

Simulating VHDL Faster and more Efficiently with Cycle-Based Techniques.

Andy Ladd
SpeedSim, Inc.
234 Littleton Road, Suite 2E
Westford, MA 01886
aladd@speedsim.com

Abstract

Advancements in silicon technology are increasing the number of transistors per die while synthesis capabilities are increasing the number of gates that can be designed. These two industry trends are putting heavy pressure on design verification because "gate" creation is rapidly outpacing "gate" verification. With the advent of Cycle-based logic simulation, improvements have been achieved that provide orders of magnitude improvement in software VHDL gate simulation throughput and memory utilization without the high cost of accelerators and emulators.

Section 1. Introduction

For large custom chips or ASICs, design teams are struggling to keep up with the increasing circuit densities available for chip design. ASICs have reached densities of over 1M gates. Custom chip densities are reaching 10M transistors and 2 million logic gates. For very large and complex designs, engineers need to significantly increase the amount of computer-based simulation they employ to assure correctness before they incur the time and expense of a physical prototype.

Static timing verifiers have provided a huge improvement in the effort required to get the timing correct in a large design. On the other hand, Boolean logic (or functional) verification cannot

be done in this way and has remained a significant portion of most new design schedules. Specialized hardware accelerators and emulators offer big performance gains in functional verification for those that can afford to buy and support them. However, there hasn't been a major breakthrough in the performance of commercial software simulators until now. That breakthrough is the arrival of commercially available Cycle-based, logic simulators.

Section 2. Definition of Cycle-Based Simulation

Cycle-based software simulators use computational methods that eliminate unnecessary calculations to achieve huge performance gains in verifying Boolean logic: 1.) Results are only calculated at the end of every clock cycle; 2.) The digital logic is the only part of the design simulated (no timing calculations); and 3.) Typically, only two logic states (1s, & 0s) are computed. By limiting the calculations, Cycle-based simulators can provide very large increases in performance over conventional Event-driven software simulators.

In comparison, Event-driven simulator methods sacrifice performance for rich functionality: every active signal is calculated for every device it propagates through during a full clock cycle. Full Event-driven simulators support: 4 to

28 states; concurrent simulation of Behavioral VHDL, RTL, gate, and transistor representations; full timing calculations for all devices; and the full VHDL standard. Cycle-based simulators only focus on the logic and therefore can highly optimize the calculations for that one function.

Cycle-based simulators are designed for different levels of abstraction. For instance, Redwood Design Automation's Cycle-based simulator accepted RTL input and SpeedSim/3 currently accepts gate level input. Cycle-based simulators can either be two or four states. You will get better performance with two states. Simulating only 1's and 0's is an inherent natural fit with the bits and instruction set of the CPU that you are running on. When simulating an unknown state, the simulator has to run more instructions to mimic the unknown.

Section 3. Background

For over ten years, most of the larger computer companies, such as IBM and Digital Equipment Corporation, developed Cycle-based simulators as key internal tools for their validation process of large designs. They are used with a mix of event-driven simulation at the beginning of the verification phase and then switch over to using Cycle-based simulators on hundreds of workstations for simulating one new chip. These internal simulators were usually based on an internal proprietary design language and were not available to other companies. The advent of commercial Cycle-based simulators make this technology generally available.

Section 4. Fit in the Design Flow

From the system-verification level, Cycle-based and Event-driven simulators behave the same when it comes to the Boolean logic. A Cycle-based simulator can be used like a traditional event-driven simulator. It

can be fed stimulus with a data file or run a program loaded into simulated memory. It is possible to write a VHDL source-code design file that can be simulated on either an Event-driven or Cycle-based simulator with little or no modification.

Cycle-based simulation is particularly effective for pseudo-random testing to get exhaustive testing (functional verification) of a particular section of a chip. For example, verifying the floating point unit (FPU) of a new design could be done with a Cycle-based simulator. You would feed the FPU being simulated a randomly generated set of operands from a behavioral model which would check the results of the model. A Cycle-based simulator running this FPU at 1,000 cycles per second could run four million tested instructions per day on a desktop workstation. This assumes 20 cycles per FP instruction. Running these tests across 100 workstations for 10 days would check four billion instructions!

As a design is rapidly being debugged and fixed, it is good practice to rerun all previously completed sections of the regression tests. As these tests need to be run over and over again, and you need to complete a lot of tests that haven't yet been tested. Cycle-based simulation can be the optimal performance solution for scenarios like this. The verification demands of most smaller designs (< 50K gates) usually wouldn't require a Cycle-based simulator.

Section 5. Performance and Memory Usage Comparison to Event-Driven Simulators

Depending on the options used, Cycle-based simulators can offer one to three orders of magnitude improvement in run-time speed and one to two orders of magnitude improvement in memory utilization at run-time over conventional event-driven simulators.

Below are some actual performance results from SpeedSim/3 to

demonstrate the memory usage and speed of cycle-based simulation:

Benchmark Test *	A.	B.	C.
Size of Design (logic gates)	140K	600K	1000K
Compile Time (minutes)	1	6	9
Runtime Image (Mbytes)	4.5	23	34
Performance, Single Stream (cycles/sec)	112	15	9
Performance, Multi-Stream (cycles/sec)	3226	450	272

* All Tests run on a Single Processor SPARCstation 10

For very large designs, the reduced memory requirements of Cycle-based simulation would allow a design team to simulate the design on almost every workstation on their network. This typically is not possible with an event-driven simulator due to the much larger memory usage. For instance, running a gate level model of a million gate design with an event-driven simulator can generally only be done on a few compute servers with the maximum amount of physical memory.

