SystemVerilog Interoperability Checklist

DVCon-2005 Paper

by Sutherland HDL, Inc., Portland, Oregon

design & verification conference & exhibition

February 14 - 16, 2005

SystemVerilog Interoperability Checklist

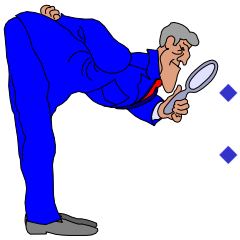
Stuart Sutherland

Sutherland HDL, Inc.

- **Update (Feb 2006): This checklist was never finished (sorry!)**
  - **The standard kept evolving to fast**
  - **As a small company, we just didn't have time to finish**
  - **Major tools have implemented so much of SystemVerilog that there is less need for this checklist**

This presentation will…

SUTHERLAND HDL
Training engineers to be HDL wizards



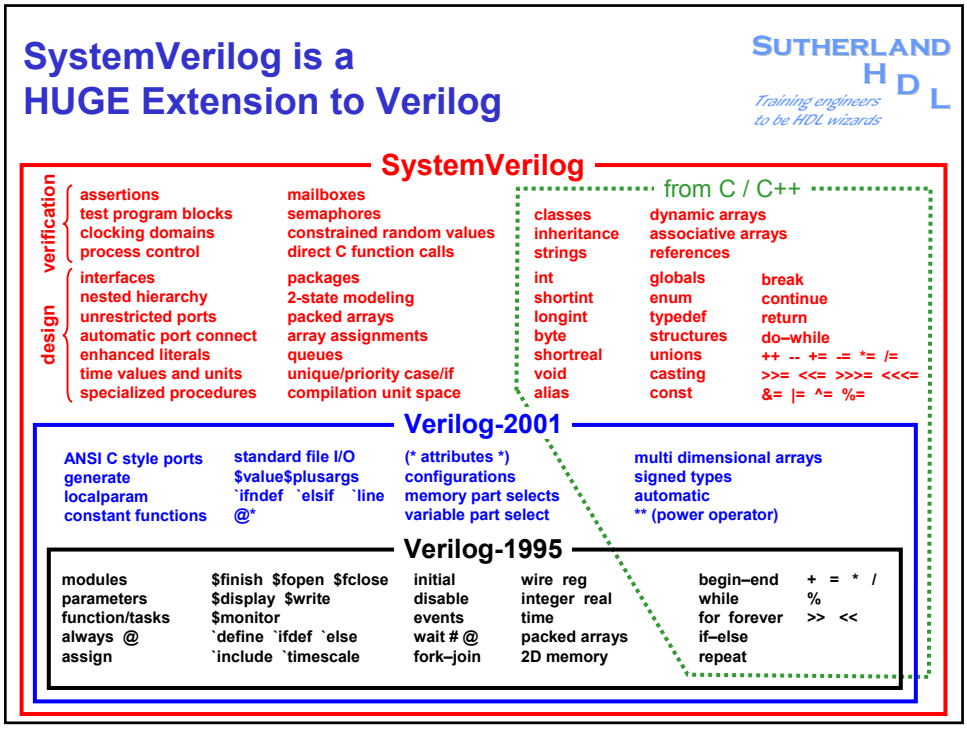
- Discuss EDA tool support for SystemVerilog
- Discuss obstacles that are preventing companies from adopting SystemVerilog
- Provide a solution to these obstacles

