Name: ______________________________

CS 4/55111

Final Exam

VLSI Design

Monday 8 May 2006

1. **VHDL divides the description of a module into an Entity and an Architecture section.**

   a. **What is the purpose of each of these two sections? (10 points)**

   The Entity section specifies the name of the entity as well as the names and types of its input and output ports.

   The Architecture section specifies the name of this architecture for a specific entity and then contains either a structural or behavioral description of the entity.

   Essentially, the Entity section specifies the interface to the entity and the Architecture section specifies the functionality or composition of the entity.

   a. **Would it ever make sense to have multiple alternative Architecture sections for a single Entity section? Explain. (5 points)**

   Yes. A behavioral description might specify the functionality of the entity for simulation purposes, whereas a structural description might specify the composition of the entity in terms of lower-level components for documentation purposes. Or alternative behavioral descriptions might be written to explore tradeoffs between functionality and chip area.

2. **What is a STD_LOGIC type? (10 points)**

   VHDL uses the STD_LOGIC type to model represent logic values. It includes not only values of 0 and 1, but also Z, U, X, –, L, W, and H which are needed for some designs.

3. **Are statements in VHDL concurrent (like in AHDL) or sequential (like in C or Java)? Explain. (10 points)**

   VHDL statements are generally concurrent, with the exception of statements inside a process, which are executed sequentially.

Name: ______________________________

4. **Consider the following two VHDL code fragments:**

```
PROCESS
BEGIN
    WAIT UNTIL (Clock'EVENT AND Clock='1');
        IF reset='1' THEN
            Q2 <= '0';
        ELSE
            Q2 <= D;
        END IF;
END PROCESS;
```

```
PROCESS (Reset, Clock)
BEGIN
    IF reset='1' THEN
        Q3 <= '0';
    ELSEIF (Clock'EVENT AND Clock='1') THEN
        Q3 <= D;
    END IF;
END PROCESS;
```

How do these two code fragments differ? Be specific. (15 points)

In the leftmost process, the process begins and then waits for a rising clock edge ("WAIT UNTIL…). At the rising clock edge, it checks first for an active reset signal, and if it finds ones it stores 0 and sends that 0 through to output Q2. If reset is low it stores D and sends that D value through to output Q2. Thus it specifies a rising-edge triggered D flip-flop with synchronous reset.

In the rightmost process, the process is "sensitive" to Reset and Clock, meaning that a change on one of those signals activates the process. If Reset activated the process, it stores 0 and sends that 0 through to output Q3. If a clock event activated the process and that event was a rising clock edge, it stores D and sends that D value through to output Q3. Thus it specifies a rising-edge triggered D flip-flop with asynchronous reset.

5. **It is easy to write code in VHDL that is syntactically correct, but that will not generate the expected hardware once it is compiled onto an FPGA. Explain, preferably with an example or two. (10 points)**

Some statements in VHDL, such as ASSERT statements or file operations, may be useful for simulation but are not synthesizable. Other statements have to be used in the proper way to get the expected results — for example, values must be assigned to all outputs in each path through an IF or CASE statement, else a (probably unwanted) latch will be added. Or a value written in one state of a state machine can not be read until the next state.

6. **Compare the relative sizes of the programming points in field-programmable logic device that use antifuse, EEPROM, and SRAM programming technologies. (10 points)**

An antifuse programming point is the size of a via, which is smaller than a transistor. An EEPROM programming point is slightly larger than a transistor. An SRAM programming point is the size of 4-6 transistors.

Name: ______________________________

7. **How does the local interconnect and sharing of functionality between the macrocells inside an Altera MAX LAB compare to that between the LEs within an Altera FLEX LAB? (20 points)**

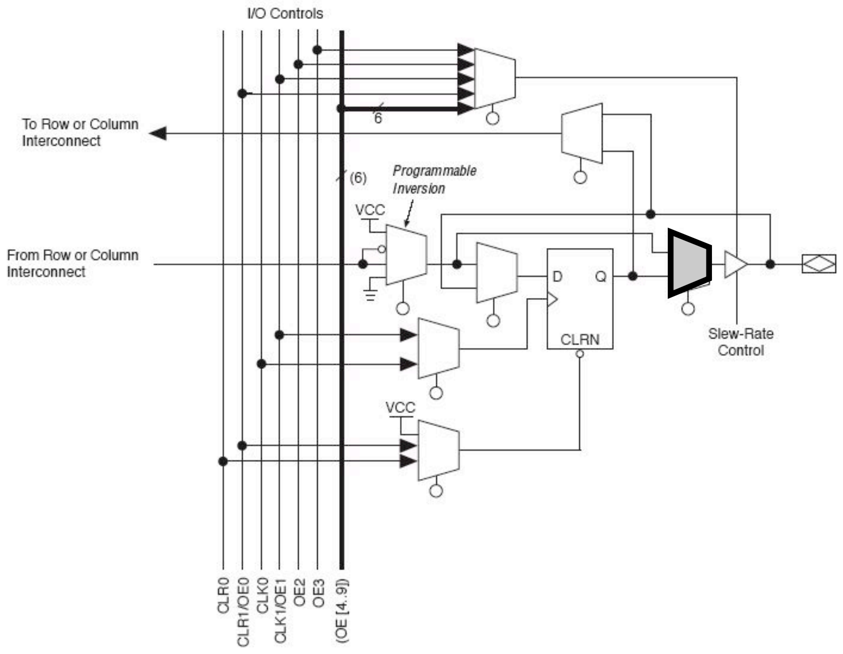
In the MAX LABs, the macrocells can use sharable expanders to broadcast values to other macrocells inside the LAB, and can use parallel expanders to borrow product terms from adjacent macrocells to provide more inputs to their OR gates.

In the FLEX LABs, the LEs use the local interconnect to send values from the outputs of LEs to the inputs of LEs within the LAB, and use the LE's cascade and carry chains to combine LEs in a LAB into multiple-bit arithmetic and wider logical functions, respectively.

So both MAX and FLEX provide local broadcast of values and combining cells for wider logical functions, but only FLEX provides cascade mode for multiple-bit arithmetic.

(As a matter of strategic test-taking, note that this was a 20-point question, yet some people wrote answers here that were much shorter those they wrote for the 10-point questions!)

8. **Consider the FLEX 8000 I/O element shown to the right. What functionality is provided by the multiplexor on the right side of the figure (shown shaded with a thick border)? Be specific. (10 points).**



That multiplexor sends its output to a tri-state device that in turn connects to an I/O pin, so if the tri-state device is activated it sends the output of the multiplexor to a pin that acts as an output pin.

The multiplexor's lower input comes from a register that holds a value received from the row/column or from the pin acting as an input pin.

The multiplexor's upper input comes directly from the row/column, bypassing the register.

In summary, the multiplexor chooses either the data coming directly from the row/column or the data from the register, and passes that data on to the tri-state device and the output pin.