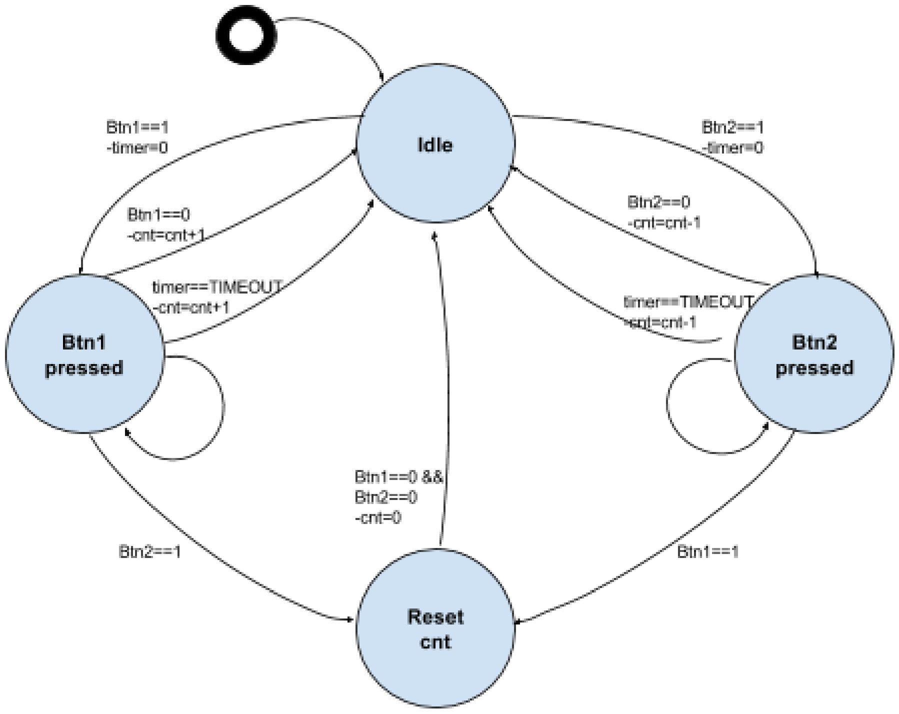
Lab 3 exercise

BLDC motor with sensor feedback

In this lab exercise, we will be working towards building a three-phase BLDC motor driver. In the first part, we will start with a normal brushed DC motor to make sure the PWM signal is controllable and we get to see that the code is working before adding extra complexity. On top of the brushed DC motor driver, we will ad the phase shifting pattern that is required to run a the phased BLDC. The code before the shifting pattern is the same whether if it is a brushed or brushless motor. The last part is an SPI protocol because we would like to receive feedback that could be used in a high-level controller. In this assignment, we will only be sending the motor position, but it could might as well be velocity and torque which a high-level controller often use to control the PWM duty cycle.

Brushed DC motor

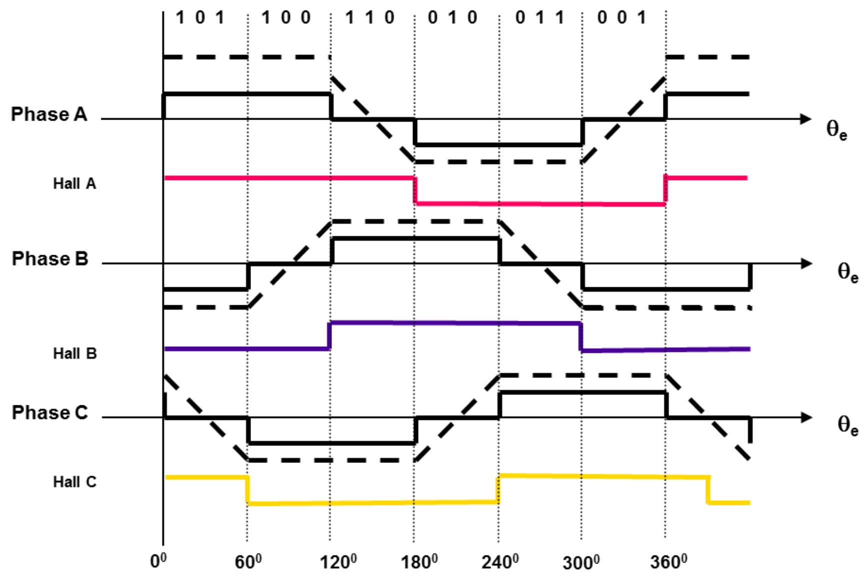
1. Create a debouncing module that reads the input signals from the two buttons on the red extension board to the FPGA. It is okay to re-use modules from earlier lectures.
2. Make a state machine that takes the two filtered button inputs and counts up or down an 8-bit vector, dependent of which button is pressed. If both buttons are pressed then the counter vector should be set to 0. You can use the state diagram below as a guideline but remember to handle over/underflow conditions.



