The SoC Verification Challenge for ARM-based Platforms

Dave Kelf
Vice President of Marketing and Chief Marketing Officer
Breker Verification Systems

Inadequate verification of Arm-based system-on-chip (SoC) designs is a common complaint throughout the chip design verification community. Standards such as the Universal Verification Methodology (UVM) work well for block-level and subsystem-level designs. They break down at the full chip level making it difficult to stimulate deep behavior from chip inputs or aspects of Arm processor integration such as cache coherency.

Verification is harder when the design is an SoC with one or more embedded Arm processors. Neither UVM nor any of the other popular methodologies can coordinate code running in the embedded processors with events taking place on the SoC I/O ports or in memory.

SoC engineers sometimes perform minimal full chip simulation to check basic IP integration, assuming individual IP blocks are well-verified and ready for assembly into an SoC. This is a risky proposition. Unless IP blocks are tested together in realistic use-case scenarios characteristic of end-user applications, bugs may escape to silicon. Worse yet, unless the design is stressed with a high degree of concurrent activity, it is impossible to assess overall system performance in simulation and unexpected corner-case issues may remain hidden.

Hardware/software co-verification is one solution since it identifies lingering bugs before tape-out. It requires production software running on a fast platform such as an in-circuit emulator (ICE) or a field programmable gate array (FPGA)-based prototype, and even then the execution performance only allows minimal testing. Furthermore, production software often is unavailable until late in the project cycle and designed to perform end-user tasks, not verify hardware. Additionally, it is hard to isolate and diagnose suspected hardware bugs found in system-level co-verification, and bugs found at that point in the cycle are expensive to fix.

Filling the gap between UVM and co-verification means hand-writing C tests to run on embedded processors in simulation and simulation acceleration, time consuming and fragile since changes in the design mean hand-coded changes to the tests. Such tests tend to be "throwaway" because they are neither leveraged on hardware platforms nor reused in simulation on future projects.

Automatic Test Case Generation

One approach gaining widespread adoption can be used for full-chip simulation and verification is Test Suite Synthesis. It automatically generates C test cases to stress an SoC design more thoroughly than conventional methods from a compact, graph-based scenario model that captures the SoC's functionality.

These self-checking tests can support multiprocessor and multithreaded SoC architectures with all C code compiled and loaded into the on-chip memory following the same process as the hand-written tests. They exercise maximum concurrency in the chip, moving realistic application-level scenarios from processor to processor and thread to thread. This stresses cache-coherency logic and looks for choke points in the system resources such as memories as buses. Test cases can allocate memory as well, placing blocks adjacent to each other and reusing blocks as soon as they are freed up, an effective technique at triggering addressing-related bugs.

The test case generator produces C programs to be compiled for each embedded processor and a list of "transactions" for the I/O ports of the SoC because some test cases will read data on or off the chip. However, synchronizing all these different tests without an operating system is complex. A runtime synchronization module services these events. Whenever the code running on the embedded processors needs this module to send data into the chip or read data off the chip and compare to expected results, it sends an event number via a system mailbox. The synchronization module looks up the event in the event file and takes appropriate action for self-verifying test cases that span the SoC's I/O ports as well as the embedded processors.

A system scenario model, the source for the generator, describes the dataflow of the SoC, including how IP blocks are interconnected and what combinations produce end-user scenarios. This model may be written using the new Accellera Portable Stimulus Standard or procedural C++ code with an appropriate class library. The scenario model captures intended design functionality, stimulus, expected

