Lab Assignment 6
Programming FPGA using Simulink

The goal of this lab is to develop a digital design and download the design on to the FPGA using Simulink. You will be using the Simulink HDL Coder to program the FPGA on the ZedBoard. The lab will span a minimum of two week. Over these weeks we will first connect switches to LEDs, then control the LED with a simple counter, and finally control an LED with a push button.

Instructions

- Each lab assignment consists of a set of pre-lab questions, the actual time in the lab, and a lab report.
- The pre-lab assignments prepare you for the challenges you will be facing in the actual lab. So that each lab member gets the same benefits, pre-lab assignments have to be solved individually. Feel free to discuss problems with class and lab mates, but complete the assignment on your own.

Reading List

The following reading list will help you to complete the pre-lab assignment. The readings will also help you in subsequent lab assignments. Please complete the readings that are marked ***[Required]*** before attempting the pre-lab assignments.

[1] Review materials on designing a counter
- http://www.facstaff.bucknell.edu/mastascu/elessonshtml/logic/Logic5.html

[2] Chapter 2 of Simulink tutorial, how to program FPGA
- https://**blackboard**.**neu**.edu/

[3] Mathworks materials on HDL coder
- https://www.mathworks.com/products/hdl-coder/?s_tid=hp_fp_list

Pre-Lab Assignment

We have only very limited time for the lab available. To make efficient use of the lab time, you will need to prepare for it. First, go through the assigned reading above and get an overview of the schematics and the manuals mentioned. Second, approach the questions below to apply your knowledge. Please submit the solution to the pre-lab assignments in PDF format via blackboard.

Counter design

Pre 6.1) Analyze counter behavior from Simulink.

1) Open Simulink.
2) Create a new Simulink Model. Clink on the "New" icon and select the Simulink Model. Wait for the model to get created (check the lower left area to see if MATLAB is busy, it may take a few minutes to start).
3) Simulink offers different solvers to cover everything from continues models (e.g. for control theory) to discrete HDL (as in our case). To make this Simulink model correctly simulate counters and discrete logic, select discrete solver with a fixed step size of 1. Open the configuration settings: "Simulation" -> "Model Configuration Parameters".
4) In the dialog "Configuration Parameters" select:
   a) Solver: "discrete (no continuous states)",
   b) Type: "Fixed-step"
   c) Max step size: 1
5) Click on the "Library Browser" icon (4 boxes, 2 of them red and blue).
6) Allocate "Manual Switch"
7) Allocate constant (from group Source), double click to get its properties. In "Signal Attributes", change the "Output data type" to Boolean. In main tab, set Constant value to 0.
8) Copy and paste the constant input to get a second input. Change the value to 1.
9) Connect 0 constant to top input of switch, 1 constant to bottom input of switch.
10) Allocate "HDL Counter" from the library. Connect the enable input to the switch output. Double click the counter to open its properties and select "Counter type" as free running. Ensure that "Word Length" is set to 8 bits. In properties check 'enable port', since the default HDL counter does not have the port 'enable'.
11) Allocate a "Scope" from the "Sink" Category.

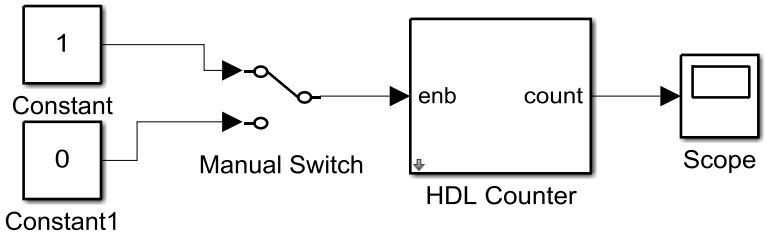

Figure 1 Simulink model to analyze counter behavior

12) Save the Simulink model (shown in Figure 1) as "counter_simple"
13) Simulate the design for 1000 cycles. Describe what you observe on the scope with depending on the switch position.

a) What is the functionality of "enb" of the counter?
b) Include in your report a print of the Simulink design, and representative scope outputs.

