


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009


1 of 59




SystemVerilog Is Getting Even Better!

An Update on the Proposed 2009 SystemVerilog Standard


Part 1


CLIFFORD E. CUMMINGS
SUNBURST DESIGN, INC.
CLIFFC@SUNBURST-DESIGN.COM
WWW.SUNBURST-DESIGN.COM

Presented by



STUART SUTHERLAND
SUTHERLAND HDL, INC.
STUART@SUTHERLAND-HDL.COM
WWW.SUTHERLAND-HDL.COM

sponsored by





© 2009, Sunburst Design, Inc.

2 of 59



50+ Major Enhancements in SystemVerilog-2009...

- Part 1:
 -  **Cliff Cummings** of **Sunburst Design** presents the details on the major new features in SystemVerilog-2009 that involve **hardware models** and **testbench models**
 - www.sunburst-design.com/papers/DAC2009_SystemVerilog_Update_Part1_SunburstDesign.pdf
- Part 2:
 -  **Stu Sutherland** of **Sutherland HDL** presents the details on the major new features in SystemVerilog-2009 that involve **SystemVerilog Assertions**
 - www.sutherland-hdl.com/papers/DAC2009_SystemVerilog_Update_Part2_SutherlandHDL.pdf

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

Cliff Cummings and Sunburst Design

3 of 59



- Verilog/SystemVerilog/Synthesis Trainer & Contractor
- Accellera & IEEE 1800 SystemVerilog Committees
- IEEE 1364 Verilog Standards Groups (VSG)
 - Chair of the Behavioral Task Force (Enhancements & Synthesis)
- IEEE 1364.1 Verilog Synthesis Interoperability Group
- Authored more than 40 technical papers
 - includes 17 "Best Paper" awards
- Verilog instructor for 17 years
 - Synthesis instructor for 15 years
 - Provides the absolute best Verilog and SystemVerilog training!
- Tektronix, FPS, IBM - board, FPGA & ASIC design & Test Lab
- MSEE-Oregon State Univ. / BSEE-BYU

www.sunburst-design.com/papers

SystemVerilog instructor
for 6 years

Stu is a
close 2nd !!

© 2009, Sunburst Design, Inc.

Acknowledgement & Disclaimer

4 of 59



- Acknowledgements
 - Our thanks to Shalom Bresticker and Brad Pierce

(Emails) SystemVerilog &
Verilog LRM expert

... both compiled lists
of enhancements

www.eda.org/sv-bc/hm/8983.html

- Disclaimer
 - Cliff & Stu have made every attempt to show legal
SystemVerilog-2009 examples

Not all enhanced features
can be tested at this time

No guarantees !!


© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

5 of 59


The SystemVerilog Mantis Database



- The Mantis database contains *corrections, clarifications & enhancement descriptions* for SystemVerilog-2009
- www.eda.org/svdb ←

Current errata & proposed enhancements
Login: guest
Password: guest
- Mantis Item numbers are noted on appropriate slides
- Mantis items details can be viewed in the Mantis database

EDA.org Mantis



After logging in ...

Enter Issue # ...

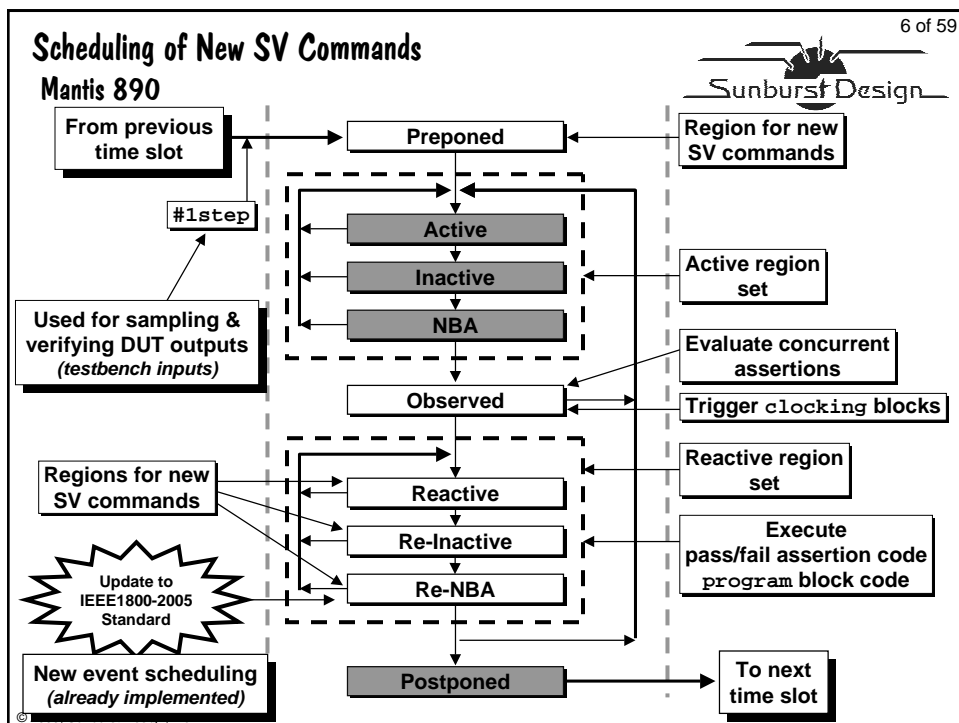
... then select the Jump button

Anonymous | [Login](#) 2009-07-17 11:39 PDT Project: SystemVerilog P1800 [Switch](#) [Exit](#)


[Main](#) | [My View](#) | [View Issues](#) | [Docs](#) | [Wiki](#)

890

Jump




7 of 59



SystemVerilog-2009 Display Enhancements

© 2009, Sunburst Design, Inc.

