University of Mumbai

Development of Software for Digital Pulse Emulator

Submitted in partial fulfilment of requirements

For the degree of

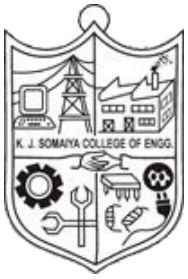
Bachelors in Technology

by

Vishal Kothari
Roll No: 1511061
Muskaan Gupta
Roll No:1611122
Simran Koul
Roll No: 1611127
Ayush Suri
Roll No: 1721021

Guide

Prof- Gopal Sonune



Department of Computer Engineering
K- J- Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch 2016-2020

Certificate

This is to certify that the dissertation report entitled **Development of software for Digital Pulse Emulator** submitted by Mr. Vishal Kothari, Ms. Muskaan Gupta, Ms. Simran Koul and Mr.Ayush Suri in the year 2019-20 under the guidance of Prof. Gopal Sonune of Department of Computer Engineering in partial fulfilment of requirement for the Bachelors in Technology degree in Computer Engineering

_____ _____

Guide Head of the Department

_____

Principal

Date:

Place: Mumbai-77

K- J- Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Development of software for Digital Pulse Emulator** is bona fide record of project work done by Vishal Kothari, Muskaan Gupta, Simran Koul and Ayush Suri This project is approved for the award of Bachelors in Technology Degree in Computer Engineering

_____

Internal Examiner

_____

External Examiner

Date:

Place: Mumbai-77

K- J- Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written thesis submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

_____ **Signature of the Student** _____ **Roll No-**	_____ **Signature of the Student** _____ **Roll No-**
_____ **Signature of the Student** _____ **Roll No-**	_____ **Signature of the Student** _____ **Roll No-**

Date:

Place: Mumbai-77

Abstract

Large scale data acquisition systems are used in big data experiments currently being carried out in International institutes like CERN. Testing of these systems is a challenging task wherein each analog input channel chain has to be tested for integrity of the digitized input data along with ensuring the robustness/correctness of the signal processing algorithms. A digital emulator is an important philosophy which can be utilized in the testing of such systems. In a digital emulator, pulses are generated depending upon the characteristics of the sensor detectors and the process information.

This project would involve development of routines required for generation of digitized pulses given the information pertaining to the process that need to be probed and analysed. Thus, it is proposed to generate pulses from the given probability and amplitude spectrum of the given process.

These pulses could in turn be used for testing data acquisition systems and associated signal processing techniques. The entire project would involve development of associated routines/algorithms in C on a FPGA (Field Programmable Gate Array) based hardware platform.

Keywords– *FPGA, CERN, acquisition system, CERN, CMAC, HDMI, ADC, DAC, High-speed data acquisition system, Pulse generator, Exponential Rise, Exponential Decay, Gaussian distribution, Nanosecond Event rate, Event Capturing based on amplitude-time spectrum, Real time Data Emulation, Digital Signal Processing*

Contents

List of Figures

Nomenclature

CERN	European organization for nuclear research
FPGA	Field Programmable Gate Array
CMAC	Computer Aided Measurement and Control
FMC	FPGA Mezzanine Card
LFSR	Linear Feedback Shift Register
DAC	Digital to Analog Converter
ADC	Analog to Digital Converter
HDMI	High-Definition Multimedia Interface
GCC	GNU Compiler Collection
DSP	Digital Signal Processing

Chapter 1

Introduction

This chapter presents the main concepts that the project is based on. It identifies what the project is actually meant to accomplish. It also describes about our motive to undertake this topic as our motive.

1.1 Problem Definition

- Large scale data acquisition systems are used in big data experiments currently being carried out in International institutes like CERN.
- Testing of these systems is a challenging task wherein each analog input channel chain has to be tested for integrity of the digitized input data along with ensuring the robustness/correctness of the signal processing algorithms.
- A digital emulator is an important philosophy which can be utilized in the testing of such systems.
- In a digital emulator, pulses are generated depending upon the characteristics of the sensor detectors and the process information.
- This project would involve development of routines required for generation of digitized pulses given the information pertaining to the process that need to be probed and analysed.
- Thus, it is proposed to generate pulses from the given probability and amplitude spectrum of the given process.
- These pulses could in turn be used for testing data acquisition systems and associated signal processing techniques. The entire project would involve development of associated routines/algorithms in C on a FPGA (Field Programmable Gate Array) based hardware platform.

1.2 Scope

- Our aim is to design an equipment for testing high speed data acquisition system and large-scale data acquisition.
- A digital emulator is an important philosophy which can be utilized in the testing of high speed data acquisition systems. In a digital emulator, pulses are generated depending upon the characteristics of the sensor detectors and the process information. This project would involve the development of routines required for the generation of digitized pulses given the information pertaining to the process that needs to be probed and analyzed. Thus it is proposed to generate pulses from the given probability.

1.3 Hardware and software requirements

Software tools:

- Xilinx Vivado
- Gcc compiler

Hardware tools:

- Zynq development board with FMC connectors
- High speed data acquisition FMC card with on board two analog inputs and outputs
- Power supply adapter

In this chapter we described the motivation, objectives and the requirements for the project. The shortcomings of conventional system, which our project will be finding solutions to. In the further chapter we will look into various paper solutions already established for our project.

1.4 Organization of the Thesis

Chapter 1 This chapter talks about the overview of the selected project and also defines the motivation behind the project and the scope of the project which is to be implemented.

Chapter 2 This chapter provides literature review done to understand the project.

Chapter 3 The software project management plan (SPMP) for this project defines the project management goals of the project and includes a plan for the project and how you intend to develop the project and a description of the various deliverables and the time for the various tasks involved in the project.

Chapter 4 This chapter contains the functional and non –functional requirement of the project .This document presents an initial description of the various requirements and functionalities of the software.

Chapter 5 This chapter provides the software design document (SDD) which is written in order to give a software development team overall guidance to the architecture of a software project.

Chapter 6 This chapter provides the software test document (STD) which defines the testing strategy that will be used for the proposed system. Objective of this document is mainly to communicate project –wide quality standards and procedures.

Chapter 7 This chapter provides conclusion drawn after working on the project it also provides future scope of the project.

Chapter 2

Literature Survey

Introduction

This chapter presents the papers referenced in the preparation for undertaking this project.These papers serve as a benchmark to enable this project to be undertaken.

Configurable Digital Emulation of Radiation Sources

Compact setup for the generation of pulses emulating radiation sources events is presented.

The core of the system is constituted by a digital configurable device (e.g. a Field Programmable Gate Array - FPGA) that guarantees adaptability to different simulated sources and operative conditions without hardware changes but only updating the firmware.

The emulator has been prototyped and tested

There is no doubt that digital spectrometry has achieved a deep and extended diffusion from nuclear physics to
astronomy, from environmental monitoring to medicine.
The massive evolution of digital processors for radiation measurements has put in evidence the extreme convenience to develop systems for the emulation of radioactive sources. In fact, the absence of radioactive source and detecting apparatus is a fundamental improvement of experimental conditions, which means total health safety of experimenters and possibility to perform remote experiments independently from the real presence of the dangerous source and expensive detection system.

Moreover, the quality of the experiment is positively affected. In fact, the availability of the configurable virtual signal source simplifies testing of processors, allows absolute and fair comparison among different processing techniques, permits to directly evaluate algorithms or adjustment to the processing flow, opens the way to the existence of central banks of experimental data to be accessed remotely.

The processor is based on a spatial computing configurable device (a Field Programmable Gate Array – FPGA) that can be configured for emulation of different kind of source and different detection conditions. The digital output of the FPGA device enters an analog front-end that plays the role of a classic preamplifier. In this way, the system actually can emulate the cascade of source, detector and preamplifier section. From a top -down point of view, the instrument is a generator of current pulses of defined variable amplitudes and occurrence time values that occur according to an assigned statistic distribution.

First, a digital section (FPGA) generates amplitude and occurrence time values of a couple of pulses.

As output of a radiation source, the pulses are generated having amplitudes that are statistically distributed according to the particular emission spectrum. Moreover, the pulse occurrence times are distributed in accordance with Poisson statistic distribution.

For a given radiation source, the configurable device is initialized with the corresponding statistic distributions of amplitudes and occurrence times by means of changeable operative configurations that are contained in a re-programmable memory EPROM.

Second, an analog section generates the output current pulses.

From the FPGA device, this stage is triggered by occurrence time events and is fed by amplitude values that are converted by a digital-to-analog converter (DAC) device.

Pseudorandom generator test

The performances of the instrument mainly depend on a good pseudorandom generation process. In order to test the realized LFSR statistic generator, different realizations of streams of pseudorandom values have been extracted from the FPGA. In order to verify uniformity and independence of the generated values, chi-square tests have been performed.

The results of the tests over the different realizations have all confirmed a confidence level within 5%.

The mechanism generating occurrence time and amplitudes of simulated pulses is the cascade of a generator of uniformly distributed random numbers and a proper transform function. Efficient and accurate algorithms for random number generation [8] are fundamental in many fields of application, from process simulation environments to cryptografy. The main evolution trend of these algorithms is to be more and more fast and suited for implementing into digital configurable devices (i.e. microprocessor and programmable logic) with minimum resource burden.

It has to be pointed out that a pure logic architecture cannot generate a pure random variable but only a pseudorandom sequence, i.e. a pattern of values whose repetition period is extremely greater than the observation time scale. In this sense, the characteristics of the implemented random generator should be: (i) independence among generated events; (ii) repetition period as long as possible; (iii) good statistic of events.

The generation algorithms split up into machine dependent (e.g. Lehmer, multiplication, quadratic, additive, Fibonacci's techniques) and machine independent architectures (e.g. Feedback Shift Register - FSR , Kendall's, Tausworthe's techniques). In the former, the implementing processor word length determines the repetition rate, which makes them suited in case of microprocessor architectures with very wide numeric representation dynamics. In the latter, the repetition period does not depend on the host hardware properties but is function of the algorithm structure that can be made relatively light and suited to be implemented by means of logic operators (e.g. programmable logic devices as FPGA or CPLD).

Learnings from this paper:

In the present application a machine independent algorithm has been implemented and in particular a Linear Feedback Shift Register (LFSR) algorithm [9] that is characterized by arbitrary long repetition periods, excellent statistic properties, high generation speed and limited resource expense. An LFSR is obtained from a shift register [10], i.e. a sequence of flip-flops, by

performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding the resulting signal back into the input of one of the flip-flop. In other words, a *n*-stage LFSR is a polynomial pseudorandom generator that outputs 2^n-1 possible patterns of 1s and 0s, according to the relation

$$(2^n\text{-}1)\cdot T_{clock} = Repetition\ period$$

where T_{clock} is the clocking signal period of the LFSR.

In conclusion, LFSRs can be used to realize extremely good pseudorandom pattern generators that only need the clock as necessary generating signal.

The instrument is able to emulate two different radiation sources at time and provide it on two independent outputs. The two emulation chains are fully independent in all configuration parameters: energy spectra, signal shapes, temporal distributions of the events, noise characteristics, etc. Moreover it is possible to link some parameters or set the instruments in a master/slave mode where the first channel works as a trigger of the second one.

Multichannel Digital Emulator of Radiation Detection Systems:

A digital system for emulating in real time signals from generic setups for radiation detection is presented. The instrument is not a pulse generator of recorded shapes but a synthesizer of random pulses compliant to programmable statistics for energy and occurrence time. Completely programmable procedures for emulation of noise, disturbances and reference level variation can be implemented. The instrument has been realized and fully tested.

The process of debugging of processing systems such as digital pulse processors, pulse discriminators, Time-to-Digital/Amplitude converters, demands an ever-increasing effort as processing algorithms required by the new systems are becoming more complex and therefore more and more critical.

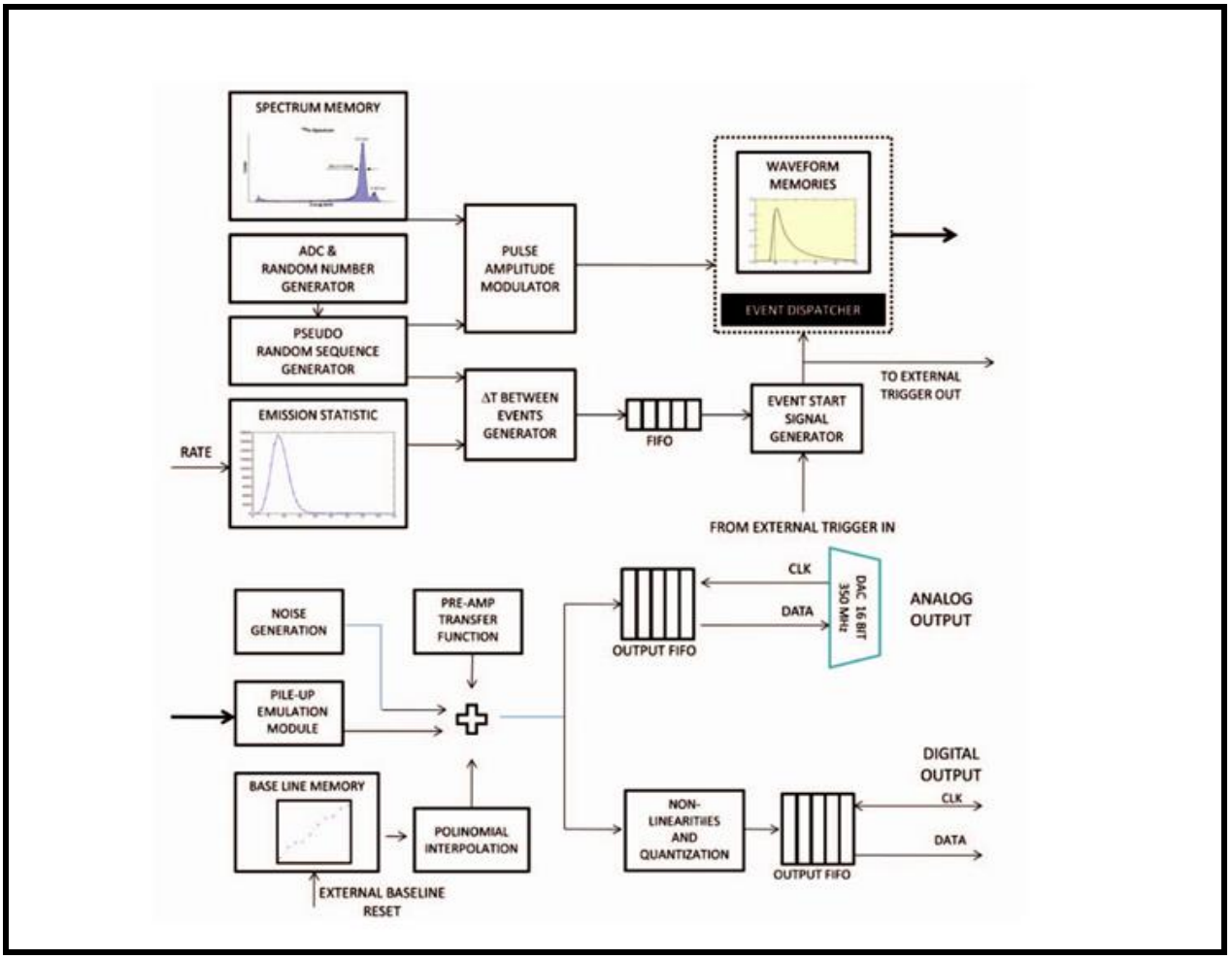
There is growing evidence of the need to generate test vectors that are the most similar as possible to the actual data produced by the experiment, not only in the software of simulation but

especially at the hardware level. This means generate an electrical signal with completely controlled characteristics that is compliant to the real output of a radiation detection setup [1,2].

Although the use of a source and a detector is the best way to generate a reliable data set, it involves considerable disadvantages, especially during the preliminary tests. The use of the source inherently involves a risk for the health of the experimenters, and in addition requires labs equipped in accordance with the regulations in term of use of radioactive substances.

Moreover, the emission spectrum depends on nature of the source, e.g. the polarization of an X-ray tube or the process of decay. The statistical distribution of the events is Poissonian and the only parameter under the control of the experimenter is generally the rate. Also the spectrum of noise, interferences and the pulse shape are issues on which the experimenter can hardly affect. Furthermore, the natural emission process is not repeatable and therefore it is not possible to evaluate the behaviour of different implementations of the processing system on a set of equal data.

It is common practice to use electronic instruments capable of generating analog signals with shapes and characteristics similar to those of the sensors. Since more than forty years there are instruments capable of generating exponential signals with fixed amplitude and Poissonian temporal distribution. These allow a first debugging of systems such as DPP since, for instance, they can emulate effects such as pile up. However they can ’t modulate the amplitudes of generated signals according to a generic spectrum of emission.



Learnings from this paper:

A fully programmable instrument for the generation of signals emulating the output of detection setups is presented. The instrument is a synthesizer of true random pulses compliant to user-defined statistics.

The rise time (10% -90%) of the output analog signal can go down to 3 ns.

The instrument is able to emulate two different radiation sources at time and provide it on two independent outputs. The two emulation chains are fully independent in all configuration parameters: energy spectra, signal shapes, temporal distributions of the events, noise characteristics, etc. Moreover it is possible to link some parameters or set the instruments in a master/slave mode where the first channel works as a trigger of the second on

FPGA-BASED RANDOM PULSE GENERATOR FOR EMULATION OF A NEUTRON DETECTOR SYSTEM IN A NUCLEAR REACTOR

In this work an FPGA-based emulator of a pulse-mode neutron detector system is presented. The equipment emulates the digital output of a discriminator circuit and permits the generation of

pulse trains ranging from 0.5 pulse/s to 1 Mpulse/s. The emulation is based on a synchronous version of a Poisson process generator using Bernoulli trials. The emulator is controlled via a serial connection to a computer and both the pulse-width and the mean pulse-rate can be dynamically updated; thus, making possible the emulation of a detector evolution when a nuclear reactor is in its start-up/shutting-down phase. To also consider the dead-time effects present in pulse-mode detector systems, the equipment implements the paralyzable and nonparalyzable dead-time models. The emulator also incorporates a data-logger module to record interarrival-time values of the generated pulse train. This last feature can be used to calibrate a rate-meter equipment and to verify the statistics of the emulators output signal.