3. Create an 8-bit PWM generator module that takes a 200 MHz clock, an 8-bit vector for duty cycle and a reset as input signals and returns the PWM out signal. The PWM frequency should be higher than 20 kHz but as low as possible using only bit-shift for the PWM counter.
4. Connect the counter vector from 2 to the PWM duty input and create a testbench that shows the PWM out signal at different values of the counter.
5. If everything is working in the simulator, then this should be enough code to control a brushed DC motor.
    a. Look in the BTN8962TA datasheet and verify how the half-bridge works and what to do with the input signals to get high, low or float on the output. Find the FPGA ports that are connected to the inputs on the half bridges and edit your code to give ground on the output of one-half bridge and PWM/float on one of the other bridges.
    b. Test your code on a brushed DC motor in the lab. Add 10 V on the bullet connectors on the extension board. When you have verified that you can control the speed of the motor by changing the PWM duty and there is no noise caused by too low frequency, you are ready to start working with the BLDC.

BLDC motor

1. Create a module that takes the BLDC sensors and the PWM signal as input and returns the control signal to the three half bridges power switches. The VHDL module should be coded directly in combinational logic, it is therefore not necessary to add the CLK as an input signal to the module.



2. Create a testbench that manipulates the sensor inputs. Check if the output is as expected by running simulation.
3. Connect the VHDL modules in Vivado and test the circuit on the FPGA with a BLDC motor. Set the power supply to 10 V and add 3.3 V to the red and black encoder wires. There is a 3.3 V pin available on the pin-header on the extension board.
4. Measure the with an osilloscope the voltage of the spikes and the frequency on the hall sensor signals. Add a lowpass filter on the signal before connecting them to the FPGA.
5. start with a low PWM duty, increase if current consumption and motor sound are what you would expect.
6. Consider if it is necessary to add debouncing/filter on the sensor input from the motor. Use an oscilloscope to see if the signal has flaws and add the code to fix it if necessary.
7. Use the oscilloscope to measure how many electrical revolutions per sec and compare the result with mechanical revolutions.

SPI position feedback

1. Add the position counter from Lab 2 to the project and connect the sensor signals to the module.
2. Create an SPI master module that has a bit rate on 10 MHz. The data is the position counter value and it should be transmitted with 100 Hz. It is up to the team to decide the frame size, but if you chose smaller than the data width then send the LSBs of the position counter. The MISO signal is not necessary since we are only sending data one way.

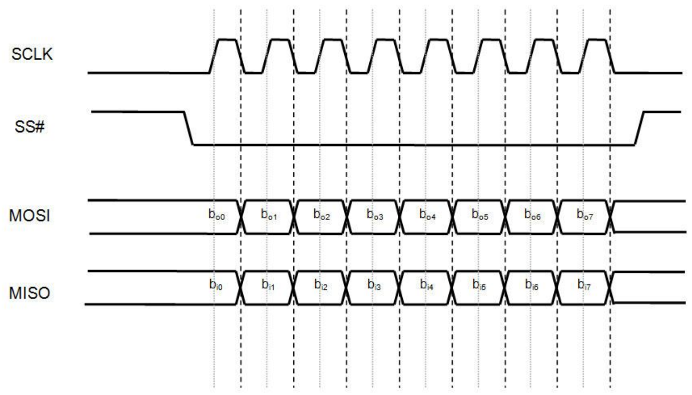


Figure 2 : A simple SPI communication. Data bits on MOSI and MISO toggle on the SCLK falling edge and are sampled on the SCLK rising edge. The SPI mode defines which SCLK edge is used for toggling data and which SCLK edge is used for sampling data.

3. Read the SPI messages with an oscilloscope. The oscilloscopes from Drone lab can read and translate the SPI directly.