Lab 7: Sequential Logic - Counters and Debouncing

In this lab, you will code a simple counter design, and use it later to debounce a button.

The ports of the counter design are

Port	Direction	Type
`clk`	in	`std_logic`
`rst`	in	`std_logic`
`count`	out	`integer`

- The `rst` signal is synchronous and active-high.
- When `rst` is high, `count` is zero
- When `rst` is low, `count` is increased by 1 at each positive edge of `clk`.

A testbench is provided to test the functionalities of the design.

Exercise

1. Code the Counter

Open the counter design with a file editor,

```
cd ~/labs/lab07
kate src/counter.vhd &
```

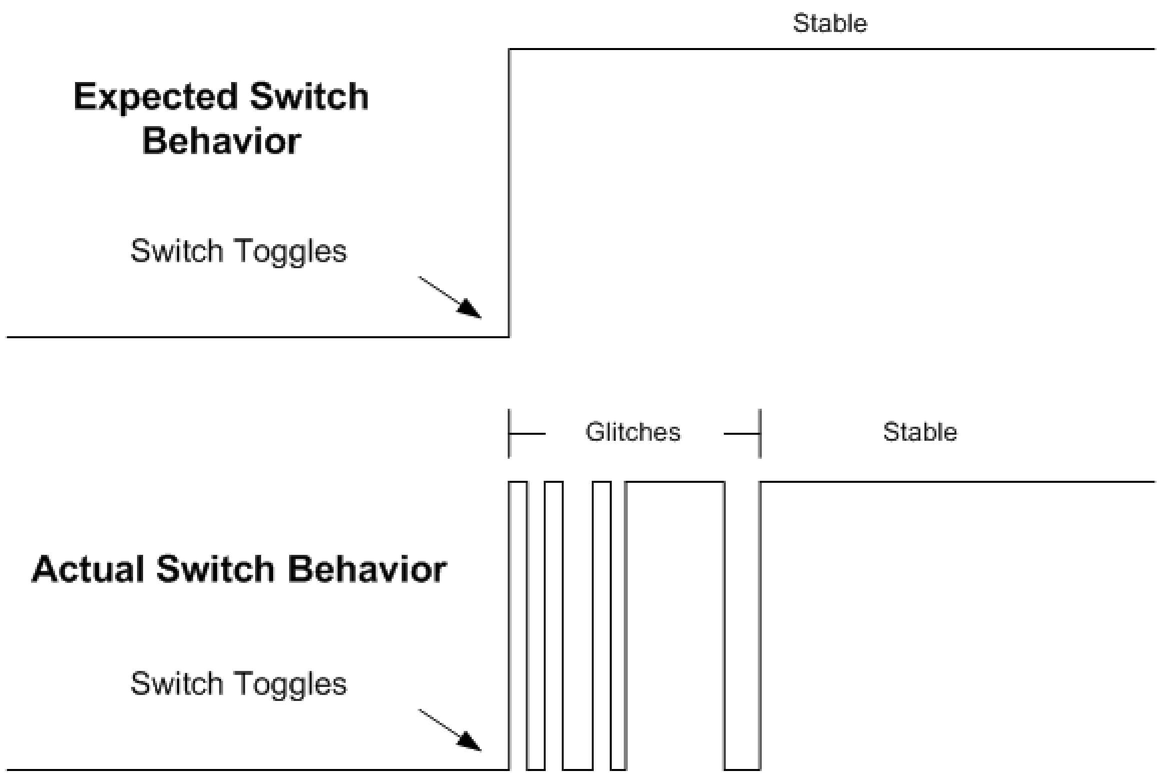
and implement the design, following the comments inside the file.

2. Run the simulation

Run the simulation to check your design

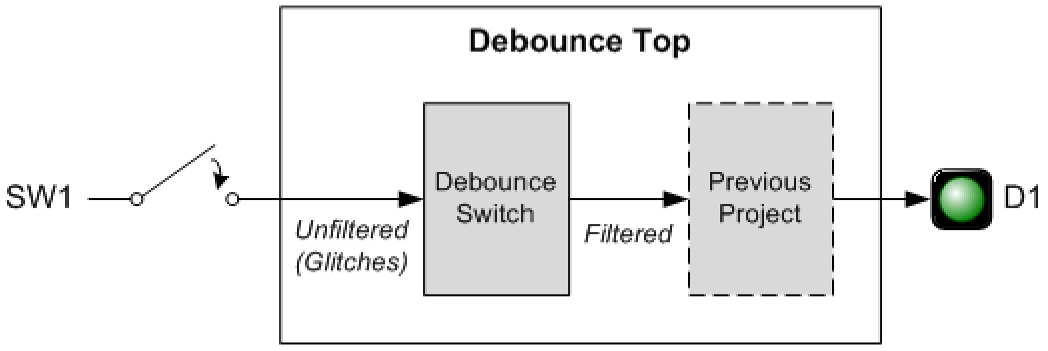
```
./run_sim.sh
```

3. Debouncing a button using a counter



Any physical button or switch on a board is subjected to *bouncing* or rapid signal fluctuation. You would expect a switch to behave like in the top half of the figure, but it actually behaves like the bottom.

To avoid these glitches, we have to design a *debouncing filter*. The module should check that the signal coming from the button or switches is stable for a certain time, before being used in the rest of the logic.



Open the file `src/debouncer.vhd`, and implement the design as follows.

Port	Direction	Type
clk	in	std_logic
btn	in	std_logic
filtered_btn	out	std_logic

- `btn` is the input unfiltered signal from a button or switch
- `filtered_btn`, is the output debounced signal

Make use of the `counter` module you design earlier, to check the stability of `btn` signal. If `btn` is stable for at least 1ms, then `filter_btn` takes its value, otherwise it keeps its old one.

Remember, that our board runs with a clock of 100 MHz.

Run the `run_debounce_sim.sh`, to check your design.

```
./run_debounce_sim.sh
```