Professor Fearing EECS150/Problem Set 10 Fall 1998

Due at 10 am, Fri. Nov. 6 in box outside 218 Cory Hall

(This problem set may be done individually or by a group of two people maximum.

1. Synchronous Design Rules (35 pts)

For each of the seven synchronous design rules on the *Debugging Checklist* handout, design a simple test case and draw a simple timing diagram which illustrates how violating the particular design rule could lead to design failure using the Xilinx parts. For example, for Rule 3, "Never gate the clock", show how ANDing the clock with another signal, even a glitch free signal, could generate an output glitch.

2. Microcode for FSM emulation (30 pts)

In this problem you will emulate a Moore type FSM for a sequence recognizer, using the micro-programmed control unit (MCU) shown below. The MCU should assert OUTPUT once for every (possibly overlapping) sequence "101" detected in the synchronized input data stream Z. The MCU clock is 3 times as fast as the input bit rate of Z, thus you have time to execute three instructions for every input bit processed. Let Z, and $\overline{Z}$ be connected to D0, and D1 respectively on the 4:1 mux. Write a micro-program which will emulate the sequence recognizer listing your program in mnemonic and numeric form. (Note that branches only occur when test variables are asserted; do not add any inverters. Also note that OUTPUT may not necessarily change states on the same clock edge as Z changes, and OUTPUT could be asserted from one to three clock cycles.) Show a timing diagram with the output register OUTPUT, CLOCK, microPC LOAD, Output Register CE, and ROM O7 for the input sequence Z = 010101. Label each clock period with the current address (micro-program counter contents).

3. Microcode for CPU control (35 pts)

Using the micro-programmed control unit below as a starting point, draw a detailed block diagram of an MCU (making sure to specify widths of ROM and registers) which could control the processor of Ex11.7 in the text, using a vertical micro-code format. Specify which controller outputs need to be latched (if any). List, in symbolic form, the micro-code needed to implement the ADD and RSR instructions.