DIGITAL DESIGN THROUGH VERILOG HDL (DDTV)

UNIT-I

Introduction to Verilog HDL:

What is Verilog HDL

Verilog HDL is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic-level to the gate-level to the switch level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete electronic digital system, or anything in b/w.

The Verilog HDL language includes capabilities to describe the behavioral nature of a design, the dataflow nature of a design, a design's structural-composition, delays & a waveform generation mechanism including aspects of response monitoring & verification, all modeled using one single language. In addition, the language provides a programming language interface through which the internals of a design can be accessed during simulation including the control of a simulation

The language not only defines the syntax but also defines very clear simulation semantics for each language construct. Therefore models written in this language can be verified using a Verilog simulator. The language inherits many of its operator symbols & construct from the C programming language. Verilog HDL provides an extensive range of modeling capabilities, some of which are quite difficult to comprehend initially.

History:

The Verilog HDL language was first developed

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Automation in 1983 as a hardware modeling language for their Simulator product. At that time it was a proprietary language. Because of the popularity of their Simulator product, Verilog HDL gained acceptance as a Usable and practical language by a number of designers. In an effort to Increase the popularity of the language, the language was placed in the public domain in 1990. Open Verilog International (OVI) was formed to promote Verilog. In 1992, OVI decided to pursue Standardization of Verilog HDL as an IEEE standard. This effort was Successfull & the language became an IEEE standard in 1995. The complete standard is described in the Verilog hardware Description language Reference Manual. The Standard is called IEEE Std 1364-1995.

A new Revision of the language was Standardized in 2001, called IEEE Std 1364-2001. This revision Includes many few features including alternate port and argument declaration styles, generate Statements and configurations.

Major Capabilities:

Major Capabilities of the VHDL.

- primitive logic gates, Such as and, or & nand, are built-in into the language.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

→ Flexibility of creating a User-defined primitive (UDP). Such a primitive could either be a Combinational logic primitive or a Sequential logic primitive.

→ Switch level modeling primitive gates, such as pmos & nmos, are also built-in into the language.

→ Explicit language constructs are provided for specifying pin-to-pin delays, path delays & timing checks of a design.

→ A design can be modeled in three different styles or in a mixed style. These styles are: behavioral style - modeled using procedural constructs; dataflow style - modeled using continuous assignments; & structural style - modeled using gate & module instantiations.

→ There are two data types in Verilog HDL; the net data type & the Variable data type. The net type represents a physical connection between structural elements while a Variable type may represent an abstract data storage elements.

→ Hierarchical designs can be described, up to any level, using the module instantiation construct.

→ A design can be Arbitrary Size; the language does not impose a limit.

→ Verilog HDL is non-proprietary & is an IEEE standard.

→ It is human & machine readable. Thus it can be used as an exchange language between tools & designers.

→ The capabilities of the Verilog HDL language can be further extended by using the programming language Interface (PLI) mechanism. PLI is a collection of routines that allow foreign functions to access information within a Verilog

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

module & allows for designer interaction with the simulator.

→ A design can be described in a wide range of levels, ranging from switch-level, gate-level, Register Transfer level (RTL) to algorithmic level, including process & queuing-level.

→ A design can be modeled entirely at the switch-level using the built-in switch level primitives.

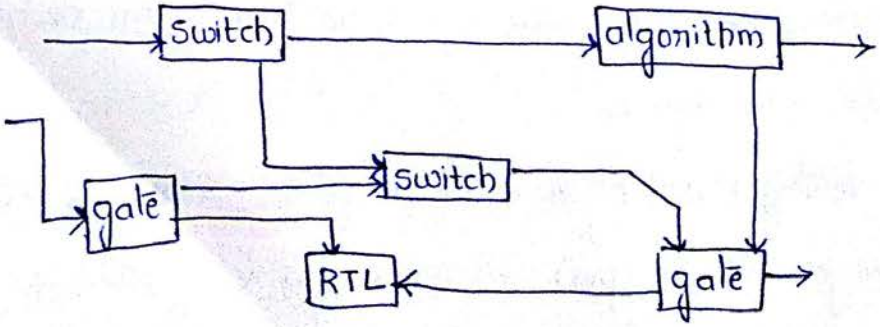
→ The same single language can be used to generate stimulus for the design & for specifying test constraints, such as specifying the values of inputs.

