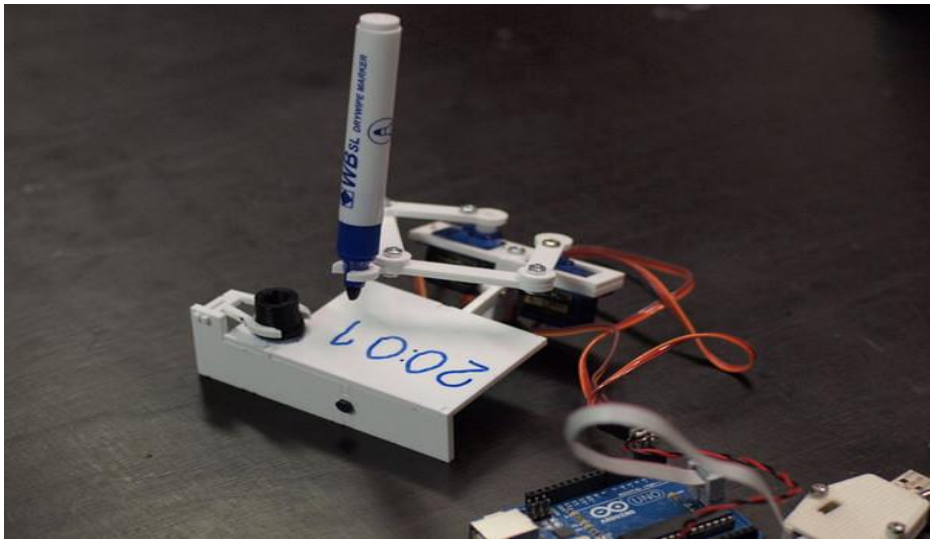
ECE 544

Embedded Systems Design with FPGAs

Winter 2015

Final project report

‘Counter-Plotter’



Abhishek Yadav

Venkata Hemanth Tulimilli

Bala Prashanth Reddy Chappidi

Saketh Kumsi

TABLE OF CONTENTS

1. INTRODUCTION

From getting the proposal for this project approved to successfully demoing this project in the class, 'Counter-Plotter' has been a busy ride for us. The video of the 'Oreo splitter' embedded system shown in the first lecture of the course plus the interest in making mechanical working models sparked the idea to do the final project in similar direction. But what could be achieved in a 12-15 days span was a big question confronting us.

We could not waste time in designing a mechanical assembly from scratch. Even though some of us had experience in robotics, the overhead of testing the mechanical assembly and making it work was predicted to be too much. We began searching for projects whose mechanical assembly could be replicated or 3D printed and we found out exactly what we were looking for in the project 'PlotClock' – a mechanical arm that writes the time on a whiteboard after every minute.

This led to further discussions and we then finalized the decision to make the 4 digit 'Counter-Plotter' which would count up and down and write corresponding values on the whiteboard. Additional functionalities that were added were the digital clock mode and support for writing alphabets.

This project report strives in the detailed description of every aspect of this project. All aspects ranging from project planning to the hardware and software configurations are explained. We hope this report can guide any future students of engineering to build this system from scratch.

2. PROJECT MANAGEMENT

2.1 Project tasks

The CHECKLIST:

- **Building the Model**
  - Buying parts.
  - 3D printing.
  - Making the 3D model.
- **Hardware / Software in Vivado / SDK**
  - Creating the Hardware in Vivado.
  - Creating a basic application.
  - Circuitry for impedance matching of servos and FPGA output port.
  - Integration and first check of three servos working together.
  - Modifying the application to use different modes of operation.
- **Calibration and final steps**
  - Making a ‘map’ of positions on the white board.
  - Using those positions to check the pattern written on board.
  - Testing with different delays and marker tip positions.
  - Final touches to application to make accurate from end user perspective.

2.2 Managing project work

We are a team of four. We tried to divide the project work on the basis of our strengths.

The division of labor was done as follows-

- **Abhishek** and **Saketh**
  - Hardware model 3D printing.
  - Making the model.
  - Creating hardware in Xilinx Vivado.

