Martin Skriver
maskr@mmmi.sdu.dk
RMEMB1 fall 2018

# Lab assignment part 2/2

In this exercise it is only allowed to have the pl_top.vhd module instantiated in the block design.

There will be no hand-ins for this exercise, but make sure you take the notes you need to use the modules in the final project and for reading up for the exam.

## IMPORTANT:

- DO ONLY USE 3.3 V WHEN WORKING WITH THE SoC.
- Only connect RX(SoC output), TX(SoC input) and GND from the FTDI cable to the SoC. DO NOT connect the VCC wire to the FPGA.

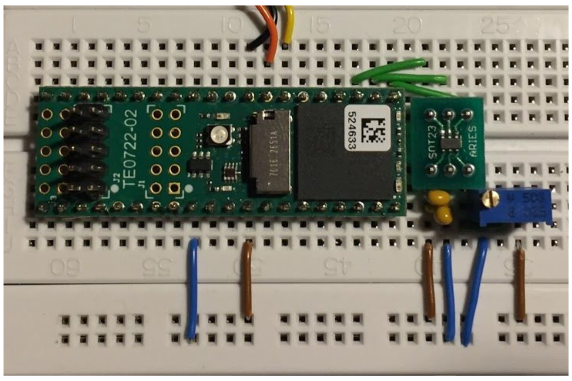
## Materials:

- TE0722 propeller SoC board
- TE0790 XMOD programmer
- Breadboard
- FTDI cable (TTL-232R-3V3-WE)
- Ads7885 ADC

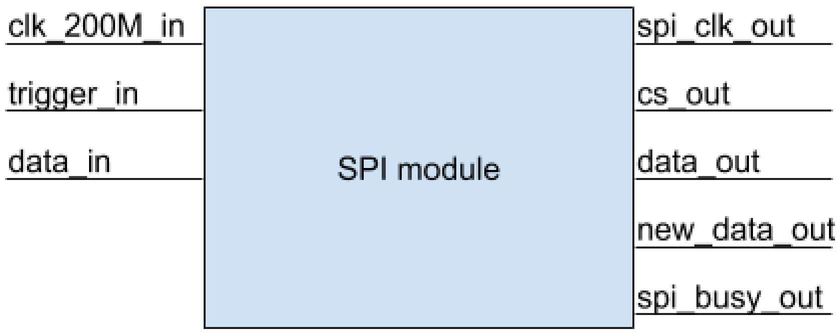
## Exercise 1: SPI ADC sensor

In this exercise communication with the ADC should be established. Use the points as guidelines and if you feel better about another structure for your code then feel free to do so. The only limitations are that you have to instantiate your code with text in the pl_top module and you have to write the code as FSM. Use the examples from the reading material to help you structure your code.

- Mount the ADC close to the propeller board on the breadboard.



- Make connections to three GPIO pins on the SoC and use 3.3 V from the propeller board to supply the ADC. Make sure you are using short wires.
- Find the suggested size of decoupling capacitors in the datasheet on connect them in the breadboard.
- Connect a trim potentiometer to 3.3 V and ground and connect the last pin to Vin on the ADC.
- Create a new module from where you instantiate every modules dealing with collecting and filtering signal from the ADC. Instantiate the new module it in the pl_top module.
- In the module create a timer that gives a high output in one clock cycle when a new sample should be collected. Make sure you make it easy to change the frequency for getting new samples or even better make it changeable from unity.
- Create a module that contains a FSM that handles getting a new sample from the ADC. Suggestions on in and outputs is shown below. Use the datasheet as reference to bit rate and timing of your state machine.



- Port the output to a register in Unity.
- Remember to test your module in the simulator, this will safe you a lot of time.
- After implementing on the SoC, check with an oscilloscope the frequency is as expected and that the signal have an okay (breadboard) quality.
- Trim your parameters and see how fast a sample rate you can get from the ADC.
- (Option) Unity have a publish service functionality that can be used to print data registers autonomous. Setup publish service to print the data register containing the ADC data.

## Exercise 2: Filter

To make sure the signal

- Create a low pass filter of your choice. Preferable a first order IIR but and average filter is also a possibility.

## Bonus exercise: Control a motor with a potentiometer

Connect the 8-bit ADC value to a PWM generator and use the signal to control a DC motor.