

# Laboratory 3

(Due date: **005**: March 9<sup>th</sup>, **006**: March 10<sup>th</sup>)

## OBJECTIVES

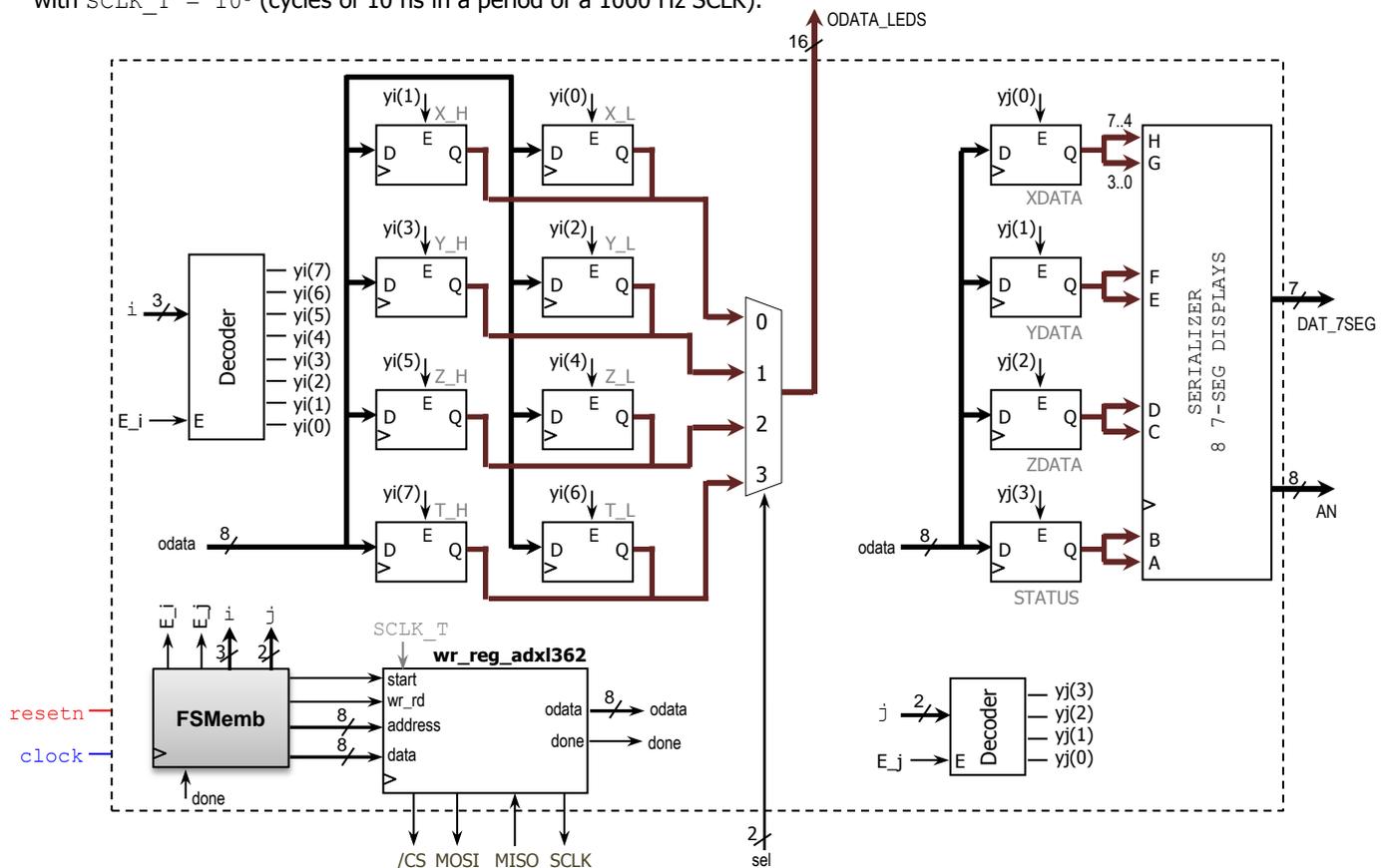
- ✓ Implement a Digital System: Control Unit and Datapath Unit
- ✓ Describe Algorithmic State Machine (ASM) charts in VHDL.
- ✓ Learn interfacing with SPI devices.

