HDL+ Committee Meeting
June 2002

Vassilios Gerousis
Verilog++ Committee Chairman + 
Accellera Technical Chairman
Infineon Technologies
Agenda

1- Introduction --- 9:00 - 9:30 AM.
2- Accellera Verilog++ Formal Issue List.
3- List Of Issues Generated By Accellera Board Members:
   a- Cadence.
4- Assertions (OVL) Plans -- 10:00 - 10:30. -- David Lacey
5- SystemVerilog 3.1 Plans -- 10:30 - 11:00.
6- Proposed enhancements and related proposals: two hours.
   a) Testbench features
   b) Extended API
   c) C interface
   d) Unified Assertions  will start at 2:00 pm
7- Planning, milestone development, scheduling 3.1, etc. 3:00 PM - 4:00 PM
Facts

• Verilog++ and Assertions (HDL+) started one year ago to define new direction for architectural design and assertion-based verification methodology. Thanks to innovative companies like Co-design, Verplex and Real Intent.

• An excellent team of world-wide known experts assembled within HDL+ committee and its two sister subcommittees.

• We have worked as an excellent team.

• We have analyzed the donations, modified, and executed to every milestones we developed. On Time.
Outcome

- We have produced the best LRM for SystemVerilog 3.0 standard.
  - SystemVerilog is here to stay.
- The committee made the decision to postpone certain topics for 3.1 discussions.
- The technical chairs have made the decision to coordinate our activities with the help of Dennis Brophy.
  - The scope of every committee was defined, committee approved and coordinated by close cooperation of all chairs.
  - The scope of every committee was approved by the Accellera board.
The Results

• The best LRM that OVI/Accellera has generated.
• The teamwork is surpassed any other team I have worked with for the last 12 years of standardization.
• The efforts to drive these activities by the user community is unparalleled.
• The support by EDA industry is excellent.
• SystemVerilog is born to drive technology forward for system architects, algorithmic designers and verification engineers.
• The support by the team and Accellera board illustrates this.
Moving Forward

• As more people join, we keep the same spirit of cooperation and teamwork still remains.

• This activity has been open and will be open to everyone.
  • From Day One: The scope was agreed by the Accellera board and more importantly by the large membership of the Verilog++ and Assertion companies.
  • To limit or slow its growth is not welcomed by me.
SystemVerilog for SOC Design

Language Evolution

SystemVerilog 2002: Communication interfaces, Dynamic processes/Pipeline, Packed arrays/Struct, enum deftype, casting, string types, globals, break, continue, return, data types, OVL, procedural assertions.

IEEE 2001 Verilog Standards

IEEE 1995 Verilog Standards

Design/Verification Space

Design Space:
System Architecture
SOC Platforms
Algorithms,
Behavior
IP Reuse

Verification Space:
Assertion
Extended Testbench
SOC Platform Verification
OVL
Verification Reuse
Changes to voting structure

• Instead of individual voting, we will conduct company voting.

• Voting rules will apply to THE designated company representative.
  • Each company must designate a person for voting. No alternate is allowed unless the person leaves the company.

• Each one of SystemVerilog 3.0 IEEE members, who is not an Accellera member or have a commercial affiliation, will have an individual vote.
  • Proxy can vote.

• We make decisions with the best inputs at the time.
  • We commit to track issues even after the vote
I understand politics

• I am here to limit politics and keep technical focus
  • If you come to limit the strength and to slow the progress, you came to the wrong committee.
  • If you plan to increase the scope of the committee and/or improve the language usability you are welcome.

