# EGRE 365 Digital Systems
## Final Design Project – SPI 3-axis Accelerometer Interface

**The technical portion of this project may be completed individually or in groups of two. Each individual must turn in their own design report. Be sure to list your partner in the project report if applicable.**

For this project you will implement an FPGA design for the Digilent Nexys4 DDR board that interfaces an external 3-axis accelerometer to the FPGA. The accelerometer is an Analog Devices ADXL345 which has three independent accelerometers in the X-, Y-, and Z-axes. The ADXL345 communicates via a Serial Peripheral Interface (SPI) bus and is implemented on a Digilent Pmod board (PmodACL).

The SPI communications are based on an IP core developed and distributed by Surf VHDL as freeware. The SPI core is configured in SPI mode 3 (CPOL=1 and CPHA=1) with a 4-wire SPI communications bus and has a simple parallel interface on the input/output side.

The ADXL345 will be configured to perform continuous acceleration measurements at a 25Hz rate. Acceleration data in the X-, Y-, and Z-axis directions can then be requested at whatever rate the user specifies. The user-designed hardware in the FPGA should request a complete set of 3-axis acceleration data 5 times each second (i.e., at 5 Hz) and display the resulting 10-bit digital acceleration measurement via the methods described below.

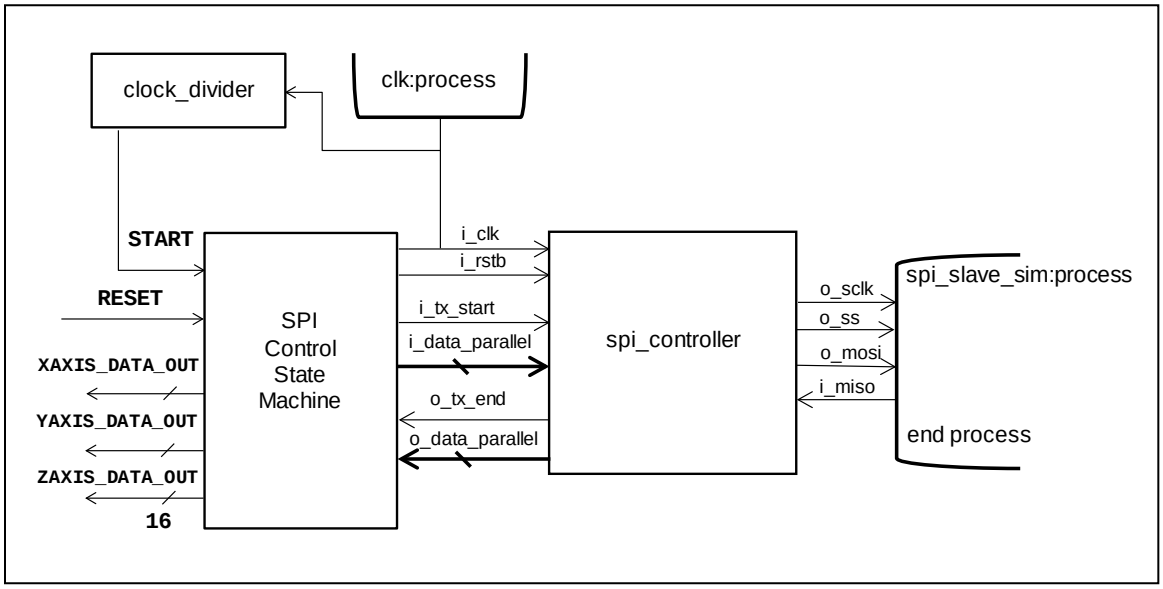
The ADXL345 accelerometer project should be implemented in four phases. Upon completion, each phase must be demonstrated by submitting a demonstration video by the phase due date. There is a 2% penalty per phase, for not completing and documenting each individual phase by the required due date and a 5% penalty per phase, for not completing and documenting each individual phase before the final project date.

Phase 1
In this phase, you will design and simulate a state machine that controls the SPI interface as necessary to perform two configuration write operations followed by six data read operations as shown in the lecture slides. A simulation of this sequence of operations was provided as part of lab 9. In addition to a clock and reset input, the state machine should have a "START" input which, when it transitions from '0' to '1', causes a new communications sequence to the ADXL345 to be started. After that sequence is completed, the state machine should wait for the next '0' to '1' transition on the START signal to start the next communications sequence. The most straight-forward method to implement this is to use a clock divider to divide the 100 MHz system clock down to a 5 Hz clock to serve as the START signal. The state machine should itself operate on the 100 MHz system clock.

When the first read cycle of the Least Significant Byte (LSB) of the X-axis data (DATAX0) is complete, the state machine should capture the LSB into the lower 8 bits of a 16-bit data output signal. When the second read cycle of the MSB of the X-axis data (DATAX1) is

complete, the state machine should capture the MSB into the upper 8 bits of the 16-bit output signal. A similar process should be completed for the Y-axis and Z-axis data

A schematic of the full simulation system for the state machine is shown below. Note that there may be additional signals or components that you may need to add to the system, as you design it, to make it operate correctly.




The simulation should show at least two complete SPI bus operation sequences controlled by your state machine and initiated by two separate ‘0’ to ‘1’ transitions of the START signal.

For this phase, you must demonstrate a working simulation with your state machine controlling the SPI interface in a test bench with a structure as shown above. The master_stimulus process, included in the testbench that was provided for lab 8, is not needed in the test bench as the state machine you design will perform that function. However, the slave_stimulus process that simulates the response of the ADXL345 on the SPI bus, will be needed in this simulation.

