Suggested Teaching Guidelines for

Verilog HDL PG-DVLSI March 2024

Duration: 34 class room hours + 86 Lab hours

Objective: Introduce Verilog HDL Concepts and explore verification capability.

Prerequisites: Good knowledge of Advanced Digital Design,
awareness of HDLs and verification.

Evaluation method: CCEE Theory exam – 40% weightage
Lab exam – 40% weightage
Internal exam – 20% weightage

<u>List of Books / Other training material</u>

Courseware:

Reference:

1. Verilog HDL: A Guide to Digital Design and Synthesis by Palnitkar, Samir
2. IEEE 1364-2005 Verilog Language Reference Manual
3. The Verilog Hardware Description Language – Thomas & Moorby
4. Verilog HDL Synthesis – J Bhaskar

Session 1 & 2
Lecture: Verilog Basics

- Introduction to Verilog
  - History
  - Sample design and sample testbench.
- Getting Started to Verilog
  - “module”
  - Component Instantiation
  - Types of ports
  - Built in primitives
  - Modeling styles: Behavioral, Dataflow, Gate level, Switch level modeling
  - Data Types
  - Minimum, typical and maximum delay triplet
  - Operators
  - Continuous assignment

Session 3 & 4
Lecture: System Tasks and Procedural Flow Control

- System Tasks and Functions
  - $display, $write, $strobe and $monitor
  - $time
- Procedural assignment its type
- “initial” and “always” statements

- “begin-end” and “fork-join” blocks
- “wait” statement
- Procedural Flow Control
  - “if-else” statement
  - Parallel and Full case

Session 5
Lecture: Looping Statements

- Loops
  - forever
  - repeat
  - while
  - for

Session 6
Lecture: Additional Features

- Task and Function
  - automatic
- Parameter Overloading
  - defparam and # operator
- Compiler directives
  - `define and its variant
- Disable

Session 7 & 8
Lecture: Modeling Delays using Specify Block

- `timescale, $timeformat, $printtimescale
- Comparision between $time, $stime, $realtime
- Modeling Delays
  - Lumped Delay
  - Distributed Delay
  - Pin to pin delay
- Specify block
  - Specparam
  - Full and parallel connection
  - Edge Sensitive path
- Pulse Filtering behavior (PATHPULSE$)

Session 9
Lecture: Timing Parameters

- Timing checks
  - $setup, $hold, $setuphold
  - $recover, $width, $skew
  - $fullskew, $removal, $recrem, $timeskew
- Timing Check with Notifier Behavior

Session 10 & 11

Lecture: Race Condition, Assignments and SEQ

- Race Condition
- Blocking and Nonblocking assignments guidelines
- Determinism, non-determinism and race condition
- The Stratified event queue. (Examples to explain relation between coding guidelines and SEQ)

Session 12 & 13
Lecture: User Defined Primitives (UDPs)

- User Defined Primitives (UDPs)
- Combinational UDPs
- Level-sensitive sequential UDPs
- Edge-sensitive sequential UDPs
- UDP instance
- Configuration introduction

Session 14 & 15
Lecture: VCD and PLI

- Programming Language Interface (PLI)
    - tf_ and access_ routines of PLI
- Value Change Dump (VCD)

Session 16 & 17

- Introduction of FSMD,
- Block Diagram of FSMD
- Data path, Control path, Register Transfer (RT) operation,
- ASMD chart, Design and Verilog Implementation of FSMD case studies

Lab Assignments:

1. DFF Modeling
2. Asynchronous memory modeling (Using a Datasheet)
3. Assignment should be based upon use of basic primitives and operators.
4. Design D Latch using logic gate. (Hint: design D Latch from SR Latch).
    - NAND Gates only
    - NOR Gates only
5. Design T Flip Flop using Latches having asynchronous set and clear.
6. Design frequency divide by 8 circuit using T Flip Flop. Identify if there would be any problem if reset and set were synchronous.
7. Design parallel in parallel out 8-bit rotational right shift register with a clock, synchronous load input using
   a. Blocking
   b. Non-Blocking.
8. Write a function to perform XOR operation on two inputs A and B (1- bit each). Using this function, write a parity function which accepts m-bit input and returns parity. Design a module which accepts 16-bit input din and returns 1-bit output Parity using parity function.

9. Design 16 x 16 bidirectional memory. Use gate primitive bufif1or bufif0 to design bidirectional data bus. Use Tri-state buffer only at the data output from the memory, it should not be present at data input to memory.
10. Design an 8x8 sequential multiplier. The multiplier has asynchronous reset, synchronous load and output valid signal. Use the concept of self-testing Test Bench to verify that multiplication result is correct.
11. Design and verify synchronous FIFO (128 x8). The width and depth of the FIFO should be configurable. Use separate Verilog file for defining width and depth. Verify your design for 8 x 4 and 16 x 8 FIFO.
12. Design UDP for 8:1 Multiplexer
13. Design UDP for positive edge triggered D Flip Flop with asynchronous reset (clear) and asynchronous set (preset).
14. Design mod 3 counter using Verilog primitives and D Flip Flop UDP designed above.
15. Design a Priority resolver. The resolver receives four requests req_a, req_b, req_c, req_d and generates four grants gnt_a, gnt_b, gnt_c, gnt_d. At initial stage A has higher priority over B, B has higher priority over C and D has the least priority. Requests are sampled only if busy is low. At the later stages, resolver uses Least Recently Used Algorithm for generating grants based on requests received. On transcript display the devices that are generating request and the device that is provided grant.