Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

FINAL REPORT

REVISION – 1
30 April 2022

Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

CONCEPT OF OPERATIONS

REVISION – 2
30 April 2022

Concept of Operations
for
Implementing a GRU based network to detect stock prices on FPGA

Team 10

Approved by:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	2/9/2022	Kevin Sipple		Draft Release
2	2/30/2022	Sean Kwatra	Kevin Sipple	Updated the block diagram

Table of Contents

List of Figures

1. Executive Summary

The purpose of this system is to address the need for an easy to use and affordable stock data analysis tool. The solution is to make use of a gated recurrent unit (GRU) based neural network to make predictions of the future price movement of user selected stocks. The network will make use of past price trends of the company's stock as well as select exterior data about the company and the impact on this stock price. Exterior data that will be considered are: market sentiment, earnings per share (EPS), and insider sale/buy orders. Once data has been collected the neural network will be run on a field-programmable gate array (FPGA) which limits the processing power available to use. Once the data is analyzed the neural network will make an up-to-date and simple to understand prediction on the future of a stock.

2. Introduction

The purpose of this document is to present the GRU Based Stock Prediction System (GBSPS), a low cost machine learning system capable of running on limited processing power devices. This project has the potential to provide retail traders with an easy-to-use tool for analyzing relevant stock data that will provide an accurate prediction of the direction of stock price movement.

2.1. Background

In recent years the development of stock trading platforms that do not charge the user commission fees has dramatically increased the amount of retail traders entering the stock market. Current estimates place 32% of total U.S. equity volume to be represented by retail traders[5]. As of January 2021, the combined number of active retail traders across the six largest non-commission platforms was 100 million [5]. With many of these traders being brand new to trading with little to no knowledge of the market dynamics they remain vulnerable to the risk of losing money on trades without having accessible tools to analyze relevant stock data.

Predicting a stock's movement in the market is a strenuous but essential task for both amateurs and experienced retail traders. The price action of stocks is reactive to many factors and it can be difficult to track and manage all of the relevant data needed in order to make accurate predictions. Much of the data is spread out across many sources and can be time consuming and cumbersome to compile in an analyzable form. Having this relevant and up to date information about a company is extremely important for a stock trader to make decisions on their investments. Larger banking firms offer tools and professional analyst breakdowns on the risk/potential of a company's stock, but these analyses are often thousands of dollars in price, making them unobtainable to most retail traders.

Common metrics used by traders today such as EPS, trade volume, and short interest are often only updated in intervals and too cumbersome to fully analyze their history leaving traders vulnerable to acting on dated information. Accessing data which is more current from professional analysts comes at an expensive fee and thus leaving retail traders at an even further disadvantage. Traders being unable to obtain fully analyzed data can often lead to them missing out on opportunities and/or, losing money in the stock market. Current movement predictions are few and far between and often out of date for the current state of a company. Our system will improve this by being able to provide up-to-date predictions based on current market data and other external data. Also, our system will have a small impact overhead to allow the use of an FPGA to reduce the commonly large up-front costs of obtaining hardware capable of running neural networks. Another way this system will decrease the cost factor is it will replace the need to pay for subscriptions needed to access AI services.

2.2. Overview

The GRU Based Stock Prediction System (GBSPS) is split into three main sections for its complete operation. The first section will be the FPGA itself. This section requires enabling the FPGA to utilize Python programs within a safe operation region. To enable this a Linux kernel will be installed onto the FPGA with a Python interpreter. The next, data analysis, shall collect and examine data to serve as the input of the GBSPS. Once data is collected it must be categorized into what the GRU can utilize to predict the stock price. Finally, the third section of operation is the implementation of the GRU based neural network in order to predict the stock price. This subsystem entails utilizing the data collected by the second subsystem to predict the future stock price. After all of these subsystems are completed we will then merge them together to have the machine learning algorithm, written in Python, run on a Linux kernel on an FPGA.

2.3. Referenced Documents and Standards

[1]. “All Symbols in Tensorflow 2 :   Tensorflow Core v2.8.0.” TensorFlow, https://www.tensorflow.org/api_docs/python/tf/all_symbols.

[2]. Sweta, et al. How to Port PetaLinux Onto Your Xilinx FPGA. 2015, https://www.xilinx.com/publications/xcellonline/Xcell90_p46.pdf.

[3]“Yahoo Finance - Stock Market Live, Quotes, Business & Finance News.” *Yahoo! Finance*, Yahoo!, https://finance.yahoo.com/.