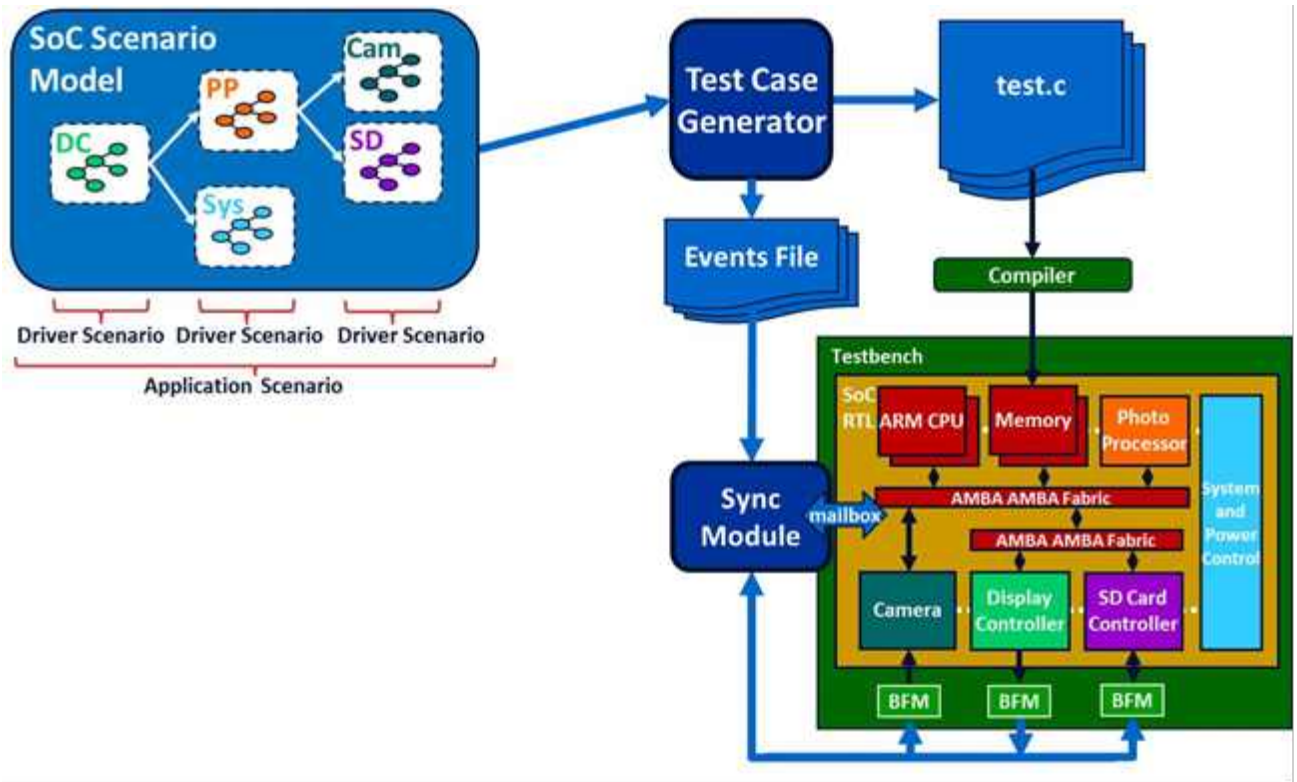

Figure 1 caption: An approach for full-chip simulation and verification uses automatic generation of C test cases from scenario models as shown here.