Neutron detectors play a fundamental role in nuclear reactor instrumentation as they take part in reactor control and protection systems. Among radiation detectors, there is a basic distinction in their operating modes: pulse mode, current mode and Campbell mode. In the pulse mode, the instrument is prepared to detect each individual interaction between an incident neutron and the detector, producing a current pulse at the detector output. Following the detector, a series of equipments are chained up; they usually consist of an amplifier, a pulse shaping circuit and a discriminator, in this order. This group of equipments is herein called neutron detector system. The discriminator circuit takes this name as it can filter a neutron interaction from other types of interaction (e.g. with gamma ray radiation). Its output consists of a digital signal with random behavior and a pulse-rate that keeps relation with the detectors neutron interaction rate. The purpose of the random generator presented here is to produce a signal that emulates the discriminator output behavior.

In general, methods for generating random outcomes of a particular distribution start with uniform random numbers. Additionally, a Bernoulli trial is characterized by only one parameter p, which represents the probability to have a success. With these considerations, then the most direct way to obtain a Bernoulli outcome is by simply generating a pseudorandom number R in the range [0, 1] and then comparing it with the parameter p. Finally, if R < p, then a success event is obtained. A straightforward way to implement this approach in a FPGA is by generating an N bits pseudorandom uniform number in the range [0, 2^N-1] and then comparing theoutcome with the value P = round(p·(2^N-1)). The quantity N defines the granularity of the pseudorandom number and

also limits the values that p can assume, discretizing the [0, 1] interval into 2N values. As now p can only adopt discrete values, the same will happen with λ. This introduces a resolution error in the mean pulse-rate obtained.

Chapter 3

Project Design

This chapter contains a fully developed Software Management Plan for the project. The pla highlights the deliverables roles task and schedule for the project.

3.1 Proposed System model

The proposed system would consist of the following:

- A zynq based embedded development board with FMC connectors with on-board SATA interface. This board is a self sufficient computer consisting of required interfaces like HDMI, SATA, PCIe etc. The PC would have an ARM processor with a Xilinx Zynq FPGA

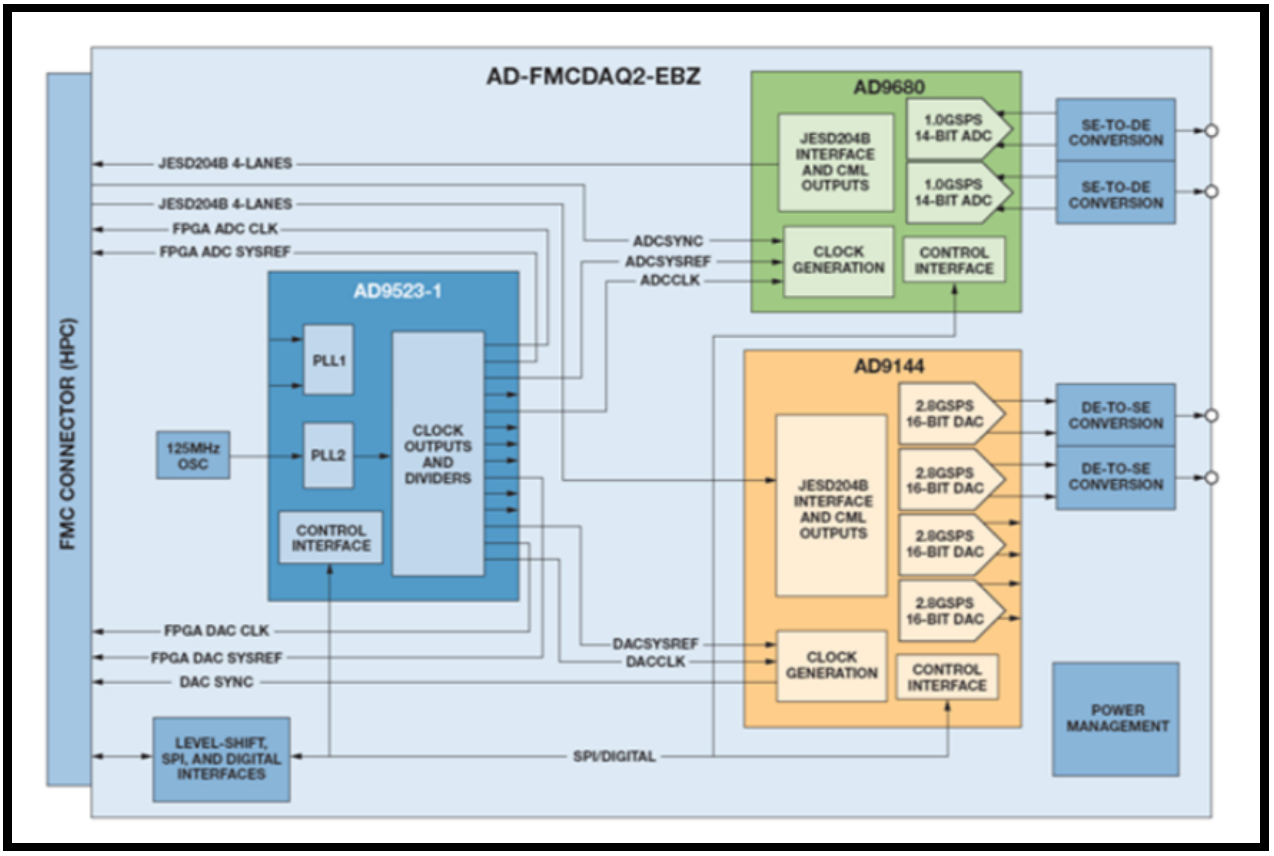


Figure 1 – AD FMCDA (The proposed system)

3.2 PROJECT ORGANIZATION

3.2.1 Software Process Model

The process model which we plan to use for making the project is spiral model.

The spiral model is a risk-driven process model generator for projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

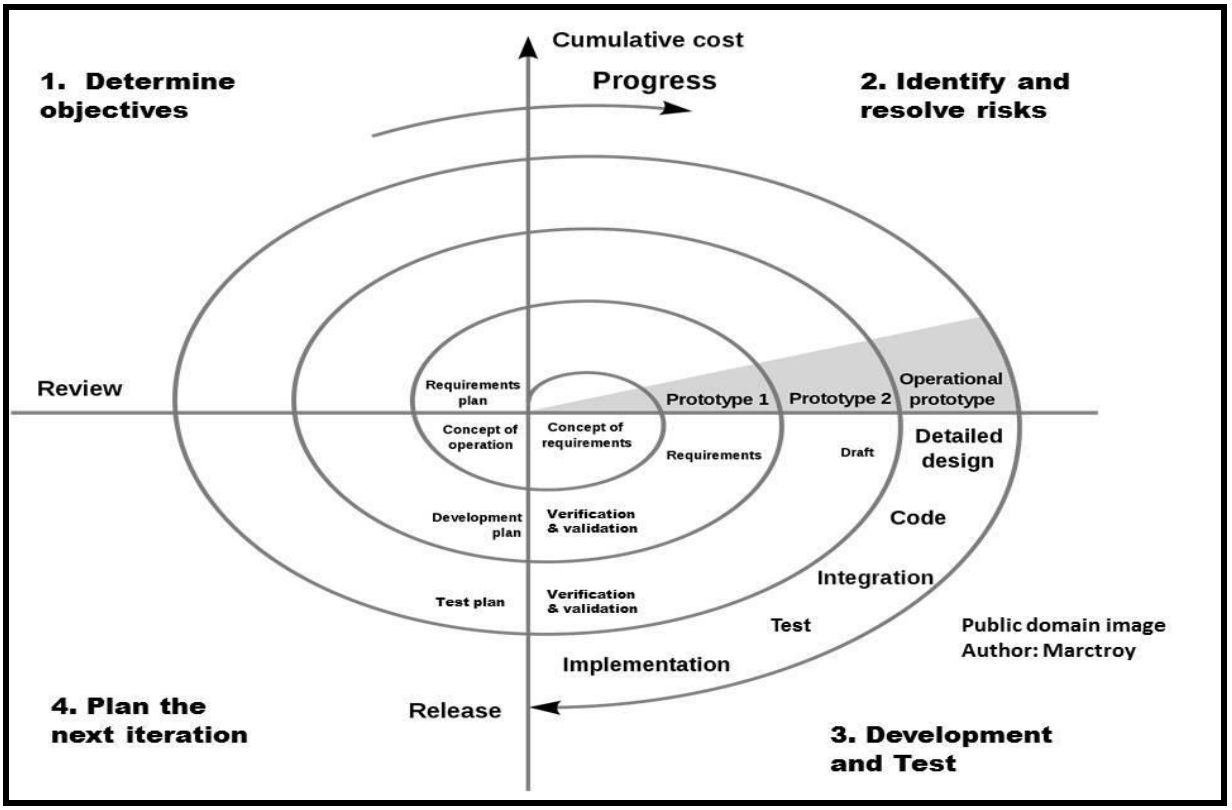


Figure 2 - Spiral Model

1. **Determine objectives, alternatives and constraints:**
    1. System requirements are defined in as much detail as possible and include artefacts for functionality, performance, hardware/software interfaces, key success metrics, etc.
    2. Alternatives such as build vs. buy, can we reuse existing components or do we sub-contract are examined
    3. Constraints such as cost, schedule and interfaces are addressed
2. **Identify and resolve risks, evaluate alternatives:**

1. All possible and available alternatives for developing a cost effective project are evaluated and strategies developed to determinate their use
2. Identify and resolve all the possible risks in the project such as lack of experience, new technology, tight schedules, poor process, etc.
3. Resolve any found risks and uncertainties. Subset reviews may be commissioned to investigate other process models until all high risk situations are resolved

3. Development and test:

Prototype the system from the preliminary design. Follow the usual pattern of create and review design, code, inspect code and test

4. Plan the next iteration:

1. Review the first prototype for strengths, weaknesses, and risks
2. Elicit the requirements for the second prototype

Plan and design the second prototype;

i. Create the project plan
ii. Document the configuration management plan
iii. Construct a test plan
iv. Devise an installation plan

3.2.2 Roles and Responsibilities

Name Of Student	Roles & Responsibilities
Vishal Kothari	Development, Testing
Muskaan Gupta	Development, Documentation
Simran Koul	Development, Documentation
Ayush Suri	Development, Testing

Figure 3 – Roles and Responsibilities

3.3 SOFTWARE REQUIREMENT SPECIFICATION

Purpose

The purpose of the proposed system would involve the development of routines required for the generation of digitized pulses given the information pertaining to the process that needs to be probed and analyzed. The system will be used to generate pulses from the given probability and amplitude spectrum. These pulses could in turn be used for testing data acquisition systems and associated signal processing techniques.

Scope

Large scale data acquisition systems are used in big data experiments currently being carried out in International institutes like CERN. Testing of these systems is a challenging task wherein each analog input channel chain has to be tested for the integrity of the digitized input data along with ensuring the robustness/correctness of the signal processing algorithms.

Overall Description

A digital emulator is an important philosophy which can be utilized in the testing of high-speed data acquisition systems. In a digital emulator, pulses are generated depending upon the characteristics of the sensor detectors and the process information. This project would involve the development of routines required for the generation of digitized pulses given the information pertaining to the process that needs to be probed and analyzed. Thus, it is proposed to generate pulses from the given probability and amplitude spectrum of the given process. These pulses could in turn be used for testing data acquisition systems and associated signal processing techniques. The entire project would involve the development of associated routines/algorithms in C on an FPGA (Field Programmable Gate Array) based hardware platform. As we are focused to test high speed data acquisition systems so giga samples will be given as input. We will be requiring 2000 channels or more to process such kind of data.

Operating environment

Software:

Xilinx Vivado

Gcc compiler

Python compiler

Hardware:

Computer-Aided Measurement And Control (CAMAC) . CAMAC is a standard bus and modular-crate electronics standard for data acquisition and control used in particle detectors for nuclear and particle physics and in industry. The bus allows data exchange between plug-in modules (up to 24 in a single crate) and a crate controller which then interfaces to a PC or to a VME-CAMAC interface.

Zynq development board with FMC connectors.

High speed data acquisition FMC card with on board two analog inputs and outputs.

Power supply adapter.

Front End:

Tinkter:Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Back End:

Intel Core i9-9900KS

Minimum 8 GB RAM

Minimum 2 GB Hard Disk space

Design and Implementation Constraints

The Arm architecture

The ARM architecture, like most 32-bit architectures, is well-suited to a using a C or C++ compiler. The majority of control code is written using high-level programming languages like C and C++ instead of assembly language, so we need to write the code in C Language

Time Constraints

The Program shouldn't take too much time to run the code.

External Interface Requirements

A user interface will be created so that it would be easy for the user to analyze the output. The Tinkter GUI tool is used to provide gui to the user.Buttons are used to let users perform operations such as calculating the fall time, rise time, standard deviation, gaussian curve and the analysis of the summed up pulses, i.e., the pile up phenomenon. We have graphically represented the data so that user can understand the output. We have used functions like Math.plot to represent the data.

Hardware Interfaces

The components used as hardware interface are:

1. Keyboard
2. Mouse
3. Monitor
4. Camac
5.Zynq development board
6 Field Programmable Gate Arrays (FPGAs)
7.IEEE 583

Software Interfaces

1.Tinkter
2.Operating system-Linux
3.Peta-Linux
4.Xilinx Vivado
5. Gcc Compiler

Communication interfaces

To use the System efficiently, the user needs to enter the standard deviation and mean value. After entering the values, user can click on the compile button to compile a C-file on a gcc compiler.The output will be generated using Tinkter tool. User needs to

have gcc compiler, python to run the code and import Tinkter functions from python library.

Specific Requirements

User Interfaces

The end user of the system will be researchers or individuals or groups who want to test high speed data acquisition systems.

The time delay between giving random numbers to the system and getting gaussian curve and other parameters should be minimal.

Software product features

The system is aimed to be simplistic, with minimal complexity and provides ease of use for even beginners. Minimal training will be required to use the system. The GUI provided to the user will be self-explanatory as user will be provided buttons and explanation of the functionality of that particular button

Performance requirements

The data set provided will be very large i.e., hundreds of mega samples or giga sample. As the data set will be very large so the system should be able to process such data and it should not affect the performance. They should be optimized to increase performance of the system.

3.5 PROJECT MANAGEMENT PLAN

3.5.1 Tasks

3.5.1.1 Resource Gathering

3.5.1.1.1 Description

This task involves gathering all the required information required to understand and implement the project. It includes technical papers for reference, gathering medical images for the project etc.

3.5.1.1.2 Deliverables

Technical papers and other information required to understand and implement the project, medical images for the project.

3.5.1.1.3 Resources Needed

Hardware, software and manpower.

3.5.1.2 Prepare Project Scope

3.5.1.2.1 Description

Project scope defines what will or will not be included in the project, and controls what gets added or removed as the project is executed.

3.5.1.2.2 Deliverables

Complete project scope document.

3.5.1.2.3 Resources Needed

Microsoft Word

3.5.1.3 Prepare SRS

3.5.1.3.1 Description

The SRS is a specification of the requirements for the software “product” which will be produced in the project.

3.5.1.3.2 Deliverables

Complete Software Requirements Specification document.

3.5.1.3.3 Resources Needed

3.5.1.4 Prepare SPMP

3.5.1.4.1 Description

This document describes how you intend to develop your software; how you expect to approach the development process; the times and deliverables involved; problems foreseen and constraints imposed. The plan accompanies the project life span and is usually subject to a number of revisions.

3.5.1.4.2 Deliverables

Complete Software Project Management Plan document.

3.3.1.5 Prepare SPMP

3.3.1.5.1 Description

This document describes how you intend to develop your software; how you expect to approach the development process; the times and deliverables involved; problems foreseen and constraints imposed. The plan accompanies the project life span and is usually subject to a number of revisions.

3.5.1.5.2 Deliverables

Complete Software Project Management Plan document.

3.3.1.6 Design

3.3.1.6.1 Description

Project design is the phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out.

3.3.1.6.2 Deliverables

Complete Software Design.

3.3.1.6.3 Resources Needed

Project Scope, SRS, SPMP and project resources.

3.3.1.7 Implementing GUI

3.3.1.7.1 Description

The GUI for the project will be developed based on the design.

3.3.1.7.2 Deliverables

Complete Software GUI.

3.3.1.9 Testing.

3.3.1.9.1 Description

All the components and the functionalities of the project will be tested thoroughly.

3.3.1.9.2 Deliverables

Completely test the project and prepare the test document.

3.3.1.10 Deploy the final project.

3.3.1.10.1 Description

Present the complete project which is properly performing all the functionalities as stated and is completely tested.

