BREKER™

TrekSoC WHITE PAPER

Breker Integration with Synopsys Verification

Introduction

It is widely acknowledged that verification is the most resource-intensive portion of a deep submicron chip development project. The Universal Verification Methodology (UVM) standard was a significant leap forward in a common testbench architecture and project-to-project reuse. For all its benefits, the UVM does not encompass system-on-chip (SoC) designs with code running on embedded processors. Further, the UVM addresses only simulation and provides no test solution for hardware platforms.

Graph-based scenario models support portable test cases, which include portable stimulus, checks, and coverage. These test cases can be automatically generated, with as much or as little guidance as the user desires. They are portable from IP block to subsystem to full SoC and from virtual platforms to simulation, acceleration, in-circuit emulation (ICE), FPGA prototypes, and even actual silicon in the lab. These test cases also integrate seamlessly into existing UVM testbenches as well as Synopsys-based verification environments.

Graph-Based Scenario Models

Figure 1 shows a typical SoC design, in this case a chip for a digital camera, which must be verified. It contains two CPUs and two memories, sharing a common bus or fabric interconnect. There are four IP blocks also connected to the fabric. Three are I/O channels for the camera, the display, and the SD card interface. The fourth, the photo processor, encodes and decodes JPEG images. There is also a block related to system-level operation and power management.

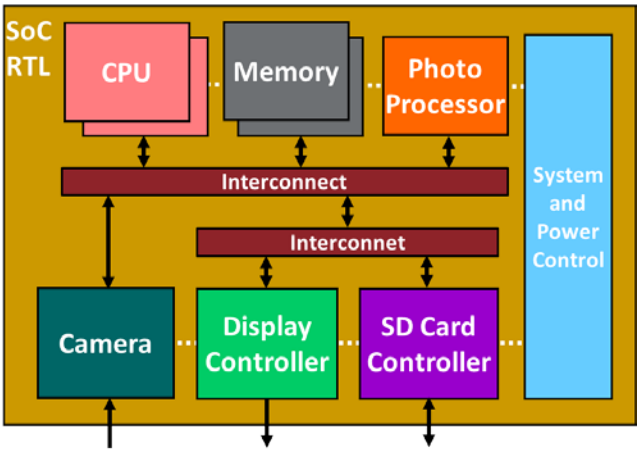

Figure 1: SoC for a Digital Camera

The key to enhancing the UVM in simulation testbenches and expanding to other verification platforms is a better representation of the verification intent for this SoC. The only proven format to achieve this goal is the graph-based scenario model. This model shows how the IP blocks that compose the chip are connected, defines the system-level use cases that must be verified when the blocks are working together, and uses constraints to prevent scenarios that are prohibited by the specification.

Graph-based scenario models come to the rescue since they enable the automatic generation of

test cases that run on the embedded processors while coordinating activity in the testbench. The graph "begins with the end in mind" by showing the possible outcomes on the left and the inputs on the right. Given the appropriate scenario model, a generator can "walk the graph" from left to right, following realistic use-case scenarios, to generate both C code for the processors and UVM transactions for the testbench.

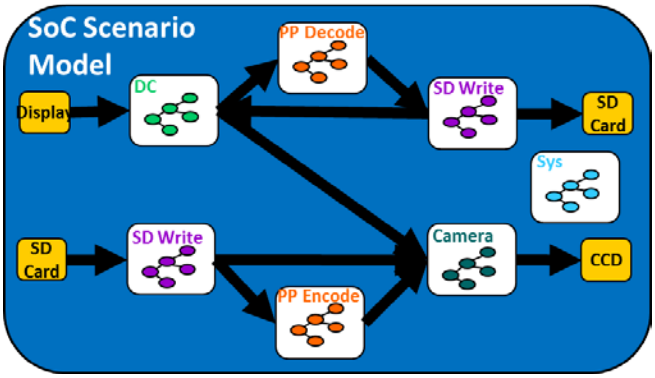

Figure 2: Scenario Model for a Camera SoC

Figure 2 shows a top-level scenario model for the camera SoC. It begins on the left with the two possible outcomes, displaying or saving an image, and flows to the right for the two possible inputs, a new or saved image. As the generator walks the graph, the test cases would cover all user scenarios, including:

- Reading a raw image from the camera, displaying it, and writing the bitmap to the SD card
- Reading a raw image from the camera, displaying it, encoding it in the photo processor, and writing JPEG to the SD card
- Reading a bitmap image from the SD card and displaying it
- Reading a JPEG image from the SD card, decoding it in the photo processor, and displaying it

The multi-threaded, multiprocessor, self-verifying test cases generated by this process are far more complex than could ever be generated by a UVM testbench alone or written by hand. They are portable to any platform or model in which the embedded processor can run C code. They provide far more coverage than production software, which exercises the SoC sparsely and is not designed to find hardware bugs.

