EGR680 High Level Implementation on FPGA

Laboratory 10

PYNQ Embedded Design using Jupyter Notebooks

Professor: Dr. C. Parikh

Student: Dimitri Häring

November 13, 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 Jupyter Notebook

The Jupyter Notebook is an integrated development environment (IDE) integrated in a web browser. This allows to program code on the system and directly executes it on it in an cell. Shift + Enter executes a cell. A cell can contain Code, Markdown, RawNBConvert, and Heading. The Markdown can be used to document the code and due to the fact that it is a subset of HTML the browser can interpret it nicely. The web server is accessible on Internet protocol (IP) address 192.168.2.99 with password "xilinx". The web server allows also to brows the tree and edit most of the files. Nevertheless, it is recommended to access files via network address directly.

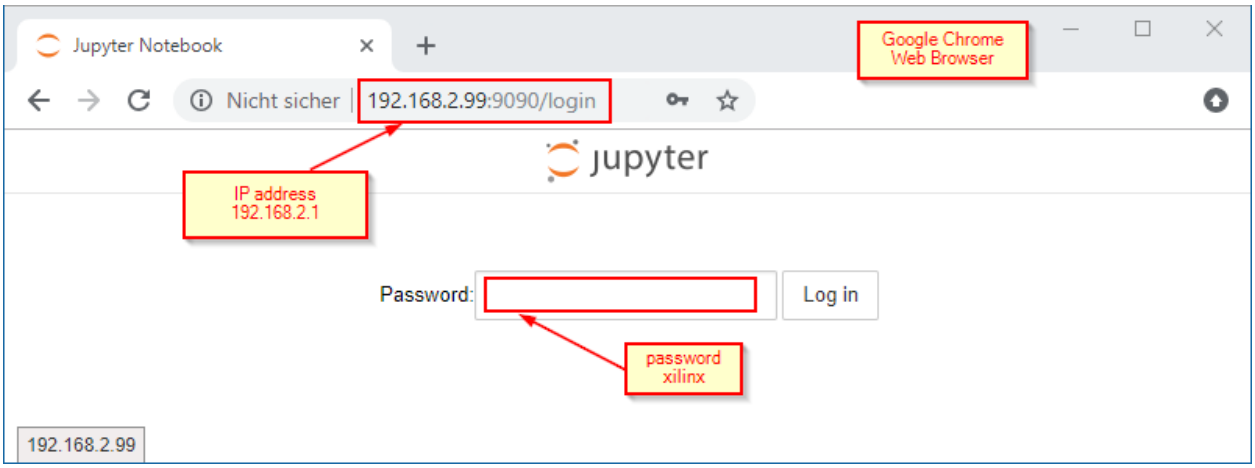

Figure 1: Jupiter Notebook login shown in Google Chrome.

2.2 Part II - Let's Make a Deal

In part two the Jupyter Notebook is used to program the game Let's Make a Deal. Is a game where a player can choose between four different doors. The computer decides behind which door win is placed.

You can do the following to control the game:

Button 0 pressed:	Door 1
Button 1 pressed:	Door 2
Button 2 pressed:	Door 3
Button 3 pressed:	Door 4
Switch 0 on:	Exit program
Switch 1 on:	Exit program

Figure 2 shows the console output shown in the web browser after executing the file with "Run All" and multiple rounds of choosing a door.

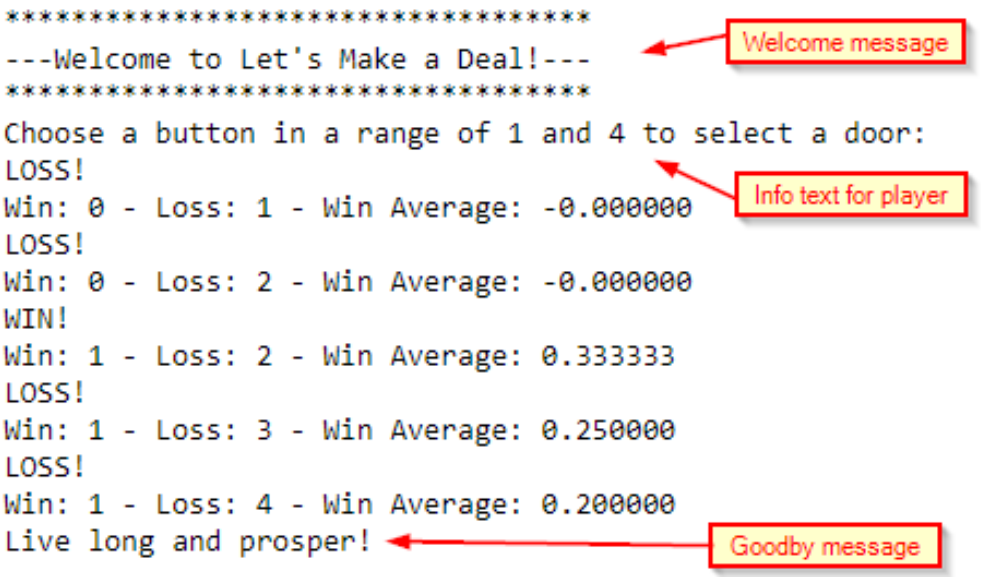

Figure 2: Let's Make a Deal program output.