- **Venkata** and **Prashanth**
    - Create software application in Xilinx SDK.
- **Whole Team together**
    - Integration of the project hardware and software.
    - Working on the calibration to make the arm write lucidly and accurately.

NOTE 1: 3D printing and hardware model

- Hardware model 3D printing and making the model was thought to be a straightforward task but the size of the model being small and the frail body of 3mm wide 3d printed parts was a slight negative. Had to be extra careful while handling those parts.
- Printing an extra model is a good idea to be on the safe side.
- The 3d printed ‘slate’ has a rough surface. We used a laser-cut acrylic sheet as a slate for writing on with a marker.

NOTE 2: Calibration

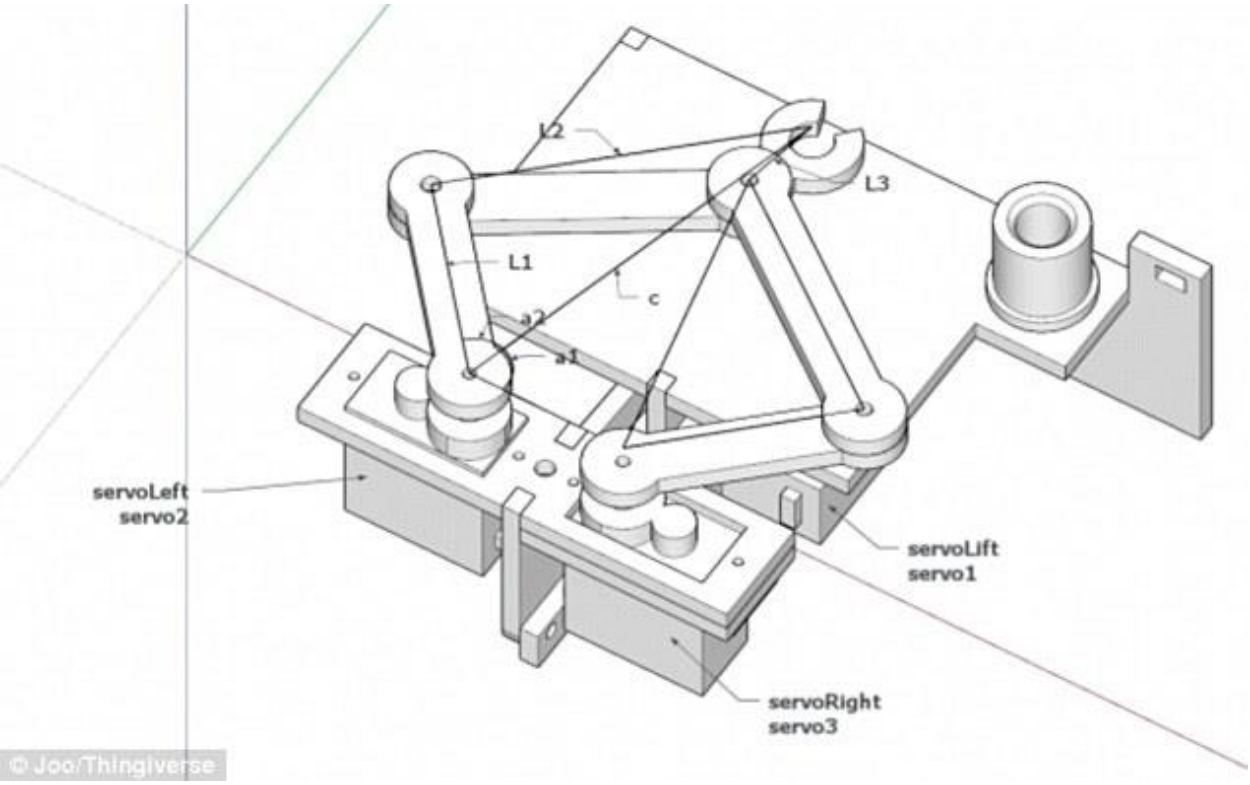
- We did the calibration from scratch. In the initial stages of the project we thought calibration would be easy task. But many factors in the actual assembly tend to change the calibration causing the plotted points to skew.
- A lot of time should be given to calibration to get it accurate.
- More details about the challenges of calibration is given in later part of the report.

