Debounce Logic Circuit (with VHDL example)

Code Downloads

Version 2.0: debounce.vhd

Added asynchronous active-low reset

Made stable time higher resolution and simpler to specify

Version 1.0: debounce_v1.vhd

Initial Public Release

Features

- VHDL source code to debounce mechanical switches and buttons
- Configurable time the input is required to be stable
- Configurable system clock frequency

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. The debounce component presented here is a simple digital logic circuit that addresses this temporary ambiguity (a common task when interfacing FPGAs or CPLDs with pushbuttons or other switches). Figure 1 illustrates a typical example of this Debounce component integrated into a system. This component was designed using Quartus Prime 17.0.0 Lite Edition.

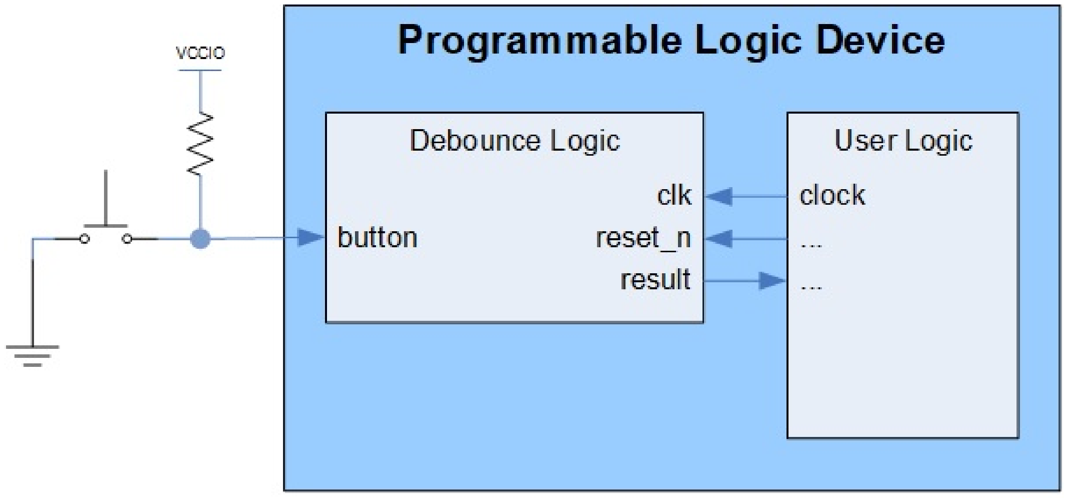

Figure 1. Example Implementation

Theory of Operation

Figure 2 illustrates the debounce circuit. It 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 the counter accomplish the timing. If the button's level changes, the values of FF1and FF2 differ for a clock cycle, clearing the 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.

