

OCP Performance Monitoring with Automated Insertion of Programmable Instruments

Open Core Protocol™ offers many benefits, including simplified integration and implementation of multi-core designs. OCP's unified interconnection scheme allows designers to create complex systems composed of third-party IP and legacy IP. As with most complex and heterogeneous systems, the ability to visualize and analyze performance characteristics is key to understanding (and verifying) system behavior, and subsequently fine-tuning the system for optimal power and performance. Naturally, visualization and performance analysis capabilities are necessary during initial modeling and design verification and later, subsequent to chip tapeout, in the lab, in order to observe the device running at-speed in the target system.

Such on-chip capability requires embedded instrumentation IP. Given the predictable reluctance to on-chip instrumentation, a lightweight, easily inserted, and dynamically programmable scheme is desirable. These characteristics allow embedded instruments to be used for a variety of functions including performance monitoring, assertions, functional analysis, and debug, – and even fault insertion and transaction stimulus. While these functions have utility in hardware debug, they are nowadays more often utilized by software and systems engineers who can begin to leverage these on-chip resources in unique and compelling ways – turning initial reluctance into opportunity for streamlining a complex and time-consuming test and validation process.

A few additional requirements must be considered:

- **Configurability:** Given the configurability of OCP, the embedded on-chip instruments must also be configurable.
- **Flexibility:** The SoC will be composed of a variety of buses, interfaces, and IP blocks; therefore, the solution must not be limited to OCP circuits.
- **Easy insertion:** Such a configurable and flexible system requires automated insertion.

Ultimately, the solutions must be compliant to the OCP Debug Specification. However, even in the absence of IP cores or switch fabrics compliant to the OCP Debug Specification, practical performance monitoring implementations can be realized today.

The Solution: Instruments, Insertion, and Applications

This trifecta is a combination of 1) programmable instruments, 2) automated insertion tools, and 3) instrument programming and analysis applications. Following is a description of an embedded-instrumentation solution delivering a comprehensive set of performance monitoring and analysis functions.

The overall objective of this solution is to provide the user with a spectrum of visualization and analysis methods, from coarse views on many interfaces (e.g., aggregate system throughput or worst-case latency) down through increasingly granular, targeted, and highly specific views at the socket level (e.g., discrete read/write transactions). This approach is consistent with most conventional analysis, diagnostic, and debug techniques; it seamlessly marries broad views of

system behavior with “telescoping” views that are informed by the discoveries at each level of the visualization and analysis process.

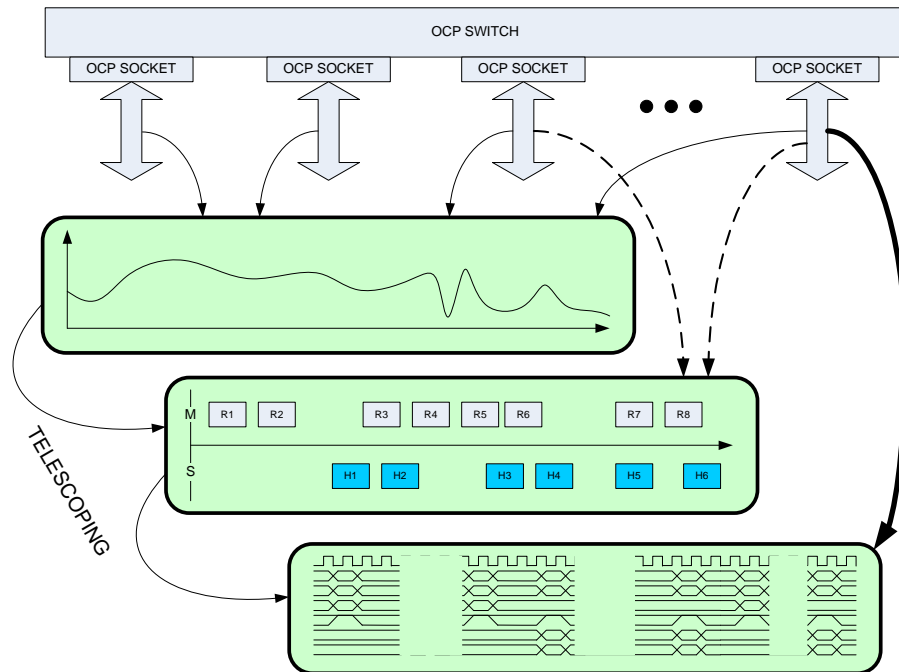


Figure 1. Dynamic Multi-level Views

The solution presented here is configurable, flexible, and lightweight. While our primary interest is the OCP interconnect, the instruments shown can be applied to almost any bus or interface in the design. The entire solution is user-configured and inserted into the RTL design automatically.

