



design & verification
conference & exhibition

February 19 - 21, 2008

Verilog Is Not Called Verilog Anymore!

*The Merging of the Verilog and
SystemVerilog IEEE Standards*

Stuart Sutherland
Sutherland HDL, Inc.



*Training engineers
to be HDL wizards*

www.sutherland-hdl.com

About the Presenter...



- **Stuart Sutherland, a SystemVerilog wizard**
 - Independent Verilog/SystemVerilog consultant and trainer
 - Hardware design engineer with a Computer Science degree
 - Has been working with Verilog since 1988
 - **Specializing in Verilog and SystemVerilog training**
 - Member of the IEEE 1800 SystemVerilog standards group
 - Involved with the definition of SystemVerilog since its inception
 - Technical editor of SystemVerilog Language Reference Manual
 - Member of IEEE 1364 Verilog standards group since 1993
 - Past chair of the Verilog PLI task force
 - Technical editor of the IEEE 1364-1995, 1364-2001 and 1364-2005 Verilog Language Reference Manuals

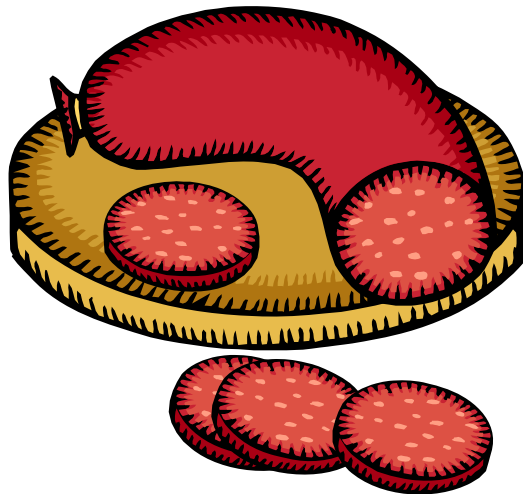
www.sutherland-hdl.com

Sausage and Standards



SUTHERLAND
training engineers to be **HDL**
Verilog and SystemVerilog wizards
www.sutherland-hdl.com

What do Sausage and EDA Standards
have in common?



Those who like sausage or EDA standards
should never watch either one be made!

First The Ugly Stuff, Then Good Stuff

This presentation will discuss...

- ❑ How we ended up with two Verilog standards
 - IEEE 1364 Verilog
 - IEEE 1800 SystemVerilog
- ❑ The merging of Verilog and SystemVerilog
 - There's a twist that might surprise you!
- ❑ Lots of new stuff in the next version of SystemVerilog
 - The standard keeps getting better...
- ❑ Summary





Verilog versus SystemVerilog

IEEE SystemVerilog-2005

verification	assertions	mailboxes	classes	dynamic arrays	
	test program blocks	semaphores	inheritance	associative arrays	
design	clocking domains	constrained random values	polymorphism	strings	
	process control	direct C function calls	data hiding	pass by references	
design	interfaces	packages	int	globals	break
	nested hierarchy	2-state modeling	shortint	enum	continue
	unrestricted ports	packed arrays	longint	typedef	return
	automatic port connect	array assignments	byte	structures	do-while
	enhanced literals	queues	shortreal	unions	++ -- += -= *= /=
	time values and units	unique/priority case/if	void	casting	>>= <<= >>>= <<<=
	specialized procedures	compilation unit space	alias	const	&= = ^= %=

IEEE Verilog-2005

uwire

``begin_keywords`

``pragma`

`$clog2`

IEEE Verilog-2001

ANSI C style ports
 generate
 localparam
 constant functions

standard file I/O
`$value$plusargs`
``ifndef `elsif `line`
`@*`

`(* attributes *)`
 configurations
 memory part selects
 variable part select

multi dimensional arrays
 signed types
 automatic
`**` (power operator)

IEEE Verilog-1995 (created in 1984)

modules	<code>\$finish \$fopen \$fclose</code>	initial	wire reg	begin-end	+ = * /
parameters	<code>\$display \$write</code>	disable	integer real	while	%
function/tasks	<code>\$monitor</code>	events	time	for forever	>> <<
always @	<code>`define `ifdef `else</code>	wait # @	packed arrays	if-else	
assign	<code>`include `timescale</code>	fork-join	2D memory	repeat	

Why Two Standards?

