# Problem Statement (10-02-2016)

## Part-I

In this assignment, we want to design a stopwatch in VHDL. The stopwatch should have the following input/output signals at its interface:

Inputs

*Clk* signal : A 1-bit signal.

*Reset* signal: A 1-bit input signal.

*Start_stop* signal: A 1-bit input signal.

Outputs

*Timer_out_minutes* signal: A 6-bit output signal.

*Timer_out_seconds* signal: A 6-bit output signal.

**Functional Requirements:**

The *Clk* input should be driven by a clock waveform (from the test bench) of frequency 10 Hz (i.e. clock period = 0.1s) and a duty cycle of 50% (i.e. clock stays high for 0.05s, or 50% of the clock period, and remains low for the remainder of the clock period).

When the *reset* signal is made high, it should immediately (not at the next rising/falling edge of the clock) set all internal signals in the stopwatch to their initial values (you can decide what these initial values should be), and all the outputs to a default value of 0. After the reset signal is made low, the stopwatch should operate as described below

The *Start_stop* signal is used to indicate when to start and stop the stopwatch timer. Initially (i.e. after reset), when the *Start_stop* signal is made high, it indicates that the stopwatch should start counting time. However, we require the *Start_stop* signal to stay high for at least one clock cycle before the stopwatch starts counting time. After the timer starts counting time, the *Start_stop* signal can be made low at any time, but the timer should keep counting the time elapsed. To indicate when to stop counting time, the *Start_stop* signal should be made high again and held high again for at least one clock cycle. Thus, a rising transition on *Start_stop* followed by *Start_stop* staying high for at least one clock cycle is used to indicate when to start counting, and also when to stop counting. Whether we start counting or stop counting depends on what the counter was doing (i.e. whether it was stopped, or whether it was counting).

You can implement this timer as a counter of sufficiently many bits. You need to count time in *minutes* and *seconds*. The Timer_out_sec signal represent number of seconds counted and *Timer_out_min* represent number of *minutes* counted. Each time when the value of *Timer_out_sec* reaches 60 it will start counting from zero and will also increment the *Timer_out_min* by 1.

The timer value must always assigned to the *Timer_out_seconds* and *Timer_out_minutes* output signals. For example, if 100 seconds has elapsed, *Timer_out_minutes* should have the value 000001 (i.e. 1 minute) and *Timer_out_seconds* should have the value 101000 (i.e. 40 seconds). The values of *Timer_out_seconds* and *Timer_out_minutes* should stay unchanged when the timer is stopped (as described above using the *Start_stop* signal), and these values should start from all 0's again when the timer is re-started (as described above using the *Start_stop* signal).

Remember your clock must run at 10Hz which means *Timer_out_seconds* signal should be incremented by 1 on every 10th clock cycle.

## PART-2

In this part you need to modify your stopwatch designed in part-1. You will have three more output signals.

| | | |
|---|---|---|
| *laps_count* signal | : | it is a 8-bit signal. |
| *last_time_min* signal | : | it is a 6-bit signal. |
| *last_time_sec* signal | : | it is a 6-bit signal. |
| *best_performance* | : | it is a 12-bit signal |

*Laps_count* signal gives the count of number of instances when you started your timer without making reset high in between these instances. If the reset signal becomes high, then the *laps_count* signal will be reset to 0, otherwise it will be incremented by 1 every time timer starts counting.

*Last_time_min* signal gives the counted *minutes* when the stopwatch was stopped the last time. Every time the timer is stopped, *Last_time_min* output is updated by the value of *Timer_out_minutes* value when the timer was stopped.

*Last_time_sec* signal gives the counted *seconds* when the stopwatch was stopped the last time. Every time the timer is stopped, *Last_time_sec* output is updated by the value of *Timer_out_seconds* when the timer was stopped.

*best_performance* signal denotes a performance metric. Specifically, it gives the smallest value of *Last_time_sec* and *Last_time_min* (taken together) among the last 4 instances when the stopwatch was stopped.