# THE IEEE VERILOG-2001 SIMULATION TOOL SCOREBOARD

*Clifford E. Cummings - Sunburst Design, Inc., Beaverton, OR*

## POST-DVCon NOTES (Rev 1.2):

The testing of so many simulators and synthesis tools proved to be more than I could do in a reasonable amount of time, so this paper only focuses on simulators and the title of the paper was changed to reflect this fact. Perhaps I will get around to testing synthesis tools by the next DVCON.

Verilog-XL was removed from the compliance tables because Cadence does not intend to add support for Verilog-2001 features to Verilog-XL. See Section 12.0 for more details.

## ABSTRACT

Verilog-2001 added many valuable enhancements to the IEEE1364-1995 Verilog Standard, but when can we safely use them? When the full suite of tools used by your company to do design all support Verilog-2001 enhancements, your company can safely start taking advantage of the enhancements.

This paper details a number Verilog-2001 coding examples and indicates which simulation tools support the enhancement. This paper is not intended to run performance benchmarks against the different simulation vendors and indeed does not include performance data. This paper is intended to inform the Verilog design and synthesis community which Verilog-2001 enhancements have been implemented by the various vendors so that the end-user can scan the list of vendors for implemented enhancements to determine when their company can start coding with the enhanced Verilog-2001 coding styles.

This paper includes multiple "scorecards" (tables) to show which simulation vendors support the important Verilog-2001 enhancements. The latest version of tools from major EDA vendors are represented on the "scorecards."

## 1.0 Introduction

The IEEE Verilog-2001 Standard introduced a number of enhancements intended to make designs more concise and more powerful. Stuart Sutherland has published a book on Verilog-2001 enhancements and ordered those enhancements by number. This paper re-orders the enhancements, according to user requested priorities and RTL-coding partitions, but I do cross reference the enhancements discussed in this paper with the enhancement numbers as reported in Sutherland's book for easy correlation.

At the time that this paper went to publication, I was not done testing as much as I had wanted. This paper will continue to be updated and readers are encouraged to go to the Sunburst Design web page referenced at the end of this paper to download copies of this paper with updated information.

## 2.0 Test Suite and Tool Versions

Simulation tools that were tested with the beta version of the Sunburst Design Basic Verilog-2001 Compliance Commercial Test Suite, included (abbreviations used in the tables):

**[SR#]** Sutherland Reference # - *Verilog 2001*
**VCS -** Synopsys, VCS version 7.0
**SS -** Synopsys SystemSim version 2.1.1 (the Superlog simulator)
**MTI -** Model Technology ModelSim version 5.7
**NC -** Cadence NC-Verilog version 4.2 (beta)
**SIL -** Simucad Silos version 2002.100
**VXL -** Cadence Verilog-XL - will not support Verilog-2001. See Section 12.0.

### 2.1 KEY - Table abbreviations

Some abbreviations were used in the tables shown in this paper. The following abbreviations were used in the compliance data tables.

**X** - Feature is supported
**-** - Feature not supported - syntax error reported
**IG** - Syntax was ignored - feature not supported
**MSG** - Tool recognized the syntax but gave a message indicating that the feature is not yet supported.

## 3.0 Top Five Enhancements

At the International Verilog Conference (IVC) in 1996, a "Birds Of a Feather" panel session was held where panelists and audience members submitted enhancement ideas and the entire group voted for the top-five enhancements that they wanted added to the Verilog language.

Although numerous enhancements were ultimately considered and many enhancements added to the Verilog 2001 Standard, the top-five requested enhancements were:

#1 - Verilog generate statement
#2 - Multi-dimensional arrays
#3 - Better Verilog file I/O
#4 - Re-entrant tasks
#5 - Better configuration control

### 3.1 #1 Generate Statements

Verilog generate statements are divided into three main groups: the generate for-loop, the generate if-else statement and the generate case statement. In conversations with vendors, the flexibility of the Verilog generate for-loops seems to be proving the most difficult aspect to implement of this requested enhancement.

