The Argument for Leaving the Verilog Language Unchanged

Steve Meyer
Pragmatic C Software Corp.
220 Montgomery Street, Suite 925
San Francisco, CA 94104
(415) 296-7017

Abstract: Reasons, both functional and methodological, for the wide acceptance of the Verilog electronic hardware description language and simulation environment are presented to show why the author believes new features should not be added to the Verilog language standard. Since Verilog is able to model the generally accepted hardware elements, enhancement requests have recently been derived from the desire to apply various design methods based on formal mathematics. Those methods are falsified by means of methodological analysis to support exclusion of such features from Verilog. The paper concludes with suggestions for research projects within the Verilog simulation environment research program.

1. Introduction

During the previous half decade, Verilog has revolutionized digital hardware design. It is widely accepted and is the design method of choice for the most advanced hardware development. The purpose of this paper is to offer a possible explanation for Verilog's wide acceptance, to analyze the experimental methodology that led to Verilog's superior functionality and to argue that Verilog should be left unchanged. Any changes should be adopted only after application of the same experimental methodology used to develop the current language.

The second section of this paper discusses the hardware description advances embodied in Verilog and the development method responsible for those advances. The material in this section is derived from the authors observations of Verilog development and customer usage of the Pragmatic C into Verilog net list translators: Vcmp (from Valid Ged), Tdlver (from Ndl/Tdl) and Hiver (from Hilo) that were marketed by Cadence (Gateway at the time) starting in 1988.

A number of proposed Verilog enhancements have been made for the purpose of simplifying the application of one or another hardware design methodology. At the last Open Verilog International (OVI) user group meeting, proposals were made to add features for top down design (the very theme of last year's meeting), for synthesis (Lagnese[1992], Brown[1992]), for object oriented design and programming (Flake[1992]) and for true RTL modeling (Rich[1992]). The third section shows that these methodologies are flawed and no better than any other psychological concept found to be efficacious by an individual designer during hardware development. The addition of features required by design methodologies will not improve Verilog.

The next section argues that the current set of Verilog features is sufficient for digital hardware description and therefore any addition of new features or higher level constructs to the Verilog standard should be avoided. In cases where changes are made, a list of technical steps that should occur before adoption is suggested.

The final section considers novel uses of Verilog for research in circuit design. Verilog is an ideal test bed for experimental support and falsification of methods and theories because the important step of model environment validation has already been completed. Since this paper expresses the view that there is no current need for new Verilog features let along for new HDLs and shows that the currently popular higher level methodologies are flawed, the final section discusses methodology free experimental research projects important for continued progress in circuit design. Even though there is no current need for new HDL languages and features, that should not reduce circuit design research.

It is assumed here that analyzing methodologies is as important to the growth of electronic design knowledge as describing hardware implementations and algorithms. Methodology testing should play a part in every development project as well as every research project. The discussion uses the model of scientific experimentation based on the historical study of scientific progress (Popper[1959], Lakatos[1970], Lakatos[1978a] and Lakatos[1978b]). The terms *proof* and *refutation* as used in this paper

mean support and criticism by means of experiments or arguments.

2. Reasons for Verilog's Wide Acceptance

In order to argue against significant changes to the Verilog standard it is first necessary to describe the features that make Verilog popular and the development methodology that led to the current Verilog language.

2.1 Verilog Feature Discussion

1. **Comprehensive Capabilities**
   Verilog combines a rich collection of operators using the popular C (Kernighan[1988]) language semantics with simple building block style data objects and control constructs. This combination has proven convenient for modeling all aspects of digital hardware and embodies the coding ease provided by modern programming languages. Hardware models combining descriptions from switch to behavioral level can be built without concern for modeling level interfaces.

2. **Superior Gate Level Simulation**
   The gate level simulation capability and efficiency is competitive with the best of the specialized gate level simulators. This is important since hardware designs must finally be fabricated as gates.

3. **Avoidance of Inefficient Features**
   All Verilog features are amenable to efficient implementation. An efficient implementation from gate level simulation exists for every simulator feature and from compiler writing for every programming feature. This is important because simulation of the next generation of hardware must run on the simpler and slower previous generation computers. Every Verilog feature follows the principle that no construct should invoke long unexpected computations.

