

# Criteria for Evaluating Benchmarks of VHDL Simulators

Ron Werner

Computer Integrated Design and Manufacturing

Motorola

*Government and Space Technology Group*

*Scottsdale, Arizona 85252*

---

## Abstract

Benchmarks are often referenced to make engineering tool performance evaluations. Over the past year, several groups have completed benchmarks to quantify the ability of VHDL simulators. Due to the relative immaturity of VHDL, many people are in a learning mode and are creating test cases as a good way to quickly learn about a VHDL tool. Many of the evaluators that have published benchmark information, have had an active history of working with VHDL and are regarded as experts in the VHDL arena.

This paper will describe criteria to be considered when developing and reviewing a benchmark:

- What criteria should one consider when performing or evaluating a benchmark?
- What is the internal design environment?
- Internal test results of criteria will be provided.
- What are limitations in applying results?

## **Introduction**

Before a new tool or process is accepted by most large companies, testing of the system is performed to insure that it will meet and pass minimum performance criteria. Tools and processes are evaluated to demonstrate what the added value is, and to determine "the real world" limitations of the product. Testing of a system is often termed as benchmarking information. These benchmarks often determine the proposed "standard" values of a product. Therefore it can be seen that benchmarks can lead to unrealistic expectations and false hopes.

Benchmarking, or evaluating the results of a benchmark, can consume a significant amount of time. Therefore, there has to be some benefit that will inspire one to properly evaluate and perform additional benchmarks when necessary. As a group, we believed that VHDL had matured sufficiently to select a simulator that would be made available for broad engineering use. Now that the scope of the project is defined it is critical to weigh benchmark information in the light of our design environment and to evaluate published benchmark information.

### **Internal design environment**

When bringing a new tool or process into our group, the tool or process must be integrated into the environment that the engineers are currently using. The engineers must acknowledge the value of the tool/process to gain support from the critical mass of experienced people needed to make the tool/process a success.

There are three main areas that must be addressed when evaluating a design environment. These areas include:

- networking/platforms
- design types
- engineering methodologies

To illustrate how the items should be reviewed, I will utilize our design system as an example.

### **Networking/platforms.**

We have multiple local networks consisting of PC, IBM RS6000 and SUN workstations connected together into one large network. Licenses are served from common servers and most application software also resides on these servers. Most of our work stations are IBM RS6000's and we utilize Xvision extensively; Xvision enables access to workstations from a PC. This system requires substantial overhead (user will notice lower operating speeds) and for specific applications, may affect the system to the point that it will significantly diminish cycle time performance.

Due to this networking scheme, application software performance can be greatly affected by network capability and usage. An issue that continually affects us is diversity of operating systems (OS). In our networking scheme, the OS of individual nodes has to be compatible with the OS of other nodes. Due to this limitation, the OS version of each node cannot vary significantly. Therefore, vendors must supply application software that is able to operate on a wide variety of OS, as utilized by different customers. Broadening the capability of the application software allows us to

change OS based more on the network requirements and less on the application requirements. When software is developed to operate on a wide variety of OS it often slows the performance of the application software. We now have the effects of a network communications scheme, plus the effects of a versatile application that tend to slow the performance of our operation.

Most performance benchmarks of simulation tools avoid the negative effects of networking by utilizing a single node and loading on the most optimum OS. Some software vendors take advantage of improved OS calls to boost the performance of their product. This to their advantage but makes the benchmark more dependent on a specific OS release. As a result, when most simulation benchmarks are performed, the data may be considerably skewed, based on the effects of networks and OS.

### **Design types.**

Although it is difficult, if not impossible, to define the universal constructs of a design, it is possible to categorize major digital design electronic processes. Most designs may be placed into one of the following major categories (circuit drivers):

Synchronous	Asynchronous
Many flip-flops	DSP based
Micro Processor	Communication

Some simulators operate more effectively in one of these design categories and most benchmarks are developed to cover, at best, a few design categories. At the conclusion of a benchmark it would be very easy to assume that equal performance could be expected in all categories. This is not always the case and performance could be dramatically different with the other untested design style categories. Test case development has a large impact on the outcome of a specific test. In our business, we utilize most of the design categories and these are often utilized by different design groups. When the results of a benchmark are to be reviewed by other groups, we insure that the benchmark environment is clearly defined so that the entire group can see how their environment overlays our test evaluation.

