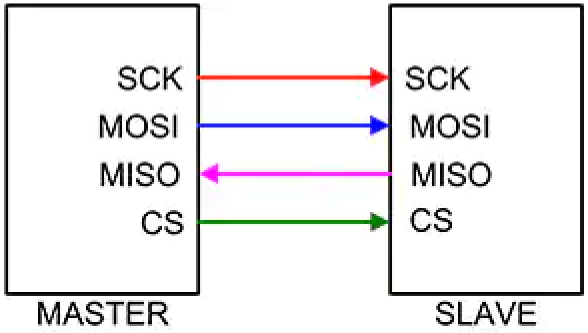
RGB Controller with an ADXL362 Accelerometer

David Sheridan, Jermaine Juarez,
Tajwar Eram

Approach to RGB Control

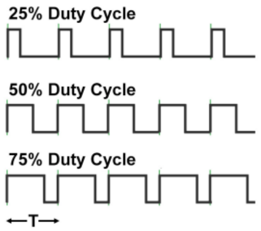
SPI Communications

- Able to communicate between an Artix 7 (Master) and ADXL362 (Slave) through SPI
  - SCLK, nCS, MOSI, MISO



Approach to RGB Control

Accelerometer Data and PWM



- Variation based on change in orientation read from the Accelerometer
- Pulse width modulation to change intensities of RLED and BLED
- The Green LED is not changed by the data from the Accelerometer
- Selector used for reading different registers for axes (X, Y, and Z)