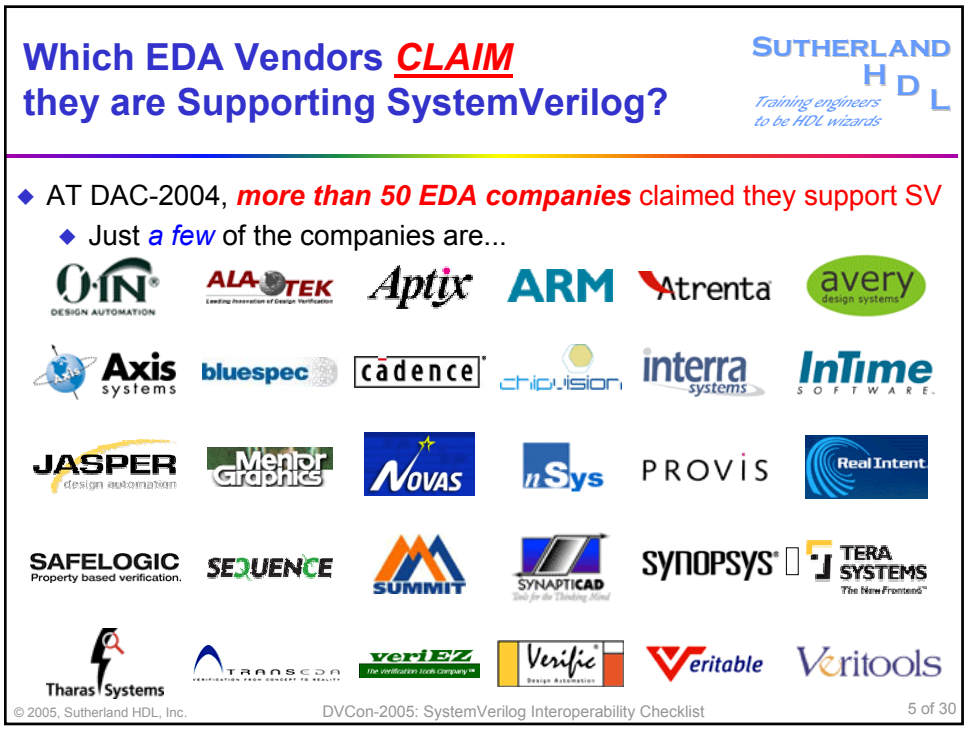
The primary goal is enable design and verification engineers to begin using SystemVerilog ASAP!

SystemVerilog Interoperability Checklist

DVCon-2005 Paper

by Sutherland HDL, Inc., Portland, Oregon





Which EDA Vendors *REALLY* Support SystemVerilog?

SUTHERLAND HDL
Training engineers to be HDL wizards

- At Sutherland HDL, we found…

EDA Marketing claims regarding SystemVerilog support are grossly exaggerated!

- The truth (as observed by Sutherland HDL) is
  - At this paper date, no product has 100% SystemVerilog support
  - Every product supports different features of SystemVerilog
  - It is not easy to find a common subset that works with all tools

Top 10 ~~Excuses~~ Reasons That No EDA Vendor Supports 100% of SystemVerilog (yet)

SUTHERLAND HDL
Training engineers to be HDL wizards

1) We will support all of SystemVerilog, but it will take time
   - The SV LRM is 616 pages — just to describe extensions to Verilog!
2) Some SystemVerilog features are not needed for some products
   - E.g., a synthesis compiler should not support testbench features
3) Some features of SystemVerilog are too hard to implement
4) We have to first support Verilog-2001 before we can support SV
5) We are waiting for the IEEE SystemVerilog standard to be approved
6) Some SystemVerilog features came from our competitor
7) We have a better way to do certain SystemVerilog features
8) Our customers are not asking for some SystemVerilog features
9) We don't think anyone will ever use certain SystemVerilog features
10) SystemVerilog is not needed; use SystemC, VHDL, *e*, …

Obstacles to Adopting SystemVerilog

SUTHERLAND HDL
Training engineers to be HDL wizards

How can I, as a user, adopt SystemVerilog, when my EDA vendors have not implemented everything in SystemVerilog?

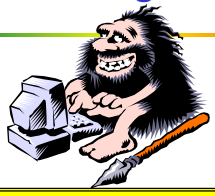
This nifty SystemVerilog feature would really make my life easier...BUT, Do all the tools in my design flow support that feature?