[4]*Zynq 7000 SOC (Z 7007s, Z 7012s, Z 7014s, z 7010 ... - Xilinx*. https://www.xilinx.com/support/documentation/data_sheets/ds187-XC7Z010-XC7Z020-Data-Sheet.pdf.

[5]McCrank, John. “Factbox: The U.S. Retail Trading Frenzy in Numbers.” *Reuters*, Thomson Reuters, 29 Jan. 2021, https://www.reuters.com/article/us-retail-trading-numbers/factbox-the-u-s-retail-trading-frenzy-in-numbers-idUSKBN29Y2PW.

3. Operating Concept

3.1. Scope

The GBSPS will enable retail traders to have access to a system with a low end cost that is capable of analyzing large amounts of stock data efficiently. With this prediction traders

will be able to make better informed decisions that have current market data considered on stocks of their choice. Our system will first be able to collect and analyze relevant data from the internet. Second, it will be able to make a prediction of the future movement of the stock price.

3.2. Operational Description and Constraints

The GBSPS is to be used to give users an affordable alternative system for analyzing stocks of their choice to predict stock price movement utilizing both relevant and up to date stock data.

The intended device for usage will be an FPGA with an embedded Linux operating system. The user will use the program on the FPGA by typing in the stock ticker they want to analyze. To display the output of the program the FPGA Linux will connect to a computer using an Secure Shell Protocol (SSH) terminal. An individual should use this system to get a prediction based upon current stock data for any stock they wish to gain insight on.

Since an FPGA is a minimal processing power device, our system should be capable of being run on almost any other hardware, but it will not use that hardware to its full capability. Since the FPGA is limited in processing power and a neural network will strain it, the CPU temperature will need to be monitored to avoid permanently damaging the hardware. To train the GBSPS data will need to be taken from multiple different sources. However, since the training also must be done on the FPGA this limits how much data can be used at a given time. Thus, the amount of data used must be carefully calibrated to avoid the GRU from becoming too complex. The GBSPS will not be capable of predicting an exact price, but rather a direction of movement with possible ranges. Exact price unfortunately cannot be predicted because stocks are inherently volatile and random and many do not behave similarly. The machine learning training process must be constrained to complete training in a time frame of a few hours so that the user can have predictions about a stock between market days. Since all stocks respond differently to market data it will be a burden to generalize a training process for the machine learning model and a brute force approach will be needed.

3.3. System Description

GRU Architecture Subsystem: The GRU architect subsystem will handle the implementation of the machine learning GRU model in Python 3.7. This subsystem will be in control of the data usage for training the GRU model. Considerations that will be made in this subsystem is training the model with enough data that it generates accurate predictions, but does not strain the final implementation on an FPGA device.

Data Scraping and Processing Subsystem: This subsystem will consist of data acquisition and the processing of the data. This subsystems goal is to optimize large data sets from multiple sources and time frames into manageable files for training the GRU model.

FPGA Subsystem: The FPGA subsystem will manage the Linux Kernel and implementation of Python on the device. This subsystem will also manage the health of the FPGA ensuring the trained model does not damage the device.

3.4. Modes of Operations

This system will have one mode operation. First, the user will enter the stock ticker they wish to examine. Next, GBSPS will then collect the relevant data via the data scraper and process it. Finally after running the data through the trained model on the FPGA the system will predict the direction of stock price movement and return it to the user.

