Introduction to Electronic System Design (DAT093) Lab 4: Clocking, I/O, and FPGA board verification

Sven Knutsson, Lars Svensson

Version 2.1, September 28, 2018

1 Introduction

In this lab session, you will use your knowledge of hierarchical design in VHDL to implement a small system including input and output signals which connect the FPGA to simple peripherals. You will make use of the VHDL `package` concept to group several design elements and hide their internals. You will learn how to adapt processing rates using clock-enable signals (a useful technique when dealing with slow peripherals). You will also take design verification one step further by synthesizing your design and downloading it to an FPGA board, and verify the functionality there. These tasks require you to refer to the documentation for both Vivado and the FPGA board.

In this and subsequent labs, you will work in pairs; the pair assignments should be posted well before the scheduled Lab-4 sessions. Get in touch with your lab partner in advance and *work through the preparation steps together*, or at least (if you have had to prepare individually) *compare your notes and solutions* before coming to the lab.

In the lab, you will be sharing one workstation and one keyboard. Thus, when one student is typing, the other student is reduced to observing, in the manner of "pair programming"[1]. Your learning will benefit greatly if you change roles regularly!

[1] `https://en.wikipedia.org/wiki/Pair_programming`

2 Preparation

This lab introduces configurable hardware in the form of a Nexys4 FPGA board, where you will verify the results of your synthesis. The reference manual for this board is available on the course homepage. Familiarize yourself with this manual, particularly with section 9! You will not need all the information in the manual, but you should be able to find the parts you need.

The Vivado synthesis tool needs some information about the board to generate correct output. This information is specified in a *constraints file* in `.xdc` format. Two constraint files are posted on the course homepage (this is since we have two versions of the FPGA board; check if the `NEXYS4` logo printed on your board includes the text `DDR` and choose constraint file accordingly). Again consult the document "Introduction to Xilinx Vivado". In Lab 3, we left off before the heading **Program and Debug**. Now continue past that point to find out how to modify the constraint files as needed in this lab, and how to actually download the design to the board.

The document "Hints on clocking", which is available on the homepage, outlines design practices for clock definition and distribution in FPGA implementations, in particular when different parts of a design have different speed requirements. Read through this document.

Two new VHDL concepts are introduced in this lab: the `function` declaration and the `package` declaration. Make sure that you understand what they do.

Read through this entire lab PM and work through all tasks marked **Preparation**.

3 System description

Your task is to design a small system that lets you manually store a number of data words and display them on 7-segment LED data displays. The block diagram is shown in Figure 1; the blocks to be implemented are described further in Appendix A. The board has sixteen switches, eight 7-segment displays, and several push buttons; all of these will not be used in this simple design.

Data will be entered using slide switches available on the board, and read into storage when a button is pressed. Each 7-segment display will be used to display one hexadecimal digit, i.e., four bits of information (half a byte, sometimes known as one "nibble" or "nybble"). You will need to store and display eight such