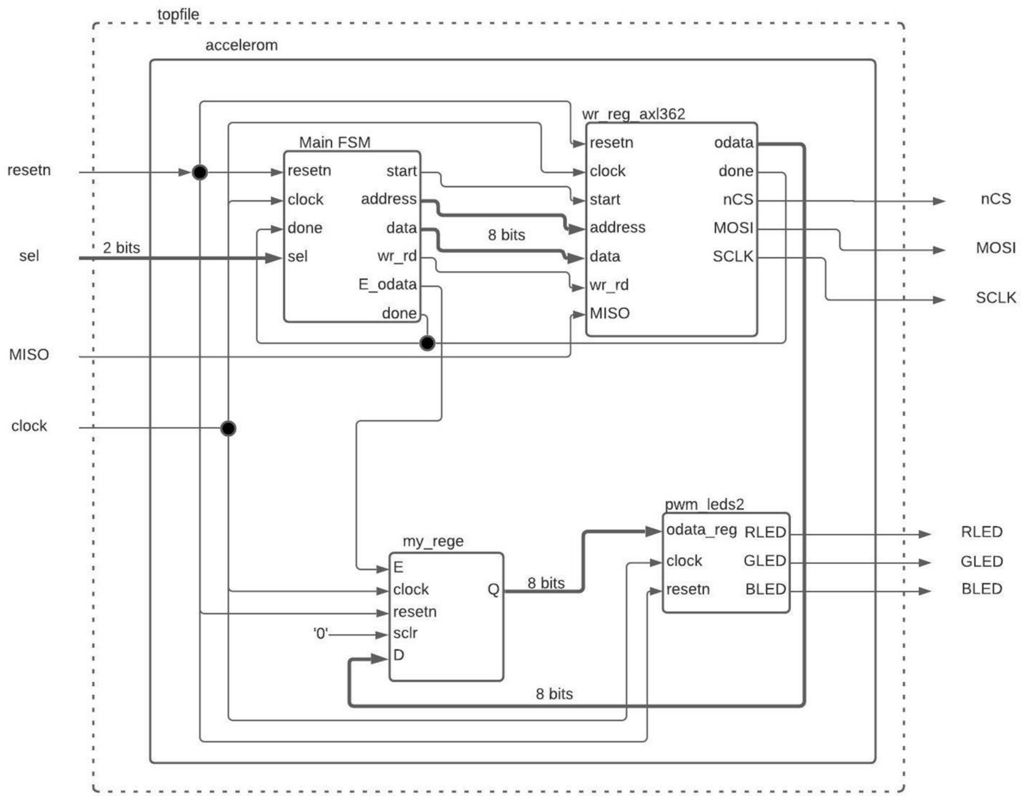
Approach to RGB Control

FSMs and Reference code

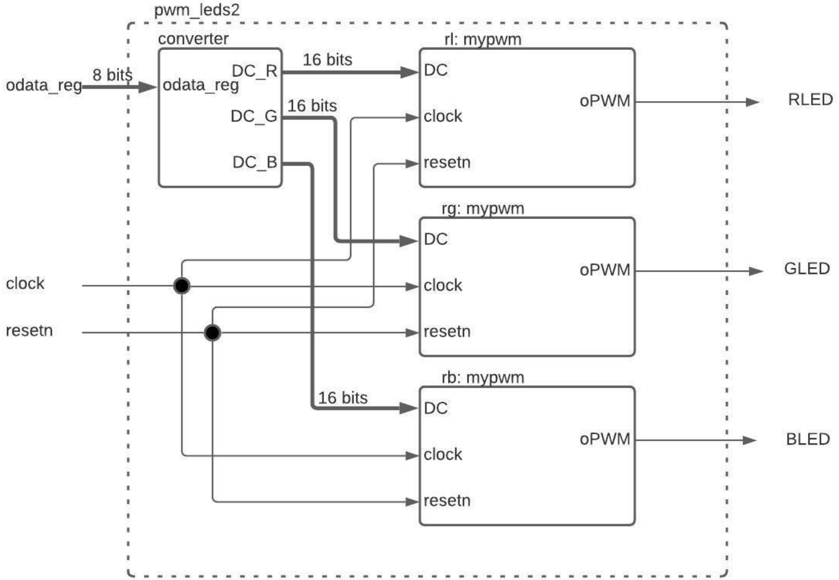
- Through FSMs and referencing the ADXL362 “wr_reg_adxl362” data acquisition code and also allows the right SPI communications to send the measurements from accelerometer to Artix 7

Overall Block Diagram

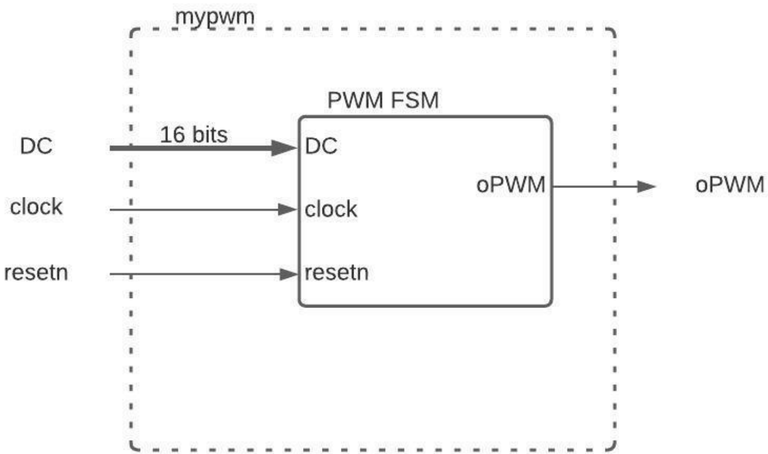
- Input sel selects the axis to be displayed
- Inputs resetn and clock are universal



