Getting the Most out of the New Verilog 2000 Standard

What's New, and What Is Synopsys Supporting

by Don Mills, LCDM Engineering

and Stuart Sutherland, Sutherland HDL, Inc.

Verilog and VHDL Training and Consulting Experts

Presented at the SNUG-Europe Conference, March 2001

Verilog-2000 Update

Sutherland HDL
&
LCDM Engineering

- The specification of the Verilog-2000 standard is complete
  - Final draft completed March 1st, 2000
  - The final IEEE balloting process completed in December, 2000
  - The IEEE RevCom committee will ratify Verilog-2000 in March, 2001
  - The official standard will be IEEE Std. 1364-2001

Why a New Standard?

Sutherland & HDL
LCDM Engineering

- Add enhancements to Verilog
  - Cliff Cumming's "Top Five Enhancement Requests" from HDLCon-1996
  - Support evolving design methodologies
    - System level design
    - Intellectual property models
    - Design re-use
    - Very deep submicron, etc.

- Clarify ambiguities in Verilog 1364-1995
  - The IEEE 1364-1995 reference manual came from the Gateway Design Automation Verilog-XL User's Manual
  - Verilog-2000 more clearly defines Verilog syntax and semantics

Goals for Verilog-2000

Sutherland & HDL LCDM Engineering

- Enhance Verilog for
  - Higher level, abstract system level modeling
  - Intellectual Property (IP) modeling
  - Greater timing accuracy for very deep submicron
- Make Verilog even easier to use
- Correct errata and ambiguities
- Maintain backward compatibility — existing models will work with the new standard
- Ensure that EDA vendors will implement all enhancements!

The IEEE 1364 Verilog Standards Committee

Sutherland & HDL LCDM Engineering

- A main working group
  - Final approval of all changes to 1364-1995
  - About 20 active participants
- Three task forces
  - Behavioral Task Force (Cliff Cummings, chair)
    - Responsible for RTL and behavioral modeling enhancements
  - ASIC Task Force (Steve Wadsworth, chair)
    - Responsible for ASIC and FPGA library modeling enhancements
  - PLI Task Force (Drew Lynch, Stu Sutherland, co-chairs)
    - Responsible for PLI enhancements

Overview of HDL Enhancements

- 23 major modeling enhancements were added to Verilog
  - This paper presents:
    - Brief descriptions and examples
    - Current and planned support in Synopsys VCS and Presto

- Several additional changes are not listed in this paper
  - Library vendor enhancements for Deep Submicrom
  - Programming Language Interface (PLI) enhancements
  - Value Change Dump (VCD) file enhancements
  - Support for the latest Standard Delay File (SDF) standard
  - Errata, clarifications and minor changes

1: Verilog Configurations

Sutherland & HDL LCDM Engineering

- Verilog-1995 leaves design management up to software tools
  - Every tool has different ways to manage large designs
- Verilog-2000 adds configuration blocks
  - All software tools will have a consistent method
  - The version for each module instance can be specified
    - Virtual libraries specified within Verilog source code
    - Physical file locations specified in a “map” file
  - New reserved words added: ***config***, ***endconfig***, ***design***, ***instance***, ***cell***, ***use***, ***liblist***

Synopsys support:	VCS 6.0 no	VCS 6.1 no	PRESTO no

Verilog Configuration Notes

Sutherland & HDL LCDM Engineering

- Verilog designs are modeled the same as always
- Configurations specify which module source code to use for each instance of a module.
  - With Verilog-1995, it is up to the simulator on how to specify which model version should be used for each instance
- The configuration block is specified outside of all modules
  - Verilog model source code does not need to be modified in order to change the design configuration!
- A separate file maps logical library names to physical file locations
  - Verilog source code does not need to be modified when a design is moved to a different physical source location!

