

Style Guidelines for Effective Use of Parallel and Multithreaded VHDL Simulators

Dr. John Willis
Senior Engineer / Scientist
IBM System Technology and Architecture Division
Rochester, Minnesota
jwillis@acm.org

Dr. William Paulsen
Senior Engineer
Vantage Analysis Systems Division of Viewlogic
Fremont, California
paulsen@viewlogic.com

Abstract

Parallel VHDL simulators are a technical and commercial reality thanks in part to the intrinsically parallel nature of VHDL. The simulation efficiency achieved when using such simulators is heavily dependent on the modeling style used to write the VHDL model source.

From this paper, readers will gain an understanding of the style guidelines which the author of VHDL source can use to effectively exploit a range of parallel simulators. Our primary focus is on modeling guidelines boosting performance. For practical reasons, the utilization efficiency of processors, memory and network resources is a critical but somewhat secondary concern motivating these guidelines (in the absence of significant performance gains, efficiency is immaterial).

IEEE's DASC (Design Automation Standards Committee) has chartered a group to develop a recommended practices document providing VHDL coding guidelines for effective use of parallel simulators. Beyond its immediate value to VHDL simulation users, this paper is an effort to gain feedback on the approach being taken by this DASC group for vendor-independent parallel simulation style guidelines.

1 Introduction

How effective is parallel VHDL simulation? Depending on who and when you ask, you are likely to get an amazingly wide range of answers [JS93], [HDWP94]. Answers will range from something like, "using N processors, my simulation slowed by a factor of two" through "wonderful, N processors

speedup the simulation by M times". *Why such a wide variety of answers?*

The somewhat puzzling explanation stems from the wide differences in the processing environment (processors, interconnect, system software), VHDL compiler and VHDL models used for parallel simulation. A companion paper in this session "A Taxonomy of Parallel VHDL Simulation Techniques" [GPJW95] addresses the first and second sources. This paper will address the third source of variance, the VHDL model.

Many modeling issues impacting simulation speed are effectively beyond the model author's control. Issues such as model complexity (e.g. 4 bit adder versus parallel supercomputer), model abstraction (e.g. queueing versus gate-level model representing physical delays) and stimulus (e.g. four test inputs versus booting the operating system) all have tremendous impact on parallel (as well as sequential) simulation performance.

Aspects of VHDL modeling which are under the model author's control are the most interesting aspects and thus represent the focus of this paper. The aspects we will deal with in this paper include: time, processes, data types and shared variables. The intent of this paper is to provide VHDL model authors with practical suggestions for improving the performance of models under parallel simulation.

This paper takes a broad view of parallel simulation: generally any simulator with more than one concurrently executing data stream and the ability for communications between the data streams. See our companion paper [GPJW95] for discussion of the many forms a parallel VHDL simulator can take.

2 DASC Parallel Simulation Group

The IEEE Design Automation Standards Committee chartered a Parallel Simulation Group, chaired by John Willis, to develop a recommended practices document for efficient, high performance parallel VHDL simulation modeling. This paper is a first step toward this group's recommended practices document.

After considerable discussion of how to factor out the processing environment and compiler, the group decided to key recommended practices to a taxonomy of processing environment and compiler approaches [GPJW95].

The methodology for preparing keyed recommended practices begins, where possible, with anecdotal recommendations from a particular simulator implementation group. The IEEE group refines this anecdotal recommendation into one or more test cases in order to determine the applicability of a particular recommendation in the context of specific simulator implementations. This approach closely follows the AFIT VHDL "stress test" [KS89]. These tests are intended as discriminates, *not* any form of absolute performance metric. The results can be reported in a form resembling the responses you may have given last time you went for a new eyeglass prescription; "A looks better than B (for my simulator implementation)".

The process of test case development, evaluation and analysis is just beginning. The group expects the effort to take about 2 to 3 years before a recommended practices document is ready for IEEE publication. As a result of developing and disseminating this document, the group hopes to make parallel VHDL simulation a productive, positive experience for the VHDL user.

