

Lab 5 Assignment

ECE 525.442 FPGA Design using VHDL

VGA Checkerboard Display with SPI Accelerometer

This lab assignment requires you to modify Lab 4 in order to use the on-board SPI accelerometer to move the red square. The goal of this lab is to understand how to interface the FPGA to an SPI device using the SPI device's datasheet. The FPGA will interface to the following on-board SPI device:

- Analog Device ADXL362 Accelerometer with an SPI interface (I recommend using a 1MHz SCLK frequency)

Please look at the datasheets and read the Nexy's 4 Reference Manual section 14 on the accelerometer to understand the interface. To get started, read the top level IO modifications listed below and follow the suggestions for how the SPI interface can be implemented.

VGA Lab I/O Modifications

In addition to the Lab 4 requirements, the following requirements will need to be implemented for Lab 5:

- The ADXL362 Accelerometer should be activated coming out of reset and power-up. The 0x00, 0x01, 0x08, 0x09, and 0x0A registers should continuously be sampled and stored in registers.
 - Register 0x00 contains the fixed value 0xAD
 - Register 0x01 contains the fixed value 0x1D
 - Register 0x08, 0x09, and 0x0A contains the X, Y, and Z axis data respectively
- The switch SW(1) shall be used to switch from button control (when SW(1) = '0') to accelerometer control (when SW(1) = '1'). Accelerometer control means the up movement should be a forward tilt of the board. The down movement should be a backwards tilt of the board. The right movement should be a right tilt of the board, A left movement should be a left tilt of the board. Please make sure tilting the board only causes 1 movement of the red square. Make sure the board is repositioned back to center prior to capturing another move.
- The switch SW(4:3) will be used as follows
 - For any value of SW(4:3) displays 3 through 0 should ALWAYS have the X/Y position of the red square
 - SW(4:3) of '00' shall show the values of register 0x01 in 7-segment displays 7 and 6 and the values of register 0x00 in displays 5 and 4
 - SW(4:3) of '01' shall show the value of register 0x08 in display 5 and 4 and should have all zeros on display 6 and 7
 - SW(4:3) of '10' shall show the value of register 0x09 in display 5 and 4 and should have all zeros on display 6 and 7
 - SW(4:3) of '11' shall show the value of register 0x0A in display 5 and 4 and should have all zeros on display 6 and 7



These modifications require adding a VHDL component that interfaces to the SPI device, which is explained in the next section. In addition to the SPI VHDL component, there needs to be another set of registers that will store the values of registers 0x00, 0x01, 0x08, 0x09, and 0x0A. The final modification would be to add a comparator to check the ranges on the X and Y registers from the accelerometer and once they reach a threshold, moves the red square. The threshold is a value that needs to be determined by you based on initial testing.

SPI Interface Component

The SPI interface should be written as an entity/architecture pair that will interface to the accelerometer to write to its internal registers. The VHDL SPI interface should also read data values back from the accelerometer and store it in registers internal to the FPGA. This interface is called *accel_spi_rw* but the component and signal names for this interface is left to you.

Out of power-up, the SPI interface should initialize the accelerometer to enable the device to generate the X, Y, and Z data. To do this, the SPI interface should **write** the value 0x02 to address 0x2D. Check with the ADXL362 datasheet to see how this value and address was determined or watch the lab 7 lecture video for details on the SPI write. Once this initial write is performed, the SPI VHDL component will only be performing SPI **reads** to capture the X, Y and Z data values as well as registers 0x00 and 0x01 for the fixed analog device response of 0xAD and 0x1D respectively.

Your SPI interface component should have the following minimal signals:

```
entity accel_spi_rw is
  Port ( clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        --Values from accelerometer used for movement and display
        DATA_X : out STD_LOGIC_VECTOR(7 downto 0);
        DATA_Y : out STD_LOGIC_VECTOR(7 downto 0);
        DATA_Z : out STD_LOGIC_VECTOR(7 downto 0);
        ID_AD : out STD_LOGIC_VECTOR(7 downto 0);
        ID_1D : out STD_LOGIC_VECTOR(7 downto 0);
        --SPI Signals between FPGA and accelerometer
        CSb : out STD_LOGIC;
        MOSI : out STD_LOGIC;
        SCLK : out STD_LOGIC;
        MISO : in STD_LOGIC);
end accel_spi_rw;
```

The signals are defined as follows:

