Professor Fearing EECS150/Problem Set 8 Fall 2003

Due at 10 am, Fri. Oct. 31 in box outside 125 Cory Hall

Quiz 2: Thurs. Nov. 6 in class.

(This problem set may be done individually or by a group of two people maximum. A group of two should hand in a single copy of the solution with both names affixed).

Reading: Ciletti 7.3

1. RDI Instruction Execution (40 pts)

For the data path of Ciletti Fig. 7-10 (as discussed in class), a new instruction $RDI\#\texttt{value}R_{DEST}$ is to be added. The 2 byte Read Immediate instruction loads the destination register R0 ... R3 with the value of the second byte of the instruction. ($RAM[PC] \rightarrow R_{DEST}$).

a. List necessary register transfer operations for execution of this instruction.

b. Show with a timing diagram the operation of the processor during the *RDI* instruction. Show CLOCK, all relevant control signals, Bus1 and Bus2 data values, memory address and data, and indicate the clock edge when a particular register transfer operation occurs.

c. Draw an ASMD diagram for the RDI instruction execution states.

d. Write Verilog for the *RDI* portion of the instruction execution states, following the Control_Unit module style given in the text.

2. SWAP Instruction Execution (30 pts)

For the data path of Ciletti Fig. 7-10 (as discussed in class), a new instruction $SWAP\ R_{SRC},\ R_{DEST}$ is to be added. The 1 byte SWAP instruction swaps the source and destination register contents. (Hint: use a temporary register.)

a. List necessary register transfer operations for execution of this instruction.

b. Show with a timing diagram the operation of the processor during the *SWAP* instruction. Show CLOCK, all relevant control signals, Bus1 and Bus2 data values, and indicate the clock edge when a particular register transfer operation occurs.

c. Draw an ASMD diagram for the SWAP instruction execution states.

3. One Hot vs. Consecutive State Encoding (10 pts)

The Control_Unit Module for the RISC SPM in the text uses 12 consecutive states. In the Xilinx architecture, what would the drawbacks be to using one hot encoding instead? (Hint: #1 its not the number of FF. #2 consider `default:` #3 consider expansion of controller.)

4. ROM Based FSM Design (20 pts)

Design a Moore FSM that counts nickels (N) and Dimes (D), and will assert an output OPEN for one clock cycle when at least $0.40 have been inserted. After global Reset or OPEN, the system returns to a waiting state. The inputs N.H and D.H, are synchronized to the clock.

Draw a state diagram and state table, and design the FSM using D type flip flops in a ROM based FSM. Make sure your FSM is self-starting. Both NSD and OD should be in the same ROM. List ROM contents in Hexadecimal. Use ROM Out[3:0] for the NSD, and state assignment $Q_3Q_2Q_1Q_0$ should be the equivalent number of nickels entered.