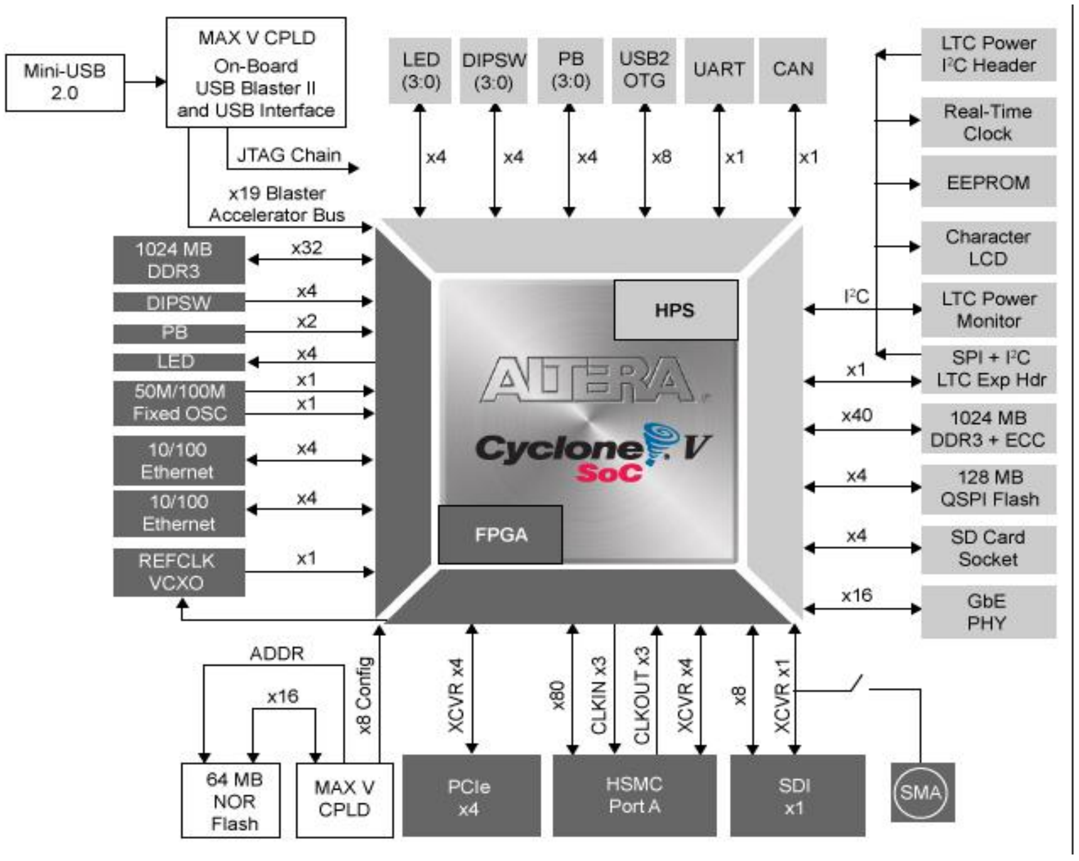
WORKING SOC BASED SYSTEM USING CYCLONE V SOC DEVICE



For

Project 3 of Programmable Logic Design (ECEN-5863) at

University of Colorado Boulder, Spring 2018

By

Anay Gondhalekar and Vikrant Waje

Date: 2nd May 2018

Contents

Executive Summary

In this report, we aim to build a Real time clock (Module 1), Morse Code generator (Module 2), HPS to FPGA communication(Module 3), SolarTracker(Module 4) using Altera DE1-SOC hosting a Cyclone V FPGA . The DE1- SOC features Cyclone V FPGA which can perform a wide variety of functions and is flexible enough to support many applications. The Cyclone V FPGA is used to design wide variety of application with use of schematic and Hardware Description Language with Quartus prime software.