8 of 59



Field Widths in Print Formats

Mantis 1175

```

program print;
  int a, b;

  initial
    repeat (8) begin
      ...
      $display("a=%h b=%h", a, b);
    end
endprogram
        
```

```

randcase
  3: a = $urandom_range( 5'h10);
  2: a = $urandom_range( 9'h100);
  2: a = $urandom_range(13'h1000);
  2: a = $urandom_range(17'h10000);
endcase
randcase
  1: b = $urandom_range( 5'h10);
  2: b = $urandom_range( 9'h100);
  2: b = $urandom_range(13'h1000);
  1: b = $urandom_range(17'h10000);
endcase
        
```

Show all leading 0's

```
$display("a=%h b=%h", a, b);
```

a=000071ec	b=000000fb
a=00000003	b=00000048
a=00000ed4	b=000000f4
a=00008fbb	b=00000860
a=00000003	b=0000003a
a=000000b1	b=00004895
a=00000010	b=0000007c
a=0000097a	b=000000da

Remove leading 0's (ragged display)

```
$display("a=%0h b=%0h", a, b);
```

a=71ec	b=fb
a=3	b=48
a=ed4	b=f4
a=8fbb	b=860
a=3	b=3a
a=b1	b=4895
a=10	b=7c
a=97a	b=da

4-character field with leading 0's (orderly display)

```
$display("a=%4h b=%4h", a, b);
```


a=71ec	b=00fb
a=0003	b=0048
a=0ed4	b=00f4
a=8fbb	b=0860
a=0003	b=003a
a=00b1	b=4895
a=0010	b=007c
a=097a	b=00da

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

9 of 59



Print Format Specifier %x

Mantis 1749

%x is a synonym for %h

```
program print;
  int a, b;

  initial
    repeat (8) begin
      ...
      $display("a=%4x b=%4x", a, b);
    end
endprogram
```

```
randcase
  3: a = $urandom_range( 5'h10);
  2: a = $urandom_range( 9'h100);
  2: a = $urandom_range(13'h1000);
  2: a = $urandom_range(17'h10000);
endcase
randcase
  1: b = $urandom_range( 5'h10);
  2: b = $urandom_range( 9'h100);
  2: b = $urandom_range(13'h1000);
  1: b = $urandom_range(17'h10000);
endcase
```


Same orderly printout

a=71ec	b=00fb
a=0003	b=0048
a=0ed4	b=00f4
a=8fbb	b=0860
a=0003	b=003a
a=00b1	b=4895
a=0010	b=007c
a=097a	b=00da

%x is just "syntactic sugar"
(C-like - not really needed)

© 2009, Sunburst Design, Inc.

10 of 59



Print Format Specifier %p (%4p)

Mantis 331

```
package complex;
  typedef struct {
    logic [7:0] re;
    logic [7:0] im;
  } complex_s;
  ...
endpackage
```

```
import complex::*;

module structprint;
  complex_s a, b, sum;
  logic clk;

  initial begin
    clk <= '0;
    forever #('CYCLE/2) clk = ~clk;
  end

  cplx_adder u1 (.*);
```

Sized format specifier %4p for orderly printing

```
initial $monitor(...);
...
endmodule
```

```
module cplx_adder (
  output complex_s sum,
  input complex_s a, b);

  assign sum = add(a, b);
endmodule
```

```
$monitor("sum=%4p a=%4p b=%4p ", sum, a, b);
```


sum='{re: x, im: x}	a='{re: x, im: x}	b='{re: x, im: x}
sum='{re: 216, im: 240}	a='{re: 193, im: 148}	b='{re: 23, im: 92}
sum='{re: 158, im: 85}	a='{re: 138, im: 154}	b='{re: 20, im: 187}
sum='{re: 218, im: 240}	a='{re: 36, im: 160}	b='{re: 182, im: 80}
sum='{re: 180, im: 4}	a='{re: 108, im: 41}	b='{re: 72, im: 219}

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

11 of 59



\$sformatf Returns a Formatted String

Mantis 1589 & 1651

```
module test1;
  int      a = 32;
  logic [31:0] b = 32'haabbccdd;
  string    s;

  initial begin
    $sformat(s, "\nThe value of b = %0d'h%h\n", a, b);
    $display("%s", s);
  end
endmodule
```

SystemVerilog2005
\$sformat task assigns
string to output argument (s)

Output display

The value of b = 32'haabbccdd

```
module test2;
  int      a = 32;
  logic [31:0] b = 32'haabbccdd;
  string    s;


  initial begin
    s = $sformatf("\nThe value of b = %0d'h%h\n", a, b);
    $display("%s", s);
  end
endmodule
```

SystemVerilog2009 adds
\$sformatf function
that returns a string

Mantis 1651:
\$sprintf was rejected
(Proposed synonym for \$sformatf)

© 2009, Sunburst Design, Inc.

12 of 59



\$fatal/\$error/\$warning/\$info Display Tasks

Mantis 1641

```
module tb;
  ...

  function void check_output;
    if (cb1.y===exp) $display("PASS: y=%h exp=%h", y, exp);
    else             $display("FAIL: y=%h exp=%h", y, exp);
  endfunction
  ...
endmodule
```

In SystemVerilog2005
\$fatal / \$error / \$warning / \$info
could only be used in assertions

```
module tb;
  ...

  function void check_output;
    if (cb1.y===exp) $info("PASS: y=%h exp=%h", y, exp);
    else             $fatal("FAIL: y=%h exp=%h", y, exp);
  endfunction
  ...
endmodule
```

In SystemVerilog2009
\$fatal / \$error / \$warning / \$info
can be used anywhere \$display can be used

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

13 of 59

SystemVerilog-2009 Design Enhancements

© 2009, Sunburst Design, Inc.