Section 6. Limitations

While Cycle-based simulators offer tremendous advantages in speed and in memory use, one has to understand what the limitations are in using them. One, if the Cycle-based simulator has only two states, you still need to use an event-driven simulator to derive a correct initialization sequence before starting regression tests with the Cycle-based simulator. This is a small price to pay since a good initialization sequence would normally take on the order of 5,000 to 20,000 cycles of test in comparison to millions of cycles of testing needed for regression.

Second, designs should be synchronous. Asynchronous pieces can be simulated correctly on some cycle simulators but with a performance penalty. Within SpeedSim/3, asynchronous portions of the design are simulated at about one-half the speed of

the rest of the design. From a recent review of different ASIC designs from six different companies, it appears that there is very little asynchronous code as a percent of the total design.

Third, as detailed below, timing must be calculated with a static timing verifier.

And fourth, behavioral modeling has to be done carefully or it can significantly degrade the possible performance gains of Cycle-based simulation. As you dramatically speed up the simulation of the RTL or gate level design, you must do something differently to achieve an equivalent speedup in the behavioral code of your test bench. For example, our experience has shown that writing a test bench in C can be typically much faster than doing the same task in VHDL or Verilog. On a recent Internet conference, some designers quoted gains of 5-10X with C over Verilog behavioral. C has other advantages: the test bench can be portable across VHDL and Verilog; and C programmers who don't have VHDL experience can help write and debug models. There is a growing number of sites that have made a transition to C for performance and feature reasons.

Amdahl's law can come into play for co-simulation. If you co-simulate between an event-driven simulator and a Cycle-based simulator, the performance could largely be determined by the event-

driven simulator depending on the size of the load in each simulator. You also pay a penalty for the overhead of exchanging data between the two simulators if a simulation backplane is used.

Section 7. Combined Use with Static Timing Verification

As an alternative to using full Event-driven simulators for timing analysis, static timing tools are able to provide better coverage faster. By only focusing on timing, they get the same kind of advantages that Cycle-based simulators get for Boolean logic verification. Both tools are targeted at one single job and thus are extremely efficient at that job. Using a Cycle-based simulator would assume the use of static timing verification as a key companion tool rather than calculating timing with an Event-driven simulator. Stated differently, when you use a Cycle-based simulator you are focusing on the pure digital portion of your design and depending on other tools for the analog portions.

Section 8. Availability of Commercial Products

Cycle-based simulation is applicable to high-end designs. There are three reasons why Cycle-based simulators can be offered commercially today. First, the widespread acceptance of VHDL and Verilog make it possible to build a product that will fit into the existing design verification systems of many companies. Second, the use of static timing verification has broadened considerably in the last few years even though commercial tools (i.e. Valid Logic) have been available for the last 9 years. Third, the sizes of designs that need to be verified are growing rapidly and debug complexity is climbing exponentially. The speed and low memory usage of Cycle-based simulators offer relief for this increase in complexity.

Section 9. Fit with Large Designs

The project teams that spend many months simulating the logic design should look at Cycle-based simulation. These teams typically need to run 10 million to 10 billion cycles of test patterns on the new design. The best example is a new microprocessor design. Many of these are now over a million logic gates in complexity. It just isn't feasible to use event-driven simulators to get extensive regression testing. With Cycle-based simulators, it is possible to run new tests in software simulation that couldn't be done with an event-driven simulator. For example, booting a version of the operating system, which has routines like memory array tests removed, is feasible with Cycle-based simulation.

Other appropriate examples of Cycle-based simulation include custom DSP chips; 100K+ gate ASICs; a board design with a number of new custom or ASIC parts; or a large design with many instances of the base design. This is particularly true for something like an 8 processor Symmetric MultiProcessor (SMP) design.

Section 10. Performance on Symmetric MultiProcessing

Design verification teams of large designs look for every opportunity to get improved performance. With Cycle-based simulation methods, a Symmetric MultiProcessing (SMP) platform offers the opportunity to take a single design and break it up so segments of the design are simulating on different CPUs within the SMP box. The result is that this segmented, single-program simulation will run much faster across the multiple CPUs than it would on a single CPU. For designs over 500K logic gates, it is not unusual to achieve super-linear performance gains when going from one CPU to four or more CPUs.

Cycle-based calculation methods lend themselves very well to SMP segmentation in comparison with event-driven. One reason is that inter-processor traffic and cache invalidation rates are minimized because results are only examined at the end of clock cycles. Another reason is that it is much harder to cleanly partition a design in an event-driven simulation to get optimal segmentation.

Section 11. Running Multiple Vector Files

SpeedSim/3 has an additional performance option called Simultaneous Test. Because only 2 states are stored for each netlist node, up to 32 independent streams of vectors can be simulated at the same time on one simulator on one netlist on one computer. Overall verification throughput is increased up to 32 times. For example, amount of data that must be flushed through a Video Processor (say an MPEG) design is huge. Simulating one video image can take a week on an Event-driven simulator. The ability to simulate 32 images at the same time can mean that a Cycle-based simulator can run one second worth of images (about 30 images) in a few hours.

Summary

Large design verification projects are getting increasingly more difficult. The advent of software Cycle-based simulators can provide one to two orders of increased test throughput over the current generation of event-driven simulators,