Lab 2: Verilog and FPGA Refresher (cont.)

Objectives

The objective of this lab is to review some Verilog and FPGA basics and to familiarize you with Vivado Design Suite and the Nexys A7 Trainer Board.

This lab is organized as follows:

1. Introduction
2. Experiments to be conducted on the Nexys A7 using Vivado (should be conducted during the lab session and shown to the instructor)
    a. Experiment 1: A 4-bit ripple carry adder (RCA)
    b. Experiment 2: Seven Segment Display Negative Number
    c. Experiment 3: Addition of two signed numbers
3. Deliverables: Online Submission regulations and lab report questions.

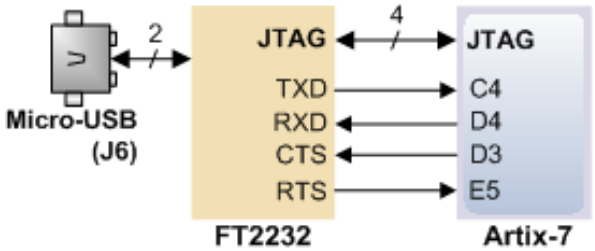
Introduction

An adder is a digital circuit that performs addition of numbers. Adders are not only used in ALUs (Arithmetic Logic Units) in PCs but are also used in address calculation, Program counters increments, stack pointers increments, etc.

There are many types of adders but our focus will be on the ripple carry adder-subtractor (RCA). Then we will extend our experiment to handle signed numbers. Negative numbers are represented as two's complement with the most significant bit showing the sign (1 → negative, 0 → positive)

Supplying inputs remotely

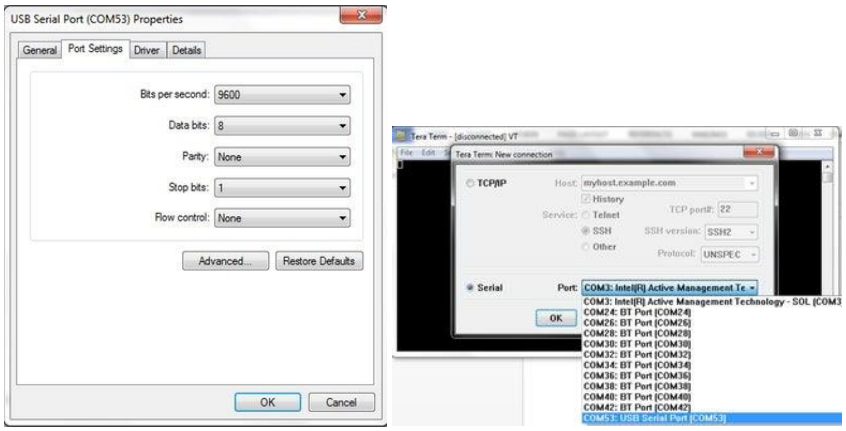
When you remotely access the lab PC and monitor the connected FPGA board via the PC camera, you still can simulate switch inputs and button presses remotely. As shown in the figure below, the Nexys A7 board includes an FTDI USB-UART bridge (attached to connector J6 that we connect our USB cable through) that allows you to use PC applications to communicate with the board using standard Windows COM port commands. This J6 connector acts as a shared UART/JTAG USB port. The JTAG port is used to program the FPGA, while we can rely on the UART to simulate switch inputs and button presses remotely.



Nexys A7 FT2232HQ connections

Add the UART receiver module to your code (provided on BB), and instantiate it. The module receives the 100MHz clock and from the TXD pin C4 receives the ASCII value of the character you send using your PC keyboard. The module gives a 0/1 output on each keyboard button press that you can use as an input signal in your module. So make sure to set your user constraint file properly accordingly. To test the UART receiver, use its User Constraint File provided on BB.

Our J6 port is connected to our lab PC, you can find the right port at the Device Manager (if you have window PC). The port setting should be 9600 baud rate as in the figure below. The figure also shows how to send the Windows COM port commands using the Tera Term tool.
Tera Term can be downloaded from this link: https://ttssh2.osdn.jp/index.html.en, then you can open the ttermpro executable; you can download and use the zipped portable version if you are not admin on the lab machine and cannot install new software.



Setting the baud rate in a Windows PC, making the connection and sending the Windows COM port commands using Tera Term tool

Experiments

Note: Students should be working in pairs to implement the following experiments

Experiment 0: Send UART data to supply inputs to FPGA remotely

a. Create a new RTL project targeting the Nexys A7 board using Vivado.
b. Use the UART_receiver modules and the associated constraint file (provided on BB) to examine how to supply eight inputs remotely to the FPGA.
c. Test your program by providing inputs through keyboard buttons a,s,d,f,z,x,c,v using TeraTerm software, and monitor the corresponding FPGA LEDs values.

