# IAS0600 LAB REPORT 3 COUNTER

Kayode Hadilou ADJE

194360MAHM

## Introduction

This lab dealt with sequential systems design; in sequential systems, outputs depend not only on its present inputs such as in combinational systems (previous labs) but also on its past history of inputs. The are many sequential systems such as memory elements and counters. In this lab, it was about designing counter (with debouncing effect) system. The lab was performed after sequential systems lecture during which theoretical information about such systems was given. At the end of the lab, the system should be able to count the number of times a button has been pressed.

## Background

For this practical lab, the background sources I benefitted from are listed and detailed below:

-Lecture 7 Explanation and Slides: I relied on and used lecture 6 slides and explanations to understand more the notion of sequential systems and counters in particular;

-Lab Sheet: The explanations in the lab sheet was a guide for the completion of the tasks;

-The cumulated knowledge gained in previous lab work and first trainings practical classes about VHDL and Vivado were of great use.

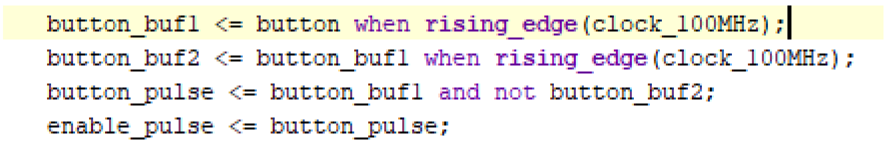
## Workflow

This lab was about implementing a counter system able to count the numbers of times a button has been pressed. The lab was divided into two tasks:

-implementation of a counter system without debouncing effect. Here, the counter may increment more than once due to bouncing effect.

-implementation of the counter with debouncing effect. At this step, the system is expected to count only once because the bouncing effect is eliminated.

Task1: The direct way to detect a button press is to watch the button signal and detect cases where the signal goes from low(unpressed) to high (presses), so that at each positive edge transition, the counter value can be increased by one. The button input signal can be split and the last two values can be compared, if they are different it is then possible to say that there is a transition in button signal.



```
button_buf1 <= button when rising_edge(clock_100MHz);
button_buf2 <= button_buf1 when rising_edge(clock_100MHz);
button_pulse <= button_buf1 and not button_buf2;
enable_pulse <= button_pulse;
```

*Listing1: Edge Detection*

Below is the RTL schematic of the approach used.

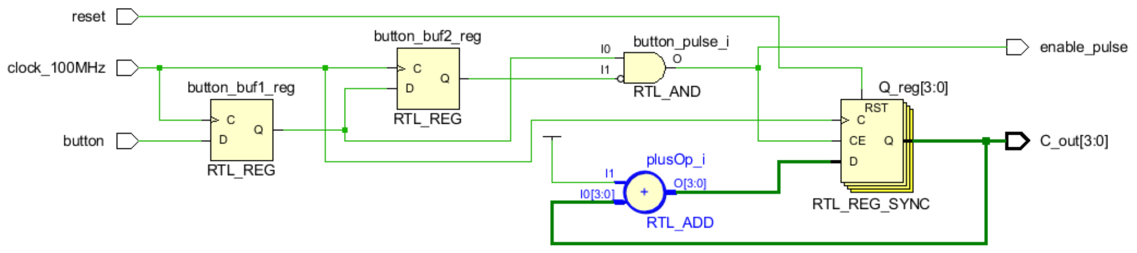


*Figure 1 Edge Detection Circuit*

As it can be seen, the button input signal is split into two, and stored in two different registers. Later, the AND NOT arithmetic operation is applied so that if the two values are different the result will be '0' and '1' in case the values are the same. The output of this arithmetic operation is used as Enable bit for the register storing the count value.

Task2: The problem after the completion of the first task is that counter may increase more than once due to the bouncing effect. During each press, the mechanical part included in the button can make contacts several times and the high-speed circuit can detect this as a button press while it is basically not one. There was a need for a hardware level implementation to filter the bounce. The approach proposed in the lab sheet consisting of implementing a 4-bit shift register to save samples values from button's input and enabling the counter register only after all four bits of the shift registers are high was used.