3. THE 3D PRINTED MODEL

We got the 3D model for our project online. The credits for this incredible idea and model goes to 'JOO' who uploaded his project for everyone at this webpage-

Webpage link - http://www.thingiverse.com/thing:248009/#files

Here is a mechanical drawing of the model



We downloaded the .stl files and 3D printed the parts at Electronics Prototyping Laboratory in the Engineering Building at Portland State University.

The size of the slate on which the marker writes is approximately 7cm x 5 cm. So it was decided to plot just 4 digits or letters on the slate. The slate we used is made from acrylic.

If a choice was given to make the model again we would like to use acrylic instead of the 3D printed material. 3mm acrylic material is more rigid than 3mm 3D printed plastic material.

4. FUNCTIONAL SPECIFICATION

4.1 Brief description

From functional specification point of view the application is straightforward. We are using Nexys4 FPGA with LCD to ensure user interaction with the system. Values provided by the user makes the plotter work in various modes of operation described below.

4.2 Actual working of the system

- At startup or at system reset, the system asks/waits for the user to input the mode of operation of Counter-Plotter.
- User changes the mode of operations with the help of switches on the FPGA and changes and feeds the input values for a particular mode of operation with the use of buttons on the FPGA.
- Once the values are fed, the Plotter starts plotting the counter values or the time accordingly.

4.3 Detailed description

a. SWITCHES: *(Active high)*

- **<u>All switches low</u> : Waiting mode**

  Asks and waits for user input from first three switches.

- **<u>Switch 0</u> [SW0] : Up counter mode**

  The plotter will start from 0000 to some point ‘xxxx’ being set by the user.

- **<u>Switch 1</u> [SW1] : Down counter mode**

  The plotter will start from some point ‘xxxx’, being set by the user, to 0000.

- **Switch 2 [SW2] : Digital clock mode**

  The user will be asked to set the current time and the plotter will write the time and erase and write the new time again after one exact minute.

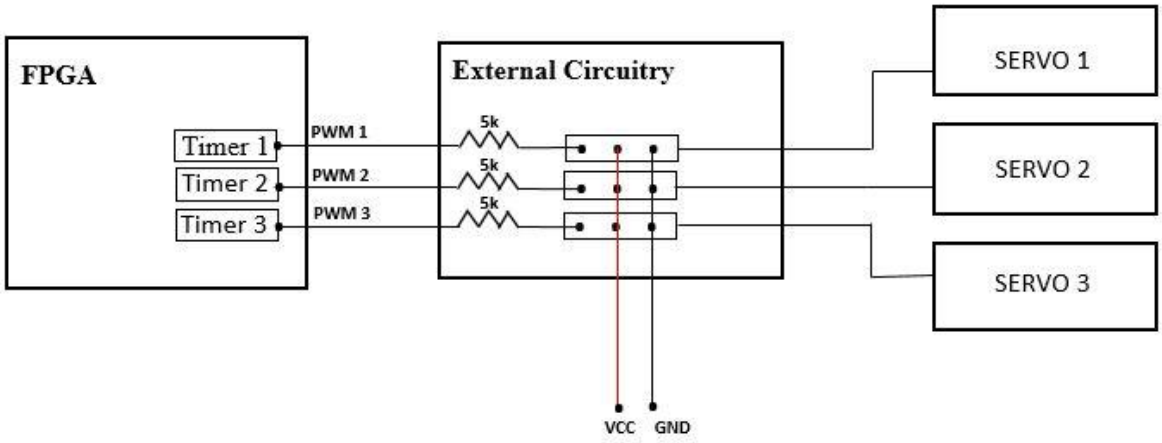
- **Switch 15 [SW15] : Terminate / stop the application**

b. BUTTONS:

- **Up button [btnU]:** Increments the value to be entered by the user.
- **Down button [btnD]:** Decrements the value to be entered by the user.
- **Left button [btnL]:** Moves the cursor position to the left to feed the value to that position.
- **Right button [btnR]:** Moves the cursor position to the right to feed the value to that position.
- **Center button [btnC]:** Accepts the value entered by the user and starts the plotter.