2.3 Part III - Jupyter Notebook GUI using ipywidgets

The description of part three of the lab is as followed.

A Jupyter Notebook is not limited to just text output. By using the iPywidgets library, an interactive GUI can be created to interact with the I/O of the PYNQ board. For more information on ipywidgets, you can refer to the document "ipywidgets_Userguide.pdf" available on the course website.

Figure 3 shows the view programmed for part three which is used to control the LEDs. The first four buttons control LED zero to three and the status is shown with a label below the button. The two sliders control the RGB LEDs and can be used to change the color of each LED independently. Further more it shows how GUI elements are embedded in code and will appear after the cell where the display function is called. This gives the user an interesting interfacing option where he can adjust a code snipe and only execute the cell instead of the entire program.

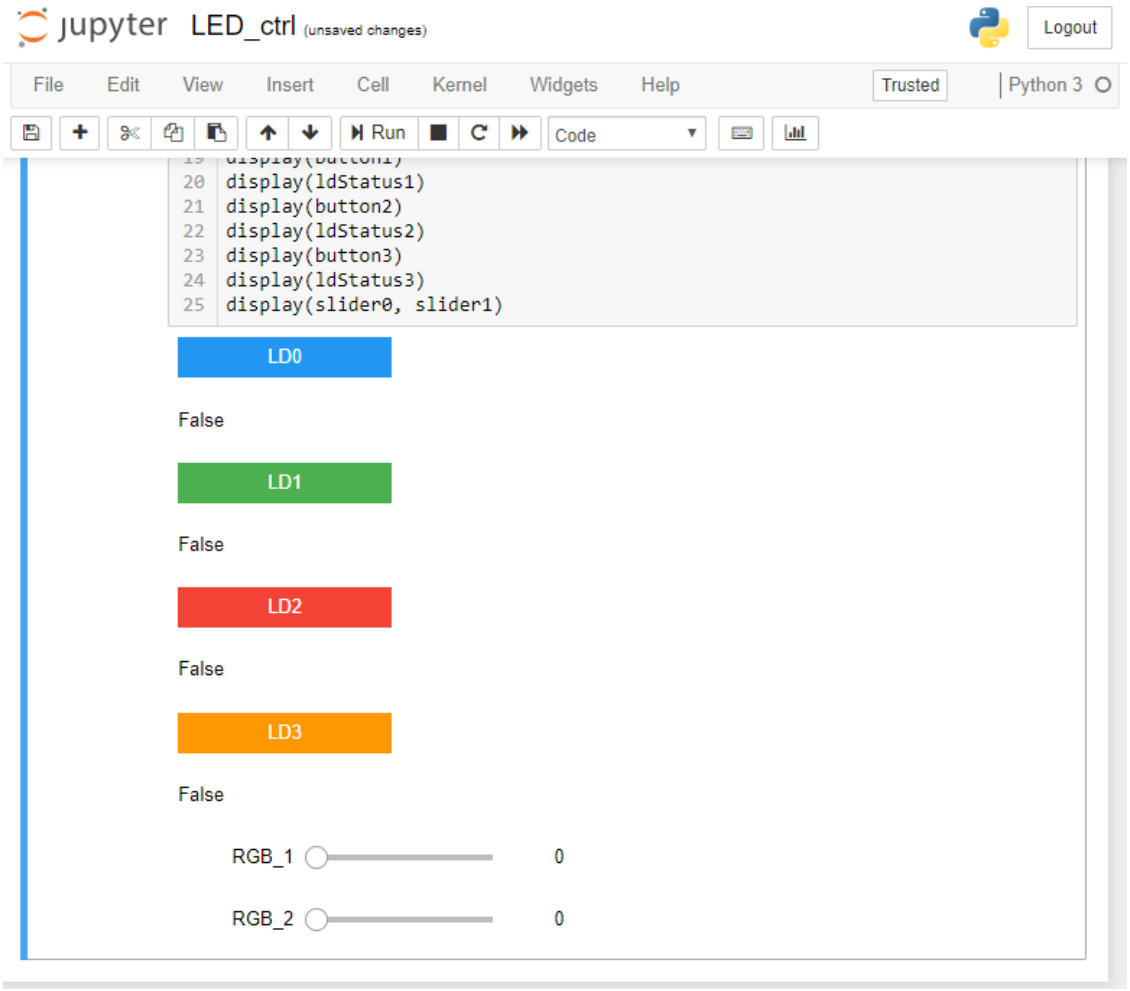

Figure 3: Buttons and slider for LED control.

