Exercise 02: Sequential logic

Edge detector

Write an entity that generates a pulse of one clock cycle when an input signal transitions from low to high (rising edge).

Use following entity definition:

```
entity edge_detector is
port (
    -- input
    i_d     : in    std_logic;
    -- ouput pulse of one clock cycle
    o_q     : out   std_logic;
    -- clock
    i_clk   : in    std_logic--;
);
end entity;
```

Instructions:

Use a register (flip-flop) to save the state of the input.

Generate a pulse signal when the state transitions from low at previous clock cycle to high at current clock cycle.

Validate you entity with the provided test bench.

Debouncer

Physical buttons/switches may bounce between on and off states during switching for a finite time before settling on a stable state. During this time the digital logic may see unwanted transitions when the state of the switch is sampled by a clock.

Write a debouncer entity which produces an output signal (equal to the input) when the input signal is stable for some period of time (e.g. `1 ms`).

Use the following entity definition:

```
entity debouncer is
generic (
    -- number of clock cycles of stable signal
    g_N : positive := 4--;
)
port (
    -- input
    i_d         : in    std_logic;
    -- output debounced signal
    o_q         : out   std_logic;
    -- active low reset
    i_reset_n   : in    std_logic;
    -- clock
```

```
    i_clk       : in    std_logic--;
);
end entity;
```

Instructions:

Use a counter to measure the time during which the input stable, i.e. reset the counter when the input changes (from 0 to 1 or from 1 to 0).

If the signal is stable and counter reached required number of cycles, copy the input to the output, otherwise increment the counter.

Finally implement reset logic (note that `i_reset_n` is an active low signal) such that entity can be initialized to known state.

Note the sequence of checks in the clocked process, and remember that all accesses to registers see values just before clock edge and all of the registers are updated during clock transition.

Validate you entity with the provided test bench (adjust the time period to clearly see the transitions in the wave diagram).

Test the debouncer in hardware by using a time period of `0(1) s` (estimate required number of cycles and set generic `g_N`).

The labs take place in Staudingerweg 9 building: group 1 in room 4-516 on Thurday and group 2 in Newton room on Friday.

Send your solutions through MOODLE (https://moodle.uni-mainz.de).

The solution should include:

- Source code (only **your** `.vhd` files)
- Brief description if applicable, questions, etc.
- Screen shot of the simulation (use simulation range of 1 microsecond)
