BREKER™

TrekUVM WHITE PAPER

Why Graph-Based Verification?
Don't Push on a Rope!

Introduction

The semiconductor industry is starting to use graph-based verification as a critical addition to the constrained-random Universal Verification Methodology (UVM) standard. As chips get ever larger and more sequential in nature, exercising deep state requires the stimulus to propagate through a series of stages that might have very specific rules for what can pass. The verification engineers can use biasing to try to "steer" the stimulus along a particular path, but fundamentally the UVM sequencers are "pushing on a rope" and trying to maneuver that rope around corners and through tight spaces. Having control only at the pushing end is highly likely to result in inadequate verification.

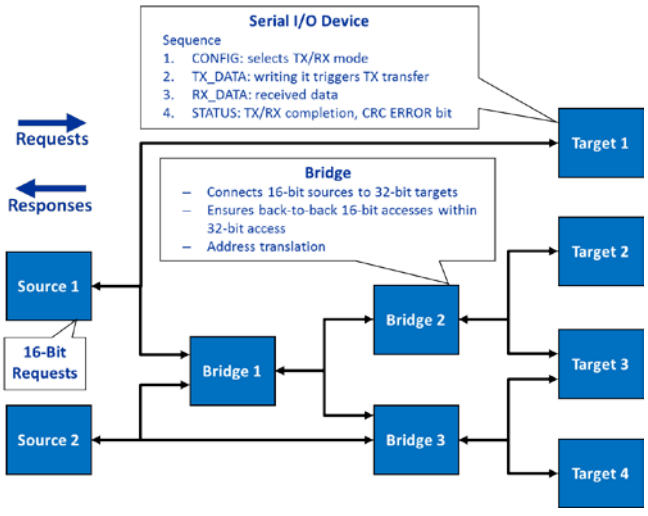

Figure 1: Sequentially Deep Networking Chip

Example Design

Figure 1 shows a representative example of a networking chip design with deep sequential state. This design has two source blocks, each of which accepts 16-bit requests from external agents. There are four possible targets for each request. The first is a serial I/O device that conforms to the following signaling sequence:

1. CONFIG: selects TX/RX mode
2. TX_DATA: writing it triggers TX transfer
3. RX_DATA: received data
4. STATUS: TX/RX completion, CRC ERROR bit

Each target is close to the outputs of the chip, yet it requires a very specific series of events in order to send or receive data. Achieving this purely from constrained-random stimulus on the source inputs is challenging. However, the rest of the chip is even more complicated. There are three bridges that enable asynchronous data of different speeds at the sources and targets. The second bridge performs several additional functions:

- Connects 16-bit sources to 32-bit targets
- Ensures back-to-back 16-bit accesses within each 32-bit access
- Performs address translation when sources and targets have different address spaces

When the second bridge is involved in an operation, the two 16-bit data contents from the sources must arrive within a specified period of time and be properly aligned for the 32-bit target. In addition, there are multiple configuration registers that must be loaded with information on how the addresses are to be translated. These registers are also quite deep in the design and unlikely to be accessed by constrained-random stimulus.

UVM Verification Limitations

Clearly the burden is on the verification engineers to set up the input sequences with appropriate constraints to reach all parts of the design and to exercise all the operational cases. Even if there is enough time and expertise to achieve full functional coverage, which is highly unlikely, the verification team gets limited insight into the real extent of verification. Code coverage and user-specified functional coverage should be supplemented by system-level user-case coverage. This form of coverage shows which of the possible paths through the design, as shown in Figure 1, have been exercised.

As shown in Figure 2, the decoupling of UVM constrained-random stimulus from specific functionality may have unexpected consequences. An input sequence intended to hit a specific coverage monitor may not do so, resulting in an iterative process by the verification engineers to try to achieve coverage closure. Even worse, a specific coverage monitor may be hit by an input sequence that was not intended to hit it and is not triggering checking of the results in that part of the design. Coverage without results checking is meaningless and should not be counted.

