Introduction to Electronic System Design (DAT093) Lab 5: A sampled system

Sven Knutsson, Lars Svensson

Version 2.1, October 8, 2018

1 Introduction

The previous lab session introduced code synthesis and verification on a development board, including the handling of simple peripherals on that board. This lab session takes the concept one step further by introducing somewhat more complex peripherals, namely, analog-to-digital and digital-to-analog converters (ADCs and DACs). Such converters are very common in embedded systems which need to interact with the environment through sensors and actuators.

You will use the converters MCP3202 and MCP4822, both manufactured by Microchip, and communicate with them via a Serial Peripheral Interface (SPI). The converters are not included on the Nexys4 board, but rather on a separate expansion board (such boards are commonly used when developing a system from selected components). The converters use 12-bit words and each has two channels (that is, an analog signal which corresponds to the 12-bit digital value may be sent to or read from two separate circuit pins). To use the converters, you will need to find out how digital 12-bit values can be transfered to and from them, and generate the necessary control signals in the FPGA using VHDL code.

In this lab, the signal converted into digital form by the ADC will simply be passed to the DAC to be converted back to the analog domain (some extra processing will be added later in Lab 6). The converters are *unipolar*, that is, they assume that the analog voltage is strictly positive. The expansion board includes circuitry to shift the signal if desired such that its DC level is 0.

Just as Lab 4, this lab and its preparations are carried out in pairs. To make the most of your lab session, set aside preparation time with your lab team-mate before coming to the lab.

2 Preparation

You will use the same FPGA lab system as in Lab 4. Make sure that you have its documentation at hand. The expansion board with the two converters is described in the companion document "The AD/DA Board". You will use this documentation to design the VHDL code that uses the converters.

Find the data sheets of the converters online[1]. In addition to the wordlength and the number of channels, the important specifications are the conversion rate (that is, the maximum number of conversions per second) and the digital communication interface. The Serial Peripheral Interface (SPI) is described in the converter data sheets (more information is available in a Wikipedia entry[2].) Read up on this interface and make sure you understand how it works. Many flowcharts can also be found on the web; you may find it useful to draw some of your own which describe how to communicate with these chips.

Study the data sheets to find the maximum *conversion rate* possible with each of the two circuits, and the minimum *SPI clock frequency* needed to support a certain conversion rate with each of the circuits (the SPI protocol needs to finish the transfer of one value before continuing with the next one). The SPI clock frequency will be lower than the typical FPGA clock frequency. Re-acquaint yourself with the clock-enable block of Lab 4.

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

3 DAC control

The DAC is simpler to control than the ADC, since digital communication is unidirectional from the FPGA to the converter.

Preparation: Find an SPI clock frequency that will support a conversion rate of 40kHz in both converters.

[1]Designers are nowadays expected to be able to find component data sheets and such information without special prompting, using Google or similar search engines. In this and future courses, you should not expect such information to be made available in any other way.

[2]`https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus`

Preparation: Start from your Lab-4 clock-enable block and add an output signal which has the same frequency as the original clock output but with a a 50% duty cycle.

Preparation: Drawing on your experiences from previous labs, design hardware that will transfer a 12-bit value to an output port, following the SPI protocol as used by the MCP4822 DAC. As specified in the converter data sheet, the complete SPI command also includes control bits to select the DAC channel, to optionally enable a 2× voltage gain, and to optionally turn off the output. Choose one of the channels and enable the 2× gain. Use a clock-enable block to generate the necessary clock frequency for the SPI signalling.

Implementation and testing follow the now-familiar steps.

- Implement the SPI protocol hardware in VHDL and test your design by simulation as in previous labs.

To test your design on the development board, you need to provide digital input data to the DAC. A simple way is to use the slide switches on the board to set consecutive bits of the data word, and read them into the FPGA at the press of a button, as you did in Lab 4.

- Modify the Nexys4 `.xdc` file so that a 12-bit input value may be read into the FPGA from the slide switches. Also make it possible to use one of the board LEDs to indicate that a conversion is complete, and enable one push button to start the conversion. Document your choices of LED, push button, and slide switches.
- Select and configure the clock-enable reduction factor and the FPGA board clock frequency to achieve an SPI clock signal of the required frequency[3] as calculated during preparations.
- Synthesize your design. Download the synthesized design onto the board and test its functionality by entering a digital value with the slide switches and use a voltmeter to observe the voltage value at the DAC output. Use the push button to start a new conversion and the LED to indicate end of conversion.

Demonstrate your design to the lab TAs at this point.

[3]The Nexys4 manual outlines briefly how to configure the FPGA clock signals.

4 ADC control

Communication with the ADC requires bidirectional communication, with control signals going from the FPGA to the ADC and the converted data coming back the other way. As descibed in the ADC data sheet, the SPI protocol supports such operation.

Preparation: Implement an SPI-interface design which handles the reading of data from the peripheral unit. Choose one of the two channels and use the MSB-first format. Use a push-button to start conversion and a LED signal to signal completion.

Testing of the ADC system may be done by using the Nexys4 LEDs to display the digital converted values. Use a potentiometer connected from +5V to 0V to generate an analog voltage to use as the input for the ADC.

- Implement the new SPI interface design and simulate it to verify correctness.
- Further modify the Nexys4 `.xdc` file to allow control of 12 LEDs to display the ADC data bits.
- Synthesize the extended design. Download to the FPGA board and verify the functionality.

Again demonstrate your design to the lab TAs.

5 Combined system

The two converter interfaces will now be connected back-to-back, such that each value read from the ADC will be sent unmodified directly to the DAC. You will need to use your two SPI interfaces to communicate with the two peripherals.

In the combined system, conversion will not be controlled by a pushbutton but rather by a sampling clock, which controls the conversion rate. The sampling clock may be derived from the system clock in the same way as in Lab 4.

- Implement a combined system with two SPI interfaces to the ADC and DAC. Samples from the ADC are to be passed directly to the DAC for conversion back to the analog domain. Verify by simulation (you will need to create "dummy" signals to be read through the ADC interface and written back through the DAC interface, and verify that the values are identical).

- Synthesize and download the design to the FPGA board. Test it by applying a sine wave from a signal generator and inspect the result with an oscilloscope.

Show your design to the lab TA, with the signals displayed on the oscilloscope.

6 Wrap-up

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

- Seek out information on hardware peripherals as needed for implementing interfaces to them.
- Implement and configure SPI interfaces for bidirectional communication with simple peripheral circuits.