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 Verilog numeric literals in binary, decimal and hexadecimal bases; declare arrays and arrays of arrays; evaluate the value and length of expressions containing `logic` signals, arrays, numeric literals and the operators described below; use `assign`, `always_ff`, and component instantiation statements to create combinational logic, registers, and to instantiate one module in another.

2: 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.

3: Applications of State Machines

After this lecture you should be able to write Verilog to implement: a shift register, edge detector, sequence detector, and state machines with interdependent state transitions.

4: Applications of State Machines

After this lecture you should be able to implement the following: shift register, edge detector, sequence detector,...

5: More Verilog

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, declare modules with parameters and ports, and instantiate modules using positional, named and wildcard parameters and signals.

6: Simulation

After this lecture you should be able to write a testbench that can: set initial values, generate clocks, read test vectors from a file, display values, and terminate on a condition.

7: Interfaces

After this lecture you should be able to: classify an interface as serial or parallel and uni- or bi-directional and explain the advantages of each; ; determine when data is transferred over a ready/valid interface; draw the schematic or write the Verilog for an SPI transmitter or receiver; convert data transmitted over an SPI interface to the interface waveform(s) and extract the data from these waveforms.

8: Timing Analysis

After this lecture you should be able to be able to: identify features and specifications on a timing diagram, identify a specification as a requirement or guaranteed response, apply the terms defined in this lecture, and do calculations involving clock rate, propagation delays and setup/hold time requirements.

9: Implementation of Digital Logic Circuits

After this lecture you should be able to: state which transistors are on and off in a CMOS totem-pole output; determine the direction of current flow between driver and receiver; determine from a data sheet: if an input or output voltage would be high, low or invalid and calculate noise margin; 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; describe the causes and consequences of ESD; design simple circuits to convert between logic levels; distinguish between DIP, TQFP, BGA and CSP packages.

10: Analog Interfaces

After this lecture you should be able to solve problems involving: sampling rate vs signal frequencies; number of bits vs resolution and quantization SNR; clock rate, sample rate and resolution for binary-weighted DAC, PWM DAC, flash ADC, SAR ADC.

11: 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.