4. **Avoidance of methodological Assumptions**
   Features that embody methodological assumptions have been avoided. This makes Verilog easier to learn and allows more rapid model coding than VHDL (VHDL[1987]) because VHDL requires first learning and then applying the abstract object oriented data model. In general, the simpler and more flexible the feature the better since features requiring a medium level of complexity can be built by combining primitives without requiring the most general form of the feature to be built into the language. HDLs that embody methodological assumptions become obsolete if the embodied methodology is falsified.

5. **Programming Language Interface**
   A programming language interface is included because no language regardless of how perfectly designed can anticipate the requirements of every application.

2.2 Verilog Development Methodology

The most important reason for Verilog's wide adoption is the development method used to make enhancements. This pattern can best be illustrated by considering the evolution of particular features. In late 1986 Verilog did not yet support vectors with independently changeable individual bits (only *vectored* wires). Vectors behaved as RTL registers. The impetus for change came from user enhancement requests. It was impossible to model connections expressed in terms of schematic drawing plumbing bodies in Verilog. I believe Cadence (Gateway at the time) then went through trial implementations ranging from treating vectors internally as scalars (i.e. vectors were effectively removed by pre and post processors) to experimental implementations that left vectors as monolithic objects but allowed some independence of individual bits (Sandler[1989]). Some versions were sent to beta evaluation customers so that any encountered problems could be used to make improvements. The final result provides both *vectored* and *scalared* multi-bit wires. This allows the larger overhead of per bit scheduling to be avoided if possible. The solution enables the simulator to select the proper implementation from vector usage while allowing the user to force a particular implementation by means of declaration keywords (OVI[1991], 3.5, 3.6, 12.15-12.17). The only minor restriction to allow efficient implementation requires one bit objects (scalar wires or bit selects) in places where the semantics requires one bit. This pattern allows a rich assortment of features to be provided to satisfy user design problems while allowing the implementors to preserve language consistency, ease of implementation and efficient execution. This development method represents the experimental scientific method at it best (Popper[1959], Lakatos[1970]).

The development of procedural timing control (OVI[1991], 8.25-8.33) shows a slightly different pattern. The ability to trigger procedural execution from continuously occurring changes of gates and continuous assignments is one of Verilog's best features and was available in early releases. Here the the only missing capability was the ability to trigger after the right hand side of a procedural assignment is evaluated but before executing the assignment. Since intra-assignment timing control could use the already available event mechanism required for continuous assignments, it was added even though this feature can always be avoided by coding an assignment to a temporary variable. A number of proposals have been made to add arrays of events. but in fact the wait statement in combination with procedural assignment can be used implement such arrays. A vector register can be used to store the array of events where vector bit i corresponds to event position i. To wait for cause of event i, write:

```
wait (ev[i] == 1)
 begin
  ev[i] = 0;
  <execute event code>
 end
```

The cause statement is replaced by a procedural assignment (ev[i] = 1;). Events were intended to model hard wired connections such as reset buttons.

3. Refutation of Formalist Design Methods

Many proposed Verilog enhancements are no longer the result of an inability to model a particular hardware construct but rather have the purpose of simplifying the application of some particular design methodology. The methodology almost always applies formal mathematics to circuit design. Their intention is to remove concrete human understanding of the low level functioning of circuits from hardware design. Any such method can be refuted using the following economic argument: Imagine you are constrained to applying formal methods (coding only in abstract behavioral terms) but your competitor is not. Since your competitor can optimize special cases but you can not, your device will be larger and more complicated and therefore economically uncompetitive. Another form of this argument that applies to synthesis assumes a company that uses synthesis would be consistent and use computers for other corporate functions. Therefore, instead of hiring lawyers, a legal document preparation program would be used, instead of going to a medical doctor, employee illness would be treated by a program, a program would prepare marketing plans, etc. Such a company would not remain in business for very long.

A second general argument against any such method is based on the uneven development of scientific theories. Formal mathematics such as recursive function theory was seen as the basis of artificial intelligence (AI) in the 1950s. AI was to replace all human knowledge with formal computation. That research program has failed and in fact has been degenerating at least since the 1966 National Academy of Science report discrediting natural language translation ([NAS[1966]). But in spite of the original theory's falsification, that falsification has not yet reached the circuit design area and for that matter computer programming (for example Fetzer[1988], Lakatos[1978a]).

