Lab Assignment 7
Controlling a Counter with Push Buttons by Hardware

Introduction

In this lab we will show how to read the state of the push buttons by hardware, and how to deal with the signal bouncing effects introduced by these simple input devices. We will begin with a basic design where the state of an LED is controlled by a push button. This design will be progressively extended to the point where the push buttons will allow us to reset a counter, enable or disable its periodic increments, change its counting direction and speed.

Pre-Lab Assignment
The following reading list will help you to understand how push buttons work so that we can make use of them in the lab assignment.

Require Reading: Contact Bounce
http://www.allaboutcircuits.com/textbook/digital/chpt-4/contact-bounce/

You don’t need to turn in anything for this pre-lab.

Lab Assignment

Connecting to ZedBoard

1. Login into the Windows desktop PC (we will refer to this system as the host) using your myneu credentials.
2. Plug in and connect the power cable to your Zedboard.
3. Connect the USB cable, to the ZedBoard JTAG port (the microUSB port next to the power plug on the ZedBoard) and your PC.

Opening your required tools

4. Open the “System Generator” from All Programs -> Xilinx Design Tools -> ISE Design Suite 14.4 -> System Generator.
5. With Matlab open, open the Simulink library browser by clicking on its icon or typing simulink in Matlab command line.

Overall Lab Goal

This lab will guide you through the steps necessary to develop a pushbutton-controlled counter, similar to the one implemented in software in Lab 3. This time, you will design the counter in Simulink, and run it on the FPGA. The overall functionality and switch assignment is as follows:

Switches:	define the start value of the counter
PBTNL:	load counter value from switches

PBTNC:	start or stop counting
PBTNU:	increase speed
PBTND:	decrease counting speed
PBTNR:	change the direction (i.e., increment vs. decrement)

Do not try to design this all at once. The design would be too complex. As with any design, it is a good idea to start with small steps. The following steps guide you through the steps individually.

This lab assignment will ask you to reuse the components built in Labs 5 and 6. Only minor modifications should be needed. If you find yourself designing a lot of new logic, you probably did not take full advantage of the components developed in the previous labs.

Lab 7.0 Turn LED on/off with a push button

Out first goal is configuring the ZedBoard to control the state of an LED using the center push button (PBTNC). Every time this push button is pressed, the LED should change its state, from on to off, or from off to on. This behavior can be achieved with a simple free-running 1-bit counter, whose output is connected directly to the LED. A 1-bit counter is one that can take only values 0 and 1. When it reaches its maximum value 1, the counter overflows and goes back to 0.

The *enable* input of the counter should be activated whenever the push button is pressed, that is, whenever a transition from 0 to 1 is found in the button input signal. This behavior can be tricky to implemented, since the physical imperfection of the contacts within a push button typically causes signal bouncing, as illustrated in the figure below. Every time a button is pressed, its output signal will quickly transition between 0 and 1 before it takes a final stable value of 1.

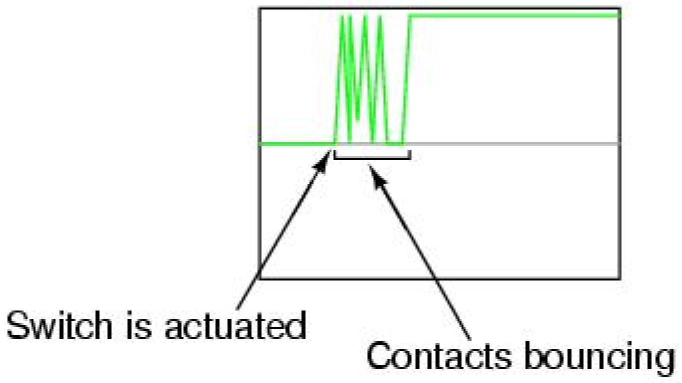
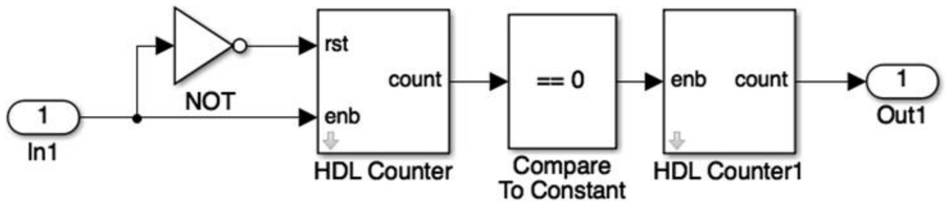


Figure 1 Contact bouncing when actuating a switch viewed on an oscilloscope.
Please see the description at http://www.allaboutcircuits.com/vol_4/chpt_4/4.html for more details.

To solve this problem, we will design a cascaded counter in order to guarantee that a signal coming from a push button has settled with a value of 1 for long enough before we consider the button to have been effectively pressed.

1. Create a cascaded counter design and save it as *button_to_led.slx* (similar to Lab 6.7 design).
2. Configure the first counter to have a *Local reset port* and a *Count enable port*. Connect the *enable* port to an *In1* component, which will later be mapped to the center push button (PBTNC). This makes the first counter increment its value for as long as the button is pressed.
3. Add an inverter (NOT gate). Connect its input to the push button signal (PBTNC) and its output to the *reset* port of the counter. This makes the counter go back to 0 when the bush button is released.