As shown in Table 1, vendors have started to implement the generate statements but as of this publication, none of the vendors had supported nested generate for-loops. Future testing will also examine generate for-loops with non-contiguous incrementing and decrementing.

### 3.2 Array of Instance

It should be noted that most contiguous generate for-loops could be more easily coded using the Array of Instance construct that was added to Verilog-1995 and is now well supported by vendors. Engineers should think first about the array of instance and then fall back to a generate for-loop. For

instantiating a simple contiguous set of I/O pads, the array of instance is better supported and far simpler than an equivalent generate for-loop.

Table 1 - Verilog-2001 - The Top Five Requested Enhancements

| [SR#] | Top-Five Requested Enhancements | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| | Verilog-1995 Array of Instance | X | X | X | X | X |
| [36] | (1) generate for-loop | X | X | X | - | ?? |
| [36] | (1) nested generate for-loop | - | - | - | - | - |
| [36] | (1) generate if-else | X | X | X | - | X |
| [36] | (1) generate if-else-if | X | X | X | - | **BUG** |
| [36] | (1) generate case | X | X | X | - | X |
| [15-17] | (2) multi-dimensional arrays | X | X | X | - | - |
| [16] | (2) 2-D array of reals | - | X | X | - | - |
| [30-31] | (3) enhanced file I/O | X | X | X | X | X |
| [30] | (3) file I/O opening files for modification | X | X | X | - | X |
| [7] | (4) automatic tasks | X | X | **MSG** | X | - |
| | (4) recursive functions (Verilog-1995) | X | X | X | X | - |
| [8] | (4) recursive automatic functions | X | X | **MSG** | X | - |
| [37] | (5) Verilog configuration files | - | - | - | - | - |

**??** - The generate for-loop is almost useless without multi-dimensional arrays

### 3.3 #2 Multi-Dimensional Arrays

Verilog-2001 permits the declaration and use of multidimensional arrays. Former Verilog-1995 restrictions that only allowed two dimensional arrays, and then only word access into the arrays, have been removed. The Sunburst suite tested 2-D arrays with word, part-select and bit access as well as 3-D arrays also with word, part-select and bit access. The suite also tested for 2-D declarations of real values.

### 3.4 #3 Enhanced File I/O

Verilog-2001 offers much more powerful file I/O and string manipulation capability over Verilog-1995. As of this date, the Sunburst suite is incomplete in testing all of the numerous new file I/O capabilities, but the suite will be expanded and used to do additional testing of vendor tools. The suite did open and close files using every new read, write and append mode, and did some other file I/O testing. Although the file I/O capabilities are now native to many simulation tools, users can still download a nearly identical set of capabilities using PLI routines from Chris Spear's web site, referenced at the end of this paper. Many of the Verilog-2001 file I/O enhancements were patterned after Chris' pre-existing file I/O PLI routines.

### 3.5 #4 Reentrant Tasks and Functions

Verilog-1995 tasks and functions use static variables, which means that a task with delays that is called a second time before the first invocation is finished, will share common-static variable, usually with undesirable results. Verilog-2001 allows users to add the keyword "automatic" to Verilog tasks and functions to force the automatic versions to dynamically allocated variables for each task or function call.

Some simulators have started to support automatic tasks and functions, while other simulators like ModelSim do not support this functionality yet but give a cute message that "this Verilog-2001 feature is not yet supported."

In the testing that I did with recursive function calls, I noted that some vendors even partially support recursive function calls in Verilog-1995 while others do not.

### 3.6 #5 Verilog Configuration Files

Verilog-2001 configuration files are intended to give the user better control of binding files to instances in a design during simulation (to replace the ugly and non-standard **`uselib** directive) while also offering a language method for selecting library directories (to replace the command line switches **-y** and **+libext**) as well as library files (to replace the **-v** command line switch). Verilog configuration files add new keywords such as **library**, **config-endconfig**, **design**, **default**, **liblist** and **instance**.

Unfortunately, none of the vendors tested supports any of the features of this valuable design-control enhancement.