## VHDL CODING

- ✓ Refer to the [Tutorial: VHDL for FPGAs](#) for parametric code for: Registers, busmux.

## FIRST ACTIVITY: DESIGN OF AN ACCELEROMETER DATA RETRIEVER (100/100)

- **ACCELEROMETER ADXL362:** This 3-axis MEMS device communicates via a 4-wire SPI and operates as a SPI slave device. We read/write 8-bit data via a register-based interface. ADXL362 parameters (range, resolution, ODR are selectable):
  - ✓ Range:  $\pm 2g$  (default at reset),  $\pm 4g$ ,  $\pm 8g$ .
  - ✓ Resolution: 1mg/LSB (default at reset), 2 mg/LSB, 4 mg/LSB
  - ✓ Output data rate (ODR): 12.5 – 400 Hz. Default at reset: 100 Hz.
  - ✓ Output resolution: 12 bits. Representation: signed.
- **wr\_reg\_adxl362:** This circuit handles basic SPI communication with the ADXL362. The user provides address, data, and read/write. Then, a read/write transaction is executed via the SPI bus (data structure specified in the ADXL362 datasheet). Use the VHDL code `wr_reg_axl362.vhd` (use all the design .vhd files in [accelerom.zip](#) except `accelerom.vhd`) with  $SCLK\_T = 10^5$  (cycles of 10 ns in a period of a 1000 Hz SCLK).

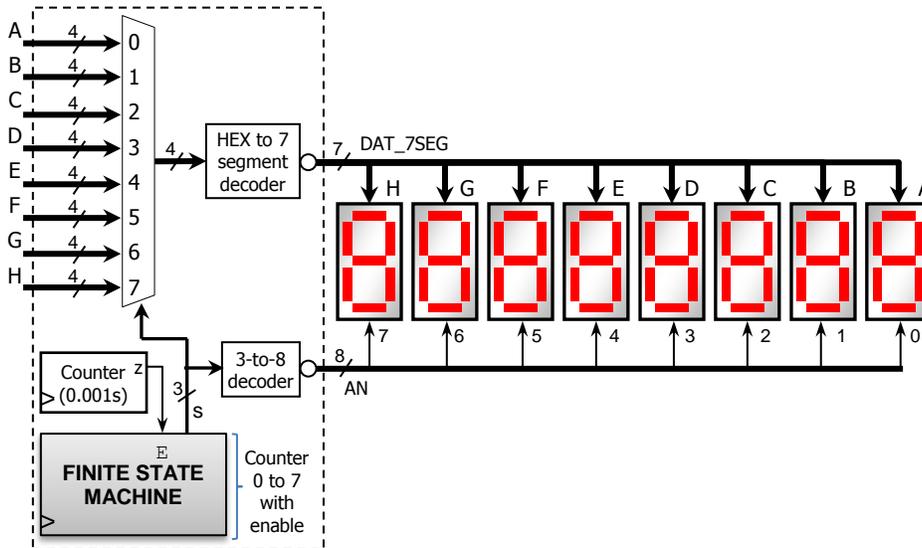


- **FSMemb:** FSM embedding counters  $i$  and  $j$ . It configures 2 ADXL362 registers ( $0 \times 1F$ ,  $0 \times 2D$ ), and then cyclically requests read from 12 8-bit ADXL362 registers containing accelerometer data and places retrieved data on 12 8-bit output registers.
- Data is organized into:
  - ✓ 4 16-bit measurements (X, Y, Z, Temperature). We display this on 16 LEDs (selectable by `sel`). Note that since the actual measurements are only 12-bit wide, the 4 MSBs (of the 16 bits) are sign-extended.
  - ✓ 3 8-bit measurements (low precision X, Y, Z) and 8-bit Status: Shown (as hex) on 8 7-segment displays: `|X|Y|Z|ST|`

- This is the list of registers we deal with in this experiment, which is a basic operation mode. Refer to the ADXL362 datasheet for a full list of registers and operation modes.