Tool A supports (or claims to):
- ✓ structures
- ✓ unions with real types
- ✓ importing from packages
- ✓ compilation unit declarations
- ❑ exporting tasks to interfaces
- ❑ Constrained random testing

Tool B supports (or claims to):
- ✓ structures
- ❑ unions with real types
- ❑ importing from packages
- ✓ compilation unit declarations
- ❑ exporting tasks to interfaces
- ✓ Constrained random testing

Solution 1: Don't Use SystemVerilog, Stick with Trusty Old Verilog-1995

SUTHERLAND HDL
Training engineers to be HDL wizards



John Cooley, moderator of DeepChip (www.deepchip.com) asked engineers:

Are you using SystemVerilog today?

"SystemVerilog looks promising. We like the concept. We won't use it until all of our tools in the flow support it...I don't think we will adopt it yet for a few years."

Maynard Hammond, Scientific Atlanta

How much design/verification productivity is lost by not taking advantage of what can be used in SystemVerilog today?

Solution 2:
Use a subset of SystemVerilog Today
(and a larger subset on the next project)

SUTHERLAND HDL
Training engineers to be HDL wizards

John Cooley, moderator of DeepChip (www.deepchip.com) asked engineers:

Are you using SystemVerilog today?

ANSWER:

- **1 in 5 respondents are already using some of SystemVerilog**
- **Many respondents indicated they would begin using some aspects of SystemVerilog very soon**

"Using SystemVerilog on a current project…Today we are using the [design] extensions. In the future we will be using assertions."
Don Monroe, Enterasys Networks

A competitive company might say…

We hope all of our competitors wait to benefit from SystemVerilog!

Leap Over the Obstacles!
You Do Not Need to Wait to Use SystemVerilog!

SUTHERLAND HDL
Training engineers to be HDL wizards

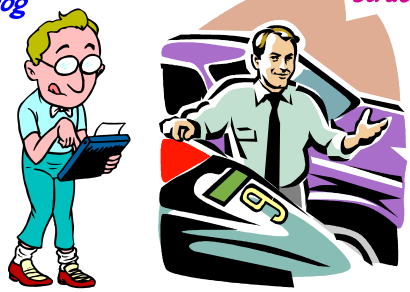
- There are good reasons NOT to wait to use SystemVerilog
  - SV can help engineers create successful designs more quickly
  - SV can eliminate many types of subtle RTL coding errors
  - SV enables verifying complex designs using a single language
- It is not necessary to wait for 100% support of SystemVerilog to benefit from SystemVerilog
  1) Identify which SV constructs would be useful for a specific project
  2) Identify what EDA tools will be used in the project
  3) Identify what in-house tools will be used in the project
  4) Determine which SV constructs from the subset that you would like to use in the project, are supported by the tools to be used
  5) Re-evaluate what features can be used for each new project (or more often, if necessary and possible)

Asking the Right Questions…

SUTHERLAND HDL
Training engineers to be HDL wizards

- If you ask a general question, you will get a general answer!

Does your XYZ tool support SystemVerilog structures?



Of course we support structures — trust me!

Is the sales rep lying if the tool supports the structure syntax, but does not yet support all of the ways a structure can be used?

Digging Down to the Nitty-Gritty Details of SystemVerilog Support

SUTHERLAND HDL

- It is not enough to ask:
- You must ask questions on how you will use each construct:
  - Are structure declarations supported in modules?
  - Are structure declarations supported in interfaces?
  - Are structure declarations supported in packages?
  - Are structure declarations supported in the compilation unit space?
  - Are typedefs of structures supported?
  - Are both packed and unpacked structure declarations supported?
  - Can all (or specific) variables types be declared within structures?
  - Can net types be declared as a structure type?
  - Can structures be passed through module ports?
  - Can structures be passed to/from tasks and functions?
  - Can structures be initialized at declaration using an expression list?
  - Can structures be initialized at declaration using a default value?
  - Can structures be assigned a list of values?
  - Can …