Verilog Configuration Example

Sutherland & HDL LCDM Engineering

Verilog Design

```
module test;
  ...
  myChip dut (...);
  ...
endmodule
```

```
module myChip(...);
  ...
  adder a1 (...);
  adder a2 (...);
  ...
endmodule
```

Configuration Block (part of Verilog source code)

```
/* define a name for this configuration */
config cfg4

  /* specify where to find top level modules */
  design rtlLib.top

  /* set the default search order for finding
     instantiated modules */
  default liblist rtlLib gateLib;

  /* explicitly specify which library to use
     for the following module instance */
  instance test.dut.a2 liblist gateLib;
endconfig
```

Library Map File (separate from Verilog source)

```
/* location of RTL models (current directory) */
library rtlLib "./*.v";

/* Location of synthesized models */
library gateLib "./synth_out/*.v";
```

2: Verilog Generate

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds true generate capability
  - Use ***for*** loops to generate any number of instances of:
    - Modules, primitives, procedures, continuous assignments, tasks, functions, variables, nets
  - Use ***if–else*** and ***case*** decisions to control what instances are generated
    - Provides greater control than the VHDL generate
  - New reserved words added:
    - ***generate***, ***endgenerate***, ***genvar***, ***localparam***

Synopsys support:	VCS 6.0 no	VCS 6.1 no	PRESTO no

Verilog Generate Example


```
module multiplier (a, b, product);
  parameter a_width = 8, b_width = 8;
  localparam product_width = a_width + b_width;
  input   [a_width-1:0]       a;
  input   [b_width-1:0]       b;
  output  [product_width-1:0] product;

  generate
    if ((a_width < 8) || (b_width < 8))
      CLA_multiplier #(a_width, b_width) u1 (a, b, product);
    else
      WALLACE_multiplier #(a_width, b_width) u1 (a, b, product);
  endgenerate

endmodule
```

localparams are constants, which cannot be redefined

- If the input bus widths are 8-bits or less, generate and instance of a carry-look-ahead multiplier
- If the input bus widths are greater than 8-bits, generate an instance of a wallace-tree multiplier

3: Constant Functions

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds constant functions
  - Same syntax as standard Verilog functions
  - Limited to statements that can be evaluated at compile time
  - Can be called anywhere a constant expression is required
    - Vector width declarations
    - Array declarations
    - Replicate operations
- Provides for more scalable, re-usable models

Synopsys support:	VCS 6.0 no	VCS 6.1 no	PRESTO no

Constant Functions Example


```
module ram (...);
  parameter RAM_SIZE = 1024;
  parameter ADDRESS = 12;
  input [ADDRESS-1:0] address_bus;
```

Verilog 1995:

Vector widths can be calculated using simple constant expressions

```
module ram (...);
  parameter RAM_SIZE = 1024;
  input [clogb2(RAM_SIZE)-1:0] address_bus;
  ...
  function integer clogb2;
    input depth;
    integer i;
    begin
      clogb2 = 1;
      for (i = 0; 2**i < depth; i = i + 1)
        clogb2 = i + 1;
    end
  endfunction
  ...
```

Verilog 2000:

Vector widths can be calculated using complex constant functions

4: Indexed Vector Part Selects

- Verilog-2000 adds the capability to use variables to select a group of bits from a vector
  - The starting point of the part-select can vary
  - The width of the part-select remains constant
  - A **+:** indicates the part-select increases from the starting point
  - A **-:** indicates the part-select decreases from the starting point

```
reg [63:0] word;
reg  [3:0] byte_num; //a value from 0 to 7
wire [7:0] byteN = word[byte_num*8 +: 8];
```

The starting point of the part-select is variable

The width of the part-select is constant

Synopsys support: VCS 6.0 no VCS 6.1 no PRESTO no

5: Multi-dimensional Arrays

Sutherland & HDL LCDM Engineering

- Verilog-1995 allows 1-dimensional arrays of reg, integer and time variables
  - Typically used to model RAM and ROM memories
- Verilog-2000 adds:
  - Multidimensional arrays of any variable data type
  - Multidimensional arrays of any net data type

```
//declare a 3-dimensional array of 8-bit wire nets
wire [7:0] array3 [0:255][0:255][0:15];

//select one word out of a 3-dimensional array
wire [7:0] out3 = array3[addr1][addr2][addr3];
```

Synopsys support: VCS 6.0 no VCS 6.1 yes PRESTO yes

6: Array Bit and Part Selects

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds:
  - Bit-selects out of an array
  - Part-selects out of an array

```
//select the high-order byte of one word in a
//2-dimensional array of 32-bit reg variables
reg [31:0] array2 [0:255][0:15];
wire [7:0] out2 = array2[100][7][31:24];
```