14 of 59

always_ff Logic-Specific Process

- **always_ff**
 - Conveys designer's intent to infer clocked logic

Correct sensitivity list


```
module dff1 (  
    output bit_t q,  
    input  bit_t d, clk, rst_n);  
  
    always_ff @(posedge clk, negedge rst_n)  
        if (!rst_n) q <= 0;  
        else      q <= d;  
endmodule
```

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

15 of 59



Edge Event - For DDR Logic

Mantis 2396

- SV2009 adds **edge** keyword ← Equivalent to both **posedge** / **negedge**

Currently illegal syntax
for synthesis

No posedge (clk)
No negedge (clk)

```

module ddrff (
    output bit_t q,
    input  bit_t d, clk, rst_n);

    always_ff @(clk, negedge rst_n)
        if (!rst_n) q <= 0;
        else      q <= d;
endmodule
            
```

Remove posedge to permit
triggering on both edges
??


**always_ff shows
designer's intent**

always_ff @(edge clk, negedge rst_n)

Could this synthesize to a DDR flip-flop
in an ASIC vendor library ??

© 2009, Sunburst Design, Inc.

16 of 59



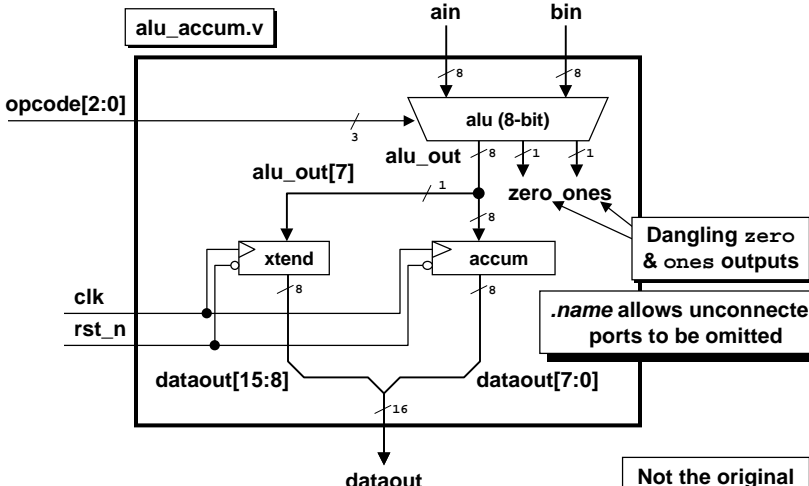
.name Instantiation & Unconnected Ports

Mantis 1660

SystemVerilog-2009
clarification

alu_accum.v

ain bin



.name allows unconnected
ports to be omitted

Not the original
intent for .name

© 2009, Sunburst Design, Inc.


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

17 of 59

.name Instantiation & Unconnected Ports

Mantis 1660



```

module alu_accum (
    output [15:0] dataout,
    input  [7:0] ain, bin,
    input  [2:0] opcode,
    input      clk, rst_n;
    wire  [7:0] alu_out;

    alu alu (.alu_out, .zero(), .one(), .ain, .bin, .opcode);
    accum accum (.dataout(dataout[7:0]), .datain(alu_out), .clk, .rst_n);
    xtend xtend (.dout(dataout[15:8]), .din(alu_out[7]), .clk, .rst_n);
endmodule
    
```

Legal (but not required):
list unconnected ports

```

...
alu alu (.alu_out, .ain, .bin, .opcode);
accum accum (.dataout(dataout[7:0]), .datain(alu_out), .clk, .rst_n);
...
    
```

Legal:
omit unconnected ports

```

...
alu alu (.ain, .bin, .opcode);
accum accum (.dataout(dataout[7:0]), .datain(alu_out), .clk, .rst_n);
...
    
```

Legal (but WRONG!):
omit design port (alu_out)


Another good reason to avoid using .name

© 2009, Sunburst Design, Inc.

18 of 59

2-to-4 Decoder w/ Enable

SystemVerilog-2005 Style



```

module dec2_4a (
    output logic [3:0] y,
    input logic [1:0] a,
    input logic      en);

    always_comb begin
        y = 0;
        unique case ({en,a})
            3'b100: y[a]=1;
            3'b101: y[a]=1;
            3'b110: y[a]=1;
            3'b111: y[a]=1;
        endcase
    end
endmodule
    
```

unique case with initial default assignment

Line	full/parallel
#	user/user

unique
Same as full_case parallel_case

parallel_case (uniqueness)

full_case
all possible cases are defined (others are "don't cares")

unique means:

- (1) case expression can only match 1 case item
- (2) case expression **MUST** match 1 case item


SystemVerilog simulators are required to give a run-time warning when en=0

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

19 of 59



2-to-4 Decoder with Enable

WRONG Synthesis Results!

Dangling enable! → en


SystemVerilog simulation should warn about this error

unique case dec2_4a

unique infers the wrong logic

© 2009, Sunburst Design, Inc.

20 of 59



2-to-4 Decoder w/ Enable

SystemVerilog-2005 Current Work-Around

```

module dec2_4b (
  output logic [3:0] y,
  input logic [1:0] a,
  input logic en);

  always_comb begin
    y = 0;
    unique case ({en,a})
      3'b100: y[a]=1;
      3'b101: y[a]=1;
      3'b110: y[a]=1;
      3'b111: y[a]=1;
      default: ;
    endcase
  end
endmodule
    
```

unique case with initial default assignment

Empty default kills full_case

Ugly, but fixes the synthesis results

Line	full/parallel
#	auto/user

unique with empty default same as parallel_case

© 2009, Sunburst Design, Inc.


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

21 of 59

Unique0 - parallel_case Equivalent

Mantis 2131



```

module dec2_4c (
  output logic [3:0] y,
  input  logic [1:0] a,
  input  logic      en);

  always_comb begin
    y = 0;
    unique0 case ({en,a})
      3'b100: y[a]=1;
      3'b101: y[a]=1;
      3'b110: y[a]=1;
      3'b111: y[a]=1;
    endcase
  end
endmodule
        
```

Simulation is correct when en=0

unique0 case with initial default assignment

Line	full/parallel
#	auto/user

unique0 same as parallel_case

parallel_case (uniqueness)

NOT full_case (others covered by initial default assignment)

unique0 means:


- (1) case expression can only match 1 case item
- (2) case expression can match 1 or 0 case items

© 2009, Sunburst Design, Inc.