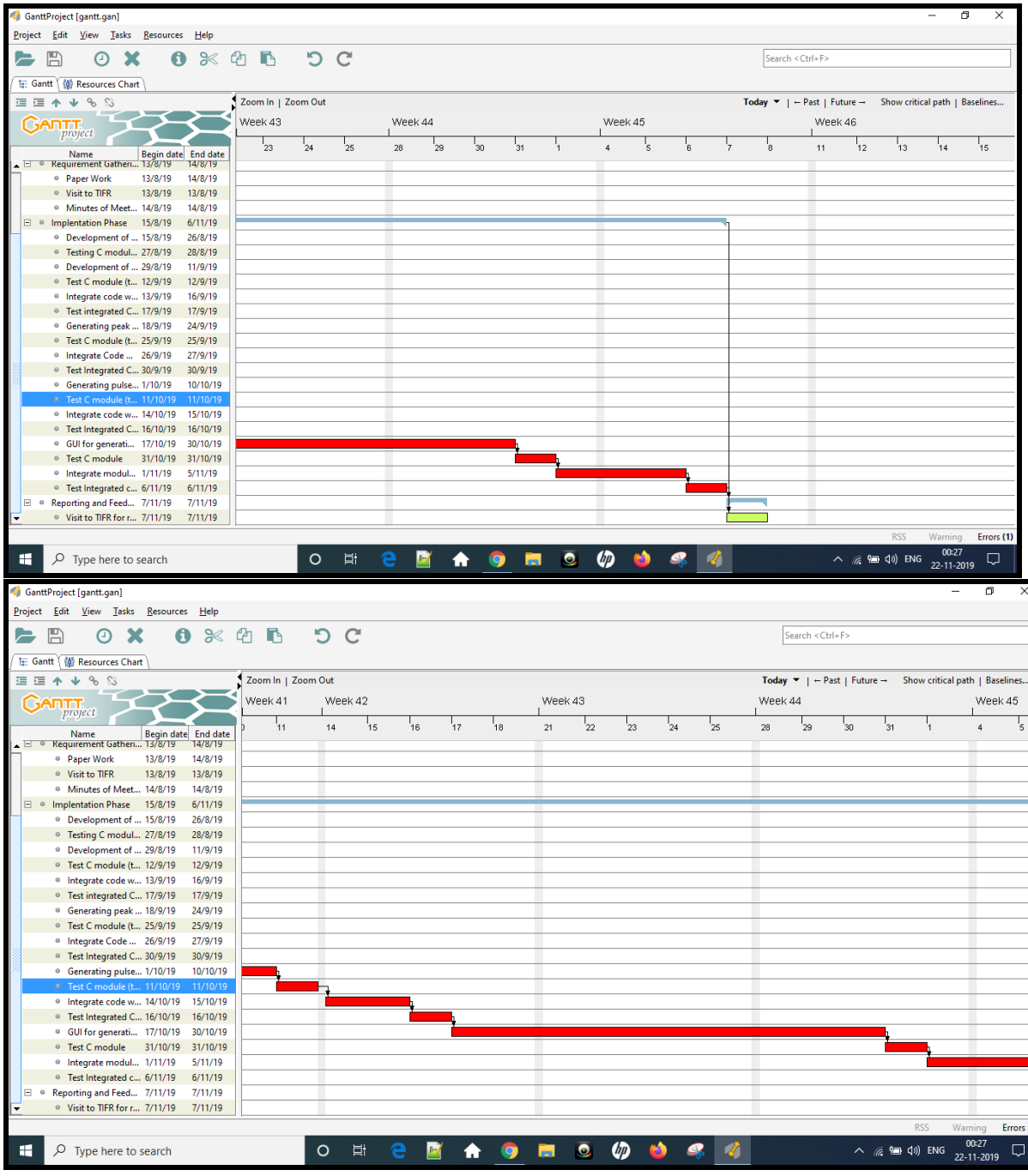
3.3.1.10.2 Deliverables

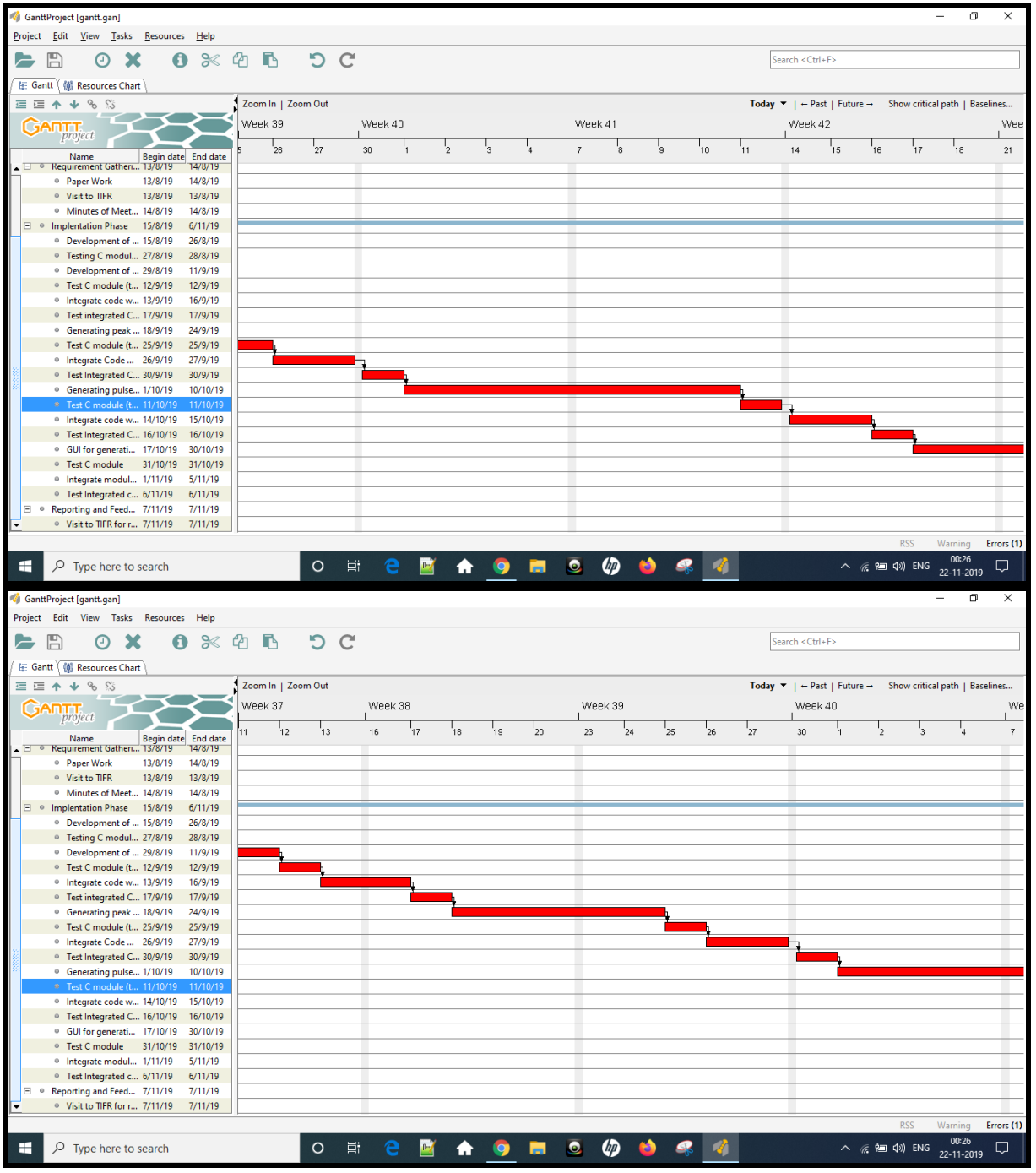
Final project.

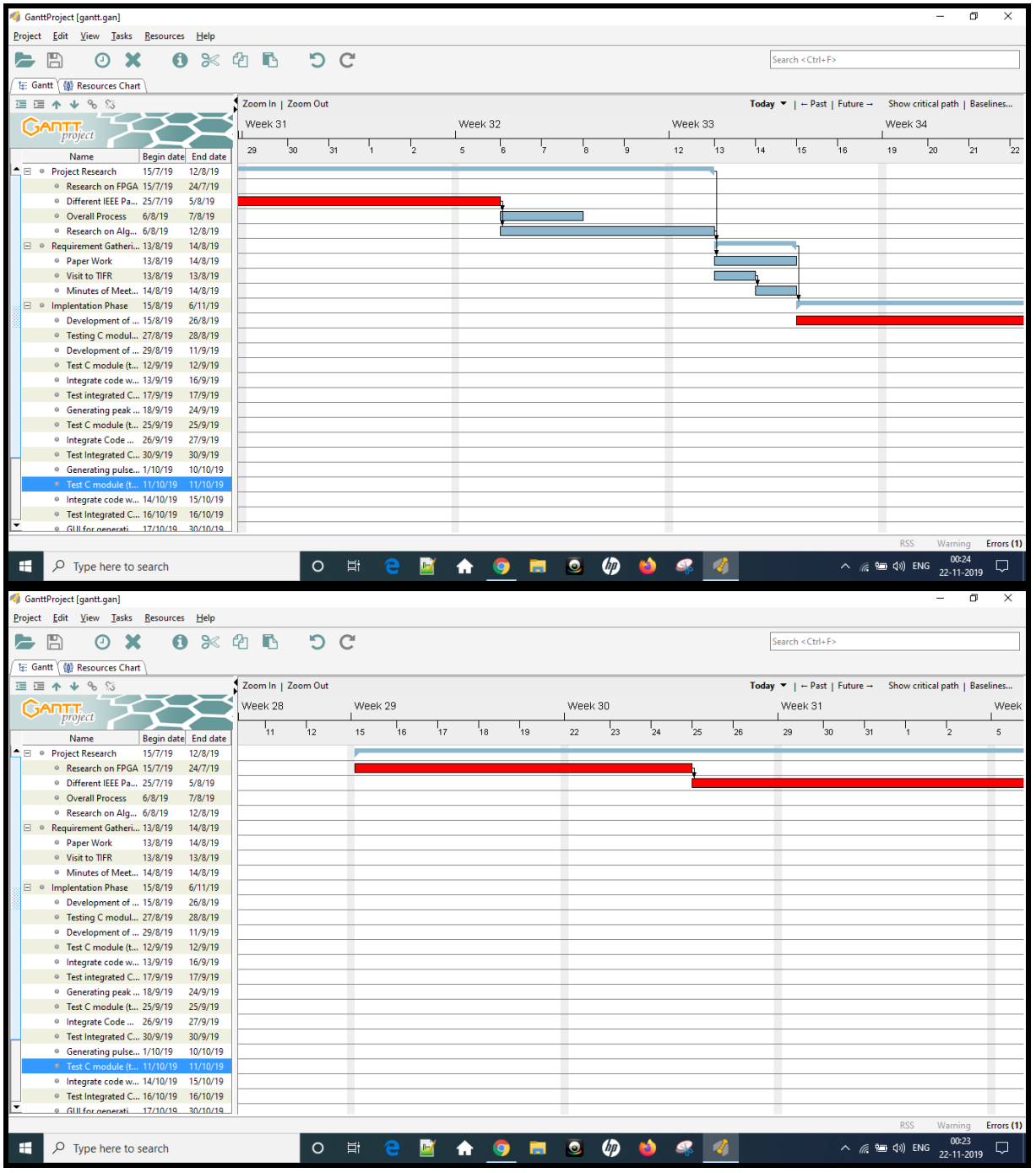
Risks and Contingencies

- Loss of valuable resources required.
- Unable to complete the project or parts of the project within the allotted time.
- Unable to complete all the functionalities mentioned in the scope.

3.5 Gantt Chart:




GanttProject [gantt.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out
Today | ← Past | Future → | Show critical path | Baselines...
Week 39 Week 40 Week 41 Week 42
Name Begin date End date
Requirement Gatheri... 13/8/19 14/8/19
Paper Work 13/8/19 14/8/19
Visit to TIFR 13/8/19 13/8/19
Minutes of Meet... 14/8/19 14/8/19
Implentation Phase 15/8/19 6/11/19
Development of ... 15/8/19 26/8/19
Testing C modul... 27/8/19 28/8/19
Development of ... 29/8/19 11/9/19
Test C module (t... 12/9/19 12/9/19
Integrate code w... 13/9/19 16/9/19
Test integrated C... 17/9/19 17/9/19
Generating peak ... 18/9/19 24/9/19
Test C module (t... 25/9/19 25/9/19
Integrate Code ... 26/9/19 27/9/19
Test Integrated C... 30/9/19 30/9/19
Generating pulse... 1/10/19 10/10/19
Test C module (t... 11/10/19 11/10/19
Integrate code w... 14/10/19 15/10/19
Test Integrated C... 16/10/19 16/10/19
GUI for generati... 17/10/19 30/10/19
Test C module 31/10/19 31/10/19
Integrate modul... 1/11/19 5/11/19
Test Integrated c... 6/11/19 6/11/19
Reporting and Feed... 7/11/19 7/11/19
Visit to TIFR for r... 7/11/19 7/11/19
RSS Warning Errors (1)
Type here to search
ENG 00:26 22-11-2019
GanttProject [gantt.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out
Today | ← Past | Future → | Show critical path | Baselines...
Week 37 Week 38 Week 39 Week 40
Name Begin date End date
Requirement Gatheri... 13/8/19 14/8/19
Paper Work 13/8/19 14/8/19
Visit to TIFR 13/8/19 13/8/19
Minutes of Meet... 14/8/19 14/8/19
Implentation Phase 15/8/19 6/11/19
Development of ... 15/8/19 26/8/19
Testing C modul... 27/8/19 28/8/19
Development of ... 29/8/19 11/9/19
Test C module (t... 12/9/19 12/9/19
Integrate code w... 13/9/19 16/9/19
Test integrated C... 17/9/19 17/9/19
Generating peak ... 18/9/19 24/9/19
Test C module (t... 25/9/19 25/9/19
Integrate Code ... 26/9/19 27/9/19
Test Integrated C... 30/9/19 30/9/19
Generating pulse... 1/10/19 10/10/19
Test C module (t... 11/10/19 11/10/19
Integrate code w... 14/10/19 15/10/19
Test Integrated C... 16/10/19 16/10/19
GUI for generati... 17/10/19 30/10/19
Test C module 31/10/19 31/10/19
Integrate modul... 1/11/19 5/11/19
Test Integrated c... 6/11/19 6/11/19
Reporting and Feed... 7/11/19 7/11/19
Visit to TIFR for r... 7/11/19 7/11/19
RSS Warning Errors (1)
Type here to search
ENG 00:26 22-11-2019


GanttProject [gantt.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out
Today | ← Past | Future → Show critical path | Baselines...
Week 31
Week 32
Week 33
Week 34
Name Begin date End date
Project Research 15/7/19 12/8/19
Research on FPGA 15/7/19 24/7/19
Different IEEE Pa... 25/7/19 5/8/19
Overall Process 6/8/19 7/8/19
Research on Alg... 6/8/19 12/8/19
Requirement Gatheri... 13/8/19 14/8/19
Paper Work 13/8/19 14/8/19
Visit to TIFR 13/8/19 13/8/19
Minutes of Meet... 14/8/19 14/8/19
Implentation Phase 15/8/19 6/11/19
Development of ... 15/8/19 26/8/19
Testing C modul... 27/8/19 28/8/19
Development of ... 29/8/19 11/9/19
Test C module (t... 12/9/19 12/9/19
Integrate code w... 13/9/19 16/9/19
Test integrated C... 17/9/19 17/9/19
Generating peak ... 18/9/19 24/9/19
Test C module (t... 25/9/19 25/9/19
Integrate Code ... 26/9/19 27/9/19
Test Integrated C... 30/9/19 30/9/19
Generating pulse... 1/10/19 10/10/19
Test C module (t... 11/10/19 11/10/19
Integrate code w... 14/10/19 15/10/19
Test Integrated C... 16/10/19 16/10/19
RSS Warning Errors (1)
Type here to search
ENG 00:24 22-11-2019
GanttProject [gantt.gan]
Project Edit View Tasks Resources Help
Search <Ctrl+F>
Gantt Resources Chart
Zoom In | Zoom Out
Today | ← Past | Future → Show critical path | Baselines...
Week 28
Week 29
Week 30
Week 31
Week
Name Begin date End date
Project Research 15/7/19 12/8/19
Research on FPGA 15/7/19 24/7/19
Different IEEE Pa... 25/7/19 5/8/19
Overall Process 6/8/19 7/8/19
Research on Alg... 6/8/19 12/8/19
Requirement Gatheri... 13/8/19 14/8/19
Paper Work 13/8/19 14/8/19
Visit to TIFR 13/8/19 13/8/19
Minutes of Meet... 14/8/19 14/8/19
Implentation Phase 15/8/19 6/11/19
Development of ... 15/8/19 26/8/19
Testing C modul... 27/8/19 28/8/19
Development of ... 29/8/19 11/9/19
Test C module (t... 12/9/19 12/9/19
Integrate code w... 13/9/19 16/9/19
Test integrated C... 17/9/19 17/9/19
Generating peak ... 18/9/19 24/9/19
Test C module (t... 25/9/19 25/9/19
Integrate Code ... 26/9/19 27/9/19
Test Integrated C... 30/9/19 30/9/19
Generating pulse... 1/10/19 10/10/19
Test C module (t... 11/10/19 11/10/19
Integrate code w... 14/10/19 15/10/19
Test Integrated C... 16/10/19 16/10/19
RSS Warning Errors (1)
Type here to search
ENG 00:23 22-11-2019