Integration with Simulation

The solution for portable test cases described in the previous section is available today from Breker Verification Systems and has been in use by major chip vendors for several years. Numerous Synopsys customers have been able to take advantage of graph-based scenario models and automatically generated test cases because the Breker TrekSoC product has been integrated seamlessly into the Synopsys verification flow.

Specifically, the TrekSoC product from Breker has been integrated with Synopsys products at six major points. The first is simulation of the generated test cases. Figure 3 shows how the test cases generated from the graph-based scenario model for the camera SoC are downloaded into the chip and run in the Synopsys VCS simulator.

As mentioned earlier, one key advantage of these generated test cases over the UVM is that the processors and testbench remain fully coordinated and synchronized. The C code running in the processors is "in charge" of the test case, as suggested by the emerging industry term "software-driven verification." In addition to the C code, the generator produces a list of "events" that the testbench must handle during the simulation. These include supplying data to chip inputs, accepting data from chip outputs and

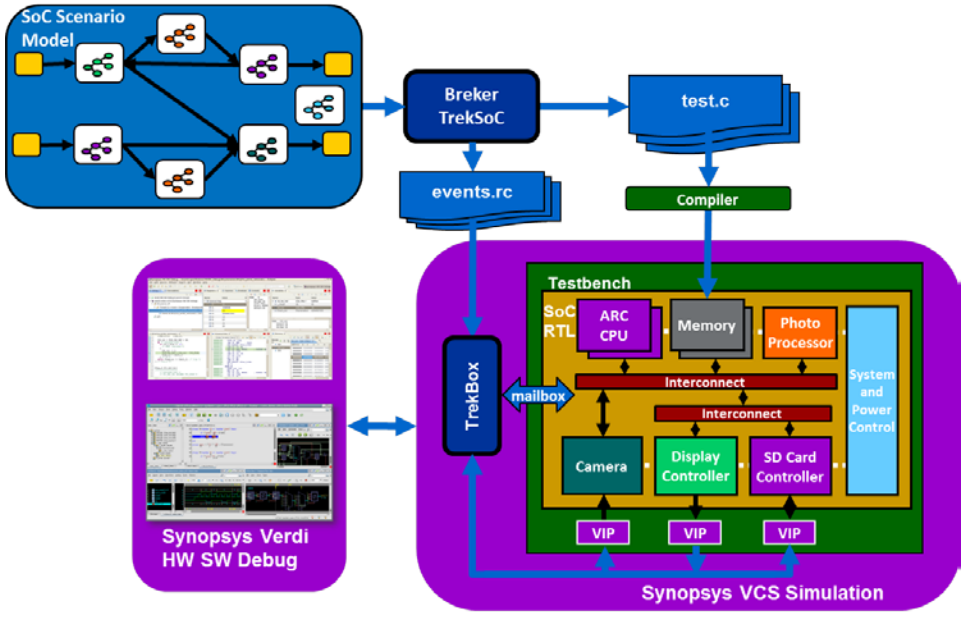

Figure 3: TrekSoC Integration into the Synopsys Verification Suite

checking against expected results, printing status information to a log file, and displaying real-time progress of the test case.

When one of the C programs is ready to trigger the testbench to perform some action, it writes an event ID to a memory-mapped mailbox. The TrekBox runtime module reads the ID, looks it up in the event list, and performs the specified action. This interlock scheme ensures, for example, that data is never provided to the chip until it is ready for it.

The second link to Synopsys products is the reuse of UVM-compliant VC Verification IP (VIP). Synopsys provides VIP titles for a broad range of industry-standard protocols, significantly reducing the amount of testbench code written by the verification team. In Figure 3, Synopsys VIP is shown for all the I/O channels. TrekBox connects directly to these VIP components, subsuming the sequencer functions performed by a traditional UVM testbench. This leverages existing UVM VIP, and also greatly simplifies the scenario models since they don't have to reflect bit-level protocols of I/O interfaces. The UVM virtual sequencer is not needed in simulation with TrekBox, although the verification team may choose to keep some passive testbench components active. These might include scoreboards, protocol checkers, assertions, functional coverage, and even code coverage.