5. HARDWARE CONFIGURATION

5.1 Block diagram



Counter-Plotter Block Diagram

Brief description of block diagram components:

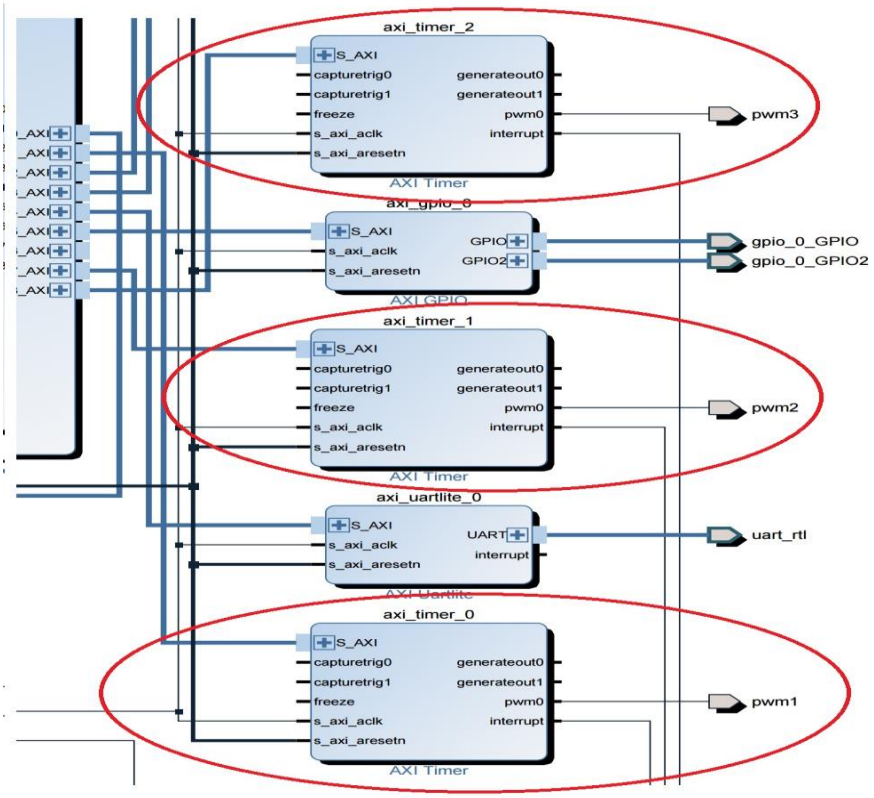
- **Embedded system created in Vivado (On FPGA)**
  The hardware created in Xilinx Vivado tool provides 3 Pulse width modulated signals through three different Axi-timers. These PWM signals are provided to the external circuitry.
- **External Circuitry**
  Matches the output impedance of the ports on FPGA by providing a 5K ohm resistor in series and provides the PWM signal to the servos. 5 volts VCC and ground are provided by a USB to the external circuitry.

- **Servo Motors**
  Tower pro 9g servo motors are used in the system. They are connected to the external circuitry which provides them with the PWM signal and the required voltage of 5 volts.

5.2 Embedded system configuration

We used Vivado and the IP Integrator to create an embedded system with this minimum specification.

- Microblaze, mdm, interrupt controller, etc.
  Axi clock provides 50 KHz clock for PWM signal for the servo motors to operate.
- Program/Data memory (board-dependent):
  - Local (BRAM) memory of 64KB for program/data memory
- 3 x AXI-Timers– used for 3 PWMs to drive the servo motors



3 AXI Timers in block design (Vivado)

- FIT timer set to 40 Khz
- GPIO, two ports, 8 bits wide. One input one output.
- UartLite peripheral. Configure the Uart Baud Rate to 19200

- Nexys4IO and Pmod544IO
  IPs used in earlier projects which are created and packaged by *Professor Roy Kravitz* are used to provide access to the Nexys4 devices and the PmodCLP and PmodENC.