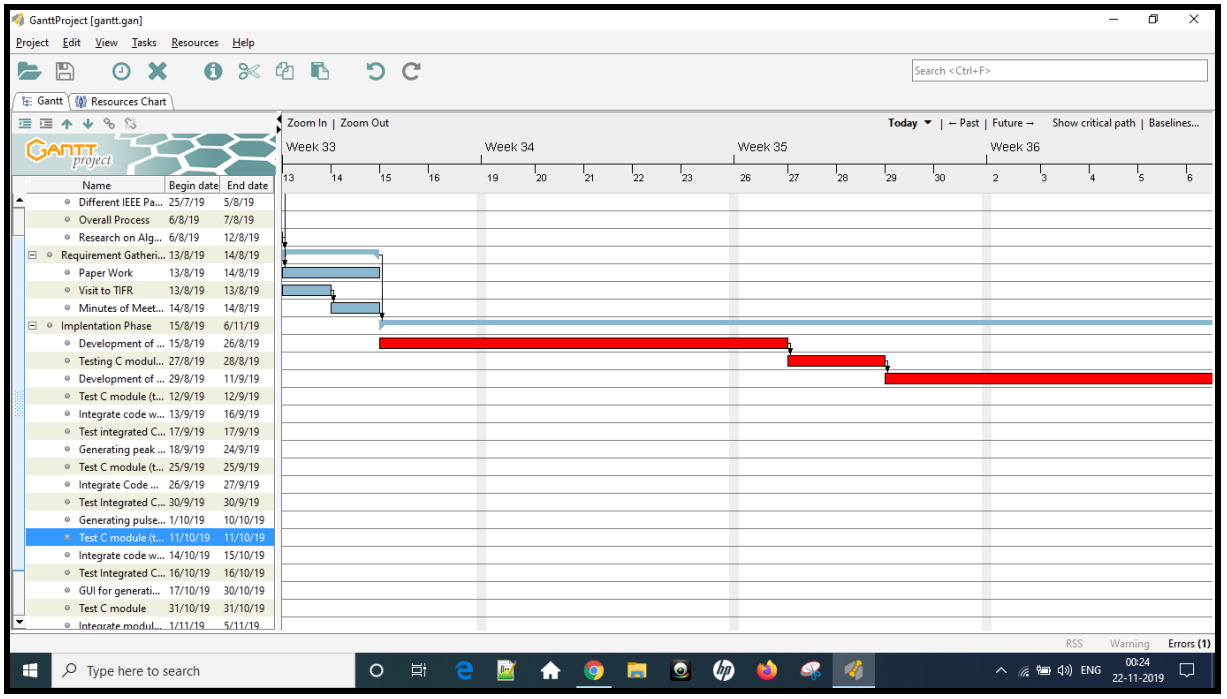


Figure 4: Gantt Chart

3.5 Software Design Document (All applicable UML diagrams)

3.5.1 Use Case Diagram

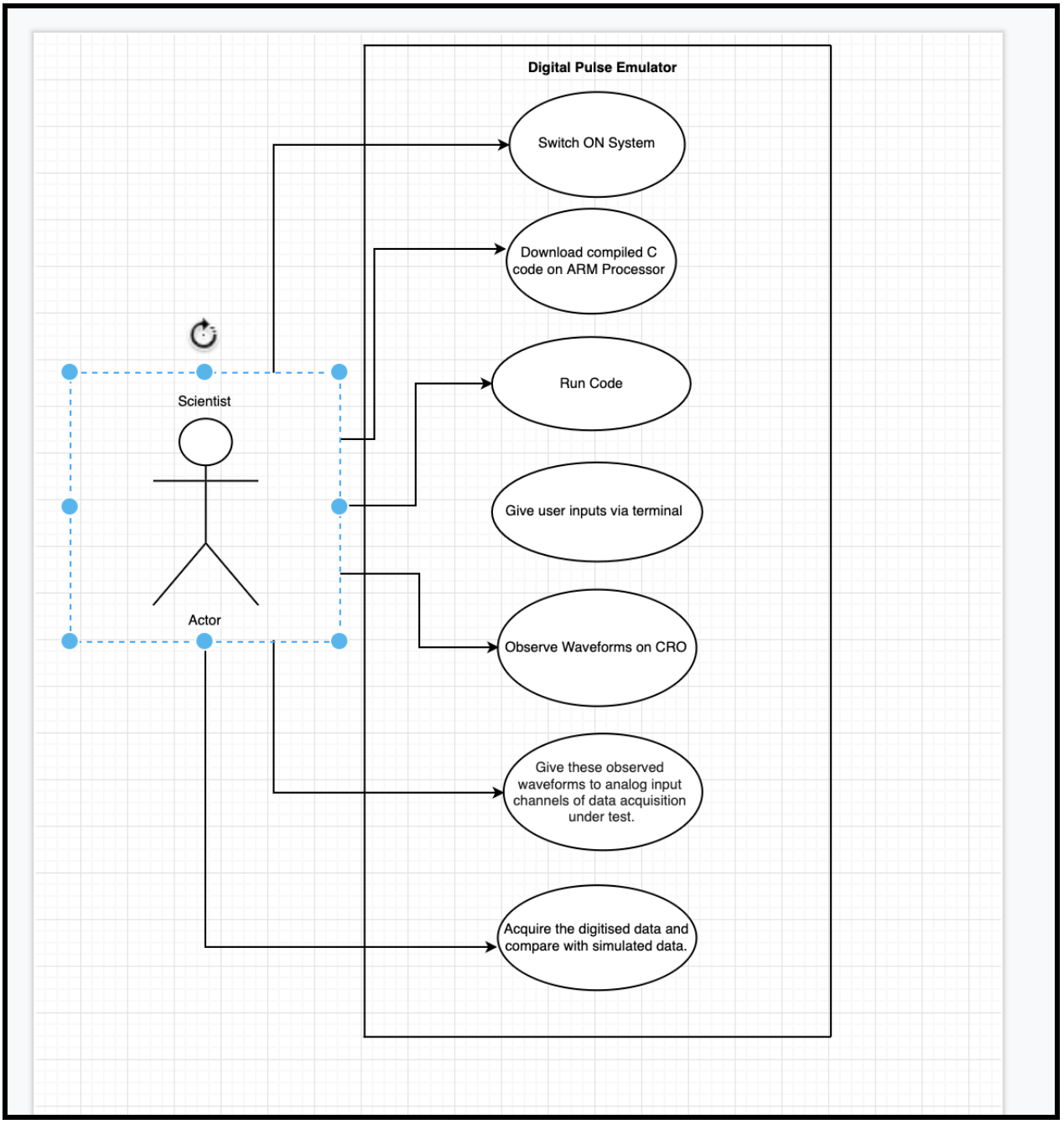


Figure 5 – Use Case Diagram

3.5.2 Activity Diagram

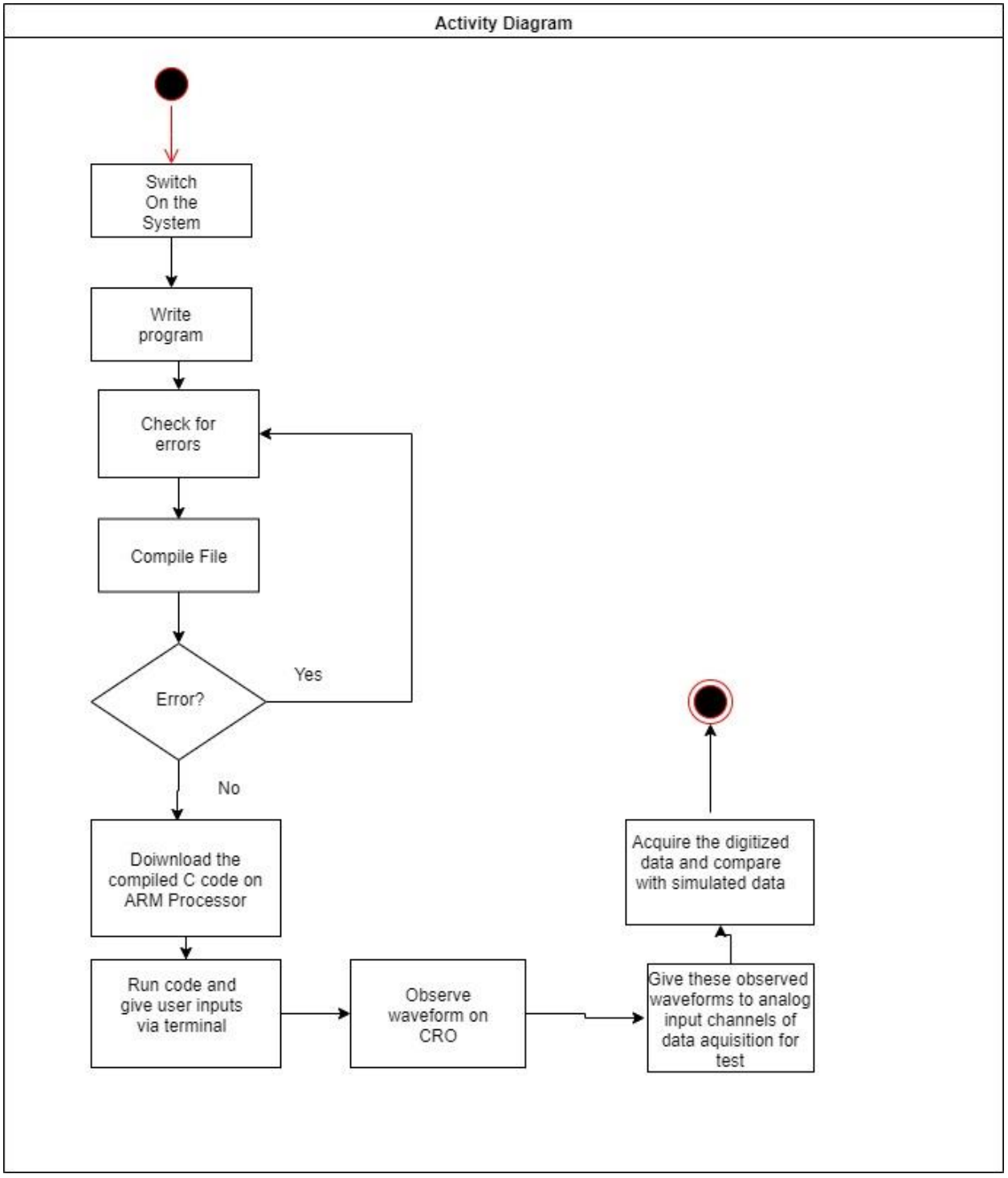


Figure 6 – Activity Diagram

3.5.3 Collaboration Diagram

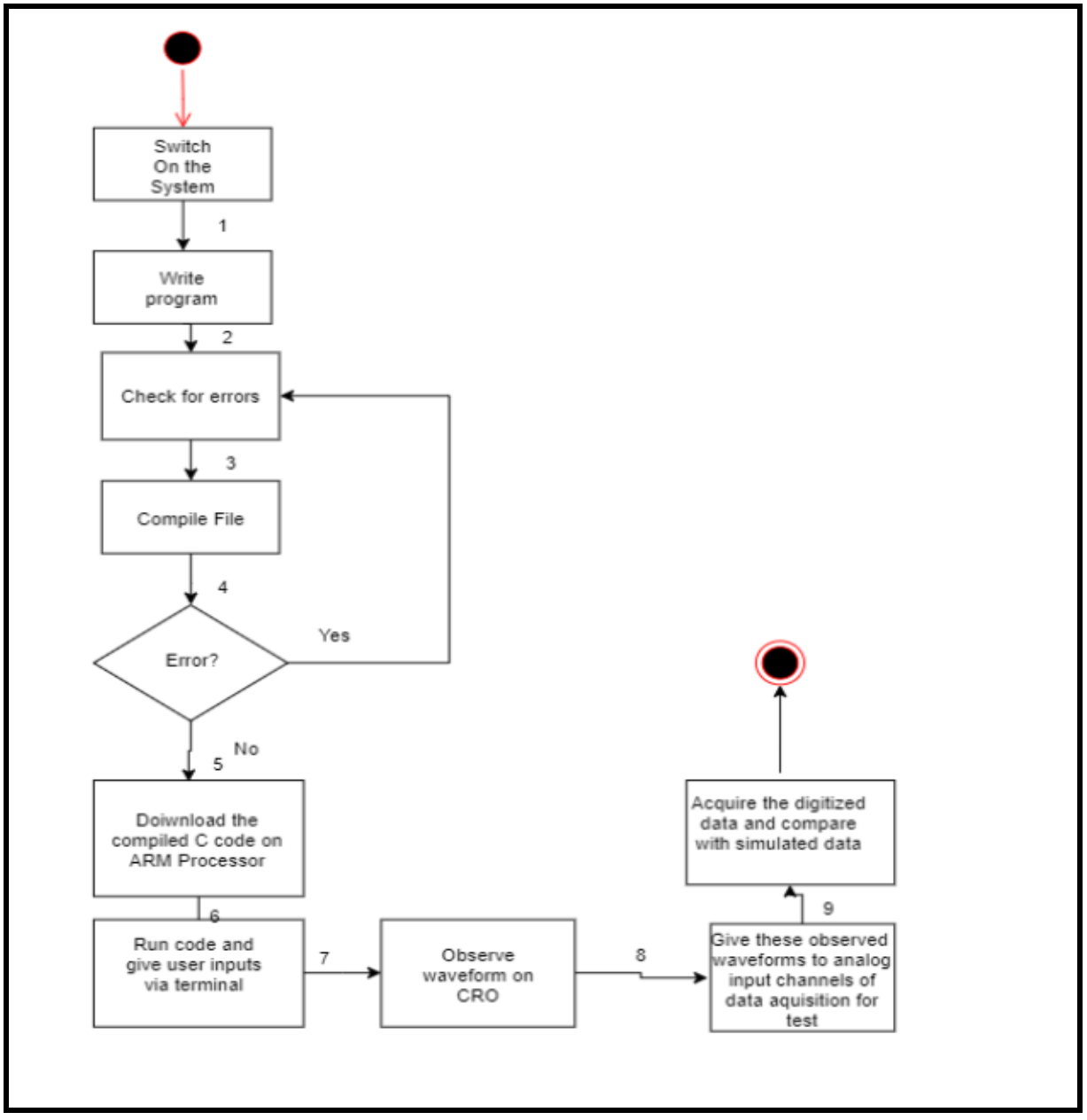


Figure 7 – Collaboration Diagram

3.5.4 Deployment Diagram

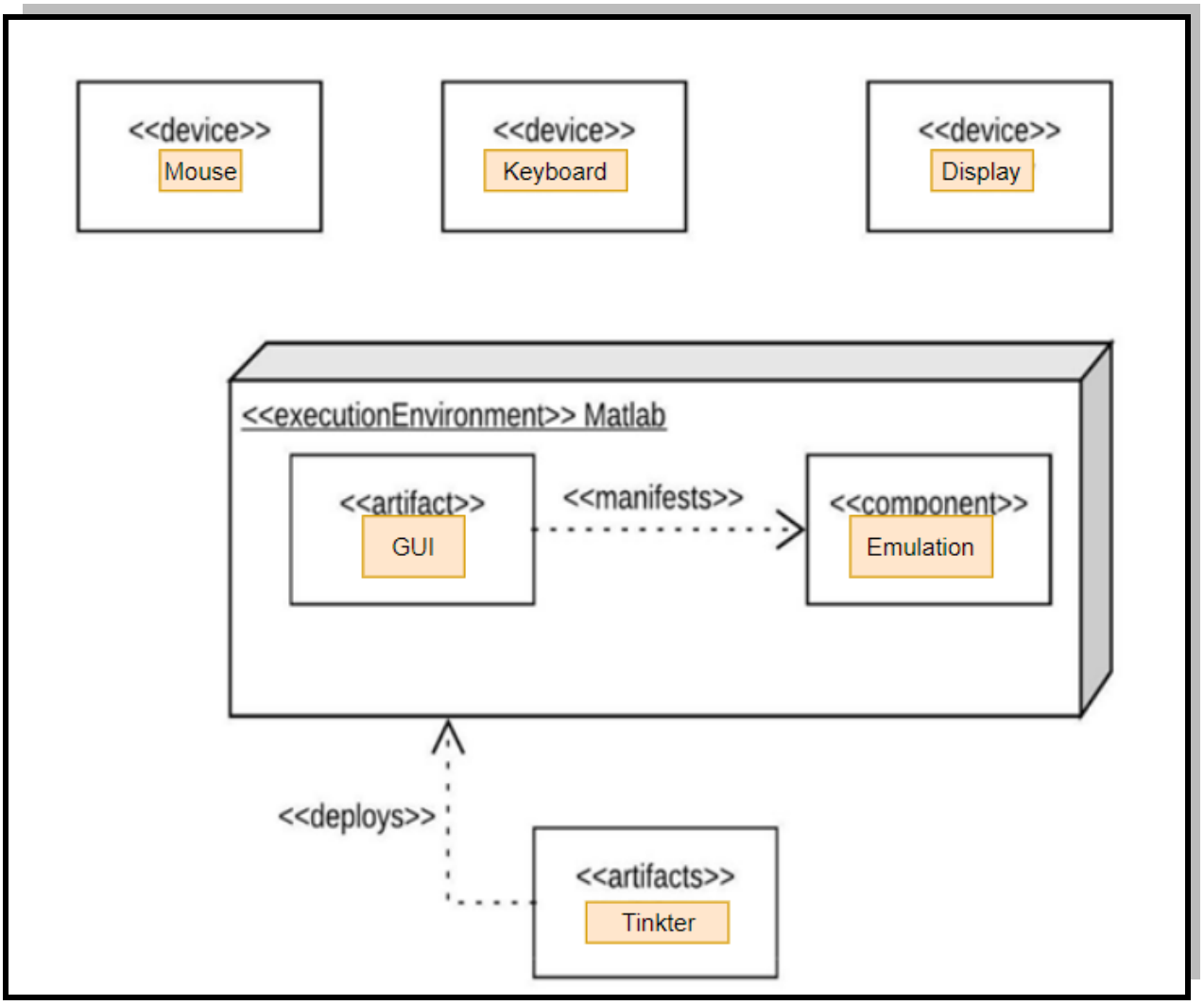


Figure 8 – Deployment Diagram

Thus, in this chapter we had a fare glimpse of the project methodology, software requirements, entire plan and course of duration. Various user interactions with the software were visualized with the help of UML diagrams. The next chapter presents the output of the implemented model and various recommendations from the mentor.