Module-1(Light the lights, hit the heights) deals with designing a Real-time clock, Morse code generator and wide variety of small project that involves seven segment displays and LED's as output by using the Quartus prime software. In this, Module we have primarily dealt with keys and switches and how to make them interact with the LED's and seven segment display by means of counter, Real time clock and Morse code generator.

Module-2 (Embedded Software development on an SOC using Python) focuses on using the Xilinx Pynq board to make Morse code generator. The Pynq board is programmed using Python language. The device is programmed to respond asynchronously to button press using asyncio library in Python. Pressing button 1 prints one morse code character on terminal while pressing another button prints another character on terminal console.

Module -3(Connecting HPS to FPGA) concentrates on building output executable on linux operating system which is loaded in DE1 SOC using SD card. The DE1 SOC was able to host the Linux operating system and was able to program various peripherals successfully using makefiles. This module was good enough to get familiar with HPS/ARM

	MODULE 1	MODULE 2	MODULE 3
Objective	Light the lights, hit the heights	Embedded software development on an Soc using Python	Connecting HPS to FPGA
Device Used	Cyclone V on DE1SoC Board and Xilinx Pynq board		
Fmax(in MHz)	No clock for Lab1 Lab2: G. 374.67 H. 315.96 I. 254.0 J. 156.76	-	717.36

% Utilization (Out of 32,070)	A. 1(<1%) G. 10(<1%) B. 3(<1%) H. 44(<1%) C. 2(<1%) I. 242(<1%) D. 1(<1%) J. 357(<1%) E. 6(<1%) F. 16(<1%)	-	2195(7%)

Module 4 (Solar-Tracker) involves using a Servomotor and Light dependent resistor circuit. Based on the intensity of light that is detected on those LDR, the servomotor will rotate towards direction which has highest amount of light intensity.

MODULE 4	Objective	Device Used	Fmax	% Utilization
	FPGA based Solar Tracker	DE1 SoC	85.11 Mhz	209/32070 (<1%)

Module 1-Light the lights, hit the heights

Objectives

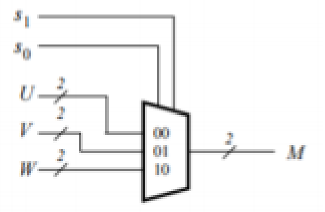
- **Project objectives**
  1. Design simple input and output devices for FPGA CYCLONE V using HDL
  2. Design a Real time Clock system, Morse code generator, and multiplexers.

- **Learning objectives**
  1. Become Familiar with how to interface output devices to an FPGA
  2. Learn to do pin-mapping of pins using Pin Assignment Editor.
  3. Learn how to use counters in designing complex system such as Real Time clock and Morse code generator.

Procedure

1. First, we start with Quartus prime software to write Hardware Description Language for the given digital circuit
2. There are various parts within Module 1 that involves HDL coding, they are as follows:
3. **LAB 1:**

   1) **LAB1 PART1:**
      Design a circuit that involves ten switches as input which will display the state of switch on ten LED's. We write a VHDL code and use pin planner for pin assignments
   2) **LAB1 PART2:**
      Design a four bit wide 2:1 multiplexer using switches and LED's as output. If select input is logic 1, the LED's shall glow corresponding to SW[7:4], however, if logic 0, then it would glow corresponding to SW[3:0]
   3) **LAB1 PART3:**
      Design a two bit wide 3:1 multiplexer with switches and LED's. The circuit is implemented as follows:



   4) **LAB1 PART4:**
      Design a digital circuit to display character on 7 segment display. The character 'd', 'E' and '1' are displayed on the 7 segment display.
   5) **LAB1 PART5:**
      Design a digital circuit that involves displaying character on the 7 segment display using switches. The goal of the design was to build a a circuit that could display any word on three 7 segment display that is composed of three switches to display the character

   6) **LAB1 PART6:**

The goal of this lab was to design a system that could display the rotating character on all the seven-segment display. We write VHDL code such that based on the select lines, we change the position of “DE1” word on the display.

