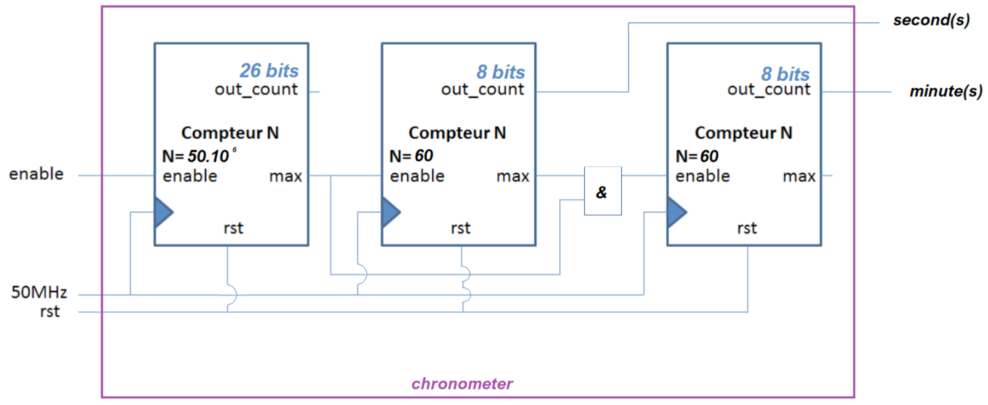
| **Course:** **CE312** | **Report** **TP2 - Chronometer and implementation on FPGA** |
|---|---|
| **Date:** | 15/12/2020 |
| **Students:** | CELLARD Rémi - PIERSON-MAURY Damien |
| **Subject:** | Chronometers and implementation on FPGA |
| **Attached files:** | *fpga_chronometer.zip* - Project folder for Vivado 2014.3.1 |

## 1. About this report

The project folder with the code is included in the .zip file which contains this report. Please refer to these files for the VHDL code.

## 2. Chronometer architecture



*Architecture of the chronometer*

The schema above represents the chronometer. Each counter has its own modulo and number of bits in output. An AND gate has been added to avoid counting the minutes at the same speed as the first counter. Indeed, the "max" output of the second counter remains active for the time of one count from the first counter. Therefore, we wait until the max is reached with both the first and second counter to add one minute.

For this chronometer, we need to have 44 D-“Flip-Flops” inside the chronometer. Here’s the explanation:

- 1 D-“Flip-Flop” for each bit at the output of each counter (26 + 8 + 8 = 42)
- 1 D-“Flip-Flop”  for each "max" output of each counter (1 + 1 + 1 = 3)
- At the last counter, the "max" output is not used, so we don’t need a D- “Flip-Flop”.
- Total : 42 + 3 - 1 = 44

We found these values when we synthesized the chronometer. But we can remove 4 more D-“Flip-Flop” if we use only 6 bits for the 2 outputs of the counter of seconds and minutes (we only have 60 seconds per minute so we can store this value in 6 bits).

## 3. FPGA Implementation

To implement our chronometer on the FPGA board, we used the project device *xc7a35tcpg236-1* with the constraint Basys3 on Vivado.
One of the problems that we only see with the library of Basys3 is to display the number. Each digit is hexadecimal (from 0 to F). It uses 4 bits for each digit, but we have 8 bits at the output of the chronometer for the seconds and minutes.

To split our 8 bits into 2x4 bits to display the seconds (or minutess) in 2 digits, we use these two lines of code. It allows us to get only the first or second numeral that we need to display on each digit. Here's an example:

```
 hex2led_int1 <= std_logic_vector(resize((unsigned(out_secondes_value) mod
10),4));
 hex2led_int2 <= std_logic_vector(resize((unsigned(out_secondes_value) / 10),4));
```

hex2led_int1 is for the second numeral (0**0**) and, hex2led_int2 is for the first numeral (**0**0).

We can represent our 4 variables in the 4 digits available as follows:

| Minutes | | : | Seconds | |
|---|---|---|---|---|
| *hex2led_int4* | *hex2led_int3* | | *hex2led_int2* | *hex2led_int1* |
| 0 | 0 | : | 0 | 0 |