Chapter 4

Implementation

This chapter presents the actual implementation model of the project.Various suggestions and the recommendations of the Guide.

4.1 Prototype model Implementation

1. 2 files event.py and gaussian.c are the main code sources of our project

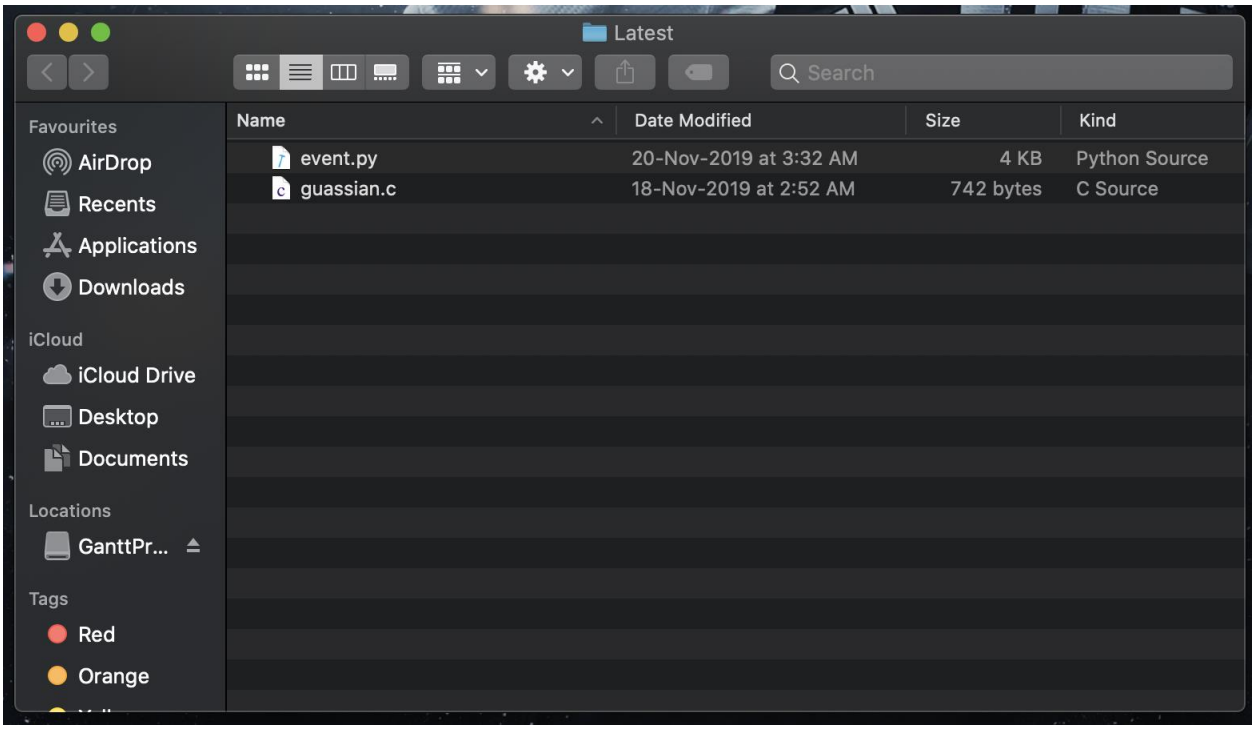
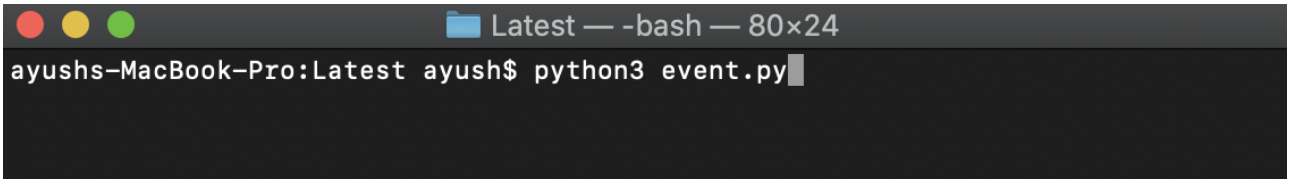


Figure 9- Python3 event.py

2. Run python3 event.py



3. Tkinter GUI is displayed

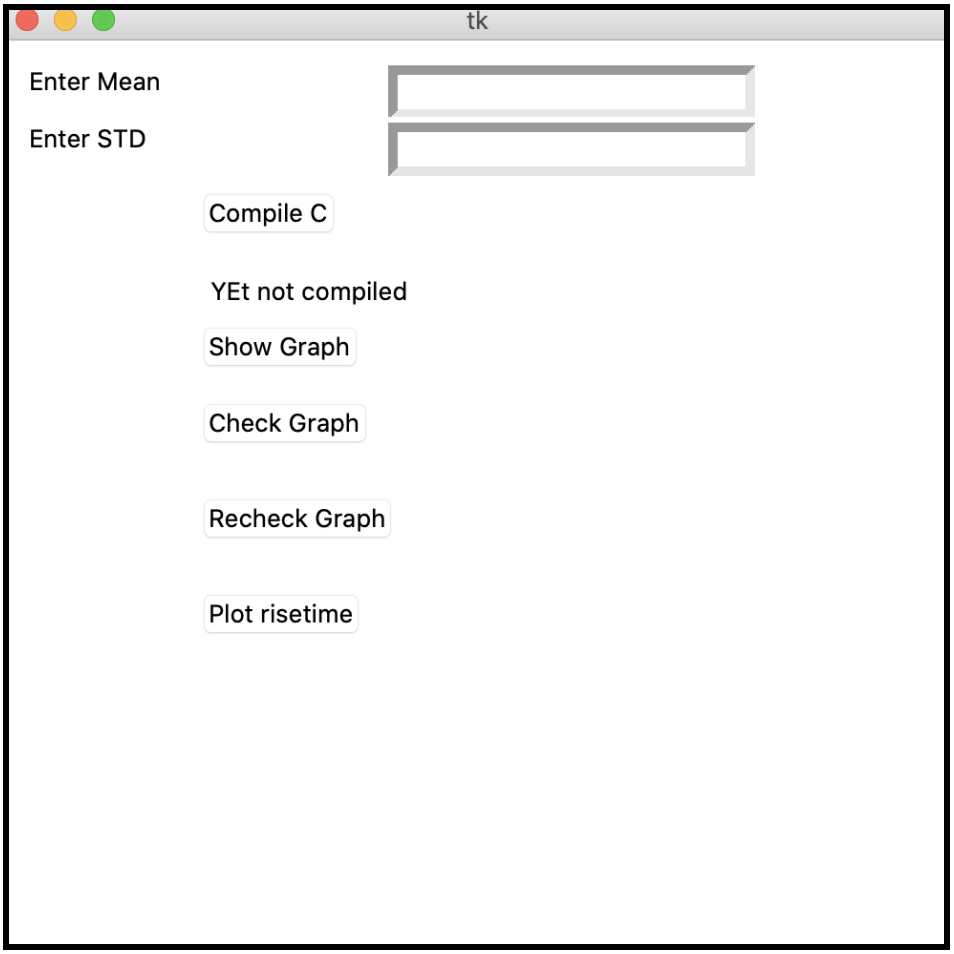


Figure 10- Tkinter GUI Display

4. Give input for Mean and Standard Deviation

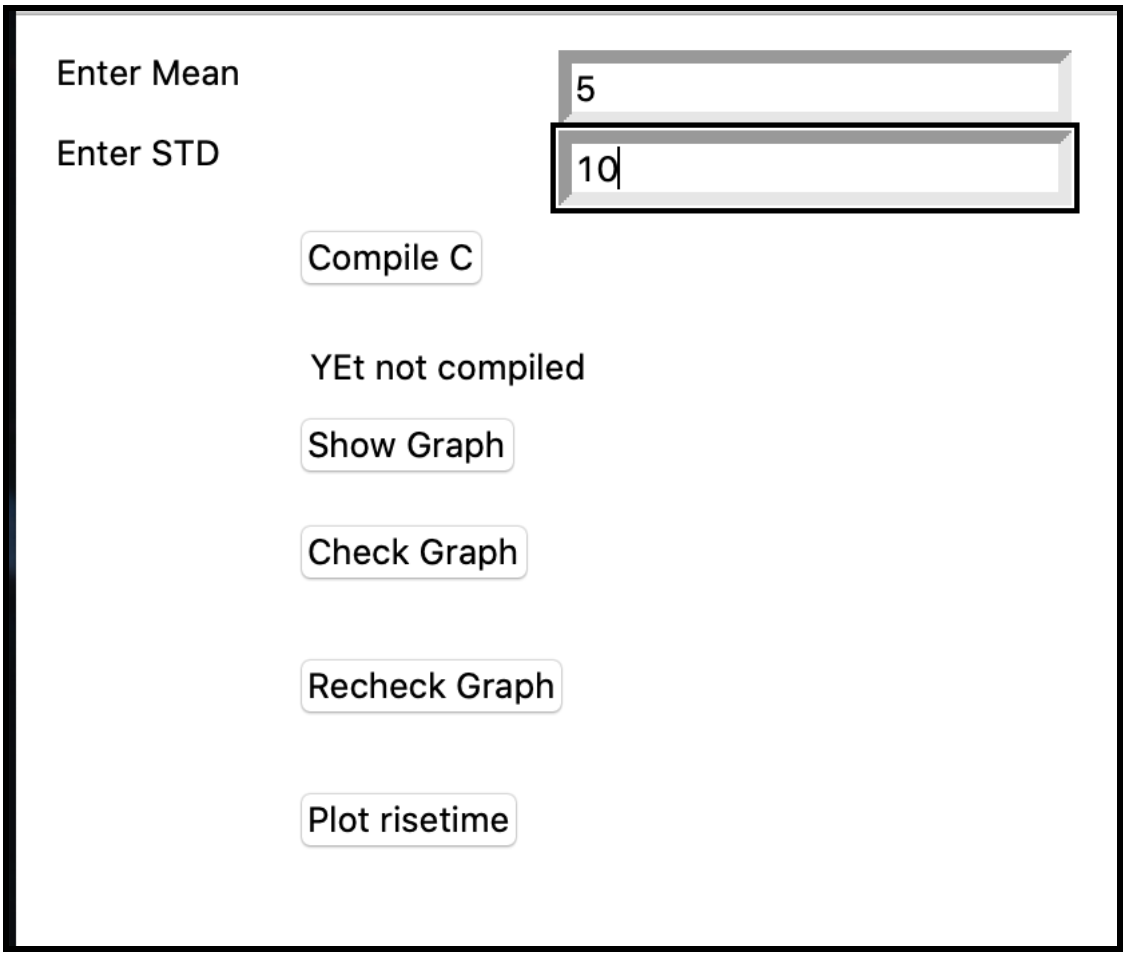
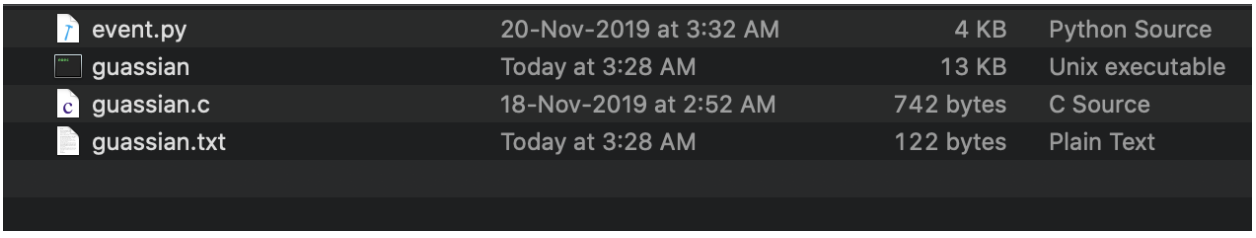


Figure 11- Input for mean and Standard Deviation

5. Press Compile on the GUI and the folder : 2 new files are added w.r.t (1)



Yet not compiled status changed to Compiled

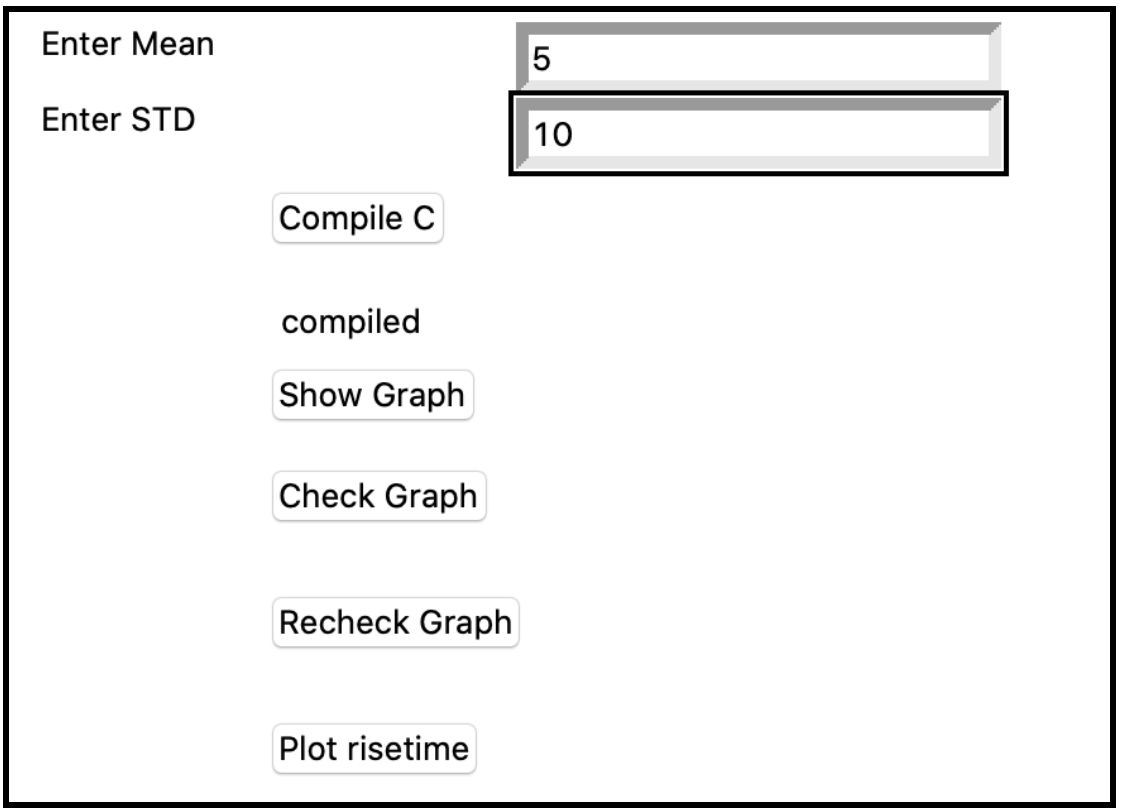


Figure 12-Verification of Graph

6. Click on Show Graph and a graph will be displayed

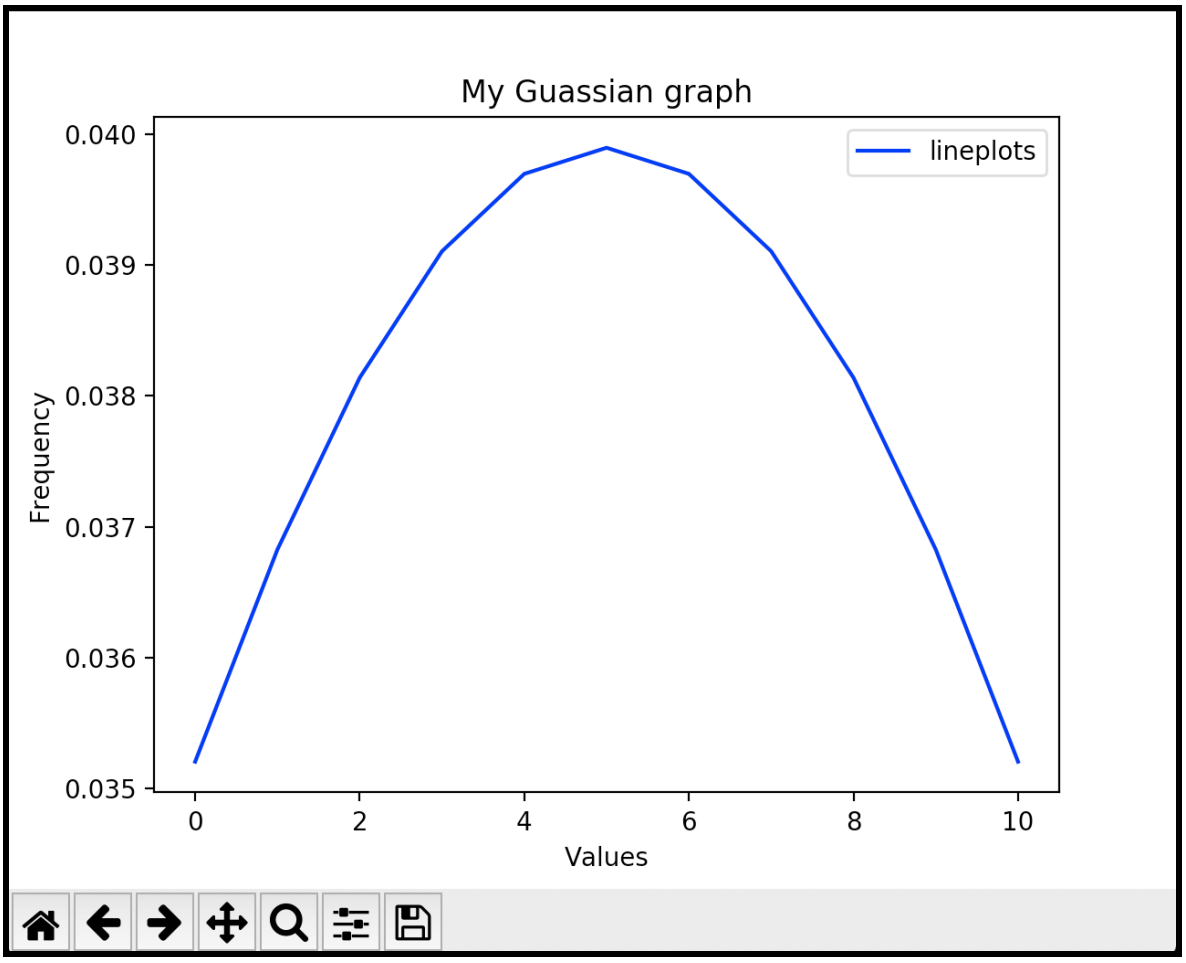
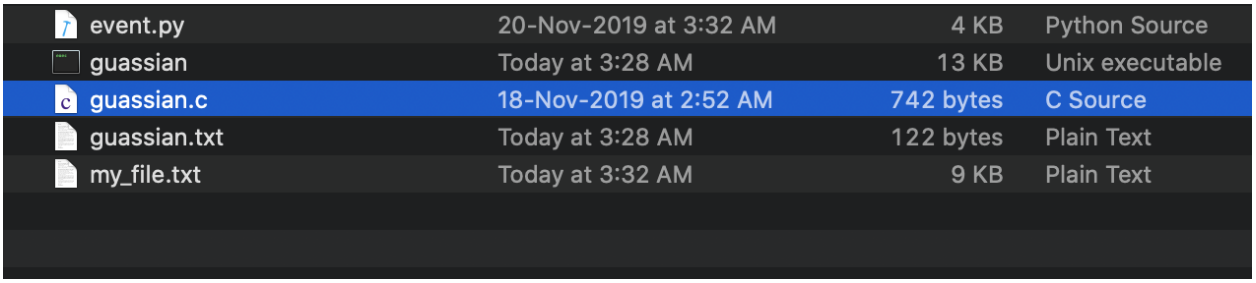


