Evaluation and Benchmarking of VHDL Simulators

Kam H. Wong and Mark D. Feinstein
Pacific Communication Sciences, Inc.
10075 Barnes Canyon Road
San Diego, CA 92121
(619)535-9500
Email: kwong@pcsi.com mdf@pcsi.com

Abstract

Three VHDL simulators were evaluated. Different categories of tests were conducted addressing the classic "interpreted" and "compiled" type simulation and related issues, and performance was compared. With a variety of behavioral, structural and mixed-level benchmark designs, we tested the compilation and simulation speed, monitored memory and disk space usage and design capacity of each simulator. We also investigated their debugging and stimulus creation capabilities, user-programmability, full VHDL support capabilities, PC X-window interface support and on-line help features. In this paper, we will present the benchmarks we used, the testing methodology, results obtained and performance analysis. In addition, we will discuss compatibility with leading commercial synthesis tools, accommodation of third party behavioral-level hardware models for system simulation, and interfaces with other high-level languages such as C to enable software and hardware co-simulation. Some interesting observations and implications will also be presented. The final part shows a rating scheme we constructed to judge the overall performance of each simulator. In general, as a user, we want to maintain an objective viewpoint and try to avoid biasing throughout the evaluation. The results and findings of this research turned us into an informed consumer and, as a result, led us to a logical and reasonable purchase decision.
Evaluation and Benchmarking of VHDL Simulators

October 21, 1992

Kam H. Wong and Mark D. Feinstein
Pacific Communication Sciences, Inc.
10075 Barnes Canyon Road
San Diego, CA 92121
Tel: (619)535-9500
kwong@psei.com mdf@psei.com

Introduction

Three commercial VHDL simulators were evaluated - Simulators A and B are interpreted types and Simulator C is a compiled type

A variety of tests were conducted to benchmark each simulator

Behavioural, structural and mixed-level benchmark designs from the three simulator vendors, public domain, and PCSI were used

Benchmarks were run on an IPX SPARC station with 64 MBytes main memory, 400 MBytes swap and more than 600 MBytes disk space

A scientific rating scheme was constructed to analyze the performance of each simulator under test

In this paper, we will present:
• Results of 2 benchmarks
• Rating methodology
• Interesting observations and implications
BENCH1 - "Big" ALU

Source provided by Vendor B

Test Objectives:
- Design Capacity
- Simulation of synthesizable code

Circuit Details:
- 64 bit addition or 64 X 64 bit multiplication depending on a select signal
- 128 bit wide output signal
- Arithmetic and other packages from synthesis vendor were used

Behavioral model (31 lines of code) was synthesized to approx. 38,000 gates
(27,610 lines of code) using a generic component library

Simulation ran for 128,000ns (64 additions and 64 multiplications)

Experience gained:
- To get both the behavioral and structural code to compile under the three
different simulators, modification of source was done
  (It was due to the flexibility of the language and different interpretations of
  the same piece of code by different simulator vendors)

Pacific Communication Sciences, Inc.

BENCH1 Results

<table>
<thead>
<tr>
<th>BENCH1</th>
<th>Compile Elapsed Time</th>
<th>Compile Max Swap Space</th>
<th>Compiled Data Disk Space</th>
<th>Simulate Elapsed Time</th>
<th>Simulate Max Swap Space</th>
<th>Simulate Temp Disk Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator A Behavioral</td>
<td>3.3 Sec.</td>
<td>N/S</td>
<td>N/S</td>
<td>19.5 Sec.</td>
<td>N/S</td>
<td>N/S</td>
</tr>
<tr>
<td>Simulator B Behavioral</td>
<td>3.3 Sec.</td>
<td>N/S</td>
<td>N/S</td>
<td>11.5 Sec.</td>
<td>N/S</td>
<td>N/S</td>
</tr>
<tr>
<td>Simulator C Behavioral</td>
<td>7.5 Sec.</td>
<td>N/S</td>
<td>N/S</td>
<td>8.0 Sec.</td>
<td>N/S</td>
<td>N/S</td>
</tr>
<tr>
<td>Simulator A Structural</td>
<td>11.5 Min.</td>
<td>25.0 MB</td>
<td>9.8 MB</td>
<td>65.5 Min.</td>
<td>71.0 MB</td>
<td>64.7 MB</td>
</tr>
<tr>
<td>Simulator B Structural</td>
<td>25.1 Min.</td>
<td>59.0 MB</td>
<td>13.5 MB</td>
<td>19.5 Min.</td>
<td>91.0 MB</td>
<td>&lt; 1.0 MB</td>
</tr>
<tr>
<td>Simulator C Structural</td>
<td>105.7 Min.</td>
<td>135.0 MB</td>
<td>26.7 MB</td>
<td>23.2 Min.</td>
<td>60.0 MB</td>
<td>&lt; 2.0 MB</td>
</tr>
</tbody>
</table>

