CPE166 Lab 3 Part 4
By: Prof. Pang

Lab 4

Part 4: Stopwatch Design

This project is to design a stopwatch in VHDL by using the hierarchical design approach shown in the figure below.

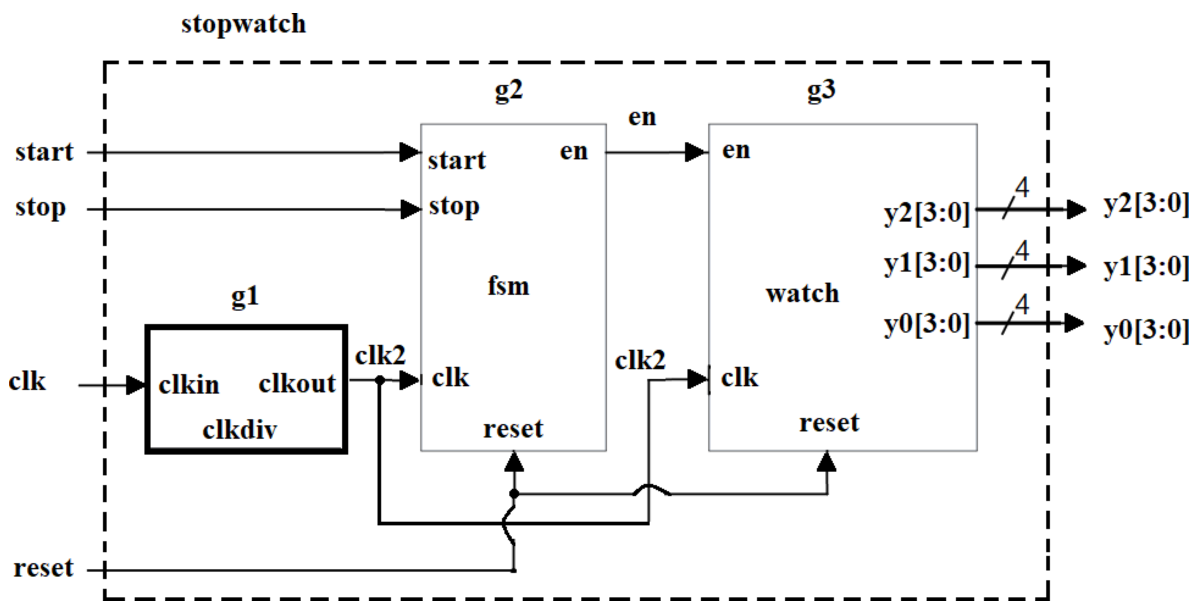

Figure 1. Stopwatch block diagram

The top-level I/O ports used in this design are shown below.

Table 1. I/O ports for the stopwatch design

Port Names	Port Direction	Port Size
start	Input	1
stop	Input	1
clk	Input	1
reset	Input	1
y3	Output	4
y2	Output	4
y1	Output	4
y0	Output	4

CPE166 Lab 3 Part 4
By: Prof. Pang

The stopwatch design has the following features:

1). The frequency of the input "clk" signal is 4 Hz.

2). At any time, if the "reset" input is logic high, the output of the stopwatch will be zero.

3). When the "start" input is logic high, the stopwatch will start counting.

4). When the "stop" input is logic high, the stopwatch will stop, but the stopwatch output will maintain its value. After that, when the "start" input is logic high again, the stopwatch will resume counting from its old value.

5). The output y2 y1 y0 is a 3-digit binary number, and each digit ranges from 0 to 9.

Assuming y2 = $(0110)_2$ = 6, y1 = $(0101)_2$ = 5, y0 = $(0111)_2$ = 7, this means 657 seconds.

If you press the "stop" button, the stopwatch will stop at 657 seconds and keep the value unchanged. After pressing the "start" button, the stopwatch will continue counting every second until: y2 = $(1001)_2$ = 9, y1 = $(1001)_2$ = 9, y0 = $(1001)_2$ = 9, which means 999 seconds. After 999 seconds, the stopwatch will return to 0 and then increase its value every second.

<u>Demo Requirement</u>

You need to demonstrate the final simulation waveform of the stopwatch.vhd design to your lab instructor.

CPE166 Lab 3 Part 4
By: Prof. Pang

Lab Procedure

Step 1. clkdiv Block

The clkdiv block gets the 4 Hz input clkin signal and generates the 1 Hz output clkout signal.

Design clkdiv.vhd and write a testbench to check your simulation results. The simulation waveform only needs to confirm that the output frequency is 4 times smaller than the input frequency.

Step 2. fsm Block

When the reset signal is logic high, the state machine enters the first idle state. Idle state means that the current watch is not counting, and the displayed output value is 0. Output "en" as 0 to disable counting. This can be represented in the state diagram below.

