

VHDL - Verilog Co-simulation Environment

Gunjeet Baweja & Zia Khan
PCI Components Division
Intel Corporation
1900 Prairie City Road, Folsom, CA 95630

1.0 Abstract

ASIC designs developed in VHDL need to be simulated at netlist level for functional and timing verification prior to sign off. Most ASIC vendors only provide macro cell libraries in Verilog-HDL. To make use of these libraries the VHDL test setup must be ported to Verilog environment. This problem can be solved by simulating the Verilog netlist in VHDL simulation environment. This paper discusses the use of VHDL-Verilog co-simulation environment.

2.0 Introduction

The power and flexibility of VHDL is most useful for modeling VLSI systems at behavioral and dataflow levels. However, using VHDL at netlist level faces many hurdles. For instance, most ASIC vendors do not provide sign-off quality macro cell libraries in VHDL that may be used for simulating the netlist. Also, there is little support for back annotation of parasitics for post-layout timing validation.

ASIC designs in Intel use VHDL for design specification in synthesizable dataflow constructs. The validation environment models the target system where a chip under design will be used. This system environment is described using behavioral VHDL models. The operation of the chip is verified using test benches containing large vector set that exercise various functions of the chip.

The chip netlist is generated by a synthesis tool using an ASIC macro cell libraries provided by the silicon vendor. Following synthesis and layout the functionality and timing of the netlist are verified using both static and dynamic analysis tools. For dynamic simulations we use a simulator that is accepted by the silicon vendor as the golden simulator. For historical and technical reasons, many silicon vendor consider Verilog-HDL simulator using libraries with accurate timings and parasitics to be the golden simulator. To accommodate this requirement, test benches developed in VHDL are translated to a Verilog environment. Translation of VHDL test benches to Verilog format is tedious and time consuming because design specific information must be manually included in Verilog-HDL files.

Current work on VITAL specifications will be useful for building macro cell libraries in VHDL. With availability of VITAL libraries it would be possible to forego Verilog simulations and use VHDL simulations for sign-off. However, it is unclear if simulating large ASICs at this level of detail will be a practical possibility because of simulator performance limitations.

While work continues on enhancing VHDL tools and methodologies to solve these problems there are some attractive alternative solutions that make use of existing capabilities. One such solution is the use of existing sign-off quality Verilog libraries for doing

netlist simulations in a VHDL environment.

In this paper we describe the use of a co-simulation environment where VHDL and Verilog-HDL simulators are used to simulate concurrently parts of a large system in one environment. We discuss how we set up the environment, the problems we encountered and their solutions and point to further work that will be undertaken in future.

3.0 Validation Environment

The PCI Components Division of Intel develops peripheral chipsets for personal computers that enable new technologies and showcase the power and capabilities of Intel microprocessors. The very short development cycles require the use of techniques and methods that provide fast turn-around times. Most components are designed using ASIC technology. Because these components need to work in an industry standard architecture, significant engineering resources are dedicated to verify compliance to existing standards. In this section we present an overview of a PCI chipset introduced by Intel and the system simulation environment used for its validation.

Peripheral Component Interconnect (PCI) is a high performance local bus architecture for personal computers providing interface to high speed and high throughput peripherals such as LAN, SCSI, graphics and video. Intel has introduced chipsets that provide PCI interface to Intel486 and Pentium™ processors. A Pentium™ processor based system implemented with 82430 PCIset is shown in Figure 1 [5]. The 82430 PCIset consists of a Host/PCI bridge with cache/main memory controller (82434 PCMC), a Local Bus Accelerator (82433 LBX) and an I/O subsystem core with interface to ISA bus (82378 SIO).