PWM Block



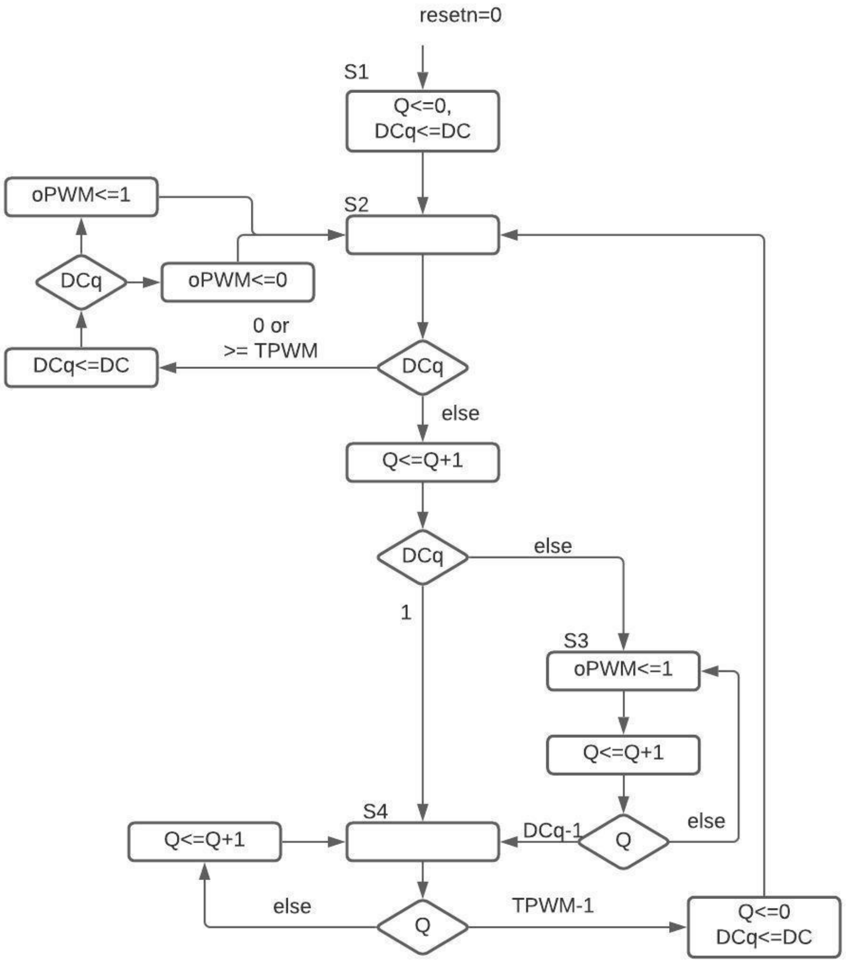
pwm_leds2
converter
odata_reg
8 bits
odata_reg
DC_R
DC_G
DC_B
16 bits
16 bits
16 bits
rl: mypwm
DC
clock
resetn
oPWM
RLED
rg: mypwm
DC
clock
resetn
oPWM
GLED
rb: mypwm
DC
clock
resetn
oPWM
BLED
clock
resetn



mypwm
PWM FSM
DC
16 bits
DC
clock
clock
resetn
resetn
oPWM
oPWM

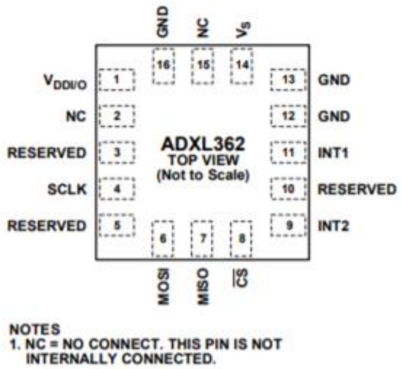
PWM FSM

- Duty Cycle is value between 0 and TPWM
- Can be thought of as a percentage
- Duty Cycle determines amount of time that LEDs are on, resulting in a certain “Brightness”
- State 1
  - Reset
- State 2
  - Ensures DC is within bounds
- State 3
  - Counts until DC value is reached
- State 4
  - Counts until TPWM is reached

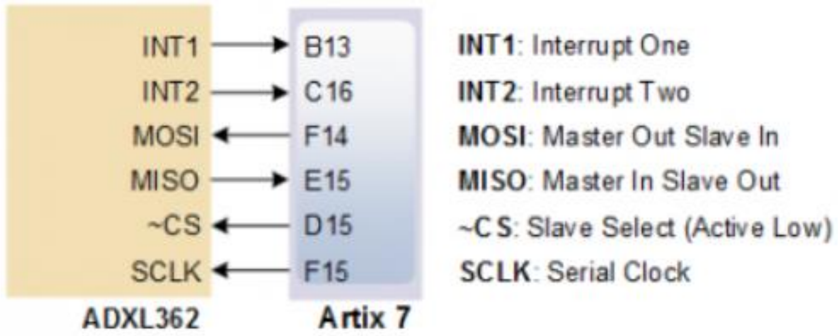


Pins for ADXL362

- SPI Communication
- Interrupts are not used
- MOSI signal comes from board
- MISO signal comes from sensor
- Slave Select is required for communication to be initiated
- Serial Clock comes from the board