3 Conclusion

The lab demonstrates the use of the ipython as simple and fast scripting language that allows access to vast number of packages that allows an decreased development time. The Jupiter Notebook provides a way to program and develop interactive GUIs that allows partially run code in separate cells.

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 II

```
1
2  # coding: utf-8
3
4  # ## Let's Make a Deal
5  #
6  # Is a game where a player can choose between four different doors. The computer decides
       behinde which door win is placed.
7  #
8  # The game is started by select Menubar -> Cell -> Run All
9  #
10 # You can do the following to control the game:
11 #
12 #     Button 0 pressed:         Door 1
13 #     Button 1 pressed:         Door 2
14 #     Button 2 pressed:         Door 3
15 #     Button 3 pressed:         Door 4
16 #
17 #     Switch 0 on:              Exit program
18 #     Switch 1 on:              Exit program
19 #
20 #     ************************************************************
21 #     * FSM State machine from C program                          **
22 #     ************************************************************
23 #                         000
24 #     +-------------------------+
25 #     | Idle                    |
26 #     | Display user input      |<---------+
27 #     |                         |          |
28 #     +-------------------------+          |
29 #         |                                |
30 #         |                                |
31 #         |                 001            |
32 #     +-------------------------+          |
33 #     |                         |          |
34 #     | Wait for button         |          |
35 #     | coin_val on Pmod A      |          |
36 #     +-------------------------+          |
37 #         |                                |
38 #         |                                |
39 #         |                 002            |
40 #     +-------------------------+          |
41 #     | user output             |          |
42 #     | if (rnd == door)        |----------+
43 #     | Display output          |
44 #     +-------------------------+
45 #
46
47 # In[61]:
48
49
50 import time
51 from pynq.overlays.base import BaseOverlay
52 import random
53
54
55 # ### Load overlay bitstream file generated by Vivado
56
57 # In[62]:
58
```

```
59
60 base = BaseOverlay("base.bit")
61
62
63 # ### Seed random number generator
64
65 # In[63]:
66
67
68 random.seed(time.localtime)
69
70
71 # ### Variables
72
73 # In[64]:
74
75
76 Delay1 = 0.3
77 Delay2 = 0.1
78 color = 0
79 rgbled_position = [4,5]
80 randomNo = random.randint(1,4)
81 winCnt = 0
82 lossCnt = 0
83 avgMedium = 0
84
85
86 # ### Define functions here
87 # Function decision() provides the computaion of the win and loss with average and a consol
      ouput accordingly.
88 # #### Colors RGB LED No 4 and 5
89 #      off = 0
90 #      blue = 1
91 #      green = 2
92 #      tÃ¼rkies = 3
93 #      red = 4
94 #      purple = 5
95 #      yellow = 6
96 #      white = 7
97 #
98
99 # In[65]:
100
101
102 def decision(rnd, door, win, loss, avg):
103     #print("\r\nComputer chose %d" % rnd)
104     # print("----------------------\r\n")
105     if (rnd == door):
106         color = 2
107         print("WIN!")
108         win = win + 1
109     else:
110         color = 4
111         print("LOSS!")
112         loss = loss + 1
113         # print("----------------------\r\n")
114     avg = win/ (win + loss)
115     print("Win: %d - Loss: %d - Win Average: %f" % (win, loss, avg))
116
117     for led in rgbled_position:
118         base.rgbleds[led].write(color)
119         base.rgbleds[led].write(color)
120
121     return [win, loss, avg]
122
123
124 # ### Start progarm
125
```

```
# In[66]:


print("\r\n\r\n**********************************")
print("---Welcome to Let's Make a Deal!---")
print("**********************************")
# print("Select between 1 and 4 to seed the Random Number Generator: ")
print("Choose a button in a range of 1 and 4 to select a door: ")


for led in base.leds:
    led.on()
while (base.switches.read() == 0):
#while (base.buttons[3].read()==0):
    if (base.buttons[0].read()==1):
    #    color = (color+1) % 8
        for led in base.leds:
            led.off()
        time.sleep(Delay2)

        for led in base.leds:
            led.toggle()
            time.sleep(Delay2)

     #  for led in rgbled_position:
     #      base.rgbleds[led].write(color)
     #      base.rgbleds[led].write(color)
     #   time.sleep(Delay1)

        ret = decision(random.randint(1,4), 1, winCnt, lossCnt, avgMedium)
        winCnt = ret[0]
        lossCnt = ret[1]
        avgMedium = ret[2]


    elif (base.buttons[1].read()==1):
        for led in base.leds:
            led.off()
        time.sleep(Delay2)
        for led in base.leds:
            led.toggle()
            time.sleep(Delay2)
        ret = decision(random.randint(1,4), 2, winCnt, lossCnt, avgMedium)
        winCnt = ret[0]
        lossCnt = ret[1]
        avgMedium = ret[2]

    elif (base.buttons[2].read()==1):
        for led in reversed(base.leds):
            led.off()
        time.sleep(Delay2)
        for led in reversed(base.leds):
            led.toggle()
            time.sleep(Delay2)
        ret = decision(random.randint(1,4), 3, winCnt, lossCnt, avgMedium)
        winCnt = ret[0]
        lossCnt = ret[1]
        avgMedium = ret[2]

    elif (base.buttons[3].read()==1):
        for led in reversed(base.leds):
            led.off()
        time.sleep(Delay2)
        for led in reversed(base.leds):
            led.toggle()
            time.sleep(Delay2)
        ret = decision(random.randint(1,4), 4, winCnt, lossCnt, avgMedium)
        winCnt = ret[0]
```

```
        lossCnt = ret[1]
        avgMedium = ret[2]

print('Live long and prosper!')
for led in base.leds:
    led.off()
for led in rgbled_position:
    base.rgbleds[led].off()



### End Program
```

