INTRODUCTION TO VERILOG HDL

Verilog as HDL

- Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL).
- It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip−flop.
- It means, by using a HDL we can describe any digital hardware at any level.
- Designs, which are described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits.

Levels of Design Description

- Verilog supports a design at many levels of abstraction.

The major four are –

- **Switch Level (circuit level)**
- **Behavioral level (algorithmic)**
- **Register-transfer level-RTL (data flow)**
- **Gate level (structural)**

- **Switch Level:**
  - layout of the wires, resistors and transistors on an IC chip
  - Easiest to synthesize, very difficult to write, not really used
- **Gate (Structural) Level:**
  - logical gates, flip flops and their interconnection
  - Very easy to synthesize, a text based schematic entry system
- **RTL (dataflow) Level**
  - The registers and the transfers of vectors of information between registers.
  - Most efficiently synthesizable level
  - Uses the concept of registers with combinational logic
- **Behavioral (algorithmic) Level**
  - Highest level of abstraction
  - Description of algorithm without hardware implementation details
  - Easiest to write and debug, most difficult to synthesize

The Overall Design Structure in Verilog

- The possibilities of design description statements and assignments at different levels necessitate their accommodation in a mixed mode.
- In fact the design statements coexisting in a seamless manner within a design module is a significant characteristic of Verilog.
- Thus Verilog facilitates the mixing of the above-mentioned levels of design.
- A design built at data flow level can be instantiated to form a structural mode design.
- Data flow assignments can be incorporated in designs which are basically at behavioral level

Concurrency

- In an electronic circuit all the units are to be active and functioning concurrently.
- The voltages and currents in the different elements in the circuit can change simultaneously. In turn the logic levels too can change.
- Simulation of such a circuit in an HDL calls for concurrency of operation.
- A number of activities –may be spread over different modules –are to be run concurrently here. Verilog simulators are built to simulate concurrency.
- This is in contrast to programs in the normal languages like C where execution is sequential.

- Concurrency is achieved by proceeding with simulation in equal time steps.
- The time step is kept small enough to be negligible compared with the propagation delay values.
- All the activities scheduled at one time step are completed and then the simulator advances to the next time step and so on.
- The time step values refer to simulation time and not real time. One can redefine timescales to suit technology as and when necessary and carry out test runs.
- In some cases the circuit itself may demand sequential operation as with data transfer and memory-based operations.
- Only in such cases sequential operation is ensured by the appropriate usage of sequential constructs from Verilog HDL

Simulation and Synthesis

- The design that is specified and entered as described earlier is simulated for functionality and fully debugged.
- Translation of the debugged design into the corresponding hardware circuit (using an FPGA or an ASIC) is called "synthesis."
- The tools available for synthesis relate more easily with the gate level and data flow level modules.
- The circuits realized from them are essentially direct translations of functions into circuit elements.
- In contrast many of the behavioral level constructs are not directly synthesizable; even if synthesized they are likely to yield relatively redundant or wrong hardware.

- The way out is to take the behavioral level modules and redo each of them at lower levels.
- The process is carried out successively with each of the behavioral level modules until practically the full design is available as a pack of modules at gate and data flow levels (more commonly called the "RTL level").

Programming Language Interface (PLI)

- PLI provides an active interface to a compiled Verilog module.
- The interface adds a new dimension to working with Verilog routines from a C platform.
- The key functions of the interface are as follows:
  - One can read data from a file and pass it to a Verilog module as input. Such data can be test vectors or other input data to the module. Similarly, variables in Verilog modules can be accessed and their values written to output devices.
  - Delay values, logic values, etc., within a module can be accessed and altered.
  - Blocks written in C language can be linked to Verilog modules.

Importance of HDLs

- **Designs can be described at a very abstract level by use of HDLs.** Designers can write their RTL description without choosing a specific fabrication technology. Logic synthesis tools can automatically convert the design to any fabrication technology. If a new technology emerges, designers do not need to redesign their circuit. They simply input the RTL description to the logic synthesis tool and create a new gate-level netlist, using the new fabrication technology. The logic synthesis tool will optimize the circuit in area and timing for the new technology.

- **By describing designs in HDLs, functional verification of the design can be done early in the design cycle**. Since designers work at the RTL level, they can optimize and modify the RTL description until it meets the desired functionality. Most design bugs are eliminated at this point. This cuts down design cycle time significantly because the probability of hitting a functional bug at a later time in the gate-level netlist or physical layout is minimized.
- New tools and languages focused on verification have emerged in the past few years. These languages are better suited for functional verification. However, for logic design, HDLs continue as the preferred choice

- **Designing with HDLs is analogous to computer programming**. A textual description with comments is an easier way to develop and debug circuits. This also provides a concise representation of the design, compared to gate-level schematics. Gate-level schematics are almost incomprehensible for very complex designs

- With rapidly increasing complexities of digital circuits and increasingly sophisticated EDA tools, HDLs are now the dominant method for large digital designs. No digital circuit designer can afford to ignore HDL-based design.

Popularity of Verilog HDL

Verilog HDL has evolved as a standard hardware description language. Verilog HDL offers many useful features

- Verilog HDL is a general-purpose hardware description language that is easy to learn and easy to use. It is similar in syntax to the C programming language. Designers with C programming experience will find it easy to learn Verilog HDL.
- Verilog HDL allows different levels of abstraction to be mixed in the same model. Thus, a designer can define a hardware model in terms of switches, gates, RTL, or behavioral code. Also, a designer needs to learn only one language for stimulus and hierarchical design.

- Most popular logic synthesis tools support Verilog HDL. This makes it the language of choice for designers.
- All fabrication vendors provide Verilog HDL libraries for post-logic synthesis simulation. Thus, designing a chip in Verilog HDL allows the widest choice of vendors.
- The Programming Language Interface (PLI) is a powerful feature that allows the user to write custom C code to interact with the internal data structures of Verilog. Designers can customize a Verilog HDL simulator to their needs with the PLI.

THANK YOU