Experiment 1: A 4-bit ripple carry adder (RCA)

a. Create a new RTL project targeting the Nexys A7 board using Vivado.
b. Write a new Full-adder module in Verilog.
c. Design and implement a 4-bit RCA using full adders and `generate` blocks. Eight switches of the Nexys A7 board should be used to provide two 4-bit inputs to the adder. The 5-bit result (including carry out) should be displayed on one of the two 4-digit seven segment display using the driver module you implemented in the previous lab.

d. Create the necessary constraint file.
e. Program the Nexys A7 board and test it.
f. Record both the utilization and the delay of the critical path. You can find the delay in the timing report after you have completed the implementation step.

Note: You need to show your instructor the result of this experiment within Lab.

Experiment 2: Seven Segment Display Negative Number

We implemented the Four Digit Seven Segment display in the previous lab. In this lab we will modify Four Digit Seven Segment to be (1 Sign + 3 Digit) Seven Segment.

a. Create a new RTL project targeting the Nexys A7 board using Vivado.
b. Design and implement a (1 Sign + 3 Digit) 7-segment display driver module controlled by the 8 input switches. The Nexys A7 has 16 switches, but we choose to use only 8 switches to specify the binary value we want to visualize on the 7-segment display. What is the minimum and maximum number that could be displayed?
    i. Copy files from Lab1 (Experiment#3). Don't modify on the same files.
    ii. A positive number should show no sign.
    iii. A negative number will show a negative sign (which segment should be lit?)
    iv. The most significant bit of the input indicates its sign (1 → negative, 0 → positive)
    v. Negative numbers are provided in two's complement format. E.g. -7 → 1111 1001, 7 → 0000 0111)
    vi. What should be modified to handle negative numbers?
c. Create a constraint file attaching:
    i. The inputs of the module to switches 0 to 7 (eight bits only)
    ii. The outputs of the module to the cathodes and anodes of the 7-segment LEDs.
    iii. The 100 MHz clock input to pin E3
d. Program the Nexys A7 board and test it.
e. Record the utilization (number of LUTs, FFs, and IO ports) consumed by the driver circuit you designed. You can find this information after you have completed the implementation step in the utilization report.

Experiment 3: Addition of two signed numbers

a. Create a new RTL project targeting the Nexys A7 board using Vivado.
b. Use Modules from Experiment#1 and Experiment#2 (**Copy them, don't override**).
c. Using the designed 4-bit RCA from Experiment#1, modify it to handle **4-bit signed numbers.** Negative numbers are represented as two's complement and the most significant bit shows the sign (1 → negative, 0 → positive).
d. The **4-bit** (without carry) result should be displayed on one of the two 4-digit seven segment display using the driver module you implemented in Experiment#2.
e. Create the necessary constraint file.
f. Program the Nexys A7 board and test it.
g. Record both the utilization and the delay of the critical path. You can find the delay in the timing report after you have completed the implementation step.

Deliverables

General Report Submission Notes:

- Each student is required to submit an individual report.
- For Verilog descriptions or constraint files, please attach the .v and .xdc files (not the whole project) or copy and paste the code. No photos or screenshots accepted.
- The submission should be a zipped folder with the following Structure
    a. YourID_Report.pdf
    b. Exp1 : Folder containing your codes for experiment#1
    c. Exp2 : Folder containing your codes for experiment#2
    d. Exp3 : Folder containing your codes for experiment#3
- **Submission deadline**: half an hour before the start of next week's session
- **Submission method**: On Blackboard

Lab Report [10 pts]

Report of Lab2 should be a single pdf/word file and should include (in sequence):

1. [0 pts] Your name and student ID.
2. [2 pts] A technical summary of experiments conducted in the lab (Steps, Results, components (a screen shot with Schematic design might help), code functionality, etc...] and any Verilog descriptions or constraint files you wrote for the lab.
3. [1 pts] Results recorded in the last step of experiment 1 & 2 & 3 (Photos or screen shots)
4. [2 pts] Compare the utilization and delay of experiment 3 vs experiment 1, please add your comments on the results.
5. [3 pts] Repeat experiment 1, but this time use a 4-bit carry lookahead adder (CLA). Report the utilization and the delay. You might need to do a little research to know how a CLA works. Write your own Verilog code. You are NOT allowed to copy Verilog code from ANY source.
6. [2 pts] Compare and comment on the results obtained for the RCA vs the CLA adder types.