22 of 59

2-to-4 Decoder with Enable

Correct Synthesis Results




SystemVerilog
 dec2_4b
 dec2_4c

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

23 of 59



priority/unique/unique0 Violations

Mantis 2008

- SV-2005: priority/unique warnings
- SV-2009: priority/unique/unique0 violations

Reported as warnings
(vendors did not want to report false-errors)

Reported as violations

Ask vendors to provide
fatal-on-violation option

unique: Only one case
item should match the
case expression

```
always_comb begin: a1
  unique case (1'b1)
    a      : z = b;
    not_a  : z = c;
  endcase
end


always_comb begin: a2
  not_a = !a;
end
```

Simulation scenario:

- (1) a goes to 1
- (2) always_comb (a1) triggers
- (3) unique case detects violation
(a & not_a both match 1'b1)
- (4) violation report generated
(scheduled into Observed Region)
- (5) always_comb (a2) triggers
- (6) not_a goes to 0
- (7) always_comb (a1) re-triggers
- (8) unique case detects NO violation
- (9) violation report cleared
(before entering Observed Region)

© 2009, Sunburst Design, Inc.

24 of 59



Default Inputs For Module/Interface Ports

Mantis 2399 (Corrects Mantis1619 Enhancement)

Old version of
register module

```
module register (
  output logic [7:0] q,
  input  logic [7:0] d,
  input  logic      clk, rst_n);

  always_ff @(posedge clk, negedge rst_n)
    if (!rst_n) q <= '0;
    else      q <= d;
endmodule
```

Default values might be useful
for infrequently used inputs

Default values only legal
with ANSI-style port list

New version of
register module
with set_n input

```
module register (
  output logic [7:0] q,
  input  logic [7:0] d,
  input  logic      clk, rst_n,
  input  logic      set_n='1 );

  always_ff @(posedge clk, negedge rst_n, negedge set_n)
    if (!rst_n) q <= '0;
    else if (!set_n) q <= '1;
    else      q <= d;
endmodule
```

New input has
a default value


CAUTION: instantiation rules
can be confusing

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

25 of 59



Default Inputs For Module/Interface Ports

Mantis 2399 (Corrects Mantis1619 Enhancement)

```
module register (  
    output logic [7:0] q,  
    input logic [7:0] d,  
    input logic      clk,  
    input logic      rst_n='1',  
    input logic      set_n='1');  
    ...  
endmodule
```

set_n and rst_n inputs have default values

```
module tb;  
    logic [7:0] q;  
    logic [7:0] d;  
    logic      clk;  
    logic      rst_n, set_n;  
  
    // register  
    ...  
endmodule
```

set_n/rst_n both declared

register instantiation

```
register r1 (.q(q), .d(d), .clk(clk), .rst_n(rst_n), .set_n(set_n));
```

tb overrides default set_n/rst_n values

```
register r1 (.q, .d, .clk, .rst_n, .set_n);
```

tb only overrides default rst_n value

```
register r1 (.q(q), .d(d), .clk(clk), .rst_n(rst_n));
```


register keeps default set_n/rst_n values

```
register r1 (.q, .d, .clk, .rst_n);
```

CAUTION: this can be confusing

© 2009, Sunburst Design, Inc.

26 of 59



Default Inputs For Module/Interface Ports

Mantis 2399 (Corrects Mantis1619 Enhancement)

```
module register (  
    output logic [7:0] q,  
    input logic [7:0] d,  
    input logic      clk,  
    input logic      rst_n='1',  
    input logic      set_n='1');  
    ...  
endmodule
```

set_n and rst_n inputs have default values

```
module tb;  
    logic [7:0] q;  
    logic [7:0] d;  
    logic      clk;  
    logic      rst_n;  
  
    assign set_n = d[7];  
    // register  
    ...  
endmodule
```

set_n NOT declared

set_n assigned

register instantiation

```
register r1 (.q(q), .d(d), .clk(clk), .rst_n(rst_n), .set_n(.set_n));
```

tb overrides default set_n/rst_n values

```
register r1 (.q(q), .d(d), .clk(clk), .rst_n(rst_n));
```

tb only overrides default rst_n value

```
register r1 (.q, .d, .clk, .rst_n, .set_n);
```

ERROR: set_n not declared in tb


```
register r1 (.q, .d, .clk, .rst_n);
```

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

27 of 59



Bit Selects & Part Selects of Expressions

Mantis 1197 Correction to DAC2009 presentation

```
module expr_range;
  logic [2:0] y;
  logic [7:0] a, b, c;
  logic [7:0] tmp;

  assign tmp = (a & b) | c;
  assign y[4:2] = tmp[4:2];

  ...
endmodule
```

```
module expr_range;
  logic [2:0] y;
  logic [7:0] a, b, c;

  assign y = {(a & b) | c}[4:2];

  ...
endmodule
```

Might require
extra code

Verilog only allows bit
and part selects of
nets and variables

SystemVerilog-2009
allows bit and part selects
on RHS concatenations


ILLEGAL - to reference a part
select on the LHS expression

NOTE: {concatenation} braces
are required to do a part select
on the result of the expression

CAUTION: more concise but
perhaps more confusing(?)

© 2009, Sunburst Design, Inc.

28 of 59



SystemVerilog-2009 Packages, Parameters, Compiler Directives

© 2009, Sunburst Design, Inc.


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

29 of 59

Package Import in Design Element Header

Mantis 329



**SystemVerilog-2005
package usage style #1**

```
package data_pkg;
  typedef logic [7:0] data_t;
  typedef logic      clk_t;
endpackage

package bit_pkg;
  typedef bit clk_t;
  typedef bit rst_t;
endpackage

module register (
  output data_pkg::data_t q,
  input  data_pkg::data_t d,
  input  data_pkg::clk_t  clk,
  input  bit_pkg::rst_t  rst_n);
  ...
endmodule
```

Declare the data_pkg and bit_pkg

bit_pkg::clk_t not used

Use the data_t and clk_t from the data_pkg

Use the rst_t from the bit_pkg


... somewhat verbose

© 2009, Sunburst Design, Inc.

