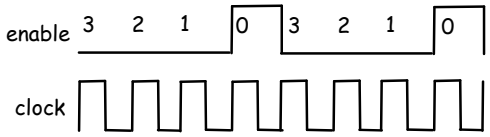
Timers and Counters in VHDL

Introduction

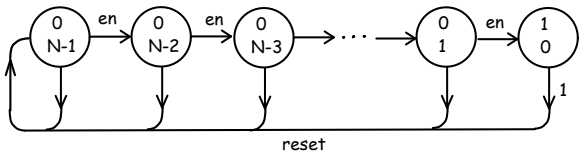
In this lab you will practice using `unsigned` types and components by implementing a circuit that increments a displayed value once per second.

Clock Divider

A clock divider is a circuit that generates a signal that is asserted once every N clock cycles. This signal can be used as a condition for state transitions so that they happen at the desired rate[1]. An example of the clock and output ("enable") waveforms for $N = 4$ is:



The clock divider state transition diagram below labels states with the value of the output and the number of clock cycles remaining until it is asserted:



The output is only asserted ('1') for the state in which the count is zero. The state transition conditions are:

- if the reset input is asserted or the count is zero, return to the first state
- else transition to the next state by decrementing the count value.

Implementing the clock divider requires one flip-flop [2] for the output and a register to count clock periods. We use a binary number corresponding to the number of clock cycles remaining before the output is asserted [3].

[1] Note that this output is *not* a clock.

[2] According to the design rules for this course which require registered outputs.

[3] Other encodings such as one-hot, binary and LFSR are also commonly used.

The VHDL for a clock divider is given on the course web site. You can use this code to instantiate a clock divider for your design and as an example of how to implement a state machine using `unsigned` values.

PLL Clock Generator

The clock for this design will be generated by a component on the FPGA called a phased-locked loop (PLL). The input to this component is a 50 MHz clock supplied by an oscillator on the DE0-Nano board. The output is a 10 kHz clock.

The VHDL to instantiate a PLL is normally generated using the Quartus II "MegaWizard Plug-in Manager". To save time the VHDL required to instantiate the PLL is given on the course web site.

Requirements

Your circuit will have the same pushbutton inputs and LED outputs as previous labs. In addition it will have a clock input, `clock_50`, from the 50 MHz oscillator on the DE0-Nano board.

The behaviour of your circuit should be as follows: If KEY0 is released at any time the circuit should be reset. When the circuit is reset the binary value N_1 should be displayed on LED4 through LED0. While KEY0 is pressed the count value should increment once per second until it reaches N_2 and then hold that value.

N_1 should be the least significant digit of your BCIT ID plus 1. N_2 should be the second least-significant digit of your student number plus 12. For example, for the ID A00123456 $N_1 = 7$ and $N_2 = 17$.

Structure

Your top-level entity should declare and use the following two components:

- `clock`: a PLL which supplies the clock for your design. It has a 50 MHz input and a 10 kHz output.

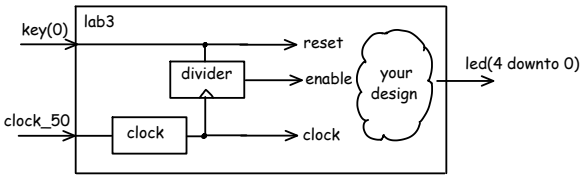
- `divider`: a clock divider whose output is asserted once per second so your count can increment at that rate. It also has clock and reset inputs.

You can declare these components in your top-level architecture, before `begin`, as:

```
component clock is
  port ( inclk0 : in std_logic ; c0 : out std_logic );
end component ;

component divider is
   port ( clock, reset: in std_logic ;
          enable: out std_logic ) ;
end component ;
```

Your design should instantiate these components and "connect" them to appropriate signals in the body of the architecture using `port map`. A block diagram of the overall design is:



Use the `std_logic_1164` and `numeric_std` packages from the `IEEE` library. Use *only* the `std_logic`, `std_logic_vector` and `unsigned` types. Constants may be of any type.

You may use selected, conditional or unconditional assignment statements, any arithmetic or logical operations and any number of additional signals. You must follow the course restriction on process statements[4].

Do *not* use the output of the clock divider as a clock.

Instructions

Use the instructions in Lab 1 but add the following before compiling your code:

- download `lab3.zip` from the course web site and extract the contents into your project folder. Then select Project > Add/Remove Files in Project... and add the files `clock.vhd` and `divider.vhd` to your project.

[4] Only a single `if` statement allowed, see the lecture notes for details.

Pre-Lab Report

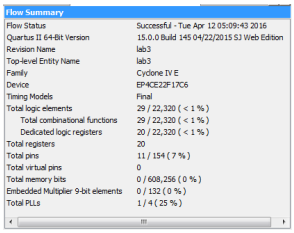
Submit a pre-lab report in PDF format to the appropriate dropbox on the course web site. The report should contain the required identification information plus the following:

- a state transition diagram for your top-level code showing with the outputs for the states and the transitions labelled with the input condition(s). You may omit intermediate states as long as the first two (N_1, N_1+1) and last two states (N_2-1, N_2) are shown. You may simplify the state transition conditions.
- A listing of VHDL code that implements a solution to the requirements above.

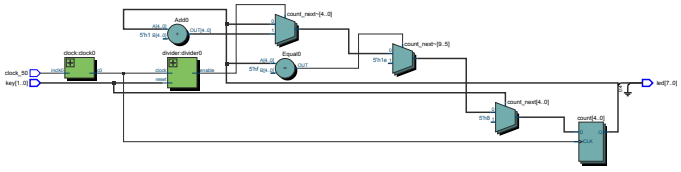
Lab Report

Submit a lab report in PDF format to the appropriate dropbox on the course web site. The report should contain the required identification information plus the following:

- your VHDL code with all errors corrected
- a screen capture of the Flow Summary showing the number of logic elements, registers and PLLs used. For example:



- the block diagram produced by running Tools > Netlist Viewers > RTL Viewer. For example:



Bonus marks may be awarded for the design that is the most concise (fewest statements), smallest (fewest logic elements) and clearest (easiest to understand).