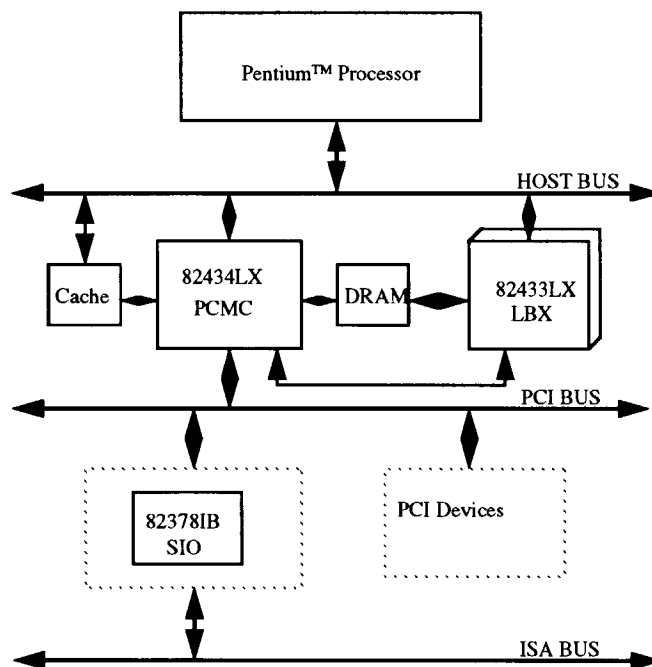


Figure 1. Pentium™ Based High Performance PC

These components are designed by first creating their models in synthesizable dataflow VHDL constructs. These models are then tested to ensure correct functionality. The complex interactions of various busses must be verified using very elaborate test vectors. It is difficult to conceive of all possible scenarios that might happen when these components are connected in a system. Therefore, validation of these components in a stand alone environment is not feasible. A better alternative is to model the entire target system and instantiate all components being designed and verify their interactions in this environment.

A system simulation environment consisting of VHDL models of the components being designed and relevant components of the target system is built to create a validation environment. This environment attempts to model the architecture of a personal computer; the target system for these chips. The Pentium™ processor is modeled as a VHDL bus functional model [1]. Other components such as PCI master, PCI slaves, ISA master, ISA slave, cache and DRAMs are all modeled in behavioral VHDL code. The PCMC, LBX and SIO are represented in dataflow VHDL description. In addition, a number of instrumentation models are also include in this system environment. These models are useful in debug as they collect information of various transactions taking place in the system. The details of these models are described in [3].

Following functional verification of dataflow VHDL models of components under development, these models are synthesized to obtain gate level netlist. The netlist is simulated at the gate level for functional and timing verification [4]. During VHDL simulation, signals are captured at the chip interface. These are converted, using a translation program, to create a Verilog test bench. Although this process works well it does have a number of limitations. The data value of bi-directional signals captured in VHDL environment is not enough to determine if they are acting as inputs or outputs at a particular time. This can be determined only by noting the value of the enable signal driving the bi-directional buffer of the chip. Since every chip has a unique pin assignment the translator must be customized for each component. The VHDL models are developed in synthesizable dataflow style that is devoid of exact timing; only clock or cycle timing are implied in such models. Therefore, test vectors captured in this environment often violate hold time requirements of timing-accurate netlist. These timings are added when VHDL vectors are translated to verilog testbench. The translation process also creates large files resulting in high disk space requirement. Because of these limitations only a small subset of vectors is chosen for gate level simulation.

4.0 Co-simulation Environment

Running a large set of vectors can greatly increase confidence in functionality and timing performance of the netlist and help reduce number of bugs in silicon. The number of vectors can be increased by using the VHDL system simulation environment for generating stimulus and applying it to the netlist that is instantiated in same environment. In this section we describe a co-simulation environment that can be used to achieve this.

A VHDL-Verilog environment has two simulators operating in a synchronized fashion. The portion of system simulation model described in VHDL is executed on the VHDL simulator and the chip netlist in Verilog is executed on a Verilog simulator. A block diagram of this scheme is shown in Figure 2. VHDL and Verilog simulators could be running on different or same machine. Communication between the two simulators is provided by using unix sockets or shared memory. A communication manager links the two simulators via a back plane. This communication manager keeps track of simulation time in both simulators and synchronizes them at regular intervals. The communication manager also oversees information exchange between the two simulators and helps

translate logic values and strengths of signals that cross simulator boundaries.

