# Lab 3 Report

**Dennis Zang, 704766877**
**Partner: Hirday Gupta**
**Lab Section 5, November 10, 2019**

**Problem Statement:**

The task at hand that was assigned is to program a functional digital stopwatch in Verilog and, unlike previous labs, actually implement it on the Nexys$^{TM}$3 Spartan-6 FPGA board. The input will be 4 separate switches ("RESET", "PAUSE" , "ADJ", and "SEL"), and the output will be the standard digital clock output (4 seven-segment digits, with the two rightmost representing the minutes and the two leftmost representing the seconds). The "RESET" switch, if set, will set the clock time to and keep it at "00 00". "PAUSE" will keep the clock time from incrementing, keeping the time at the current time. "ADJ" is used to adjust the time by having the seconds or minutes increment by 2 each second (which "SEL" is used to select minutes [0] or seconds [1]). In addition, when "ADJ" is active, the field being incremented by 2 each second (minutes or seconds) should blink on a per-second basis.

**Implementation/Algorithm:**

As per the specifications for the lab, we are recommended to implement the stopwatch in a module-submodule hierarchy. In our case, we implemented 3 submodules: "clock_divider", "counter", and "seven_seg_display", which are all operated under the top-level module "stopwatch". See the second picture in the "Netlists" section of this report to see how the submodules are operated under "stopwatch".
Below are the description of how each module and submodule are implemented.

(clock_divider)

The purpose of this submodule is to convert the clock signal on the Nexys$^{TM}$3 Spartan-6 FPGA board into clock signals for 1Hz and 1kHz. Given that the original clock signal frequency on the FPGA board is 50MHz, we need to divide the clock signal in order to get the clock frequencies that we need.

The inputs to this submodule would be "inp_clk" and "rst", and the outputs would be "out_clk_1hz" and "out_clk_100hz". In this module, we made 2 27-bit counter registers to count the number of clock cycles (positive edges of the signal "inp_clk"). For the 1Hz clock, we count 50000000 clock cycles before toggling the current value of "out_clk_1hz", and 50000 before toggling the current value of "out_clk_100hz". Of course, if the reset switch "rst" is on, we set the output signals off (the clock wouldn't toggle and would effectively stop).
(Note: In our Verilog program, although the clock signal may be named with "100_hz", it is really going at 1kHz. We were just too lazy to change the names of all the clock signal variables, which were originally 100Hz.)

(counter)

The purpose of this submodule is to implement a counter in "mod 60" for the time. In other words, we would like to have a time counter that is fed the clock signal, and count them in terms of minutes and seconds.

To implement this, we would need to feed in the input 1hz clock signal from "clock_divider", "clk". Generally, we would have two 6-bit counter registers (one to count number of minute, the other seconds) which would be incremented like a normal digital clock for each positive edge of the "clk" signal. The two registers would be the outputs "min" and "sec" for minutes and seconds, respectively.

In addition, for the user's switch functions, we would add 4 more inputs: "rst", "pause", "adj", and "sel". "rst", when set, would have "min" and "sec" set to 0. "pause" would determine whether the conditional to update the "min" and "sec" for each positive edge of "clk" (keeping the same value). "adj", if set, would have "min" increment by 2 each clock cycle if "sel" is 0, and "sec" increment by 2 if "sel" is 1.
(Of course, in both regular operation and the "adj" operation, we would need to check for clock overflow: "min" and "sec" must both be less than 60.)

(seven_seg_display)

The purpose of this module is to calculate what values to pass onto the FPGA board's interface foe the seven segment LED display. To use the FPGA board's led display, we would need to pass in an 4-bit integer "an" representing which of the 4 LED displays to shine ("0111" for the tens digit for "min", "1011" for the ones digit for "min", "1101" for the ones digit for "sec", "1110" for the ones digit for "sec", and "0000" for the "rst" case [set all seven segment displays to the same integer]). As for the seven segment display, we would pass in an 8-bit integer "seg" (in which 7 bits would represent the on/off state of each segment of the display, and one for the decimal point). Note that for both cases, we are using an inverted bit notation to indicate which digit and which segment to turn on.

For this module, we are passing in the 5-bit values of "min" and "sec" from the "counter" module. Of course, to get the correct digits for the tens and ones digits for "min" and "sec", we would need to take the resulting value when we divide by 10 ("/") and mod 10 ("%") the values, respectively. Then we would convert the number into the corresponding 8-bit number for the seven segment display.

