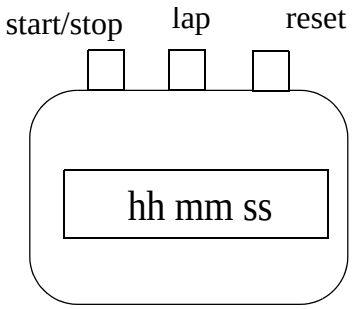
EEET2038 Assignment No 1

Stop Watch design

This assignment is a small practical exercise to give you an opportunity to tackle some basic challenges of the Verilog language and digital systems design.

Aim:

To specify then implement a Hardware Description Language (HDL) design for a simple stop watch to the point at which the design can be compiled and simulated with a simple testbench.



Methodology.

There are two components to this assignment
1. The state machine specification and design
2. Implementation and simulation of the design in Verilog

Students should prepare the first component before attending the lab. Coding, simulation and demonstration of results can be performed during the lab. Students are advised to review Verilog language guides prior to the lab.

Instructions: Stop Watch Specification.

Design a model for a stopwatch in Verilog that satisfies the following requirements:

The stopwatch shall have three input 'buttons':
Button 1: Start/Stop
Button 2: Lap
Button 3: Reset

The only output shall be the watch display, which may be implemented as a simple register, the value of which may be observed during simulation as the output. (That is, it is not required to implement a 7-segment display driver, just a six digit binary register).

The display shall contain six characters that represent hours, minutes and seconds in the format *hhmmss*. *hh* = hours, *mm* = minutes, *ss* = seconds. All numbers are decimal base (range 0-9).

Ranges:
ss 00..59
mm 00..59
hh 00..99

Ask your lecturer for clarification of any requirements you do not understand.

Functionality
Pressing RESET at any time stops the watch running and resets *hhmmss* to 000000.

Pressing START/STOP when stopped starts the watch running, such that time increments one second at a time.

Pressing START/STOP when running freezes time at the current time.

Pressing LAP once freezes the display at the current time but time shall keep incrementing in the background (the LAP_DISPLAY state).

If LAP is pressed again in the LAP_DISPLAY state, the diplay reverts to showing present time as it increments.

If LAP is pressed again while running the display freezes into LAP_DISPLAY state again.

If START/STOP is pressed while in LAP_DISPLAY state the timer stops but the display remains in LAP_DISPLAY state until LAP is pressed again, at which time the final stopped time is displayed.

If START/STOP is pressed when the timer has stopped (START/STOP pressed at least twice since RESET) the timer starts running again.

Implementation details:
A system clock of any desired frequency can be used.

As we have not yet dealt with asynchronous systems, to simplify the design you can assume button presses will be synchronous to the system clock and maintained for a duration of your choosing (e.g one clock cycle).

It is suggested that students begin by drawing a state diagram.

Simulation only is required, not synthesis.

Testbench can be simple/manual provided demonstration can be performed in a few minutes.

Students already familiar with Verilog should seek to implement a synthesizable implementation. Students new to verilog may implement a behavioural implementation.

Any timescale may be specified (see ‘timescale directive)

Report:
Document your design in a report. Include state diagrams and/or state transition tables, an overview of your design including, if appropriate an architectural diagram. Include a brief description of how you went about coming up with the solution and why you made any key decisions you did. Include printed copies of your source code (soft copies not required for this report).

Include a statement summarizing what was achieved in the design so far. What worked, what did not work (yet). A clear and honest statement of progress achieved, as if you were reporting to your project manager on your work so far, and any unique problems encountered.