- SystemVerilog is an extension to Verilog, but...
- The IEEE, in it's infinite wisdom, chose to create **two standards**
 - IEEE 1364-2005 — the Verilog standard
 - IEEE 1800-2005 — the SystemVerilog standard



Why did the IEEE do this?

A Brief History

- SystemVerilog was created by Accellera, not the IEEE
 - Accellera is a “think tank” non-profit organization
 - Accellera can define a standard faster than the IEEE...in theory
- The Accellera SystemVerilog committee intended SystemVerilog to be transferred to the IEEE 1364 Verilog standards group
- **The Accellera Board of Directors blocked this transfer**
 - IEEE 1364 was under the IEEE DASC organization
 - Open voting and **large voter base** on ballots
 - Verilog 1364-2001 was approved by **hundreds of voters**
 - A large ballot pool can significantly slow down approval
 - Accellera transferred SystemVerilog to IEEE CAG organization
 - Restricted voting — **only paying corporate members can vote**
 - The SystemVerilog 1800-2005 ballot had **just 14 corporate voters**

Did Accellera's Plan Work?

■ On the positive side...

- Defining a huge set of extensions to Verilog was completed in record time
- The real work was done by subcommittees
 - Subcommittee participation and **voting remained open**
 - **Diverse representation** from both EDA vendors and end users
 - **No ugly politics** at the subcommittee level
- The end result is a usable and excellent standard!
 - Rapid adoption of SystemVerilog is the proof



■ On the negative side...

- Users don't understand that there is only one language
- Using two standards documents for one language is difficult



What's Next...

This presentation will discuss...

- ✓ How we ended up with two Verilog standards
 - IEEE 1364 Verilog
 - IEEE 1800 SystemVerilog
- **The merging of Verilog and SystemVerilog**
 - There's a twist that might surprise you!
- Lots of new stuff in the next version of SystemVerilog
 - The standard keeps getting better...
- Summary



Merging Verilog and SystemVerilog

- In January 2007, the IEEE 1800 SystemVerilog standards group approved merging the Verilog and SystemVerilog standards
 - Contentious vote
 - Some insisted that merging the standards was a waste of time
 - The vote to merge the standards barely passed
- An editor was hired to create a trial merged document
 - The merge could be cancelled if the trial document revealed that the merge would take too much effort
 - The trial merge document was created in just 3 months!
 - The result was unanimous approval to continue with the merge
- As of the DVCon-2008 conference (February 2008):
 - The merge of Verilog and SystemVerilog is complete
 - The definition of many new enhancements is nearly complete

The “Verilog” Name Is Dead!

- The IEEE 1364 Verilog base language has been merged into the IEEE 1800 SystemVerilog standard
 - The unified Verilog and SystemVerilog standard is called SystemVerilog, not Verilog!
 - The IEEE 1364 Verilog standard will soon be defunct
 - This was not the original intent!
 - Intent was for SystemVerilog extensions to be merged into Verilog
- When will the merged standard be available?
 - The trial merged document is available from the IEEE (June 2007)
 - The target is to have an **IEEE 1800-2008** SystemVerilog standard
 - Only if balloting and final approval can be completed by late 2008
 - Might slip to early 2009 if balloting starts after July 2008

What's Next...

This presentation will discuss...

- ✓ How we ended up with two Verilog standards
 - IEEE 1364 Verilog
 - IEEE 1800 SystemVerilog
- ✓ The merging of Verilog and SystemVerilog
 - There's a twist that might surprise you!
- **Lots of new stuff in the next version of SystemVerilog**
 - The standard keeps getting better...
- Summary



Following is a very brief summary of 45 new features that are being added in SV-2008

New in SystemVerilog-2008: 1 and 2

■ Defining Parameter Values Using Configurations

- Verilog can only redefine parameters from within modules
 - Reconfiguring a design requires modifying design files
- In **SV-2008**, parameters can be redefined from configurations
 - Can reconfigure a design without touching the design files

```
module adder #(parameter SIZE=32)  
(input  [SIZE-1:0] a, b,  
 output [SIZE-1:0] y);
```

```
module chip (...);  
  ...  
  adder a1 (.a, .b, .y)
```

```
module chip (...);  
  ...  
  adder (.a, .b, .y);
```

■ Default parameter assignment made optional

- Parameters can now be declared without assigning a value
 - A value must be defined when the design block is instantiated

No default value for **SIZE**

```
module adder #(parameter SIZE)  
(input  [SIZE-1:0] a, b,  
 output [SIZE-1:0] y);
```