→ Verilog HDL can be used to perform Response monitoring of the design under test. that is, the values of a design under test can be monitored & displayed. These values can also be compared with expected values, & in case of a mismatch, a Report message can be printed.

→ At the behavioral-level, Verilog HDL can be used to describe a design not only at the RTL-level, but also at the Architectural-level & its algorithmic-level behaviour.

→ At the structural-level, gate & module instantiations can be used.

→ The fig shows the mixed-level modeling capability of Verilog HDL, that is, in one design, each module may be modeled at a different level.



Mixed-level modeling.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

→ Verilog HDL also has built-in logic functions such as & (bitwise-And) And | (bitwise-or).

→ High-level programming language constructs such as Conditionals, Case Statements, & loops are Available in the language.

→ Notion of the Concurrency & Time can be explicitly modeled.

→ powerful files read & write capabilities are provided.

→ The language is non-deterministic Under certain situations, that is, a model may produce different results on different simulators; for example, the Ordering of events On an events queue is not defined by the Standard.

VIVA Questions:

(1) In which year was Verilog HDL first Standardized by the IEEE.

(2) what are the three basic description styles supported by Verilog HDL.

(3) Can timing of a design be described Using Verilog HDL.

(4) What feature in the language can be Used to describe parameterized design

(5) Can a test bench be written Using Verilog HDL.

(6) Verilog HDL was first developed by which company.

(7) what are the two main data types in Verilog HDL.

(8) what does UDP Stands for

(9) Name two Switch-level modeling primitives gates.

(10) Name two logic primitive gate.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Levels of Design Description

The components of the target design can be described at different levels with the help of the constructs in Verilog.

Circuit level:

At the ckt level, a switch is the basic element with which digital ckt are built. Switches can be combined to form inverters & other gates at the next higher level of abstraction. Verilog has the basic MOS switches built into its constructs, which can be used to built basic ckt like inverters, basic logic gates, simple 1-bit dynamic & static memories. They can be used to build up larger designs to simulate at the ckt level, to design performance critical ckt. fig shows the ckt of an inverter suitable for description with the switch level constructs of Verilog.

Gate level:

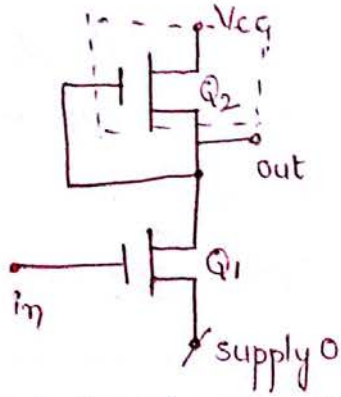
At the next higher level of abstraction, design is carried out in terms of basic gates, All the basic gates are available as ready modules called "primitives." Each such primitives is defined in terms of its I/p's & O/p's. primitives can be incorporated into design descriptions directly. Just as full physical hardware can be built using gates, the primitives can be used repeatedly and judiciously to build larger systems. fig an AND gate suitable for description using the gate primitive of Verilog. The gate level modeling or structural modeling as it is sometimes called is ~~skin~~ akin to building a digital circuit on a bread board, or on a PCB. One should know the structure of the design to build the model here. One can also build hierarchical ckts at this level. However, beyond 20 to 30 of such gate primitives in a ckt, the design description becomes unwieldy; testing & debugging becomes laborious.

Data flow:

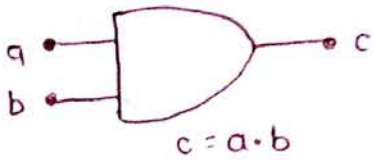
Data flow is the next higher level of & abstraction. All possible operations on signals & variables are represented here in terms of assignments. All logic & algebraic operations are accommodated. The assignments

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

functioning of the concerned block. At the data flow level, signals are assigned through the data manipulated equations. All such assignments are concurrent in nature. The design descriptions are more compact than those at the gate level. Fig shows an A-O-I relationship suitable for description with the Verilog constructs at the data flow level



A Simple Inverter ckt at the Switch level



A Simple AND gate represented at the gate level.