The Sutherland HDL SystemVerilog Interoperability Checklist

SUTHERLAND HDL
Training engineers to be HDL wizards

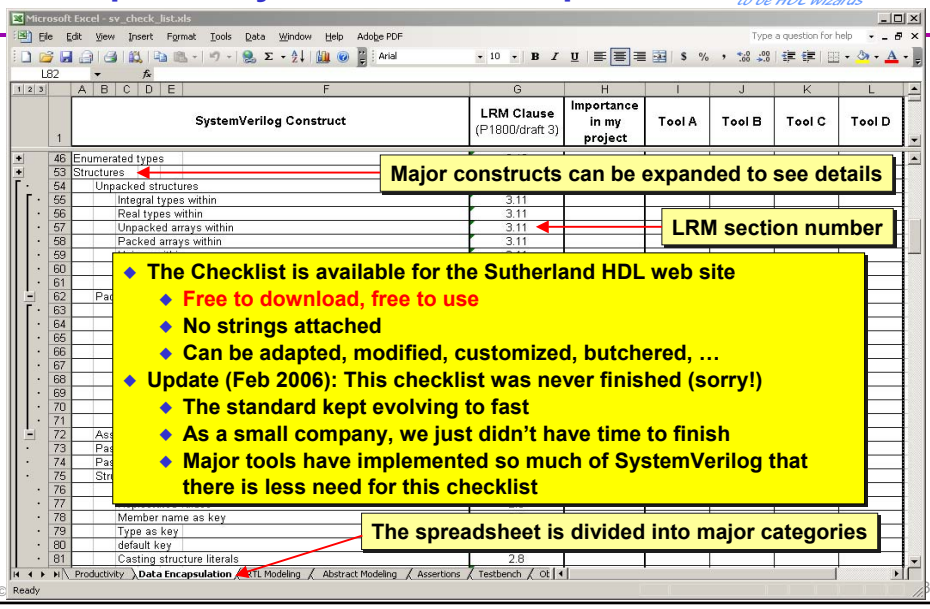
- To determine the nitty-gritty details on SystemVerilog support, at Sutherland HDL we have been developing the

SystemVerilog Interoperability Checklist

  - A list of EVERY feature in SystemVerilog
  - Each feature is broken down to detailed aspects of that feature
  - Checkboxes to fill in for:
    - What features are important for a specific project
    - What features are supported in each of several tools
  - The Checklist is in the form of a Microsoft Excel spreadsheet
    - Too large to include in the paper proceedings (8 page limit)
    - PDF would limit the Checklist usefulness

SystemVerilog Interoperability Checklist Example

SUTHERLAND HDL
Training engineers to be HDL wizards

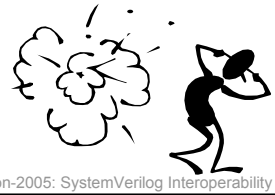


Who Should Use the SystemVerilog Interoperability Checklist

SUTHERLAND HDL
Training engineers to be HDL wizards

- Primary intent:
  - For users (design engineers and verification engineers)
    - To determine which SV features are important in a current project
    - To determine which SV features are important for a future project
    - To determine an SV subset common to the tools that can be used in a project

- Secondary intent:
  - EDA vendors might find the checklist useful as they implement SystemVerilog



SystemVerilog Interoperability Checklist Organization

SUTHERLAND HDL
Training engineers to be HDL wizards

- The Checklist divides SystemVerilog into major categories
  1) Productivity enhancements
  2) Data encapsulation enhancements
  3) RTL enhancements
  4) Abstract modeling enhancements
  5) Assertions
  6) Testbench enhancements
  7) Object-oriented verification
  8) API Enhancements

NOTE: The categories used in this Checklist in no way imply any subsets to the SystemVerilog standard, and should not be construed as such. The categories are merely for convenience in organizing a very large spreadsheet.