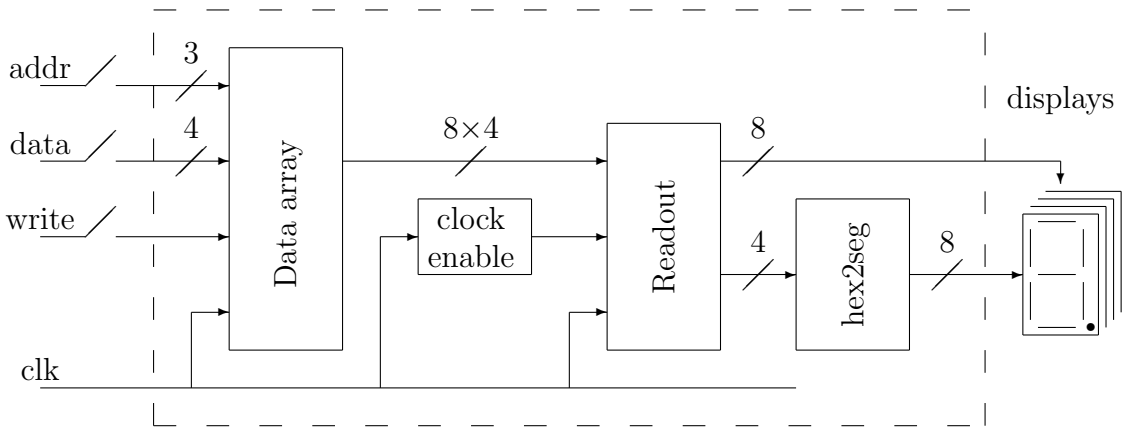

Figure 1: Tentative system block diagram. The blocks involved are described further in Appendix A. The dashed box indicates the FPGA contents.

hexadecimal digits simultaneously; thus, your total storage requirements will be eight nibbles. Both data values and storage addresses will be read from the slide switches when the button is pressed.

The 7-segment displays do not work directly with the hexadecimal digits. Each segment of the display can be turned on and off with a separate signal; the decimal point is an additional segment, for a total of eight control signals. Thus, to display a hexadecimal value, you need to convert the stored hexadecimal data to the individual segment control signals.

The eight displays are addressed in a time-multiplexed, round-robin fashion: one display at a time is activated by its individual selection signal and then displays the data present on the segment inputs. Thus, displaying a value entails the following steps:

1. read the data nibble from the correct storage position in the data array,
2. convert the data nibble from a hexadecimal value to the segment control signals,
3. write the segment control signals to the displays,
4. and activate the correct display selection signal.

Note that two types of 7-segment LED displays are in common use: the *common-cathode* type, where the eight LED cathodes (negative poles) are wired together, and the *common-anode* type where the positive poles are all connected. The practical difference is whether the signals are active-low or active-high. Consult the board manual to determine what polarity you should use.

Preparation: Work through the design with your lab partner to produce implementations and test benches for each block needed for the system. It is recommended that you stay close to the block diagram of Figure 1; if you stray from it, the lab TAs will need more time to familiarize themselves when you ask for assistance. Appendix A lists additional considerations for each of the blocks in Figure 1.

Testing and verification is done during the lab session.

- Implement each of the design blocks according to your preparations. Follow the general procedure used in the previous labs. Create test benches and *do* files and verify each of your blocks with simulations, again as illustrated in previous labs.
- Create a top-level design as in Figure 1, using your design blocks as components, and test-bench and do files for it. Simulate the top-level design to verify its functionality.

In order to correctly synthesize your design, you will need to edit the FPGA-board constraint file to specify to what FPGA pins your input and output signals should be connected. Thus, you need to select which of the slide switches, displays, and buttons to use.

- Download the constraints file for your board, and open it for editing.
- Using the information from "Introduction to Xilinx Vivado", assign pins to ports as needed to realize the system in Figure 1. *Document your choices of slide switches and push buttons.*
- Synthesize your design using your modified constraint file.

The final verification of your design is to download the synthesis results to the FPGA board and check the functionality.

- Following the instructions in the companion documents, download the bitstream to the FPGA card. Verify that the system works as specified.

To be marked as passed for the lab, show your working design to the lab TA and be prepared to discuss the design.

4 Wrap-up

After completing this lab session, you are expected to be able to carry out the following tasks:

- Define and use a VHDL `function` to specify combinational behavior.
- Define and use a VHDL `package` to group associated types and functions.
- Generate and use clock-enable signals to adapt processing speed to slow peripheral units.
- Define simple synthesis constraints to identify FPGA pins with VHDL ports, and thereby use simple input and output devices available on the FPGA board.
- Download a synthesized bitstream file to an FPGA board and manually verify design functionality.

A Blocks

Note: In addition to the inputs and outputs described below, all sequential blocks need some means for resetting their internal state.

