Lab 5 Hierachy 4-digit stopwatch. **You must use the two-segment method (see slides 30 and 31) in the Part 2 Sequential power point reader).** Use pushbutton KEY[3] for start/stop and KEY[2] for async clear. Note that the momentary push buttons on the board are *active low* and are also debounced: Goes from 00.00s to 99.99 seconds and then rolls over back to 00.00 seconds. Start/stop button should toggle enable and disable the stopwatch. Clear button should clear the count and *stop the counting*. Use four of your Mod10 counters (with synchronous clocks) connected to four of your seven seg components. The seven seg decimal points are not connected. You will probably need to make small modifications to the code of the Mod10 counter, but you should not need to change the o_rco code. Do not use the load pin or the load inputs (don't have pins for them in the top level; instead, tie then all low in the port map). Use the 100 Hz output on the Georgia Tech clock divider. Using the old way (COMPONENT) to instantiate the Mod10 Counters and the new way (VHDL-93) to instantiate the seven seg decoders and the Georgia Tech clock divider. Connect the unused outputs on the clock divider to OPEN. You will need 4 std_logic_vector signals and 5 std_logic signals to connect the components (give them meaningful names). **Draw the top-level design (on paper and take a picture, or use software) and label the top-level pins, component pins and internal signal wires to help you do the port map. (You must submit this drawing.)**

You will also need to add one new component to start/stop the stopwatch and make it stop when you clear (you must figure out what it is), and it **must** be from Altera's library (so you learn how to do it). However, it is difficult to simulate library components in ModelSim, so if you want to simulate your design, you should make your own component (or change the start/stop function to enable, for the simulation, then change the design back to the Quartus version). You should also leave out the clock divider if you simulate (it takes 50,000 clock cycles for every 100Hz clock cycle! Simulating is not required, but it can help with troubleshooting. The libraries that include these components can be found at C:\altera\90sp2\quartus\libraries\vhdl or a similar folder. Search the folders to find the component that you are looking for. For example, the multiplexer component "mux" is found in the maxplus2 library *(you do NOT need a mux, this is just an example of how to get the component you do need).* If you want to use this component (mux), simply add "LIBRARY Altera" & "USE altera.maxplus2.all" at the top of your program (see slide 30 in Powerpoint Part 1 Combinatorial). By doing this, you don't have to declare the component! Just use the same name as the component in your port map, so you would use the name "mux" if you wanted to use the multiplexer. Also, the input and output names of the component should be the same as what is declared in Altera's library. The mux component uses the names "data" & "sel" for inputs and "result" for outputs. If these instructions are not clear, find and open the maxplus2 library (it's a VHDL file) on your computer, scroll down to where you find the "mux" component, and reread this paragraph. Also, you should not use port map statements for simple operations like inverting because you can simply use the not() function.

- For credit for this assignment, submit:

- Demo the stopwatch. Show that it starts, stops, restarts. While it's counting (running), show that the reset clears the stopwatch as well as stops it.
- Your top-level stopwatch.vhd file.
- A jpeg, png, pdf, word, or powerpoint of your top-level wiring schematic.
- The pdf of the circuit exported with the RTL viewer.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Mod10Counter is
    generic (N : integer := 4); -- default 4-bit width
    port (
        i_clk     : in std_logic;
        i_enable  : in std_logic;
        i_load    : in std_logic;
        i_clear   : in std_logic;
        i_dvector : in std_logic_vector(N-1 downto 0);
        o_rco     : out std_logic;
        o_q       : out std_logic_vector(N-1 downto 0)
    );
end Mod10Counter;

architecture TwoSegment of Mod10Counter is
    signal r_reg  : unsigned(N-1 downto 0) := (others => '0'); -- current state
    signal r_next : unsigned(N-1 downto 0);                    -- next state
begin

    process(i_clk, i_clear)
    begin
        if i_clear = '1' then
            r_reg <= (others => '0');
        elsif (i_clk'event and i_clk = '0') then
            r_reg <= r_next;
        end if;
    end process;

    process(i_enable, i_load, i_dvector, r_reg)
    begin
        -- Default: hold current value
        r_next <= r_reg;

        if i_load = '1' then
            if unsigned(i_dvector) > 9 then
                r_next <= (others => '0'); -- load 0 if input > 9
            else
                r_next <= unsigned(i_dvector);
            end if;
        elsif i_enable = '1' then
            if r_reg = 9 then
                r_next <= (others => '0'); -- roll over at 9
            else
                r_next <= r_reg + 1;
            end if;
        end if;
    end process;

    o_q   <= std_logic_vector(r_reg);
    o_rco <= '1' when (r_reg = 9 and i_enable = '1') else '0';

end TwoSegment;
```