Analog to Digital Converter

1 Objective

We want to pilot an electric train using an FPGA BASYS-3. This board will control the electric motor of the train by a PWM command and the train speed will be measured by reading the back electromotive force of the motor using an ADC MCP3002.

In this first part, we focus on the reception of the speed measurement. A sensor generates a voltage proportional to the motor speed, then this voltage is measured by an analog-to-digital converter, called MCP3002. The latter transmits this digital value obtained to the FPGA by an SPI bus.

The documentation and datasheet are given in appendix.

2 The SPI protocol

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol, which has the advantage of being in full duplex (communication in both direction is allowed simultaneously). SPI uses a master-slave architecture with a single master. The master device initializes the communication, and sets the clock that gives the flow rate. Multiple slave-devices are supported by this protocol, but in our case we will only have one slave: the ADC. The Figure 1 present a basic SPI bus, with a single slave.

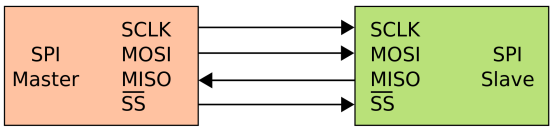

Figure 1: SPI bus with single master and single slave (source: Wikipedia)

The SPI bus has four logic signals:

- $SCLK$: Serial Clock (given by the Master)
- $MOSI$: Master Out Slave In (data written by the Master)
- $MISO$: Master In Slave Out (data written by the Slave)
- $\overline{CS}$: Chip Select (given by the Master) (sometimes called $\overline{SS}$ for Slave Select)

A typical frame of 8 bits is presented on Figure 2. The Serial Clock is setted by the Master on $SCLK$. The Master indicates that he wants to initiate a communication, by setting $\overline{CS}$ down. Then, as soon as there is a tick on the Serial Clock, the communication begins. The Master sends its message bit by bit on $MOSI$, and the Slave on $MISO$. The communication ends when the Master set $\overline{CS}$ up.

Note that there are several modes of SPI communication, the only one that is presented and used afterwards is this one: writing is done on a $SCLK$ falling edge, for the Master as for the Slave. So to ensure that the reading of the values is good, the reading will be done on the rising edge of the $SCLK$.

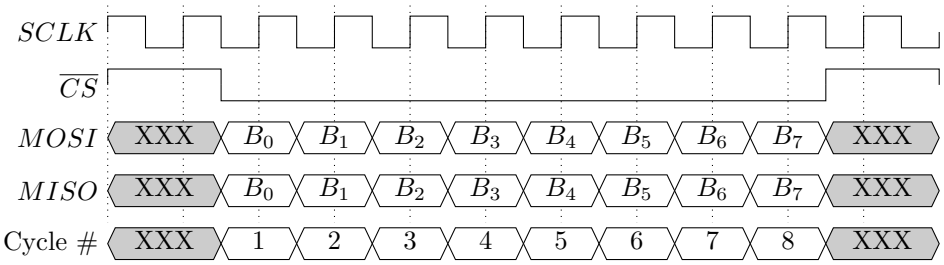

Figure 2: Simple SPI frame

3 The board

We use in this lab the Basys 3 development board (Figure 3) which is based on an FPGA of the family Artix-7 by Xilinx. The documentation of the board is given in the file *basys3.pdf*.

We built an extension board embedding the Analog-to-Digital Converter MCP3002 (Figure 4), which connects to the basys 3 using a PMOD port. A description of the extension board is given in the file *mcp3002_board.pdf*.

The ADC MCP3002 communicates using an SPI protocol (via the PMOD port). This component can take two analog inputs ($CH0$ and $CH1$), but we will only use the first channel $CH0$, to measure the speed of the motor. Its documentation is given in the file *mcp3002.pdf*.