SIZE *must* be defined
when **adder** is used

```
module chip (...);  
  ...  
  adder #(.SIZE(32)) a1 (.a, .b, .y);
```

New in SystemVerilog-2008: 3, 4, 5

- Package import in design element header
 - **SV-2008** allows packages to be imported in the module header

```
package chip_types;  
  typedef ... opcode_t;  
  typedef ... boolean_t;  
endpackage
```

The package name does not need to be repeated for every port

```
module chip  
  (output chip_types::boolean_t status,  
   input chip_types::instruction_t iw);
```

SV-2005 style

```
module chip  
  (import chip_types::*;  
   output boolean_t status,  
   input instruction_t iw);
```

SV-2008 style

- Package chaining

- A package can import definitions from another package
 - In **SV-2008**, package imports can be chained (if exported)

```
package sub_types;  
  import chip_types::boolean_t;  
  export boolean_t;  
  ...
```

```
module chip (...);  
  import sub_types::*;  
  boolean_t status_flag;  
  ...
```

Error in SV-2005

Legal in SV-2008

- Automatic packages

- Changes the default lifetime for tasks and functions within the package

New in SystemVerilog-2008: 6, 7

- Localparam definitions in ANSI-style module headers
 - SV-2008 adds localparam declarations in module headers
 - Verilog only allows parameter constants in ANSI-style headers

```
module adder
#(parameter SIZE=32, localparam MAX=64)
(output [SIZE-1:0] y, ...);
initial assert (SIZE <= MAX) else $fatal;
...
```

- Parameters can be redefined
- Local parameters cannot be redefined

- Default values for input ports

- SV-2008 allows input ports to be defined with a default value

```
module IP // original version
(inout [31:0] transform,
input      clk, reset);
...
```

```
module IP // new version
(inout [31:0] transform,
input      clk, reset,
input      preset = 1'b1);
```

Use default value
if port is left
unconnected

```
module my_design (...);
IP d1 (data, clock, rst);
```

Designs using old version of IP do not need to be
changed because IP added a new input port

New in SystemVerilog-2008: 8, 9, 10

- Bit selects and part selects of expressions
 - SV-2008 allows bit selects and part selects of expressions

```
logic [2:0] y;  
logic [7:0] a, b, c;  
logic [7:0] temp;  
temp = {a & b | c};  
y = temp[4:2];
```

Verilog only allows bit selects and part selects of nets and variables;
Can require extra code

```
logic [2:0] y;  
logic [7:0] a, b, c;  
y = {a & b | c}[4:2];
```

- Task to invoke operating system commands
 - SV-2008 adds a **\$system** task that calls operating system commands

```
initial  
$display("today is:"); $system(date);
```

Most Verilog already have the \$system command, but it was never officially in the Verilog standard

- Alternate syntax for timeunit

```
timeunit 1ns / 10ps;
```

 - SV-2008 can specify time unit and precision with one keyword
 - SV-2005 requires two keywords, **timeunit**, **timeprecision**

New in SystemVerilog-2008: 11, 12

- Synthesis parallel_case equivalent statement
 - SV-2005 **priority** checks that a decision is **full**
 - SV-2005 **unique** checks that a decision is both **full** and **parallel**
 - SV-2008 **unique0** checks that a decision is **parallel**

```
always_comb begin
    next_state = state;
    unique0 case (state);
        ...
    endcase
end
```

Case select items must be mutually exclusive, but case statement does not need to decode all possible values

- Functions can spawn processes
 - SV-2008 allows nonblocking & clocking-drive assigns in functions
 - The function returns immediately; the assignment completes later
 - SV-2008 allows functions to spawn parallel processes using **fork...join_none**
 - The function returns immediately; the processes continue to run

New in SystemVerilog-2008: 13, 14, 15

- Specifying severity levels with general print messages
 - SV-2008 allows the **\$fatal**, **\$error**, **\$warning**, and **\$info** severity-level tasks to be used anywhere \$display can be used
 - In SV-2005, these tasks could only be used in assertions

```
if (really_bad_error)  
    $fatal(2, "The design did a bad thing!");
```

- New print formatting specifiers
 - SV-2008 adds a **%p** format specifier which can be used to print all values contained in aggregate expressions such as unpacked structures, arrays, and unions
 - SV-2008 adds a **%x** hexadecimal format specifier (same as **%h**)
- Specifying field widths in integer print formats
 - SV-2008 extends the integer print formats to allow field widths

```
$display ("time=%10d", $time);
```