But, the FPGA board (Nexys-4 FPGA) used in this lab has a 100Mhz clock source which is too fast to be used, the first toward implementing the debouncing effect will be to slow down the original clock. As proposed in the lab shit, an 100Hz clock signal can do the trick, so the clock featured in Nexys-4 should be slow down by 10^6 times. A *counter* integer signal initialized to zero is incremented at each rising edge of the 100Mhz clock. When the counter variable increments 10^6 times, then one 100Hz clock is obtained and can be used in the design of the 4bit shift register. A visual illustration can be seen in the RTL schematic representing the clock division part.

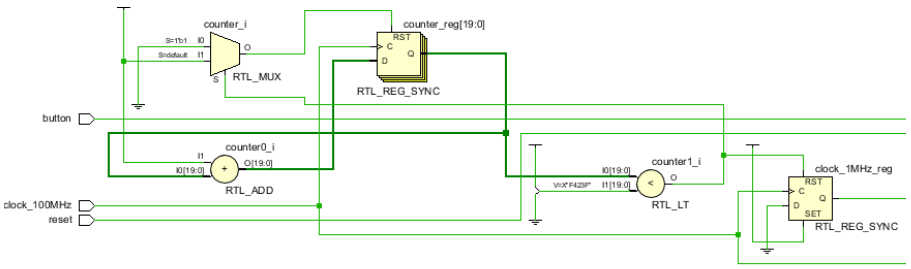
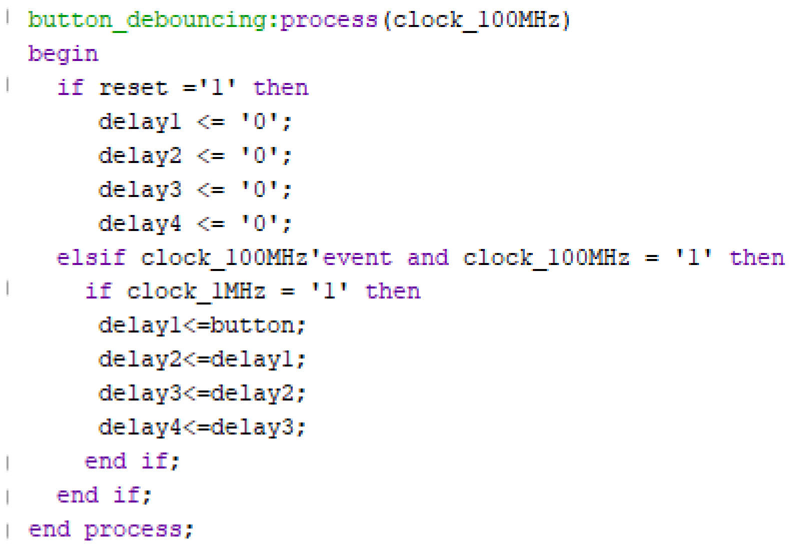


*Figure 2 Clock Division*

The 100Mhz clock signal is then used as enable bit to store four delayed samples of button pulse signal. If the 100Mhz clock takes N ns per cycle, then it will take 4*N ns in order for the button pulse signal to be propagated over the four registers. Any signal (bouncing effects) lasting less than this would go unseen and will not trigger any count. Below is the RTL schematic description the approach.



```
button_debouncing:process(clock_100MHz)
begin
  if reset ='1' then
     delay1 <= '0';
     delay2 <= '0';
     delay3 <= '0';
     delay4 <= '0';
  elsif clock_100MHz'event and clock_100MHz = '1' then
    if clock_1MHz = '1' then
     delay1<=button;
     delay2<=delay1;
     delay3<=delay2;
     delay4<=delay3;
    end if;
  end if;
end process;
```