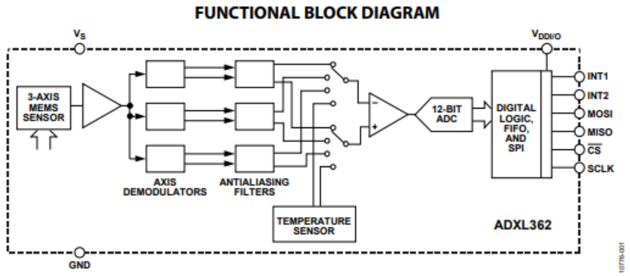


ADXL362 Pins



Pins Between Board and Sensor

Accelerometer



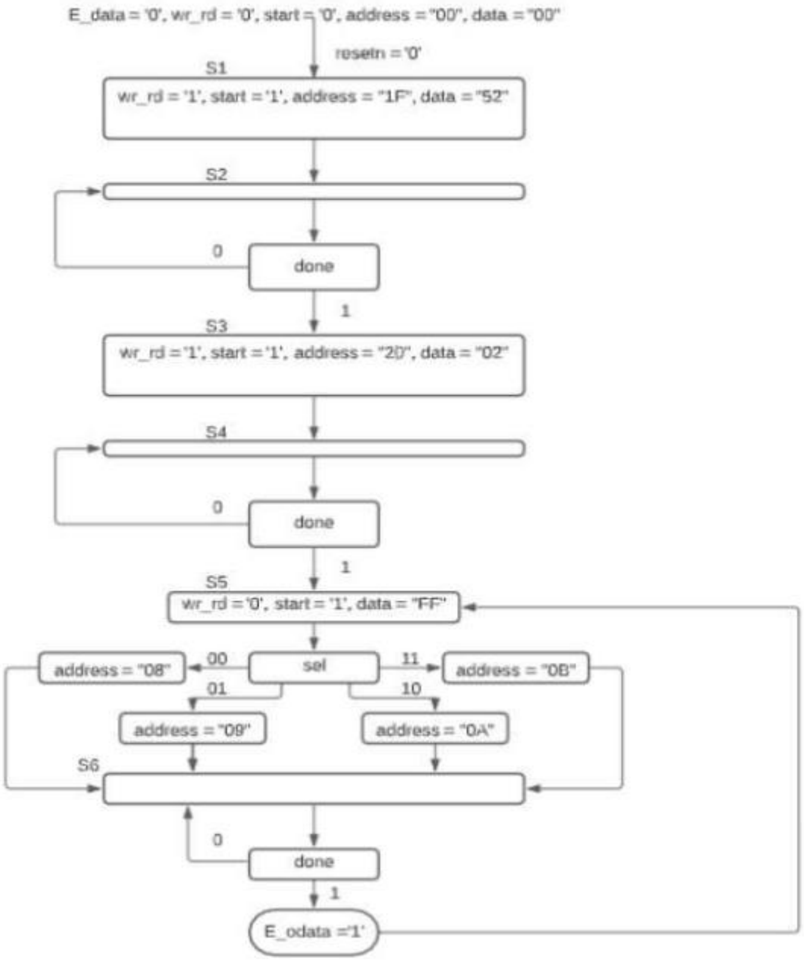
- 4 main components: PWM Block, Accelerometer FSM, wr_reg, and register.

- Measures 3 axis: x-axis, y-axis, and z-axis data.

- Option for the 12 bit register or 8 bit register. 12 bit accuracy is not really needed, so we chose an 8-bit register for effectiveness.

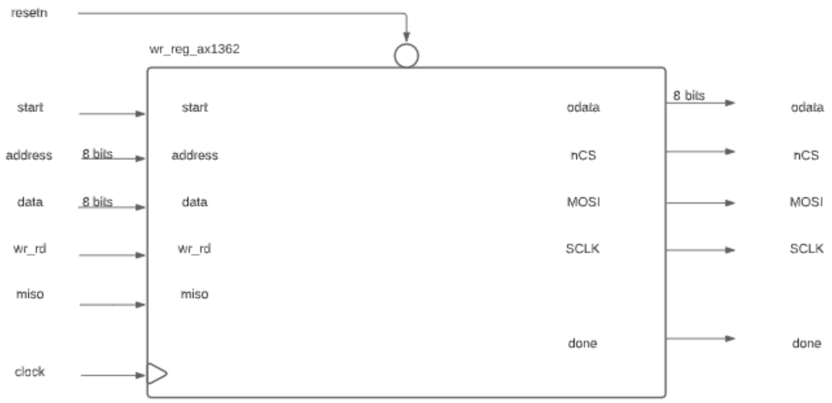
Accelerometer FSM

- States 1-4 prepare for measuring
- At State 3, when the address = “2D” and B1, B0 = “10” then the register is set to measure and moves on to State 4, then State 5 if done = 1
- At State 5, select chooses which address to access X, Y, Z axis 8 bit data or the Status Register then goes to State 6.
- At State 6, it will keep looping until done = 1, then E_odata = 1, then it loops back to State 5. Repeating the process.
- Data clears when resetn = ‘0’