4. Configure the comparator in such a way that it yields 1 only after the button signal has been set to 1 for 250ms straight. Remember that the HDL counter is running at a frequency of 50MHz, which will determine the value for the comparator's constant. Calculate this value for the comparator (currently set to 0).
5. Configure the first counter with type *Count Limited*. As you observed in the previous lab assignment, the maximum value that the counter can take determines the frequency at which the *Compare to Constant* block is sending a pulse set to 1. In our case, this will determine the frequency at which the LED will blink if we just keep the push button pressed for a longer period of time. Configure the *Count to Value* property of the first counter to make the LED blink at 1 frequency of 1Hz (once per second) when the push button is kept pressed.
6. Configure the second counter as a 1-bit free-running counter. Connect the output of the counter to an *Out1* component, which will be mapped into an LED. Your design should look like this (except for the value shown in the *Compare to Constant* block):
7. 

8. For simulation purposes, insert a *Scope* component in your design. Configure the scope to have three inputs: double-click on the scope, click on the *Settings* icon in the pop-up window, select tab *General*, and enter “3” in field *Number of axes*. The scope now has three inputs, which you can connect to the output of the first counter, the magnitude comparator, and the second counter, respectively.
9. By default, the scope limits its history to the last 50,000 cycles. In order to allow the timing diagram to record a signal for the entire simulation, double-click on the scope, click on the *Settings* icon in the pop-up window, select the *History* tab, and uncheck option *Limit data points to last*.
10. Also for simulation purposes, you can temporarily replace the *In1* port by a *Step* logic block followed by a *Data Type Conversion* block. You can configure the *Step* block to mimic a user pressing the push button, by generating a pulse of 500ms—this period should be given in number of cycles. You will also need to set the *Sample Time* property of the *Step* block to 1.
11. Simulate your design during 10s of virtual time, using the appropriate amount of simulation cycles. In order to keep reasonable simulation speeds, you can assume that the FPGA clock runs at a frequency of 50kHz (instead of the actual 50MHz). This means that you can divide the counter limits, constant values, and number of cycles by 1000.
12. Synthesize your design into the FPGA using the HDL Workflow Advisor. Follow the directions presented in the previous lab to upload your design into the FPGA.

Assignment 1
Take a screenshot of your design and add it to your lab report. Explain the numeric values chosen for the Compare to Constant component and the counter limits during the simulation.
Add another screenshot for the simulation of your design, showing the complete output of the Scope component, appropriately zoomed in (you can use the Autoscale button). Describe the output you observe, and justify its correctness.

Lab 7.1 Start / stop counter
Implement an 8-bit counter that outputs its value to the LEDs, and can be controlled by pushbutton PBTNC to start/stop counting. Save this model as LedCount.slx.

Hint: utilize the cascaded counter design (lab 6.7) as a basis for the counter. Use the result of lab 7.0 (turning LED on/off with a pushbutton) above to enable (start) / disable (stop) the counter.
Define the counting speed to be 2Hz as used in lab 6. Report in your lab report how you set the parameters of the counter.

Assignment 2
Validate the functionality on the ZedBoard. Add a screenshot of your new design. Report in your lab report how you set the parameters of the counters, and why.

Lab 7.2 Change Direction
Expand the design of LedCount.slx to toggle the direction of counting with each push of PBTNR. Modify the properties of the 8-bit HDL counter to add a Direction port, and use the same design principles you applied in part 7.1 for enabling/disabling the counter. Save this model as LedCount_dir.slx.

Assignment 3
Validate the functionality on the ZedBoard. Add a screenshot of your new design. Report in your lab report how you set the parameters of the counters, and why.

Lab 7.3 Load Value
Expand the design of LedCount_dir.slx to allow the user to set a starting value of the 8-bit counter through the switches (the number is entered as a binary). The 8-bit counter should load the value from the switches only if the button PBTNL is pushed. Otherwise, the 8-bit counter should count freely, as designed before. Save this model as LedCount_load.slx.

Note: you can configure the 8-bit counter to expose a load port. It will then also automatically expose a “load enable” port. If the load enable is 1, then the counter is loaded with the value present at the load port. Note that the load enable port should be active for only one cycle. Hint: the control of the load enable is very similar to 7.1 for enabling/disabling the counter with small modification.

Assignment 4
Validate the functionality on the ZedBoard. Add a screenshot of your new design. Report in your lab report how you set the parameters of the counters, and why.

Lab 7.4 Change Counting Speed
Extend the design LedCount_load.slx to control the counting speed of the 8-bit counter. The PBTNU pushbutton should increase the counting speed, and PBTND pushbutton should decrease the speed. Save this model as LedCount_speed.slx.

Hint 1: Expose the reset port of the first counter in the cascaded counter design. Implement a circuit to trigger a reset when the counter reaches a certain value even if the counter is set to "free running mode". The reset value can be set by an additional HDL counter that changes output value by an appropriate step value when one of the push buttons is pressed. PBTNU should decrement the value to increase speed and PBTND should do the reverse to decrease counting speed.

Hint 2: How can you use two push buttons to control one input?

Reminder: Downloading the design to the FPGA takes about 15 minutes. Simulate your design in Simulink and make sure it works before you download it to the platform.

Assignment 5
Validate the functionality on the ZedBoard. Add a screenshot of your new design. Report in your lab report how you set the parameters of the counters, and why.

Show the functionality of design to the instructor or the TA for validation. If you finish at any other time, record a short video showing the functionality of all the push buttons.

Laboratory Report – Full Report

You should follow the lab report outline provided on Blackboard. Your report should be developed on a wordprocessor (e.g., OpenOffice or LaTeX), and should include graphics when trying to present a large amount of data. Include the output of compiling and running your programs. Upload the lab report on blackboard. Include screenshots of your design in the appendix of your lab report.