Figure 13- Gaussian Graph 1

7. Check Graph function inserts a new file



8. The pulses generated are shown as follows -

The pulses generated are shown as follows -

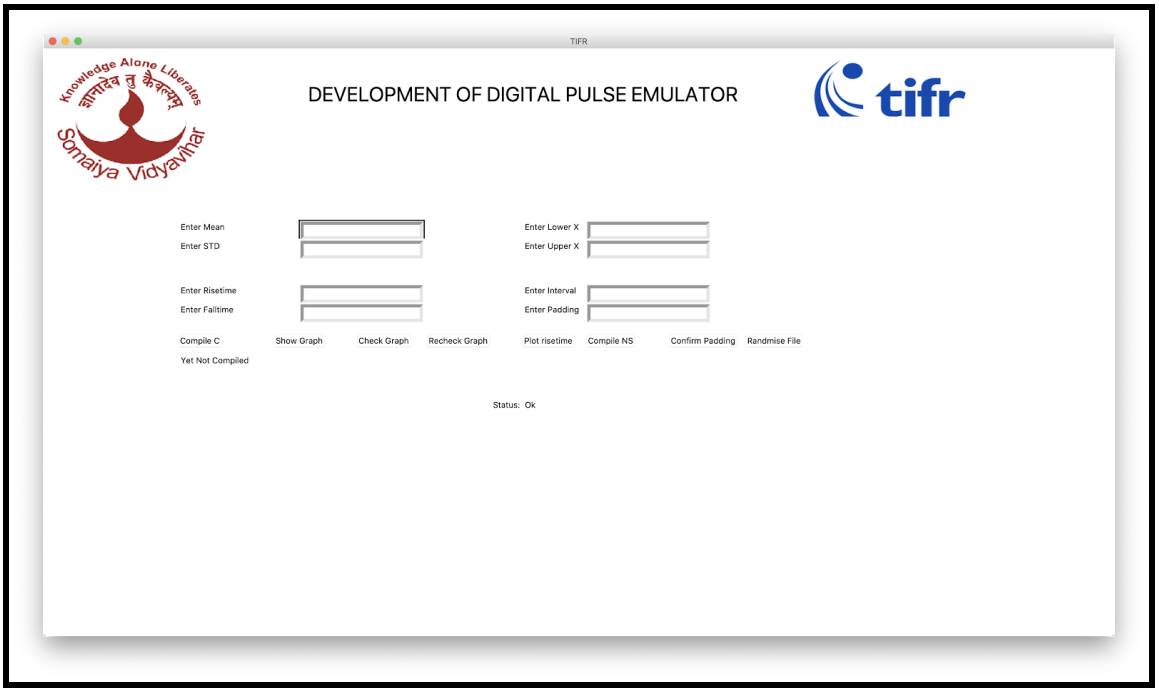


Figure 14- Gaussian Graph 2

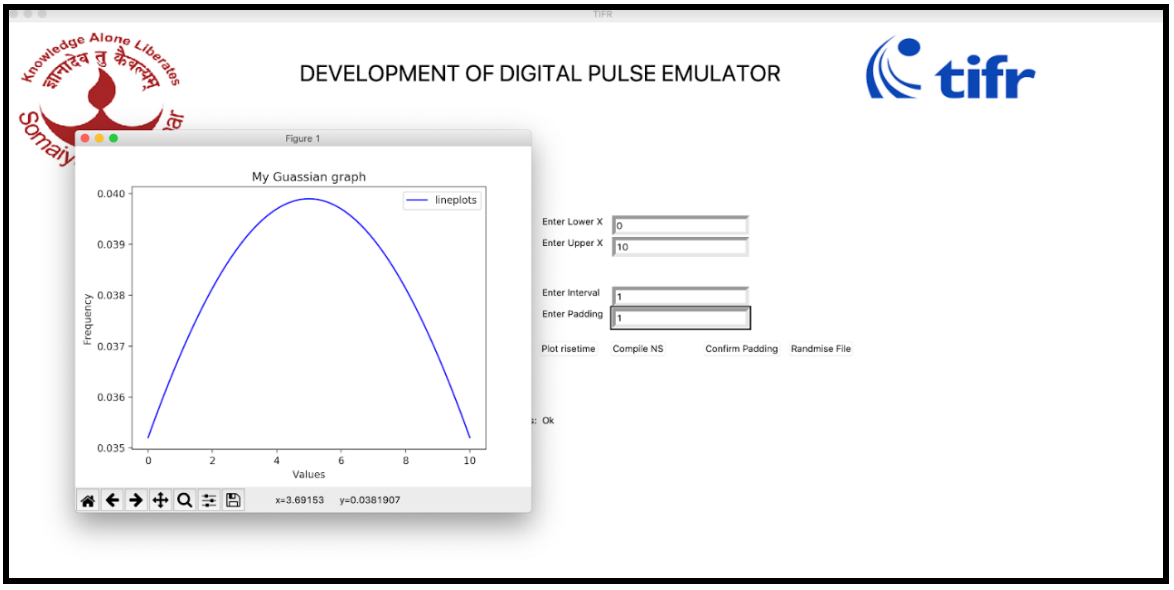


Figure 15- Gaussian Graph 3

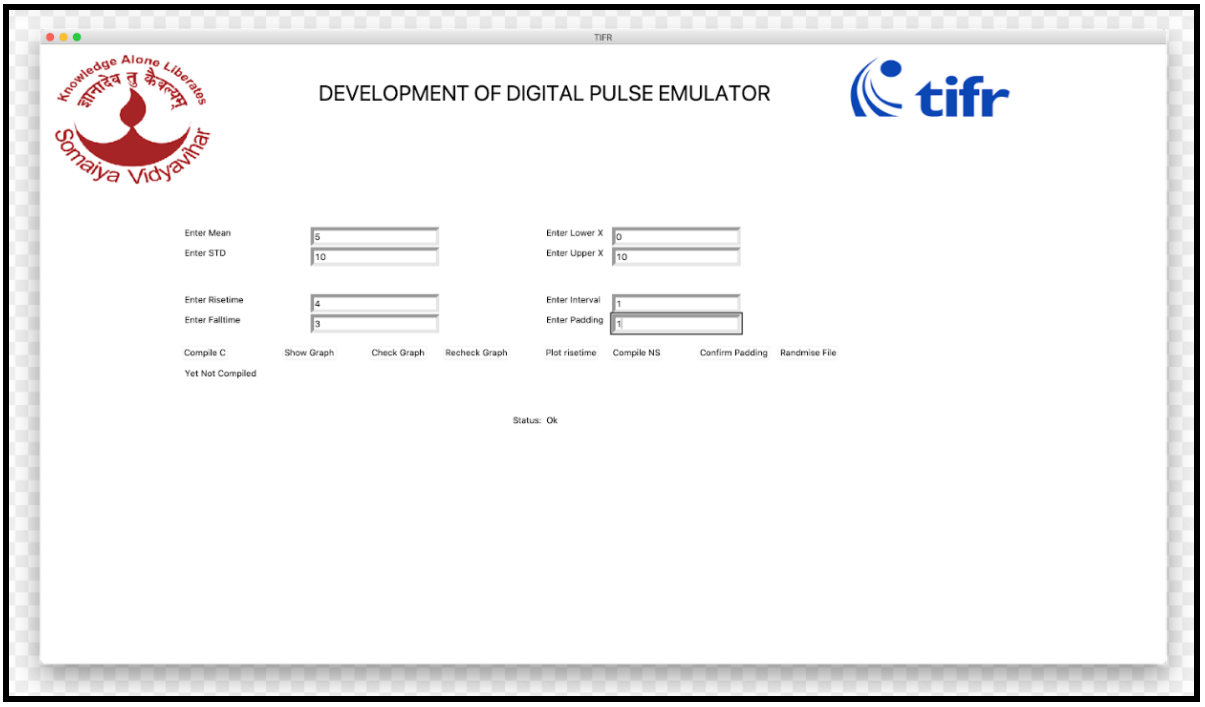


Figure 16- Gaussian Graph 4

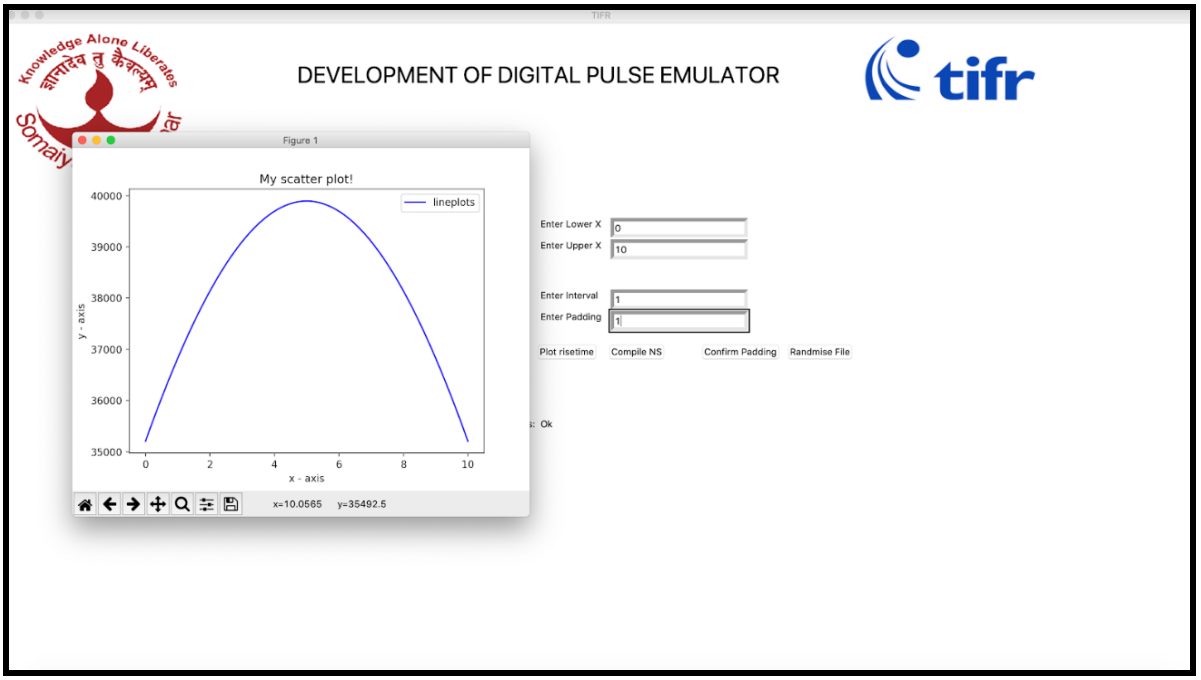


Figure 17- Gaussian Graph 5

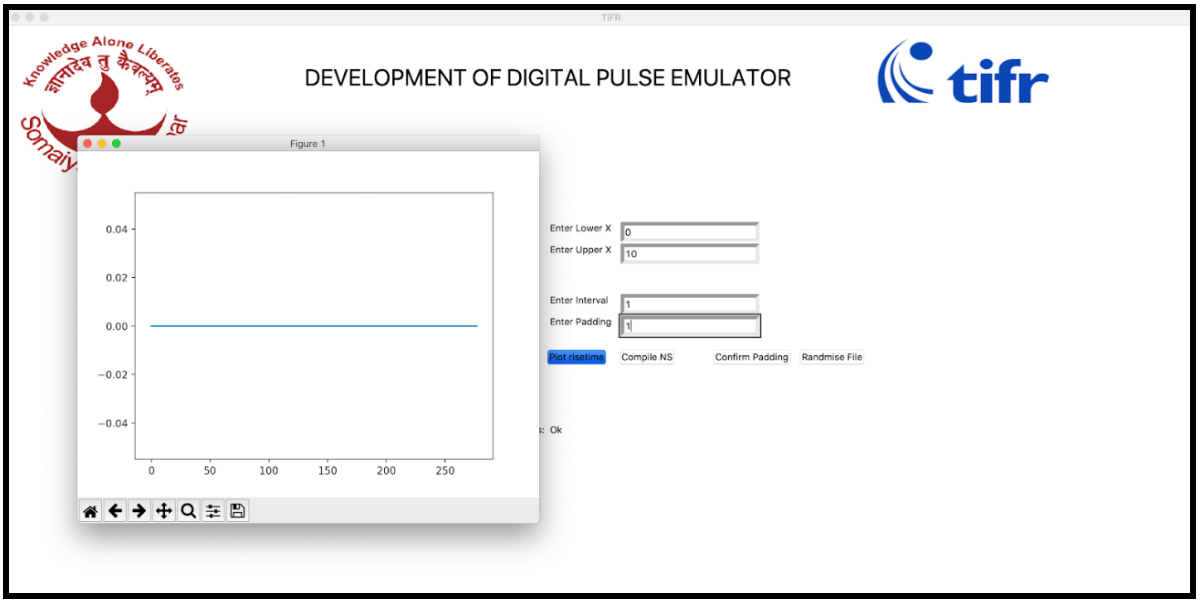


Figure 18- Gaussian Graph 6

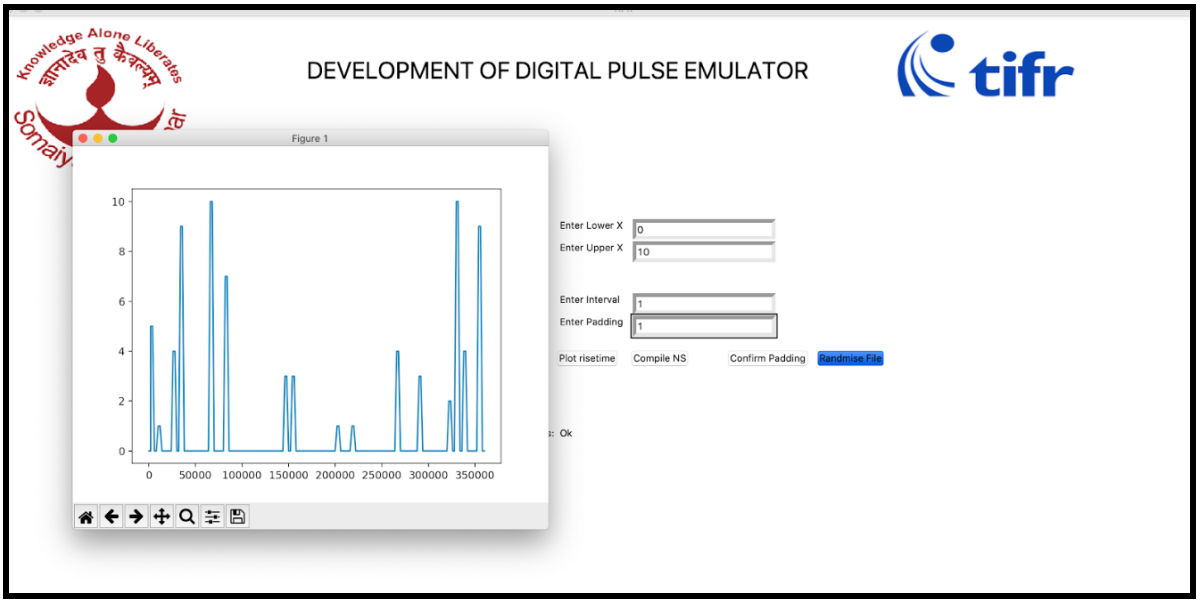


Figure 19- Gaussian Graph 7

4.2 DSP Simulation

In probability theory, the central limit theorem establishes that, in some situations, when independent random variables are added, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed [1].