LAB 5:

1) **LAB5 PART 1 :**
   Design a variable sized counter by using GENERIC declaration. We have written VHDL code for counter such that push button 0 acts as a Reset signal while push button 1 acts as a clock input. We display the output of counter on the LED’s and use one out of those 10 LED’s to act as a terminal count signal.

2) **LAB 5 PART 2**:
   In this, we have used our part 1 code to develop a 3 digit BCD counter. The BCD counter is programmed as follows, when the BCD counter in k-1 position reaches it’s terminal count which is 9 in this case, we increment the count value of kth position counter so that the BCD counter counts from 000 to 999. We have used push button 0 as Reset input and push button 1 as Clock input.

3) **LAB5 PART 3:**
   Design a Real time counter that can display minutes, seconds and hundredths of a second**.** For this, we have used six seven-segment display such that the 7 Segment display in the last two position counts from 0 to 99 and hence increment the counter value of next two 7 segment display after it reaches 99. The seven segment display at 3rd and 4th position count from 0 to 59 as they count the number of second’s whose range is from (0-59). So this two Seven segment increment the count value of minute display when it reaches 59. The 7 segment display at the MSB counts from 0-59

4) **LAB5 PART 4:**
   Design a morse code generator that can generate morse code for 8 letters plus a “SOS” message**.**

Test Results and Questions

1. **What is fmax for each lab?**
   Ans:

Section	Fmax
Lab 1 Part 1	No clock
Lab 1 Part 2	No clock
Lab 1 Part 3	No clock
Lab 1 Part 4	No clock
Lab 1 Part 5	No clock
Lab 1 Part 6	No clock
Lab 5 Part 1	374.67
Lab 5 Part 2	315.96
Lab 5 Part 3	254
Lab 5 Part 4	156.76

2. **What is the percentage utilization of logic of each lab?**
   Ans:

Section	Logic Utilization
Lab 1 Part 1	1(<1%)
Lab 1 Part 2	3(<1%)
Lab 1 Part 3	2(<1%)
Lab 1 Part 4	1(<1%)
Lab 1 Part 5	6(<1%)
Lab 1 Part 6	16(<1%)
Lab 5 Part 1	10(<1%)
Lab 5 Part 2	44(<1%)
Lab 5 Part 3	242(<1%)
Lab 5 Part 4	357(<1%)

3. **After programming, does it behave as expected?**
   Ans: Yes, for each part of Lab 1 and Lab 5, the device works as expected.

Lesson Learned

1. We learned how to do pin assignment using TCL scripts and Altera’s pin assignment editor
2. We learned how to design counters of variable size using Generic keyword and use it to generate delay that was used to build Real time clock and Morse code generator.
3. We learned that some FPGA’s have Active low pin which only get activated when voltage is logic 0.
4. We learned how to slow down the clock frequency using Clock divider circuit which we used in designing of Real time clock

Module 2- Embedded Software Development on an SOC using Python

Objectives

- **Project Objectives**
    1. The main objective of this module is to generate morse code by encoding two keys as two letters of Morse code
    2. The program should be such that it should asynchronously interrupt the FPGA and there should be no polling done whatsoever.

- **Learning Objectives**
    1. Learn how to use Python to program the FPGA , switches and LED's on pynq board.
    2. Learn about the Xilinx's Pynq board ability to program the FPGA using Python .
    3. Learn how to use Jupiter Notebook which is the online IDE used to program the Pynq board.
    4. Learn how different Python programming is as compared to HDL language.