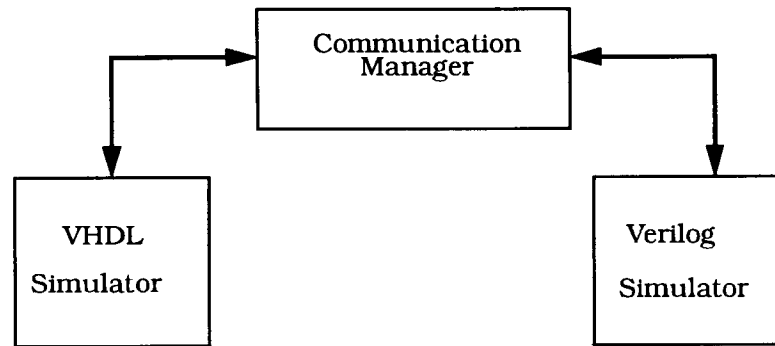


Figure 2. Cosimulation Environment

5.0 VHDL - Verilog Co-simulation Connection

In this project we used a co-simulation tool from Precedence Incorporated that provides an interface between Vantage's VHDL simulator and Cadence's Verilog-XL simulator. The Verilog to VHDL connection is implemented using the SimMatrix simulation backplane. Both Vantage and Cadence simulators have programming language interfaces; Styx interface in Vantage and PLI in Cadence. The interfaces are used to connect the two simulators with SimMatrix tool. Physical communication between VHDL and Verilog simulators is through UNIX shared memory and semaphores [2].

SimMatrix consists of two major components: the partitioner and the interactive controller. The partitioner separates the designs into VHDL and Verilog portions in preparation for mixed-mode simulations. The interactive controller provides event transfer, synchronization and interactive debug capabilities. A top level Verilog model is created that instantiates both VHDL and Verilog sub-models. Each VHDL module is identified by attaching a parameter with it. The partitioner then automatically identifies all nets that cross simulation boundary between Verilog and VHDL simulators and generates one hierarchical netlist containing all identified VHDL components. Special communication devices are transparently inserted into Verilog netlist so that event transfer and synchronization can take place.

Once the design has been partitioned and compiled co-simulation can be run. The process of co-simulation requires time synchronization between simulators, transfer of control between simulators and strength mapping between VHDL and Verilog simulators. These requirements are enforced by SimMatrix.

Time synchronization is achieved by ensuring that both simulators maintain the same notion of time. This is enforced by using a synchronization mechanism during simulation. Each simulator has its own independent time wheel. The SimMatrix communicates with both simulators to ensure that they both remain in synchronization. As each synchronization point is reached, events that occur on the boundary of VHDL models are transferred to/from SimMatrix to other simulators [2].

Simulation control can be provided from either Verilog or VHDL simulator's interactive environments. Control can be passed back and forth between the two environments. This co-simulation environment requires that Verilog simulator be the master simulator i.e., one can instantiate

VHDL models in Verilog but not the other way around; the top level must be Verilog. This is not as limiting as it might seem. One can usually work around this limitation by adding an extra level of hierarchy in Verilog that instantiates the actual VHDL and Verilog models. Vantage's VHDL and Cadence's Verilog simulation and debug environments are fully supported. One can use Vantage tools to debug/monitor VHDL models and the Verilog-XL tools for Verilog models.

Our VHDL models use the nine valued logic system defined by IEEE 1164. But Verilog only supports four valued logic system. Therefore, during co-simulation a mechanism must exist that allows for mapping between VHDL's 9-value logic system to Verilog's 4-value logic system. The mapping scheme used for co-simulation is shown in Tables 1 and 2. SimMatrix implements the bus resolution function at the VHDL-Verilog simulation interface.

Table 1: VHDL to VERILOG

VHDL	VERILOG	STRENGTH
U	X	6-6
X	X	6-6
0	6	6-6
1	1	6-6
Z	Z	0-0
W	X	5-5
L	0	5-5
H	1	5-5
-	X	0-0

Table 2: VERILOG to VHDL

VERILOG	STRENGTH	VHDL
X	0-7	X
Z	0-7	Z
0	0-7	0
1	0-7	1

6.0 Using Co-simulation for Netlist Verification

Co-simulation environment described in previous section was used to validate SIO netlist in VHDL system simulation environment. To prepare for co-simulation the system model had to be modified slightly. A top level Verilog model was created because co-simulation requires that the top level model be in Verilog. This top level Verilog module instantiated a VHDL testbench and SIO netlist. The VHDL testbench consisted of behavioral VHDL models of all system components except PCMC and LBX which were in dataflow style. Since this project was aimed at validating SIO design we only instantiated SIO netlist in Verilog. The new hierarchy is shown in Figure 3.