Similarly in statistics, the importance of Gaussian distribution comes from the central limit theorem. This, in essence, says, if you sum up a lot of independent distributions, the eventual distribution is a Gaussian[2].

Also in probability theory, a normal (or Gaussian or Gauss or Laplace–Gauss) distribution is a type of continuous probability distribution for a real-valued random variable. The general form of its probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The parameter mu is the mean or expectation of the distribution (and also it's median and mode), and sigma is its standard deviation. The variance of the distribution is sigma ^{2}.

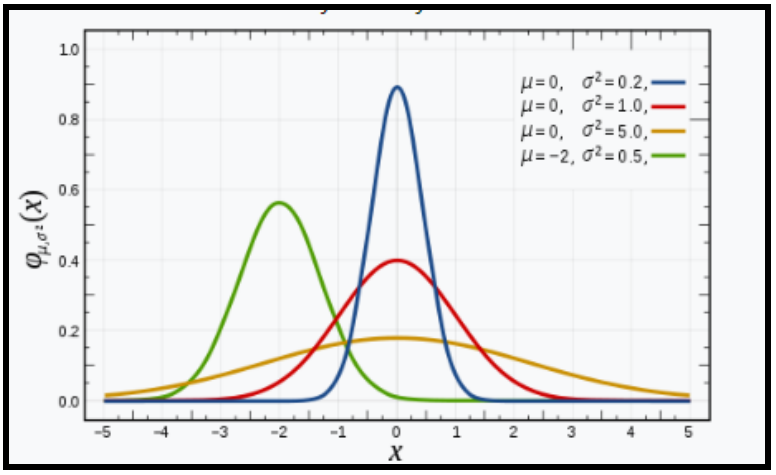


Figure 20-Variance of distribution

4.3 Exponential rise and fall

A random variable with a Gaussian distribution is said to be normally distributed and is called a normal deviate.

In our system, we need to model this behaviour as function. We generate different digital signal algorithms to do so. In mathematical modelling, we choose a familiar general function with properties that suggest that it will model the real-world phenomenon we wish to analyze. In the case of rapid growth, we may choose the exponential growth function:

y=A0e^kt

where A0 is equal to the value at time zero, e is Euler's constant, and k is a positive constant that determines the rate (percentage) of growth. We may use the exponential growth function in applications involving doubling time, the time it takes for a quantity to double.
Such phenomena are seen in real life in wildlife populations, financial investments, biological samples, and natural resources where they may exhibit growth based on doubling time.
On the other hand, if a quantity is falling rapidly toward zero, without ever reaching zero, then we should probably choose the exponential decay model. Again, we have the form y=A0e^−kt where A0 is the starting value, and e is Euler's constant. Now k is a negative constant that determines the rate of decay[3].
We illustrate the phenomenon with the example of an X-ray tube IRF where in a rapid growth and a never ending decay of the event can be observed in figure 2

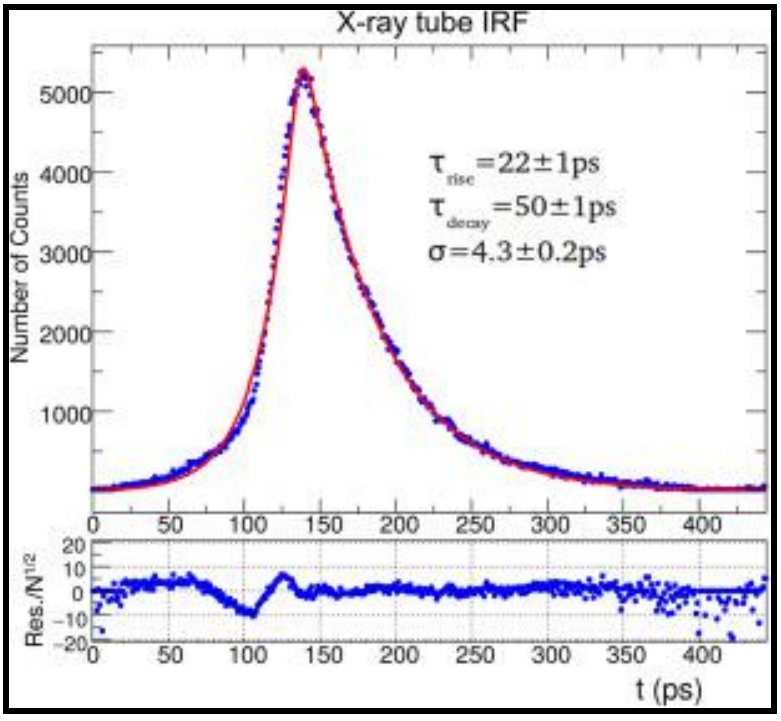


Figure 21- X-ray tube IRF

1. **Simulation**

4.1 Guassian graph plots

Now the system designed generates different in order to understand the flow we generate Gaussian curve and plot it as shown in figure 3 here we illustrate how we apply this above concept of gaussian distribution in similar fashion in the event generator module of the system.

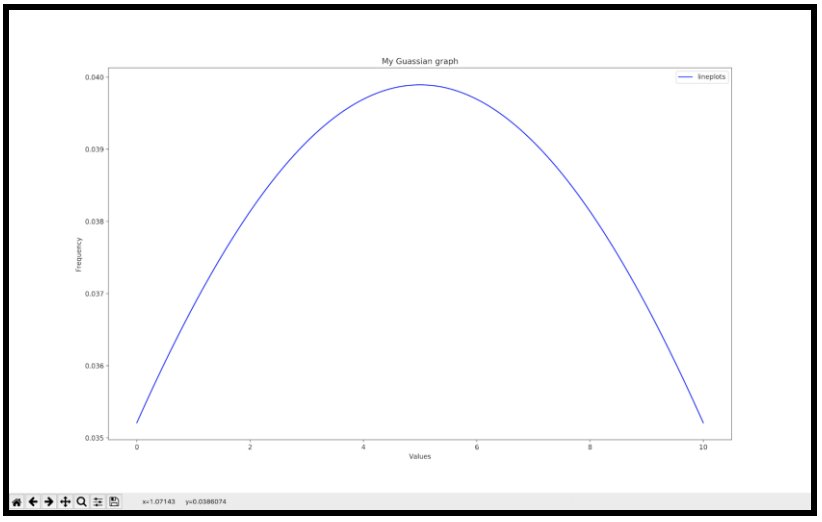


Figure 22- Gaussian Curve of the System plot

We first find out events based on different input parameters provided like mean and standard deviation which generate a Gaussian curve. An event generator generates events based on the input from the Guassian file. Now a reverse process is applied to recalculate the Gaussian graph given the event file of the event. We notice that it still retransforms to a gaussian distribution as in the initial plot as seen in figure 4.

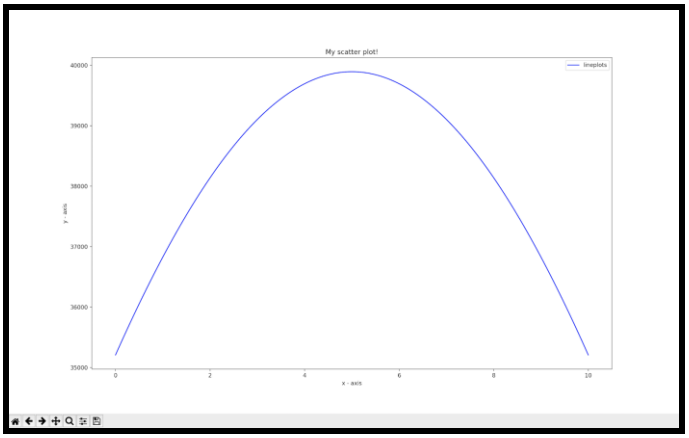


Figure 23- Re-Plot of Events

Pulse Event graph plots

Now step three we generate the plot for the events using plotting tools. The figure described how events are increasing in a linear fashion.

The system uses the random generator to redistribute the events with different time lapse. From this we draw a sawtooth curve events and plot these pulses in the User Interface The figure 6. below shows such pulses drawn. We observe that they have steep rise and fall

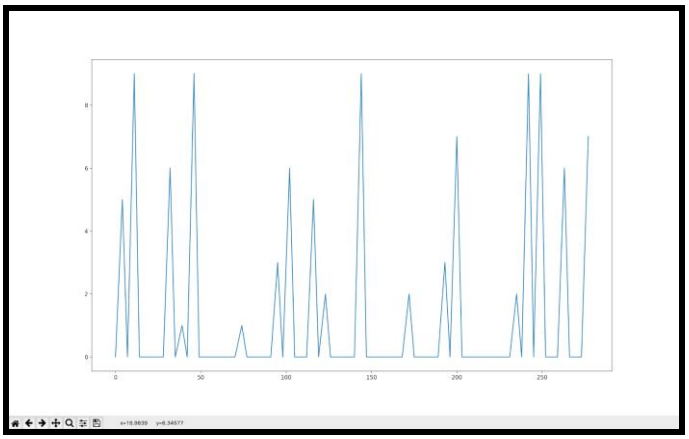


Figure 24-Sawtooth curve of pulses

Finally, as we see in figure 7, we apply the exponential rise and exponential fall to generate a function similar to that in a reactor. Problems like pile up are yet to be addressed.

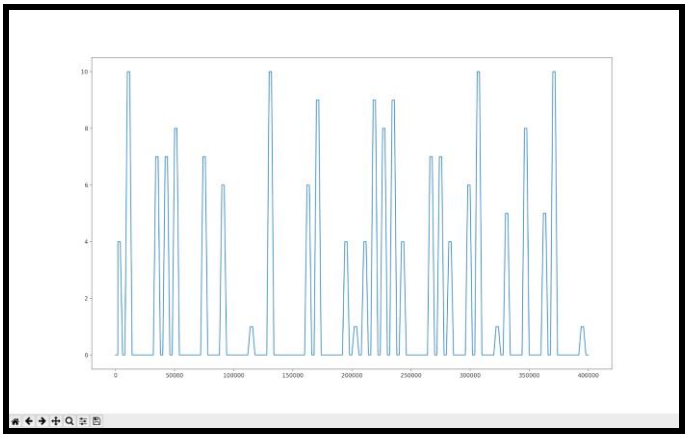


Figure 25-Figure exponential rise and fall of events

In order to understand how events are generated, one can run the application that our group has created on any UNIX or Linux architecture system with different plots for various purposes. And here we are ready with the huge number of rising and falling events.

CHAPTER 5

SOFTWARE TEST DOCUMENT

This chapter includes all the test scenarios such as testing approaches and type of testing used. It also shows you the functionalities which are to be tested and or not.

5.1 System Overview

The aim of this phase is to test the whole project providing coverage of the system, which includes all the main features and some small check parts.

5.2 Test Plan

A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice,and any risks requiring contingency planning. It is a record of the test planning process. The test plan includes:

1) A description of the condition under which the test will run

This project involves the development of routines required for the generation of digitized pulses given the information pertaining to the process that needs to be probed and analyzed. Thus it is proposed to generate pulses with the help of algorithms based on probability and amplitude spectrum. These pulses could, in turn, be used for testing data acquisition systems and associated signal processing techniques. The system will be used to generate pulses from the given probability and amplitude spectrum.

2) A description of the test data to be used.

The Test data would involve

Mean Value

Standard deviation: A quantity expressing by how much the members of a group differ from the mean value for the group.

Rise Time:Rise time refers to the time it takes for the leading edge of a pulse to rise from its minimum to its maximum value. Rise time is typically measured from 10% to 90% of the value.

Fall Time fall time is the measurement of the time it takes for the pulse to move from the highest value to the lowest value.

Padding:Padding is a term used to describe the process of filling a field with pad characters.

Interval: The interval between the events.

3) A description of the expected result.

We first find out events based on different input parameters provided like mean and standard deviation which generate a Gaussian curve. An event generator generates events based on the input from the Guassian file. Now a reverse process is applied to recalculate the Gaussian graph given the event file of the event.The system uses the random generator to redistribute the events with different time lapse. From this we draw a sawtooth curve event. From this curve we will get a graph that will have steep rise and fall and we apply the exponential rise and exponential fall to generate a function similar to that in a reactor.

5.3 Features to Be Tested

- Matching of gaussian curves(Emulation validation)
- Performance optimization
- Memory optimization

5.4 Features Not To Be Tested

Some areas in the project low priority modules and low risk areas that features are not tested and specified in the test plan. If the feature or the module has been used before and was considered stable then we can skip testing. The basic aim of this is reduced effort and time.

5.5 Testing Environment

Software Tools:

Xilinx Vivado

Gcc compiler

Hardware Tools:

Zynq development board with FMC connectors

High speed data acquisition FMC card with on board two analog inputs and outputs

Power supply adapter

5.6 Test Cases

A test case signification refines the test approach and identifies the feature to be covered by the particular case. It also identifies the procedures required to accomplish the testing and specifies the pass/fail criteria.It also documents the actual values used for the input along with the anticipated outputs.

5.6.1 White box testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

```
import unittest
```


```
import unittest
```

```
class TestDetection(unittest.TestCase):
        def test_detection(self):
                fileName = 'myfile.pdf'
                print("File Extension:",fileName[-3:])
                self.assertEqual(fileName[-3:], 'txt',f'File type is {fileName[-3:]} rather than
txt')

if __name__ == '__main__':
        unittest.main()
```

Testing with valid input: my_file.txt

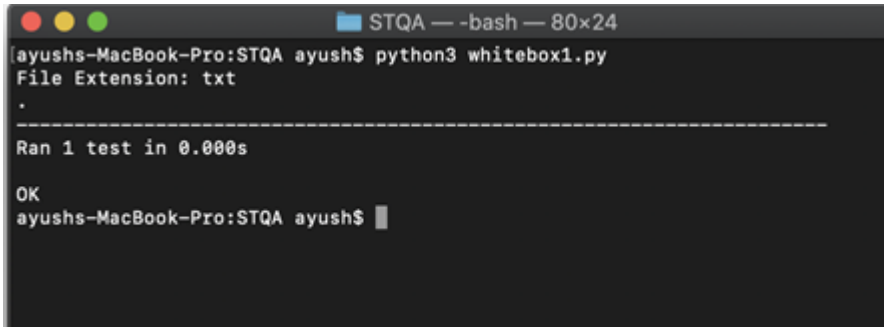


Figure 26- Testing environment 1

Testing with invalid input: myfile.pdf

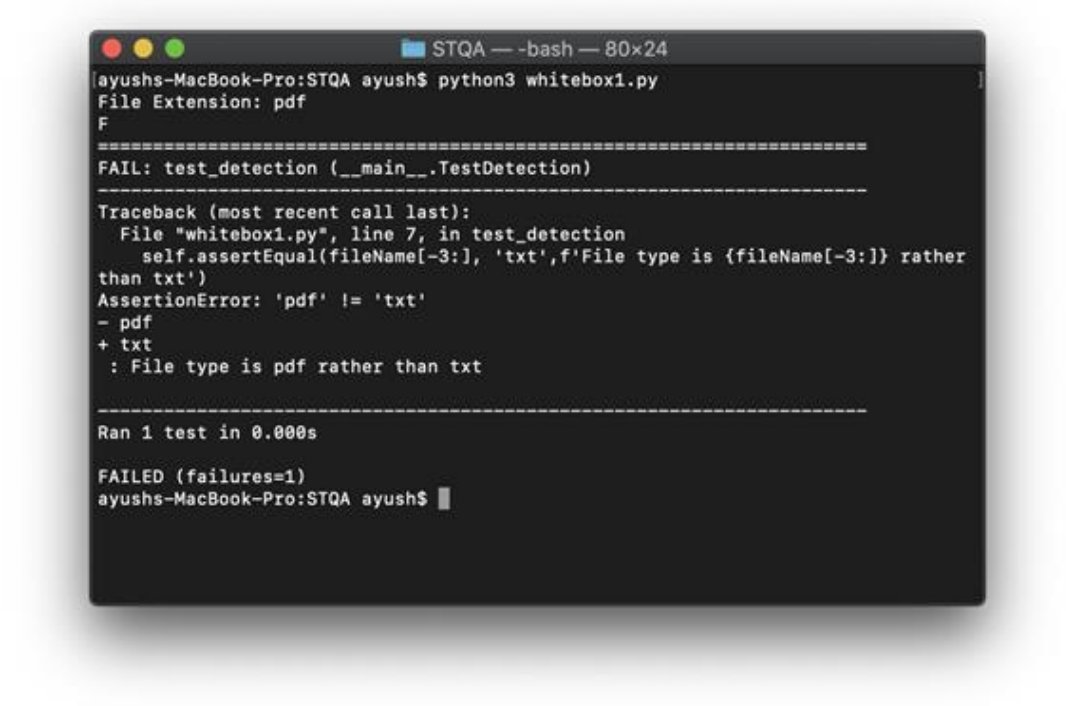


Figure 27- Testing environment 2

Testing with invalid input: tifrlogo.png

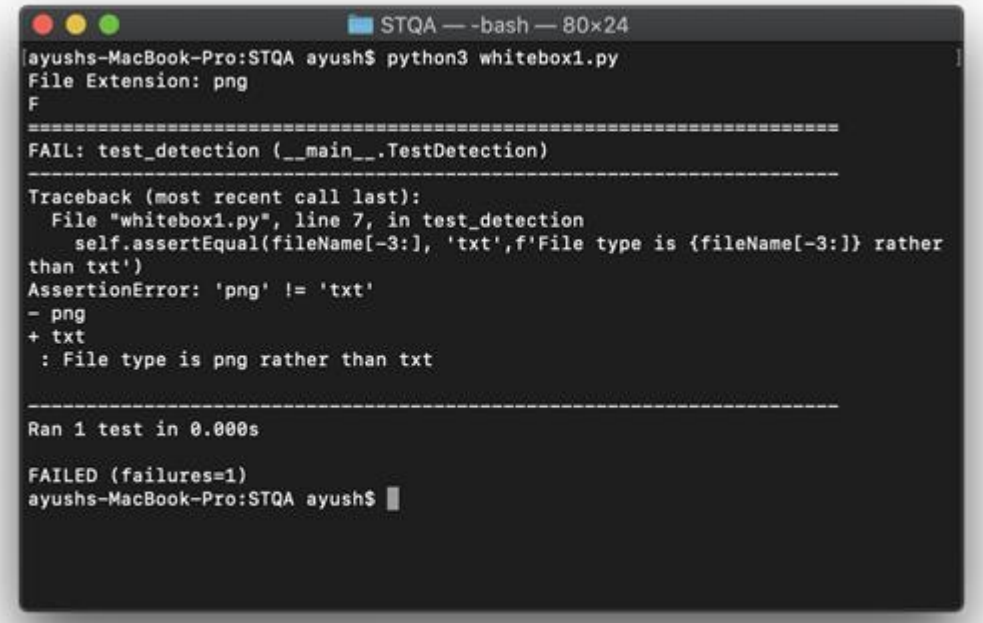


Figure 28- Testing environment 3

5.6.2 Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings.