Procedure

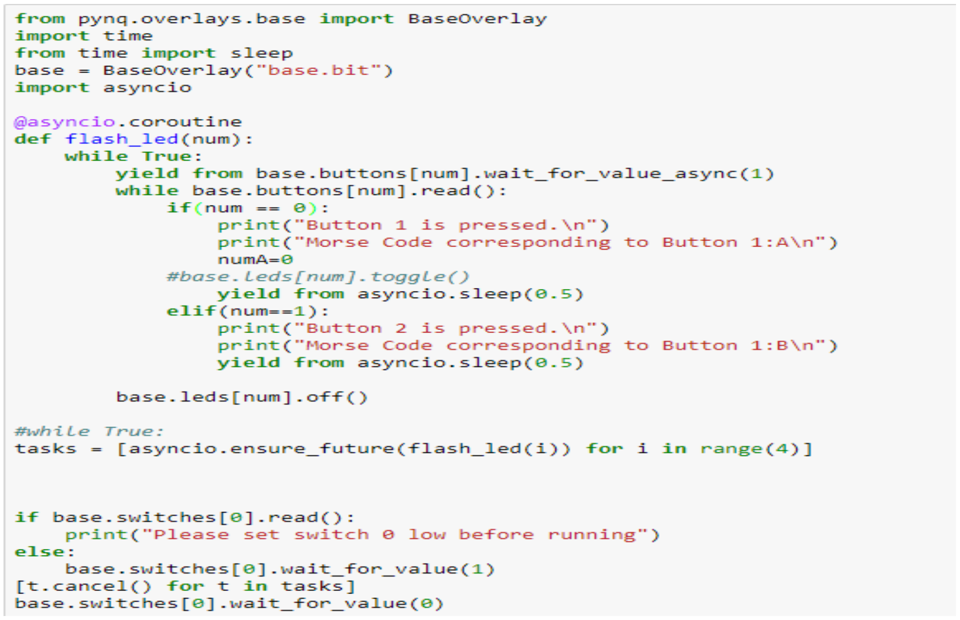
1. First, we open connect the Xilinx's Pynq board to Router using Ethernet cable

2. After, the LED's on Pynq board blink up, it indicates that the Xilinx Pynq board is able to load the image that was loaded onto the SD card

3. Having done this, connect the Pynq board to Jupiter's notebook website.

4. On the website, we get an online Python IDE that can be used to write python programs and execute them to get required output

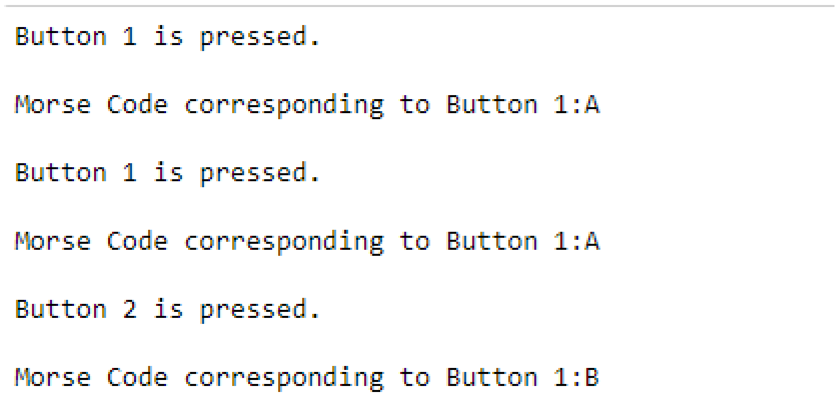
5. Since we are using asynchronous interrupt method, we used the asyncio library to write the program as follows and got the expected results.



```
from pynq.overlays.base import BaseOverlay
import time
from time import sleep
base = BaseOverlay("base.bit")
import asyncio

@asyncio.coroutine
def flash_led(num):
    while True:
        yield from base.buttons[num].wait_for_value_async(1)
        while base.buttons[num].read():
            if(num == 0):
                print("Button 1 is pressed.\n")
                print("Morse Code corresponding to Button 1:A\n")
                numA=0
            #base.leds[num].toggle()
                yield from asyncio.sleep(0.5)
            elif(num==1):
                print("Button 2 is pressed.\n")
                print("Morse Code corresponding to Button 1:B\n")
                yield from asyncio.sleep(0.5)

        base.leds[num].off()

#while True:
tasks = [asyncio.ensure_future(flash_led(i)) for i in range(4)]


if base.switches[0].read():
    print("Please set switch 0 low before running")
else:
    base.switches[0].wait_for_value(1)
[t.cancel() for t in tasks]
base.switches[0].wait_for_value(0)
```