The methodologies from last year's conference for which additional Verilog features are needed to simplify their application suffer from the following more specific problems.[1]

---

1. One possible reason for the popularity of these flawed theories may be that circuit design research funding has been provided by the defense department. Intuitionist methods that were widely accepted well into the 1970s were replaced by the current formalist methods without falsification of the previous methods or proof of the value of new methods. The following funding pattern contributed to this failure of the scientific method. A research area would be "sold" to probably the Advanced Research Projects Agency (Arpa). The formalist theory would be portrayed as completely revolutionizing some area. Large grants would be budgeted and awarded. Since a mission oriented agency could hardly be expected to allocate half of the budget to development of the new area and half to falsification, the new theory is never tested. Imagine the likelihood that an academic researcher be awarded a grant to falsify object oriented programming or synthesis even now. Hopefully now that the cold war has ended, research grants in the circuit design area will be moved to traditional scientific granting agencies who will have the mandate to award grants for both proofs and refutations.

1. **Top Down Design**
   Obviously it is helpful for designers to plan ahead and think top down but if top down design is used exclusively, it suffers from a number of problems. Top down design requires early design component (ASIC) partitioning, but such partitioning needs to be changed after completion of the various gate level net lists. Many times during detailed circuit element interconnection, ideas arise that then can be applied everywhere in a design (such as a more efficient latch macro) but this usually invalidates the previous top down design effort. Finally, one skill of experienced designers is the ability to work toward intuitive patterns that have been shown by experience to produce good results.

2. **Synthesis**
   Synthesis is based on the seemingly reasonable idea that eliminating clerical labor by means of computer programs improves circuit design. The problem is that even though high level algorithms are part of electronic design, using a computer program to produce a net list from such a high level design remains an unsolved problem. Since it is no easier than the general AI problem, the quality of such net lists will be much below those designed by skilled designers. Verilog already allows rapid coding of various complicated net list connection patterns through vectored wires or through translation from schematics into Verilog.

   I had the opportunity to observe the design effort using Verilog of one of the projects often cited as support for synthesis (Miranker[1989]) because an early version of Vcmp was being debugged during the design of the first Ardent vector processor. My observation was that each designer used a different method ranging from coding something that worked the first time since the designer had previously designed similar ASICs, to multiple net list drafts with much Verilog debugging, to designs that were finished quickly but then required considerable rework even during final system simulation. In spite of having a book documenting the design, most of the detailed design decisions were made during the by hand net list coding phase. Later improved versions of the same Ardent design were re-engineered using synthesis to go from behavioral descriptions taken from the behavior of the running hand coded implementation to gate level net lists in a more advanced technology. This is the reason for the paper's support of synthesis, but it seems to me this is a recipe for uncompetitive hardware.

3. **True Register Transfer Coding**
   True register transfer level coding as opposed to opportunistic coding of register transfers using the full power of a general purpose programming language is an expression of the view that something more closely related to mathematical formalism is better. This seems to me to be obviously wrong, but, of course, if true register transfer level coding could be shown through experimental evidence to be an improvement, features to support it should be added to Verilog.

4. Why Verilog Should not be Changed

Verilog should not be changed because it already facilitates modeling of generally accepted types of digital hardware. The main reason to leave Verilog unchanged is that all the generally accepted types of digital hardware can be modeled. There are experimental circuit modeling areas that are beyond the scope of Verilog, but the Verilog language language standard should not be the place to develop or validate those experimental methods. Hopefully, experimental tools that use the Verilog language syntax will be built to develop such new methods. It is important that the impetus for new features either come from the experiences of designers of hardware manufactured in high volume or are implemented as enhancements by Verilog simulator vendors since both sources must satisfy a general design audience. The main reason to enhance Verilog would be caused by changes in digital electronics that require new modeling capabilities.

No feature should be added to the Verilog standard without first being implemented in an experimental tool and no feature should be added that effects the efficiency of simulation (i.e. negatively effects the efficiency of other features). Finally no feature should be added for which a general consensus has not been reached. I believe the current Verilog modeling standards should be the most rigidly preserved part of the standard. The modeling conventions are: limitation to inertial delays, 4 logic levels, 8 strength levels, plus some delay selection conventions. This will assure efficient Verilog implementations.