Implementation Details:

CONDITION	Rule1	Rule2	Rule3	Rule4	Rule5	Rule6	Rule7	Rule8	Rule9
Mean	F	T	T	T	T	T	T	T	T
S.D	T	F	T	T	T	T	T	T	T
Rise Time	T	T	F	T	T	T	T	T	T
Fall time	T	T	T	F	T	T	T	T	T
Lower X	T	T	T	T	F	T	T	T	T

Upper X	T	T	T	T	T	F	T	T	T
Interval	T	T	T	T	T	T	F	T	T
Padding	T	T	T	T	T	T	T	F	T
Output	F	F	F	F	F	F	F	F	T

Rule 1: The mean value is float and other values are integer

Rule 2: The standard deviation is float and other values are integer

Rule 3: The Rise time is float and other values are integer

Rule 4: The Fall time value is float and other values are integer

Rule 5: The Lower X value is character and other values are integer

Rule 6: The upper X value is character and other values are integer

Rule 7: The padding is float and other values are integer

Rule 8: Every parameter has integer value

Chapter 6

Results and Discussion

In this chapter, we put our inferences after the simulation has been carried out. This will give a holistic view of the performances of each of the modes of the plot.

6.1 Oscilloscope mode VS Pyplot mode for DSP :

There are two choices of modes to display the system generated event files. Each one displays output files with dynamic scaling of the amplitude against the increasing time. A sample file of different events was passed to test the performance of these tools which are discussed in the upcoming sections. The figure 8 and 9 below shows random files being displayed in the different modes of the system.

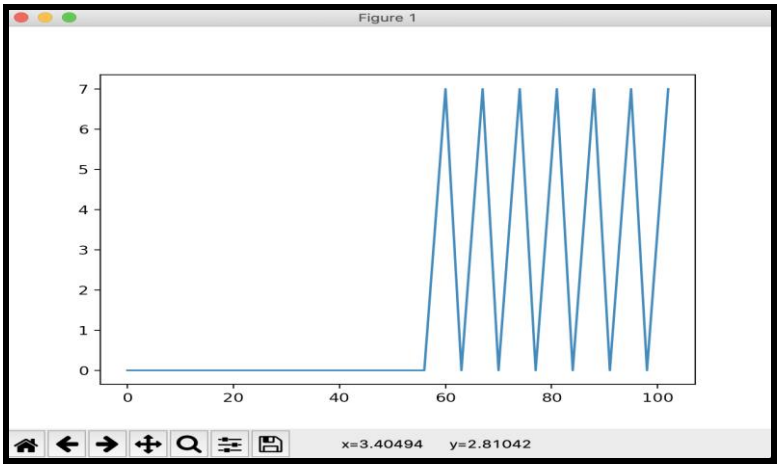


Figure 29- PyPlot Mode Graph

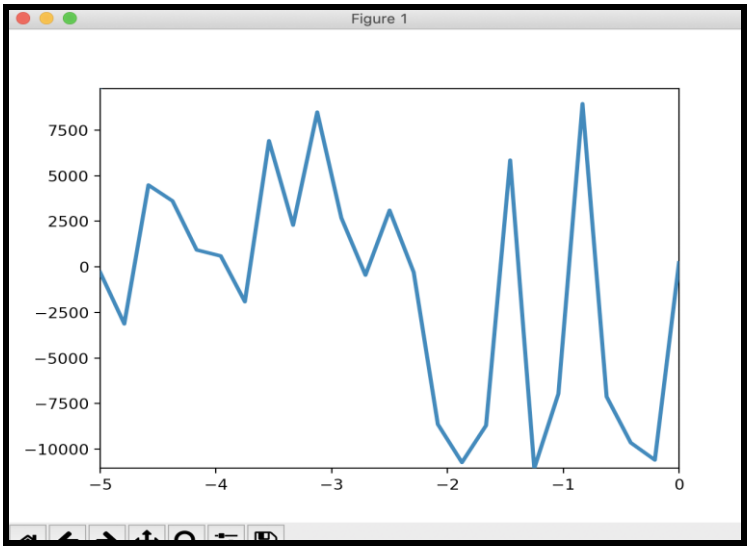


Figure 30- Oscilloscope Mode Based on Random file

6.2 Read/Write In/Out to evaluate performance

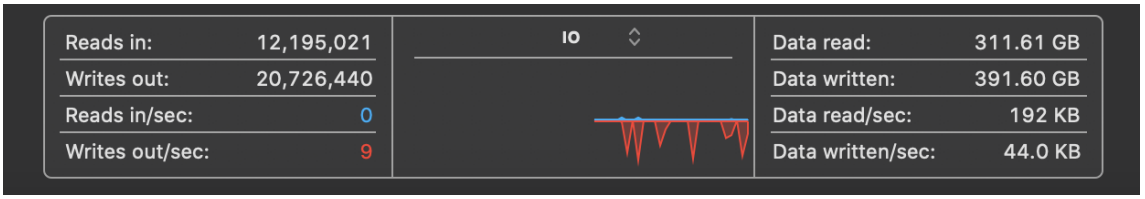


Figure 31-PyPlot Mode Image

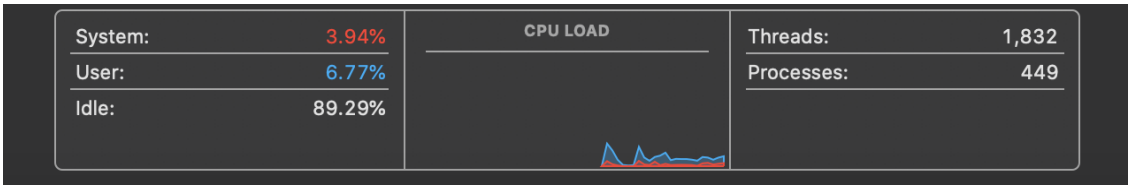


Figure 32- Oscilloscope Mode Image

The idle time in pyplot mode is 94.6% and that in OSC has a value of around 89.2%. Which further brings to the discussion that CPU is extensively utilized more efficiently in the later model. Also, the number of threads in the latter is greater by a value of 44 which tells how efficiently the plot is internally synchronized. Additionally, there is a 2.54% increase in user oscilloscope mode and speed of the system usage is doubled.

6.3 Summarising discussion based on the system process reports

Oscilloscopes can measure the amplitude of the signal, the frequency of a signal and we can also verify if the signal is of the shape that we are expecting, which makes it a preferred choice over pyplot and matplotlib

Also, an oscilloscope helps us to recognize the peak to peak amplitude of the signal along with the frequency of the signal.

Additionally, an oscilloscope is perfect for troubleshooting DC power supplies with excessive ripple resulting from component failure.

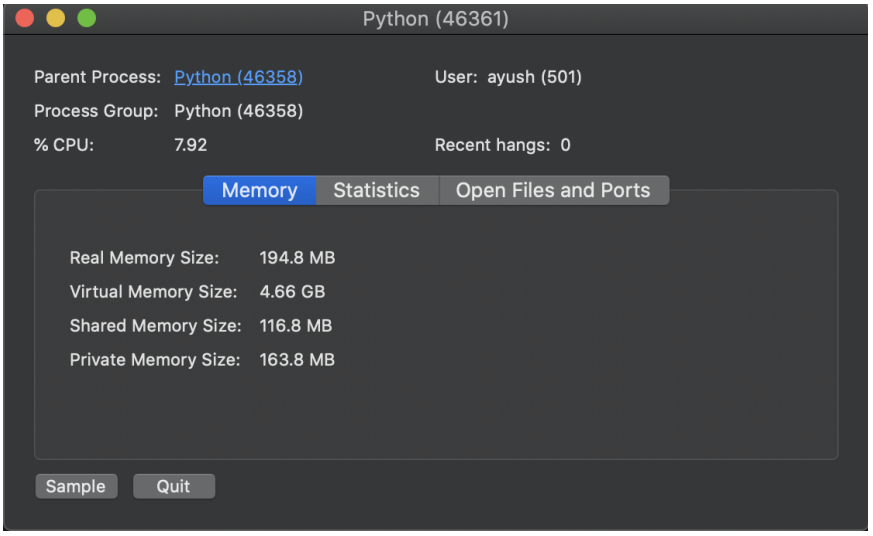


Figure 33- PyPlot Process Memory Summary

Figure 34- OSC Process Memory Summary

Based on the vast difference in the shared memory size, one can tell that the OSC mode is much more efficient than the normal plot mode for acquisition and visualization of signals coming in nanosecond time frames. Also, the percent CPU usage is more than double the value in oscilloscope mode. Which tells how CPU utilization has been the focus in oscilloscope mode.

6.4 Statistics of both the modes:

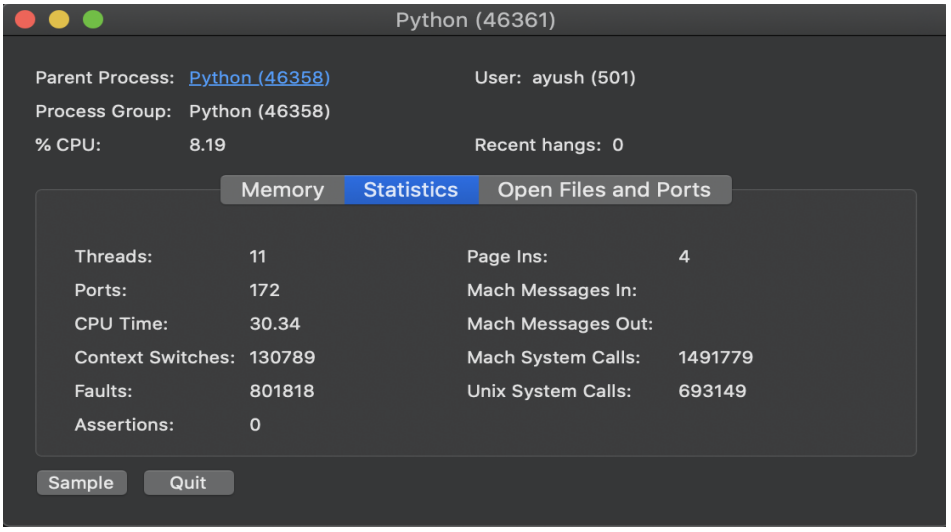


Figure 35-Pyplot Mode Statistics Image

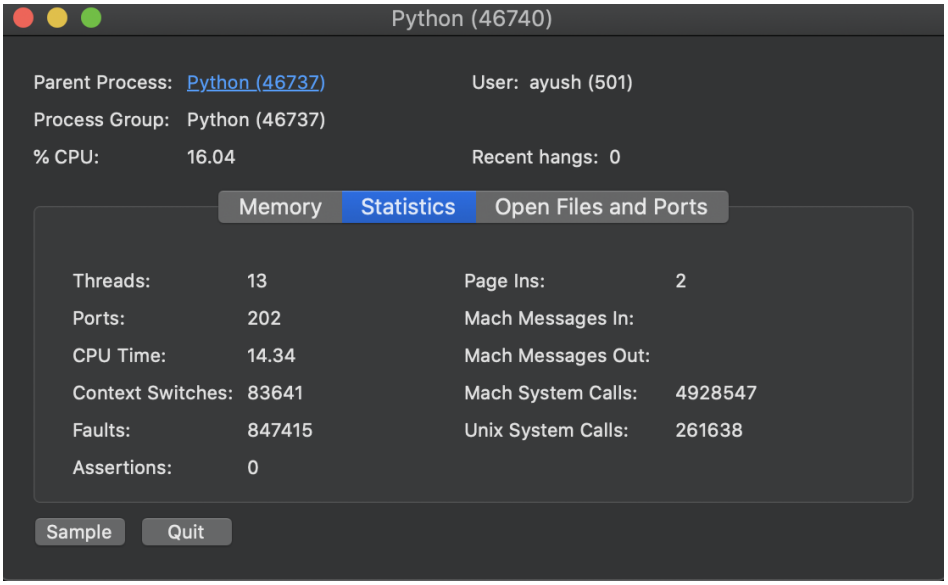


Figure 36- Oscilloscope Mode Statistics Image

So overall based on the processing of the programs one can say that the CPU utilisation is to be kept in mind while running programs in different modes also there is more amount of

parallel processing which makes oscilloscope mode better at the real-time acquisition of the data.

Chapter 7

Conclusions and Further Work

This chapter discusses the lessons learned and the knowledge gained after the completion of our project and the possible future scope of our project.

7.1 Conclusions

This project involved the development of routines which looked after the amplitude and time spectrum. The optimised subroutines were used for the generation of digitized pulses given the information pertaining to the process that was probed and analysed. These pulses which were generated were in turn used for testing of data acquisition systems and associated signal processing techniques. As these systems require data in nanosecond time, we built a library which could generate these events in the specified time. These libraries were created using C and then imported in the Python library. The algorithm was initialized by a reference pulse shape, with the statistical distribution of amplitude and time.

7.2 Further Work

Encouraged by the results, In the foresight, we will be building a Desktop Application for our project specifically for the Windows Operating System. A problem called “Pile-up problem” which frequently occurs in randomly generated events will be dealt with by writing appropriate algorithms. We plan to change the way to acquire data where we will be using ADCs in the Zynq hardware using PetaLinux tools.

7.3 References

[1] A. Abba, A. Geraci, "Dynamic Maximization of Filter Length in Digital Spectroscopy", IEEE Trans. on Nucl. Sci., Vol.59, No. 5, Oct. 2012, pp. 2451 – 2456.

[2] Abba, F. Caponio, F. Guerrieri, A. Geraci, G. Ripamonti, "A Novel algorithm for pulse amplitude modulation for digital emulation of radioactive sources", Proc. of the IEEE Nucl. Sci. Symp., Oct. 30 – Nov. 6, 2010, Knoxville, Tennessee, USA.

[3] A. Abba, F. Caponio, A. Geraci, G. Ripamonti, " Design and test equipment of digital processors for output analysis from radiation detectors", Proc. of 2011 IEEE Nuclear Science Symposium, October 23-29, 2011, Valencia, Spain.

[4] R. Abbiati, A. Geraci, G. Ripamonti, "Self-configuring digital processors for on-line pulse analysis", IEEE Trans. Nucl. Sci., Vol.51, Issue:3, June 2004, Pages:826-830.

[5] M.Buffa, R.Abbiati, A.Geraci, G.Ripamonti, "Configurable digital data system for simulation and processing of detected events", Proc. of 2005 IEEE Nuclear Science Symposium, October 23 - 29, 2005, Wyndham El Conquistador Resort, Puerto Rico.

[6] Papoulis, S.U.Pillai, "Probability, random variables and stochastic processes", McGraw-Hill, 2001.

[7] D.E.Knuth, "Art of Computer Programming, Volume 2: Seminumerical Algorithms", 3rd Edition, Addison-Wesley Professional, 1997.

[8]McAnney Bardel, Savir Bardel, "Built-in test for VLSI: Pseudorandom Techniques", John Wiley and Sons, 1987.

[9] J.Millman, A.Grabel, "Microelectronics", McGraw-Hill, 19

ACKNOWLEDGEMENT

We have immense pleasure in successful completion of this work titled "Development of Digital Pulse Emulator" The special environment at KJSCE, that always supports educational activities, facilitated our work on this project, special thanks to our principal, vice principal and Head of the Computer department.

We acknowledge the support, guidance and encouragement, extended for this project by our guide Prof. Gopal Sonune Sir and Prof.Bhakti Palkar Ma'am who responded promptly and enthusiastically to our requests, despite their congested schedules.
We are thankful to TIFR for having given us this opportunity to work on this project. We thank our mentor cum guide Mr. Jaydeep Gore Sir for being so helpful at every point during the different project steps.

We are also thankful to Library Staff and Administrative Staff of KJSCE, who directly or indirectly have been helpful in some or the other way.
We thank our parents and friends, who encouraged us to extend our reach. With with their help and support, we have been able to complete this work well and on time.