### **Engineering methodologies.**

Engineering methodologies are continually changing and, to a certain extent, are dependent on the technology being used. The main components that make up engineering methodologies are:

- Process flow
- Simulation Models
- Stimulus generation/response vector evaluation

## **Process flow.**

The questions that must be considered when benchmarking process flow are: What is the design flow to complete a project? How is the initial design entered? What is the design information required to pass a design to the next design process?

In our group, initial digital design entry is accomplished with tools like SPW, i-Logix, schematic capture and/or VHDL behavioral design. Each of these design entry tools require different platforms and different engineering methodologies. Therefore, it is critical that tool output files be created in a form that can be immediately utilized by the receiving tool or group (individuals that will be performing functions based on the data).

Each of these design entry methods must be tested to insure that the simulator can simulate a design in a reasonable amount of time. We have found that simulator vendors attempt to promote their view of the design world and optimize their tool for that view. In the schematic capture world, there are often many ways of inputting design information. The vendor normally has a myopic view of how the information should be entered. This view is often derived from an interpretation of system operation and what tools are available to create support files. In your system, if you have not followed the vendors' flow explicitly, when you attempt to move from schematic to simulation utilizing the vendor tools the system may not find all of the required data. In both cases, none of the rules for inputting information has been violated but the method you have adopted will not yield the same results.

In the top down approach to design (i.e. behavioral, RTL, gate), it is expected that when problems are detected in one level of the design process one should go back to the top level and make necessary changes. For several reasons this often does not happen. When evaluating a simulation system one must look at how incremental design changes affect productivity of the tool. This incremental design change methodology forces the simulation to handle local iterative steps as well as coming down the pike again (total redesign).

## **Simulation Models.**

Obtaining models to enable simulation has been an issue since the first simulators came into existence and it will continue to be a problem for many more years. For board simulations, many organizations, including ourselves, rely on third party vendors (e.g. LMG/Synopsys) to provide many of the models required. LMG supplies us with software models (Smart Models) and hardware models (Physical Modeler). It is important that the simulator be able to pass control information to these models and that system performance is not radically degraded when one utilizes these types of models. In the upper stages of a top-down methodology the models are developed by the design organization. It is at the lower levels of the design that third party vendors are brought in to aid in solving the problem of simulation model development. Again, most benchmarks do not include testing of third party suppliers. It is often assumed that utilization of these vendors products is straight forward and will not cause any significant simulation penalty.

We have found that utilization of third party vendor models may have a significant impact on the ability of the simulator to provide the information required. In particular, model control information, defined as "model directives", that could be sent to the third party vendors' model in some simulators, could not be passed in other simulators. These "model directives" are documented in the vendor literature and lack of this ability limits the capability of a simulation.

### **Stimulus response vector generation.**

Many hours of effort are often applied to stimulus and response test vectors. Stimulus vectors may be generated by the creation of VHDL testbench, WAVES files or "force" files. Response vectors may be evaluated by utilizing VHDL testbench, WAVES files or by capturing the output results and manually evaluating them.

When creating or evaluating a benchmark, look closely at stimulation vector generation techniques and response analysis routines. These two activities will have significant impact on the outcome of the benchmark data when there has to be manual interaction by the operator for stimulus generation or response evaluation. This manual interaction may be detected by the operator having to use force files and those simulations that need to be captured by the system for manual evaluation. Although these two activities are the least preferred methods, they are very effective during the initial design phase when one is attempting various alternatives.

### **Summary of internal criteria/requirements and testing results.**

In an attempt to understand the above criteria we established some feature criteria driven by our requirements, that would directly affect the design team capabilities. These feature criteria established the first gate in the evaluation process and were placed on a chart so that we could make a side by side comparison of several simulators. Please refer to the attached figure 1 "Simulation Comparison Matrix". The "Simulation Comparison Matrix" is a compilation of our feature criteria and the testing results that we obtained.

Some of the values on the chart were solicited from internal people that were actively using those products. The blank spaces indicate that the product literature states that the feature exists but were not evaluated by internal personnel. Weighting factors were applied to the feature criteria as a method for us to determine relative importance of features to us.