Reg. Address	Name	Reg. Address	Name
0x1F	SOFT_RESET	0x0E	XDATA_L
0x2D	POWER_CTL	0x0F	XDATA_H
		0x10	YDATA_L
		0x11	YDATA_H
0x08	XDATA	0x12	ZDATA_L
0x09	YDATA	0x13	ZDATA_H
0x0A	ZDATA	0x14	TEMP_L
0x0B	STATUS	0x15	TEMP_H

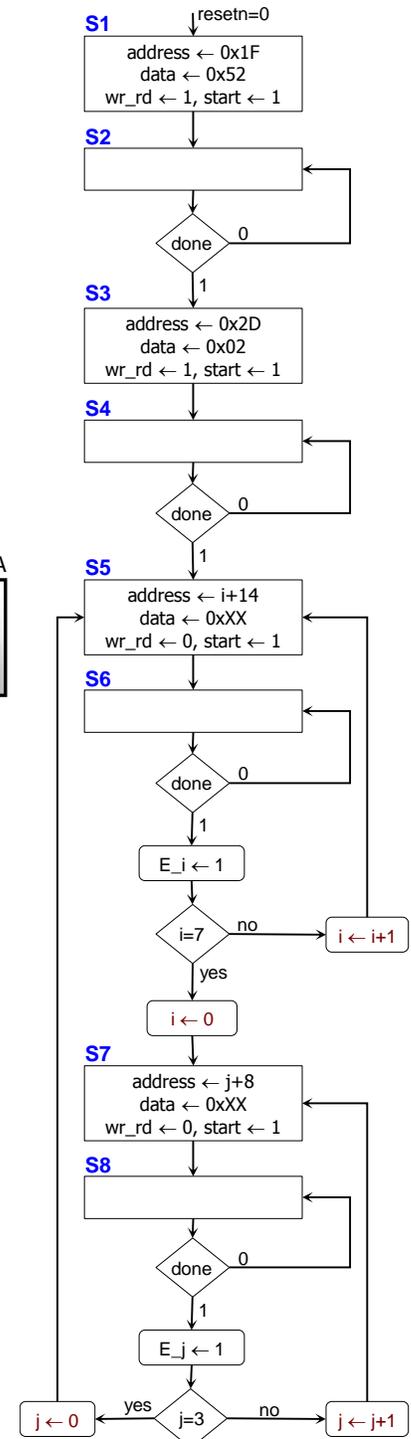
- 8-display Serializer:** Eight 7-segment displays.



**PROCEDURE**

- Vivado: Complete the following steps:**

- ✓ Create a new Vivado Project. Select the corresponding Artix-7 FPGA device (e.g.: the XC7A50T-1CSG324 FPGA device for the Nexys A7-50T board).
- ✓ Write the VHDL code for the given circuit. Run Synthesis to clear your circuit of syntax errors as well as to evaluate the warnings.
  - Use the **Structural description**: Create (or re-use) a separate .vhd file for the components (note that the Serializer has its own components) and interconnect them all in a top file:
    - Register with enable: my\_rege. You need 12 of these units. Use the proper parameters and I/O connections (the input sclr is not used in these registers; however, you still need to assign it a value of '0').
    - Basic AXL362 controller: wr\_reg\_axl362. Use parameter SCLK\_T = 10<sup>5</sup>.
    - 8-display Serializer: It includes its own components (hex-to-7seg decoder, 3-to-8 decoder, counter, BusMUX 8-to-1). You can use the [VHDL code](#) for the 4-display serializer and modify it for 8 displays.
    - Other components: BusMUX 4-to-1, 3-to-8 decoder with enable, 2-to-4 decoder with enable.
- ✓ Write the VHDL testbench (generate a 100 MHz input clock for your simulations).
  - With SCLK\_T = 10<sup>5</sup>, we get a SCLK of 1000 Hz. For SPI, this means that a bit is read/written every 1 ms. This is the expected behavior; however, simulating this behavior will take a very long time in our Vivado simulator window. In addition, we need to emulate data coming from the accelerometer (MISO).
  - Thus, for simplicity's sake, we will do the following:
    - Set the parameter SCLK\_T=16 for the wr\_reg\_axl362 block. For SPI, this means that a bit is read/written every 16 clock cycles.
    - In the testbench, set MISO = 1. This means that our accelerometer controller will only read 1's.
    - Input sel. Set it to "00". This way, only the data X\_H X\_L will be available in ODATA\_LEDS.



