely created from the ground up – it typically includes a mixture of newly-developed custom functions along with existing Intellectual Property (IP) blocks. The existing IP may be pulled from other internal projects or acquired from a third-party supplier. In some cases the IP blocks will be well-known and trusted. In others they will be new to the design and verification teams. Even in the case of a well-known IP block, a new design may add new stresses that could cause it to behave unexpectedly.

In addition to the silicon itself, modern ASICs involve hundreds of thousands of lines of firmware and tens of millions of lines of software application code. For most projects, it is vital that the software has been completed and fully tested before final silicon becomes available. This means that software developers need some form of prototype that can boot the operating system and run firmware and application software at close to real-time speeds. Moreover, teams need the ability to test the interaction of the hardware and software as a complete system to determine whether problems are the result of hardware or software flaws.

As if that weren't enough, the power of simulators continues to fall behind. While simulation speed improves 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. FPGA prototypes meet these demanding pre-silicon verification requirements; however, from the perspective of someone attempting to debug unexpected behavior, they are essentially a very complex black-box without proper instrumentation.

On-chip Signal Capture

Key Takeaway:

The Certus observation network provide access to any combination of signals without the need to iterate back to the RTL while using no more chip resources than a simple mux, which is limited to pre-defined groups of signals.

FPGA vendor-provided debug tools require the up-front definition of (a small number) of signals of interest and in most cases a complete re-run of the synthesis and place-and-route flow for each new selection. For simple designs with quick runtimes, this is an adequate approach. However, for FPGA prototypes with very complex designs, long runtimes, multiple clock domains and functionality split across multiple FPGAs, this approach leads to an extreme bottleneck in the debug process. For instance, observing 32 signals at a time in an FPGA prototyping system that contains over 10 million RTL-level signals partitioned across 10 FPGAs is like trying to read a novel using a microscope.

To attack this bottleneck it is important for the debug team to be able to see a large number of signals with the same embedded instruments and to, focus on different areas of the design without the need for a synthesis and place-and-route stage in the loop.

Beyond the FPGA vendor tools, there are other debug tools that attempt to address these issues using a mux network. This mux network provides n/m different signal combinations, where n is the number of probe points and m is the number of signals viewed concurrently (i.e. "debug bus" width). Although simple to understand and implement, this is a very restrictive option. To be effective, this approach requires significant upfront time to create groups of signals that correspond to a variety of debug scenarios. Even with this upfront effort, there will almost always be a need for multiple synthesis or place-and-route runs, further reducing productivity.

Another simple approach would be to create a full crossbar mux that gives complete signal flexibility, but this requires m muxes of n:1 in size. This approach can get very expensive quickly relative to area, making it impractical for all but the smallest cases.

Certus offers a far more efficient solution by leveraging multi-stage, unordered networks (often called concentrator networks). This approach in effect creates an observation network. Using this unique network architecture and complementary routing algorithms, Certus provides the signal flexibility of a full crossbar mux while in most cases requiring no more die area than shared simple muxes. A comparison of signal visibility with the different approaches is shown in Table 1.

Approaches	Visibility
Shared Select Mux	n/m
Full Cross-bar	2^n
Observation Network	2^n

Table 1. Calculation of signal visibility for muxes and an observation network.

With Certus, designers use automated tools to implement on-chip signal capture probes in the 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 essentially moves the complexity of determining the routing off the silicon and into software. This significant area/performance improvement is accomplished through the use of sophisticated algorithms that determine routing and control signal selection.

How significant an advantage does Certus 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 (i.e., "visibility") = 256/32 = 32

Observation Network:

- Number of signal combinations (i.e., "visibility") = 2^{256} = 1.2 x 10^{77}

The difference is 76 orders of magnitude! While it's a huge difference, the more practical reality is that the first approach is highly restrictive while the observation network provides any possible combination of signals. Clearly, for roughly the same chip utilization there is a huge productivity advantage to the increased flexibility.

On top of debug flexibility, this approach helps to manage resource utilization. Let's take a look at this from the perspective of block RAM utilization. Using the FPGA vendor tools, probing 1,024 signals on one FPGA would require a 1024-bit-wide memory. Assuming a trace depth of 1,024 words, with 36 Kb block RAMs this would require 1024 * 1024/36864 = 29 block RAMs. Certus, on the hand, would just need two block RAMS and provide tens of thousands of samples for any combination of a subset of 1,024 signals probed.

Other vendors allow a mux network as described above with up to eight groups (i.e., an 8-input mux). This scenario, would require ((1024/8) * 1024) / 36864 = 4 block RAMs, still twice as many as the Certus approach. The drawback of this approach is that visibility is restricted to pre-defined groups of 128 signals; any combination of these 1,024 signals can't be seen. For difficult bugs, or even not so difficult ones – you can often end up wanting to see signals that span multiple groups. Gaining access to those signals means re-running synthesis or place-and-route. This translates into hours and hours of lost productivity. By contrast, the advanced observation network of Certus guarantees a view of any combination of 128 signals from the 1,024 probed. This offers significant time savings — at least a 10x improvement — for large or highly utilized designs that can have routing times that run for many hours and often do not complete without multiple attempts.

