Reconfigurable Computing: Homework 1
(CPRE 583, Fall 2010)
Due: Fri 9/10/2010

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. Getting Started: Getting the files
    a. Download HW1.tar.gz (or HW1.zip) to you U: drive (i.e. ISU network drive)
    b. Download Xilinx_12_src.txt (see MP1 link, you may have already downloaded it)
        i. source ~/Xilinx_12_src.txt
        ii. 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. And place your ISE project and any other material associated with that problem in the folder. In the end you will be turning in a single .zip or .gz.tar file for this homework.

2. See the "Tools Overview" slides for a quick tutorial on VERY basic Linux, ISE, Modelsim, and Impact.

3. VHDL Hardware design practice
    a. Implement the following 7 basic components using VHDL (AND, OR, XOR, D-flip flop, 2:1 multiplexer (mux) , 4:1 mux build from 2 2:1 muxes, a 1-bit full adder (using your AND, OR, and XOR components).
        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

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 using 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
    a. What are the final values for A, X, Y, and Z? Draw the block level circuit for this VHDL.

```
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

```
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

```
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;
```

d. For refactor.vhd (see HW_4_d in HW1.zip or HW1.gz.tar)
    i. Clearly comment refactor.vhd to show you understand what it is computing and how (turn in)
    ii. Refractor this VHDL to be an entity made up of exactly 3 components. Each component should only contain one process. And each process should contain an IF or a CASE statement. The top level entity should be called refactor_struct.vhd (turn in)
    iii. Draw the block diagram of refactor_struct.vhd (turn in)
    iv. Make a testbench called refactor_TB that sends the same stimulus to refactor and refactor_struct, and show in simulation that the response of refactor_struct.vhd exactly matches the behavior of refactor.vhd. Turn in the .wlf and .do file.

5. VHDL debug practice. For debug1.vhd and debug2.vhd (see HW_5_debug1/2 in HW1.zip or HW1.gz.tar)
    a. Debug1 has 2 issues, Debug 2 has 3 issues
    b. Draw a block diagram for each circuit based on the given VHDL.
    c. Make a testbench for debug1 and debug2
    d. Show in simulation (using a marker) when possible, an example of where each error is manifested. Explain in what way the error is being manifested. (turn in a .wlf and .do file for each bug)
    e. Identify what in the VHDL code is causing the error. Explain why (turn in)
    f. Fix the bug. Comment out the bug causing code (if appropriate). And comment why this was causing a problem and how your code fixes the issue. (turn in)
    g. Draw the block diagram for the corrected circuit
    h. Note: Steps d, e, and f will need to be done in parallel to some degree

6. FPGA basics
    a. For Figure 2, draw and label the following
        i. Virtex-5 FPGA
        ii. DRAM memory (how much DRAM does the ML507 have?)
        iii. PowerPC processor
        iv. BlockRAMs
        v. Look up tables (LUTs)