- clk: Master 100MHz clock
- reset: Active-high reset
- DATA_X: 8-bit value from register address 0x08 of accelerometer used to determine up/down movement of red square
- DATA_Y: 8-bit value from register address 0x09 of accelerometer used to determine left/right movement of red square
- DATA_Z: 8-bit value from register address 0x0A of accelerometer not used for red square movement



- ID_AD: 8-bit value, the result of reading from register address 0x00 of accelerometer. This value should always read 0xAD (to know it's an Analog Devices device)
- ID_1D: 8-bit value, the result of reading from register address 0x01 of accelerometer. This value should always read 0x1D
- CSb: SPI interface's Chip Select. See datasheet for proper timing. The FPGA drives this signal to the accelerometer
- MOSI: Master Output, Slave Input SPI signal. Driven by FPGA on falling edges of SCLK and defined by datasheet
- SCLK: SPI serial Clock, 1MHz frequency, 50% duty cycle. The FPGA drives this signal to the accelerometer.
- MISO: Master Input, Slave Output SPI Signal. Used by FPGA to read data from SPI device on rising edges of SCLK and defined by datasheet

First Week Task

The first part of this lab assignment will focus on understanding the ADXL362 datasheet and designing the SPI interface described as the component *accel_spi_rw* above. The Lab 5 hints provides a good starting point to design the SPI interface that you can use in your Lab 5 design. A testbench and very simple ADXL362 model is provided for you to simulate and debug your *accel_spi_rw* component. If you designed the *accel_spi_rw* component correctly, you should be able to correctly read two ID values, namely 0xAD and 0x1D respectively, and you should read the X, Y, and Z value as 0x12, 0x34, and 0x56 respectively. The X, Y, and Z values will not change for this simulation model. This is not required though.

If you are having issues with the testbench, you can reduce the initial test to try to only read values 0x00 and 0x01 which do not require the accelerometer write at power-up or initialization. If you can get this working then work on adding back in the SPI write and X/Y/Z reads.

Second Week Task

During the second week of the lab assignment and after you confirmed your SPI interface works correctly using the simulation models, you should then proceed to build the lab5_top component and instantiate your *accel_spi_rw* component that you designed in the first week. Use the VGA controller from Lab 4 and the X, Y data values from the *accel_spi_rw* component to move the red square. Set the thresholds appropriately to control the sensitivity of tilting the board to generate a movement of the red square. Remember to display the two ID values, as well as the X, Y and Z values from the accelerometer on the seven segment display. Also, you should display the X/Y coordinates of the red square on the seven segment display.

Files to Submit (minimum), zipped together as **Lab5_<last_name>.zip**:

- Complete set of VHD, XDC, BIT files for Lab 5
- **Synthesis VDS file (**This is new from previous submissions**)**
 - Located in <project_dir>\<prj_name>.runs\synth_1\<top_level_filename>.vds
 - Contains FSM breakdown from Synthesis Tools



- Complete Synthesis Results
- Report to include the following:
 - Summary for estimate of resource utilization for Lab 5 and how does that compare to the actual resource utilization reported by the tools
 - All Finite State Machine diagrams used to generate the SPI interface (hand drawn or block diagram tools such as Visio submitted as a PDF or image in the report)
 - Block Diagrams of your entire Lab 5 design

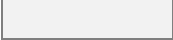








Note: The Grading sheet on the next page provides a good overview of the key points of the design and what the instructor will look for when grading. Please note that this is a guide and isn't comprehensive. Please read over the lab assignment carefully to verify all the proper documents are submitted.



Grading Sheet

Lab 5/Module 7: VGA Controller with Accelerometer

Student:

Grade	Out of	Notes
	100	
	10	DEMO: Correct Accelerometer Data is displayed on 7-segment displays at all times
	20	DEMO: Tilting board moves red square one block at a time in the correct direction when active
	10	DEMO: Switch1 enables and disables accelerometer function. Pushbuttons should still work when Switch1 is low.
	15	DEMO: Switch 4,3 determine which accelerometer data to display on upper four 7-segment displays (independent of Switch 1) --Switch 4,3 value of 00 shows register 0x01/0x00 --Switch 4,3 value of 01 shows register 0x08 --Switch 4,3 value of 10 shows register 0x09 --Switch 4,3 value of 11 shows register 0x0A
	5	X/Y coordinates of red square position displayed correctly
	10	SPI State Machine Diagram(s)
	10	Quality of VHDL code and state machine coding style
	20	Report that includes detailed analysis of resource utilization and comparison to estimated resources, FSM diagrams, and block diagrams