Pre 6.2) Counter with Comparator
14) Refine the counter_simple from the previous assignment.
15) Add a "Compare to constant" block. Place it in between the counter and the scope.
16) Change the compare value to 25.
17) Save the model into "counter_compare"

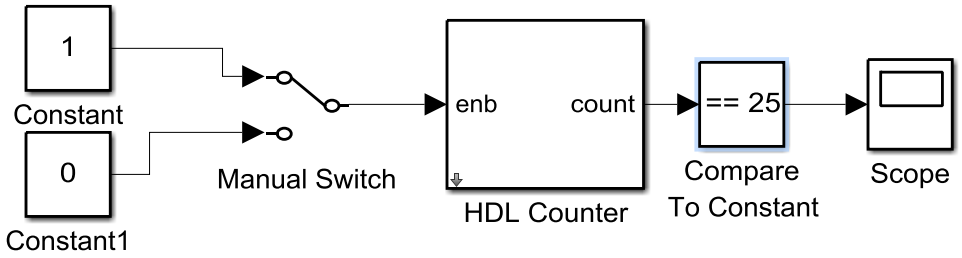

Figure 2 Simulink model to analyze counter with comparator

18) Simulate the design (shown in Figure 2) with the switch turned to the "1" position and describe your observations on the scope. What was created?

Pre 6.3) Cascaded Counter
19) Refine the counter_compare from the previous assignment.
20) Copy the counter, and insert it between the scope and the comparator.
21) Add another scope between after the first counter.
22) Save the model into "counter_cascaded"

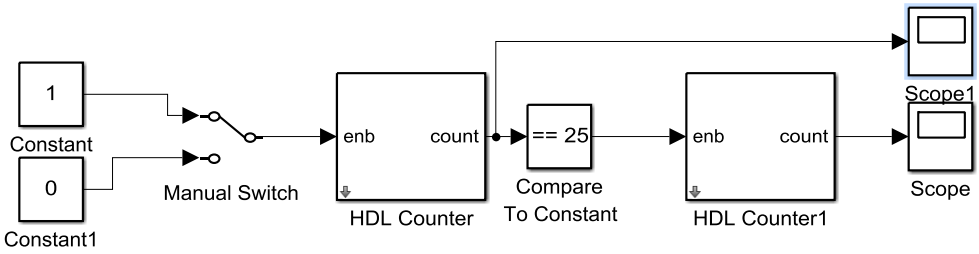

Figure 3. Simulink model to analyze counter cascading

23) Simulate the design (shown in Figure 3), observe the scope output with switch in "1" position.
24) Describe your observations between the scopes. What was constructed by cascading the two counters?

Pre 6.4) Cascaded Counter Control
Discuss, how can the counting speed of the second counter be impacted? What has to be done to increase or decrease the counting speed of the second counter?

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. Make sure your Zedboard is powered (plugged in and the power switch on the board is in the ON position), and has been given time to boot up (generally takes a few minutes). Then connect the Zedboard to the PC host via Ethernet to the RJ45 connector.
3. Make an SSH connection to the Zedboard, where the IP address is 192.168.1.10, and port number of 22.
   Don't work as root, you already created your user accounts, use that!

Opening your required tools

1. Open the "System Generator" from All Programs -> Xilinx Design Tools -> ISE Design Suite 14.4 -> System Generator.
2. In opened MatLab, open Simulink library browser via clicking on its icon or typing simulink in matlab command line.

Lab 6.1 Programming FPGA: Connecting the switches to the LEDs

The following steps will create a design that will connect the switches on the ZedBoard to the LEDs on the ZedBoard using the FPGA.

