GRAND VALLEY STATE UNIVERSITY
SEYMOUR AND ESTHER PADNOS COLLEGE OF ENGINEERING AND COMPUTING

EGR680 High Level Implementation on FPGA

Final Project

PYNQ Embedded Design using Jupyter Notebooks

Professor: Dr. C. Parikh

Student: Dimitri Häring

December 02, 2018

Contents

List of Figures

Listings

1 Introduction

The goal of laboratory ten is to familiarize the student with the Jupyter Notebook and debugging of hardware in Vivado.

2 Design

In this section the design and decisions that where made to achieve the laboratory are discussed.

2.1 Part I - RGB LED Driver

From project description, create the RGB LED color mixer using the ipywidgets integer or float slider. In theory, you should be able to create an infinite amount of colors with the combination of red, blue, and green LEDs. However, with digital electronics, we are limited to the width of the driving data bus or processing system to create the colors.

- Create individual methods for enabling/disabling RED, GREEN and BLUE color of the RGB LED.
- PWM functionality for color mixing of the RGB LED.
- Create 3 color mixing sliders using ipywidgets. One slider per color (R-G-B).
- Create toggle buttons for all four of the green LEDs using ipywidgets.
- Create a way to set a flashing rate of the green LEDs using ipywidgets.
- Create a neat and organized GUI for all of the LED functionality.

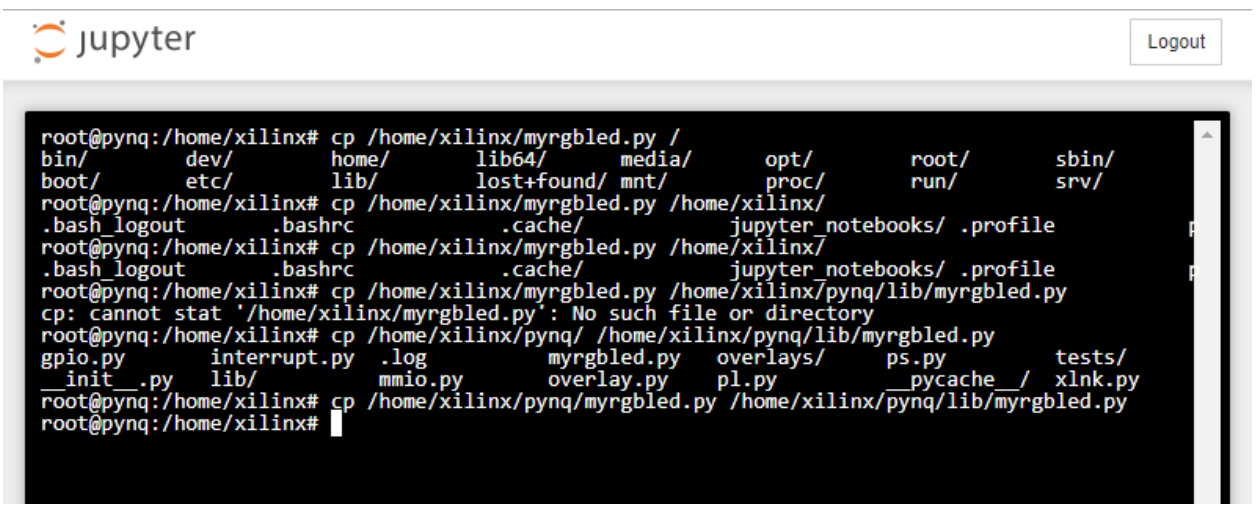

Figure 1: Jupiter Notebook terminal, copy a file.

Listing 1: Jupiter Notebook terminal, copy a file.

```
root@pynq:/home/xilinx# cp /home/xilinx/pynq/myrgbled.py /home/xilinx/pynq/lib/myrgbled.py
```

```python
from .myrgbled import MYRGBLED
```

Listing 2: Python code changed on line 45 of file __init__.py.

```python
self.rgbleds = ([None] * 4) + [pynq.lib.MYRGBLED(i)
                                    for i in range(4, 6)]
```

Listing 3: Python code changed on line 99 of file base.py.

2.2 Part II - LED Groove Bar

The Groove LED Bar can be turned on in level increments from 0 to 10 where 0 is off and 10 all segments on. The brightness of the leds can be defined independently with an value from 0 to 3 where 0 is off, 1 is low, 2 is medium, and 3 is led brightness high. As graphical user interface (GUI) two integer slider are used SL1 and SL2 as shown in Figure 2. The source code is shown in Listing ?? in Section 4.2 of the appendix.

Figure 2: Groove LED Bar program output.