Behavioral level

Behavioral level constitutes the highest level of design description, it is essentially at the system level itself. with the assignment possibilities, looping constructs & conditional branching possible, the design description essentially looks like a "c" program. The statements involved are "dense" in function. Compactness & the comprehensive nature of the design description make the development process fast & efficient. Fig 2.4 shows an A-O-I gate expressed in pseudo code suitable for description with the behavioral level constructs of Verilog.

The Overall Design structure in Verilog:

The possibilities of design description statements & assignments at different levels necessitate their accommodation in a mixed mode. In fact the design statement coexisting in a seamless manner within a design module is a significant characteristics of Verilog. Thus Verilog facilitates the mixing of the above –

created with nitroPDF professional
download the free trial online at nitropdf.com/professional
www.smartzworld.com jntuworldupdates.org

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 Unit are to be active & functioning concurrently. The Voltages & currents in the different elements in the circuits can change simultaneously. In turn the logic levels too can change. Simulation of such a ckt in a 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 language 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 & then the simulator.

$e = \overline{a \cdot b + c \cdot d}$	If (a, b, c or d changes) compute e as $e = \overline{a \cdot b + c \cdot d}$
• An A-O-I gate represented as a data flow type of relationship	• An A-O-I gate in pseudo code at behavioral level.

advances to the next time step & so on. The time step values refer to simulation time & not real time. One can redefine timescales to suit technology as & when necessary & carry out test runs.

In some cases the ckt itself may demand

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

as with data transfer & memory-based operations. Only in such cases sequential operations is enswred by the appropriate usage of sequential constructs from Verilog HDL.

Simulation And Synthesis.

Translation of the debugged design into the corresponding hardware circuit using an FPGA or an ASCII) is called "synthesis". the tools available for synthesis relate more easily with the gate level & data flow level modules.

- Many of the behavioural 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 & 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 & data flow levels (more commonly called the "RTL level").

Functional Verification:

Testing is an essential ingredient of the VLSI design process as with any hardware circuit. It has two dimensions to it - functional tests & timing tests. Both can be carried out with Verilog. Often testing or functional verification is carried out by setting up a "test bench" for the design. the test bench will have the design instantiated in it; it will generate necessary test signals & apply them to the instantiated design. The o/p from the design are brought back to the test bench for

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

for further Analysis. The Input Signal combinations, waveforms & Sequences required for testing are all to be decided in advance & the test bench configured based on the same.

The test benches are mostly done at the behavioral level. The constructs there are flexible enough to allow all types of test signals to be generated.

In the process of testing a module, one may have to access variables buried inside other modules instantiated within the master module. Such variable can be accessed through suitable hierarchical addressing.

Test Inputs for test benches:

Any digital system has to carry out a number of activities in a defined manner. Once a proper design is done, it has to be tested for all its functional aspects. The system has to carry out all the expected activities & not falter. Further, it should not malfunction under any set of input conditions. Functional testing is carried out to check for such requirements. Test inputs can be purely combinational, periodic, numeric sequences, random inputs, conditional inputs, or combinations of these. With such requirements, definition & def design of test benches is often as challenging as the design itself.

As the ckt design proceeds, one develops smaller blocks & groups them together to form bigger circuit units. The process is repeated until the whole system is fully built up. Every stage calls for tests to see whether

created with nitroPDF professional
download the free trial online at nitropdf.com/professional
www.smartzworld.com jntuworldupdates.org

the subsystem at that layer behaves in the manner expected. Such testing calls for two types of observations.

- study of signals within a small unit when test inputs are given to the whole unit.
- Isolation of a small element & doing local test to facilitate debugging.

Verilog has constructs to accommodate both types of observation through a hierarchical description of variables within.

Constructs for Modeling Timing Delays

Any basic gate has propagation delay & transmission delays associated with it. As the elements in the circuit increase in number, the type & variety of such delays increase rapidly; often one reaches a stage where the expected function is not realized thanks to an unduly large time delay. Thus there is a need to test every digital design for its performance with respect to time. Verilog has constructs for modeling the following delays:

- Gate delay
- Net delay
- path delay
- pin-to-pin delay.

