G1 Debounce External Signals

Jesse Larson, Jing Lu, and Qingqing Liu

The goal of this app note:

To show you how to use a our variable length and timing sampling debouncer

Why is this needed:

Put simply all non-digital signals coming into the DE2 board should be debounced. The fact is if your signal lingers a little too long in the voltage range between ‘1’ and ‘0’ this will cause your circuit to perform some interesting and rather baffling behaviour. It is also need with mechanical devices like switches as their signal bounces between ‘1’ and ‘0’ many times before settling at its final value. This will cause several switch flips or button presses to be logged by your circuit when you really have only flipped the switch once. Also be sure to confirm that the device you want to debounce does not have a voltage bounce that exceeds a board friendly value. If you do add a simple RC debounce circuit to protect the board before passing its value into this component.

Understanding how it works:

This hardware component continually samples the signal you want to debounce. When a given number of sample values are the same component changes its output to match the sampled value.

There are three generics in my component you need to set to get the debouncing time you want.

1. CYCLES_TO_SAMPLE. This sets the timing interval that the component samples the signal on. This value is in relation to the clk signal you give the component
2. SAMPLE_SIZE. This is the number of samples that all must have the same value in a row for the output to be changed
3. INTIAL_INPUT_VALUE. This should be set to ‘1’ or ‘0’ depending on what you know the initial value of your external signal is to avoid an improper first output

There are three ports into my custom component.

1. clk - this is simply the clock frequency input

2. bounce_sig - this is connected to the GPIO pin you want to bebounce
3. debounce_sig – this outputs a clean ‘1’ and ‘0’ logic based off of the previous values of the GPIO pin you connect it to

The demonstration:

What you need:

- DE2 Board
- Some signal to debounce

Setting up the demo:

- Simply restore the Quartus II project from the archived Debounce_P.qar file
- Compile the design
- Program it to the DE2 Board
- Connect the signal you want to debounce to the GPIO_0(0) pin.

Using the demo:

The debounce component interacts with another component in this demo which simply counts rising edges from the output of the debounce component. The total count of rising edges to the green leds is output to the green leds, this way you can test if you are are properly debouncing your signal.

Congratulations:

You now have a custom component that debounces an external signal. This component can also be used to filter out noise. If noise occurs in small bursts these output changes can be smoothed by having the total time required to alter a signal longer then the length of the noise burst. This can also cover up some hardware failures with infrared transmitters if used properly.

Cheers,

Jesse Larson, Jing Lu, and Qingqing Liu.