30 of 59

Package Import in Design Element Header

Mantis 329



**SystemVerilog-2005
package usage style #2**

```
package data_pkg;
  typedef logic [7:0] data_t;
  typedef logic      clk_t;
endpackage

package bit_pkg;
  typedef bit clk_t;
  typedef bit rst_t;
endpackage

import data_pkg::*;
import bit_pkg::rst_t;

module register (
  output data_t q,
  input  data_t d,
  input  clk_t  clk,
  input  rst_t  rst_n);
  ...
endmodule
```

Declare the data_pkg and bit_pkg

bit_pkg::clk_t not used

Use the data_t and clk_t from the data_pkg

Use the rst_t from the bit_pkg


... packages have been imported into the \$unit/\$root space (depending upon the simulator)

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

31 of 59



Package Import in Design Element Header

Mantis 329

SystemVerilog-2009 adds import of local package declarations in module, interface and program headers

```
package data_pkg;
  typedef logic [7:0] data_t;
  typedef logic      clk_t;
endpackage

package bit_pkg;
  typedef bit clk_t;
  typedef bit rst_t;
endpackage

module register
  import bit_pkg::rst_t, data_pkg::*;
  (output data_t q,
   input  data_t d,
   input  clk_t  clk,
   input  rst_t  rst_n);
  ...
endmodule
```

Declare the data_pkg and bit_pkg

bit_pkg::clk_t not used

Packages are available to the module header


Use the data_t and clk_t from the data_pkg

Use the rst_t from the bit_pkg

... packages have been imported *locally* into the register module

© 2009, Sunburst Design, Inc.

32 of 59



Package Chaining - Export in a Package

Mantis 1323

SystemVerilog-2005 did not permit package nesting/import

```
package pkg1;
  typedef logic [2:0] src1_t;
  typedef logic [2:0] dst1_t;
  typedef logic [7:0] data_t;
  typedef struct {
    src1_t fld1;
    dst1_t fld2;
    data_t fld3;
  } pkt_t;
endpackage
```

```
package pkg2a;
  typedef logic [2:0] src1_t;
  typedef logic [2:0] dst1_t;
  typedef logic [2:0] src2_t;
  typedef logic [2:0] dst2_t;
  typedef logic [15:0] data_t;
  typedef struct packed {
    src1_t fld1;
    dst1_t fld2;
    src2_t fld3;
    dst2_t fld4;
    data_t fld5;
  } pkt_t;
endpackage
```

If used, re-declaration of src1_t and dst1_t was required

import and use the expanded pkg2a package

import pkg2a::*;


```
module register (
  output pkt_t q,
  input  pkt_t d,
  input  logic clk,
  input  logic rst_n);
  ...
endmodule
```

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

33 of 59



Package Chaining - Export in a Package

Mantis 1323

SystemVerilog-2009 allows package nesting/import/export

```
package pkg1;
  typedef logic [2:0] src1_t;
  typedef logic [2:0] dst1_t;
  typedef logic [7:0] data_t;
  typedef struct {
    src1_t fld1;
    dst1_t fld2;
    data_t fld3;
  } pkt_t;
endpackage
```

```
package pkg2;
  import pkg1::*;
  export pkg1::*;
  typedef logic [2:0] src2_t;
  typedef logic [2:0] dst2_t;
  typedef logic [15:0] data_t;
  typedef struct packed {
    src1_t fld1;
    dst1_t fld2;
    src2_t fld3;
    dst2_t fld4;
    data_t fld5;
  } pkt_t;
endpackage
```

import/export the src1_t and dst1_t types

Override the data_t and pkt_t types


import and use the expanded pkg2 package

Add the src2_t and dst2_t types

```
import pkg2::*;
module register (
  output pkt_t q,
  input  pkt_t d,
  input  logic clk,
  input  logic rst_n);
  ...
endmodule
```

© 2009, Sunburst Design, Inc.

34 of 59



Multiple Package Export

Mantis 1323

```
package pkg3a;
  typedef logic [7:0] byte_t;
  ...
endpackage

package pkg3b;
  ...
endpackage

package pkg3c;
  ...
endpackage

package pkg3d;
  ...
endpackage
```

```
package pkg4;
  import pkg3a::byte_t;
  import pkg3b::*;
  import pkg3c::*;
  import pkg3d::*;
  export *::*;
endpackage
```

import each package separately

export all packages (using wildcards)

Like doing package extension ...

... with multiple inheritance !!

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

35 of 59

Package Automatic Declarations

Mantis 1524

SystemVerilog-2005 had:

- module automatic
- program automatic
- interface automatic

SystemVerilog-2009 adds:

- package automatic

Normal package ...

```
package complex;
typedef struct {
  logic [7:0] re;
  logic [7:0] im;
} complex_s;

function complex_s add;
input complex_s a, b;
complex_s sum;
sum.re = a.re + b.re;
sum.im = a.im + b.im;
return(sum);
endfunction
endpackage
```

... static function

automatic package ...

```
package automatic complex;
typedef struct {
  logic [7:0] re;
  logic [7:0] im;
} complex_s;

function complex_s add;
input complex_s a, b;
complex_s sum;
sum.re = a.re + b.re;
sum.im = a.im + b.im;
return(sum);
endfunction
endpackage
```

... automatic function

© 2009, Sunburst Design, Inc.