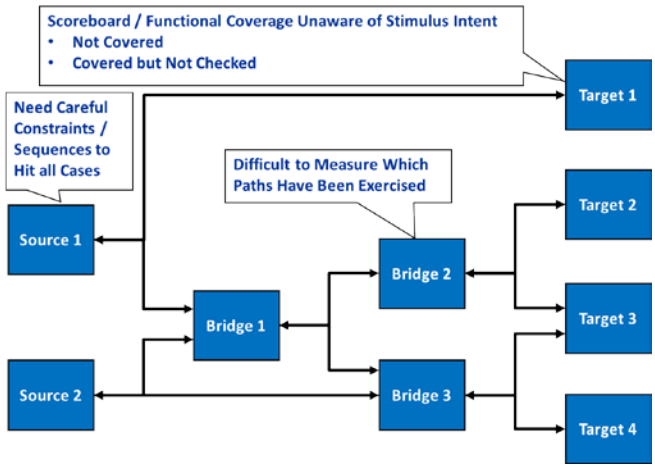

Figure 2: Verification Challenges

Graph-Based Scenario Models

Fortunately, there is a supplement to constrained-random stimulus that addresses every one of the inadequacies and limitations mentioned so far. The key is transforming the data-flow block diagram of the chip design into a graph-based scenario model. The diagram is flipped so that the results, or outcomes, are on the left and the inputs are on the right. This arrangement allows an automatic test case generator to works backwards through the design to determine which inputs are needed to yield each possible result.

As shown in Figure 3, a graph-based scenario model truly "begins with the end in mind." As the generator moves through the graph, it explores every possible outcome and every possible alternative path to achieve that outcome. The term "delegate" is used to describe the process whereby the inputs to a block are set up by using the previous block(s) in the data flow by working backward. This process continues all the way from the chip outputs to the chip inputs.

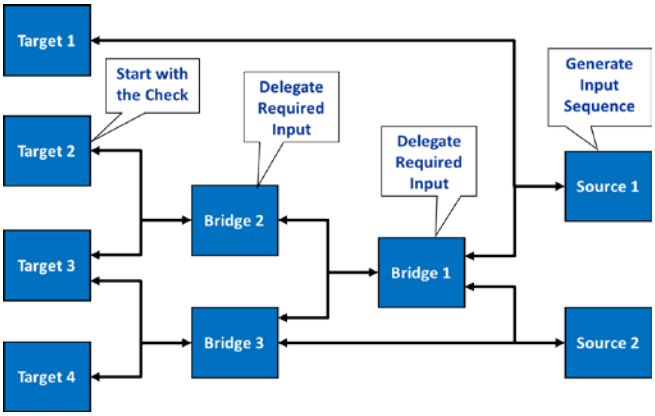

Figure 3: Tracing the Scenario Model

This involves randomization at decision points, such as which bridge provides the input to the third target. Address and data values are also randomized as they would be for constrained-random tests. As shown in Figure 4, each walk backward through the graph-based scenario model produces a different test case.

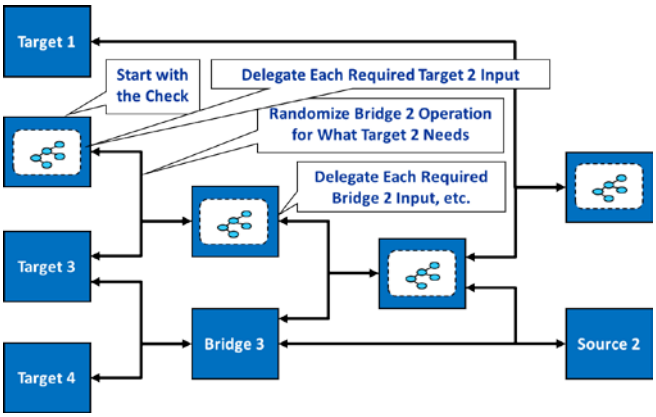

Figure 4: Graph Path Randomization

The result is a much more thorough test suite, generated completely automatically.

Additional Benefits of Graphs

For a very large chip, the list of every possible path might be too large to ever simulate, so verification engineers have a lot of control over the test case generation process. They can bias for certain paths, and constrain out blocks in the design that are not yet ready for verification. The result is that the generator and the verification team are pulling on a rope rather than pushing it, ensuring that all important parts of the design are not only exercised by appropriate stimulus, but results checked as well.

Graphs are composable, so each block in the chip-level scenario model may be a hierarchical scenario model or sub-graph. Thus, the same methodology may be used for IP blocks, subsystems, and the full chip.

Traditional coverage metrics are supplemented by true system-level coverage showing which paths and blocks in the graph have been exercised and which have not. Test case generation from scenario models is a closed-loop coverage system. All that the verification engineers have to do is point at any uncovered block or path and the generator automatically provides a test case guaranteed to fill the gap.

Graphs also support reachability analysis, so the verification team knows which blocks and paths are unreachable and don't waste time trying to hit them. UVM constrained-random stimulus can neither guarantee coverage closure nor determine unreachable coverage.

Summary

Constrained-random stimulus was a big step for functional verification and it continues to work for the block and subsystem levels. At full-chip complexity, constrained-random is insufficient. With the power of graph-based scenario models and automatic test case generation, verification teams can achieve efficient, thorough verification with rapid, guaranteed coverage closure.