EE3610 Tentative Course Schedule – Fall 2013

Date	Reading	Subject	Homework	Lab
8/26	1.1-5	Combinational Logic Review	#1) 1.2, 3, 4a	Lab 0 ISE Webpack
8/28	1.6-11	Sequential Logic Review, Lab 1	#2) 1.11a, 15, 20a, 23, 26	
9/2		Labor Day		Lab 1 – Serial Receiver
9/4	2.1-7 2.9-12	Design Process, Modeling in VHDL (Review), Lab 1 Example.	#3) 2.5, 8, 10, 13	
9/9	2.13-16	VHDL arithmetic, loops, registers, counters, Lab 1	#4) 2.21, 23, 24	Lab 2 – Serial Transmitter.
9/11	2.17-2.19	VHDL state machines, arrays, test benches	#5) 2.30a, 36b, 40 (no delays), 45	
9/16	3.1-3.2	Simple Programmable Logic, Lab 2	#6) 3.3, 11, 12klm	Lab 3 – PS/2 Keyboard interface (Lab 1 late after this period)
9/18	3.3-3.4 4.1-4.2	Complex Logic, Example: BCD Adder	#7) H1, 4.13	
9/23	4.3, 4.6-4.7	Example: Fast Carry Adder, Debouncing, Lab 3	#8) H2	Catch up (Lab 2 late after this period)
9/25	4.8-4.9			
9/30	4.11-12	Example: Key scanner, Lab 4		Catch up
10/2	5.1-5.3	Example: Sequential Divider, SM Charts	#10) 5.1, 5, 15a	
10/7	5.5-5.6	Microcode, linked FSMs	#11) H4	
10/9		Review/Catch-up		
10/14		Mid-term Exam		Lab 4 – Video Timing (Lab 3 Due)
10/16		Lab 4		
10/21		Go Over Exam, Lab 5	#12) omit	Lab 5 – Character Generator (due 11/7)
10/23	6.1-10	LUTs, FPGA Resources	#13) 10,15, 23, H5	
10/28	6.11-12	Synthesis	#14) 6.34-36	
10/30	7.1	Floating Point Representation	#15) 7.2-3 (i, v, viii) 7.4-5 (i, ii)	
11/4	8.1-3	Functions/Attributes, Lab 6	#16) 8.4, 6	Lab 6 - Character memory (due 11/21)
11/6	8.4-6, 9-11	Overloading, Resolvers, Generics and Association	#17) H7	
11/11		Array Multipliers, Pipelining	#9) H3	
11/13	9.1-3	MIPS ISA	#18) 9.1, 5i-iii, 7i-iii	
11/18	9.4	Implementation of MIPs subset, Lab 7	#19) H8, H9	Lab 7 – Final Integration (due 12/5)
11/20	9.5	VHDL implementation of MIPS		
11/25		Classic 5-stage pipeline	#20) Handout	
11/27	10.1-2	Testing combinational & sequential logic		
12/2	10.3-4	Scan path testing	#21) 10.3, 10, 15	Catch-up
12/4		Catch-up		

Homework Not From the Text

H1. Design, Simulate & Synthesize a 2-bit ripple-carry adder. Turn in VHDL code for the adder module, the simulation output (waveform) with at least 8 test cases and the constraints file for synthesis. You will be on your honor to test your synthesized design on your Spartan board. For this homework, you may work with your lab partner(s).

H2. Design and simulate a 16-bit fast carry adder (See p197 of the text). Make sure your test bench tests at least 16 cases, including boundary cases (maximum and minimum values with and without carry). Write your test bench in such a way that if your code has an error, the test bench reports it. Use arrays to hold input and output values (test vectors) as shown on page 121 of the text. Turn in vhdl module(s), test bench and simulation results (waveform). Make sure to use the hexadecimal radix for the simulation output so both addends and the sum can be readily seen.

H3. Design and simulate a 4-bit x 4-bit carry-save array multiplier with one pipeline stage after the first two banks of adders. Use the same guidelines for the test bench as you used on H2, but remember that the product will come 1 clock cycle after the multiplier and multiplicand are input.

H4. Do problem 5.17, except use the 1-address microprogramming structure in Figure 5-33. Note. Make sure to use the MUX assignments shown in the table in Problem 5.16, Start your microcode program with S0. (Hint: You will need to make a duplicate S2 state because there are 2 paths to S2 when X2 = 0.) For part (d), refer to Figure 3-8.

H5. Explain why you would use the modified 1-hot state assignment given in Problem 6.26. (Hint: Consider Problem 6.27.)

H6. Explain the difference between distributed (LUT based) memory and dedicated memory, then explain how to get the synthesis tool to infer one type of memory or the other.

H7. Write a VHDL package that defines a new bit-like type called x01z that may take on the values ‘0’, ‘1’, ‘X’ and ‘Z’. Use Figure 8-12 as a start. Overload the functions “and” and “or” and write a function to convert x01z to std_logic. Write a VHDL module that takes 2 inputs of type x01z and generates 3 outputs of the same type. The first output is the OR of the two inputs, the second is the AND of the two inputs. Using two concurrent assignment statements, assign the OR and the AND result to the third output. Write a test bench that tries all 16 possible cases. Use your conversion function to convert the inputs and outputs to std_logic so they can be displayed in the simulator. Simulate the test bench. Turn in a copy of the VHDL package and the results of the simulation. (Hint: the third output should be ‘X’ (red) everywhere except where the inputs are both ‘0’ or both ‘1’.)

H8 Use VHDL to model an ALU that takes two 8-bit inputs, A and B and an op-code. (The op-code may have any number of bits you wish). The ALU generates an 8-bit result based on the op-code (carry and overflow are ignored). The ALU must support at least the following 7 operations: (A+B), (A-B), (A or B), (A and B), (A xor B), (A), (B). Turn in a copy of your ALU module and a simulation that shows all 7 operations when inputs A = 9D and B = 56. Make sure the radix is set to hexadecimal.

H9. Design a register file that holds 4, 8-bit registers. The register file must have one write-port and 2 read ports. The register named by the write-port is overwritten on the rising edge of the clock. Write a behavioral model in VHDL. Initialize register 0 to X”9D”, register 1 to X”56”, and all others to “XXXXXXXX”. Write a structural model in VHDL that connects your register file to the ALU you designed in H8. Connect read ports of the register file to the inputs of the ALU and connect the output of the ALU to the write port of the register file. Write a test bench that provides (a) the ALU operation, and (b) the register file addresses so that your VHDL model computes 0x9D + 0x56 and stores it in register 2 and then ANDs that result with 0x9D and stores it in register 3. Simulate and turn in a copy of your results showing the contents of all four registers. Also turn in copies of your VHDL modules.