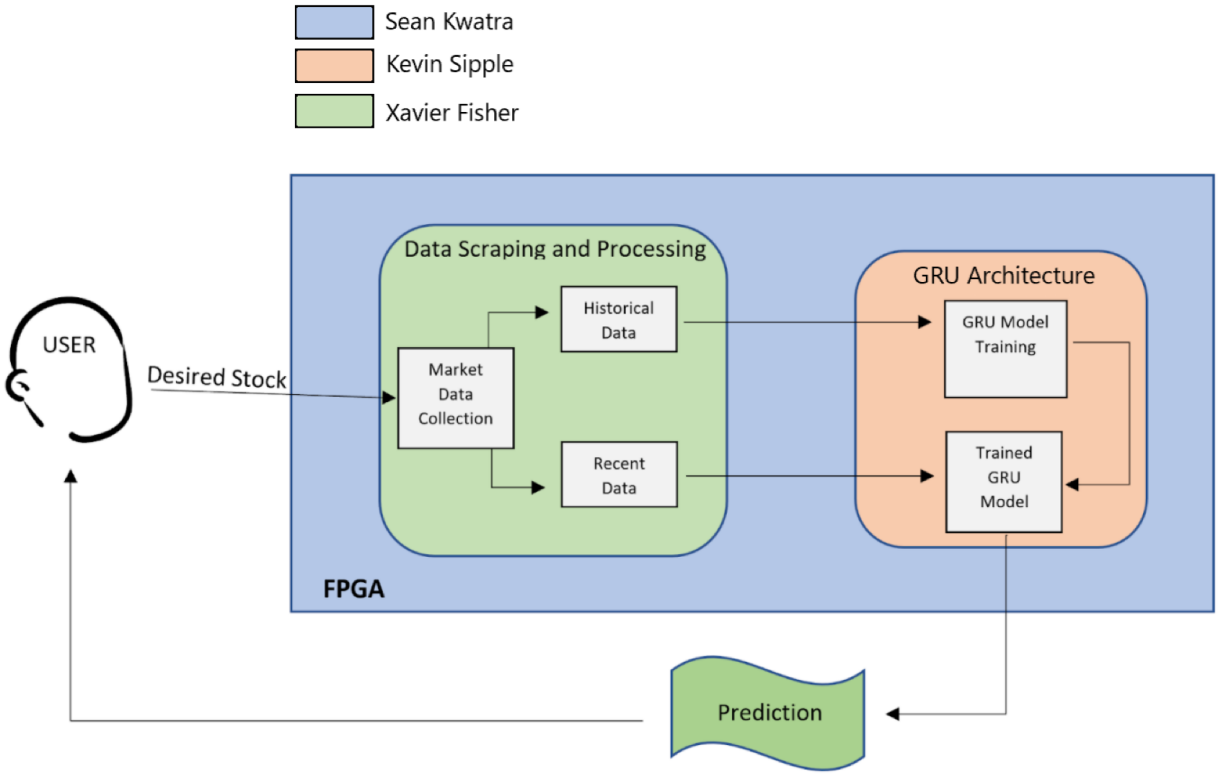


Figure 1: Flow chart of system usage/operation

3.5. Users

The GBSPS is intended for use by retail traders, our program will not require advanced technical knowledge of python or statistics because the output of the FPGA will be a simple to understand prediction of the stock price. It will be user friendly enough for novice stock traders. However, anyone desiring to use this system should be capable of navigating through the Linux operating system. Users will need to be able to run programs on linux and have knowledge of what stock ticker symbol they want analyzed. Those who will most benefit from the system are users looking for accurate stock price movement predictions in order to buy, sell, or otherwise trade in the stock market that do not have previous in-depth experience analyzing stock/market data.

3.6. Support

An operation manual for user interface (UI) will be provided. The user manual will provide documentation of what keywords can be used within the UI as well as how to efficiently run the program within the Linux terminal. An installation guide for all required software will be provided for the user inside this operation manual.

4. Scenario

4.1. Basic Stock Analysis Program

Our systems function will best be used to process relevant stock data for a retail trader. By processing this intricate and sometimes extensive data for those who do not have advanced experience in analyzing stocks then better informed trading decisions can be made by the user. The result of this processing will be a prediction of the direction of price movements for a selected stock. From there the user will make a better informed decision for how to proceed with their investment strategy.

5. Analysis

5.1. Summary of Proposed Improvements

This system is a rather minimalistic program that can be run on lightweight processing power devices, this enables the user to use the system on an FPGA while still being capable enough in data analysis to take much of the work off of the user and provide an accurate prediction of the direction of movement of a stock's price. This minimalistic approach will allow our program to be sold at a much lower price than the more in-depth stock analysis programs that many financial analysis companies currently sell.

Our program will also be very easy to use requiring only the ticker symbol of the stock to analyze the data. It will not require the user to find and input the data themselves in order to predict stock price movement which will lower the skill and time needed to use the program.

5.2. Disadvantages and Limitations

The upfront limitation of our system is the computing power of the FPGA being used, as this will restrict the program to need to function in the most practical manner possible with a low overhead. As it is a lightweight program it will not be able to do as in-depth of an analysis of larger data sets as one may need. Without this extra data the machine learning algorithms our program uses may not give as accurate predictions that we would like and may require more data to be used from different stocks to compensate. Limitations on the user interface and may result in a simple command line operation. This is due to the existence of an intricate GUI not being feasible since the FPGA's processing power must be available to the greatest extent possible in order to operate the GRU neural network.

5.3. Alternatives

The most obvious alternative to the GBSPS would be manually tracking the relevant stock data and trends in order to predict future movements of the stock price. The two most blatant issues with this method would be figuring out what data is relevant and analyzing all of this data in order to see how it affects stock price movement. With the GBSPS these issues are alleviated as it does the data gathering and analyzing for the user.

