**CONTENT**

1. Acknowledgement
2. Introduction to VLSI
3. Introduction to Verilog and the Designing Tool used(Xilinx 14.1i)
4. Designing in Verilog HDL
5. Data types in Verilog HDL
6. Introduction Of Different Modelling Style
7. Gate Level Modelling
8. Dataflow Modelling
9. Behavioural Modelling
10. Finite State Machine (FSM)
11. Hardware Implementation
12. Useful Modelling Technique
13. Memory Modelling
14. Conclusion
15. References

**ACKNOWLEDGEMENT**

I am grateful to Cetpa Infotech Pvt. Ltd, Noida for giving me opportunity to carry out the project work in the area of VLSI during my training. I would like to also thank my institute, Jaypee Institute of Information Technology, Noida for giving permission and necessary administrative support to take up the training work at the training institute, Noida.

First of all, I would like to thank respected, Mrs. Ramandeep Kaur, (trainer) who gave me the opportunity to carry out this training under his guidance. He provided me with all the necessary, which were required for the completion of my training. I also thank my college library staffs for providing me with the books related to the training. A grateful acknowledgement goes to my project mates, who helped me with timely suggestions during the entire period of references and data collections.

**Noida**

**JUNE 2015 Jimmy Rajpal**

1. Introduction to VLSI

**Very-large-scale integration** (**VLSI**): is the process of creating an integrated circuit (IC) by combining thousands of transistors into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI lets IC designers add all of these into one chip.

**Levels of integration of Integrated Circuit(IC)**

1. SSI: Small Scale Integration :: Number of Transistors: less than 10
2. MSI: Medium Scale Integration:: Number of Transistors: between 10 to 100
3. LSI: Large Scale Integration :: Number of Transistors: approximately 10,000
4. VLSI: Very Large Scale Integration: : Number of Transistors: around one million
5. ULSI: Ultra Large Scale Integration :: Number of Transistors: much larger than one million

Classification of VLSI

Back End

Front End

Front End: considers all the logical designing and verificationpart, In simple words we can say all the work up to the Gate level or RTL Level designing and verification  considered as VLSI Front End Designing and Verification, We have multiple ways for logical designing of IC (Integrated Circuits)’s in VLSI Front End, For Example in early days when we have very less chip complexity, designers used Manual Logical Designing and they used concept of number system, Basic Gates Concept, K-maps, Boolean Mathematics, expression handling etc.

Back End: consider all the designing and verification part after logical designing means Gate level or RTL level designing, That may include Floor Planning, Place & Route, and All the foundry work like fabrication, packaging etc.

Introduction to Verilog and the Designing Tool used(Xilinx 14.1i)

**Hardware Description Language** (**HDL**): is a specialized computer language used to program the structure, design and operation of electronic circuits, and most commonly, digital circuits. A enables a precise, formal description of an electronic circuit that allows for the automated analysis, simulation, and simulated testing of an electronic circuit. It also allows for the compilation of an HDL program into a lower level specification of physical electronic components, such as the set of masks used to create an integrated circuit.

Design by HDL

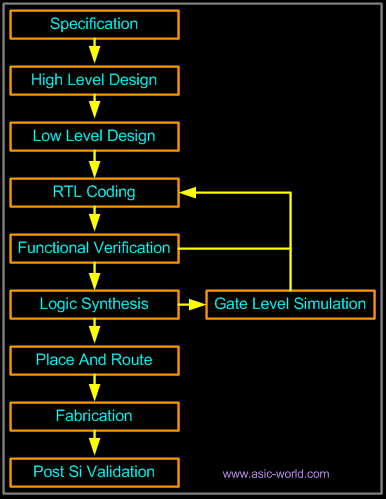
1. ABEL: advanced Boolean expression logic
2. AHDL: Altera hardware description language
3. VHDL: VHSIC(Very High speed integrated circuit) Hardware description language
4. Verilog HDL: Verifying Logic Hardware description language
5. System Verilog: extension of Verilog HDL and OOPS concepts

Xilinx ISE

It is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Xilinx ISE is a design environment for FPGA products from Xilinx, and is tightly-coupled to the architecture of such chips, and cannot be used with FPGA products from other vendors. The Xilinx ISE is primarily used for circuit synthesis and design, while the ModelSim logic simulator is used for system-level testing. Other components shipped with the Xilinx ISE include the Embedded Development Kit (EDK), a Software Development Kit (SDK) and ChipScope Pro.

Designing in Verilog HDL

* Design Methodology
  + Top-Down Methodology: The traditional method of electronic design is bottom-up. Each design is performed at the gate-level using the standard gates (refer to the Digital Section for more details). With the increasing complexity of new designs this approach is nearly impossible to maintain. New systems consist of ASIC or microprocessors with a complexity of thousands of transistors. These traditional bottom-up designs have to give way to new structural, hierarchical design methods. Without these new practices it would be impossible to handle the new complexity.
  + Bottom-up Methodology: The desired design-style of all designers is the top-down one. A real top-down design allows early testing, easy change of different technologies, a structured system design and offers many other advantages. But it is very difficult to follow a pure top-down design. Due to this fact most designs are a mix of both methods, implementing some key elements of both design styles.
* Design Simulation and Design Synthesis
* Verilog HDL Design Flow



* Keyword description in VERILOG HDL
* Module Description

DATA TYPES IN VERILOG HDL

* Data Values:
  + 0::low or false
  + 1:: high or true
  + X:: unknown
  + Z:: high impedance state
* Description of Data types
  + Net: The net data types represent physical connections between structural entities, such as gates. A net does not store a value (except for the trireg net). Instead, it must be driven by a driver, such as a gate or a continuous assignment.
  + Register: A register is an abstraction of a data storage element. The keyword for the register data type is reg. A register stores a value from one assignment to the next. An assignment statement in a procedure acts as a trigger that changes the value in the data storage element. The Verilog language has powerful constructs that allow you to control when and if these assignment statements are executed. These control constructs are used to describe hardware trigger conditions, such as the rising edge of a clock, and decision-making logic, such as a multiplexer.
* Data Representation
  + Scalar Data Representation: A net or reg declaration without a specification is one bit wide; that is, it is scalar.
  + Vector Data Representation: Multiple bit net and reg data types are declared by specifying [size]identity, and are known as vectors.
* Array Description Arrays are used to hold several objects of the same type constituting collection of vectors or scalars. for this, only reg identifier is used. It is represented as [size]identity[dimension].

INTRODUCTION OF DIFFERENT MODELING STYLE

* Gate level Modeling
* Dataflow modeling
* Behavioral Modeling
* Switch level Modeling

GATE LEVEL MODELING

* Logic Gate Primitive
* Gate Instantiation
* Design RTL from Logic Diagram
* Delays in Gate-Level Design
  + Rise Delay
  + Fall Delay
  + Turn off Delay

DATAFLOW MODELING

* Continuous Assignment statement
* Implicit Assignment statement
* Delay
  + Assignment Delay
  + Implicit Assignment Delay
  + Net declaration Delay
* Expressions
* Basic operators
* Verilog specific operators(case equality etc)
* Operands
* Operator Precedence

BEHAVIORAL MODELING

* Structured Procedural Statements
  + Always Statements
  + Initial Statements
* Blocking Statement
* Non blocking Statement
* Timing Control Statement
  + Delay Based Timing Control
  + Event Based Timing Control
* Conditional statements
  + If-else statements
  + Case statements
* Loops
  + While loop
  + For loop
  + Repeat loop
  + Forever loop
* Block Statements
  + Parallel block
  + Sequential block