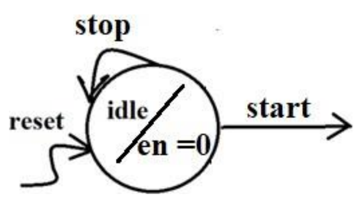

Figure 2. IDLE state of the fsm block

Figure 2 shows the state diagram of the idle state. In the idle state, the watch enable signal "en" is logic 0. if the "stop" button is pressed, the next state will still be the idle state. If the "start" button is pressed, the finite state machine needs to enter another state.

You can add more states to this design. In each state, you need to determine which state will become the next state based on the input signal values of "start", and "stop". In addition, you need to determine the output value of "en" based on whether you enable the watch to count or not.

To complete this step, you need to write fsm.vhd and testbench for simulation.

Step 3. watch Block

The watch block has three inputs: reset, en and 1 Hz clk. It has three 4-bit outputs: y2, y1, and y0.

1). When "reset" is logic high, the watch will output 0.

2). Only when "en" is logic high, the watch will increase its value every 1 second.

3). Each of the y2, y1 and y0 signals ranges from $(0000)_2$ to $(1001)_2$.

For example, if y2= $(0001)_2$ y1= $(1001)_2$ and y0= $(1001)_2$ at the current time, it means 199 seconds. After another 1 second, the watch needs to output y2= $(0010)_2$ y1= $(0000)_2$ and y0= $(0000)_2$ which means 200 seconds.

To complete this step, you need to write watch.vhd and testbench for simulation.

Step 3. Final stopwatch Block

The final stopwatch design is shown in Figure 1, with four 1-bit inputs, including reset, clk, start and stop, and three 4-bit outputs, including y2, y1, and y0.

In the architecture part of the design, two internal signals must be declared, including "en" and "clk2", and three components must be declared, including clkdiv, fsm and watch.

In addition, the main design part requires a clkdiv instance, an fsm instance and a watch instance. The two internal signals "en" and "clk2" are used for wiring of different modules.

Complete the final design and write testbench for simulations.

CPE166 Lab 3 Part 4
By: Prof. Pang

Note: Before starting this experiment, all the necessary knowledge required to complete this work has been introduced in the CPE166 lecture session. The following examples are for you to refresh your learning.

Sample VHDL Codes:

```
-- clk division circuit:  divide by 8

Library ieee;

Use  ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

use ieee.std_logic_arith.all;

entity example1 is

        port ( clkin: in std_logic;

               clkout:  out std_logic

           );

end example1;

architecture arch1 of example1 is

signal  cnt:  std_logic_vector (3 downto 0) := "0000";

begin

  process( clkin)

  begin

    if (rising_edge(clkin)) then

      if ( cnt = 7 )  then

        cnt <= (others=>'0');

        clkout <= '1';

      elsif (cnt < 3 ) then

        cnt <= cnt + 1;

        clkout <= '1';

      else

        cnt <= cnt + 1;

        clkout <= '0';
```

```
      end if;
    end if;
  end process;
end arch1;


library IEEE;
use IEEE.std_logic_1164.all;
entity testbench is
end testbench;
architecture tb of testbench is
component example1
      port ( clkin: in std_logic;
             clkout:  out std_logic
         );
end component;
signal clkin, clkout: std_logic;
begin
  DUT: example1 port map(clkin, clkout);
  clocking: process
  begin
      clkin <= '0';
      wait for 5 ns;
      clkin <= '1';
      wait for 5 ns;
   end process;


   End tb;
```

Sample VHDL Codes:

```
--  General counter: counting from 0 to 18 and then repeat.

Library ieee;

Use  ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

use ieee.std_logic_arith.all;

entity example2 is

        port (  clk:  in std_logic;

               dout:  out std_logic_vector(4 downto 0)

           );

end example2;

architecture arch2 of example2 is

signal  cnt:  std_logic_vector (4 downto 0) := "00000";

begin

  process( clk)

  begin

    if (rising_edge(clk)) then

      if ( cnt = 18 )  then

         cnt <= (others=>'0');

      else

         cnt <= cnt + 1;

        end if;

      end if;

    end process;


   dout <= cnt;

end arch2;
```

CPE166 Lab 3 Part 4
By: Prof. Pang

```
library IEEE;
use IEEE.std_logic_1164.all;
entity testbench is
end testbench;
architecture tb of testbench is
component example2
        port (  clk:  in std_logic;
              dout:  out std_logic_vector(4 downto 0)
           );
end component;
signal clk:  std_logic;
signal dout: std_logic_vector(4 downto 0);
begin
  DUT: example2 port map(clk, dout);
  clocking: process
  begin
    clk <= '0';
     wait for 5 ns;
     clk <= '1';
     wait for 5 ns;
  end process;


end tb;
```