Maximizing Capture Depth

Key Takeaway:

Certus performs extensive compression to enable deep trace captures with minimal on-chip resources. In conjunction with the flexible observation network, Certus avoids the limitations of, or requirement for, off-chip storage.

For debug challenges that span hardware and software, the ability to capture long traces is critical to tracking 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, since 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.

The FPGA vendors' and other debug 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, Certus employs advanced compression techniques to boost capture depth.

It is important to note that most well-known compression algorithms are poorly suited to trace compression because they were developed for visual media and communications applications. By contrast, Certus employs specialized trace compression layers that use multiple compression techniques tailored to common trace data patterns. For most FPGA prototype applications this provides between ten and a thousand times more depth with no loss in resolution.

Another alternative approach used in FPGA debug is to use off-chip memory to improve capture depth when there are limited block RAM resources available for debug. While suitable for some applications where the prototyping board has been pre-provisioned with this capability, this approach typically requires time division multiplexing to offload data. As such, it places limits on the maximum frequency for capture, typically around 60 MHz. For FPGA prototyping applications, higher frequencies are commonly used in order to prototype real-world conditions. Furthermore, off-chip capture is typically limited to a single clock domain at a time. For modern SoC debug on FPGA platforms, capturing asynchronous signals in their respective asynchronous clock domains and then viewing the signals in parallel is the most typical practice. Since Certus uses block RAM much more efficiently, there is no need to store off chip and therefore no restriction on clock speeds or the number of clock domains.

Efficient System-wide Debug

Key Takeaway:

The single trace capture and debug scenario in Certus delivers at least a 10x improvement in productivity for effective hardware debug of many functional units and clock domains simultaneously.

The last piece of the puzzle to more efficient FPGA prototype debug is a time-correlated system-wide view that spans multiple clock domains and FPGAs running in parallel. Historically, when problems require correlation across multiple instrumented areas, the design team would be looking at a time-consuming process of obtaining individual traces and then having to correlate 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 that with the FPGA vendor and other debug tools the team 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.

A much more efficient approach is to use logic analyzer software provided with Certus to produce a time-correlated view from independent instruments operating in multiple clock domains and across multiple devices as shown in Figure 2. The Tektronix software collects data from each instrumented area of the chip, reverses the compression algorithms, and then aligns the captured data to produce a system-wide time-correlated view. This leads to a single trace capture and debug scenario, providing at least a 10x improvement in productivity and an effective hardware debug of many functional units and clock domains simultaneously. Often this process reveals emergent system behaviors that where never even considered when the device was architected.