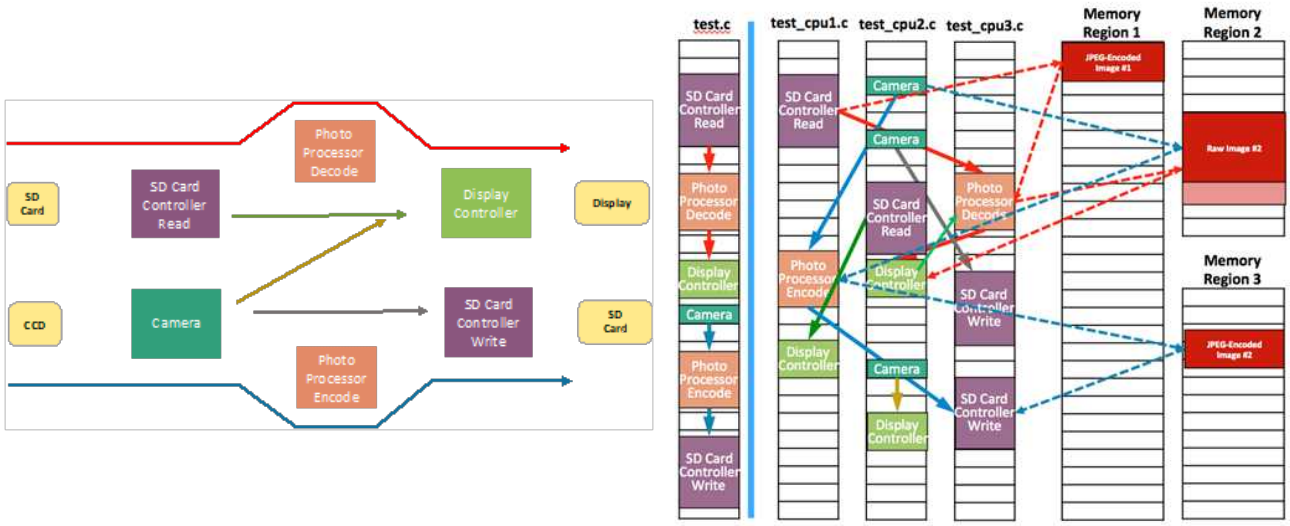

Figure 2 caption: Concurrent tests and resource allocation stress the SoC to find unexpected corner-case issues.

results, debug detail, and system coverage. Because it looks like a hand-drawn dataflow diagram, it is intuitive to create and easily shared across projects.

Test Case Scheduling

Exercising an SoC requires aggressive scheduling of scenarios and memory allocation, straightforward in a chip with a single embedded ARM processor. The generator walks the graph, making random decisions at selection points and randomizing other aspects of the test, including data in accord with the scenario model and user guidance, giving the user control over the process.

A test case, or application, consists of a series of application scenarios, each a realistic example of how an end user would use the device. Scheduling becomes more interesting when multiple processors, multiple threads on a single processor, or multiple threads on multiple processors are available. The test case generator can schedule multiple application scenarios running in parallel, subject to the rule that an IP block cannot be used by two different scenarios at the same time. The application scenarios can be interleaved and moved from one processor or thread to another, as needed.

Moving data around from processor to processor verifies the cache controller, while interleaving application scenarios stresses buses, memory interfaces and other choke points in the SoC. The generated test cases can exercise the chip well enough for the user to capture architectural performance metrics, such as bandwidth and latency.

Test cases are “bare metal” and run on a simple kernel designed for verification purposes, with no operating system or RTOS. This is desirable for simulation performance and necessary for test cases to have a fine degree of thread control to schedule the sort of test case. The same is true of memory allocation. The generator can allocate memory more aggressively than would an operating system, in an effort to uncover addressing bugs.

Conclusions

System-level coverage models can be automatically extracted from a scenario model, offering a wide range of possibilities for coverage metrics at a higher level than traditional code and functional coverage. Because this coverage can be directly correlated to driver and application scenarios, it can extend test generation and coverage beyond simulation into hardware. The same scenario models can generate test cases to run on in-circuit emulators, FGPA prototypes and even the actual silicon to assess the total system coverage achieved across these platforms.

Test cases scheduled across multiple threads, processors and memories improve SoC quality and increase chances for end-product success. Chip design verification groups across the globe use automatic test cases in production with the Breker Verification Systems TrekSoC Test Suite Synthesis solutions.

About Author

Dave Kelf is vice president and chief marketing officer at Breker Verification Systems responsible for all aspects of Breker’s marketing activities, strategic programs and channel management. He most recently served as vice president of worldwide marketing solutions at formal verification provider OneSpin Solutions. Earlier, Kelf was president and CEO of Sigmatix, Inc. He worked in sales and marketing at Cadence Design Systems and was responsible for the Verilog and VHDL verification product lines. As vice president of marketing at Co-Design Automation and then Synopsys, Kelf oversaw the successful introduction and growth of the SystemVerilog language, before running marketing for Novas Software, noted for the Verdi product line, which became Springsoft and is now part of Synopsys. Kelf holds a Bachelor of Science degree in Electronic Computer Systems from the University of Salford and a Master of Science degree in Microelectronics from Brunel University, both in the U.K., and an MBA from Boston University.