5. Proposed Use of Verilog for Circuit Design Research

Even though this paper argues the Verilog standard should not be an area for experimental research, circuit design research still has large value even for industrial concerns.[2] Since Verilog has already been validated by successful industrial use, it offers a number of opportunities for basic research in the circuit design area. According to the view expressed here such research should either develop or criticize new theories or designs. Some possible research projects within the Verilog research program are:

1. Computers using coarse grain parallelism are now quite popular. Verilog's fine grain parallelism can be applied to test and measure the efficacy of coarse grained parallelism and possibly lead to the development of improved parallel interconnection schemes. I believe current interconnection schemes are too often derived from studying the theoretical algorithm analysis efficiency proof (i.e. proof analysis generated interconnection schemes) rather than schemes derived from experimental study.

   The various synchronization problems with such parallelism may be related to the high amount of information flow between behavioral and declarative Verilog simulator parts. It would also be interesting if the various theoretical models of computation could be modeled and the number of events required for element coordination and shared data could be measured.[3]

2. A number of new random access memory organizations have been proposed, Verilog provides the ideal environment for measuring and developing such new hardware architectures without a need to build physical hardware.

3. Verilog allows such detailed monitoring of circuit behavior that it can be used to improve or falsify the methodology of hardware performance evaluation.

4. Finally it would be valuable to see measurements and tests of circuit properties and new simulation algorithms, or possibly ideas pertaining to one particular aspect of simulation, within the the Verilog environment. I believe the event/delay control interface between procedural and declarative simulation is crucial to an efficient Verilog implementation. It would be interesting to have this conjectured proven or refuted since this research should lead to improved Verilog implementations.

6. References

Brown[1992] [... get authors and pages from 1992 OVI proceedings?} Open Verilog International Users Group Meeting Proceedings. March 1992, ??-??.

Fetzer[1988] Fetzer, J. Program verification: the very idea. *CACM.* 31, 9 (September 1988) 1048-1063.

Flake[1992] Flake, P., Moorby, P., and Brady, L. Messages - another Level of abstraction. Open Verilog International Users Group Meeting Proceedings. March 1992, 61-65.

Kernighan[1978] Kernighan, B., and Ritchie, D. *The C Programming Language.* Second Edition, Prentice Hall, Englewood Cliffs NJ, 1988.

Kuhn[1962] Kuhn, T. S. *The Structure of Scientific Revolutions.* Princeton University Press, 1962.

Lagnese[1992] Lagnese, E., and Thomas, D. Defining scheduled behaviors for high-level synthesis using Verilog. Open Verilog International Users Group Meeting Proceedings. March 1992, 111-116.

Lakatos[1970] Lakatos, I. The Methodology of Scientific Research Programs. In Lakatos, I. and Musgrave, A. (Eds). *Criticism and the Growth of Knowledge.* Cambridge, 1970, 91-196.

Lakatos[1976] Lakatos, I. *Proofs and Refutations, The Logic of Mathematical Discovery* Cambridge, 1976.

Lakatos[1978a] Lakatos, I. *The Methodology of Scientific Research Programmes.* philosophical

---

2. Miller[1992] presents the argument for 'useless' basic research in general.

3. Event volume here is probably the wrong measure, but the debate involved in determining better measures is exactly the type of scientific problem solving that has proven crucial for theoretical discoveries (Kuhn[1962]). See Stone[1987?] for a refutation of a parallel data base scheme.

papers volume 1, Cambridge, 1978.

Lakatos[1978b] Lakatos, I. *Mathematics, science, and epistemology.* philosophical papers Volume 2, Cambridge, 1978.

Miller[1992] Miller, M. The value of 'useless' research. *Wall Street Journal*,, September 21, 1992, p. A10.

Miranker[1989] Miranker, G., Rubenstein, J., and Sanguinetti, J. Getting it right the first time: the Ardent design methodology. IEEE Compcon 89 Proceedings, March 1989.

NAS[1966] National Academy of Science, [... Report on Nature Language Translation?], 1966.

OVI[1992] Verilog Hardware Description Language Reference Manual, Version 1.0, Open Verilog International, 1991.

Popper[1959] Popper, K. *The Logic of Scientific Discovery.* London, Hutchinson, 1959.

Rich[1992] Rich, D. New RTL modeling constructs in Verilog. Open Verilog International Users Group Meeting Proceedings. March 1992, 67-72.

Sandler[1989] Sandler, S. Private Communication, 1989.

Stone[1987?] Stone, H. [... Computer article on parallel data base practical analysis?].

VHDL[1987] IEEE. *IEEE Standard VHDL Language Reference Manual.* Std-1076, 1987.