Printed value with a minimum field width of 10 characters

New in SystemVerilog-2008: 16, 17, 18

- Function to return a formatted string
 - **SV-2008** adds a **\$sformatf** system function
 - Like Verilog's **\$sformat**, except that the string result is passed back as the function return instead of as a task argument
- ```
string time_str = $sformatf("%s", $time);
```
- Add predefined macros for file and line number
    - **SV-2008** adds C-like **`\_\_FILE\_\_** and **`\_\_LINE\_\_** macros
      - Allow access to the current file and line number from within SystemVerilog code
- ```
$display("Error detected at %s, line %d", `__FILE__, `__LINE__);
```
- SDF support for timing skew
 - **SV-2008** defines SDF support for **\$timeskew** and **\$fullskew**
 - Verilog-2001 added these timing checkers, but did not define how they should be updated from SDF delay back-annotation files

New in SystemVerilog-2008: 19, 20

- Default values for macro arguments
 - SV-2008 ``define` text substitution macros can have default argument values
 - Allows macros to be called without passing values to arguments

```
`define SYNCH_FLOPS(N=2) ##[N:N+1] // N is number of synchronization flops
property pEnableReq;
  @(posedge master_clk)
  $rose(enable) |-> `SYNCH_FLOPS() $rose(req) ...
endproperty
```

If ``SYNCH_FLOPS` is not passed a value, then the default of **2** is used

- Removing text substitution macros
 - SV-2008 adds a ``undefineall` directive
 - Removes all previously defined user-defined macros
 - Does not affect built-in macros (e.g. Verilog-AMS macros)

```
`undefineall // remove any user macros from other files
module my_chip ...
```

New in SystemVerilog-2008: 21, 22, 23, 24

- Extended queue **delete()** method
 - **SV-2008** extends the **delete()** method to delete an entire queue
 - In SV-2005, only specific members of a queue can be deleted
- External methods with parameterized class types
 - **SV-2008** allows a class to use both parameters & extern methods
 - SV-2005 does not allow this combination
- Action blocks in assume property statements
 - **SV-2008** adds the ability to specify pass and fail action blocks in **assume property** statements
 - Makes **assume property** orthogonal to **assert property**
- Covergroup sample method with arguments
 - **SV-2008** allows cover groups to be used with assertions
 - In SV-2005, several lines of “glue” code were often needed

New in SystemVerilog-2008: 25, 26

- More restrictive rules for in-line static variable initialization
 - SV-2008 requires the **static** keyword in order to have in-line initialization of static variables in **task**, **function**, or **begin...end**
 - Prevents difficult-to-debug coding errors

```
always_comb begin
    static temp = 5;
    ...
end
```

“static” indicates **temp** is only initialized the very first time the **always_comb** block triggers

- Local variable initialization in assertions
 - SV-2008 allows assertion local variables to be initialized at the time of declaration

```
property pipeline;
    transaction_t pipe_in = data_in;
    @(posedge clk) en |-> ##6 data_out == pipe_in;
endproperty
```

local variable **pipe_in** is initialized when assertion thread starts

New in SystemVerilog-2008: 27, 28

- PSL-like assertion shortcut operators

- **SV-2008** adds 4 PSL-like assertion shortcut operators

```
property pReqAck;  
  @(posedge clock)  
  ena |-> ##[+] req[+] ack;  
endproperty
```

- **##[+]** is short for the operation **##[1:\$]**
- **##[*]** is short for the operation **##[0:\$]**
- **req[+]** is short for the operation **req[*1:\$]**
- **req[*]** is short for the operation **req[*0:\$]**

- Assertion logical implication and equivalence operators

- **SV-2008** adds logical **implication** and a **equivalence** operators
 - These operators can be used anywhere, not just in assertions

```
always_comb begin  
  a_implies_b = (a -> b);  
  a_equiv_b   = (a <-> b);  
end
```

The implication and equivalence operators return true or false

- **->** is short for the operation **(!a || b)**
- **<->** is short for the operation **((a -> b) && (b -> a))**