5.3 FPGA port connections and PMODs

We used Nexys4 DDR FPGA and PMOD CLP (LCD) and PMOD ENC (Rotary encoder) for the project.

Connections to the FPGA

- **PMOD CLP (LCD):** As discussed earlier, the LCD module acts as a user interface and enhances user interaction with the FPGA. It is connected to JA and JB ports of the FPGA.
- **PMOD ENC (Rotary encoder):** The rotary encoder was used in the initial phase of the project to debug the signals provided to servo motors. It provided easy changes in PWMs to check the servo motor positions. Its functionality was taken off from the application in the end. It was connected to input port JD of the FPGA.
- **PWM signals:** The three Pulse width modulated signals coming from the three Axi-timers were routed to JC[4], JC[5] and JC[6] ports of the FPGA.

5.4 External circuitry

FPGA port and Servo motor configurations

- The Nexys4 DDR FPGA ports have an output impedance of 3.6K ohms.
- The servo motors had an input impedance of 8.6K ohms.
- Therefore, 5K ohm resistors were used in series for maximum power transfer.

Power supply design decision

- Servo motors needed input voltage of 5 volts.
- As our project needed precision of the servo positions, we needed constant current throughout. Therefore idea of using batteries was discarded. Carrying a power supply every-time we needed to run the project was not a good idea too.

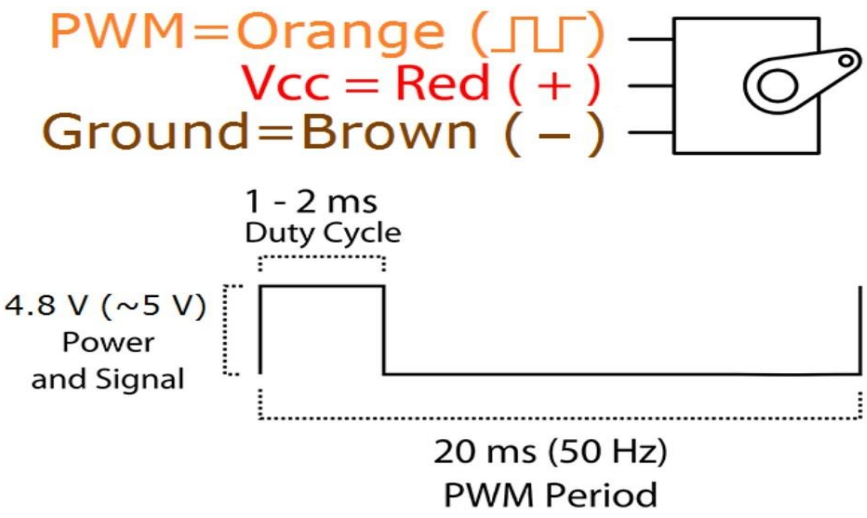
- We decided to use **USB port** to give constant voltage and current.
- So, the current design uses 5 volt VCC and ground from a USB port on a laptop.

5.5 Servo motor specifications

As discussed earlier our design used Tower pro 9g servo motors. These motors fit perfectly in the design and hence are used.

Servo Specifications:

- Operating voltage: 4.8 (~5v)
- Operating speed: 0.1s/60 degrees
- Temperature range: 0 to 55 degree celcius



Servo motor specifications

Duty cycle ranges for the servo motors:

- **Checked on frequency generators**
  0 degrees – 2% duty cycle
  180 degrees – 12.8% duty cycle
- **Used in the design**
  Minimum – 4% duty cycle
  Maximum – 11% duty cycle

NOTE: **Servo care**

- Such mini/micro servos have nylon gears inside of them and a circuit which has a low temperature range. (Tower pro 9g has range of 0 to 55 degree celcius)
- At servo stall, these circuits tend to take in more current to try to make the servo move. This leads to heating up and crashing of a servo motor.
- Any forceful tugging of servo horn when the servos are connected to a power supply and pwm signal circuit can break the teeth of the gears, crashing the servo motor.