Other alternatives to the GBSPS are the programs that companies currently sell to analyze stocks and professional analysts reports. However, these programs have significant costs associated with them and oftentimes have a steep learning curve. Other alternatives exist in the area of AI predicting such as Danelfin who offers a subscription service to access their AI for $20/month, another includes Kavout which offers a service to analyze stock signals and data for $49/month.

5.4. Impact

The GBSPS has the potential to directly impact any user of this system by enabling a simple and straightforward entry into stock trading analysis. This will allow even the novice trader to begin making educated decisions in the stock market which in turn will help promote potential capital growth for the user and growth of the stock market in general. Our program will undercut competing programs which are currently utilized by non-retail traders. Since the GBSPS is aimed at retail traders it must be able to provide quick and easy access that was in the past unavailable to this audience. However, if the GBSPS is wrong in its prediction it may cause the users to invest into stocks that may either lose value or short stocks that gain value.

Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – Final
30 April 2022

Functional System Requirements
for
Implementing a GRU based network to detect stock prices on FPGA

Prepared by:

Team 10 4/30/2022

Author Date

Approved by:

Kevin Sipple 4/30/2022

Project Leader Date

Prof. Kalafatis Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	2/23/22	Kevin Sipple		Draft Release
2	4/30/22	Sean Kwatra		Final Release

Table of Contents

List of Tables

No Table Entries.

List of Figures

The following definitions differentiate between requirements and other statements.

Shall: **This is the only verb used for the binding requirements.**
Should/May: **These verbs are used for stating non-mandatory goals.**
Will: **This verb is used for stating facts or declaration of purpose.**

1. Introduction

1.1. Purpose and Scope

This document outlines the requirements necessary for the functionality of our project. The purpose of this system is to improve stock analysis for novice/retail traders. The GRU Based Stock Prediction System (GBSPS) will enable retail traders to have access to a system with a low end cost that is capable of analyzing large amounts of stock data efficiently. The system shall include a query interface so that the user can request specific stocks of interest. The system shall provide the user with accurate assessments and predictions of the movement of stock price. With this prediction traders will be able to make better informed decisions that have current market data considered on stocks of their choice. The system shall first be able to collect and analyze relevant data from the internet. The GBSPS shall do this by retrieving historical stock market data and outside contributing factors in order to train its models. Second, the result of the trained model shall then be able to interpret recent data to present an accurate prediction of the movement of the stock price.

This specification defines the technical requirements for the development items and support subsystems delivered to the client for the project. Figure 1 shows a representative physical product of the project in the proposed CONOPS. The verification requirements for the project are contained in a separate Verification and Validation Plan. The system's software layout and operation is further detailed by Figure 2.

Figure 1. Conceptual Image of FPGA

1.2. Responsibility and Change Authority

Team leader, Kevin Sipple, is in charge of ensuring that functionality specifications of the GBSPS are met. Each subsystem handler is responsible for ensuring that the requirements of their specific subsystem are met. All changes to the functionality of the overall system must be approved by the project sponsor Sambandh Dhal and team leader Kevin Sipple. Only the project sponsor Sambandh Dhal will have the authority to change any performance requirements of this project. Subsystem ownership is as follows:

- Kevin Sipple: GRU Architecture
- Xavier Fisher: Data Scraping and Processing
- Sean Kwatra: FPGA

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
IEEE Standard 802.3	2018-08-31	IEEE Standard for Ethernet transmission

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
UG114	6/16/2021	PetaLinux Tools Documentation

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

This section defines the minimum requirements GRU Based Stock Prediction System (GBSPS) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered in the following sections. These sections include: the requirements of the FPGA operability during usage, the controlled usage of relevant and accurate data from trusted sources, and the constraints and rationale for the time required to train the GRU neural network. The term “system” in this documentation will refer to the grouping of all subsystems or the GBSPS as a whole. The term “user interface” will refer to the method of interacting with the system, currently through the Linux command terminal. This term will remain consistent with its meaning in the event that the sponsor changes the method of user interaction.

3.1. System Definition

The GRU Based Stock Prediction System (GBSPS) consists of three key subsystems, detailed in Figure 2 below: GRU Architecture, Data Scraping and Processing, and FPGA.