Test Results and Questions

1. **Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?**
   Ans: Yes, the board behaved as it is expected.



```
Button 1 is pressed.

Morse Code corresponding to Button 1:A

Button 1 is pressed.

Morse Code corresponding to Button 1:A

Button 2 is pressed.

Morse Code corresponding to Button 1:B
```

Lesson Learned

- We demonstrated Xilinx's Pynq board ability to develop a Morse code generator so that it can display a letter on console when corresponding key is being pressed.
- We learned that Python is also one of the way to program FPGA board. This programming also made us little familiar with python programming syntax.
- We learned how to make use of Pynq board to design high end applications which can be easily programmed using Python

Module 3- Connecting the HPS to FPGA

Objectives

- **Project objectives**
  1. Using HPS/ARM to communicate with FPGA
  2. Using Linux to build executable using makefiles
  3. Taking advantage of serial communication for transferring the executable from Linux operating system to DE-1 SOC

- **Learning objectives**
  1. Learn how to use linux on FPGA (how to flash Linux Operating system on SD card and use it)
  2. Learn how to use HPS/ ARM for setting up communication with FPGA.

Procedure

1. First, we start with my_first_hps_fpga project that included the pin declarations for HPS and FPGA

2. We use Qsys to use different predefined components such as JTAG to Avalon Master Bridge, System ID peripheral, ON chip memory,Clock source, and Interrupt Capture Module. We make the connection between all those components so that we can generate HDL code accordingly.

3. After this is done, we compile the project and once that is done, we use the .sof file to program our CYCLONE V FPGA(DE-1 SOC) using USB blaster.

4. We then shift our focus to design C program for ARM to control the pio_led controller. We understand the file that is already provided to ue, we review how the mapping of physical to virtual address is done.

5. Now, we will generate the executable using the makefile through Linux command line

6. Once we are ready with the executable file, we need to copy the executable from to DE1 SOC board. We need some serial terminal software such as Putty, Teraterm which can be used to transfer the file and communicate with board using a baud rate of 9600. We use the copy command(scp) to transfer the executable file into DE1-SOC board

7. The next step is to change the attributes of executable file so that it can be executed successfully on the DE-1 SOC board to get the desired output.

Test Results and Questions

1. **What is the Fmax of compiled design? What is the highest clock speed used in the design?**
   Ans: 717.36 Mhz

2. **Total Registers(Flipflops) and percentage utilization of FPGA logic?**

Ans: Total Registers= 3261

Percentage utilization= 2195/32070 (7%)

3. **Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?**
   Ans: Yes, the board behaved as per it was programmed. Th LED's blinked incrementally as well as decremented.

Lessons learned

1. We learned how to do mapping of physical to virtual address
2. We manifested the ability of serial interface that can be used between the Host computer and DE-1 SOC board to perform important actions as instructed by the user.
3. We learned about how to use DE-1 board's HPS and FPGA to build the design.