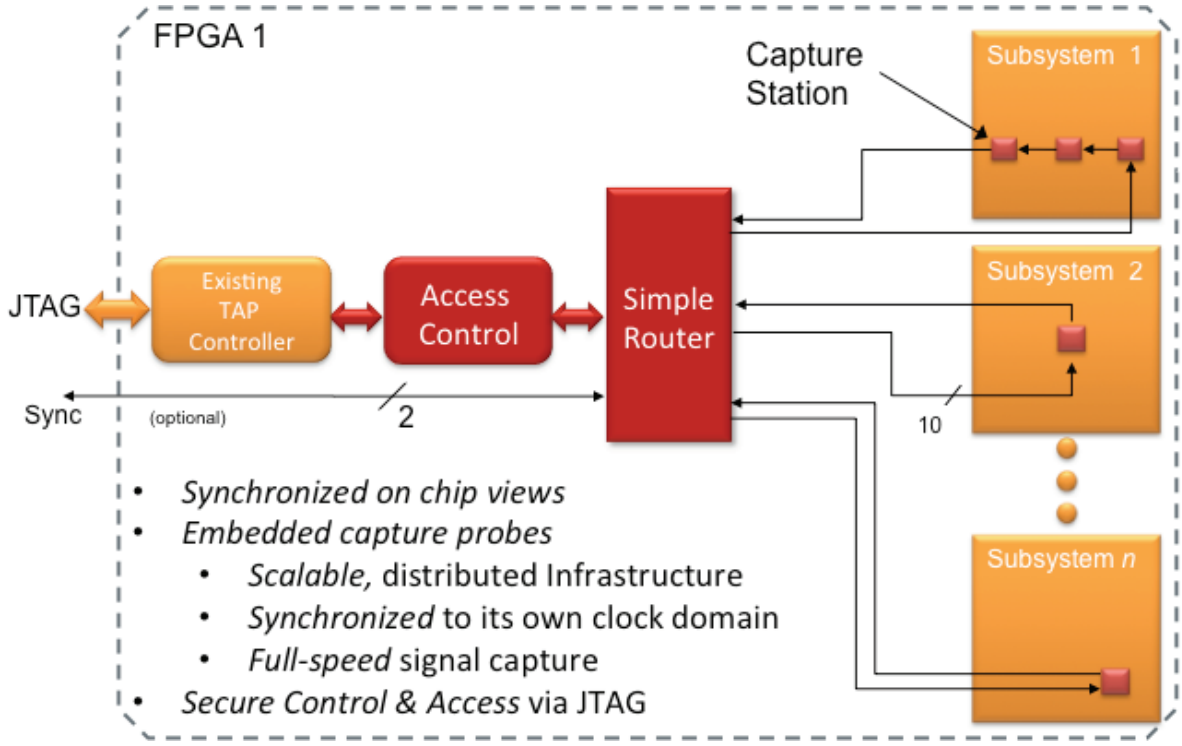

Figure 2. Time-correlated views speed system-level debug.

One alternative approach is to perform multiple captures and route them to external instruments, such as a logic analyzer, for further analysis. In Certus, due to deep memory capture, this is rarely needed. Obviously, setting triggers and obtaining system-wide traces is more efficient since there is no need to route signals to the external interface nor is there a need to set up the logic analyzer to map to the RTL names.

10x Productivity

Key Takeaway:

Certus delivers a number of key features for FPGA prototyping that significantly reduce the time spent debugging the system. Taken together, Certus delivers at least a 10x increase in overall productivity versus other debug strategies.

With increasing complexity and limited access to internal signals, FPGA prototype debug has become tedious and time consuming. As more and more functionality is integrated into each ASIC generation, this challenge increases. Incorporating a distributed and flexible on-chip observation infrastructure like Certus makes it possible to achieve an order of magnitude (10X) increase in debug productivity.

The current FPGA vendor tools as well as those from EDA vendors require extensive block RAM resources to capture a large number of signals for multiple clock cycles, impacting overall FPGA resources utilization and further extending already long runtimes. In addition, when using vendor tools, teams must resynthesize repeatedly to change the set of signals being observed. The result is long and tedious debug cycles. For complex FGPA prototypes, those tools fall well short of “good enough.”

Tektronix Certus delivers a number of key innovations that enable faster and more efficient debug of even the most complex designs. For instance, the Certus observation network allows the debug team to access any combination of signals without time-consuming resynthesize cycles while requiring about the same chip resources as a simple mux. Other innovations supporting more efficient debug scenarios include the use of advanced compression algorithms to boost on-chip memory capture depth and analyzer software that supports a time-correlated, system-wide view that spans multiple devices and off-chip instruments.

Summary: Debug Time and Productivity Comparison

How important is on-chip instrumentation to maximizing debug productivity and getting the most out of design team resources? The table below provides an at-a-glance analysis of the various steps involved with debug and performance optimization of complex FPGA prototypes.

	No instrumentation	Basic instrumentation	Certus
Determine Signal Grouping	Not required.	Extensive effort 2-3 days or more.	Not required
Place Observation Points	Not required.	Automated, half day.	Automated, half day.
Run Analysis to Identify Bugs Following Functional Test	Incomplete analysis. Teams must make educated guesses about problems and resolve finger-pointing across teams without clear insight into what's actually going on. Weeks to months of effort.	Assuming signal groupings are effective, able to identify 80-90 percent of bugs in 1-2 weeks of effort. Remaining bugs will require multiple resynthesis/place-and-route runs, significantly impeding productivity.	Full access to all signals allows bug locations to be quickly identified without the need for productivity stealing resynthesis/place-and-route.
Time-to-root Cause	Unknown. Given the complexity and number of variables, finding the root cause of bugs using a trial-and-error method can be exceedingly slow.	3 days to 2 weeks. With limited access to signals groups, invariably resynthesis will be required slowing productivity, especially for more complex prototypes.	1-2 days. The advanced Certus observation network ensures full signal visibility along with deep captures, so the root cause of elusive bugs that occur over a long time span can be quickly identified.
Conduct Optimization Testing	Limited ability to improve design without test and analysis, so skip this step.	Slow process since there is no ability to accurately correlate events across the entire system. Therefore, designers must manually match up system events using spreadsheets. This arduous process usually adds several more weeks.	With full system-wide visibility, deep memory capture and time-correlated triggers, performance bottlenecks – whether hardware or software – can be quickly identified.
Summary	Debug is a major obstacle to project completion.	Design can be completed, but lack of signal visibility and limited capture depth extend debug time by many weeks. Some bugs may go unresolved.	Debug is a swift and comparatively painless process, helping to speed up project completion.

5/12 54W-28133-0

Tektronix®