Lab Assignment 5

Introduction to Simulink to develop a digital design

The goal of this lab is to provide an introduction to Simulink, and then use Simulink to develop a digital design.

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] Simulink tutorial in blackboard
https://blackboard.neu.edu/

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.

Tutorial overview

Pre 5.1) Go through the first chapter of Simulink tutorial [1] to design your half adder. Name the design *HAdder*. Include a screenshot of your resulting Simulink model as part of the pdf submitted in your pre-lab writeup.

Pre 5.2) Design a 1-bit full adder (two 1-bit inputs and 2 outputs, a 1-bit carry out and a 1-bit sum) with logic gates in Simulink. Include a screenshot of your design in your pre-lab report.

Number Representation/Combinational Logic

Pre 5.3) Design a combinational circuit (i.e., show the Boolean formula) that takes as an input the individual bits of an 8-bit unsigned integer value (i.e., bit0 … bit7). It should output 1 (true) if the number is exactly 42, and zero in all other cases. Discuss your design in your pre-lab report.

Lab Assignments

This lab introduces digital logic design with Simulink. In this lab you will start to get accustomed to implementing your digital logic designs with Simulink.

Lab 5.0 Opening Matlab 2014

1. From the Start menu, find Matlab 2014a browsing all programs or type Matlab 2014a. Note your computer may have different versions installed. Please use 2014a if available. Then, open Simulink library to start your design.

Lab 5.1 Extract individual bits from an integer

2. Create a new Simulink Model by clinking on New (the plus sign)->Simulink Model. Click on the “Library Browser” icon (4 colored boxes). From the Simulink library, in the “Sources” section, find the “In1” component. Drag the “In1” component and drop it into your model. In the following steps, you will add components to your Simulink Model. Change the name of your model to Decimal.
3. Now we should define the *type* of the input signal. Double click on the Decimal component you have added to the model. In the “Signal Attributes” tab, change the data type to “int8”. Also change the Sample time to 1.
4. To access individual bits from the input 8-bit integer, we will use the “Bit Slice” operator. This will allow you to “slice out” the desired bit (basically only accessing the desired bit(s)); Search in the Simulink Library for “Bit Slice” and drop it into your model. Next, we need to define which bits in our design we are interested in working with. Let’s use this bit slice for the least significant bit. Double click on the bit slice, and select under MSB (Most Significant Bit) position “0” (meaning start slicing from bit 7), select “0” for the LSB (Least Significant Bit) position (meaning, slice until bit 0). This will output only the LSB.
5. Copy the slice operator and paste it 7 times (for the other 7 bits). For each operator, change the MSB and LSB position accordingly (e.g., for the MSB 1, specify LSB 1 which is the second bit).
6. Next, add 8 “Out1” components from the library (Sinks section). Make sure the types are specified as “Inherit: auto”.
7. Connect all components as shown in Figure 1. You can use the ‘Ctrl’ key to connect a component to other components. Note, the design in Figure 1 works for unsigned ints, but not for signed ints. Attempt to fix the most significant bit for signed ints.
8. Select all components (Ctrl + A), right click on them, and click on “Create Subsystem from Selection”.
9. Now we need to test our bit sliced design. Delete the input and outputs. Add a “Constant” from the library (Source section). Also add 8 “Displays” (Sink section) to your design. Connect the Constant to the input of the Subsystem, and displays to the outputs. Also change the data type of the Constant to int8. You can change the Constant Value to any desired value you want to test for.
10. Next click the “Run” icon (the green arrow), and discuss the results that you see.

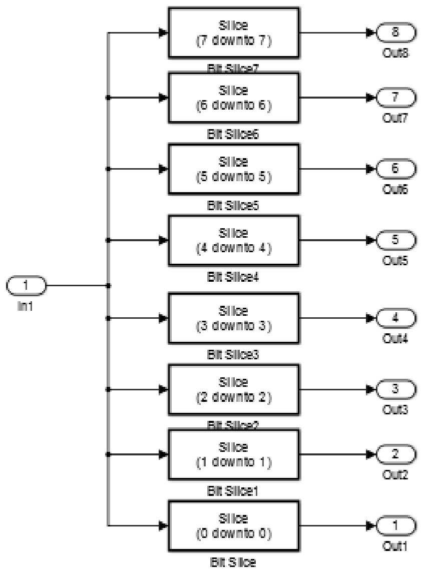

Figure 1 Slicing an 8bit integer into individual bits

Lab 5.2 Concatenate individual bits into a uint8

11. Next, we will need to design the opposite of what we just designed in 5.1, in order to combine individual bits into a single int8 value. Develop another Simulink model that takes as input 8 bits and converts them to an 8-bit value. Hint: use the Bit Concat block and change the configuration to "Number of Inputs": 8.

Lab 5.3 Design a Full Adder cell with Simulink

12. Develop a Full-Adder Cell. A Full-Adder cell has three inputs (a, b, Cin) and two outputs (sum, Cout).

Lab 5.4 Design an 8-bit Full Adder with Simulink

13. Using your previous designs, develop an 8-bit adder. The model will receive two inputs (int8) and calculate the sum.
14. Add the necessary logic to your adder to detect overflow.
15. Validate your design with examples, some should produce overflow, and some should not. Report experiments and your results.

Lab 5.5 Number Comparison

16. Use the modules developed in 5.1 to generate individual bits out of an 8-bit integer. Then, design a combinational circuit that outputs a '1' if the input value is exactly 42 (base 10). It should output zero in all other cases. Hint: use the formula you developed in Pre 5.3 to guide your design. Validate your design with different numbers. Discuss your testing procedure.
17. Expand your design to use a comparator out of the Simulink library. The comparator should be fed as input the 8-bit value (not sliced into the individual bits), and output '1' if the value

equal to 42, or ‘0’ if not equal to 42. Validate that both your combinational logic and the comparator yield the same result.

18. The Simulink comparator can perform a comparison with any arbitrary value. In contrast, your logic in step 16 was explicitly coded for one particular value (i.e., the design would need to be modified if we wanted to compare against a different value). In your report, reflect on how you think the comparator is implemented at the logic gate level in Simulink.

Lab 5.5 Extra Credit

19. Design a multiplier. The design will receive two inputs (int8) and calculate the product.

Laboratory Report

You should follow the lab report outline provided on Blackboard. Your report should be developed on a word processor (e.g., OpenOffice or LaTeX), and should include graphics when trying to present a large amount of data. Include screenshots in your report in the appendix.

Note: Use PrntScr key to get screenshot, and use “mspaint” or similar tools to edit your screenshots.