Based on the total weighted score it would indicate that vendor C is the clear winner. Without this evaluation matrix, we would have assumed that vendor B would be the best choice to continue with. Vendor B initially appeared to be better due to the benchmarks performed and the perceived integration into our system. Remember that one evaluation does not mean that the results will be adequate for all functional groups. In our situation, this fact became very apparent in regards to vendor D. Vendor D was rated very low on the matrix but it is still the best tool available to perform the specific function required for one of our design teams.

## **SUMMARY**

A complete evaluation of simulation tools should involve determining all of the aspects of your design environment as discussed in this paper. Most bench marks take into account only a small portion of a design environment. Many personnel who are creating the bench marks encompass the world as they envision it and it is reflected in the bench mark. Many reviewers of benchmark data do not envision their environment and therefore cannot project benchmark results into their scenario. The benchmark feature criteria, with the associated testing results, were very revealing for us and validated our efforts.

After our testing was completed, two other independent benchmark evaluations were published. Their benchmark data did not coincide with the results that we obtained. Upon further investigation it was determined that the bench mark results were quite valid given the specific evaluation that they were addressing. The personnel that put together the test cases were VHDL experts and were able to "skirt" (work around) problems that most novice VHDL users would have difficulty with. Our investigation showed that many external benchmarks did not accurately represent our environment and our design process flows. In a general sense, product groups that have large projects will find that most benchmarks fill only a small gap when attempting to evaluate their VHDL simulator needs.

Bench marking is an important step that must be completed when attempting to evaluate software application tools and it's affect on the design environment. It is critical that one thoroughly determine what is expected to be accomplished in a benchmark. Most importantly you need to understand what your desired environment is and proceed accordingly.

# Simulation Comparison Matrix

Key:

Value system will be from 0 - 5, with 5 being most important/best in class  
 Weighting system will be from 1 - 5, with 5 being most desired

VENDOR:		A	B	C	D	Weight
---------	--	---	---	---	---	--------

<b>TOTAL (Weighted Score)</b>		390.5	259	500	181	
-------------------------------	--	-------	-----	-----	-----	--

## VHDL simulation capability

VHDL standard compliance	5	5	5	5	5
Timing analysis capability	5	5	5	5	5
Speed of simulation	5	5	5	5	3

## Interface of model sources

LMC smart models		5	5	2	5
LMC 1200	0	1	5	5	4
Other model sources	5	5	5	5	1

## Ease of stimulating LMC

Software models		2		2	5
Hardware models		0	5	5	5
VHDL source models	5	3	5		5

9.21

<b>VENDOR:</b>		<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>Weight</b>
----------------	--	----------	----------	----------	----------	---------------

**Integration to other design/analysis tools**

Concept	0	0	0	0	4
Allegro	0	0	0	0	2
FPGA	0	2	2	2	4
Vendor tool sets	5	1	5	5	5

**Ease of bringing in VHDL code from other systems**

HUMM	4	4	4		3
SPW	5	4	5		3.5
iLogic	5	4	5		3.5

**Incremental design capabilities**

	5	5		4
--	---	---	--	---

**Maturity of product**

4	0	5	3	4.5
---	---	---	---	-----

**user intuitive**

debugging capabilities	5	1	5		5
ease of use	5	1	5		5
Error message accuracy	4	2	5		5

9.22

Simulation Comparison Matrix

Figure 1

<b>VENDOR:</b>		<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>Weight</b>
----------------	--	----------	----------	----------	----------	---------------

User interface

Stimulus wave form generation	5	1	3		3.5
Wave form display	5	1	5		4
Model template creation/support	0	3	4		3
Complexity of user interface	5	1	5		4
Complexity of support required for simulation setup	5	0	5		4
User documentation quality/quantity/form/on-line/test cas	5	0	5		4
Source level code manuverability	4	5	5		3.5
	111.5	38	120	0	

Overhead support not required

System	5	0	4		3
User training	4	2	4		2
Account Setup	5	0	5		3

Amount of installed GSTG user usage

H	N	H	H
---	---	---	---

9.23

## **Acknowledgments**

I would like to recognize the following for providing support and assistance in the preparation of this document:

Ron Lieberman, Harv Rakestrow,  
John Scruggs, Gordon Watrous