Verilog HDL Syntax And Semantics Part-I

Feb-9-2014

ASIC WORLD Directory
Verilog
Tutorial
Examples
Questions
Tools
Books
Links
FAQ
Sponsor
Home
Disclaimer
FAQ
Hot Jobs

Lexical Conventions

The basic lexical conventions used by Verilog HDL are similar to those in the C programming language. Verilog HDL is a case-sensitive language. All keywords are in lowercase.

White Space

White space can contain the characters for blanks, tabs, newlines, and form feeds. These characters are ignored except when they serve to separate other tokens. However, blanks and tabs are significant in strings.

White space characters are :

- Blank spaces
- Tabs
- Carriage returns
- New-line
- Form-feeds

Examples of White Spaces

Functional Equivalent Code

Bad Code : Never write code like this.

```
module addbit(a,b,ci,sum,co);
input a,b,ci;output sum co;
wire a,b,ci,sum,co;endmodule
```

You could download file bad_code.v here

Good Code : Nice way to write code.

```
module addbit (
a,
b,
ci,
sum,
co);
input        a;
input        b;
input          ci;
output         sum;
output         co;
wire           a;
wire           b;
wire           ci;
wire           sum;
wire           co;

endmodule
```

You could download file good_code.v here

Comments

There are two forms to introduce comments.

- Single line comments begin with the token // and end with a carriage return
- Multi line comments begin with the token /* and end with the token */

Examples of Comments

```
/* This is a
  Multi line comment
  example */
module addbit (
a,
b,
ci,
sum,
co);

// Input Ports  Single line comment
input          a;
input          b;
input          ci;
// Output ports
output         sum;
output         co;
// Data Types
wire           a;
wire           b;
wire           ci;
wire           sum;
wire           co;

endmodule
```

You could download file comment.v here

Case Sensitivity

Verilog HDL is case sensitive

- Lower case letters are unique from upper case letters
- All Verilog keywords are lower case

Examples of Unique names

```
input                   // a Verilog Keyword
wire                    // a Verilog Keyword
WIRE                 // a unique name ( not a keyword)
Wire                   // a unique name (not a keyword)
```

You could download file unique_names.v here

NOTE : Never use Verilog keywords as unique names, even if the case is different.

Identifiers

Identifiers are names used to give an object, such as a register or a function or a module, a name so that it can be referenced from other places in a description.

- Identifiers must begin with an alphabetic character or the underscore character (**a-z A-Z _**)
- Identifiers may contain alphabetic characters, numeric characters, the underscore, and the dollar sign (**a-z A-Z 0-9 _ $**)
- Identifiers can be up to 1024 characters long.

Examples of legal identifiers

data_input mu
clk_input my$clk
i386 A

Escaped Identifiers

Verilog HDL allows any character to be used in an identifier by escaping the identifier. Escaped identifiers provide a means of including any of the printable ASCII characters in an identifier (the decimal values 33 through 126, or 21 through 7E in hexadecimal).

- Escaped identifiers begin with the back slash (\ )
- Entire identifier is escaped by the back slash.
- Escaped identifier is terminated by white space (Characters such as commas, parentheses, and semicolons become part of the escaped identifier unless preceded by a white space)
- Terminate escaped identifiers with white space, otherwise characters that should follow the identifier are considered as part of it.

Examples of escape identifiers

Verilog does not allow to identifier to start with a numeric character. So if you really want to use a identifier to start with a numeric value then use a escape character as shown below.

```
// There must be white space after the
// string which uses escape character
module \1dff (
q,        // Q output
\q~ ,     // Q_out output
d,        // D input
cl$k,     // CLOCK input
\reset*  // Reset input
);

input d, cl$k, \reset* ;
output q, \q~ ;

endmodule
```

You could download file escape_id.v here

Numbers in Verilog

You can specify constant numbers in decimal, hexadecimal, octal, or binary format. Negative numbers are represented in 2's complement form. When used in a number, the question mark (?) character is the Verilog alternative for the z character. The underscore character (_) is legal anywhere in a number except as the first character, where it is ignored.

Integer Numbers

Verilog HDL allows integer numbers to be specified as

- Sized or unsized numbers (Unsized size is 32 bits)
- In a radix of binary, octal, decimal, or hexadecimal
- Radix and hex digits (a,b,c,d,e,f) are case insensitive
- Spaces are allowed between the size, radix and value

Syntax: <size>'<radix><value>;

Example of Integer Numbers

Integer	Stored as
1	00000000000000000000000000000001
8'hAA	10101010
6'b10_0011	100011
'hF	00000000000000000000000000001111

Verilog expands <value> filling the specified <size> by working from right-to-left

- When <size> is smaller than <value>, then leftmost bits of <value> are truncated
- When <size> is larger than <value>, then leftmost bits are filled, based on the value of the leftmost bit in <value>.

- Leftmost '0' or '1' are filled with '0'
- Leftmost 'Z' are filled with 'Z'
- Leftmost 'X' are filled with 'X'

Note : X Stands for unknown and Z stands for high impedance, 1 for logic high or 1 and 0 for logic low or 0.

Example of Integer Numbers

Integer	Stored as
6'hCA	001010
6'hA	001010
16'bZ	ZZZZZZZZZZZZZZZZ
8'bx	xxxxxxxx

Real Numbers

- Verilog supports real constants and variables
- Verilog converts real numbers to integers by rounding
- Real Numbers can not contain 'Z' and 'X'
- Real numbers may be specified in either decimal or scientific notation
- < value >.< value >
- < mantissa >E< exponent >
- Real numbers are rounded off to the nearest integer when assigning to an integer.

Example of Real Numbers

Real Number	Decimal notation
1.2	1.2
0.6	0.6
3.5E6	3,500000.0

Signed and Unsigned Numbers

Verilog Supports both types of numbers, but with certain restrictions. Like in C language we don't have int and unint types to say if a number is signed integer or unsigned integer.

Any number that does not have negative sign prefix is a positive number. Or indirect way would be "Unsigned".

Negative numbers can be specified by putting a minus sign before the size for a constant number, thus they become signed numbers. Verilog internally represents negative numbers in 2's complement format. An optional signed specifier can be added for signed arithmetic.

Examples

Number	Description
32'hDEAD_BEEF	Unsigned or signed positive number
-14'h1234	Signed negative number

The example file below shows how Verilog treats signed and unsigned numbers.

```verilog
module signed_number;

reg [31:0]  a;

initial begin
  a = 14'h1234;
  $display ("Current Value of a = %h", a);
  a = -14'h1234;
  $display ("Current Value of a = %h", a);
  a = 32'hDEAD_BEEF;
  $display ("Current Value of a = %h", a);
  a = -32'hDEAD_BEEF;
  $display ("Current Value of a = %h", a);
   #10  $finish;
end

endmodule
```

You could download file signed_number.v here

```
Current Value of a = 00001234
Current Value of a = ffffedcc
Current Value of a = deadbeef
Current Value of a = 21524111
```

BACK HOME NEXT