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 width 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: Sequential Logic and State Machines

You should be able to design and document state machines using: an informal description, truth tables, state transition diagrams, and synthesizable Verilog. You should be able to convert between any of these descriptions. You should be able to select an appropriate state encoding.

3: Examples of State Machines

After this lecture you should be able to write Verilog to implement: a sequence generator, a sequence detector, and circuits that combine state machines.

4: Hierarchical Design 4:

After this lecture you should be able to: declare modules with parameters and ports, and instantiate modules using positional, named and wildcard parameters and signals.

5: 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.

6: 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 and battery life; 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.

7: Digital 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: 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.

9: Static 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.

10: Implementation Technologies

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