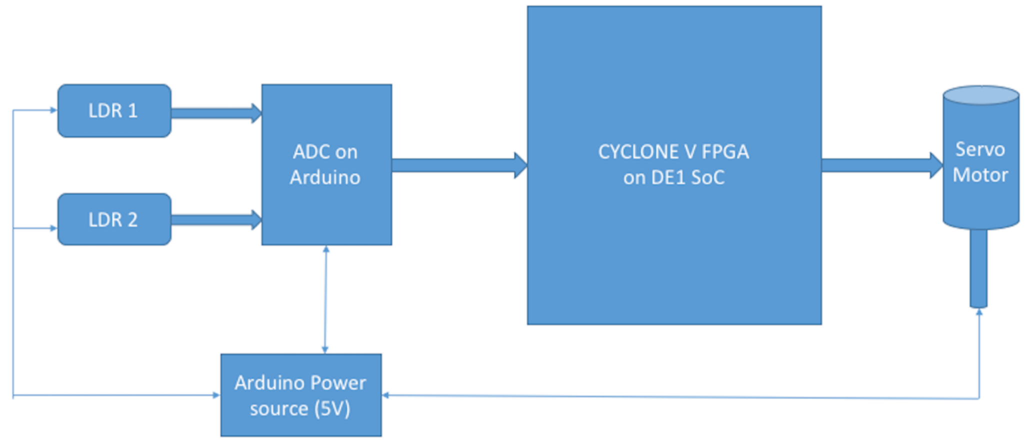
Module 4- Solar Tracker

Objectives

- **Project Objectives**
  1) Use Cyclone V FPGA to interact with LDR's (Analog input)
  2) Determine which LDR has the highest light intensity
  3) Depending on Light intensity, program the FPGA to move the servo motor at specified angle
  4) Generate PWM to encode the position of servo-motor.

- **Learning Objectives**
  1) Learn how to generate PWM signal to encode servo motor position
  2) Learn how to interface analog sensors with FPGA

System Block Diagram



Working

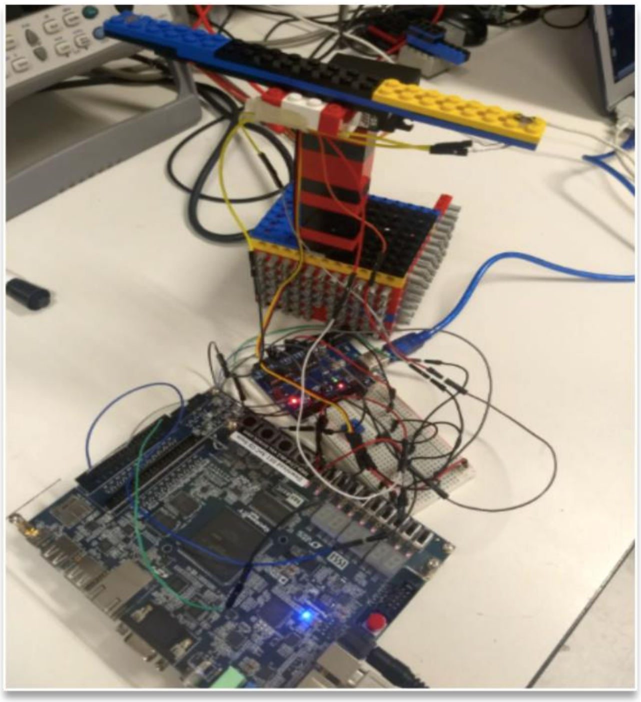
The aim of the project is to utilize the maximum solar energy through solar panel. The process involves interfacing ADC to convert the voltage across LDR to give a digital input to the input to the FPGA. Then code to obtain the sensed light and control the motor for panel movement depending upon the difference in LDR values. Further, fabricate and interface a servo moto control interfaced with PWM with the help of Verilog HDL.

Design flow includes comparing the LDR values and accordingly generating counters to create PWM for driving the servo motor. An Arduino is used to give 5V to the motor as well as to use the ADC on board. Servo motor needs a 20ms time period pulse to hold its position. The duty cycle decides the value of the angle. Counter are used to generate the PWM specific to the Servo motor as 1ms corresponds to 0 degree and 2ms to 180degree and others directly proportional to it.