36 of 59

Localparams in ANSI-Style Headers

Mantis 1134

localparam can be in the parameter port list

```
`timescale 1ns / 1ns
module ram #(parameter
  localparam DEPTH=1<<AWIDTH,
  parameter DWIDTH=8)
  (inout [DWIDTH-1:0] data,
   input [AWIDTH-1:0] addr,
   input r_wn, cs_n);

  logic [DWIDTH-1:0] mem [DEPTH];

  assign data = (!cs_n && r_wn) ? mem[addr] : 'z;

  always_latch
    if (!cs_n && !r_wn) mem[addr] <= data;
endmodule
```

Cannot override localparam

Ordered parameter replacement omits the localparam value

DEPTH not listed

ram instantiation:

```
AWIDTH=20 (1MB)
DWIDTH=16
```


ram #(20,16) ram1 (...);

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

37 of 59



Blank And Illegal Parameter Defaults

Mantis 907

NOTE: SIZE parameter not assigned a default value


```
module register #(SIZE)*(  
    output logic [SIZE-1:0] q,  
    input  logic [SIZE-1:0] d,  
    input  logic          clk, rst_n);  
  
    always_ff @(posedge clk, negedge rst_n)  
        if (!rst_n) q <= '0;  
        else      q <= d;  
endmodule
```

Parameter must be defined when the module is instantiated

```
module tb;  
    parameter SIZE = 64;  
    logic [SIZE-1:0] q;  
    logic [SIZE-1:0] d;  
    logic          clk, rst_n;  
  
    register #(SIZE) r1 (.*);  
    //...  
endmodule
```

© 2009, Sunburst Design, Inc.

38 of 59



Setting Parameters in Configurations

Mantis 2037

File: rtl.cfg

```
config rtl;  
    design tbLib.tb;  
    default liblist rtlLib;  
    instance tb use (.SZ(16));  
endconfig
```

Parameter values before config
SZ = 12
WIDTH=12 (default 4)
SIZE = 12 (default 8)

Parameter values after rtl config
SZ = 16
WIDTH=16
SIZE = 16

More config parameter capabilities have been added (not shown)

File: lib.map

```
library tbLib tb/*.sv;  
library rtlLib rtl/*.sv;
```

```
module tb;  
    parameter SZ = 12;  
    ...  
    pipeline #(WIDTH(SZ)) p1 (.*);  
    ...  
endmodule
```

```
module pipeline #(WIDTH=4) (  
    output logic [WIDTH-1:0] q,  
    input  logic [WIDTH-1:0] d,  
    input  logic          clk;  
    logic [WIDTH-1:0] p1, p2;  
  
    register #(SIZE(WIDTH)) r[1:3]  
        (.q({q,p2,p1}), .d({p2,p1,d}), .clk(clk));  
endmodule
```


```
module register #(SIZE=8) (  
    output logic [SIZE-1:0] q,  
    input  logic [SIZE-1:0] d,  
    input  logic          clk;  
  
    always_ff @(posedge clk)  
        q <= d;  
endmodule
```

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

39 of 59



Verilog-95 Macro Capabilities

```
`define CYCLE 100
...
initial begin
    clk <= '0;
    forever #(`CYCLE/2) clk = ~clk;
end
```

Define and use a macro
(every Verilog coder knows this style)

```
`define assert_clk( ... ) \
    assert property (@(posedge clk) disable iff (!rst_n) ... )
```

Line continuation character \

Define a multi-line macro
(not all Verilog coders know this)


```
`define assert_clk( arg ) \
    assert property (@(posedge clk) disable iff (!rst_n) arg )
```

Pass an argument to the macro arg

Define a multi-line macro with input arguments
(not all Verilog coders know this)

© 2009, Sunburst Design, Inc.

40 of 59



Macros with Default Arguments

Mantis 1571

- SystemVerilog-2009 permits macros to have arguments with default values

```
`define assert_clk(arg, ck=clk) \
    assert property (@(posedge ck) disable iff (!rst_n) arg)
```

By default, macro uses clk for assertion sampling

```
ERROR_q_did_not_follow_d:
    `assert_clk(q==$past(d));
```

Use default clk

```
ERROR_q_did_not_follow_d:
    `assert_clk(q==$past(d), clk2);
```


Use clk2

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

41 of 59



Verilog `define & `undef

```
`define DSIZE 8
`define ASIZE 10
`define MDEPTH 1024
module raml (
    inout  [`DSIZE-1:0] data,
    input  [`ASIZE-1:0] addr,
    input                      en, rw_n);

    logic [`DSIZE-1:0] mem [0:`MEM_DEPTH-1];

    assign data = (rw_n && en) ? mem[addr]
        : {`DSIZE{1'bz}};


    always @* begin
        if (!rw_n && en) mem[addr] <= data;
    end
endmodule
`undef DSIZE
`undef ASIZE
`undef MDEPTH
```

**`define data size,
`define address size
`define memory depth**

**`undef data size,
`undef address size
`undef memory depth**

© 2009, Sunburst Design, Inc.

42 of 59



`undefineall

Mantis 1090

```
`define DSIZE 8
`define ASIZE 10
`define MDEPTH 1024
module raml (
    inout  [`DSIZE-1:0] data,
    input  [`ASIZE-1:0] addr,
    input                      en, rw_n);

    logic [`DSIZE-1:0] mem [0:`MEM_DEPTH-1];

    assign data = (rw_n && en) ? mem[addr]
        : {`DSIZE{1'bz}};

    always_latch begin
        if (!rw_n && en) mem[addr] <= data;
    end
endmodule
`undefineall
```

**`define data size,
`define address size
`define memory depth**


**`undefineall un-defines
all `define macros**

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

43 of 59



SystemVerilog Parameterized Model

A Better Solution

Consider this:
If you are using ``undefineall`
it looks like a local constant

```
module ram1 #(parameter ASIZE=10, ←
                  DSIZE=8)
  (inout [DSIZE-1:0] data,
   input [ASIZE-1:0] addr,
   input          en, rw_n);

  localparam MEM_DEPTH = 1<<ASIZE; ←
  logic [DSIZE-1:0] mem [0:MEM_DEPTH-1];

  assign data = (rw_n && en) ? mem[addr]
    : {DSIZE{1'bz}};

  always_latch begin
    if (!rw_n && en) mem[addr] <= data;
  end
endmodule
```

Declare parameters,
local to the module


Make the memory depth
a localparam

Calculate the memory depth
from the address size

Guideline: choose parameter first
Guideline: choose ``define` only if needed

© 2009, Sunburst Design, Inc.