Synopsys support: VCS 6.0 no VCS 6.1 no PRESTO yes

7: Signed Arithmetic Extensions

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds:
  - reg and net data types can be declared as signed

```
reg  signed [63:0] data;
wire signed [11:0] address;
```

  - Function returns can be declared as signed

```
function signed [128:0] alu;
```

  - Literal integer numbers can be declared as signed

```
16'shC501  //a signed 16-bit hex value
```

  - New arithmetic shift operators, <<< >>>, maintain the sign of a value
  - New $signed() and $unsigned() system functions can “cast” a value to signed or unsigned

VCS 6.0 no VCS 6.1 yes PRESTO 2000.11 no PRESTO 2001.08 yes

8: Power Operator

- Verilog-2000 add an exponential power operator
  - Represented by the ** token
  - Works like the C pow() function
  - If either operand is real, a real value is returned
  - If both operands are integers, an integer value is returned

```
module ram (...);
  parameter RAM_SIZE = 1024;
  input [(2**RAM_SIZE)-1:0] address_bus;
  ...
```

Synopsys support: VCS 6.0 no VCS 6.1 yes (planned) PRESTO yes

9: Re-entrant Tasks and Recursive Functions

- Verilog-2000 adds automatic tasks and functions
  - Each call to the task/function allocates unique storage
    - In Verilog-1995, tasks and functions are static; each call shares the same storage space
  - Concurrent task calls will not interfere with each other
  - Recursive calls to a function are stacked
  - New reserved word added: automatic

```
function automatic [63:0] factorial;
  input [31:0] n;
  if (n == 1)
    factorial = 1;
  else
    factorial = n * factorial(n-1);
endfunction
```

Recursive function call

Synopsys support: VCS 6.0 no VCS 6.1 yes (planned) PRESTO no

10: Combinational Logic Sensitivity

- Verilog-2000 adds a “wildcard” token to indicate a combinational logic sensitivity list
  - The @* token indicates that an always procedure is automatically sensitive to any change on any input to that procedure

Verilog-1995

```
always @(sel or a or b or c or d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Verilog-2000

```
always @*
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Synopsys support: VCS 6.0 no VCS 6.1 yes (planned) PRESTO yes

11: Comma-separated Sensitivity List

- Verilog-2000 adds a second syntax style for listing signals in a sensitivity list
  - Signals in the list can be separated with a comma
    - The old “or” separated list will still work

Verilog-1995

```
always @(sel or a or b or c or d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Verilog-2000

```
always @(sel, a, b, c, d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Synopsys support: VCS 6.0 no VCS 6.1 yes (planned) PRESTO yes

12: Enhanced File I/O

Sutherland & HDL LCDM Engineering

- Verilog-1995 has limited built-in file I/O tasks
  - Up to 31 files can be opened for writing
    - Only ASCII characters can be written to files
  - More complex file I/O is done using the Verilog Programming Language Interface (PLI)
- Verilog-2000 adds:
  - The ability to open up to 2^{30} files
  - New, built-in file I/O tasks: **$ferror**, **$fgetc**, **$fgets**, **$fflush**, **$fread**, **$fscanf**, **$fseek**, **$fsscanf**, **$ftel**, **$rewind**, **$sformat**, **$swrite**, **$swriteb**, **$swriteh**, **$swriteo**, **$ungetc**

Synopsys support:	VCS 6.0 no	VCS 6.1 yes	PRESTO no

13: Width Extension Beyond 32 bits

- In Verilog-1995:
  - Verilog assignments zero fill when the left-hand side vector is wider than the right-hand side
  - Unsized integers default to 32-bits wide, which would not fill a vector greater than 32 bits correctly

Verilog-1995

```
parameter WIDTH = 64;
reg [WIDTH-1:0] data;
data = 'bz;    //fills with 'h00000000zzzzzzzz
data = 64'bz;  //fills with 'hzzzzzzzzzzzzzzzz
```

- Verilog-2000 will automatically extend a logic Z or X to the full width of the left-hand side

Verilog-2000

```
parameter WIDTH = 64;
reg [WIDTH-1:0] data;
data = 'bz;    //fills with 'hzzzzzzzzzzzzzzzz
```