In addition to the VC VIP, the Synopsys DesignWare semiconductor IP library includes cores that implement the design for many industry protocols. In Figure 3, the SD card controller is shown in purple because SD is one of the protocols available from Synopsys. Other blocks might also be provided from DesignWare, such as ARC processors for the CPUs. All of these IP titles are fully compatible with the scenario model and will simulate efficiently along with the user's custom RTL blocks.

Integration with Debug and Coverage

Since the test cases generated by TrekSoC are complex, Breker provide a number of features to help users understand what's happening across multiple threads on multiple processors. Figure 4 shows one very useful display generated and updated by TrekBox as the simulation is running. Arrows trace each use case as its constituent tasks move from thread to thread or processor to processor. In the example shown, the SD card controller reads a previously stored JPEG image and stores it into memory. The photo processor retrieves the image, decodes it, and places the result back into memory. Finally, the display

controller grabs the image and shows it on the camera screen. If the SoC supports concurrent or pipelined operations, then use cases will be interleaved in the test case and in the display.

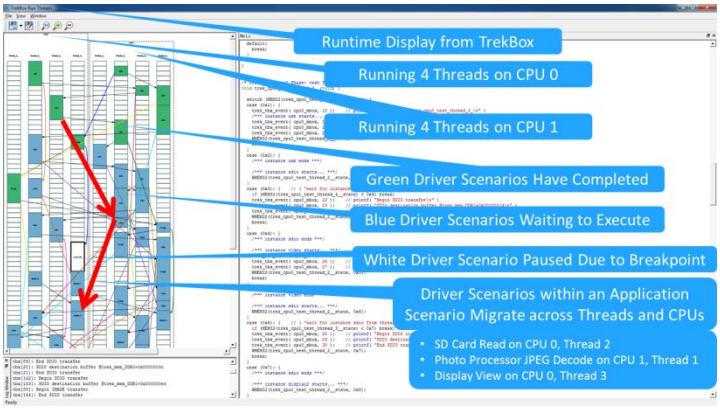

Figure 4: TrekBox Thread Display

TrekBox displays the generated C code, but Breker does not attempt to provide a complete integrated development environment (IDE). Instead, the fourth point of integration is with the industry-leading Verdi HW SW Debug solution. This integration is critical; when a failure occurs in the automatically generated test cases, the verification engineers want to debug it using the same tools and methodology they use for their own production software and hand-written C tests. All the usual capabilities of Verdi Debug for UVM testbench simulation are available, including coordinated waveform viewing and SystemVerilog source code browsing.

With Verdi HW SW Debug, these capabilities are expanded to include viewing and analysis of the C code and the compiled assembly code running on the processors. Thus, the power of RTL debug and an embedded IDE are available together. These capabilities are integrated directly with the Breker GUI, as shown in Figure 5. If the user moves the simulation timeline in one environment, it changes in the other as well. This ability is enabled by the Synopsys VC Apps Exchange partnership program.

Figure 5 shows another aspect of debugging the TrekSoC-generated test cases. When the tests are scheduled, the thread display shown in Figure 4 is generated and update by TrekBox as the test case runs. Since bus congestion and other factors make the exact runtime of each operation in each test case unpredictable, this display is an estimated flow. As the test case runs, the C programs send events to TrekBox as each task starts and ends. At the end of the run, the thread display can be updated to show the actual time consumed by each task. Figure 6 shows the pre-simulation (as in Figure 4) and post-simulation (as in Figure 5) views together.

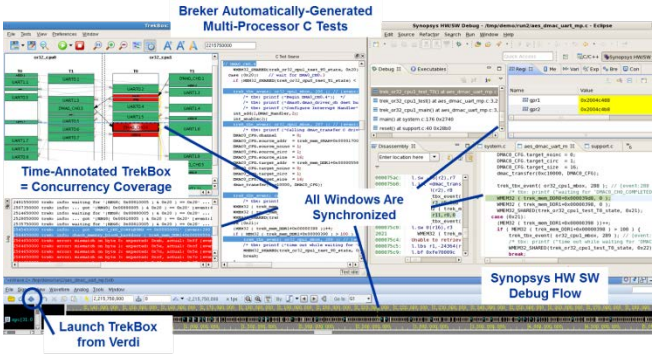

Figure 5: Integration of Verdi and TrekBox

From the timing information, TrekBox can provide a form of concurrency coverage, for example, whether the SD card controller was ever writing a JPEG image at the same time that the photo processor was encoding a new image. In fact, the Breker tools report multiple forms of coverage, all of it at the system level and all of it related to realistic user scenarios and use cases. For example, the user can find out which nodes in the graph have been exercised as well as which paths have been traversed in the process of generating the test cases. Further, system-level coverage can be closed automatically.