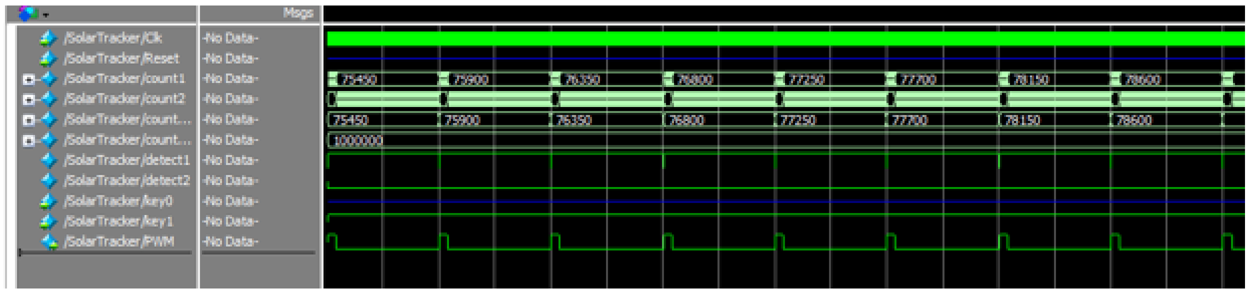
Servo motor was selected over stepper motor since stepper motor suffers from resonance issues, also only the rotation of shaft from 0 to 180degrees is needed. Servo motor is more precise in terms of angle rotation as compared to servo motor

Project Realization

The image below shows the layout and physical construction of the project.



Simulation Output:


Msgs
/SolarTracker/Clk -No Data-
/SolarTracker/Reset -No Data-
/SolarTracker/count1 -No Data-
/SolarTracker/count2 -No Data-
/SolarTracker/count... -No Data-
/SolarTracker/count... -No Data-
/SolarTracker/detect1 -No Data-
/SolarTracker/detect2 -No Data-
/SolarTracker/key0 -No Data-
/SolarTracker/key1 -No Data-
/SolarTracker/PWM -No Data-
75450 75900 76350 76800 77250 77700 78150 78600
1000000

Lessons learned:

1. Always check the hardware and code in parts before integrating.
2. Learnt to use PWM generation by counters for Servo motors with FPGA.
3. Interfacing sensors and peripherals to the FPGA.

Conclusion

In this project, we worked on DE1-SOC board that hosts Cyclone V FPGA.

In Module 1, we learned about how pin assignments can be done using Altera Pin Assignment editor or TCL scripts. We learned how to make FPGA interacts with switches, LED's and keys present on the board. We learned how to use clock of 50 MHz to derive slower clocks using clock divider logic so as to drive the Real-time clock. We learned how effectively counters can be used to generate the required delay as in case of Morse Code generator.

In Module 2, we learned about basics of Python programming which was used to program the Pynq board. We got to know the ability of Pynq board to interact with it's on board peripherals and also got to know how easy Python programming is. We learned about the online IDE which is based on Jupiter's Notebook that is used to program the Pynq board.

In Module 3, we learned how Qsys can be used to design the FPGA fabric and how to make connections between them. We got familiar how to load Linux operating system on DE1-SOC board that can be used to generate executable which can program the DE1-SOC board. We learned how HPS extends the ability of FPGA for manipulation and execution.

In Module 4, we made a Solar Tracker system, that can track the sun. We interfaced the analog sensor (LDR's) to our FPGA and hence got to know how sensors can be interfaced with the Cyclone V FPGA. Also, we learned how to generate PWM signal that was used to control the servo motor position. Working with Servo motor also gave us understanding how exactly servo motor operates and why we use Servo motor instead os stepper motor/Dc motor.

References

http://www.alterawiki.com/wiki/Category:Qsys

https://www.altera.com/en_US/pdfs/literature/tt/tt_my_first_fpga.pdf

Appendix

Project Staff

Vikrant Waje	Anay Gondhalekar
Graduate Student Embedded Systems Engineering University of Colorado Boulder vikrant.waje@colorado.edu	Graduate Student Embedded Systems Engineering University of Colorado Boulder anay.gondhalekar@colorado.edu