Name: ______________________________

CS 4/55111 # Final Exam **VLSI Design**

Monday 6 May 2002

---

1. **AHDL separates the description of a component into a Subdesign section and a logic section (the body), while VHDL separates it into Entity and Architecture sections.**

   a. **What is specified in each of these sections? (5 points)**

      The Entity section specifies the component's inputs and outputs (name, bit width, etc.), while the Architecture section specifies the implementation of the component.

   b. **Why is the description divided this way? (5 points)**

      One answer is all the usual reasons from programming languages: someone using the component only needs to know the interface and not the implementation details, etc.

      Another answer is more hardware oriented: there may be multiple implementations at different levels (register-transfer level vs. gate level) or for different purposes (e.g., one for efficient simulation, and one as input for logic synthesis).

2. **Using AHDL, VHDL, and Altera's MAX+PLUSII, state machines can be specified in a variety of ways. Briefly describe the various options available. (10 points)**

   Taking advantage of the logic synthesis tools, state machines can be specified at a high level of abstraction. In AHDL, state machines can be specified as a MACHINE WITH STATES together with a TABLE specifying outputs and next states or a CASE statement specifying the individual states, outputs, and next state logic. In VHDL, state machines are usually specified using CASE statements inside a PROCESS.

   If desired, state machines can also be designed in the traditional way, using flip-flops to hold the state, and combinational logic to provide the output and next state logic, though most people would consider this method overly tedious and error-prone.

   Altera's MAX+PLUSII also provides a novel alternative to using hardware description languages for design input, allowing state machines to be specified using the waveform editor, as was shown in class in the ChipTrip example.

3. **Consider the two VHDL processes below, each of which will be implemented by a D flip-flop during logic synthesis. How will the two resulting flip-flops differ, and why? Be specific. (15 points)**

```
PROCESS
BEGIN
    WAIT UNTIL ( Clock'EVENT AND Clock = '1' );
        IF reset = '1' THEN
```

Name: ______________________________

```
            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;
```

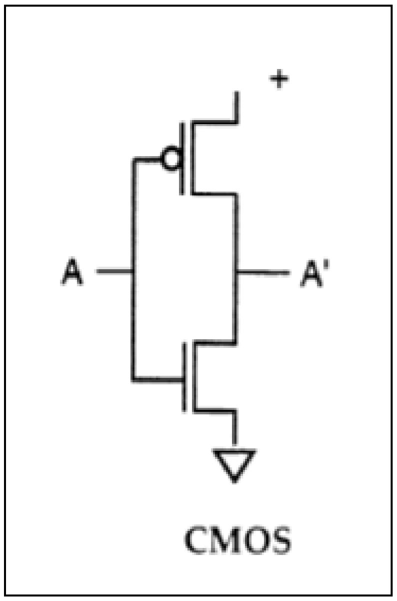
The top process waits for a rising clock edge, and then looks for a reset signal, meaning it implements a synchronous reset (if there is no reset, then it acts as a D flip-flop storing a value).

The bottom process is sensitive to both Reset and Clock, so if Reset changes at some time other than on a rising clock edge, the process will be immediately activated and the flip-flop reset, meaning it implements an asynchronous reset.

4. **CMOS transistors are made primarily of silicon, which isn’t a very good conductor. Why make transistors out of silicon, instead of a good conductor like copper? (10 points)**

Because in digital circuits a transistor is used as a switch — under some circumstances it conducts, while under others it does not. Making it out of a semi-conductor like silicon allows this switching to occur in a controlled fashion. Making the transistor out of copper would mean that it would always conduct, so it would be useless as a switch.

5. **Consider the device built from two CMOS transistors as shown at the right. Briefly describe what function it performs, and how it works. (15 points)**



If A is ‘0’, the upper (p-channel) transistor is turned on, which pulls A’ up to VDD, while the lower (n-channel) transistor is turned off, having no effect on A’. If A is ‘1’, the upper (p-channel) transistor is turned off, having no effect on A’, while the lower (n-channel) transistor is turned on, pulling A’ down to GND. In both cases A’ becomes the complement of A, meaning the circuit implements an inverter.

6. **The Altera MAX 7000 family provides sharable expanders and parallel expanders within each macrocell. What are these expanders, and how are they used? (10 points)**

Name: ______________________________

The sharable expander pterm can be inverted and fed back around to act as an input to any macrocell in that LAB. The parallel expanders allow some or all of the pterms in a macrocell to be "borrowed" by an adjacent macrocell in that LAB, connecting them to the inpt of the borrower's OR gate and allowing a macrocell to have up to 3 sets of parallel expanders (20 pterms for the OR gate). Using these two types of expanders and Boolean algebra, a function of more than 5 pterms can often be implemented by a macrocell.

7. **The primary difference between the Altera FLEX 8000 family and FLEX 10K family is the presence of Embedded Array Blocks (EABs) in the FLEX 10K. What are these EABs, and how are they used? (10 points)**

   Each EAB can be used to implement either memory or logic. When used to implement memory, each EAB corresponds to 2048 bits of RAM, and can be used to implement RAM, dual-port RAM, ROM, or FIFO memory. When used to implement logic, each EAB provides a very large lookup table, roughly equivalent to 600 gates, so is ideal for implementing complex logic functions.

8. **Consider the Xilinx Virtex family of FPGAs, introduced on the attached three pages. How does this family compare to the FPLDs we discussed in class? These few pages obviously do not provide much detail, but provide the best comparison you can, given this brief overview. Note that this question counts more than any other question on the exam. (20 points)**

   The Xilinx Virtex family is comparable to the Xilinx XC4000 family, and roughly comparable to the Altera FLEX, in that all three families are based on SRAM programming and LUTs. Like the FLEX, the Virtex has fast carry logic and cascade logic, while the XC4000 has only the carry logic. The Virtex is very different from the Altera MAX family.

   The overall organization of the Virtex is flat, with one level of hierarchy like the XC4000, and unlike the FLEX with its LABs, each of which contains LEs. Also like the XC4000, the Viretex has IOBs all around the chip, instead of at the ends of the interconnect like the FLEX. Like the XC4000, the Virtex allows each LUT to be treated as memory, instead of treating only the EABs as memory in the FLEX.

   Otherwise, compared to the XC4000 the Virtex is larger and faster. It also has some interesting innovations, such as some options for hot-swapping and an on-chip temperature sensor.