Listing 1: Jupyter Notebook file Rand_game saved as *.py file.

4.2 Python code Listings Part III

```

# coding: utf-8

## LED Ctrl
#
# Use buttons and sliders to control the LEDs on the board.
#
# The program is started by select Menubar -> Cell -> Run All
#
# Cell -> Current Outputs -> Toggle Scrolling
#

# In[1]:


import time
from pynq.overlays.base import BaseOverlay

import ipywidgets as widgets
from IPython.display import display



### Load overlay bitstream file generated by Vivado

# In[2]:


base = BaseOverlay("base.bit")


### Variables

# In[3]:


button0 = widgets.Button(description="LD0", button_style='primary')
button1 = widgets.Button(description="LD1", button_style='success')
button2 = widgets.Button(description="LD2", button_style='danger')
button3 = widgets.Button(description="LD3", button_style='warning')

ldStatus0 = widgets.Label(value='False')
ldStatus1 = widgets.Label(value='False')
ldStatus2 = widgets.Label(value='False')
ldStatus3 = widgets.Label(value='False')



### Define functions here
# Function decision() provides the computaion of the win and loss with average and a consol
    ouput accordingly.
```

```
# #### Colors RGB LED No 4 and 5
#      off = 0     blue = 1     green = 2     tÃ¼rkies = 3     red = 4     purple = 5     yellow = 6
#      white = 7
#

# In[4]:


def on_button0_clicked(b):
    base.leds[0].toggle()
    ldStatus0.value = '' + ('False' if base.leds.read() & int('0001',2) == 0 else 'True')
def on_button1_clicked(b):
    base.leds[1].toggle()
    ldStatus1.value = '' + ('False' if base.leds.read() & int('0010',2) == 0 else 'True')
def on_button2_clicked(b):
    base.leds[2].toggle()
    ldStatus2.value = '' + ('False' if base.leds.read() & int('0100',2) == 0 else 'True')
def on_button3_clicked(b):
    base.leds[3].toggle()
    ldStatus3.value = '' + ('False' if base.leds.read() & int('1000',2) == 0 else 'True')
def handle_slider0_change(change):
    base.rgbleds[4].write(change.new)
def handle_slider1_change(change):
    base.rgbleds[5].write(change.new)


# ### Start progarm

# In[5]:


button0.on_click(on_button0_clicked)
button1.on_click(on_button1_clicked)
button2.on_click(on_button2_clicked)
button3.on_click(on_button3_clicked)

slider0 = widgets.IntSlider(min=0, max=7, value=0, description='RGB_1')
slider1 = widgets.IntSlider(min=0, max=7, value=0, description='RGB_2')

slider0.observe(handle_slider0_change, names='value')
slider1.observe(handle_slider1_change, names='value')

# turn all led's off
for led in base.leds:
    led.off()


display(button0)
display(ldStatus0)
display(button1)
display(ldStatus1)
display(button2)
display(ldStatus2)
display(button3)
display(ldStatus3)
display(slider0, slider1)
```

Listing 2: Jupyter Notebook file LED_ctrl saved as *.py file.