Phase 2

In this phase, you will implement your system on the Nexys4 board such that it reads from the ADXL345 5 times a second and the 16-bit data read back from the X-Axis of the ADXL345 is displayed on the LEDs.

To implement the full system into the FPGA, a top-level structural VHDL description with the proper input and output ports must be constructed. This top-level description must then be mapped to the proper signals on the Nexys4 board with an .XDC file. A skeleton of the top-level VHDL entity and architecture and a corresponding .XDC file have been provided

on the class Blackboard site. The entity description of this top-level VHDL description is shown below.

```
entity spi_fsm_toplevel is
    Port ( CPU_RESETN : in  STD_LOGIC;                      -- Nexys 4 DDR active low reset button
           SYS_CLK    : in  STD_LOGIC;                      -- Nexys 4 DDR 100 MHz clock
           LED        : out STD_LOGIC_VECTOR(15 downto 0);  -- Nexys 4 DDR LEDs
           SW         : in  STD_LOGIC_VECTOR(15 downto 0);  -- Nexys 4 DDR switches
           SCK        : out STD_LOGIC;                      -- SCK to SPI slave
           CS         : out STD_LOGIC;                      -- SPI slave chip select
           MOSI       : out STD_LOGIC;                      -- MOSI out to slave
           MISO       : in  STD_LOGIC);                     -- MOSI in from slave
end spi_fsm_toplevel;
```

The CPU_RESETN port is the active low reset signal and the SYS_CLK port is the 100 MHz clock signal. The LED port is the 16-bit output tied to the LEDs on the board and the SW port is the 16 switches on the board. The SCK, CS, MOSI, and MISO ports are the four SPI signals that are tied to the corresponding pins on the PmodACL module.

In addition to the SPI IP core and SPI control state machine, a clock divider, or some other component, will be needed to generate the 5 Hz START signal from the 100 MHz system clock.

Phase 3
In this phase you will display the values read back from the X-, Y-, and Z-axis on the LEDs. You will use the switches SW(0) and SW(1) to select which data is displayed on the LEDs.

Phase 4
In this phase you will use the data from the ADXL345 for two of the axes, the X-axis and the Z-axis, to implement a “tilt meter” functionality on the board. The tile angle of the board, along the X-axis, must be displayed on the LEDs according to the following specification.

LED(7) will be considered as the middle of a 15-LED tilt-meter (LED(0) – LED(14)). LED(7) will always be illuminated as a reference. The other LEDs will be illuminated, one-at-a-time, as the tilt angle is increased from 0 degrees to 70 degrees. The LEDs on the right side of LED(7) (i.e., LED(6)-LED(0)) will indicate a tilt in the positive X-axis direction and the LEDs to the left side of LED(7) (i.e., LED(8)-LED(14)) will indicate a tilt in the negative X-axis direction.

For example, if the board is tilted between 0 and less than 10 degrees, in the positive X-axis direction, only LED(7) will be illuminated. If the board is tilted between 10 degrees and less than 20 degrees, only LED(7) (the reference), and LED(6) will be illuminated. If the board is tilted between 20 degrees and less than 30 degrees, only LED(7) and LED(5) will be illuminated. This will continue until the board is tilted by 70 degrees or more in the positive X-axis direction at which point LED(7) and LED(0) will be illuminated. Tilting the board in the negative X-axis direction will result in LED(7) and LED(8)-LED(14) being illuminated in a similar manner.

In addition to the above 4 required phases, unique and innovative functionality of your own design to the system is welcome. Up to 15% bonus credit may be earned based on the originality and complexity of your additional functionality and how it performs. This additional functionality MUST be demonstrated and presented during in Phase 4 demonstration video. This additional functionality must also be described and documented in the final report.

At the end of the project, you and your project partner must turn in a complete project report. This project report must include:

1) A writeup describing the architecture of the overall system and how it implements each function. The description must include a state transition diagram or state table for each state machine used in the design.
2) Simulation results which clearly show **each phase** working for the required cases. The intension is that the simulations will be completed before the design is mapped into the FPGA for that phase (test benches are highly recommended).
3) All VHDL code for the system and test benches.
4) The design summary portion of the implementation report showing the amount of FPGA resources used by your design.
5) A set of demonstration videos showing proper operation for each phase. Note that if you have already submitted a demonstration video for each phase separately, you do not have to turn in duplicates for the final report.
6) Comment on the project (optional).

Grades on the final project will be apportioned based on the following rubric:

| | |
|---|---|
| System functionality (base) | 50% |
| Maximum penalty for no phases | -20% |
| Maximum bonus for additional functionality | +15% |
| Report | |
| Functional description | 20% |
| Simulations (thoroughness, annotation) | 15% |
| VHDL Code quality (organization, correct construction, comments, etc.) | 15% |

**Due dates:**

| | |
|---|---|
| Phase 1 | Friday, November 19th, 2021 |
| Phase 2 | Thursday, December 2nd, 2021 |
| Phase 3 | Monday, December 6th, 2021 |
| Phase 4 | Monday, December 13th, 2021 (last day of classes – before 5:00PM) |

Final Report Due Date: Tuesday, December 14th, 2021

**No final reports will be accepted after 5:00PM Tuesday, December 14th, 2021 – NO EXCEPTIONS!**