3 Conclusion

4 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

4.1 Python code Listings Part I - RGB LED Driver

```
#   Copyright (c) 2016, Xilinx, Inc.
#   All rights reserved.
#
#   Redistribution and use in source and binary forms, with or without
#   modification, are permitted provided that the following conditions are met:
#
#   1.  Redistributions of source code must retain the above copyright notice,
#       this list of conditions and the following disclaimer.
#
#   2.  Redistributions in binary form must reproduce the above copyright
#       notice, this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#
#   3.  Neither the name of the copyright holder nor the names of its
#       contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
#   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
#   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
#   THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
#   PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
#   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
#   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
#   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
#   OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
#   WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
#   OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
#   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


# from .audio import Audio
# from .video import HDMI
# from .video import Frame
# from .dma import DMA
# from .trace_buffer import Trace_Buffer
# from .usb_wifi import Usb_Wifi

from .pynqmicroblaze import PynqMicroblaze
from .pynqmicroblaze import MicroblazeRPC
from .pynqmicroblaze import MicroblazeLibrary
from .axigpio import AxiGPIO
from .dma import DMA
from .dma import LegacyDMA
from .led import LED
from .myrgbled import MYRGBLED
from .switch import Switch
from .button import Button

from .arduino import Arduino
from .arduino import Arduino_DevMode
from .arduino import Arduino_IO
from .arduino import Arduino_Analog
from .arduino import Arduino_LCD18

from .pmod import Pmod
from .pmod import Pmod_DevMode
from .pmod import Pmod_ADC
from .pmod import Pmod_DAC
from .pmod import Pmod_OLED
```

```
from .pmod import Pmod_LED8
from .pmod import Pmod_IO
from .pmod import Pmod_IIC
from .pmod import Pmod_DPOT
from .pmod import Pmod_TC1
from .pmod import Pmod_TMP2
from .pmod import Pmod_ALS
from .pmod import Pmod_Cable
from .pmod import Pmod_Timer
from .pmod import Pmod_PWM

from .logictools import LogicToolsController
from .logictools import Waveform
from .logictools import BooleanGenerator
from .logictools import PatternGenerator
from .logictools import TraceAnalyzer
from .logictools import FSMGenerator

from . import video
from . import audio
from . import dma

__author__ = "Graham Schelle"
__copyright__ = "Copyright 2016, Xilinx"
__email__ = "pynq_support@xilinx.com"
```

Listing 4: Jupyter Notebook file __init__ saved as *.py file.

```
#   Copyright (c) 2017, Xilinx, Inc.
#   All rights reserved.
#
#   Redistribution and use in source and binary forms, with or without
#   modification, are permitted provided that the following conditions are met:
#
#   1.  Redistributions of source code must retain the above copyright notice,
#       this list of conditions and the following disclaimer.
#
#   2.  Redistributions in binary form must reproduce the above copyright
#       notice, this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#
#   3.  Neither the name of the copyright holder nor the names of its
#       contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
#   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
#   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
#   THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
#   PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
#   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
#   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
#   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
#   OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
#   WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
#   OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
#   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


import pynq
import pynq.lib
import pynq.lib.video
import pynq.lib.audio
from .constants import *
from pynq.lib.logictools import TraceAnalyzer


__author__ = "Peter Ogden"
__copyright__ = "Copyright 2017, Xilinx"
```

```python
__email__ = "pynq_support@xilinx.com"


class BaseOverlay(pynq.Overlay):
    """ The Base overlay for the Pynq-Z1

    This overlay is designed to interact with all of the on board peripherals
    and external interfaces of the Pynq-Z1 board. It exposes the following
    attributes:

    Attributes
    ----------
    iop_pmoda : IOP
         IO processor connected to the PMODA interface
    iop_pmodb : IOP
         IO processor connected to the PMODB interface
    iop_arduino : IOP
         IO processor connected to the Arduino/ChipKit interface
    trace_pmoda : pynq.logictools.TraceAnalyzer
        Trace analyzer block on PMODA interface, controlled by PS.
    trace_arduino : pynq.logictools.TraceAnalyzer
        Trace analyzer block on Arduino interface, controlled by PS.
    leds : AxiGPIO
         4-bit output GPIO for interacting with the green LEDs LD0-3
    buttons : AxiGPIO
         4-bit input GPIO for interacting with the buttons BTN0-3
    switches : AxiGPIO
         2-bit input GPIO for interacting with the switches SW0 and SW1
    rgbleds : [pynq.board.RGBLED]
         Wrapper for GPIO for LD4 and LD5 multicolour LEDs
    video : pynq.lib.video.HDMIWrapper
         HDMI input and output interfaces
    audio : pynq.lib.audio.Audio
         Headphone jack and on-board microphone

    """

    def __init__(self, bitfile, **kwargs):
        super().__init__(bitfile, **kwargs)
        if self.is_loaded():
            self.iop_pmoda.mbtype = "Pmod"
            self.iop_pmodb.mbtype = "Pmod"
            self.iop_arduino.mbtype = "Arduino"

            self.PMODA = self.iop_pmoda.mb_info
            self.PMODB = self.iop_pmodb.mb_info
            self.ARDUINO = self.iop_arduino.mb_info

            self.audio = self.audio_direct_0
            self.leds = self.leds_gpio.channel1
            self.switches = self.switches_gpio.channel1
            self.buttons = self.btns_gpio.channel1
            self.leds.setlength(4)
            self.switches.setlength(2)
            self.buttons.setlength(4)
            self.leds.setdirection("out")
            self.switches.setdirection("in")
            self.buttons.setdirection("in")
            self.rgbleds = ([None] * 4) + [pynq.lib.MYRGBLED(i)
                                           for i in range(4, 6)]

            self.trace_pmoda = TraceAnalyzer(
                self.trace_analyzer_pmoda.description['ip'],
                PYNQZ1_PMODA_SPECIFICATION)
            self.trace_arduino = TraceAnalyzer(
                self.trace_analyzer_arduino.description['ip'],
```

```
                    PYNQZ1_ARDUINO_SPECIFICATION)
```

