One of SystemVerilog's noticeable features is that it is basically a "design language" that has been extended with verification capabilities. This might be an advantage or not, depending on who you're asking, but obviously, if you only want to use a limited number of verification capabilities to gradually extend your existing Verilog testbench, SystemVerilog might be a good solution. This is a bottom-up approach where you want to retain your old verilog code (typically transactors) and add stuff like randomization and maybe coverage on top. Think Verification recommends, though, that you seriously consider a top-down approach as well (read: lose your old testbench), because a top-down approach provides a comprehensive solution to the verification problem. But today we're going to talk about the former approach which is a more pragmatic one in certain cases and we have a special guest for that.

Swapnil Sapre is a guest blogger here on Think Verification. He is a Module Lead at MindTree and has 7+ years of experience in verification. He has been providing consultation to various organizations through his expertise in System Verilog, Verilog, and VHDL. He also has many other areas of interest such as astrophysics, advertising, and politics. Here's Swapnil's interesting paper where he shows how SystemVerilog can be adopted to enhance legacy test benches with constrained random capabilities to verify an ARM SoC system.

Here's the abstract. The full paper can be downloaded here .

ABSTRACT

This document discusses Random constraint-based verification and explains how random verification can complement the directed verification for the generic designs. In our case this is demonstrated by an “ARM processor based platform”. The Constrained Random Techniques (CRT) can be effectively used for verifying large complex designs. As CRT can automatically generate a large number of test cases, it can hit corner cases faster and help in reaching

conditions that would normally not be easily reached with traditional methods. These features are built over and above an already existing legacy Verilog environment. Random verification for generic designs is implemented by Transaction based Models or Bus Functional Models. The language used for the Verification environment is SystemVerilog.

If you want to submit an article / paper please email us at mail@thinkverification.com