Debounce Logic Circuit (with VHDL example)

Added by Scott Larson, last edited by Tony Storey on Mar 21, 2013

Example VHDL Code

debounce.vhd

DBounce.vhd

Introduction

Using mechanical switches for a user interface is a ubiquitous practice. However, when these switches are actuated, the contacts often rebound, or bounce, off one another before settling into a stable state. Several methods exist to deal with this temporary ambiguity, using either hardware or software. Here, we look at correcting this problem with a simple digital logic circuit (a common task when interfacing FPGAs or CPLDs with pushbuttons or other switches). Generic VHDL code to implement this methodology is included.

Theory of Operation

Figure 1 illustrates the debounce circuit in question. The circuit continuously clocks the button's logic level into FF1 and subsequently into FF2. So, FF1 and FF2 always store the last two logic levels of the button. When these two values remain identical for a specified time, then FF3 is enabled, and the stable value is clocked through to the result output.

The XOR gate and N-bit counter accomplish the timing. If the button's level changes, the values of FF1and FF2 differ for a clock cycle, clearing the N-bit counter via the XOR gate. If the button's level is unchanging (i.e. if FF1 and FF2 are the same logic level), then the XOR gate releases the counter's synchronous clear, and the counter begins to count. The counter continues to increment in this manner until it (1) reaches the specified time and enables the output register or (2) is interrupted and cleared by the XOR gate because the button's logic level is not yet stable.

The counter's size determines the time required to validate the button's stability. When the counter increments to the point that its carry out bit is asserted, it disables itself from incrementing further and enables the output register FF3. The circuit remains in this state until a different button value is clocked into FF1, clearing the counter via the XOR gate.

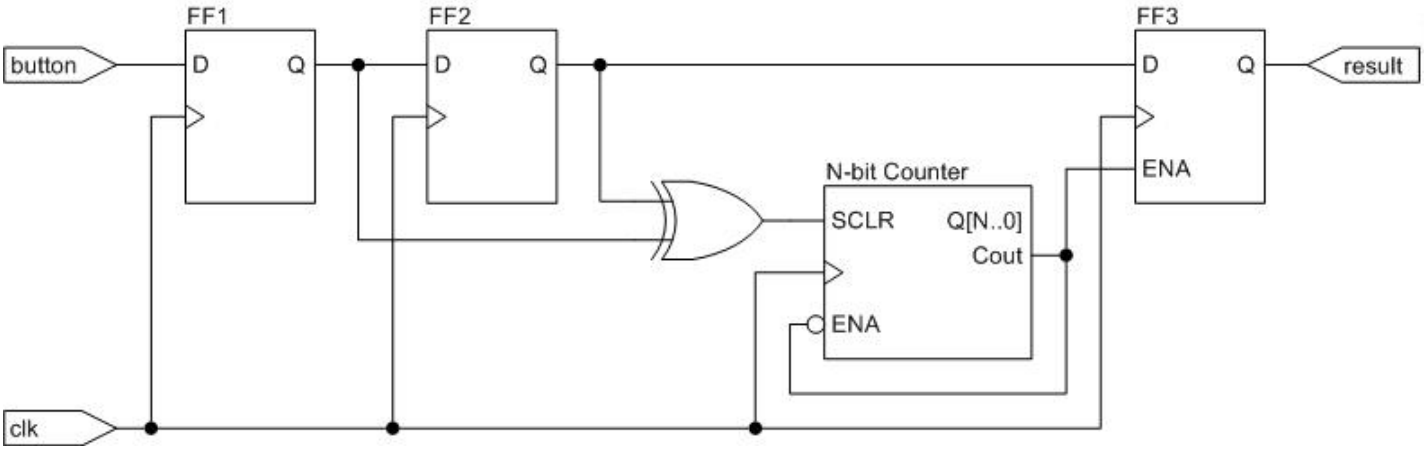

Figure 1. Debounce Circuit

Sizing the Counter

The size of the counter and the clock frequency together determine the time period *P* that validates the button's stability. Equation 1 describes this relationship.

(1)

$$P = \frac{(2^N + 2)}{f} \approx \frac{2^N}{f}$$

In typical applications, the number of clock cycles is large, so the additional two clock cycles from loading FF2 and FF3 can safely be disregarded.

Most switches reach a stable logic level within 10ms of the actuation. Supposing we have a 50MHz clock, we need to count 0.01*50,000,000 = 500,000 clock cycles to reach 10ms. A 19-bit counter fulfills this requirement. Using the counter's carry out pin, as shown in Figure 1, eliminates the requirement of evaluating the entire output bus of the counter. With this method, the actual time implemented is (2^{19}+2) / 50,000,000 = 10.49ms.

Debouncing typically does not require a high level of resolution, so the relatively small error introduced by using the carry out pin to identify the validation time is adequate for most applications. However, if greater time resolution is desired, and-ing some of the counter's most significant bits achieves this with minimal additional logic usage.

Example Code and Simulation

The example code *debounce.vhd* available here instantiates the circuit in Figure 1. The GENERIC *counter_size* defines the size *N* of the counter. Figure 2 shows a simulation for this design with the counter size set as 2 for demonstration and readability purposes.

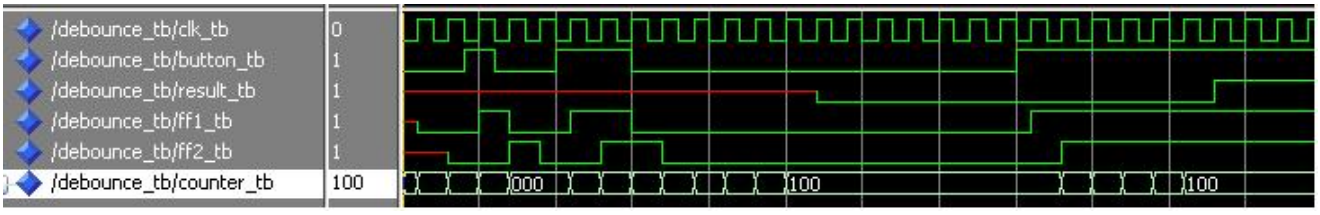

Figure 2. ModelSim Simulation Results

An alternative code implementation is also provided.

Conclusion

This simple logic circuit addresses mechanical switch debouncing for programmable logic.

Feedback for Our Sponsor

Please take a few seconds to help us justify the continued development and expansion of the eewiki.

Click on one of our Digi-Key links on your way to search for or purchase electronic components.

Is the eewiki helpful? Comments, feedback, and questions can be sent to eewiki@digikey.com.