• Teamwork must be maintained.
  • You help your fellow members.
  • Encourage cooperation.
  • Build proposal, analyze based on technical facts and not opinions.
  • Treat everyone as an expert and an equal.
  • Do not come with competing proposals.
Committee Issue List

a. Deprecation follow on: Open.

b. Time precision and timescale in general: Time Expression. Basic

c. Data Channels: Kevin -- (System -- Interfaces)


e. Force / Release extensions for strength etc -- Kevin -- mixed signal

f. State Machines: Deferred from SystemVerilog 3.0 -- Basic

g. Extern modules: Kevin -- basic

h. Object Orientation: System and Verification -- Vassilios.


j. Interfacing to "foreign" languages - e.g. VHDL and C/C++: Vassilios/System/Basic/Verification

k. Alias capability: Stephen -- Basic

l. Inferred Declarations (Types): David. Basic

m. Hierarchical and multi-clock FSMs: Basic - associated with f.

n. Dynamic process naming and control -- kevin -- System.

o. API/PLI/C-interface: Linked to J.

p. Temporal Logic: Assertion -- Jayant

Q. Implicit Reg: Alternative to declaring (or not) one bit regs, regs in general, etc. Basic. Cliff

r. DSM issues: Dennis -- ASIC -- negative timing / Basic.
Cadence List

• General - The current specification is ambiguous (Steve/Cadence)
  • needs to be more complete, or needs a reference implementation
• General - Backward compatibility problems (Steve/Cadence)
  • many new keywords will be an issue as these features are merged into IEEE 1364 Verilog
• General - Need to specify PLI extensions to access new constructs (Steve/Cadence)
• Section 2 - Issues with Literals (Steve/Cadence) -- Basic
  • width/signedness of an unsized literal without a base specifier?
  • legal constructs within an array or structure literal?
  • legal use of array or structure literals?
• Section 3 - Issues with new data types and keywords (Steve/Cadence) - Basic
  • actual utility of char, shortint, longint, byte, shortreal
  • non-orthogonality of definitions
  • inconsistent with C definitions
  • void type - is it necessary?
Number 2 (Cadence)

- Section 3 - Data packing issue (Kevin/NSC) - covered.
  - it is impossible to implement "union" from the current LRM description
  - there are many ways to do it which are not compatible
  - encoding of logic types is a factor, and "big-endian" vs. "little-endian"
  - unions should have either all logic or all bit as the base type of all elements
  - if packing is defined then 'packed' union syntax is redundant
  - may be desirable to state the packing/alignment explicitly for software compatibility

- Section 3 - Type use before definition (Steve, Paul/Cadence) - Basic
  - forces type checking to be post-elaboration
  - cause unnecessary complication of analysis, particularly separate analysis
  - useful only with pointer types

- Section 3.1 - Parameterized data types (Stuart/Cadence) - Basic
  - Elaboration issues --
  - nice, but difficult to use because they cannot be resolved until elaboration
  - (can we improve this to better support separate compilation?).
List 3

Section 3.4.1 - Issues with Time data type (Steve/Cadence) - Covered b.
- Need to detail the rules for mixed expressions, scaling, etc.

Section 3.6 - Implications of Enum type I/O (Steve/Cadence) -- Basic.
- Need to detail what is expected of Verilog I/O routines to support this

Section 3.7 - Definition of "masked" and "unmasked" (Steve/Cadence) - Ed.
- Apparently not defined?

Section 3.7 - Size requirement(?) on members of a packed union (Steve/Cadence)
- Should say "must be the same size", not "are the same size" (?) Ed.

Section 3.7 - Passing large structs/arrays (Stuart/Cadence) -- System, Basic
- Can this be done by reference instead of value (which would be inefficient)?

Section 3.8 - Conversion of short reals to 32 bits (Steve/Cadence) - Basic
- "Bit pattern is preserved" is inconsistent with other conversions.
- Should use $realtobits if the intent is to transfer the bit pattern

Section 4.2 - Packed array of signed (Erich/Cadence) -- Basic.
- Conflict between signed elements and signed whole array
List 4

Section 5.3  Constant expression (Paul/Cadence) -- Basic
- need to define precisely what can/cannot appear in a constant expression

Section 6.1  Attribute syntax (Paul/Cadence) -- Ed. -- Action for Paul./ 1364
- should factor syntax to improve readability

Section 9 - Process execution efficiency when calling C (Kevin/NSC) -- C Interfaces -
- LRM doesn't say much about calling C from Verilog processes
- should require that C functions can't suspend

Section 9.1 - Interleaving of execution (Stuart/Cadence) -- Basic. Linked to one below.
- allowing arbitrary interruption is error-prone
- should only allow interruption at synchronization points

Related - Verilog 2001 - Scheduling Algorithm (Shalom/Motorola)
- allows interleaving of processes that need to be atomic
- conflicts with requirement that non-blocking assignments execute in order of appearance
List 5

- potential problem with scheduling of PLI calls
Section 9.1 - Issues with dynamic processes (Stuart/Cadence) -- *Basic/Verification/System -- Same as N*
- need the ability to suspend/resume/abort child processes
- need process handles to support this.