Listing 5: Jupyter Notebook file base saved as *.py file.

```
#   Copyright (c) 2016, Xilinx, Inc.
#   All rights reserved.
#
#   Redistribution and use in source and binary forms, with or without
#   modification, are permitted provided that the following conditions are met:
#
#   1.  Redistributions of source code must retain the above copyright notice,
#       this list of conditions and the following disclaimer.
#
#   2.  Redistributions in binary form must reproduce the above copyright
#       notice, this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#
#   3.  Neither the name of the copyright holder nor the names of its
#       contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
#   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
#   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
#   THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
#   PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
#   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
#   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
#   PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
#   OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
#   WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
#   OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
#   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


from pynq import MMIO
from pynq import PL
import time

__author__ = "Graham Schelle"
__copyright__ = "Copyright 2016, Xilinx"
__email__ = "pynq_support@xilinx.com"


RGBLEDS_XGPIO_OFFSET = 0
RGBLEDS_START_INDEX = 4
RGB_CLEAR = 0
RGB_BLUE = 1
RGB_GREEN = 2
RGB_CYAN = 3
RGB_RED = 4
RGB_MAGENTA = 5
RGB_YELLOW = 6
RGB_WHITE = 7


class MYRGBLED(object):
    """This class controls the onboard RGB LEDs.

    Attributes
    ----------
    index : int
        The index of the RGB LED, from 4 (LD4) to 5 (LD5).
    _mmio : MMIO
        Shared memory map for the RGBLED GPIO controller.
    _rgbleds_val : int
        Global value of the RGBLED GPIO pins.

    """
```

```
    _mmio = None
    _rgbleds_val = 0

    def __init__(self, index):
        """Create a new RGB LED object.

        Parameters
        ----------
        index : int
            Index of the RGBLED, from 4 (LD4) to 5 (LD5).

        """
        # print("Changed LED Driver to MYRGBLED.") # debugging only
        if index not in [4, 5]:
            raise ValueError("Index for onboard RGBLEDs should be 4 or 5.")

        self.index = index
        if MYRGBLED._mmio is None:
            base_addr = PL.ip_dict["rgbleds_gpio"]["phys_addr"]
            MYRGBLED._mmio = MMIO(base_addr, 16)

    def on(self, color):
        """Turn on a single RGB LED with a color value (see color constants).

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
        if color not in range(8):
            raise ValueError("color should be an integer value from 0 to 7.")

        rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
        new_val = (MYRGBLED._rgbleds_val & ~rgb_mask) | \
                  (color << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

    def off(self):
        """Turn off a single RGBLED.

        Returns
        -------
        None

        """
        rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
        new_val = MYRGBLED._rgbleds_val & ~rgb_mask
        self._set_rgbleds_value(new_val)

    def red_on(self):
        """Turn on a single RGB LED with color value red.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
```

```
#            raise ValueError("color should be an integer value from 0 to 7.")
        new_val = (MYRGBLED._rgbleds_val ) | (RGB_RED << ((self.index-RGBLEDS_START_INDEX)*3))
#        print(MYRGBLED._rgbleds_val)
#        print(new_val)
#        print( (RGB_RED << ((self.index-RGBLEDS_START_INDEX)*3)) )
        self._set_rgbleds_value(new_val)

    def red_off(self):
        """Turn on a single RGB LED with color value red.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
#            raise ValueError("color should be an integer value from 0 to 7.")

        new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_RED << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

    def green_on(self):
        """Turn on a single RGB LED with color value green.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
#            raise ValueError("color should be an integer value from 0 to 7.")

        new_val = (MYRGBLED._rgbleds_val ) | (RGB_GREEN << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

    def green_off(self):
        """Turn on a single RGB LED with color value red.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
#            raise ValueError("color should be an integer value from 0 to 7.")

        new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_GREEN << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

