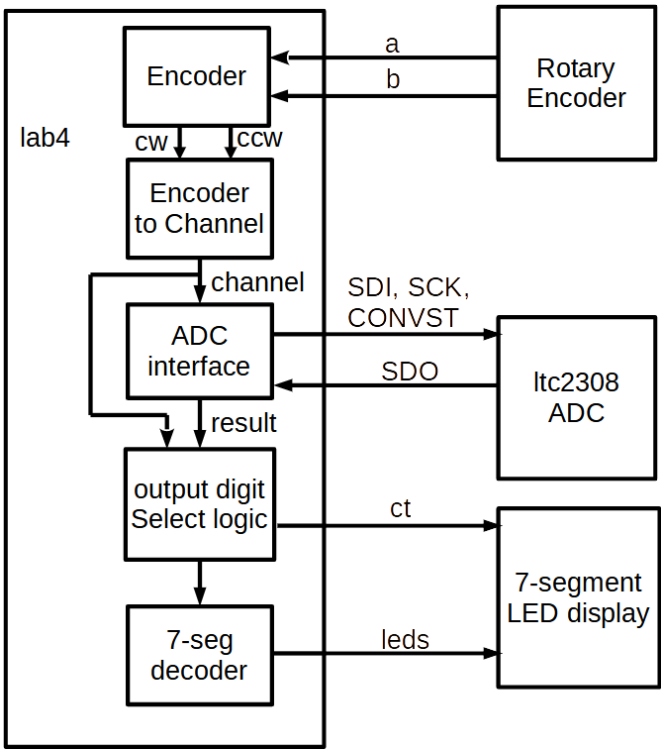
# Lab 4 – ADC Interface

## 1 Objectives

1. Implement a moderately complex finite state machine.
2. Interface to the ltc2308 Analog-to-Digital Convertor (ADC) device.

## 2 Introduction

In this lab you will design and implement a circuit to interface to the ltc2308 ADC on the DE0-Nano-SoC board. This lab will build on lab 1 and lab 2 using the encoders to select the ADC input channel to be sampled, and display the ADC conversion result on the 4-digit 7-segment LED display.




## 3 Prelab exercises

Study the "2308fb" datasheet for the ADC. Pay particular attention to the information on page 10 and the timing diagrams on page 17.

You must design and implement the ADC interface module to interface to the ltc2308 ADC using the following port declaration:

```
module adcinterface(
    input logic clk, reset_n,   // clock and reset
    input logic [2:0] chan,     // ADC channel to sample
    output logic [11:0] result, // ADC result

    // ltc2308 signals
    output logic ADC_CONVST, ADC_SCK, ADC_SDI,
    input logic ADC_SDO

);
```

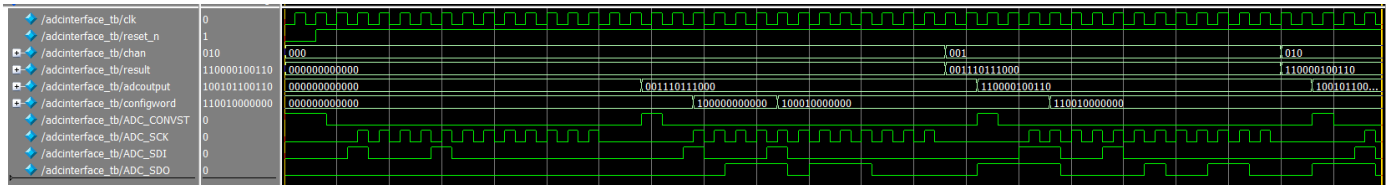
The interface you design will continually sample the ADC channel indicated by the module input port `chan`, and output the result on the output port `result`. In order to accomplish this, the module should output a pulse for one clock cycle on the `ADC_CONVST` pin, followed by one clock cycle with the `ADC_CONVST` pin low. Subsequently, the `ADC_SCK` pin will toggle for 12 cycles in order to send the configuration word and capture the result data. Following the 12 cycle data transfer, there must be at least one clock cycle without `ADC_SCK` toggling before starting the next conversion by pulsing `ADC_CONVST` again.

Your module should update the `ADC_SDI` output on the falling edge of the `ADC_SCK` clock to ensure that the signal is stable on the clock rising edge. Similarly, you can expect that the `ADC_SDO` input is stable on the rising edge of the `ADC_SCK` clock and should thus be sampled on the rising edge of the clock.

In order to "gate" the clock output to the ADC_SCK signal, you may want to use a statement similar to:

```
assign ADC_SCK = (state == ????) ? clk : 1'b0;
```

Simulate your ADC interface design using the testbench provided. Below is a sample waveform showing what your output might look like.




Once all tests pass, save a copy of the simulation results portion of the Transcript window and take a snapshot of your waveform to submit with the lab.

# 4 Lab Activities

Create an **`enc2chan`** module similar to your **`enc2chan`** module from lab 3 that uses the **`encoder`** module output to select the three bit input channel (0-7) to the ADC.

Create a **`lab4`** top module that connects the **`encoder`**, **`enc2chan`**, and **`adcinterface`** modules such that the ADC will sample the channel selected by the encoder. Display the number of the sampled channel (0-7) on the leftmost digit of the 4-digit display, and display the ADC sample result in hex on the three rightmost digits.

Synthesise your **`lab4.sv`** file along with your code. Plug in the FPGA board and program it. You may need to press the reset button before your code will work. Test your design by moving around the joystick and observing the output on the display. The joystick X-axis and Y-axis are connected to the channel 0 and channel 1 of the ADC respectively. The accelerometer X, Y, and Z are connected to channels 2, 3, and 4 of the ADC respectively. Note that the joystick output voltage range is between 0V and 3.3V while the ADC reference voltages are 0V and 4.096V.