Introduction to Verilog HDL

HDL?

- HDL stands for "Hardware Description Language." It is a specialized computer language used to describe the structure and behavior of digital electronic circuits, such as microprocessors, ASICs (Application-Specific Integrated Circuits), and FPGAs (Field-Programmable Gate Arrays).

- The primary purpose of HDLs is to enable designers and engineers to model and simulate complex hardware systems before they are physically implemented in silicon or other hardware technologies. HDLs are essential tools in digital hardware design and are used throughout the entire design flow, from concept and simulation to synthesis and verification.

- Popular HDLs are Verilog HDL & VHDL (for any complexity).

Features of HDLs

- Easy development, verification and debugging through HDLs.
- HDL descriptions are easily portable, and is also compatible to all design tools.
- HDLs can describe the digital systems at various abstraction levels & also supports hierarchical modeling.
- HDL descriptions can be functionally simulated with Logic Simulators .

Features of HDLs

- Easy development, verification and debugging through HDLs.
- HDL descriptions are easily portable, and is also compatible to all design tools.
- HDLs can describe the digital systems at various abstraction levels & also supports hierarchical modeling.
- HDL descriptions can be functionally simulated with Logic Simulators .

History of Verilog

- Developed by Philip Moorby in 1984-1985.
- Gateway Design Automation introduced Verilog in 1984 as their proprietary HDL.
- Cadence took over in 1989.
- Cadence made Verilog HDL public in 1990.
- Verilog HDL becomes IEEE 1364-1995.

Popularity of Verilog HDL

- Verilog HDL is modeled after 'C' language.
- Allows different levels of abstraction to be mixed in the same model.
- Easy to learn and easy to use.
- Almost all logic simulation & synthesis tools support Verilog.
- Verilog HDL is an IEEE 1364 standard.

Verilog basics

- Verilog HDL is case-sensitive.
- All the keywords in Verilog must be in lower case.
- Verilog constructs may be written across multiple lines, or on one line.

Identifiers and key words

- Identifiers and keywords are used to define language constructs.
- Identifiers refer objects to be referenced in the design.
- Identifiers are made of alphabets (both cases), numbers, the underscore '_' and the dollar sign '$'.
- They start with an alphabetic character or underscore.

cont'd..

- They cannot start with a number or with '$' which is reserved for system tasks.
- Identifiers are case sensitive i.e., identifiers differing in their case are distinct.
- An identifier say count is different from COUNT, count and cOuNT.

Examples of Identifiers

Count

COUNT

_R2_D2

R56_68

FIVE$

$count Illegal

12six_b Illegal

- Example:

```
reg  sum;
input data;
```

Number specification

- Sized numbers.
  - <size> ’<base format> <number>
- Unsized numbers.
  - ’<base format> <number>
- <size> in decimal
- <base format> can be b or B, d or D, o or O and h or H.
- Numbers without <base format> are decimal by default.

cont’d..

- Examples:

Sized numbers :

4'b1111 // This is a 4-bit binary number

12'habc // This is a 12-bit hexadecimal number

16'd255 // This is a 16-bit decimal number.

Unsized numbers :

23456 // This is a 32-bit decimal number by default

'hc3 // This is a 32-bit hexadecimal number

'o21 // This is a 32-bit octal number

Unknown & high impedance values

- X or x for unknown values.
- Z or z for high impedance values.
- X or Z at the MSB has the self padding property.

Examples:

32 'B z // this is a 32-bit high impedance number

6 'h X // this is a 6-bit hex number

12 'H 13x // this is a 12-bit hex number