6. SOFTWARE CONFIGURATION AND MANAGEMENT

6.1 Software application management

Creating a software application was an important part of the project which needed planning and management of different modes of operations and different points to be considered on the slate.

Look up tables/ arrays for the points on the slate and functions were used to tackle the complexity of the design.

6.2 Drivers

The drivers that were included in the application are ones that were used in first project 'Pulse width modulation and detection'.

Drivers: xparameters.h, xintc.h, xtmrctr.h, xgpio.h, mb_interface.h, platform.h, Nexys4IO.h, PMod544IOR2.h, pwm_tmrctr.h

6.3 Software application configuration

Here are the list of functions used in the software application. For more details look for the file 'final_project.c'

FUNCTIONS :

1. **int do_init(void);** -- Initializes the system
2. **void delay_msecs(unsigned int);** -- Busy-wait delay for "msecs" miliseconds
3. **void FIT_Handler(void);** -- Fixed interval timer interrupt handler
4. **void update_lcd(void);** -- Update LCD for current values
5. **void course_write(void);** -- Plots greeting message "ECE 544" on the board
6. **void PositionSet(float, float);** -- Sets position of the mechanical arm to a position
7. **void lift_hand(void);** -- It lifts marker off the board to stop writing
8. **void rest_hand(void);** -- Places marker on the board for writing
9. **void display_symbol(void);** -- writes ":" on a particular position

10. **void display_num(int, int);** -- writes numbers in which-ever position we want
11. **void erase(void);** -- erases the board
12. **void set_initial_state(u8);** -- initializes the start value for all the modes
13. **void update_up_count(void);** -- updates Up Count value for up counter mode
14. **void update_down_count(void);** -- updates Down Count value for down counter mode
15. **void update_clk(void);** -- updates Time for digital clock mode

7. CALIBRATION

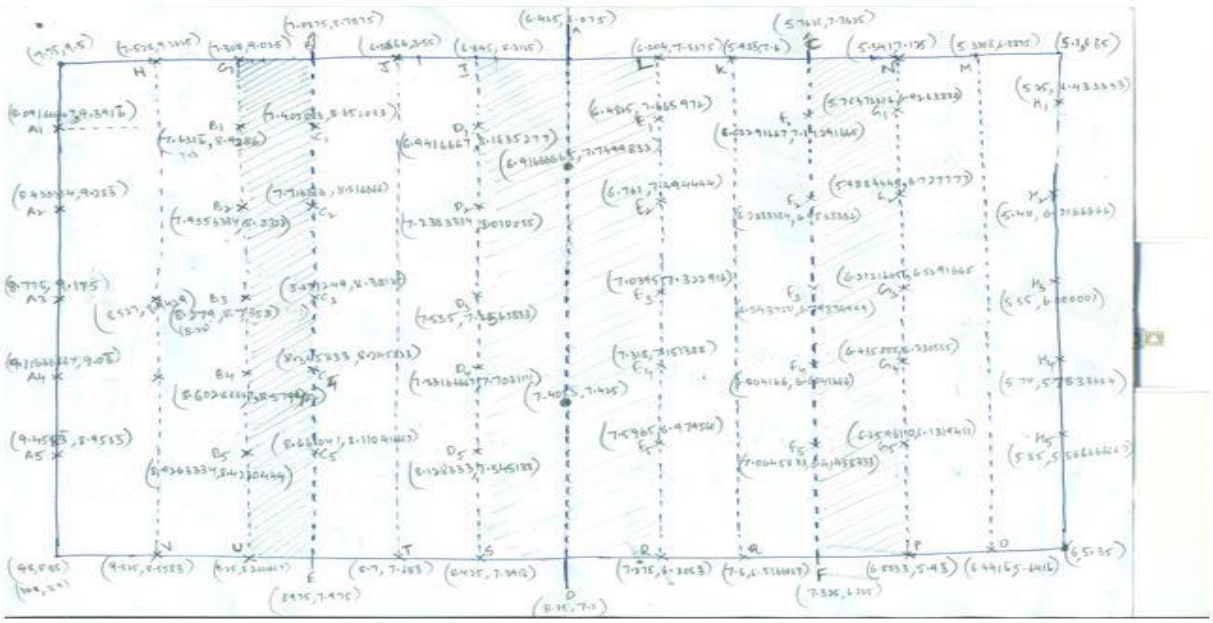
Calibration was the most important aspect of the project which needed a lot of planning and testing to get the project right. As the servos that we are using are just basic 180 degree rotation servos and not precision servo motors, a lot of factors were responsible for accurate calibration of the position of the tip of the marker to the slate. This part of the report is dedicated to discuss the plans that were executed for the calibration and the factors that were causing hindrance to an accurate calibrated model.