Perhaps the most important element in the solution is the programming and analysis application. This tool provides the user a graphical interface and a set of high-level commands (through a Tcl interface) to program and operate the on-chip instrumentation infrastructure. All command/control and data extraction is done through a standard IEEE 1149.1 (JTAG) interface.

The basic suite of OCP performance monitoring functions includes the measurement of aggregate throughput per interface, master/slave throughput, instantaneous or average request/response latency, instantaneous or average event latency, and worst-case latency.

Basic Instrument Configuration

Such performance monitoring functions can be realized with a distributed instrumentation scheme as shown in Figure 2. There are three forms of instruments used; signal multiplexors, Pattern Match Engines, and Transaction Engines. Each is a user configurable instrument (e.g., bus width, states, GPIO). All instruments are in-system programmable and run “at-speed” without disrupting the normal operation of the system.

The configuration shown is an optimal balance of functionality and low area overhead. In this configuration, each multiplexor and Pattern Match Engine operate autonomously, consequently multiple interfaces can be monitored concurrently.

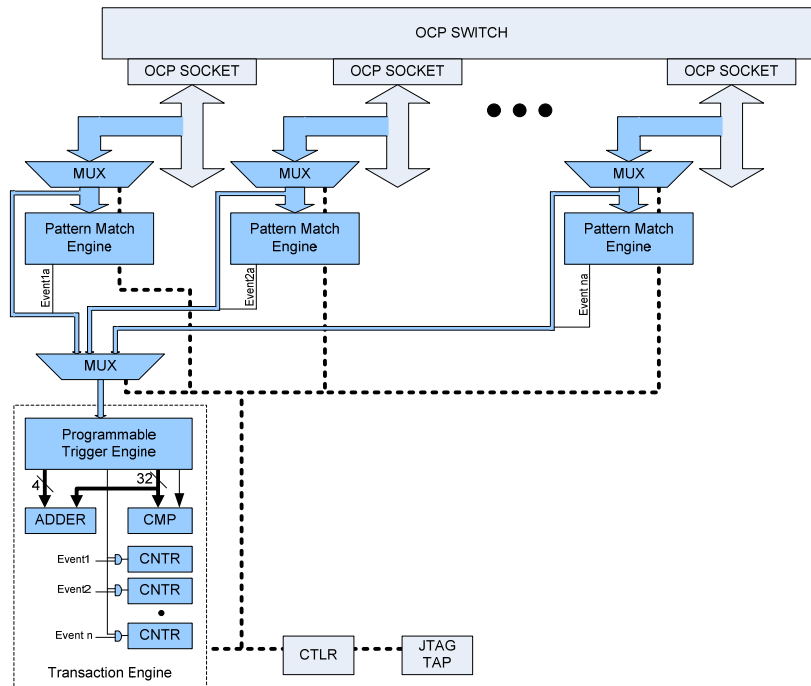
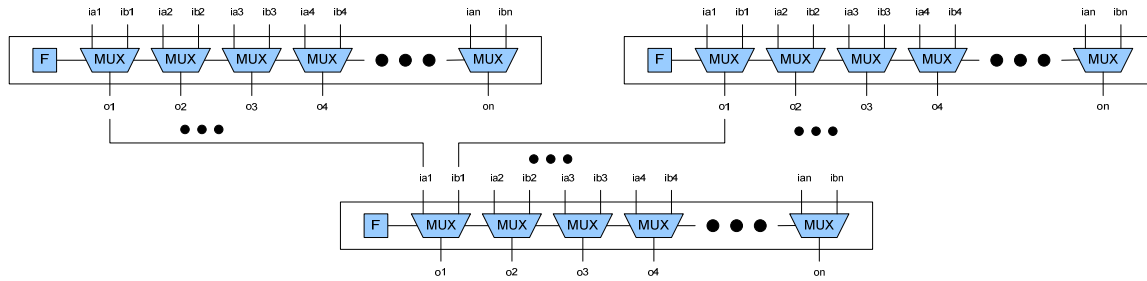