44 of 59




SystemVerilog-2009 Arrays & Queues Enhancements

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

45 of 59



Associative Array Clarifications

Mantis 1457

foreach command works with non-[*] associative arrays

SV2009 clarifies foreach fails with [*] assoc arrays

```
module aa2;
  typedef bit [7:0] byte_t;
  byte_t aa [int];

  initial begin
    aa[2143] = 8'hAA;
    aa_display;
    for (int i=0; i<8; i++)
      aa[i]=8'hCC;
    aa_display;
    for (int i=5; i<21; i+=5)
      aa[i]=8'hDD;
    aa_display;
  end
endmodule
```

byte_t aa [*];

1 ARRAY ELEMENTS
aa[1234]=aa

9 ARRAY ELEMENTS
aa[0]=cc
aa[1]=cc
aa[2]=cc
aa[3]=cc
aa[4]=cc
aa[5]=cc
aa[6]=cc
aa[7]=cc
aa[1234]=aa


12 ARRAY ELEMENTS
aa[0]=cc
aa[1]=cc
aa[2]=cc
aa[3]=cc
aa[4]=cc
aa[5]=dd
aa[6]=cc
aa[7]=cc
aa[10]=dd
aa[15]=dd
aa[20]=dd
aa[1234]=aa

foreach stores each aa[] index in the i variable

function void aa_display;
\$display("%0d ARRAY ELEMENTS %0d", aa.num());
foreach (aa[i]) \$display("aa[%4d]=%2", i, aa[i]);
endfunction

© 2009, Sunburst Design, Inc.

46 of 59



Associative Array Clarifications

Mantis 1723

- SystemVerilog 2005 array types & methods
 - Dynamic arrays: **.size()** method
 - Queues: **.size()** method
 - Asociative arrays: **.num()** method

Returns the size of the dynamic array

Returns the size of the queue

Returns the number of elements in the associative array

SV2009 defines .size() method - synonym for .num() method

```
function void aa_display;
  $display("%0d ARRAY ELEMENTS %0d", aa.num());
  ...
endfunction
```

```
function void aa_display;
  $display("%0d ARRAY ELEMENTS %0d", aa.size());
  ...
endfunction
```

© 2009, Sunburst Design, Inc.


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

Queue Delete

Mantis 1560

SV2009 adds a built-in method to delete an entire queue



47 of 59

```
program q_methods;
  bit [7:0] q[$] = '{8'hA0, 8'hA1, 8'hA2};
  bit [7:0] data;
  int      q_size;

  initial begin
    $display("q.size=%0d", q.size());
    q.insert(2,8'hff);
    q.delete(1);
    data=q.pop_front();
    data=q.pop_back();
    q.push_front(8'hbb);
    q.push_back(8'hcc);
    q.delete();
  end
endprogram
```

Initial values

data=8'h00

8'hA0
8'hA1
8'hA2

q.size=3

8'hA0
8'hA1
8'hFF
8'hA2

insert(@2)

8'hA0
8'hFF
8'hA2

delete(@1)

8'hFF
8'hA2

pop_front

data=8'hA0

pop_back

data=8'hA2

8'hFF

push_front

8'hBB
8'hFF

push_back


8'hBB
8'hFF
8'hCC

<empty>

No argument means to delete entire queue

New to SV2009 delete entire queue

© 2009, Sunburst Design, Inc.




48 of 59

SystemVerilog-2009

Classes & Verification Enhancements

© 2009, Sunburst Design, Inc.

49 of 59



Static Variable In-Line Initialization

Mantis 1556

```

module top;
  int svar1 = 1; ←
  initial begin
    for (int i=0; i<3; i++) begin
      automatic int loop3 = 0; ←
      for (int j=0; j<3; j++) begin
        loop3++;
        $write("%3d", loop3);
      end
    end
    $display("\n");
    for (int i=0; i<3; i++) begin
      static int loop2 = 0; ←
      for (int j=0; j<3; j++) begin
        loop2++;
        $write("%3d", loop2);
      end
    end
    $display("\n");
  end
endmodule
    
```

static keyword is optional -
initialization happens before time 0

automatic variable
assignment executes each
time the *outer* loop is called

Display values

1	2	3	1	2	3	1	2	3
---	---	---	---	---	---	---	---	---


The static variable
assignment executes
once before time 0

Display values

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

© 2009, Sunburst Design, Inc.

50 of 59



Pure Virtual Methods

Mantis 1308

- **pure virtual** is an abstract method prototype defined in an abstract class (**virtual class**)
- **virtual** requires:
 - argument types must match
 - argument directions must match
 - number of arguments must match
 - return types must match
- **pure** requires:
 - method shall only be a prototype
 - no code can be included for the prototype
 - not even allowed to have **endfunction**/**endtask** keywords
 - **pure virtual** methods **MUST** be overridden in non-abstract classes

virtual requires
argument compatibility

pure creates a place-holder

pure requires that a
method be overridden with
an actual implementation

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

Pure Constraint

Mantis 2514

51 of 59



- **pure constraint** is an constraint prototype defined in an abstract class (**virtual class**)
- **pure** requires:
 - constraint shall only be a prototype
`pure constraint constraint-name;`
 - Every **pure constraint** MUST be overridden in non-abstract classes