## 4.0 The ANSI-Port Enhancements

ANSI style port enhancements provide a concise, non-redundant way to make port declarations in Verilog-2001. The most useful and powerful form of ANSI style ports is making port directions, data types and port names all in the same declaration and all vendors seem to support this correctly with one notable exception. Some vendors seem to have problems when port directions are separated from explicit wire declarations. Although the latter is the less important part of the enhancements, it is nonetheless annoying.

ANSI style parameters, vital to supporting parameterized reusable or re-sizable models is somewhat poorly supported or subject to bugs. Equally important is the ability to redefine parameters on an instance by instance basis, and support for the new named parameter passing capability is also somewhat shaky from some vendors.

Cleaning up ANSI style module ports and parameters should be a priority for every vendor. Under the category of credit where credit is due, ModelSim passed all of the ANSI port tests in the Sunburst Design Basic Verilog-2001 Compliance Test Suite. Kudos to the ModelSim team for getting it right!

Vendors have mixed records of success when it comes to extending ANSI styles to tasks, functions, User Defined Primitives (UDPs), etc., but again, these are not as commonly needed as the ANSI style ports. ANSI support for tasks and functions is still relatively important but should be implemented.

Table 2 - Verilog-2001 - ANSI-Port Enhancements

| **[SR#]** | **ANSI Port Enhancements** | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| **[1]** | Combined port-data type declarations | X | X | X | X | X |
| **[1]** | Combined port-data types - explicit wires | X | **BUG** | X | X | **BUG** |
| **[2]** | ANSI style module ports | X | X | X | X | X |
| **[3]** | ANSI style parameters | - | **BUG** | X | X | X |
| **[27]** | Named parameter redefinition | X | X | X | X | X |
| **[6]** | ANSI style task/function ports | X | X | X | X | X |
| **[4]** | ANSI style UDP ports | - | - | X | - | X |
| **[26]** | Real & integer parameters | - | X | X | - | - |
| **[26]** | Sized parameters | **IG** | **IG** | X | **IG** | **IG** |

# 5.0 The Fundamental RTL Enhancements

There is another subset of Verilog-2001 enhancements that is important to RTL coders and that should not be too hard to implement, this is the fundamental RTL enhancement subset.

## 5.1 Comma-Separated Sensitivity Lists

In VHDL, the signals in a process sensitivity list are separated by commas. In Verilog the signals are separated by the "**or**" keyword. This means that a Verilog sensitivity list is generally more verbose than an equivalent VHDL sensitivity list. I personally found this to be offensive! We added comma-separated sensitivity lists to Verilog and this really should be the simplest of enhancements to implement, but one vendor flagged the comma as an error when placed into a sensitivity list with posedges and negedges and another vendor did not support it at all! This should be fixed post haste!

## 5.2 The @* Combinational Sensitivity List

Even better than the comma separated sensitivity list is the very concise **@*** combinational sensitivity list. The **@*** basically was added to mean, if the synthesis tool wants it, then so does the simulator! Note that although, **@(*)** is also currently a legal form of this sensitivity list, there is some consideration being made by the Verilog standards groups to remove support for this form. The problem is that **(*** is also the opening delimiter for the new Verilog-2001 attributes, and tool vendors have complained about the difficulty in distinguishing between the two. I personally would support removal of the **@(*)** style, and if your company writes any in-house tools, adopting a coding standard that prohibits **@(*)** will probably make your tool-creation job easier.

**Guideline: Use @* for combinational sensitivity lists and do not use @(*)**

Some vendors are doing a good job of supporting the **@*** feature while one vendor has implemented the SystemVerilog **always_comb** statement, which is currently a slight super-set of the **@*** capability. It would be nice to have the **@*** capability implemented universally by all vendors.

## 5.3 Implicit Internal 1-bit Wire Declarations

Verilog-2001 no longer requires that 1-bit internal wires be declared that are driven from continuous assignments. They should come into existence automatically. This was a non-orthogonal wart on the Verilog-1995 language. Unfortunately, most vendors still have not fixed this for Verilog-2001.

