Design a Stopwatch

We want to design a stop watch, using a Finite State Machine, counting minutes and seconds, in the following format.

00'00'' up to 99'59''

The stop watch digits should then be visible on the 4-digit 7-segment display on the Basys-3 board.

Exercise 1. Design the 7-segment display controller

Go to `~/labs/lab15` and open `src/display-controller.vhd` with a text editor.

```
kate src/display_controller.vhd &
```

This is the module that should drive the actual display. The port interfaces are already defined.

Port	Type	Width	Mode
clk	std_logic	1	in
rst	std_logic	1	in
digits	t_digits	3	in
segment_anodes	std_logic_vector	4	out
segment_cathodes	std_logic_vector	8	out

Here `t_digits` is a type defined in the `src/display_pkg.vhd` file, which is just an unconstrained array of integers.

The display_controller can also be configured using the `clocks_per_digit` integer generic, which declares how many clock cycles you need for each digit to update. By default is `400000`, which is 4ms at 100 Mhz.

Follow the instruction in the code to implement the following functionalities.

1. Implement a state machine with the following states (IDLE, DIGIT0, DIGIT1, DIGIT2, DIGIT3).
2. Add a counter to keep track of the clocks per digits
3. Modify the synchronous process with the asynchronous reset
4. If the `rst` is high, set the state to IDLE, and all the bits of the segment anodes and cathodes signal to 1.
5. Increase the counter until reaching `clocks_per_digit`, then reset the counter and set the signal next_digit to high, otherwise keep it low.
6. In the case, when state is:
    1. IDLE, go to state `DIGIT0`, copy digits into a signal `copy_digits` and set `segment_anodes(0)` to low. Set the segment cathodes, to the bit representation of

`digits(0)`. You can use the `digit_to_display` function, defined in the `display_pkg.vhd`

2. In `DIGIT0`, `DIGIT1`, `DIGIT2` and `DIGIT3`, when next_digit is high go to the next digit state (e.g. DIGIT0->DIGIT1), and set the corresponding anodes to low, and update the cathode signal for the next digit. From DIGIT3 go back to IDLE, once receiving next_digit.

Once you finish designing the module, you can run `run_sim.sh`, to validate the design.

Exercise 2. Implement the full design

Create a Vivado project for Basys-3 as in the previous labs.

- In the Add Sources, add the entire content of `lab15/src`. Add also `sim/tb_stopwatch.vhd` and set it as Simulation Only in the table.
- In the Add constraint, add `src/basys3.xdc`

Once finished, open the `stopwatch.vhd` file, which should be our top file.

The module should update the four digits of the stopwatch every second and continuously send them to the instantiated `display_controller` module, we just designed. The output of the display controller, is already wired to the output ports of the design. The RST port is connected to central push-button, the `STOP` to the upper, and `START` to the down push-button.

Implement the following functionalities:

1. A state machine with states RESET, COUNT, PAUSE.
2. A counter to keep track of the seconds.
3. A synchronous process with asynchronous reset. If RST is high, go to RESET state. Otherwise, increase the counter until reaching the CLOCKS_PER_SECOND constant, defined in the `display_pkg`. Then reset the counter and set a signal `increase_digit` high, which will trigger the state machine. Otherwise keep it low.
4. When the state is:
    1. RESET, reset the counter and set the digits to (0,0,0,0). If START is 1, go to COUNT
    2. COUNT, if `increase_digit` is 1, increase the digits, following the stopwatch order. If `stop` is 1, go to state PAUSE.
    3. PAUSE, if `START=1` go to COUNT.