SystemVerilog Productivity Enhancements

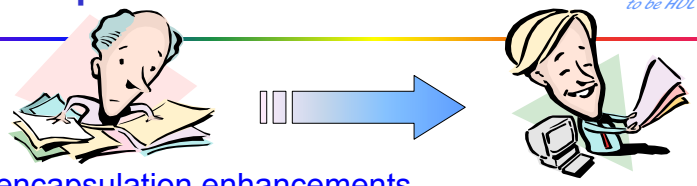
SUTHERLAND HDL
Training engineers to be HDL wizards

- Productivity enhancements (in the author's opinion):
  - Make it easier to model in Verilog
    - More functionality with fewer lines of code!
  - Eliminate common modeling errors
  - Do *not* add significant new functionality

✓Specifying time units & precision
✓Enhanced `define text substitution
✓Block names
✓Statement labels
✓Named end statements
✓Enhanced literal values
✓Replacement for "reg" keyword
✓2-state "bit" data type
✓Local for-loop variables
✓Relaxed rules for using variables
✓Relaxed rules for module ports
✓Module instantiation shortcuts
✓Task/function default argument direction
✓Passing task/function args by name
✓Passing task/function arguments by reference (pointers)
✓C-like function returns
✓Task/function implicit statement groups

SystemVerilog Data Encapsulation Enhancements

SUTHERLAND HDL
Training engineers to be HDL wizards



- Data encapsulation enhancements
  - Bundle many discrete signals or data together
  - Perform operations on bundles of signals or data

✓Interfaces
✓Packages
✓Nested modules
✓Unpacked structures
✓Packed structures
✓Unpacked unions
✓Packed unions
✓Tagged unions
✓Vectors with subfields (packed arrays)
✓Initialize arrays with list of values
✓Initialize arrays to default values
✓Assign arrays to arrays
✓Select and assign slices of arrays
✓Assign list of values to arrays
✓Passing arrays/structures/unions through ports and to tasks/functions

SystemVerilog RTL Modeling Enhancements

SUTHERLAND HDL
Training engineers to be HDL wizards

- RTL modeling enhancements
  - Make it easier to model for synthesis
  - Remove model ambiguity
  - Reduce pre- and post-simulation mismatches
  - Automatic warnings if model does not match designer's intent

✓Specialized procedural blocks for
- Combinational logic
- Sequential logic
- Latched logic

✓"unique" decision modifier
✓"priority" decision modifier
✓User-defined types
✓Enumerated types
✓Increment/decrement operators
✓Assignment operators
✓"Don't care" comparison operators
✓Void functions

SystemVerilog Abstract Modeling Enhancements

SUTHERLAND HDL
Training engineers to be HDL wizards

- Abstract modeling enhancements
  - Model more functionality with fewer lines of code
  - More like programming than register transfer level code
  - Enables higher level behavioral and bus functional modeling
  - Type compatibility with C models and SystemC models

✓C-like data types (int, longint, etc.)
✓Unsigned type modifier
✓type casting
✓vector size casting
✓signedness casting
✓"const" variables
✓Redefinable data types
✓C-like jump statements
- break
- continue
- return

✓Bottom testing do-while loop
✓Array iteration for-each loop

SystemVerilog Assertions and Coverage

SUTHERLAND HDL
Training engineers to be HDL wizards

No more doing things the hard way!

- Assertions and coverage enhancements
  - Enable verification of complex logic sequences
  - Provide control over pass/fail message generation
  - Simplifies white box and black box testing
  - Automatic reporting of verification coverage

- ✓Immediate assertions
- ✓Concurrent assertions
- ✓PSL-like property specifications
  - property blocks
  - sequence blocks
- ✓Local variables in properties
- ✓Multi-clock sequences
- ✓Assertion severity levels
- ✓Assertion control
  - disable iff
  - $assertoff/$asserton
- ✓assert/assume/cover directives
- ✓Coverage grouping and bins
- ✓Coverage reporting

SystemVerilog Testbench Enhancements

SUTHERLAND HDL
Training engineers to be HDL wizards

- Testbench enhancements
  - Make it easier to represent test programs
  - Provide special event scheduling for testing events
    - Prevent common test-to-design race conditions
  - Cycle-based test timing

- ✓Extended Verilog event scheduling
  - Preponed region
  - Observe region
  - Reactive region
- ✓Program blocks
- ✓Clocking blocks
- ✓Cycle delays (##)
- ✓Dynamic arrays
- ✓Associative arrays
- ✓String arrays
- ✓String methods
- ✓Final blocks
- ✓fork…join_any/join_none
- ✓Event data type persistence