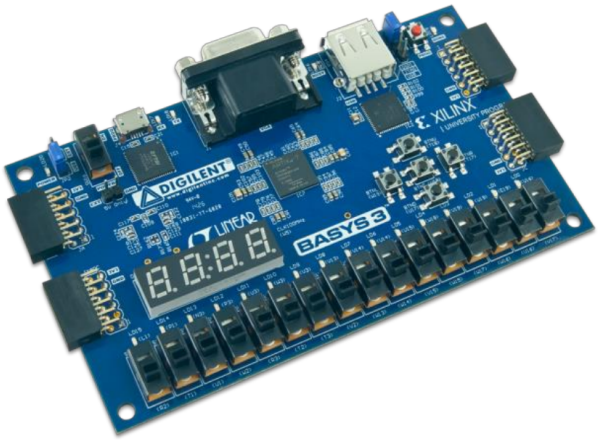

Figure 3: The basys 3 board

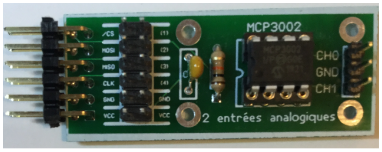

Figure 4: The extension board with MCP3002

To simplify the work, we will assume in the following that we have direct access to the MCP3002's I/O from the FPGA.

In order to unify the notations, in your VHDL code you will use :

- **not_cs**, **d_in**, **d_out**, **sclk** for the ports of the MCP3002;

- **clk** for the clock of the FPGA;
- **an**, **seg**, **led** for the 7-segment display and the LEDs.

4 Let's begin !

1. Find the communication specifications in the MCP3002's datasheet. Identify the names of the four signals used in the SPI protocol ($SCLK$, $MOSI$, $MISO$ and $\overline{CS}$). Deduce the I/O of your VHDL component, and write the VHDL Entity (do not forget the FPGA clock).
2. Find in the documentation the clock frequency of the Basys 3. Do the same for the MCP3002 component (we assume that the component is supplied with $3.3V$, thus take the frequency corresponding to $V_{DD} = 2.7V$). Deduce the VHDL of a clock divider to generate $SCLK$.
3. Write a VHDL simulator generating the clock of the Basys 3. Check by simulation its frequency. Then verify by simulation the good functioning of the clock divider (check that the frequency of $SCLK$ matches the frequency of the MCP3002).
4. We want to use the Single-Ended mode, on the channel $CH0$, with the MSB first format. Find the corresponding chronogram in the MCP3002 datasheet. Write the VHDL generating the sequence for $\overline{CS}$ and D_{in}.

   Tip: Break the communication frame into a finite number of steps, and give a value to $\overline{CS}$ and D_{in} at each step. Remember that writing is done on a falling edge of $SCLK$, and reading is done on a rising edge of $SCLK$.
5. Verify the value of $\overline{CS}$ and D_{in} by simulation.
6. Read the values returned by the MCP3002, and store them in a vector *data*.
7. At this stage, we provide a simulation file *main_sim.vhd* which simulates the behaviour of the MCP3002. When the sequence of $SCLK$, $\overline{CS}$ and D_{in} is correct, it will send the sequence on D_{out}: 0 (Null Bit) then a 10 bits value. If it is not the case, it will send only 0 on D_{out}.

   Make sure that the simulator uses the correct name of component, and I/O ports. Verify the correct behaviour of your component on simulation.
8. Compute the measured voltage, from the returned value stored in *data*. You will need to find in the MCP3002's datasheet the relation between the measured voltage V_{in}, the reference voltage V_{ref} and the returned digital value. As a reminder, the voltage provided is $V_{ref} = 3.3V$.
9. Display the value of *data* (10 bits) on the 10 LEDs of the board.
10. Display the computed voltage on the 7-segment display, with just 2 digits: one for units, and one digit after the decimal point. Reuse the VHDL code of the first lab for this question.
11. Simulate the correct behaviour, using the file provided *main_sim.vhd*. Uncomment the lines in the file to add the ports *an* and *seg*.
12. Verify the correct behaviour of your code on the real board.

5 Report

The report should answer each of the questions of the subject. When a specific VHDL code is expected, include it in the report, and explain how it works. When a simulation is expected, include the VHDL code of the simulator and a screenshot of the result, and comment on what you see.

The report should be saved in PDF format. All the VHDL files of the design should be given in appendix. Their can be found in the folder of your project: *[project-name]/[project-name].srcs/sources/new/* (with your own *[project-name]*).

Finally, all the files must be compressed in a ZIP archive, with the filename *[LastName1]-[LastName2].zip* (with your own *[LastName#]*), and sent by e-mail to the address *remi.parrot@ec-nantes.fr*.