```

```
    def blue_on(self):
        """Turn on a single RGB LED with color value blue.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
#            raise ValueError("color should be an integer value from 0 to 7.")

        new_val = (MYRGBLED._rgbleds_val ) | (RGB_BLUE << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

    def blue_off(self):
        """Turn on a single RGB LED with color value red.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
#        if color not in range(8):
#            raise ValueError("color should be an integer value from 0 to 7.")

        new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_BLUE << ((self.index-RGBLEDS_START_INDEX)*3))
        self._set_rgbleds_value(new_val)

    def status(self):
        rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
        return ((MYRGBLED._rgbleds_val )& ~rgb_mask)

    def pwm(self, color, interval):
        """Turn on a single RGB LED with color value red.

        Parameters
        ----------
        color : int
           Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
        for x in range(0, 1):
            self.blue_on()
            time.sleep(interval)
            self.blue_off()


    def write(self, color):
        """Set the RGBLED state according to the input value.

        Parameters
        ----------
```

```
        color : int
            Color of RGB specified by a 3-bit RGB integer value.

        Returns
        -------
        None

        """
        self.on(color)

    def read(self):
        """Retrieve the RGBLED state.

        Returns
        -------
        int
            The color value stored in the RGBLED.

        """
        return (MYRGBLED._rgbleds_val >>
                ((self.index-RGBLEDS_START_INDEX)*3)) & 0x7

    @staticmethod
    def _set_rgbleds_value(value):
        """Set the state of all RGBLEDs.

        Note
        ----
        This function should not be used directly. User should call
        `on()`, `off()`, instead.

        Parameters
        ----------
        value : int
            The value of all the RGBLEDs encoded in a single variable.

        """
        MYRGBLED._rgbleds_val = value
        MYRGBLED._mmio.write(RGBLEDS_XGPIO_OFFSET, value)
```

Listing 6: Jupyter Notebook file myrgbled saved as *.py file.

4.2 Python code Listings Part II - LED Groove Bar

```

# coding: utf-8

# # Part II - LED Groove Bar
# Demonstrates how the LED Groove Bar level is set with slider SL1. The brightness can be chosen in four levels with slider SL2.
#
# LED Bar Brightness
#  - 0 = off
#  - 1 = low
#  - 2 = medium
#  - 3 = hight

# In[2]:


# Steup the PYNQ board
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")

from pynq.lib.pmod import Grove_LEDbar
from pynq.lib.pmod import PMOD_GROVE_G1 # Import constants
import ipywidgets as widgets
from IPython.display import display
```

```python

# For delays
from time import sleep

# Global values
g_ledBrightness  = 3
g_leds = 0

# defined functions
def handle_slider1_change(change):
    global g_leds
    ledbar.write_level(change.new, g_ledBrightness, 1)
    g_leds = change.new
def handle_slider2_change(change):
    global g_ledBrightness
    g_ledBrightness = change.new
    ledbar.write_level(g_leds, change.new, 1)
   # ledbar.write_brightness(ledbar.read(), change.new)


# Instantiate Grove LED Bar on PMODA and on Pmod2Grove G1
ledbar = Grove_LEDbar(base.PMODA, PMOD_GROVE_G1)
ledbar.reset()

# Flash 2 extreme LEDs of the LED Bar in a loop, dubbiging only
# for i in range(5):
#     ledbar.write_binary(0b1000000001)
#     sleep(0.5)
#     ledbar.write_binary(0b0000000000)
#     sleep(0.5)

# GUI
slider1 = widgets.IntSlider(min=0, max=10, value=0, description='SL1')
slider2 = widgets.IntSlider(min=0, max=3, value=0, description='SL2')

slider1.observe(handle_slider1_change, names='value')
slider2.observe(handle_slider2_change, names='value')

display(slider1, slider2)
```

Listing 7: Part II - LED Groove Bar Python code.

4.3 Python code Listings Part III - Music Synthesizer