## 5.4 `default_nettype none

Verilog-2001 also added a compiler directive to force engineers to declare all variables, including 1-bit wires and port wires for those misguided souls who think that more declarations equates to better design practices (actually you have now doubled the number of places where you can introduce a typo into your code but now a typo in a declaration will cause an error to show up on your good RTL code - hopefully you will spot the error in the declaration instead of severely altering RTL code before realizing the RTL code really was good the whole time). Only one vendor has correctly implemented this "feature" but I am not encouraging other vendors to spend much effort on this "enhancement" until the good stuff has been implemented.

## 5.5 X & Z Extension Past 32 Bits

Making an assignment of `bz or `bx in Verilog-1995 to a left-hand-side (LHS) variable that is larger than 32 bits causes only the 32 LSBs to be assigned the X and Z values. This is fixed in Verilog-2001 but only one vendor has made the change.

### 5.6 Variable Declaration Assignments

Verilog-2001 permits variables to be declared with and initial value. The initial value is assigned as if it were in an initial block so there is no guarantee that the initial value will be assigned first, and this coding style is generally not a good idea for synthesizable RTL coding (the real hardware may not be capable of initializing to the selected value, causing a mismatch between pre- and post-synthesis simulations).

Some vendors have implemented this feature, some still flag it as an error and one vendor just ignores the initializations. Although useful, this is not a "must-have" enhancement.

### 5.7 Enhanced Conditional Compilation

Verilog-2001 adds the **`ifndef** and **`elsif** conditional compilation compiler directives to the set that already includes **`ifdef**, **`else**, **`endif**. Again, all vendors with one notable exception have implemented this very useful enhancement.

### 5.8 Standardized Random!

Courtesy of Cadence, the Verilog-2001 Standard now includes the exact code used to generate random numbers so that a user can get the exact same random numbers using any Verilog simulator. Testing shows that all simulators have implemented the same **$random** system function. The Verilog-2001 standard also standardized the less frequently used random distribution functions and all vendors have similarly implemented these functions except one. The latter functions are not frequently used by RTL coders and are not as important to fix as other enhancements.

Table 3 - Verilog-2001 - Fundamental RTL Enhancements

| [SR#] | | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| **[10]** | Comma-separated - combinational logic | X | X | X | X | X |
| **[10]** | Comma-separated - sequential logic | X | X | X | X | X |
| **[11]** | @* combinational sensitivity list | X | **X** | X | X | X |
| **[25]** | Single attributes | X | - | - | - | - |
| **[25]** | Multiple attributes | - | - | - | - | - |
| **[12]** | Implicit internal 1-bit wires | - | - | X | X | X |
| **[13]** | `default nettype none | - | - | **BUG** | X | - |
| **[23]** | X & Z extension past 32 bits | - | - | - | X | - |
| **[5]** | Variable declaration assignments | X | **IG** | X | X | - |
| **[34]** | Enhanced conditional compilation | X | X | X | X | - |
| **[26]** | Standard random number generation | X | X | X | X | X |
| **[26]** | Standard distribution number generation | **BUG** | X | X | X | **BUG** |

# 6.0 Signed Arithmetic and Power Operator

Signed arithmetic was a much requested enhancement to the Verilog-2001 language. Although all vendors have addressed signed arithmetic, a couple of the implementations have a few notable bugs. A rule of thumb to remember when working with Verilog signed arithmetic is that all operands and the destination must all be signed variables, otherwise you generally end up with an unsigned result.

The power operator has also been implemented by a subset of the simulation tool vendors.

Table 4 - Verilog-2001 - Signed Arithmetic & Power Operator Enhancements

| [SR#] | | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| [18-22] | Signed arithmetic | X | X | **BUG** | X | - |
| [24] | Power operator | X | X | X | - | - |

# 7.0 IP Block Enhancements

Two enhancements added to the Verilog language to assist in the development of robust Intellectual Property (IP) blocks are the **`localparam`** and the constant function.