7.1 Calibration plans

a. Original plan

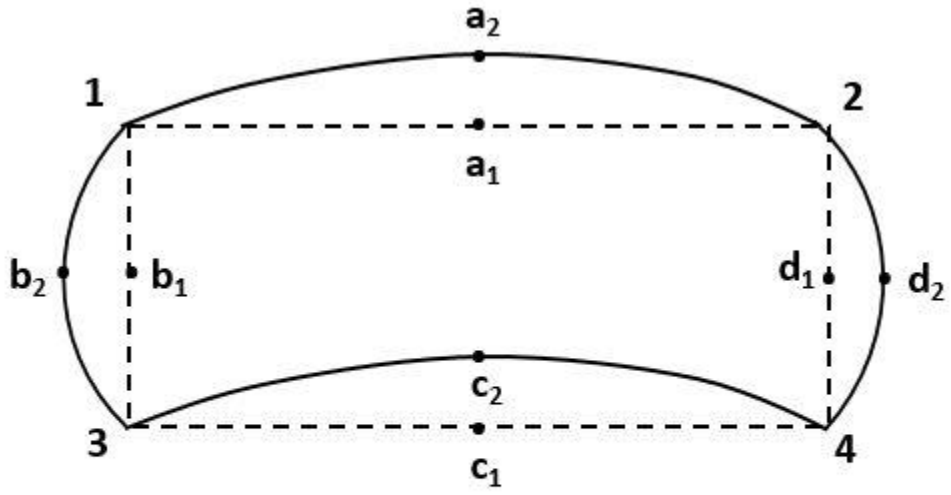
The original plan was simple.

- Use frequency generators to give pulse width modulated inputs to the two servos responsible for writing.
- Note down the values for the four corners of the slate. Calculate the average of two points and get the mid-point. Keep getting mid-points from previous midpoints, note those points down and get an array of many points.
- Here is the image of the formulated points –



Calculated points for 7-segment plot

- **Advantages:**
  - Many points on slate = accurate plot of digits
  - Easy math = easily done
- **Disadvantages:**
  - Same output duty cycles from FPGA led to different points on the slate. Have to do it again while setting different PWMs in application.
  - Even from the inputs from FPGA we can't use the 'averaging' method to calculate the mid-points because of the design of the arm and its 'Inverse Kinematics'. See the **figure** given below ('Error in plotting) for more clarity. The midpoints lie on the curves and not on the dotted straight lines we require.