To the group's knowledge, this effort is the first to produce any formal, widely distributed guidance for VHDL model authors seeking improved parallel simulation performance. Contributions, in the form of anecdotal recommendations or test case evaluations and reports, are gratefully appreciated. Please contact the group's reflector via:

- parallel@vhdl.org (to share with the group)
- parallel-request@vhdl.org (to be added to reflector)

Following are an initial set of anecdotal recommendations serving as input to the group. The domain of their applicability is to be determined.

3 Time

The single greatest impact modeling style has on parallel simulation is related to the time domain.

Synchronous parallel simulators must ideally balance the evaluation workload across all processors at each delta cycle. In practice, the work to be done at distinct time steps varies widely, from a single active process to many. Furthermore, the work associated with evaluating distinct processes at distinct time steps varies widely, from a few machine instructions to millions. These deviations from the ideal tend to reduce the performance.

Asynchronous parallel simulators are usually a little more tolerant of deviations from ideal load balancing (described above), however they too are ultimately impacted by load balancing concerns.

Modeling paradigms which insert physically-oriented delay information into a VHDL simulation (e.g. [VITAL95]) increase the correspondence of the simulation to reality at the expense of parallel simulation performance. In addition to the increased computation implied by the detail (seen equally on a uniprocessor and parallel processors), such timing models will tend to distribute evaluations at a wider range of instants in the simulation time domain. The negative performance impacts of back annotation can be reduced by discretizing timing information, either manually or within a tool.

On the other extreme, one may reduce temporal detail using a leveled compiled code or cycle-driven approach. This approach may be implemented either by compiler transformations [JW95] or a modeling style and subset compiler. Either means yields simulation runs with fewer populated time steps, many more active processes at each populated time step, and simpler process evaluations (due to the simplified time model). We would expect such approaches to not only yield good uniprocessor performance, but also increase the exploitable parallelism.

Timing Recommendation:

Try to insure that there are a minimal number of distinct time steps at which processes are active and that the workload at each such time step is relatively comparable.

4 Processes

If there the number of VHDL processes is less than the number of processors available for parallel simulation, simulation efficiency will generally be impaired. In the absence of complex compiler transformations, such models will seldom if ever approach linear speedup.

Load balancing generally works much better if there are enough processes to put at least several VHDL processes on each processor. Ideally all such processes either have approximately equal dynamic execution loads or the dynamic execution load for each process is data-independent and suitable for static approximation (by the compiler backend).

Process Recommendation 1:

Try to partition the model so that there are (many) more VHDL processes than processors used to execute the simulation.

Event driven simulation has intrinsic overhead. Even if the modeling specification calls for event-driven modeling, modeling style can:

- facilitate the clustering of VHDL processes (often described as physical to logical mapping in the parallel discrete event simulation literature),
- facilitate localized use of cycle-driven techniques by a preprocessor or compiler and
- reduced scheduling overhead.

Process Recommendation 2:

Try to simplify sensitivity lists and maintain common sensitivity lists among processes where feasible.

Use of foreign architectures and subprograms complicates the static evaluation of process complexity essential to good load balancing. Whereas use of foreign "library" architectures or subprograms is often a practical matter defined by the modeling project's specifications, it is worth keeping in mind that such libraries are often an impediment to simulation performance. The performance problems associated with such libraries are also seen for uniprocessors, where the overhead and opaqueness (preventing optimizations such as inlining) also reduce simulation performance.

Process Recommendation 3:

If your VHDL compiler does global optimization or load balancing, avoid use of foreign architectures or subprograms where characterizations of the architecture or subprogram body are not available to the VHDL compiler.

5 Signal Data Types

The fixed overhead associated with assigning to a signal is often so large that the actual size of the signal data type is often a second order concern. For example, if three signals each have a data type four bytes in size and are always driven by the same sequential statements and read by the same sequential statements, signal overhead can be reduced by combining these data types into a single record.