## 7.1 localparam

The new **`localparam`** keyword makes it possible to declare parameters that cannot be changed by **`defparam`** or other parameter redefinition techniques. The **`localparam`** will generally be calculated from other parameters that are passed to a module.

One example would be to calculate the memory depth of a RAM device based on the size of the address bus. Allowing a user to modify both the address bus size and the memory depth could lead to incompatible parameter values. Vendors have been slow to implement this useful enhancement.

## 7.2 Constant Functions

A constant function is a function that is run at compile time to calculate values that will be used to size portions of a design or to assign logical values to parameters based on other parameter values.

One example would be to calculate the *ceiling of the log base-2* of a number, such as calculating the number of address bits that are required to address an odd-sized memory.

Although a very useful enhancement, this is going to be a hard enhancement to implement and may take some time to show up in a vendor's tool. This enhancement is very important to IP developers who are trying to create complex parameterizable models.

Table 5 - Verilog-2001 - IP Block Enhancements

| [SR#] | | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| [28] | localparam | - | X | **BUG** | - | - |
| [9] | Constant functions | - | - | - | - | - |

# 8.0 Miscellaneous Nice Enhancements

Other nice enhancements that were added to Verilog-2001 include enhanced +command option testing, constant part-select indexing and a standardized SDF **`$sdf_annotate`** command.

## 8.1 Enhanced +Command Option Testing

Verilog-1995 had the ability to recognize +command options using the **`$test$plusargs`** system function to initiate specific desired simulation activity. Veilog-2001 adds the ability to read the values of the +command options using the **`$value$plusargs`** system function. Implementation by vendors ranges from sporadic to buggy.

One interesting note is that verification teams often like to use the **`+all`** user-define plusarg to tell a compiled regression suite to run all tests. When I tried this with NC-Verilog, I discovered that NC-Verilog already has **`+all`** reserved for a tool-specific command (ouch!)

## 8.2 Constant Part-Select Indexing

Verilog does not permit variable part-select indexing in a for-loop, even if the part-select really is a fixed width. Most Verilog users avoid this problem by using a shift operator to get the desired bits moved to the correct position for assignment. Verilog-2001 adds **`+:`** and **`-:`** tokens that permit a fixed width to be specified on the right-hand side of the token. A couple of vendors have implemented this enhancement and one vendor has implemented it with bugs.

Table 6 - Verilog-2001 - Miscellaneous Enhancements

| [SR#] | | VCS | SS | MTI | NC | SIL |
|---|---|---|---|---|---|---|
| [33] | $value$plusargs | **BUG** | - | X | (+all) | - |
| [14] | Constant part-selects +: -: | X | X | X | - | - |

## 8.3 Standard $sdf_annotate

Although not tested in the current compliance suite, most vendors have recognized the standard **`$sdf_annotate`** command for years. Now it is just official.

# 9.0 Non-Tested Enhancements

There are other Verilog-2001 enhancements that have not yet been tested by the Sunburst suite, these include: the line-file compiler directive (Sutherland #35), a series of enhancements added to increase the accuracy of Verilog ASIC models and timing (Sutherland 38-41), Extensions to VCD files, enhanced PLA system tasks and enhanced PLI support for Verilog-2001 (Sutherland 43-45).

Tests for these enhancements may be added at a later date, but these are enhancements that should be driven and verified by ASIC library modelers and Verilog tool vendors.

# 10.0 Conclusions

In the first revision of this paper, I found it disappointing to find that there was no Verilog-2001 subset that was reliably supported by all vendors. In revision 1.2, I found that ANSI-style module ports and both comma-separated and @* sensitivity lists are now supported by the simulation vendors I tested. If your company uses a certain subset of the tools shown in this paper, you should be able to identify Verilog-2001 features that are fairly safe to use now.

One thing that vendors have told me time and time again, is that they prioritize their development efforts based on their users feedback. After looking at these tables, perhaps you and your company can

do some serious pounding on your favorite vendor to get certain important features implemented by all of your chosen vendors.