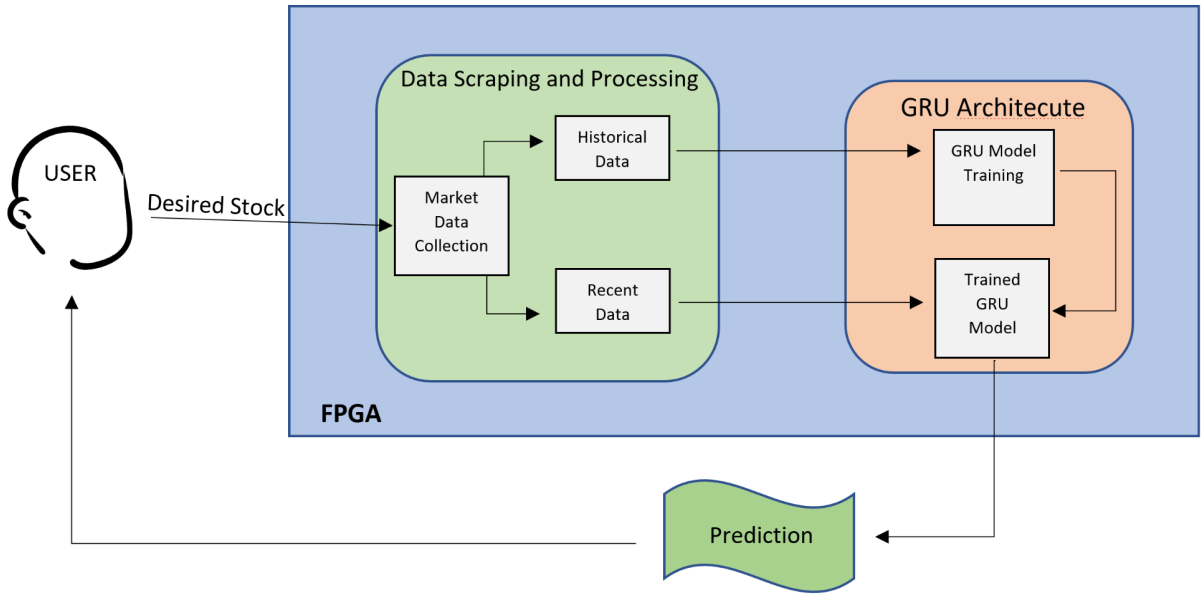


Figure 2. Block Diagram of System Operation

The FPGA subsystem, owned by Sean Kwatra, shall be tasked with providing an environment for the other two subsystems to run on and the user to interface with. This will consist of implementation of the required OS and programs for the two other subsystems to function all while ensuring the FPGA is within proper operation conditions. The user will input the desired stock into the interface on this subsystem. The user interface may be a simple command line interface.

The Data Scraping and Processing subsystem, owned by Xavier Fisher, will be tasked with gathering stock price and transaction data as well as other relevant external

information for the desired stock. This data is to be retrieved from the internet and organized into the two categories of historical and Recent data. Recent data may be data in the time frame of the past 63 trading days. This subsystem must gather this information and convert it into a form (CSV) that can readily be used by the GRU based neural network.

The GRU Architecture subsystem, owned by Kevin Sipple, will be tasked with compiling the data gathered from the Data Scraping and Processing subsystem. This data will then be used in a minimized and efficient manner to train a model for the user selected stock all on board the FPGA. The training process will be done using the historical data. Following this, the recent data will be fed into the trained model to produce an accurate prediction of the stock movement based on the recent stock data. Finally, the user will receive this prediction.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Retrieval of Stock Data From the Internet

The FPGA must connect to the internet to be able to acquire necessary stock data. The connection to the internet will be established via an Ethernet connection.

> *Rationale: This is a specified requirement from the customer. The requirement is to use the internet to acquire data from the intent without it being entered by the user themselves.*

3.2.1.2. Usage of a GRU Based Neural Network

The program must predict the near future movement of a stock price by utilizing a neural network. The selected neural network is a GRU which will be trained on data from past stock information. This neural network shall both train and run fully on the FPGA. The GRU neural network shall be implemented in the programming language Python 3.

> *Rationale: This is a requirement from the customer themselves. They specifically asked for a GRU based neural network to predict the future stock prices.*

3.2.1.3. Accuracy of Collected Data

The collection of data shall be done by collecting only from reputable sources. The system will only retrieve from sites/databases that have been known to act in accordance with the existing legislature. However, the accuracy of this data is at the discretion of the provider. Our system cannot ensure the data will always be accurate, however the data presented on the sites/databases our system will access is lawfully binded by the Sarbanes-Oxley Act (SOX 2002) to be accurate. The SOX Act establishes that public companies are inspected and audited to ensure truthful financial information reporting. Along with SOX, the Securities and Exchange Commission (SEC) regulates stock exchange data and transactions to ensure that the data being reported is accurate.