New in SystemVerilog-2008: 29, 30, 31

- Using overlapping operators in multi-clock assertions
 - **SV-2008** allows the operators **##0**, **|->** and **if...else** to be used in multi-clock environments
 - SV-2005 only allowed these operators with single-clock assertions
- New value sample function
 - **SV-2008** adds a **\$changed** value sample function
 - Returns true if an expression changed value during a clock cycle
 - Can be used in assertions and other verification code

```
assert property (counter_enable |-> ##1 $changed(count));
```

- Assertions can use values sampled on a different clock
 - **SV-2008** allows **\$rose**, **\$fell**, **\$stable** and **\$changed** sampled value functions to be specified with a different clock than the assertion

New in SystemVerilog-2008: 32, 33

- Assertion action control system tasks
 - SV-2008 adds new system tasks: **\$assertpasson**, **\$assertpassoff**, **\$assertfailon**, **\$assertfailoff**, **\$assertnonvacuouson**, **\$assertvacuousoff**
 - Gives control over execution of assertion pass and fail statements

```
a1: assert property (pReqAck)
    $info("pReqAck passed");
else
    $error("pReqAck failed");

initial $assertvacuousoff(a1);
```

By default, the "pass" action block will execute for both success and vacuous success

Turn off action block execution for vacuous successes

- Inferred assertion query functions
 - SV-2008 adds new system functions: **\$inferred_clock**, **\$inferred_disable**, **\$inferred_enable**
 - Can aid in creating generic assertion libraries

New in SystemVerilog-2008: 34, 35

- Assertion global clocking definition
 - SV-2008 adds the ability to specify a global clock definition
 - Simplifies writing assertion definitions for formal verification tools

```
global clocking @(posedge m_clock); endclocking
```

```
property @($global_clock) ... endproperty
```

- New global clock past and future value sample functions
 - SV-2008 adds:
 - `$past_gclk`, `$rose_gclk`, `$fell_gclk`, `$stable_gclk`, `$changed_gclk` —
 - Use values that occurred 1 global clock cycle in the past
 - `$future_gclk`, `$rising_gclk`, `$falling_gclk`, `$steady_gclk`, `$changing_gclk`
 - Use values that will occur 1 global clock cycle in the future

Note: Can only look 1 clock cycle back or forward

But Wait...There's More!

Several more enhancements are still being considered for SV-2008

- Additional assertions enhancements
 36. Assertion checkers — assertion library modeling construct
 37. New **let** statement — define re-usable assertion building blocks
 38. Linear Temporal Logic (LTL) operators — **##**, **next**, **until**, **always**, **eventually** ...
 39. New **reject_on**, **accept_on** operators — asynchronous aborts
 40. Deferred immediate assertions — prevent race conditions
 41. Elaboration error tasks — check conditions before simulation
 42. Concurrent assertions in loops — check each loop pass
 43. New **restrict property** statement — limits formal state space
 44. Immediate **assume** and **cover** statements
- Additional PLI enhancements
 45. VPI support for dynamic Object-Oriented verification constructs
 46. VPI support for all new SystemVerilog features

What's Next...

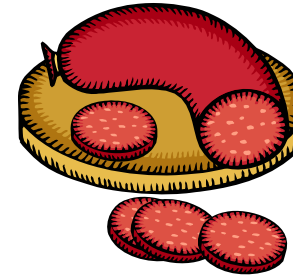
This presentation will discuss...

- ✓ How we ended up with two Verilog standards
 - IEEE 1364 Verilog
 - IEEE 1800 SystemVerilog
- ✓ The merging of Verilog and SystemVerilog
 - There's a twist that might surprise you!
- ✓ Lots of new stuff in the next version of SystemVerilog
 - The standard keeps getting better...



□ Summary

Sausage Is Delicious and Standards are Great!



- Some of the politics of IEEE standards might be ugly, but...
 - The end result is still a great standard!
- The goals of SystemVerilog have all been met
 - Model more functionality in fewer lines of code
 - Model complex, re-usable verification programs
 - Broad acceptance by EDA companies
 - Rapid adoption by Verilog design and verification engineers
- Verilog and SystemVerilog are one language, one standard
 - The official name of the single standard is SystemVerilog
 - **The Verilog name is now ancient history**
- Those who like sausage and standards should not watch either one be made, but... **I like SystemVerilog and a good Bratwurst!**

Questions?



SUTHERLAND
training engineers to be **HDL**
Verilog and SystemVerilog wizards
www.sutherland-hdl.com



**A copy of this presentation is available at
www.sutherland-hdl.com/papers**