This simulation used the same test benches that were used for verification of dataflow VHDL models in system simulation environment. These test benches ran over 1 million vectors to fully exercise SIO. This provided more confidence in quality of design and helped ensure that late ECOs did not create any new bugs.

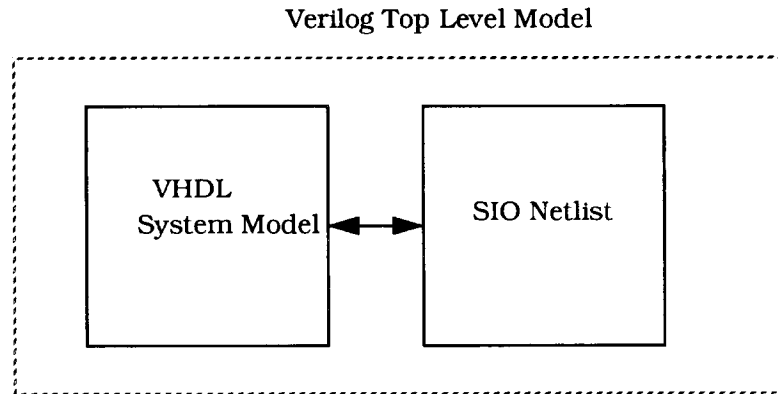


Figure 3: Co-simulation environment for SIO

VHDL and Verilog simulators in co-simulation environment can be run on the same machine and the communication provided through shared memory and semaphores or run on different machines and the communication provided using unix sockets. Both these setups were tried, under normal network traffic to determine the optimum configuration. It was determined that execution times when VHDL and Verilog were run on separate machines was about 3 times more than when the simulation was run on one machine. We believe this is due to traffic bottle neck on the network.

7.0 Future Enhancements

The present system simulation environment uses VHDL models that are clock cycle accurate. Thus these models do not have timing information at the nano-second level. Because of this limitation, the stimulus generated by the VHDL models change values at cycle boundaries that imply zero hold times. As Verilog netlist has full timing information from macro cell library (and back annotation from layout) gate level simulations start to diverge from VHDL simulations and result in loss of synchronization between VHDL system models and Verilog netlist models.

A simple way to fix this synchronization problem is to turn timing information off in verilog netlist and just treat it like a zero delay model. However, this solution may cause the zero delay netlist simulation to go into oscillations due to zero delay loops. Although this approach is fine for validating functionality of the component it does not help in verifying the AC timings.

In future we plan to work on adding timing information to all models in system environment so that VHDL environment can be used to generate (nano-second) accurate vectors to verify AC specifications of the chip. Our approach will be to build a timing shell around existing components in system model that can be selectively turned on or off.

8.0 Conclusions

The VHDL-Verilog co-simulation environment was successfully used to verify functionality of SIO component. It is a very effective tool to make sure that no bugs were introduced due to last minute ECOs. It enabled us to use VHDL system simulation environment for verifying netlist without having to do VHDL to Verilog vector translations. The co-simulation environment greatly

enhances our ability to run large vector sets on gate level models.

9.0 Acknowledgment

The authors would like to thank Srinath Anantraman of Vantage Analysis Systems and Steve Wang of Precedence. The authors also wish to thank their colleagues at Intel, Steve Rowland and Eric Magnusson for their help during this project.

10.0 References

1. S. Hunt, A. Mehta, D. Patterson and S. Shah, "A Behavioral VHDL BUs Functional Model for the Pentium™ Processor", Proceedings of VHDL International Users Conference, Spring 1993, pp 163-173.
2. Precedence Incorporated, "Using Vantage in Verilog Design Environment, users guide", 1993.
3. S. Tayal, A. Moezzi, and E. Magnusson, "System Level Verification of ASIC Chipsets", 6th Annual IEEE International ASIC Conference, 1993, pp 283-287.
4. Zia Khan, "Topdown ASIC Design Using VHDL", Proceedings of VHDL International Users Conference, Spring 1993, pp 243-251.
5. Intel Corporation, "82420/82430 PCIset ISA and EISA Bridges data book", 1993.