SystemVerilog Object Oriented Verification

SUTHERLAND HDL
Training engineers to be HDL wizards

- Object-Oriented verification
  - Adds true verification language capabilities to Verilog
    - Based on the Open-VERA verification language
  - Enables advanced, high-level verification methodologies
  - Enables modularized, re-usable verification programming

✓C++ like class objects
- With Java-like automatic garbage collection

✓Object construction using "new"
✓Class inheritance
- Enables polymorphic testing

✓Public, private and protected classes
✓Built-in semaphore class objects
✓Built-in mailbox class objects
✓Constrained random value generation

SystemVerilog API Enhancements

SUTHERLAND HDL
Training engineers to be HDL wizards

- Application Programming Interface (API) enhancements
  - Increase Verilog's PLI/VPI capabilities
  - Extend Verilog's VPI to support SystemVerilog constructs
  - Simplify interacting with C/C++ using a direct interface

✓VPI extensions
✓Assertions API
✓Coverage API
✓Extended VCD files

✓Direct Programming Interface
- Import C functions into Verilog
- Export Verilog functions to C
- Export Verilog tasks to C

The Big Question: What are the Checklist Results?

SUTHERLAND HDL
Training engineers to be HDL wizards

- Cadence
  - Simulation
  - Synthesis
  - Formal
  - Hardware acceleration
- Mentor Graphics
  - Simulation
  - Synthesis
  - Formal
  - Hardware acceleration
- Synopsys
  - Simulation
  - Synthesis
  - Formal
  - Hardware acceleration

The answer is…

How Do Your Tools Measure Up?

SUTHERLAND HDL
Training engineers to be HDL wizards

- **This paper does not answer that question!**
- Why not?
  1) How big of a fool do you think I am?
  2) No vendor supports 100% of SystemVerilog today (16 Feb 2005)
  3) No design/testbench needs 100% of SystemVerilog
     - Some projects can best from one set of SV constructs
     - Other projects can best benefit for a different set of SV constructs
     - Few, if any, engineers will master everything in SV all at once
  4) There are 75+ EDA companies, which ones are your vendors?
  5) Many companies have in-house tools that parse Verilog/SystemVerilog code
- This paper gives you a tool, so YOU can answer the questions
  - The checklist is free to download, free to modify, free to use

Caveats

SUTHERLAND HDL
Training engineers to be HDL wizards

- The SystemVerilog Interoperability Checklist:
  - Developed by Sutherland HDL, Inc. for use at Sutherland HDL
    - The Checklist is detailed, but not exhaustive (too many corner cases)
    - The Checklist may have errors, but we think it is accurate
  - Checklist categories are for convenience in organization
    - The categories do not imply language subsets
    - Many constructs could have fit into other categories
  - Has no guarantees or maintenance, expressed or implied
- The Checklist is available
  - The official P1800 SystemVerilog ballot draft renumbered sections
  - The original checklist focused on design constructs
  - Sutherland HDL is in the process of adding verification constructs

Conclusion

SUTHERLAND HDL
Training engineers to be HDL wizards

- SystemVerilog is critical to successful designs
  - Enhanced modeling capabilities to handle large designs
  - Enhanced, and unified, verification capabilities to test large designs
- Much of SystemVerilog can be used TODAY!
  - Many EDA vendors have partial SystemVerilog support
  - No vendor has 100% SystemVerilog support (as February 2005)
- To benefit from SystemVerilog right away, users must:
  - Identify which portions of SystemVerilog are needed
  - Identify which portions of the SV subset are currently supported
- The SystemVerilog Interoperability Checklist is intended to help companies begin benefiting from SystemVerilog TODAY
  - Free to download and customize from www.sutherland-hdl.com

by Sutherland HDL, Inc., Portland, Oregon

Questions?

SUTHERLAND HDL
Training engineers to be HDL wizards