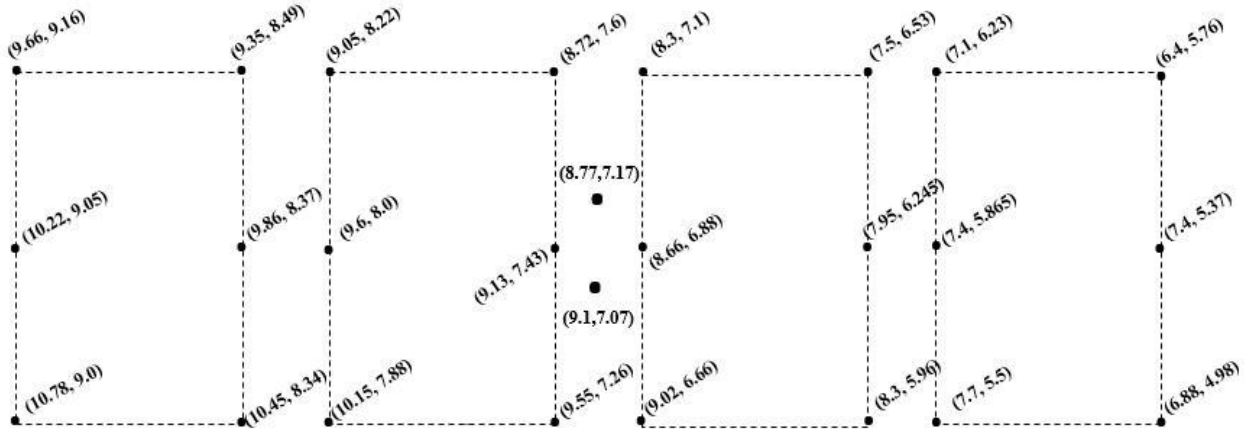
Error in plotting
Actual values Vs. Needed Values

b. Revised plan

Time was running and we needed to get things done. Formulating complex math for our solution would take time and may not be accurate enough and will need many iterations to get it right. Therefore, the following 'hardworking' method was thought of.

Manual method –

- Seven segment display has 6 vertices, get those 6 x 4 = 24 points for 4 digits plus the 2 points for semicolon (for digital clock mode). 26 points should be plotted on the slate.
- "Connect the dots" to write a digit or a letter.
- Closer the dots – Straighter the connecting lines could be.
- Find all 26 points by manually providing different PWM duty cycles to two servo motors.
- **<u>Advantages:</u>**
    - This method was getting things done.
    - Plotter wrote lucidly.
    - Exactly what we wanted.
- **<u>Disadvantages:</u>**
    - Average calibration time to get 26 points correct = 5 hours.
    - Had to do this comprehensive calibration again and again due to many factors. (discussed in 'Factors hindering accurate calibration')
- Final calibration used for project demo is shown in the **figure** below



Current 7-segment plot points

7.2 Factors hindering accurate calibration

As discussed earlier there were many factors which hindered accurate calibration of the servo motors. These factors made us recalibrate again and again for at-least 5 times in the duration of the work of the project.

Factors:

- Frail and small structure of the 3D printed design and material.
- Nuts and bolts getting loose on many occasions due to continuous moving of servo motors.
- Servo horns popped out due to pressure of the marker on the slate.
- The assembly needed some kind of base for stability.
-

Solution:

- Keeping the amount of tightening of all the nuts and bolts to the same level every-time.
- Gluing the assembly to a base.
- Gluing the ‘needed optimum state’ of the model and then calibrate for the final time.

These solutions look simple enough to carry out. But the optimum state and the consideration of the factors come from experience and working with the model for a span of time.

8. ACKNOWLEDGEMENTS

For any project to be successful the efforts have to be carried out by a team. But the advices, help and support and few little pushes in the right direction are equally necessary. First and foremost we would like to thank Professor Roy Kravitz for giving us this opportunity, sharing his ideas and pushing us in the right direction of learning throughout the span of the course. The course structure that he has planned has made us good kind of workaholics which gave us the satisfaction to complete this interesting project successfully, in time. We would like to thank Yiwei Lee for his help and advices throughout the semester. We would also like to thank our colleague and friend Abhishek Malik for multiple brainstorming sessions with our team about small and big points of discussions about the project alike. This project needed a 3D mechanical model badly and we are deeply indebted to 'JOO' on Thingiverse website. We thank him for making his project available to everyone online. And lastly we would like to thank the newly inaugurated EPL lab and its manager Nathan Bergey for all the help in 3D printing and laser cutting. The lab is a goldmine for robotics enthusiasts and engineers alike.

9. REFERENCES

1. Plotclock - http://www.thingiverse.com/thing:248009 , https://github.com/9a/plotclock

2. Servo motor specifications - http://datasheet.sparkgo.com.br/SG90Servo.pdf

3. Portland State University, ECE-544, Project 1 – Pulse width modulation and detection.