

Summary of Learning Objectives

1: Introduction to Digital Design with Verilog HDL

After this lecture you should be able to:

- define a module with single- and multi-bit **logic** inputs and outputs;
- write expressions using **logic** signals and operators;
- write Verilog numeric literals in binary, decimal and hexadecimal bases.
- use **assign** statements to generate combinational logic;
- use initialized unpacked arrays to implement arbitrary combinational logic functions

2: Flip-Flops and Registers

After this lecture you should be able to:

- convert between high/low logic levels and true/-false truth values for active-high and active-low interfaces
- predict the relationship between the input and output waveforms of D flip-flops
- draw the output waveforms for registers, counter and shift registers
- write Verilog descriptions for these
- identify specifications on a timing diagram
- identify a specification as a requirement or guaranteed response

3: System Verilog

After this lecture you should be able to: predict the size and value of a Verilog expression that uses the signals, constants and operators described below; and predict the flow of control between the statements described below.

4: HDL Idioms

After this lecture you should be able to convert a Verilog description into a block diagram, and convert a block diagram into a Verilog description.

5: Common Circuits

After this lecture you should be able to: write Verilog and draw a block diagram for a timer, clock divider, edge detector and synchronizer.

Updated Feb 2.

6: State Machines

After this lecture you should be able to: design a state machine based on an informal description of its operation, document it using state transition diagrams and tables, write a synthesizable Verilog description of it and convert between these three descriptions.

7: Interfaces

After this lecture you should be able to: classify an interface as serial or parallel, synchronous or asynchronous, uni- or bi-directional and explain the advantages of each; draw the schematic or write the Verilog for a synchronous serial transmitter or receiver; extract the data transmitted over an SPI interface from the interface waveforms.

8: Timing Analysis

After this lecture you should be able to be able to apply the terms defined in this lecture and do calculations involving clock rate, propagation delays and setup/hold time requirements.

9: Memory System Design

After this lecture you should be able to: answer short questions about different memory types and memory terminology; identify read/write timing specifications from a timing diagram (as for flip-flops); draw a schematic showing how memory ICs are connected to increase the data bus size and the total amount of memory; design (combinational) address decoding logic to place memory at specific address ranges.

10: Verification

After this lecture you should be able to select an appropriate verification strategy including: selecting simulation or hardware testing; stimulus-only or self-checking testbenches; selection of test inputs; use of “known-good” models; unit testing; regression testing; distinguish between functional (RTL) and gate-level (timing) simulations; use delays and event controls to generate waveforms in a System Verilog testbench.

11: Implementation of Digital Logic Circuits

After this lecture you should be able to: differentiate between logic levels, truth values and numbers; determine from a data sheet: if an input or output voltage would be high, low or invalid and calculate noise margin; state which transistors are on and off in a CMOS totem-pole output; determine the direction of current flow between driver and receiver; compute the effect of frequency and voltage changes on the power consumption of CMOS logic circuits; determine the RC time constant and current consumption of an open-collector output; design simple circuits to convert between logic levels; distinguish between DIP, TQFP and BGA packages.

12: Programmable Logic Applications and Architectures

After this lecture you should be able to: explain the growth of digital electronics; select software versus hardware and PLDs versus ASICs to solve a particular problem; explain the terms: Moore’s Law, ASIC, CPLD, FPGA, feature size, VLSI, fabless, wafer, die, NRE, FPGA, LE and LUT.