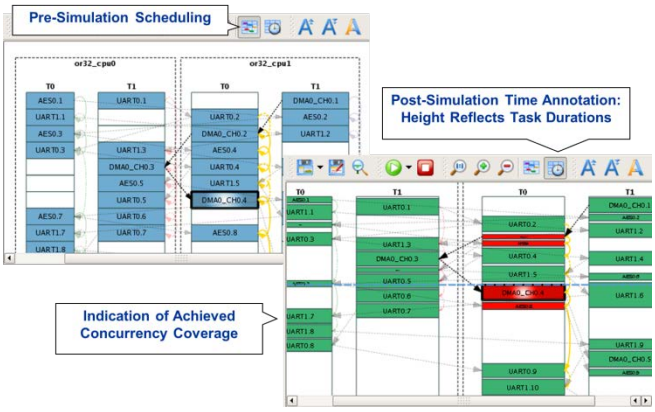

Figure 6: Evolution to Timing-Annotated View

The fifth aspect of integrating the Breker solution into the Synopsys verification environment is the ability to roll up the system-level coverage with all other available coverage metrics. TrekBox can export the system coverage information in the form of SystemVerilog cover groups that can be imported into Verdi.

Verdi combines and weights code coverage metrics, functional coverage metrics, assertion metrics, formal metrics, and more into a single comprehensive view of coverage. It is only natural for the verification team to want to include the graph-driven system coverage metrics in this view. Figure 7 includes screen shots of the Breker coverage GUI and the Verdi coverage viewer, showing how system-level coverage can be rolled up.

This coverage link also enables the final phase of integration with Synopsys products. The Synopsys Verification Planner allows the verification team to capture coverage goals in the verification plan early in the SoC project, linked directly with the design specification. Features are identified and coverage objects assigned to them, enabling links to the implementation of the coverage in the testbench or design to be back-annotated to the verification plan. Since the graph-generated system coverage is imported as SystemVerilog coverage groups, it fits seamlessly into the flow. Features and desired coverage can be flagged for implementation in the scenario model, and then linked back to the plan via the import capability.

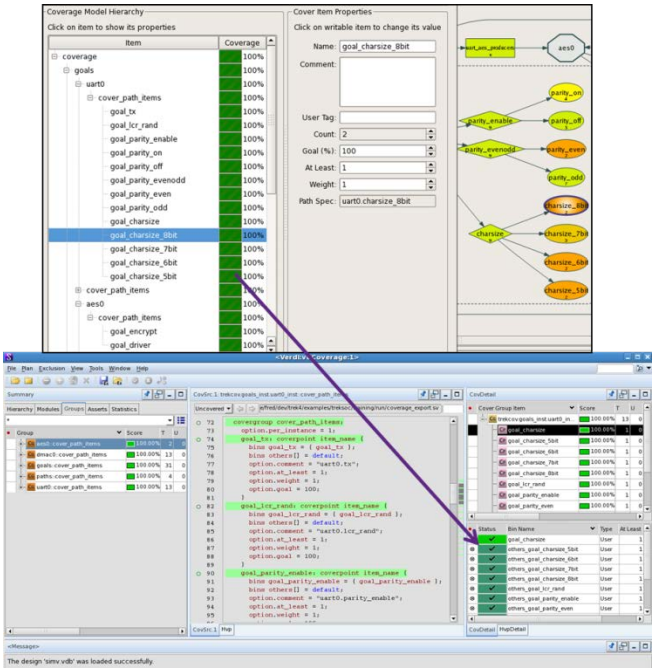

Figure 7: Rollup with System Coverage

Conclusion

The UVM was a major advance in the verification provides the ability to of large, complex chips, but more is needed for SoC verification. Graph-based scenario models and automatically generated test cases from Breker fill the gap by stressing the SoC design more thoroughly and closing coverage more quickly. Breker's TrekSoC product is entirely complementary with the Synopsys verification flow and is fully integrated into the Synopsys verification solution. The combination is a must-have for anyone on the leading edge of SoC development.