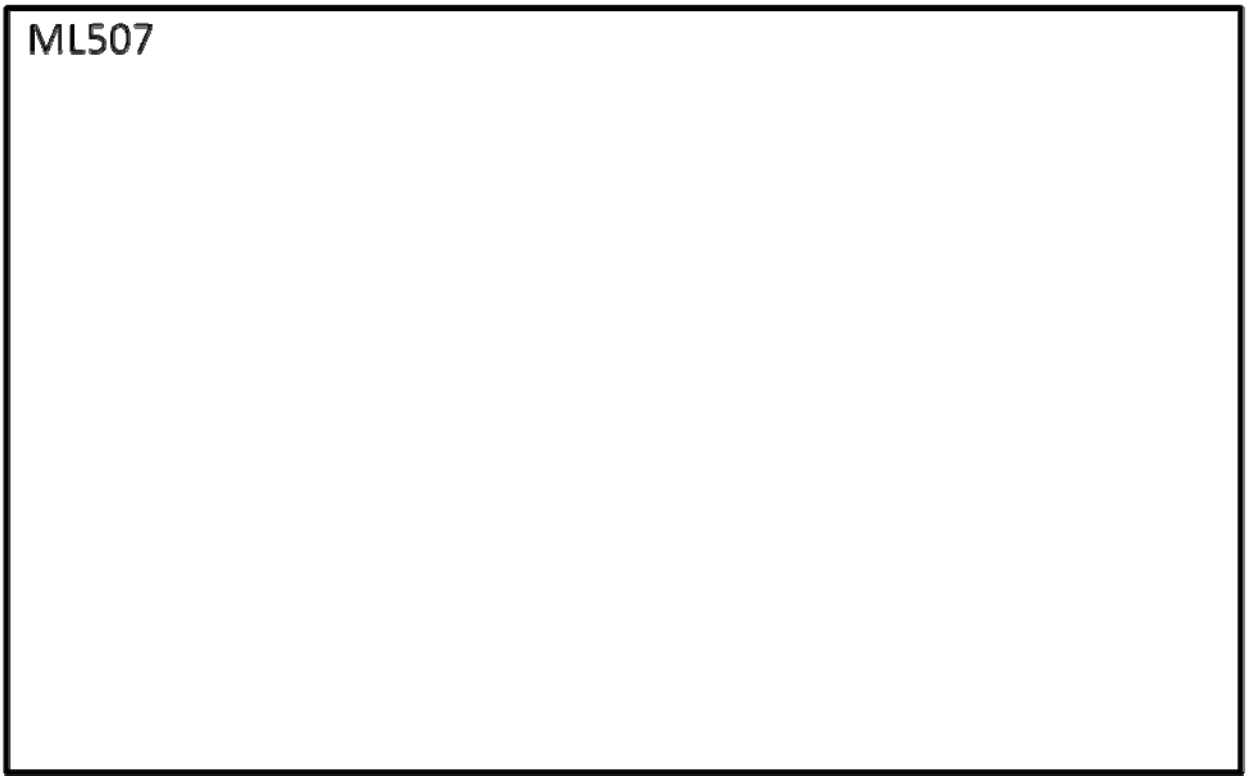

Figure 2

b. Find the "Virtex-5 User Guide" and "Virtex-5 product family table"; answer the following:
    i. How many look up tables (LUTs) does the Virtex-5 FX 70 have
    ii. How many BlockRAMs does the Virtex-5 FX 70 have
    iii. What is the memory capacity of a single Virtex-5 Block RAM
    iv. What is a DCM? What can it be used for?
    v. What is the main difference between a Virtex-5 LX, SX, and FX?
    vi. What is the main difference between a Slice-L and Slice-M

c. FPGAs, as compared to custom ASICs, are a low cost way to implement computation for specific applications. List 3 overheads within an FPGA that cause them not to be as high performance as custom ASICs.

d. Why is using a DSP48 component on the Virtex-5 more efficient then using LUTs to do multiplication?

e. What is the highest frequency processor available from Intel or AMD? How fast of an operating Frequency is the Xilinx Virtex-6 rated, How fast is the Altera Stratix IV rated?
    i. Based on these numbers how many things must a Virtex-6 or Stratix-IV do in parallel to have the same speed performance as a single core of the fastest Intel or AMD processors.

    ii. What is a major assumption that is being made when using the maximum operating frequency of these FPGAs to compare them against a general purpose processor?

7. Questions on assigned readings
    a. For assigned reading 1-4 (2a optional)
        i. Write at MOST a half page summary of each paper
        ii. Identify 2-3 key ideas, concepts, or aspects of each paper you found interesting.
        iii. Identify 1-2 topics or concepts that you would like to know more about, or were not clear.
    b. "Novel FPGA Based Haar Classifier Face Detection Algorithm Acceleration"
        i. Based on the information given on the application's profile, what is a rough estimate of the most overall speed up one would expect to gain using hardware to accelerate the portion of the application that was chosen? Why?
        ii. How does this compare to the actual overall speed up the paper shows.