Make your requests known. The more you pester your vendor the quicker the vendor will dedicate resources to the request. If your vendor says they are going to support the new Accellera SystemVerilog enhancements, ask them when they intend to support the Verilog-2001 enhancements!

## 11.0 Acknowledgements and Apologies

I would like to thank the many people from engineering, marketing and sales at Synopsys, Model Technology, Cadence, Simucad and Mentor for lending their simulation tools to make this evaluation possible.

## 12.0 About Cadence's Verilog-XL

For rev1.1 of this paper, I tested and reported compliance data related to Verilog-XL, version 4.2 beta. I was surprised to discover that Verilog-XL failed almost every single Verilog-2001 compliance test. Concerned that I would be reporting this situation to hundreds of engineers, I asked Cadence for a comment. Michael Munsey[7], NC-Verilog Product Marketing Manager for Cadence emailed that Cadence has,

> "no plans either now or in the future to enhance Verilog-XL *with Verilog-2001 extensions*. There are no plans to end of life it either ... we still actively support it. Verilog-XL is still used by a majority of *Cadence* customers for legacy designs and gate level regression runs. Our customers who require Verilog 2001 features are all NC-Verilog customers." (Italicized text added).

Since Cadence has no plans to add Verilog-2001 features to Verilog-XL, Verilog-XL was removed from the tables in this paper and will not be tested again.

Conclusion - do not use Verilog-XL for any new design work. Either use NC-Verilog or another Verilog-2001 compliant Verilog simulator.

## Revision 1.2 (April 2003) - What Changed?

For the reasons stated in Section 12.0, Verilog-XL was removed from the tables.

I was chided by VCS marketing for not using the latest version of the VCS simulator in the original paper. Feeling bad, I went to the Synopsys Electronic Software Transfer site and discovered that the latest version was not listed (now I didn't feel so bad!) VCS 7.0 was released but had not been added to the Synopsys EST web site (that situation has been corrected); nevertheless, I did acquire VCS version 7.0 and tested it for this revision of the paper. VCS 7.0 did add support for a number of important Verilog-2001 enhancements and they are reflected in the paper.

I also was able to test Simucad's SILOS 2002.10 version, which has implemented some of the Verilog-2001 enhancements as shown in the tables.

## 13.0 References

[1] Clifford E. Cummings, "Verilog-2001 Behavioral and Synthesis Enhancements," *Delivered at HDLCON-2001 but missed publication in the Proceedings*, March 2001. Available at www.sunburst-design.com/papers

[2] Donald Thomas, and Philip Moorby, The Verilog Hardware Description Language, Fourth Edition, Kluwer Academic Publishers, 1998

[3] IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, IEEE Computer Society, IEEE, New York, NY, IEEE Std 1364-1995

[4] IEEE Standard Verilog Hardware Description Language, IEEE Computer Society, IEEE, New York, NY, IEEE Std 1364-2001,

[5] Stuart Sutherland, "Verilog 2001 - A Guide to the New Features of the Verilog Hardware Description Language," Kluwer Academic Publishers, 2002.

[6] www.chris.spear.net

[7] Michael Munsey - personal communication

## Author & Contact Information

Cliff Cummings is President of Sunburst Design, Inc., a company that specializes in Verilog, Verilog synthesis and SystemVerilog training. Mr. Cummings is an independent consultant and trainer with 21 years of ASIC, FPGA, system design and verification experience and 11 years of Verilog, synthesis and methodology training experience.

Mr. Cummings has co-authored four Verilog books: the 1995 and 2001 IEEE Verilog Standards, the 2002 IEEE Verilog RTL Synthesis Standard and the 2002 Accellera SystemVerilog Standard. Mr. Cummings is the only Verilog trainer to co-develop all four of these standards.

Mr. Cummings holds a BSEE from Brigham Young University and an MSEE from Oregon State University.

E-mail Address: cliffc@sunburst-design.com

An updated version of this paper can be downloaded from the web site:
www.sunburst-design.com/papers