*Listing 2: 4bits Shift Register*

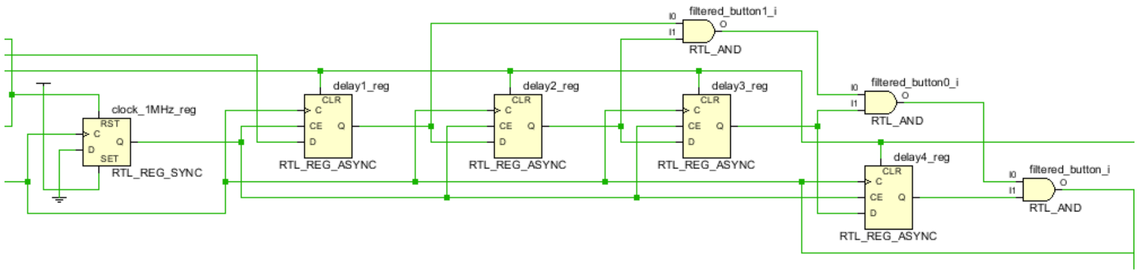


*Figure 3: 4 bits Shift Register*

As it can be seen, at each 100MHz clock rising edge if the enable bit (100Hz clock value) is high then, the button pulse is passed to the next register in the queue. *Filtered button* is the result of the AND arithmetic operation applied to the four outputs values of the shift registers.

From here the rest of the implementation is similar to the one in the first task. The ANDed output of the 4-bit register is sampled to detect state transitions. In case, there is a state transition, it is then possible the increments the value of the counter. And since, the delayed output of 4bits shift registers is used, only button signal lasting longer than or equal to 4*N (where N is the time 100Hz clock takes per cycle) will result in incrementing the counter.

## Results and Discussion

Each task was tested in testbench and the implementation on FPGA was also done. For the first task below is the screenshot of the testbench result.

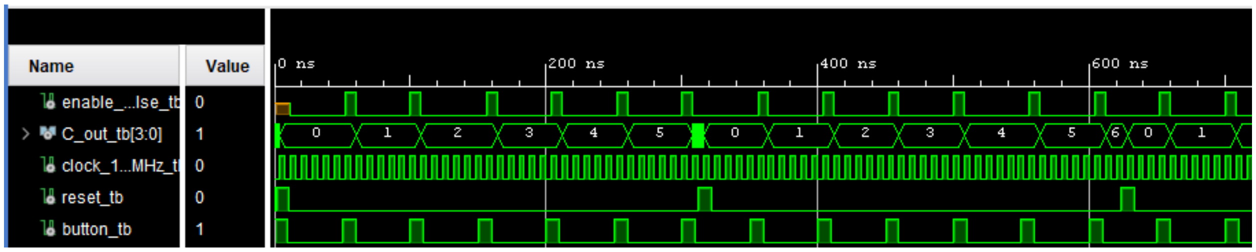


*Figure 4: Testbench waveform Task 1*

*C_Out* is the output and represent the number of times the simulated button (*button_tb*) has been pressed. At each reset, the counter value is set to '0'. The *enable_pulse* is high only when the two sampled button pulse are equals and only then the counter is incremented.

The last figure represents the screenshot for Task2.

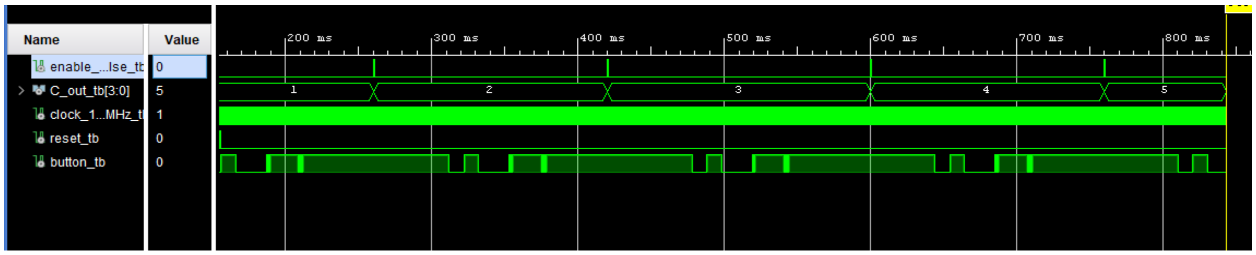


*Figure 4: Testbench waveform Task 1*

Here, the button as it can be seen, any signal lasting less than 80 ns approximately is considered as bounce and ignored. The counter increments only when the system detects a long press enough to trigger a count.

## Conclusion

To summarize, in this lab I have learned about how to think of and design sequential systems in general and clock in particular. The clock debouncing was a bit challenging due to lack of theoretical knowledge that I later acquire by revisiting the lectures slides. To finish, I think this lab was a success.