Reconfigurable Computing: Homework 1
(CPRE 583, Fall 2011)
Due: Fri 9/9/2011 (midnight)

Things to keep in mind:

I. You will lose points for not commenting your VHDL.

II. You will lose points for not using dividers to group signals that belong to a given component together in your simulations.

III. You will lose points for not cleanly indenting your VHDL (I suggest using spaces as opposed to tabs because different editors treat tabs differently. You never know what text editor someone is going to open your design with.)

1. See the HW1 "Tools Overview" slides for a quick tutorial on:
    a. Installing NX to access on campus Linux computers from off campus
    b. VERY basic Linux
    c. ISE, Modelsim (used for part 3 of this homework)
    d. Impact (used for MP1)

2. Getting Started: Getting the files
    a. Download HW1.tar.gz (or HW1.zip) to your U: drive (i.e. ISU network drive)
        i. I suggest that you make a directory in your U: drive for storing your CPRE 583 work.
    b. Download Xilinx_12_4_src.txt (see HW1 links)
        i. Use NX to log into a Linux machine from off campus, or log into any Linux machine located on the second floor of Coover.
        ii. source <location of the src file>/Xilinx_12_4_src.txt (e.g. source /home/username/Xilinx_12_4_src.txt)
        iii. you must source this file each time you open a new terminal. It tells the terminal where to find the tools.
    c. As you proceed through the homework, for each problem that requires writing VHDL or simulation make a directory call HW1_<problem label>. For example, I started you off with HW1_3_a_AND. Place your ISE project and any other material associated with that problem in that folder. In the end you will be turning in a single .zip or .gz.tar file for this homework.

3. VHDL Hardware design practice (70 pts)
    a. Implement the following 7 basic components using VHDL (20 pts)
    AND, OR, XOR, D-flip flop, 2:1 multiplexer [mux] , 4:1 mux build from 3 2:1 muxes, a 1-bit full adder [using your AND, OR, and XOR components]. Note: for the D flip-flop use a clocked process to implement it in VHDL. Do NOT try to make a D flip-flop using logic gates. If you find that it takes you more than 1-3 minutes to implement a D flip-flop in VHDL, then check with me.
        i. Draw a block level schematic for each component (turn in)
        ii. Implement the component using VHDL (turn in)
            1. I've done AND for you. Use it as a template
        iii. Create a testbench and test your component using Modelsim (trun in)
            1. I've done a testbench for AND for you. Use it as a template
        iv. Save the .wlf (data set) and .do (wave format) from Modelsim (turn in)
        v. Note: For ALL simulations make sure to insert dividers to group signals that belong to a five component together.

    b. Implement 4-bit adder using your 1-bit full adder (20 pts)
        i. Complete steps i-iv of a) for your 4-bit adder.
        ii. However, for the testbench replace the logic within the DUT_stimulus process with a behavioral 8-bit counter (i.e. my_count <= my_count + 1) to exercise all possible inputs of the 4-bit adder.

    c. Implement a 4-bit counter (30 pts)
    Use your 4-bit adder, D-flip flop, and 2:1 mux components as the basics building blocks, in addition to any other components you deem necessary. The counter should have the following functionality: 1) can be set to any arbitrary starting value, 2) reset to 0, 3) stall counting.
        i. The top level of your 4-bit counter should match Figure 1.
        ii. Draw a timing diagram that an engineer could use to understand how to use your 4-bit counter component. (turn in)
        iii. Complete steps i-iv of a) for your 4-bit counter

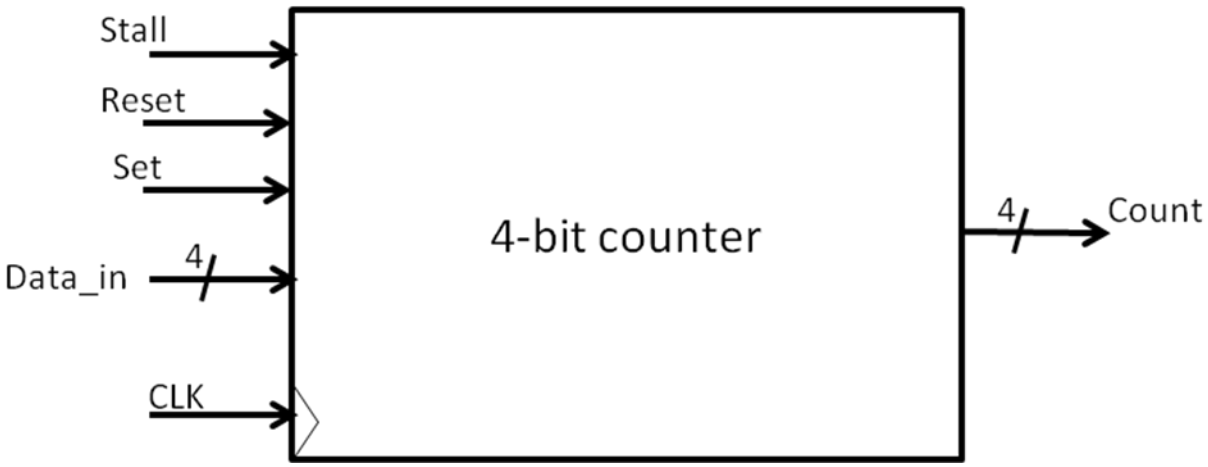

Figure 1

4. VHDL interpretation practice (30 pts)
    a. What are the final values for A, X, Y, and Z? Draw the block level circuit for this VHDL. (7 pts)

```
B <= x"A";
C <= x"5";
process (A, B, C)
begin

  Z <= A + 1;
  A <= B and C;
  Z <= x"4";
end process;

process (C, X, Z)
begin

  Y <= X + 2;
  X <= Z xor C;
end process;
```

    b. Draw a block diagram of the following VHDL (7 pts)

```
process (A,B,C,D)
begin

  if ( A = "0") then
    Z <= C or A;
  else
    Z <= A xor B;
  end if;

  case D is
  when "00"  =>
    Y <= '1';
  when  "10" | "01" =>
    Y <= '0';
  when "11" =>
    Y <= A and C;
  end case;
end process;

D <= B & C;
```

c. Draw a block diagram for the following VHDL (16 pts)

```
process (clk)
begin
  if(clk'event and clock = '1') then
    A_1 <= A;
    A_2 <= A_1;
    A_3 <= A_2;
    A_4 <= A_3;
  end if;
end process;

process (clk)
begin
  if(clk'event and clock = '1') then
    C_1 <= C;
    C_2 <= C_1;
  end if;
end process;

process (clk)
begin
  if(clk'event and clock = '1') then
    M_1 <= M;
    M_2 <= M_1;
  end if;
end process;

process (B, S1)
begin
  case S1 is
  when "00" =>
    M <= '1'
  when "01" =>
    M <= C_2;
  when "10" =>
    M<= '0';
  when "11" =>
    M <= M_1;
  end case;
end process;

S1 <= A_2 & A_4
Y <= A_4;
Z <= M_2;
```