A.1 Data array

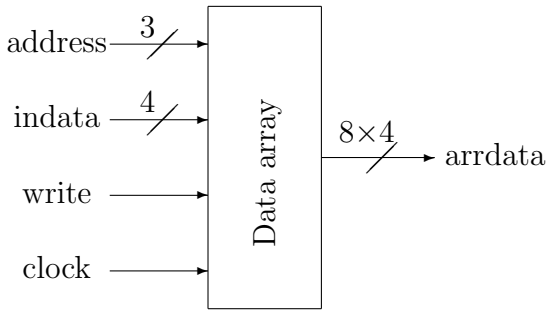

Figure 2: Data array block

The data array stores the data values and therefore must be represented as a sequential design. A natural way to represent eight four-bit nibbles in VHDL is as a two-dimensional array of bits. All the stored bits are made available at the block output port as nibbles. For storing data into the array, there are four input signals: a three-bit address value; a four-bit data value; a write signal; and a clock signal.

The two-dimensional signal type of the block output port will be needed also in the readout block (Section A.2) and in the top-level design connecting all blocks. It is beneficial to declare a VHDL `package` containing this type, and make the type visible through a `use` statement.

A.2 Readout

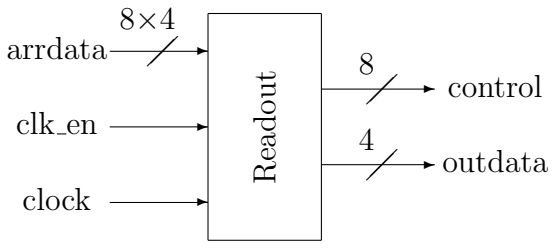

Figure 3: Readout block

The readout block must sequentially select the nibbles from the data array block and present them at its data output port. The bits of the control port are set in sequence[2] so as to select one of the displays, at the same rate as the data nibbles are presented at the output port.

The data rate requires some consideration. The LEDs in the 7-segment displays are quite slow, so they cannot be activated at the 100-MHz clock rate used by the FPGA. It might be possible to generate a slower clock signal for the readout block; in an FPGA implementation, however, it is often preferable[3] to distribute as few clocks as possible and instead use a periodic clock-enable signal to reduce the processing rate when necessary. This is the reason for the clock-enable input in Figure 3. Generation of the clock-enable signal is described in section A.3.

A.3 Clock enable

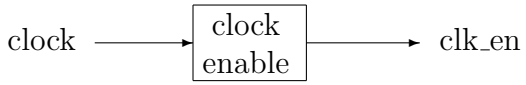

Figure 4: Clock enable block

The clock-enable signal is intended to adapt the processing rate of the FPGA to the slow LED displays. The clock enable signal should be asserted for one cycle out of every N cycles. A good way to build such a block is to use a generic counter design with N as a parameter. (You have built similar blocks in previous labs!)

With N as a parameter, it is easy to try different display refresh rates by changing the parameter value and resynthesizing. When verifying the design on the FPGA board, try a few different values. In what value range for N does the display look stable without flickering or other artifacts? What is the corresponding range of update frequencies?

A.4 Data conversion

Translation from the hexadecimal values stored in the data array to the values needed for the 7-segment display is best described with a VHDL function which takes a four-bit input value and returns an eight-bit value (the eighth bit controls the decimal point which should be off for all input values; refer to the Nexys4

[2]With short sequences such as this one, you may consider rotating a single set bit in an eight-bit vector. (Why would this solution be less suitable for longer sequences?)

[3]Refer to "Hints on clocking" for a somewhat deeper discussion.

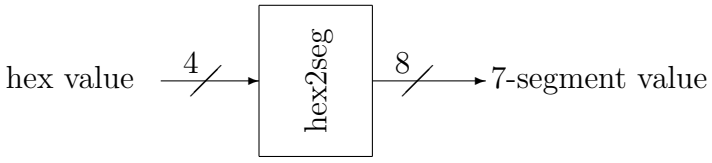

Figure 5: Data conversion block

board manual for a description of the mapping from hexadecimal to 7-segment values). Declare a type in your `package` for the eight-bit output values, and also place the function itself inside the package.