In addition, a design can be tested for setup time, hold time, clock-width time specifications, etc. Such constructs or delay models are akin to the finite delay time, rise time, fall time, path or propagation

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

delays, etc., associated with real digital circuits or systems. The use of such constructs in the design helps simulate realistic conditions in a digital circuits. Further, one can change the values of delays in different ways. If a buffer capacity is increased, its associated delays can be reduced. If a design is to migrate to a better technology, the delay values can be rescaled. With such testing, one can estimate the minimum frequency of operation, the maximum speed of response, or typical response time.

System Tasks:

A number of system tasks are available in Verilog. Though used in a design description, they are not part of it. Some tasks facilitate control & flow of the testing process. The values of signals in a module can be displayed in the course of simulation. The tasks available for the purpose display them in desired formats Reading data from specified files into a module & writing back into a module & writing back into files are also possible through other tasks. Timescale can be changed prior to simulation with the help of specific tasks for the purpose.

A set of system functions add to the flexibility of test benches: They are of three categories.

- Functions that keep track of the progress of simulation time.
- Functions to convert data or values of variables from one format to another.
- Functions to generate random numbers with specific distributions.

There are other numerous system tasks & functions associated with file operations, PLA's, etc.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

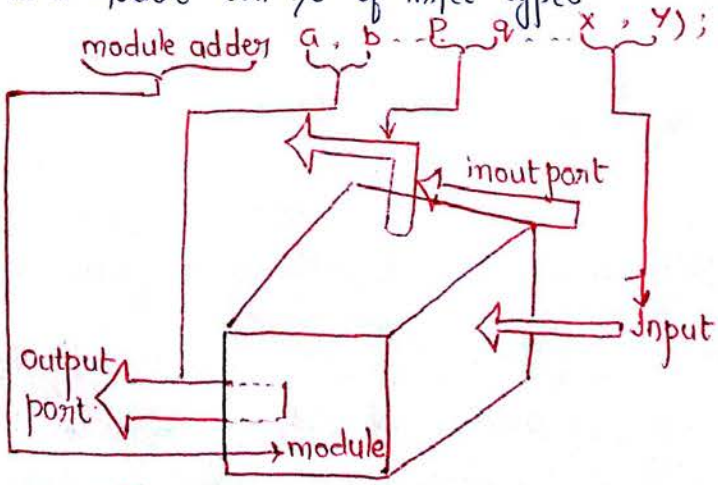
Programming language Interface (PLI)

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 & 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 & their Values written to output devices.
- Delay Values, logic Values, etc., within a module can be accessed & altered.
- Blocks written in c language can be linked to Verilog modules.

Module.

Any Verilog program begins with a keyword - called a "module". A module is the name given to any System considering it as a black box with Input & output terminals as shown in fig. the terminals of the module are referred to as "port". the ports attached to a module can be of three types.



Representation of a module as black box with its ports.

→ Input ports through which One gets entry into the module; they Signify the I/p Signal terminals of the module.

→ Output ports through which One exists the module; these Signify the o/p signal terminals of the module.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional

→ Inout ports: these represents ports through which one gets entry into the module or exists the module; these are terminals through which signals are i/p to the module sometimes; at some other times signals are o/p from the module through these.

Whether a module has any of the above ports & how many of each type are present depend solely on the functional nature of the module. Thus one module may not have any port at all, another may have only input ports, while a third may have only o/p ports, & so on.

All the constructs in Verilog are centered on the module. They define ways of building up, accessing, & using modules. The structure of modules & the mode of invoking them in a design

A module comprises a number of "lexical tokens" arranged according to some predefined order. The possible tokens are of seven categories.

→ White spaces
→ Comments
→ Operators
→ Numbers
→ Strings
→ Identifiers
→ Keywords.

In Verilog any program which forms a design description is a "module." Any program written to test a design description is also a "module". The latter are often called as "stimulus" modules or "test benches". A module used to do simulation has the form shown in fig. Verilog takes the active statements appearing between the "module" statement & the "endmodule" statement & interprets all of them together as forming the body of the module. whenever a module is invoked for testing or for incorporation into a bigger design module, the name of the module ("test" here) is used to identify it for the purpose.

created with nitroPDF professional
download the free trial online at nitropdf.com/professional