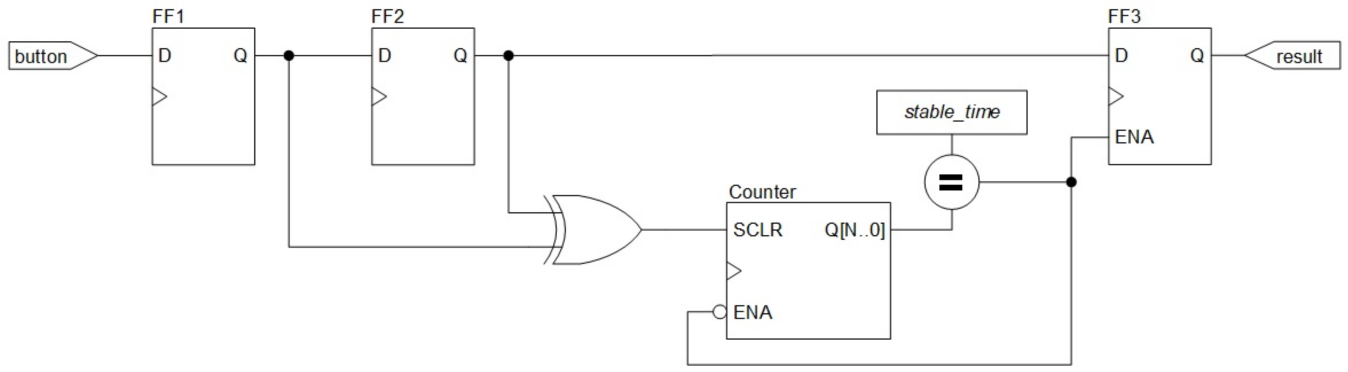

Figure 2. Debounce Component Circuit

Configuration

The Debounce component is configured by setting the GENERIC parameters in the ENTITY. Table 1 describes the parameters. Most switches reach a stable logic level within 10ms of the actuation, so the *stable_time* parameter's default value reflects this.

Table 1. Generic Parameter Descriptions

Generic	Data Type	Default	Description
clk_freq	integer	50_000_000	Frequency of the system clock input (PORT clk) (Hertz)
stable_time	integer	10	Time an input must remain stable to be considered valid and debounced (milliseconds)

Port Descriptions

Table 2 describes the Debounce component's ports.

Table 2. Port Descriptions

Port	Width	Mode	Data Type	Interface	Description
clk	1	in	standard logic	user logic	System clock
reset_n	1	in	standard logic	user logic	Asynchronous active low reset
button	1	in	standard logic	button or switch	Input signal prior to debounce
result	1	out	standard logic	user logic	Debounced signal

Reset

The *reset_n* input port must have a logic high for the Debounce component to operate. A low logic level on this port asynchronously resets the component. During reset, the component clears the three flipflops, setting the output result to '0'. Once released from reset, the Debounce component resumes operation.

Conclusion

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

Appendix: Additional Information on Version 1.0

Version 1.0 of this design used the N-bit Counter's size to determine the time required to validate the button's stability. Figure 3 depicts this circuit. 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.

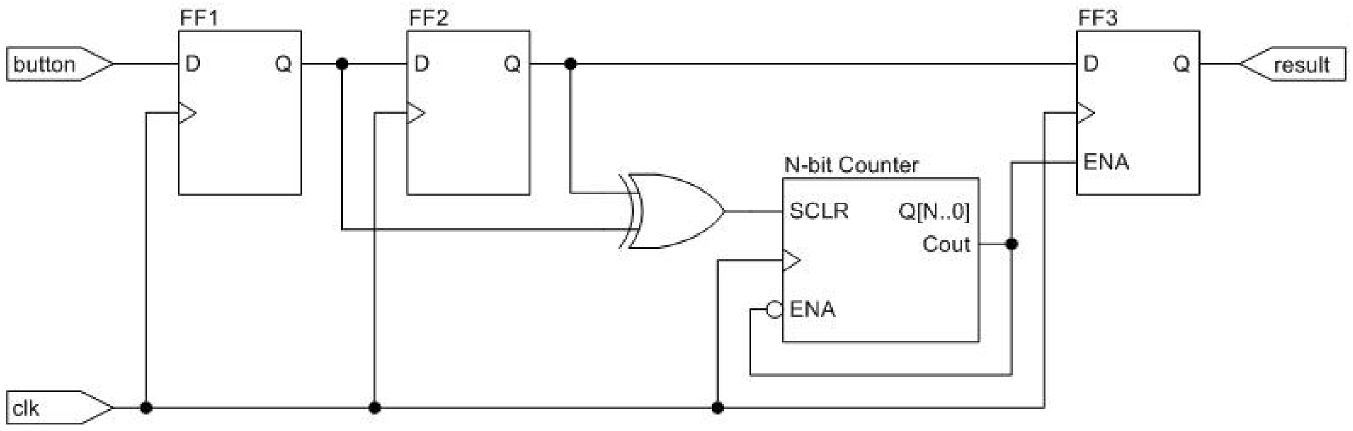

Figure 3. Version 1.0 Debounce Circuit

For this approach, 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.

$$(1) \qquad P = \frac{(2^N + 2)}{f} \approx \frac{2^N}{f}$$

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, Version 2.0 fulfills this requirement.

Contact

Comments, feedback, and questions can be sent to eewiki@digikey.com.