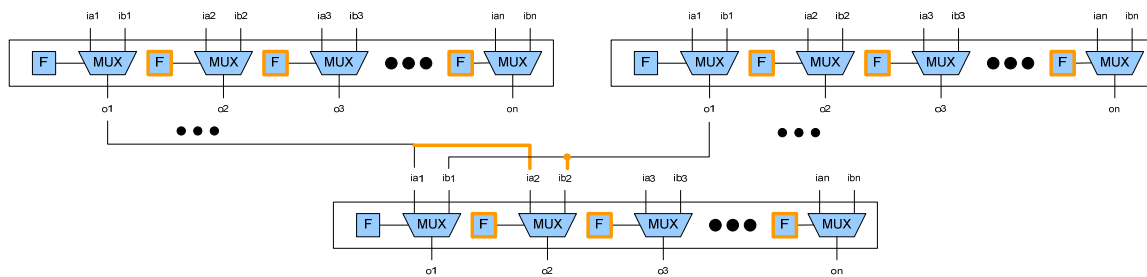
Figure 2: Performance Monitoring Instrumentation

The multiplexors are used to reduce the number of signals presented to the Pattern Match Engines and Transaction Engine. At runtime the user specifies which signals to monitor. This is done through the programming and analysis application.

The multiplexors can take on a variety of forms depending on the application, flexibility, and area overhead requirements. The Bank Select Multiplexor is the smallest and least flexible (Figure 3(a)). In this configuration, signal-bank-a or signal-bank-b is selected through a serially programming register. The Bit Select Crossbar Multiplexor (Figure 3 (b)) offers additional flexibility by providing a serial programming register for each 2:1 multiplexor, and additional signal fan-out between multiplexor stages. This gives the user the most flexibility to select a greater combination of signals. In practice, the Bank Select Multiplexor (a) can be used in most performance monitoring configurations, whereas the other configurations may be more appropriate if a variety of validation and debug functions are to be supported with the same instruments.



(a) Bank Select Multiplexor



(b) Bit Select Crossbar Multiplexor

Figure 3: Multiplexors

The basic Pattern Match Engine is capable of detecting user-specified patterns on each OCP interface. Whenever a specified pattern is detected, the Event signal is asserted. The pattern values, mask values, and state machine configuration are specified at runtime within the programming and analysis application. The Event signals are transferred through another set of multiplexors to the Transaction Engine.

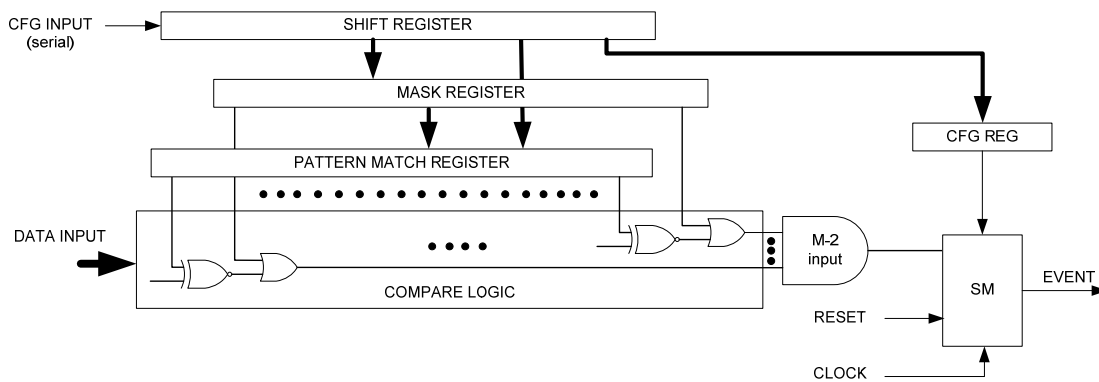


Figure 4: Pattern Match Engine

The Transaction Engine provides a wide range of functions enabled through a rich set of resources that include a programmable state machine, comparators, counters, timers, and adders. The Transaction Engine can be programmed to count events, measure intervals between events, and measure frequency of events. All such actions can be started and stopped conditionally. Conditional actions can be based on event sequences, throughput values, counter values, latency values, or sideband events detected on OCP signals mapped down to the Transaction Engine

through both multiplexor stages. In fact, even signals from other parts of the SoC can be used if they are available on the multiplexor inputs. The Transaction Engine may also be programmed with user-defined embedded memory for optional signal tracing functions. All measurement values and calculated results can be retrieved by the programming and analysis application for display and additional analysis.