Again: pure creates a place-holder

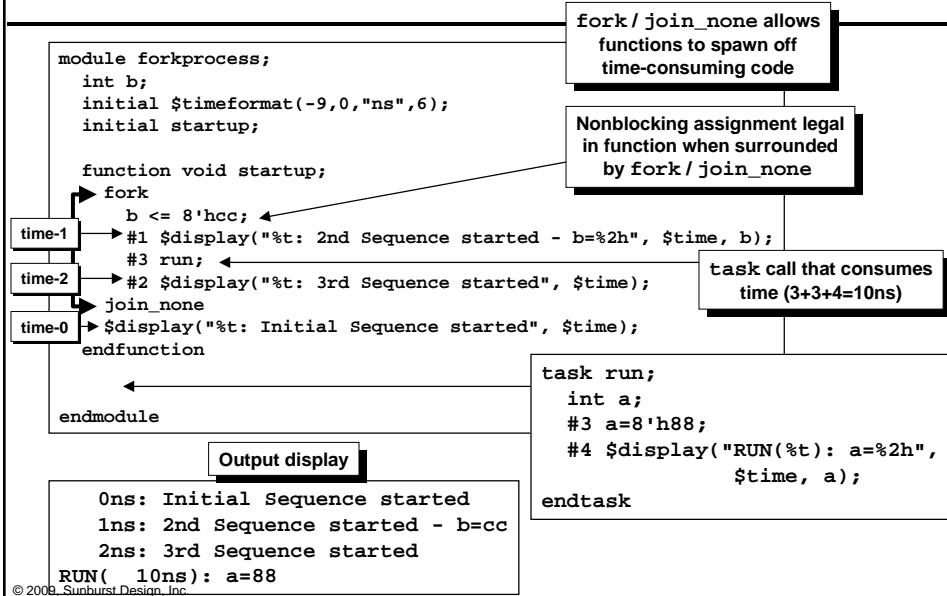
pure requires that a
constraint be overridden
with an actual constraint

© 2009, Sunburst Design, Inc.

Allow Functions to Spawn Processes

Mantis 1336

52 of 59




© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

53 of 59



External Methods w/ Parameterized Classes

Mantis 1857

```
class C #(int p = 1);
  extern static function int f();
endclass

function int C::f();
  return p + p;
endfunction

initial $display("%0d %0d", C#()::f(),C#(5)::f());
```

SystemVerilog-2005 does not allow a class type to use both parameters and external methods

Fixed method return type

Output display

2 10

```
class C #(int p = 1, type T = int);
  extern static function T f();
endclass

function C::T C::f();
  return p + C::p;
endfunction


initial $display("%0d %0d", C#()::f(),C#(5)::f());
```

SystemVerilog-2009 adds the ability to have parameterized classes with external methods

Declaration of extern static function (method) with return type T (parameterized type)

© 2009, Sunburst Design, Inc.

54 of 59



Covergroup Sample Method w/ Arguments

Mantis 2149

```
covergroup p_cg with function sample(bit a, int x);
  coverpoint x;
  cross x, a;
endgroup

p_cg cg1 = new;

property p1;
  int x;
  @(posedge clk) (a, x = b) ##1 (c, cg1.sample(a, x));
endproperty

c1: cover property (p1);
```

covergroup with sample arguments a and x

When a is high, set local variable x=b ...


... if c is high ##1 cycle later, sample the covergroup cg1 (p_cg) and pass the sampled a and x values to the covergroup

© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

55 of 59




SystemVerilog-2009 Miscellaneous Enhancements

`timeunit`
`$system task`
``__FILE__` & ``__LINE__` Macros
SDF Annotation of `$timeskew` & `$fullskew`

© 2009, Sunburst Design, Inc.

56 of 59



New Concise timeunit Syntax Mantis 1623

SystemVerilog-2005		SystemVerilog-2009
<pre>`timescale 1ns/1ns module tb; ... endmodule</pre>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">1ns/1ns timescale active</div>	<pre>`timescale 1ns/1ns module tb; ... endmodule</pre>
<pre>module register (output logic [7:0] q, input logic [7:0] d, input logic clk); timeunit 100ps; timeprecision 100ps; always_ff @(posedge clk) q <= #1 d; endmodule</pre>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">100ps/100ps local timescale</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Must be placed immediately after module header</div>	<pre>module register (output logic [7:0] q, input logic [7:0] d, input logic clk); timeunit 100ps/100ps; always_ff @(posedge clk) q <= #1 d; endmodule</pre>
<pre>module blkA (...); ... endmodule</pre>	<div style="border: 1px solid black; padding: 2px; display: inline-block;">1ns/1ns timescale again active</div>	<pre>module blkA (...); ... endmodule</pre>

© 2009, Sunburst Design, Inc.

2-line syntax

Combined 1-line syntax


DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009

57 of 59

\$system Task to Invoke SystemCommands

Mantis 1863



- The `$system` task allows SystemVerilog code to call operating system commands

```
module top;
    initial $system("mv design.v adder.v");
endmodule
```


Most Verilog simulators already have this task, but it was never part of the Verilog standard

© 2009, Sunburst Design, Inc.

58 of 59

`__FILE__ & `__LINE__ Macros

Mantis 1588



- SystemVerilog-2009 adds the C-like macros
 - ``__FILE__`
 - ``__LINE__`
- These macros allow access to the current file and line number from within SystemVerilog code


© 2009, Sunburst Design, Inc.

DAC 2009 SYSTEMVERILOG - 2009 PRESENTATION

by Sunburst Design, Beaverton, Oregon, © 2009


59 of 59


SDF Annotation of \$timeskew & \$fullskew
Mantis 1140




- Verilog-2001 added the timing skew checks
 \$timeskew
 \$fullskew
- SystemVerilog-2009 defines *how* these checks are annotated from SDF files

© 2009, Sunburst Design, Inc.



SystemVerilog Is Getting Even Better!
An Update on the Proposed 2009 SystemVerilog Standard
Part 1


CLIFFORD E. CUMMINGS
SUNBURST DESIGN, INC.
CLIFFC@SUNBURST-DESIGN.COM
WWW.SUNBURST-DESIGN.COM

Presented by


STUART SUTHERLAND
SUTHERLAND HDL, INC.
STUART@SUTHERLAND-HDL.COM
WWW.SUTHERLAND-HDL.COM

sponsored by



© 2009, Sunburst Design, Inc.