Synopsys support: VCS 6.0 no VCS 6.1 no PRESTO no

14: Explicit In-line Parameter Passing

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds the ability to explicitly name parameters when passing parameter values
  - Provides better self-documenting code
  - Parameter values can be passed in any order

Verilog-1995

```
module my_chip (...);
  ...
  RAM #(8,1023) ram2 (...);
endmodule
```

Verilog-2000

```
module my_chip (...);
  ...
  RAM #(.SIZE(1023)) ram2 (...);
endmodule
```

```
module ram (...);
  parameter WIDTH = 8;
  parameter SIZE = 256;
  ...
endmodule
```

Synopsys support: VCS 6.0 yes VCS 6.1 yes PRESTO yes

15: Sized Parameters

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds the ability to specify the size of parameter constants
  - This has been a de-facto standard for many years, but was not in the Verilog-1995 standard
  - An un-sized parameter will default to the size of the initial value assigned to it

Verilog-1995

```
module fsm (...);
  parameter IDLE  = 3'd0,
            READ  = 3'd1,
            LOAD  = 3'd2,
            SYNC  = 3'd3,
            ERROR = 3'd4;
  ...
```

Verilog-2000

```
module my_chip (...);
  parameter [2:0] IDLE  = 3'd0,
                  READ  = 3'd1,
                  LOAD  = 3'd2,
                  SYNC  = 3'd3,
                  ERROR = 3'd4;
  ...
```

Synopsys support: VCS 6.0 no VCS 6.1 yes (planned) PRESTO yes

16: Combined Port/Type Declarations

- Verilog-2000 permits combining port declarations and data type declarations into one statement

Verilog-1995

```
module mux8 (y, a, b, en);
  output [7:0] y;
  input  [7:0] a, b;
  input        en;

  reg  [7:0] y;
  wire [7:0] a, b;
  wire       en;
  ...
```

Verilog-2000

```
module mux8 (y, a, b, en);
  output reg  [7:0] y;
  input  wire [7:0] a, b;
  input  wire       en;
  ...
```

VCS 6.0 yes VCS 6.1 yes PRESTO 2000.11 no PRESTO 2001.08 yes (planned)

17: ANSI-style Port Lists

- Verilog-2000 adds ANSI C style input and output declarations
  - For modules, tasks and functions

Verilog-1995

```
module mux8 (y, a, b, en);
  output [7:0] y;
  input  [7:0] a, b;
  input        en;

  reg  [7:0] y;
  wire [7:0] a, b;
  wire       en;
  ...
```

Verilog-2000

```
module mux8 (output reg  [7:0] y,
             input  wire [7:0] a,
             input  wire [7:0] b,
             input  wire       en);
  ...
```

VCS 6.0 yes VCS 6.1 yes PRESTO 2000.11 no PRESTO 2001.08 yes (planned)

18: Reg Declaration With Initialization

- Verilog-2000 permits initializing variables at the time they are declared
  - The initialization is executed in time-step zero, just like initial procedures

Verilog-1995

```
reg clock;

initial
  clk = 0;
```

Verilog-2000

```
reg clock = 0;
```

Synopsys support: VCS 6.0 yes VCS 6.1 yes PRESTO no

19: Implicit Nets with Continuous Assigns

- Verilog-2000 will default to a net data type on the left-hand side of any continuous assignment
  - The vector width is the size of the right-hand side expression, if not connected to a port of the module
  - In Verilog-1995, the left-hand side must be explicitly declared, if not connected to a port of the module

Verilog-1995

```
module mult32 (y, a, b);
  output [63:0] y;
  input  [31:0] a, b;
  assign y = a * b;  //defaults to wire, width of port y
  assign n = a * b;  //ERROR: 'n' not declared
endmodule
```

Verilog-2000

```
  assign n = a * b;  //defaults to wire, width of (a * b)
```

Synopsys support: VCS 6.0 ? VCS 6.1 ? PRESTO ?

20: Disable Implicit Net Declarations

Sutherland & HDL LCDM Engineering

- In Verilog-1995, undeclared signals can default to a wire data type
  - The default data type can be changed to another net data type using `default_nettype <data_type>
- Verilog-2000 provides a means to disable default net declarations
  - `default_nettype **none**
  - Any undeclared signals will be a syntax error
    - Prevents hard-to-debug wiring errors due to a mistyped name
  - none is not a reserved word

Synopsys support:	VCS 6.0 ?	VCS 6.1 ?	PRESTO ?