N/S = Not Significant

Pacific Communication Sciences, Inc.
BENCH2 - 15-input Parity Generator

Source obtained from public domain

Test objective:
- Design Capacity

Circuit Details:
- 81,915 component gates and 81,932 signals
- 15-input parity generator flat structural design

367,889 lines (9.6 MB) of code

Simulation ran for 16,384ns and went through all the gates 1638 times

Experience gained:
- It posed a challenge for all 3 simulators due to its large, flat nature; Vendors claimed their tools handle hierarchical designs a lot better

BENCH2 Results

<table>
<thead>
<tr>
<th>BENCH2</th>
<th>Compile Elapsed Time</th>
<th>Compile Max Swap Space</th>
<th>Compiled Data Disk Space</th>
<th>Simulate Elapsed Time</th>
<th>Simulate Max Swap Space</th>
<th>Simulate Temp Disk Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator A *</td>
<td>1.5 Hours</td>
<td>110.0 MB</td>
<td>41.3 MB</td>
<td>4.0 Hours</td>
<td>130.0 MB</td>
<td>10.6 MB</td>
</tr>
<tr>
<td>Simulator B</td>
<td>38.8 Hours</td>
<td>269.3 MB</td>
<td>52.0 MB</td>
<td>26.0 Min.</td>
<td>113.0 MB</td>
<td>&lt; 1.0 MB</td>
</tr>
<tr>
<td>Simulator C</td>
<td>Didn't Finish</td>
<td>&gt; 400.0 MB</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

N/A = Not Applicable
* = With Beta version of the new release
(Originally released software didn’t finish compilation successfully)
Rating Methodology

Category 1 (Wc = 0.5)
- Simulation Performance  \(Wi = 10; We = 5.0\)
- Compilation Performance  \(Wi = 9; We = 4.5\)
- Vendor Technical Support  \(Wi = 9; We = 4.5\)
- Future Expectations  \(Wi = 9; We = 4.5\)
  (Future product quality, integration and business climate)
- Debugging Capabilities  \(Wi = 8; We = 4.0\)
- Graphical Interface  \(Wi = 8; We = 4.0\)
- Capacity  \(Wi = 6; We = 3.0\)

Category 2 (Wc = 0.3)
- User-programmability  \(Wi = 7; We = 2.1\)
- Stimulus Creation  \(Wi = 7; We = 2.1\)
- Current Level of Tool Integration  \(Wi = 6.5; We = 1.95\)
- Memory Usage  \(Wi = 6; We = 1.8\)
- Disk Space Usage  \(Wi = 5; We = 1.5\)

Category 3 (Wc = 0.2)
- Responses of References  \(Wi = 5; We = 1.0\)
- Degree of Full 1076 Compliance  \(Wi = 4; We = 0.8\)
- On-line Help and Manuals  \(Wi = 3; We = 0.6\)

Wi = Individual item weight factor; Wc = Category weight factor; We = Wi * Wc
Weighted rating of each item = absolute rating * We

Pacific Communication Sciences, Inc.

Interesting Observations and Implications

The generally believed principle that Compiled type simulators compile slower, use more swap and disk space but simulate and debug faster than interpreted type simulators was confirmed in our case

Simulators with simpler and more user-friendly interfaces did not provide much room for user customization

Simulators providing an easy-to-use stimulus creation feature minimized the need for a formal VHDL test bench and thus shortened coding time

Technical support from smaller vendors was better due to easier access to the developers

Graphical schematic drawing feature from simulator vendors was helpful but very difficult to use

Hardware modeling and high-level programming language interface was important for system simulation

Minimal timing model and back-annotation support from ASIC vendors

Pacific Communication Sciences, Inc.
Interesting Observations and Implications (cont’d)

- Benchmarks provided by a particular vendor usually performed the best on their own simulator

- Discuss with your simulator vendor how to optimize your code (since they know how their software was built)

- A totally independent source of benchmarks to certify simulators is needed

- Simulators do not handle flat designs well

- Design hierarchically (A good design practice, anyway)

- Different simulators sometimes interpret VHDL code differently

- Use a subset of VHDL code (and establish your style) which can be commonly understood by all your in-house simulators and even your synthesizer

Pacific Communication Sciences, Inc.