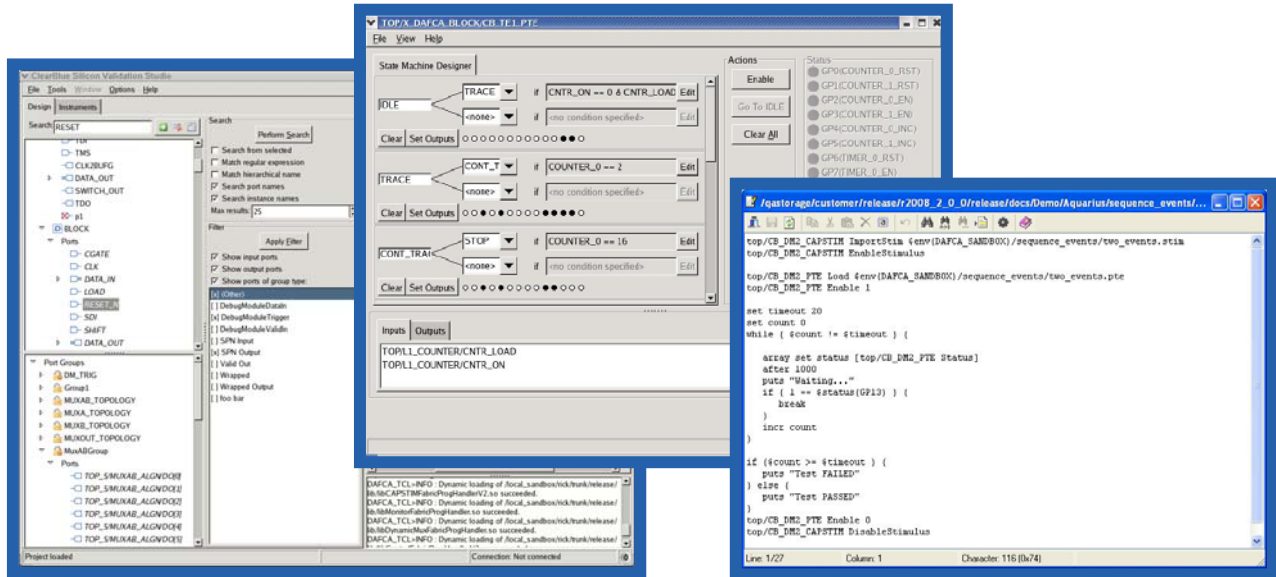


Figure 5: DAFCA's ClearBlue SVStudio (Programming and Analysis Application)

On-Chip and Off-Chip Analysis

Given the obvious bandwidth limitations of the IEEE JTAG 1149.1 interface, large amounts of real-time data cannot be streamed off-chip. While a high-speed trace port such as the Nexus interface (as described in the OCP Debug Specification) http://www.ocpip.org/socket/whitepapers/OCP-IP_Debug_Working_Group_Whitepaper_3_26_2007.pdf can be used, many designs cannot accommodate such an interface. In the baseline configuration shown, we assume a system without a high-speed trace port. Such a configuration highlights a primary advantage of programmable instrumentation: the ability to perform on-chip analysis and reduce (but not eliminate) the amount of serial data transfer. Nevertheless, there are always situations that require off-chip analysis. This is especially true when on-chip data can be transferred into a variety of visualization and analysis tools. For example, the on-chip data can be retrieved and formatted into an OCP Trace File and subsequently fed into tools such as Duolog's OCPTracker. Even if a subset of OCP Trace File fields is populated, transfer sequences as well as performance metrics such as throughput and latency can be analyzed.

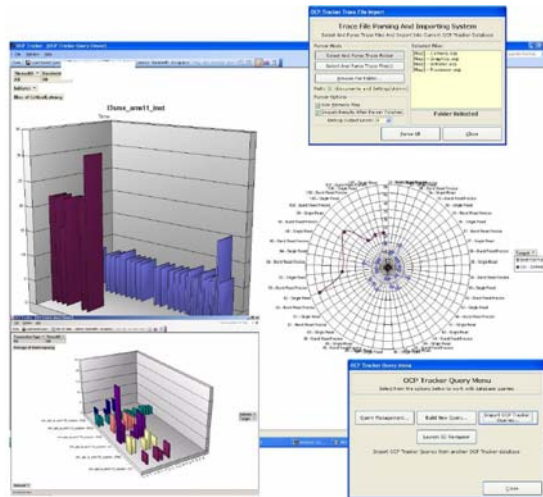


Figure 6: Duolog's OCPTracker

Here again, the programmable nature of the instruments is beneficial. When it comes to extracting on-chip data, there is an obvious tradeoff between temporal and spatial visibility. The amount of trace data to be captured is limited by the width and depth of the embedded trace memory. The user needs the means to make such tradeoffs to maximize the utilization of the embedded memory. For example, if the user wants to see all activity on multiple OCP interfaces, more transfer cycles can be captured if the MData field is omitted. Likewise, the user may choose to reduce the number of transfer cycles captured by creating a capture filter based on a combination of MAddr and MCmd, or filter using signals associated with tag or thread extensions. Each of these techniques is accommodated by a simple expansion or reduction of the observation scope; the designer has ultimate control over the tradeoff between temporal and spatial visibility. The programmable nature of the instruments allows these decisions to be made at runtime.