For clock synchronization purposes for the display, we would also have "clock_100hz" (1kHz, not 100Hz as mentioned before) and "clock_1hz" from the "clock_divider" submodule. Because we can only display one digit at a time (desides the reset case), we would need to toggle between each digit at a very high frequency (1kHz). "clock_100hz" is used to toggle a state register between the values "0111", "1011", "1101", and "1110" in an always block whenever there is a positive edge on "clock_100hz". In a separate always block with the same condition, we would set "seg" and "an" to the corresponding digit and state register value, respectively.

In addition, we added in 3 of the 4 switch inputs "adj", "sel", and "rst" ("pause" wouldn't apply for display purposes). When "rst" is set, we would set "an" to the reset state "0000" and "seg" to "00 00". Note that when "rst" is set, the clock signal will always be 0 (as returned in submodule "clock_divider"), so we need to modify the conditionals in the always blocks to respond to "rst" as well. We also need to implement blinking when "adj" and "rst" are set, so when setting "seg" at any point in the submodule, we would need to check if "adj" is set, the value of "sel", and "clock_1hz" to blink the led once per second and the correct value digits to blink (both digits of "min" and both digits of "sec").

(stopwatch)

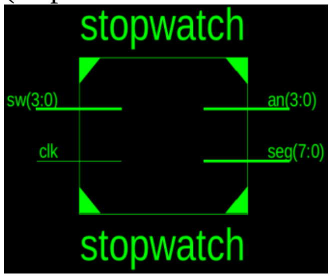
The top-level module, “stopwatch” would connect the three aforementioned submodules together. The module would take 4 switches “sw[4:0]” and and a clock signal “clk” as inputs, and output the seven segment representation for the clock output. As mentioned earlier in submodule “seven_seg_display”, we would only display one digit at a time (bar reset), but toggle between them so quickly at 1kHz that it looks like all the digits are on at once. The corresponding switch inputs (sw[0] as” rst”, sw[1] as “pause”, sw[2] as “adj”, and sw[3] as “sel”) are fed into corresponding submodules listed above, while “clk” (the 50MHz clock signal) is fed into “clock_divider” to get the 1Hz and 1kHz clock cycles to feed into the other two submodules.
(Diagram of implementation given in “Netlists” section of the report)

**Simulation Screenshots:**

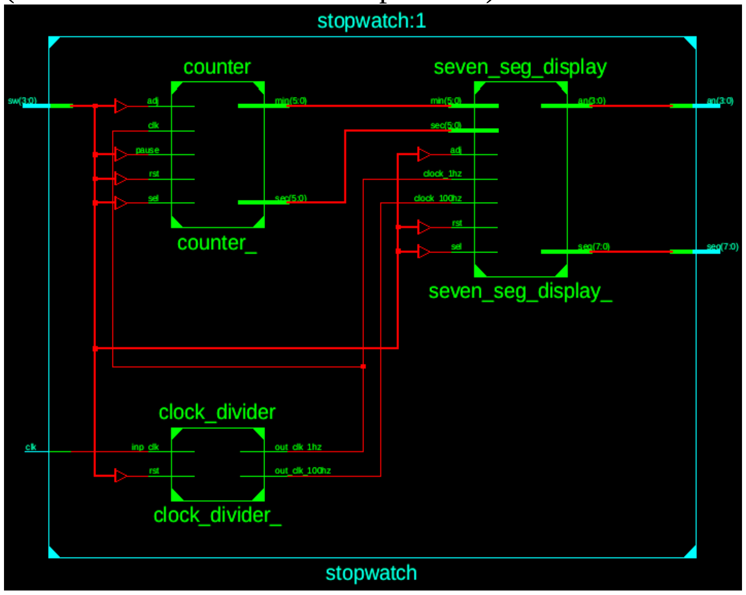
(not applicable; implementation is proven to work, and netlists are listed below)

**Netlists:**

(Top Level View of Module “stopwatch”)

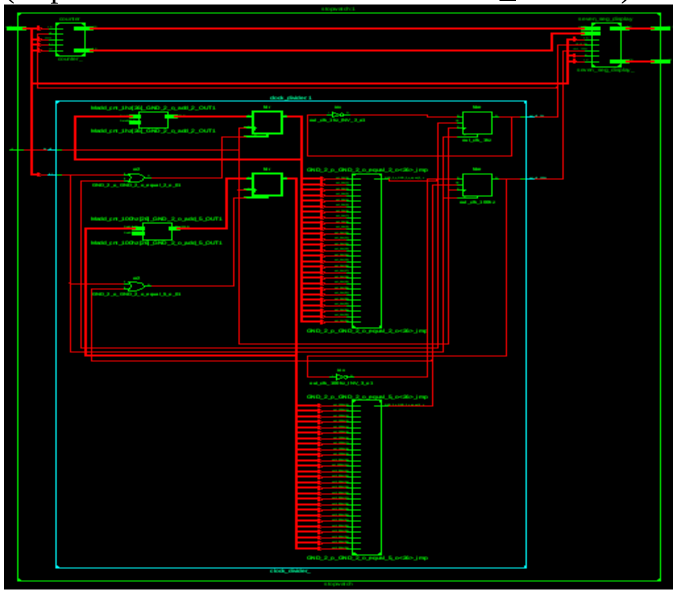


(View of Submodules of “stopwatch”)