1. Create a new Simulink Model. Click on the "New" icon and select the Simulink Model. Wait for the model to get created (check the lower left area on the screen to see if MATLAB is busy, it may take a few minutes to start). Click on the "Library Browser" icon (4 boxes, 2 of them red and blue). From the Simulink library, find the "In1" component in the "Sources" section. Drag the "In1" component and drop it into your model.
2. Double click on the "In1" component you have added to the model, and change the data type in the Signal Attributes to "boolean". Also change the Sample time to 1.
3. Add an "Out1" component from the library (see the Sinks section). Change the type to boolean.
4. Connect the input (In1) the output (Out1).
5. Select both input and output. Then, copy and paste them 7 times.
6. Select all components (Ctrl + a), right click on them and click on "Create Subsystem from Selection".
7. Right click on the subsystem you have created in the previous step, and click on "Block Parameters". Select the option "Treat as atomic unit".
8. Next, click on the "Code" menu and from the HDL Code submenu, and click on the "HDL Workflow Advisor".

9. From System Selector window (which is now open), select your subsystem. The "HDL Workflow Advisor" will be opened.
10. From "Set Target -> Set Target Device and Synthesis Tool", select "IP Core Generation" as the Target workflow. Select the ZedBoard as the Target platform. Select Desktop (the system Desktop) as the Project Folder, and click on "Run This Task".
11. In the "Set Target Interface" tab, you will see the inputs and outputs for your design. Assign each input to "DIP Switches", and each output to "LEDs General Purpose". Click on "Run This Task".
12. In "Prepare Model For HDL Code Generation" select the "Check Global Settings" tab and click on "Run This Task". When this finishes, click on "Modify All", and then click on "Run This Task" again.
13. Right click on "Generate RTL Code and IP Core" and click on "Run to Selected task". When done, click on OK in the Code Generation Report.
14. In "Embedded System Integration", select the "Create Project" tab and click on "Run This Task".
15. In "Generate Software Interface Model", select "Skip this task", and then click on "Run This Task".
16. In "Build FPGA Bitstream", click on "Run This Task". In this step, Simulink will run the Xilinx tool in the background to generate a bitstream of your design for the FPGA on the ZedBoard. You need to wait for this step to finish. It takes around 10 minutes.
17. Next, connect the USB cable, to the ZedBoard JTAG port (the microUSB port next to the power plug on the ZedBoard) and your PC. Power up the ZedBoard, wait for it to boot.
18. In "Program Target Device", click on "Run This Task".
19. Wait about 2 minutes. The FPGA will be programmed, and the ZedBoard will be restarted.
20. Play with switches on the ZedBoard and see the result. Record what you see.

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

Lab 6.2 Control LEDs with a Counter (cascaded counter)

Create a design with a free running 8bit counter and show the output on the LEDs in binary. Note, when synthesized to the ZebBoard, the HDL counter (when enabled) will be counting at a frequency of 50 MHz (which is very fast). This will be too fast for the human eye to be observable. To reduce the counting frequency, use a cascaded counter design is introduced in pre-lab 6.3.

1. Instantiate the cascaded counter design from Pre-lab 6.3. Name the design as "led_count.slx". Configure the second counter to be eight bit free running. Configure the comparator to compare with zero. Configure the first counter (which counts at 50MHz)

so that it resets to zero twice per second (every 500 ms). Change the counter type to "count limited" and adjust the "count to value" accordingly. Report your reasoning for the dimensioning in your lab report.

2. Using your bit slicing module developed in Lab 5.1, distribute the output of the second counter to 8 Boolean values and connect each bit to an 'OUT'
3. Validate the correct functionality of the design through simulation. To perform validation, use a scope (or many display blocks) to observe the outputs. In your report, briefly indicate the errors that you found (if any) through simulation.
4. Start the HDL workflow advisor that you learned how to use in Lab 6.1. Refer to 6.1 to program the FPGA. Now the 'OUT's will be mapped to the LEDs.
5. Validate that the design works correctly on the FPGA. In particular, measure the speed at which the LEDs count up. Does the speed match what you intended earlier? Report your findings.

Lab 6.3 Turn LED on/off with a push button

Create a design that turns an LED on and off by pushing a button. This can be achieved by using a 1-bit counter. The counter should increase by one each time the button is pushed (i.e., at a 0-to-1 transition). Each 0-to-1 transition increases the count by one: first from-0-to-1 turning on the LED, and then from-1-to-0 turning the LED off.