Buses, such as the buses used to connect a processor model and memory model, are one very practical application of this rule. Buses can often be represented by a collection of distinct signal declarations and ports. By gathering up some or all of the bus signals into one, two or perhaps three records, the model can be both more efficient and easier to understand. Such consolidation can also be done by some VHDL compilers, such as Auriga [JW92].

Resolved data types are particularly expensive, not only because of the resolution computation, but because the pseudo-process often created to implement resolution usually cannot be evaluated until all inputs are known for a given time interval. Thus resolution functions locally constrain the ability of even asynchronous, parallel simulators to balance computational load across time.

Data Type Recommendation:

Try to design signal types so that there will be as few drivers as possible, especially if the signal is resolved.

6 Shared Variables

Shared variables complicate parallel VHDL simulation, both from the standpoint of logical correctness and simulation performance. Intrinsically, shared variables create zero-delay communication cycles involving two or more processes.

IEEE's RevCom (Review Committee) recognized that shared variable were a problem when they were added to VHDL's 1993 revision. At the recommendation of RevCom, DASC chartered a working group to fix the problem (1076A). This group is working toward a variation of Ada's protected types. Whereas this solution, if adopted by the balloting group, should provide a completely defined behavior, the solution does not solve the intrinsic parallel VHDL simulation performance problem.

In particular, processes sharing a variable will either be unable to evaluate at different times (asynchronous parallel simulation), reducing load balancing efficiency, or are likely to be subject to rollbacks (asynchronous, optimistic parallel simulation). Any of these prospects translate into reduced parallel simulation performance and efficiency.

Shared Variable Recommendation:

Try to avoid use of shared variables when using an asynchronous parallel VHDL simulator. Where they must be used, try to maximize the shared variable's locality to as few processes as possible, ideally processes which are also closely related by signals and timing.

7 Conclusions

Parallel VHDL simulation can be a practical benefit to VHDL users running on a wide variety of parallel platforms (shared memory through massively parallel). This paper has provided a few VHDL coding guidelines intended to increase the chances that your model will benefit from parallel simulation. Through efforts of the IEEE DASC Parallel VHDL Simulation Group, we hope to both evolve additional guidelines and better understand how these guidelines are keyed to particular parallel platforms and simulators.

8 Acknowledgments

The authors wish to acknowledge the IEEE DASC Parallel VHDL Group (over 100 people) for their contributions to this manuscript. Specific contributions to this paper or other assistance came from Philip Wilsey, Praveen Chawla, John Hines, Sumit Ghosh, Paul Menchini, Thomas Hartrum, Hal Carter, Zhiyuan Li, Serafin Olcoz, Bob Parker, Moon Jung Chung, Dave Ackley, John Reed, Pam Fossey, Jeff Snyder, Vijay Vaidyanathan and others.

9 References

- [JS93] J. Sissler, "Assessing the Potential of Multi-Threaded VHDL Simulation", VHDL International User's Forum, Conference Management Services, October, 1993, pp 131-136.
- [HDWP94] Hansen Dai and William Paulsen, "Multithreading VHDL Simulation, Fall 1994 VHDL International User's Forum, Conference Management Services, November, 1994.
- [GPJW95] Gregory Peterson and John Willis, "A Taxonomy of Parallel VHDL Simulation Techniques", Fall 1995 VHDL International User's Forum, Conference Management Services, October, 1995.
- [KS89] K. Serafino, VHSIC Hardware Description Language (VHDL) Benchmark Suite, Technical Report WRDC-TR-89-5046, Design Branch, Microelectronics Division, Wright Research and Development Center, Air Force Systems Command, Wright Patterson Air Force Base, Ohio, October, 1989 (revised in 1990).
- [VITAL95] Draft Standard VITAL ASIC Modeling Specification, IEEE Standards Board, July 1995.
- [JW95] J. Willis, Z. Li and Tsang-Pu Lin, "Use of Embedded Scheduling to Compile VHDL for Effective Parallel Simulation", Proceedings of EuroDAC/EuroVHDL 1995.