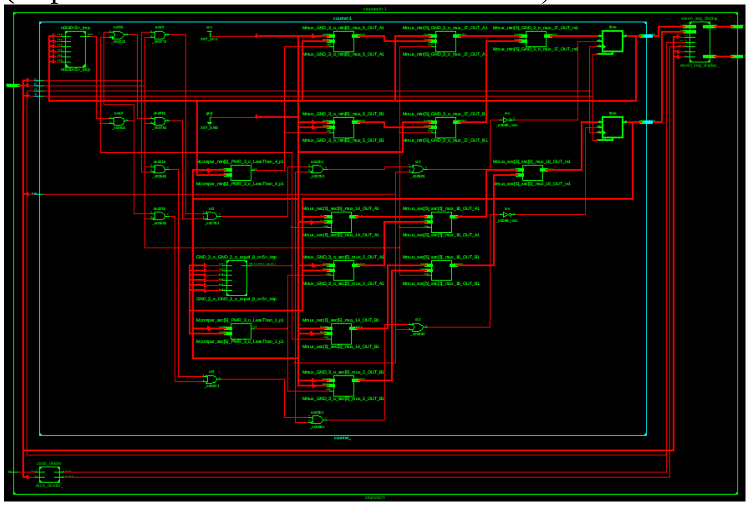


Here, we can see how each submodule is connected to the inputs and outputs of the top-level module and each other. For more details, see the section for “stopwatch” in the “Implementations/Algorithms” section of the report.

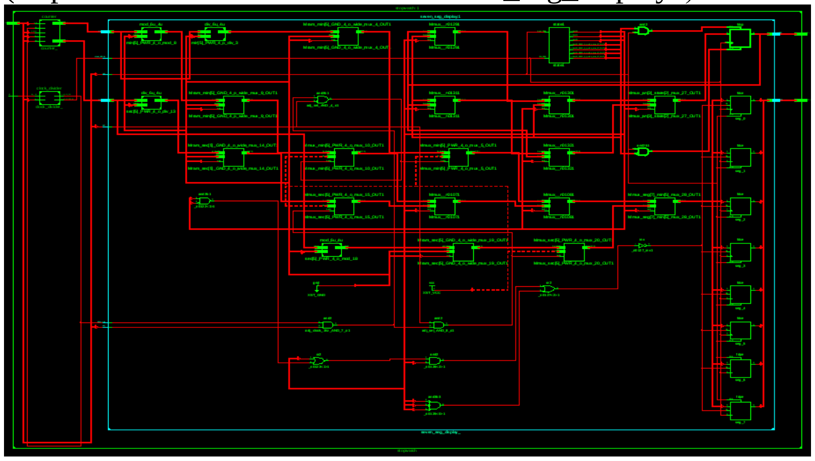
(Expanded View of Submodule “clock_divider”)



(Expanded View of Submodule “counter”)



(Expanded View of Submodule “seven_seg_display”)



**Inferences:**
(Challenges)
There were mainly two challenges with this lab: figuring how to operate the FPGA board's interface for the seven segment LED displays and figuring out how the inputs connect to each individual submodule. Operating the LEDs required reading through the specification for the Nexys$^{TM}$3 Spartan-6 FPGA board, configuring the UCF file, and a lot of trial and error (much more than needed) to figure out the orientation of the multiple-bit inputs. For figuring out how the inputs connect to each submodule, we had to continuously modify our existing implementation (through trial and error as well), like modifying our always block in "seven_seg_display" to trigger not only for the positive edge 1kHz clock signal but also for the positive edge of the reset input, or adding a 1Hz clock input conditional to modify the value of "seg" to flash the output.

(Mistakes)
We many mistakes in trying to figure out how the top-level inputs connect to the submodules (like the "rst" input stopping the clock, which in turn froze the toggling output of submodule "seven_seg_display" before we made the always blocks trigger on the positive edge of "rst" and added new conditionals). A major time-consuming issue we ran into was figuring out the orientation of the 8-bit input for the seven segment display, as mentioned above. All of the mistakes were fixed in the final code though.