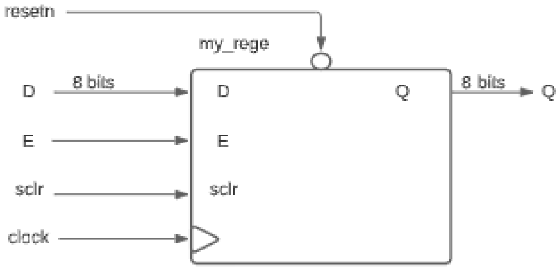


Accelerometer Registers

wr_reg



register



- Wr_reg is made of shift register, primarily, and other components all utilizing a finite state machine to process the data given

- 8-bit register with enable

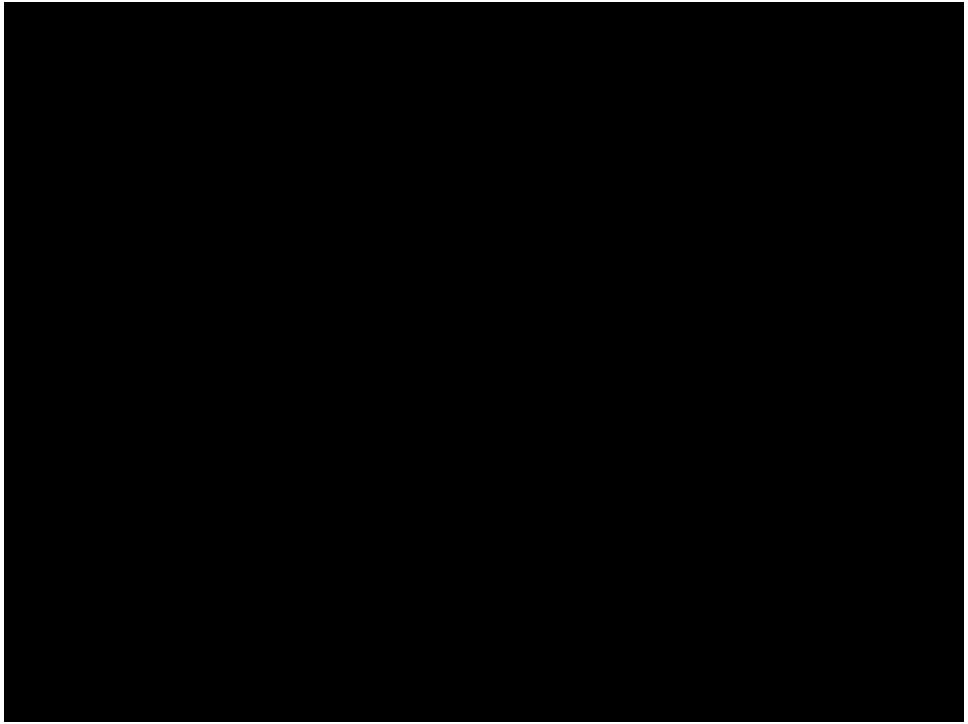
wr_reg_ADXL362

- Modulo-8 Counter
- Shift Registers (To Read the address and data for the correct mode)
    - Address (0x2D) allows writing on the register and reading from X, Y, Z data
    - Data allows the FSM to eventually reach measurement mode (0x02)
- FSM is used mainly to read data and make sure SPI communication is correctly done

Things to Improve

- Smoother Color change
- Limitation with the Green LED not really affected by orientation (Less variety)
- Dedicating more bits to each LED instead of 4 bits (More variety)
- Limitation with low precision X, Y, and Z values

Backup Demo-Video



https://drive.google.com/file/d/18-SOkhEEsZucu41_RXv8OwXxLXT6weCo/view?u