However, this is actually harder than it seems upon first glance due to the problem of bouncing on the contacts. A push button when pushed does not deliver a clean off-to-on transition. Instead, it bounces back and forth a few times, as illustrated in Figure 4 below.

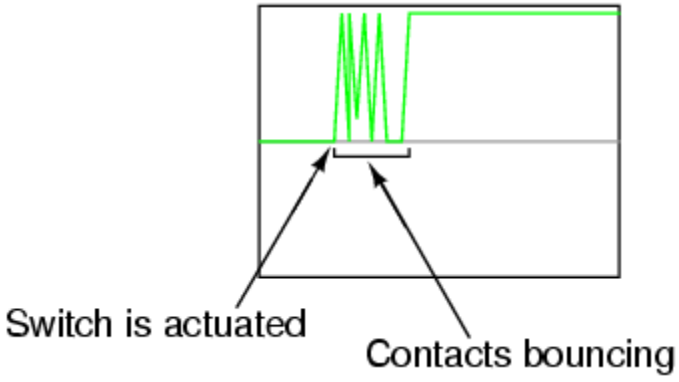

Figure 4 Contact bouncing when actuating a switch viewed on an oscilloscope.

Please see the description at http://www.allaboutcircuits.com/vol_4/chpt_4/4.html for more details. To solve this problem, we will use the cascaded counter design from above to make sure that the pushbutton was pushed long enough.

1. Use the cascaded counter as a template. Save the design as "push_onoff.slx". Configure the first counter (double-click on the counter block) to have a "local reset port" and a "count enable port". Connect the enable port to the pushbutton input (PBTNC). With this, the first counter increases as long is the button is pushed. Now, add an inverter and

connect it to the reset port of the same counter. With this, the first counter is reset each time the pushbutton is released.

2. Configure the compare to constant block at the output of the first counter so that it yields true after the button is pushed. The comparison value in the Debounce Compare in Figure 5 will to be changed to wait 250ms. Recall that the HDL counter is running at 50 MHz. **You will need to change the value (currently 0 in the figure) to the value you compute.** Show your calculations in your report.
3. The Bounce Counter is currently of type "count to". The maximum value (i.e. when the rolls over) determines the repeat rate of the button push (in our case the LED would blink). Compute the "Max Value" so that the repeat rate is 1 seconds. This means if the button is pushed for longer than 1 second, the LED turns on and off every 1 second.
4. Configure the second HDL counter to be a one bit free running counter. Connect the output of the counter to an LED.

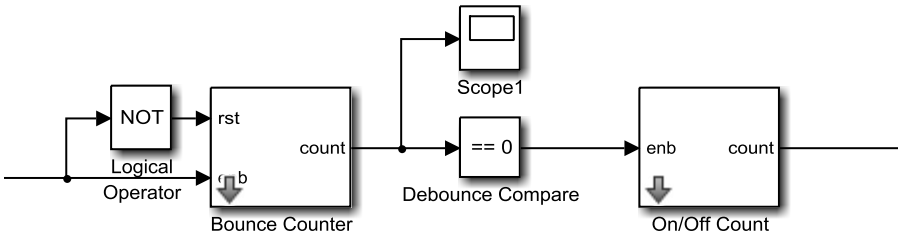

Figure 5 Debounce and turn on / off by push button

5. Validate your design through simulation. Report briefly the errors found through simulation.
6. For your lab report, create a diagram that shows the value of the "Bounce Counter" as it increases over time, as well as the On/Off count. In that diagram also highlight the thresholds for "Bounce Compare" and the Max Value of the "Bounce Counter".
7. Use the instructions from the previous steps (Lab 6.1) and follow the HDL workflow advisor to synthesize the design onto the FPGA. Validate collect operation on the FPGA, and report your findings including errors that you may observe.

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

Laboratory 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. Upload each program that you wrote on blackboard in their respective submission links.