Alternate Instrument Configurations

While the configuration shown in Figure 2 is suitable in most applications, more advanced configurations are possible. Consider, for example, scenarios that require multiple and specific *types* of transfers to be monitored simultaneously on *each* OCP interface. For such scenarios, two Pattern Match Engines may be required. This can be accomplished a variety of ways. A second Pattern Match Engine can be dedicated to each OCP interface or adjacent Pattern Match Engines can be shared between ports as shown in Figure 7. Through the multiplexor configuration, the user has dynamic runtime control over the use of each pair of Pattern Match Engines.

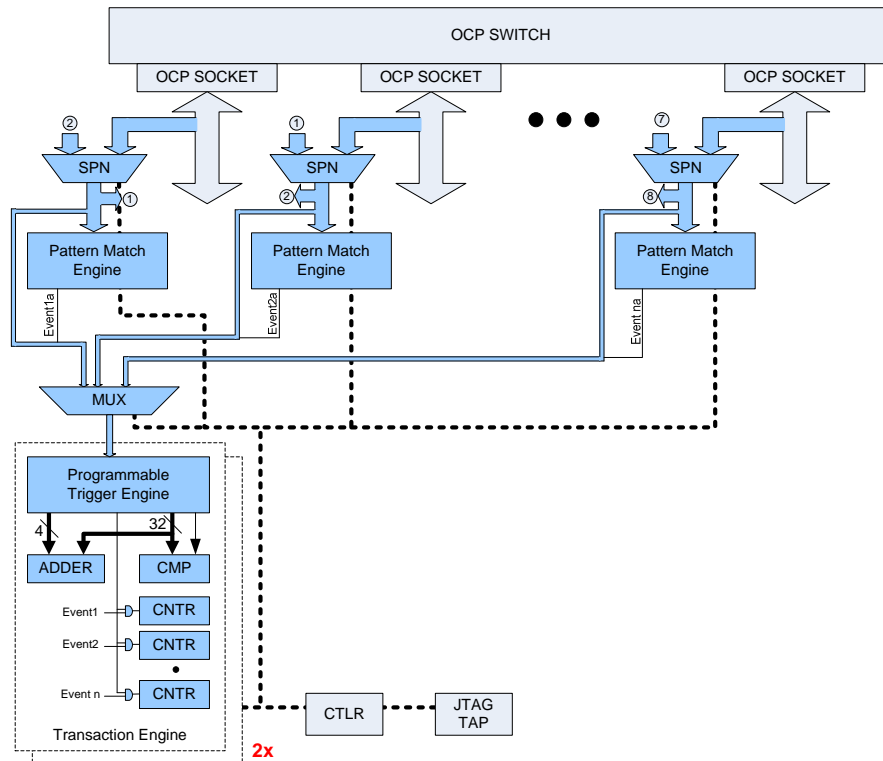


Figure 7: Sharing Pattern Match Engines

Conclusion

Many system validation teams have realized that the process of *proving* the system is working and performing optimally, under all application and environmental conditions, has become too inefficient and difficult for existing methodologies.

The traditional methods for addressing post-silicon requirements have reached a point of diminishing returns. What was once an exercise of designing, implementing, and verifying 25K to 50K gates of instrumentation and writing a few hundred or thousand lines of programming and analysis software has expanded to an effort of designing, implementing, and verifying more than 200K gates of instrumentation, and perhaps ten thousand or more lines of code.

With automated insertion of programmable instruments, the design and implementation burden can be lifted from the hardware design team. With a little upfront planning, a powerful on-chip instrumentation scheme can be realized with far less effort. Moreover, a comprehensive and extendable programming and analysis application is a fundamental part of the solution, enabling test automation, more rigorous test methodologies, and integration with a variety of third-party visualization and analysis tools.

Want to learn more? Check out www.dafca.com.

Paul Bradley is Vice President of Engineering at DAFCA, Inc. Mr. Bradley has over 20 years' experience in electronics and systems design, and specializes in product development and engineering leadership in emerging technology markets. He has held numerous engineering and

technical leadership positions at Motorola, Nortel, CrossComm, Sonoma Systems and Internet Photonics prior to joining DAFCA. He can be reached at paul.bradley@dafca.com