Functional debug: verification and beyond

This workshop will be in lecture format.

Proposed speaker: Hanan Moller

In this workshop, we will explore an alternative approach to SoC development, analysis, debug and bring up. We will describe a different approach, in which debug and performance tuning is considered from the outset, by including within the SoC a light but independent infrastructure dedicated to bringing debug visibility across the entire SoC – an approach which is independent of CPU architecture. We will outline a methodology which includes local intelligence inside the SoC to select and communicate off-chip only those monitoring data which are significant and meaningful. An approach which restores debug functionality such as logic-analyzer visibility to the signals internal to the core, and allows designers to get test data off-chip at high speed using the USB2 or USB3 connectors that are already present on their SoC. We will discuss how such an approach extends the principle of design-verify-optimize to the whole embedded system by enabling a "shift-left": activating system bring-up debug functionality from design start, through emulation to tape-out, post-silicon and on to in-field operation.

Modern SoCs are amongst the most complex devices ever created. What started out as relatively simple electronic structures, designed to provide functionality to single processors, have progressed into multi-core, heterogenous, multiple-interconnect devices with hundreds of millions of transistors. As these devices have grown in complexity, so has the difficulty of designing and verifying them.

The challenges facing multicore SoC design teams are enormous. Architects and verification teams must integrate many types of IP, from commercially available IP from multiple vendors to custom IP from internal design teams, while firmware teams must integrate software for multiple cores from a range of software providers, as well as open-source and internal.

In the SoC design process, simulation and emulation are important tools for functional verification. The use of simulation for early development has several advantages but suffers from one fundamental disadvantage. A full complement of debug features is readily available under software-based simulation because the infrastructure around the model of the processor has full access to its state and internal signals. As the simulation provides access to a virtual target long before the availability of actual hardware, software teams can begin work very early in the project life cycle.

Simulation offers excellent visibility into the operation of the SoC, but unfortunately is orders of magnitude slower than the target hardware. Emulation in hardware is faster, but still falls well short of the speed of actual silicon. The fastest prototyping platforms are generally based on boards carrying multiple field-programmable gate arrays (FPGAs) that are intended to function together as a virtual SoC. But the higher performance comes at a cost. FPGA prototyping systems greatly restrict the number of internal logic signals that can be passed to the outside. The only signals with guaranteed access are those that will be used by the final SoC to communicate with its environment.

The only guaranteed way to assess the real execution performance of a new SoC is to test the chip on a prototype board when it comes back from the fab. This is not just because the design team will be working with real silicon which can run at full speed; it is also because prototyping allows the SoC to load and run a test environment representative of the final system, including firmware, operating system, application code and external stimulus to the SoC. Functional debug is where system testing really begins, and is often a phase that runs way longer than planned and throws up all sorts of unexpected bugs.

But there is a challenge with functional debug; unless provisions have been made within the SoC, internal visibility of what is going on inside the chip is very limited. Existing approaches to this phase of debug tend to resort to the tools that have developed over the years to allow access to SoCs, such as the JTAG standard and processor trace. But JTAG offers only low-speed access, typically tens of megahertz, and processor trace now struggles to keep up with the multiple-gigahertz speed of contemporary cores, and incurs a huge overhead in the amount of data which must be transferred off-chip. And subsequently post-processed and interpreted in an external debugger. The rise of multicore SoC's complicates debug substantially, since the amount of data to be brought off chip rises exponentially.

Functional debug solutions must not only be capable of monitoring any relevant signal inside the SoC, they must be transaction-aware and support heterogeneous architectures, with flexibility to adapt to any partitioning and layout strategies, meet timing at advanced process nodes and provide on-and off-chip bandwidth flexibility as well as statistical and filtering functionality. They must also be non-intrusive so that they don't disturb the system they are measuring, support closed-chassis debug and enable advanced analytics and forensics.

In this workshop, we will further discuss the features of functional debug solutions and the benefits they bring throughout the SoC development process.