- ✓ Perform Behavioral Simulation of your design. **Demonstrate this to your TA.**
  - To help debug your circuit, add the internal signals to the waveform (e.g.: state, i, j, address, data, odata, done, etc.)
  - Run the simulation for as long as needed to observe the main FSMemb issue the 2 writing commands and 12 reading commands. This is, when the FSMemb goes from S1 to S8, and then returns to S5 for the first time.
    - Here, verify that the signals address, data, w\_rd, i, and j behave as expected (see FSMemb ASM diagram).
  - You can observe how the SPI signals (*MOSI*, *MISO*, */CS*, *SCLK*) behave for:
    - 2 writing commands.
    - 8 reading commands + 4 reading commands. These reading commands repeat cyclically (S5 to S8).
  - Right after the FSMemb returns to S5 for the first time, verify that data on *ODATA\_LEDS* and on the input to *DAT\_7SEG* is correct (you should see just 1's). You could modify *sel* in the testbench to select what to display on *ODATA\_LEDS*, but you will just see 1's.
    - *DAT\_7SEG* outputs: It would take a very long time to see changes in the Vivado simulator window (as the transitions in the 8-display serializer occur every 1 ms). Thus, this is better tested in the circuit.

- ✓ I/O Assignment: Create the XDC file associated with your board.

- Suggestion (Nexys A7-50T/A7-100T, Nexys 4/DDR):

Board pin names	CLK100MHZ	CPU_RESET	SW1-SW0	LED15-LED0	CA-CG	AN7-AN0
Signal names in code	<i>clock</i>	<i>resetrn</i>	<i>sel1-sel0</i>	<i>ODATA_LEDS[15..0]</i>	<i>DAT7_SEG[6..0]</i>	<i>AN7-AN0</i>
Board pin names	ACL_MISO	ACL_MOSI	ACL_SCLK	ACL_CSN		
Signal names in code	<i>MISO</i>	<i>MOSI</i>	<i>SCLK</i>	<i>/CS</i>		

- ✓ Generate and download the bitstream on the FPGA. Test the circuit. **Demonstrate this to your TA.**
  - **Note:** Do not forget to set the parameter *SCLK\_T=10<sup>5</sup>* when testing (this is: synthesize, implement, and generate bitstream with this parameter).
  - Low precision data (8-bit *XDATA*, *YDATA*, *ZDATA*) and *STATUS* should appear on the 7-seg displays.
    - Verify that *STATUS* is 0x41.
  - High precision data (16-bit *X*, *Y*, *Z*, *T*): Only one of them appears on the 16 LEDs based on the 2-bit input *sel*.
    - *ODATA\_LEDS[15..0]*: Verify the 4 MSBs are effectively sign-extended bits. This means that we only need the 12 LSBs: *ODATA\_LEDS[11..0]*.
    - Verify that the low precision data (8-bit *XDATA*, *YDATA*, *ZDATA*) on the 7-seg displays match the high precision data (16-bit *X*, *Y*, *Z*) for *ODATA\_LEDS[11..4]*; use *sel* to select among different high precision measurements.
  - To ensure that you are reading correct data from the accelerometer, check the following:
    - At rest, *z* should be about 0xC28 (7-seg display: 0xC2). This corresponds to -984mg. When the board is face down, the sign of *z* should be positive.
  - Feel free to tilt the axes of the FPGA Board to detect changes.
- Submit (as a .zip file) all the generated files: VHDL code files, VHDL testbench and XDC file to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.

TA signature: \_\_\_\_\_

Date: \_\_\_\_\_