---- Enhanced Interface ----
Section 13 - Interfaces vs. Modules (Stuart/Cadence) -- *Interface/Basic*
- interfaces and modules are almost the same
- should make them so and simplify definition

Section 13.1 - Interfaces restrictions (Stuart/Cadence) -- *Basic*
- should allow an interface to contain other modules
- would allow wrapping of a module with an interface
Section 13.1 - Scheduling issues (Stuart/Cadence)
- interfaces allow non-deterministic behavior due to scheduling order
- need ability to control scheduling order
Section 13.2.3 - Interface usage issues (Stuart/Cadence) --
- need to be able to specify and enforce rules about interface usage
- e.g., use of either Read or Write operation but not both
- e.g., limits on number of modules/processes invoking a given interface operation
- should be possible to define such rules in the interface itself without changing other code
- need to check rules in a way that allows for separate compilation

Section 13.4 - Modports issues (Stuart/Cadence)
- allow modports to be declared outside of interface/module, for reuse
- allow (modules and) interfaces to specify which modport(s) they implement
- import/export is confusing and unnecessary

Section 13.5.4 - Issue with extern forkjoin task (Stuart/Cadence) --
  Basic/Interface
- not necessary (at least for the example)
- unnecessarily inefficient - there are better methods

need to be able to specify that some tasks/members of an interface are private
/ System
List 7

- Clarifies the semantics (synthesis and Verification) for auto increment = Basic (Karen)
- Items from Committee and Cadence are merged together. All issues are now put under three four categories:
  - a- Basic Issues = Now belong to SV-BC (SystemVerilog Basic Committee).
  - B- Enhancement = Enhancement List Plus specific Synopsys donations are put under this committee (SV-EC).
  - C- C/C++ Committee : Synopsys C/C++ interface donations will be discussed in this committee (SV-CC).
  - D- Assertion: Assertion enhancement and also Synopsys Assertion donation will be discussed in this committee
Proposal of Discussions

• Synchronization between Sugar and SystemVerilog

• Topics related to coordination can be sent to me and I will get my fellow chairs to coordinate.
  • Proposal One: Sugar is not a standard and any changes should be determined on technical merits, Verilog-related and existing standard.
  • Proposal two: SystemVerilog 3.0 is a standard. Changes will be pushed to formal committee.

• Rule One: No discussion to limit the Scope.

• Rule Two: Discussion will center on committee approved list.
  • We will discuss Cadence technical list, and vote on each item for consideration.
System 3.1 Targets

• Cleaning 3.0 by feedback from implementation.
  • A high priority on issues raised by actual implementation rather than opinions.

• Extensions:
  • Committee approved issues list.
  • Members proposal / donation to Extend SystemVerilog.

• Target release is December 2002.
  • Extensions may move it out.
Organizational Structure

• Assertion Improvement and Issue will be discussed by the assertion committee.
  • Coordination with Sugar will be done at the TCC level first.
• Create an “issue list” subcommittee -- SV-BC
  • Prioritize The list.
  • Develop a plan how to address this list.
• Create a sub-committee to deal with C/C++. SV-CC
• Create a subcommittee to deal with SystemVerilog Language enhancement. - SV-EC
  • Pick Enhancement list into this committee
• Deprecate PLI (Stu)
Committees

• SV-BC : Chairman is Cliff Cummings.
• SV-EC : Chairman is David Smith.
• SV-CC : Stuart Sutherland.
• Assertion: David Lacey
• I will work with each chairman to get committee started:
  • Email and Web Access.
  • Initial Plans.
  • Etc.