

Integrating Hierarchical Test Benches into an Evolving VHDL Design Environment

Michael F. Sullivan, James O. Bondi, David J. Kopca, and Nayan D. Patel

Microelectronics Department
Texas Instruments, Inc.
Defense Systems and Electronics Group
PO Box 660246, MS 3145
Dallas, Texas 75266
214-480-1345 office
e-mail: sullivan@lobby.ti.com

Abstract

Flexible, hierarchical test benches are developed naturally as part of the normal model development process and support VHDL, WAVES, and company-proprietary standards. The integration of these tool-automated VHDL test benches is described.

1.0 Introduction

In the Texas Instruments (TI) Microelectronics group, the test environment was for some time based on TI-proprietary tools, languages, and systems. This test environment is now evolving into a fully VHDL-compatible design environment with integrated test bench support that comprehends VHDL standards but can still interface with the significant legacy of older TI-proprietary designs. Herein, we introduce the current environment, emphasize ongoing work in implementing automatic test bench generation for both the TI test pattern format and the Waveform And Vector Exchange Specification (WAVES) format, discuss problems overcome, and summarize future plans.

In TI-Microelectronics, the historical test environment has been based upon TI simulation scripts and TI Test Description Language (TDL) vectors. These scripts and test vectors, required for validating models in development, have typically come from non-VHDL, TI-proprietary tools. A major goal for the evolving Microelectronics VHDL design environment has been to automate production of requisite test benches in a way that integrates their development with the natural model development process and integrates their functional capabilities with the historical TI design environment.

To integrate test bench development into the natural VHDL modeling process, a novel hierarchical test bench structuring technique is employed. In this technique, a test bench shell is "wrapped around" each individual

block within the model. Each such shell is functionally uniform throughout the model, containing (1) a flexible port that can make it "transparent" and permit the encapsulated block to have full and normal interaction with surrounding elements, and (2) simple internal means for capture and even later a replay of test results as seen at the periphery of the encapsulated block.

A complete and fully functional model is built by instantiating each shell-encapsulated block wherever the bare block would normally be instantiated. With an integrated hierarchy of such shells used throughout a complex model, test results can be readily captured at any or all component blocks even though specially crafted test stimuli need be applied only to the complete model by the outermost test bench shell. The hierarchical test bench that results is, in fact, interwoven into and throughout the executable model.

To integrate test bench capabilities with the historical TI design environment, each and every encapsulating shell described above comprehends internal placements that can capture test results in TI-proprietary TDL format or replay prior test results from this TI format. Of course, placements to provide such capture and replay capabilities for WAVES format are also available, as are placements for generating various simulation reports. All of these test bench shells at all levels can be automatically generated with tools we describe herein.

2.0 Integrated Model/Test Bench Development

A VHDL model of a modern IC typically contains a hierarchy of blocks, some placed inside others, as illustrated in Figure 1. To test such a model as a whole, it is common practice to encapsulate the complete hierarchical model in a special test bench shell as depicted in Figure 2. Within this outer shell, special testing placements can provide specially crafted stimuli to the ordinary hierarchical model as it operates and can

examine results therefrom. Although such test benches can be effective, their development is usually tedious and requires significant effort. Thus, their custom development and use around each interior block as a hierarchical model development progresses can be a truly daunting task.

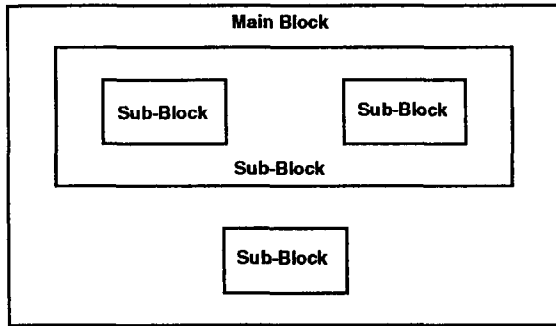


Figure 1. Ordinary Hierarchical Model

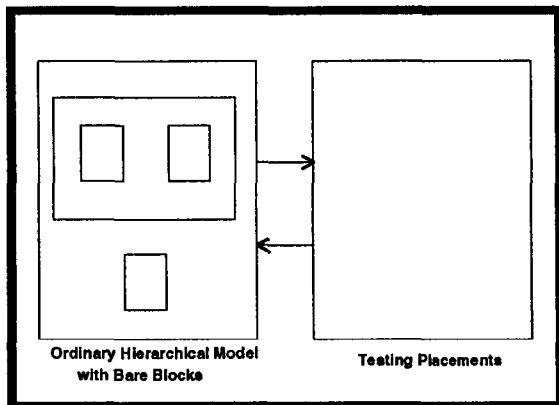


Figure 2. Traditional Test Bench Shell Surrounding Hierarchical Model

Within Microelectronics, a different test bench development methodology is now being utilized on many IC designs to substantially reduce required test bench development effort. Figure 3 shows that, in this methodology, each block of the VHDL model, not just the outer level of the overall model, is encapsulated in its own local test bench shell. Development of the shell used at each block is a rather mechanical process that is largely automated. As implied in Figure 4, shell-generating utilities can readily install any of a variety of VHDL testing placements within each shell. These placements can endow the shell with many capabilities including those to

- capture and record observed block input/output behavior as sequences of test vectors

- verifiably recreate such behavior by reapplying vector-specified inputs and checking for vector-specified outputs.