21: Enhanced Conditional Compilation

Sutherland & HDL LCDM Engineering

- Verilog-1995 supports limited conditional compilation
  - The `ifdef, `else and `endif compiler directives
- Verilog-2000 adds more extensive conditional compilation control
  - New directives: **`ifndef**, **`elsif** and **`undef**

Synopsys support:	VCS 6.0 yes	VCS 6.1 yes	PRESTO yes

22: File and Line Compiler Directives

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds file and line compiler directives
  - New directives: **`file** and **`line**
  - Document the original location of Verilog source code
    - Verilog tools often include file name and line number information in error and warning messages
    - If a pre-process utility program modifies the Verilog source code, the original file and line information could be lost

Synopsys support:	VCS 6.0 no	VCS 6.1 no	PRESTO no

23: Attributes

Sutherland & HDL LCDM Engineering

- Verilog-2000 adds “attribute” properties
  - A standard means to specify non-Verilog tool specific information to Verilog models
  - Adds new tokens **(*** and ***)**
  - Eliminates need to hide commands in comments
  - The standard does not define any specific attributes
    - Software vendors can define proprietary attributes
    - Other standards might define standard attributes

Verilog-1995

```
case (1'b1) /* synopsys parallel_case */ //1-hot FSM
```

Verilog-2000

```
(* parallel case *) case (1'b1)  //1-hot FSM
```

Synopsys support: VCS 6.0 no VCS 6.1 no PRESTO no

24: Array of Instances

- Verilog-1995 can create multiple instances of the same module or primitive, using instance arrays
  - This is *not* a new feature for Verilog-2000
  - It is mentioned in this paper because Synopsys has recently added support for instance arrays in both VCS and Presto

Verilog-1995

```
module RegN (out, in, clk, rst);
  parameter N = 16;
  output [N-1:0] out;
  input  [N-1:0] in;
  input          clk, rst);

  dff i[N-1:0] (out, in, clk, rst);

module
```

General Rules:

- **A vector signal connected to an array is split apart, with different bits connected to each instance**
- **A scalar signal connected to an array is connected to each instance**

Synopsys support: VCS 5.2 yes VCS 6.0 yes VCS 6.1 yes PRESTO yes

25: "Register" changed to "Variable"

Sutherland & HDL LCDM Engineering

- The Verilog-2000 standard changes the term "register" to "variable"
  - This is not really a language or modeling change — "register" is not a reserved word; it is just a term
  - Since its inception in 1984, Verilog manuals have used the term "register" to describe a class of data types
    - reg (unsigned variable), integer (signed variable), real (double precision variable), etc.
  - The term "register" often confuses new Verilog users
    - Register is a hardware term for storage elements
    - Verilog variables do not imply hardware registers

Synopsys support:	VCS 6.0 n/a	VCS 6.1 n/a	PRESTO n/a

Summary

Sutherland & HDL LCDM Engineering

- Verilog-2000 is complete
  - The IEEE balloting is complete, with 92% approving the standard
  - Final IEEE RevCom approval is expected in March, 2001
    - Officially, “Verilog-2000” will be the IEEE 1364-2001 standard
- Verilog-2000 contains
  - Over 30 major enhancements
  - Many clarifications and errata corrections
- Verilog-2000 adds powerful capabilities
  - Greater deep submicron accuracy
  - More abstract system level modeling
  - Scalable, re-usable modeling

About The Authors

Sutherland HDL & LCDM Engineering

- Stuart Sutherland
  - President of Sutherland HDL, Inc., Portland, Oregon, USA
  - Presents Verilog HDL and Verilog PLI training workshops
  - More than 13 years experience with Verilog
  - Author of "*The Verilog HDL Quick Reference Guide*" and "*The Verilog PLI Handbook*"
  - Member of the IEEE 1364 Verilog standards committee since 1993
    - Chair of the Verilog PLI task force within the standards committee
- Don Mills
  - President of LCDM Engineering., Salt Lake City, Utah, USA
  - Provides expert design consulting
  - Presents Verilog HDL and VHDL training workshops
  - More than 15 years experience in ASIC design
  - Has been using Synopsys tools since version 1.2