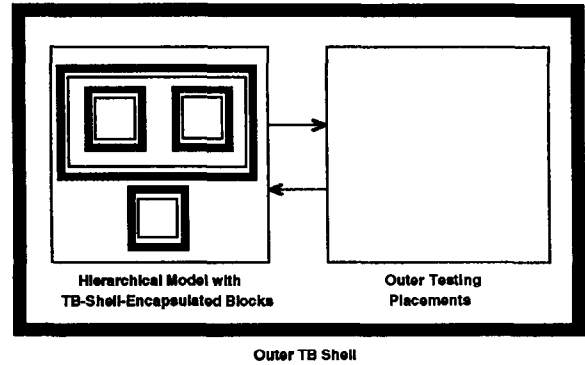


Figure 3. Hierarchical Test Bench Integrated Throughout Hierarchical Model

The port shown in Figure 4 especially differentiates this shell from a traditional test bench shell that is used at only the outer level of a model. Because of the port, this shell allows its encapsulated (bare) block to interact either with the internal test placements OR with other blocks of a larger model. This shell can thus be instantiated as part of a larger, more complex model as indicated in Figure 3. When instantiated in this manner, the shells collectively become a hierarchical test bench that is interwoven into and distributed throughout the executable model. The development of this hierarchical test bench is integrated with the natural VHDL model development process and the structure of this test bench is integrated with the overall structure of the normal executable VHDL model.

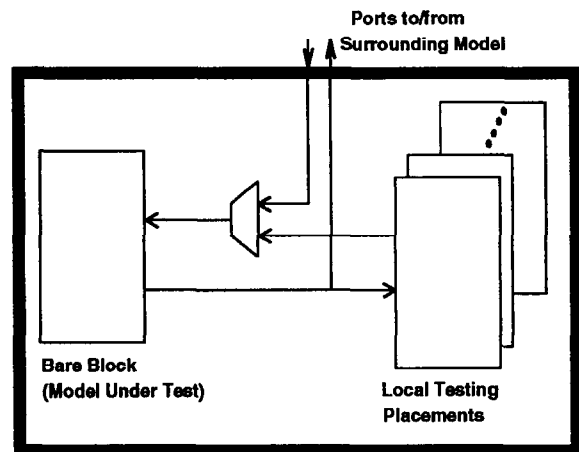


Figure 4. Individual Model Block with Encapsulating ("Transparent") Test Bench Shell

"Switches" within each shell, controlled according to user discretion during simulation runs, allow the shells to cooperate in a variety of manners. A common switch configuration would be one that permits the outer shell to apply specific stimuli to the whole model while the interior shells all capture and record test vectors at their individual block boundaries.

3.0 Integrated Test Bench Tools and Features

Normally, a test bench shell such as those surrounding the blocks in Figure 3 can have one of two styles. The first is a high level model made up of functions and procedures. The other is an I/O based model that makes use of test vector files.

Figure 5 shows the hierarchical structure of the test bench shells generated by Microelectronics-developed tools. The parent test bench shell created by the *Tbgen* tool has placements of child test utilities. Possible child test placements are Driver, ReadTDL, WriteTDL, ReadWAVES, and WriteWAVES.

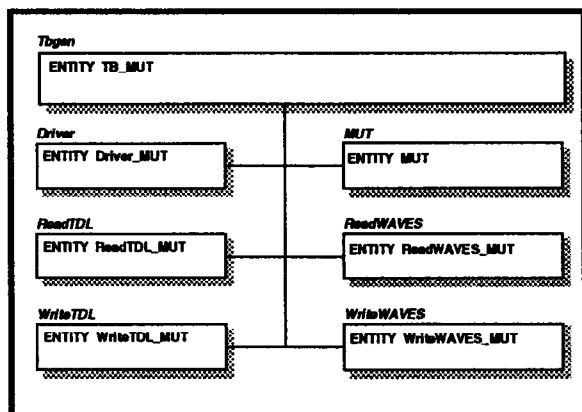


Figure 5. Structure of Parent Shell and Child Test Placements

The Driver invokes a number of functions and procedures that apply stimulus to the Model Under Test (MUT) and tests its functionality. ReadTDL reads, applies, and checks TI TDL data. WriteTDL captures TI TDL data. ReadWAVES reads, applies, and checks VHDL WAVES data. WriteWAVES captures VHDL WAVES data. The Driver test placement is created manually by the designer. The remaining test placements are automatically generated by *Tbgen*-related tools using the entity statement in the VHDL MUT and sample test vectors as guides.

4.0 Conclusion

In TI-Microelectronics, automated development of VHDL test benches and integration of test bench development into the VHDL modeling process are being employed successfully. The novel hierarchical structuring technique described integrates automated VHDL test bench capabilities with the historical TI proprietary design environment.

5.0 References

1. *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076-1987, The Institute of Electrical Engineers, Inc., New York, NY, March, 1988.
2. *Draft WAVES Standard*, Prepared for the Test Analysis and Standardization Group of the Design Automation Standards Subcommittee and the Standards Coordinating Committee Number 20 of the IEEE, December 12, 1991.
3. M. Sullivan and M. Vincze, *Integrating WAVES into an Existing ASIC Environment*, Managing System Design and Development with VHDL, Proceedings of the VIUF Conference, Fall, 1992, pp. 258-266.
4. M. Vincze, M. Sullivan, and D. Kopca, *Merging WAVES into an Existing VHDL Environment*, Texas Instruments Technical Journal, May-June, 1993, pp. 57-63.