> *Rationale: Collecting only from sites that are reputable and known to act in accordance with the law will ensure the data is as accurate as possible.*

3.2.1.4. Selection of Current Data

The selected stock's data will be split into a category of current data for data that is within 63 trading days of age.

Rationale: By design the system must consider the impact of company earnings data on a stock. The common metric of splitting a company's annual performance is in quarters of 63 trading days. This categorization will classify the current data as relevant to the current quarter.

3.2.1.5. Stock Required History

The selected stock must have been listed on the market for a minimum 5 years in order for the system to be able to collect enough relevant data to train in the GRU Architecture subsystem.

Rationale: This allows the GRU model to have enough data to train networks for a more accurate prediction.

3.2.1.6. Model Result Delay

The system will be capable of gathering all necessary data, training, producing a prediction within 8 hours maximum.

Rationale: This is in place to ensure that the user can receive a prediction and have time to make an informed decision on their following action. The basis of 8 hours has been decided due to the time between trading days being 8.5 hours.

3.2.1.7. Accuracy of Prediction

The system shall be able to achieve an accuracy of greater than 55% in predicting the direction of stock movement

Rationale: Any consistent accuracy of greater than fifty percent is a large improvement in the prediction of stock movements.

3.2.2. Physical Characteristics

3.2.2.1. Running on an FPGA

The programs created must be utilized by an FPGA board. At no point will it run or utilize another computer to do its calculations and predictions for it. The FPGA must be able to run these programs while remaining in a safe operational condition. The native language of the FPGA is VHDL, but this will be replaced by the implementation of PetaLinux and Python 3 interpreter.

Rationale: This is a customer requirement. The requirements for this system is to have all programs onboard the FPGA and capable of running while remaining operable for extended use.

3.2.2.2. FPGA Functional Operating Temperature

For the continued operation of the FPGA it must remain below a temperature of 100°C when operating for extended periods of time.

Rationale: This is a physical constraint for the FPGA in order for it to not become inoperable.

3.2.3. Electrical Characteristics

3.2.3.1. Inputs

Our FPGA will connect to a computer using a serial connection with a bit rate of 11520 Bits per second. The Ethernet will allow the board to connect to an SSH terminal. The Ethernet does not require a specific bitrate and can handle up to 100 GB/s connections.

Rationale: This is specific to the board datasheet which shows the required bitrates of each component on the board.

3.2.3.2. Power Consumption

The maximum peak power of the system shall not exceed 60 Watts from the included power supply.

Rationale: This is the parameter recommended on the datasheet for the proper operation of the selected FPGA board and the power supply given by the manufacturer.

3.2.3.3. Input Voltage Level

The input voltage will be a power supply of 12V from the included power supply.

Rationale: This is the rated voltage of the selected FPGA board. There are not specified tolerances for this supply as it is provided by the FPGA board manufacturer. The Supply operates at: 12V, 5A, Max output: 60W

3.2.3.4. External Commands

The GBSPS shall document all external commands in the appropriate ICD.

Rationale: The ICD will capture all user interface details.

3.2.4. Stock Prediction Output

The program must communicate its prediction of the stock price to the user in an understandable format. This may either be done with a simple output predicting a direction of up or down prediction may include a chart showing a range of possible movements with confidence intervals. The existence of an intricate GUI is not feasible as the FPGA's processing power must be available to the greatest extent possible in order to operate the GRU neural network.

Rationale: The customer needs the end result of the program to be easily understood. While the final form of the output has not been determined by the customer, it still must show a prediction of stock price movement.

4. Support Requirements

Our system will be provided with an installation guide to aid the user in installing PetaLinux onto an FPGA. This guide will also describe the steps to get our system running on the OS. Along with the installation guide, a user manual will also be provided in which details of how to run the installed system will be described. Functionally our system may be run on other computers, but will be solely designed to be capable of running on low processing power devices i.e. an FPGA.

We may also provide a service to rent FPGA boards with our set up with our system already installed and ready to use. Customers that rent out the FPGA boards will have access to customer support which will help them with any issues they may encounter. However, customers will be liable for any permanent damage they cause to the FPGA board they are renting.

5. Appendix A: Acronyms and Abbreviations

FPGA	Field Programmable Gate Array
GRU	Gated Recurrent Unit
GBSPS	GRU Based Stock Prediction Program
CSV	Comma-separated values, text file
V	Volts
SEC	Securities Exchange Commision
SOX	Sarbanes-Oxley Act
VHDL	Very High Speed Integrated Circuit Hardware Description Language

6. Appendix B: Definition of Terms

Market/Trading Days	Days in which the stock exchange is open
Retail Trader	Traders that buy and sell securities on personal accounts

7. Appendix C: Interface Control Documents

The Interface Control Document is attached as a separate document.

Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

INTERFACE CONTROL DOCUMENT

REVISION – 2
30 April 2022

Interface Control Document
FOR
Implementing a GRU based network to detect stock prices on FPGA

Prepared by:

Team 10 4/30/2022
_____
Author Date

Approved by:

Kevin Sipple 4/30/2022
_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	2/22/2022	Sean Kwatra		Original Release
2	4/30/2022	Sean Kwatra	Kevin Sipple	Updated Information about power supply and FPGA

Table of Contents

List of Tables

1. Overview

This document, the Interface Control Document (ICD), will provide detail of how the FPGA board will interface with the user and the internet. In this document are the reference documents used, the physical characteristics, electrical interfaces, and interface protocols used through the project.

2. References and Definitions

2.1. References

IEEE Standard 802.3-2018
Standard for Ethernet
2018 Revision

UG1144 - PetaLinux Tools Documentation -16 June 2021
PetaLinux Tools/Installation Documentation, provided by Xilinx

2.2. Definitions

FPGA	Field programmable Gate Array
GRU	Gated Recurrent Unit
TMDS	Transition-minimized differential signaling
USB	Universal Serial Bus
HDMI	high-definition multimedia interface
kg	kilo-gram (10^3 grams)
cm	centimeter (10^{-2} meters)
DC	Direct Current
V	Volts

3. Physical Interface

3.1. Weight

Module	Weight of module (kg)
ZCU104 (FPGA)	0.54
Zynq heat sink Part Number ATS-54	N/A
Custom FPGA Case	TBD

- Table 1: Weight of components

3.2. Dimensions

Module	Height (cm)	Length (cm)	Width (cm)
ZCU104 (FPGA)	1.9	18.3	14.5
Zynq heat sink Part Number ATS-544	1	2.8	2.8
Custom FPGA Case	2.3	18.5	14.7

8. Table 2: Dimensions of components

3.3. Mounting

3.3.1. Mounting FPGA In Case

The FPGA comes with 4 through holes and compatible screws. These will be utilized to mount the FPGA within the case

3.3.2. Mounting the Heat Sink

The heat sink will be fitted onto the FPGA's CPU via adhesive thermal paste. The sink may require additional mounting steps in the event that the adhesive thermal paste does not fully adhere the heat sink to the CPU.

4. Thermal Interface

4.1. Heatsink

Our FPGA board may overheat during operation of the system's software. To prevent this from occurring a heat sink will be installed onto the FPGA board via adhesive thermal paste. The heat sink will be installed upon the central processing unit of the FPGA. Additional steps may be required in the event that this does provide sufficient cooling.

5. Electrical Interface

5.1. FPGA Board Power Supply

The FPGA board is provided with a power supply that draws a nominal amount of power, thus this provided power supply will be used. The specifications of this supply is a 12V DC wall adapter, with the capability of outputting a max of 60 Watts. The wall adaptor will connect to the FPGA board using the power input on the FPGA.

5.2. Video Interfaces

The FPGA board will connect to a monitor to display the system interface and output of the system. To connect to the monitor the FPGA will use the onboard HDMI out port.

5.3. User Control Interface

Users will be able to interact with the FPGA by using a USB connected mouse and keyboard plugged in directly to the onboard USB ports.

6. Communications / Device Interface Protocols

6.1. Wired Communications (Ethernet)

The FPGA board will connect to the internet by way of wired connection utilizing an ethernet PHY interface protocol. This will be handled onboard the FPGA and will not be interfered with by our system.

6.2. USB Inputs

The USB ports on the FPGA use the standard USB 2.0 interface to connect to all input devices. These ports will be handled onboard the FPGA and will not be interfered with by our system.

6.3. Video Interface

The HDMI out port on the FPGA uses the standard TMDS encoding to transmit high amounts of data to a display device. This will be handled onboard the FPGA and will not be interfered with by our system.

Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

SCHEDULE AND VALIDATION PLAN

REVISION – 2

23 February 2022

Schedule and Validation Revision - 2

Implementing a GRU based network to detect stock prices on FPGA

Status Color Key:

Completed	In Progress	Incomplete

Schedule:

Work	End Date	Owner	Status	Date Completed
Concept of Operations	2/9/2022	All		2/8/2022
Collect primary market data (Open, close, low, high, volume)	2/28/2022	Xavier		2/18/2022
Functional System Requirements	2/23/2022	All		2/23/2022
Interface Control Document	2/23/2022	All		2/23/2022
Order Heatsink	2/26/2022	Sean		2/23/2022
Midterm Presentation	2/28/2022	All		2/28/2022
Download all modules needed for GRU and generate test network	2/21/2022	Kevin		2/18/2022
Learn the Architecture of GRU neural networks	2/28-30/2022	Kevin		
Order Parts	2/30/2022	All		2/30/2022
Read in data to create arrays for training	3/1/2022	Kevin		
Generate multidimensional arrays of different data combinations	3/7/2022	Kevin		3/7/2022
Start Learning 3D printing	ECEN 404	Sean		
Start Designing Case	ECEN 404	Sean		
Install Embedded Linux System	3/4/2022	Sean		4/22/2022
Test different layer numbers and neuron numbers and produce visualizing plots	3/28/2022	Kevin		3/26/2022
Collect EPS data	3/28/2022	Xavier		3/28/2022

Schedule and Validation Revision - 2

Implementing a GRU based network to detect stock prices on FPGA

Receive Heat Sink	3/11/2022	Sean		3/15/2022
Collect Inflation rates	4/04/2022	Xavier		4/04/2022
Collect Insider Trading data	4/04/2022	Xavier		4/04/2022
Project Update Presentation	3/21-28/2022	All		
Collect Google Trends data	4/11/2022	Xavier		4/11/2022
Implement an accuracy metric	4/18/2022	Kevin		4/8/2022
Create system to vary arrays to find best characteristics	4/25/2022	Kevin		4/21/2022
Finish Casing	ECEN 404	Sean		
Mount Heatsink onto Casing and Board	ECEN 404	Sean		
Final Presentation	4/11-18/2022	All		4/18/2022
Subsystem Demos	4/25-29/2022	All		4/26/2022
Compile/Write Final Report	4/25-30/2022	All		4/30/2022
Final Report	4/30/2022	All		4/30/2022

Validation Plan:

Task	Specification	Owner	Status
Ensure objective collected market data is correct	Visually inspect and cross reference collected objective data points from different sources and ensure accuracy	Xavier	Completed
Handles empty datasets without interrupting program	Give inputs with known empty datasets	Xavier	Completed
Correct data aligned with corresponding date in CSV	Visually inspect CSV file and compare to website data	Xavier	Completed
No duplicate data appended to CSV	Visually inspect CSV file and ensure there are no duplication anomalies	Xavier	Completed
Order parts	FPGA compatible Heatsink	Sean	Completed
Design and Mount Case	Create Case to mount Heatsink upon	Sean	Delayed to next semester
Mount Heatsink	Mount Heatsink on FPGA and Case	Sean	Delayed to next semester
Install Linux	Use PetaLinux to install an embedded linux system	Sean	Completed
Look back system produces arrays of specified structure and is verified in program: (size, lookback, attributes)	Verify the data has been organized correctly into an array with lookback for training	Kevin	Completed
Ensure model is properly cleared between tests	Clear the Tensorflow backend to make all weights zero again before building a new model	Kevin	Completed
Arrays are split properly based upon	The building of the arrays must have NaN removed and a split	Kevin	Completed

year of data and NaN rows are removed	for training and test data		
Directional Accuracy and RMSE functions output confirmed	Design a accuracy metric to compare the models performances to one another	Kevin	Completed
Create system to vary arrays to find best characteristics	Compare accuracy of different data combinations	Kevin	Completed
Test performance on FPGA and compatibility	Run the system onboard the FPGA	Kevin	Delayed to ECEN 404

Implementing a GRU based network to detect stock prices on FPGA

Kevin Sipple
Xavier Fisher
Sean Kwatra

SUBSYSTEM REPORTS

REVISION – Original
26 April 2022

Subsystem Reports
for
Implementing a GRU based network to detect stock prices on FPGA

Prepared by:

Team 10 4/26/2022

Author Date

Approved by:

Kevin Sipple 4/26/2022

Project Leader Date

Prof. Kalafatis Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	4/26/2022	Sean Kwatra		Original Release

Table of Contents

List of Tables

List of Figures

1. Introduction

The GRU Based Stock Prediction System (GBSPS) will take in the ticker of a stock the user selects and aggregate data available from reputable sources on the internet in order to train a model that in turn will give predictions of the direction and price range. The system has three main sections for its complete operation. These subsystems are: FPGA, Data Scraping, and GRU Architecture. Each of these subsystems have been rigorously tested and designed based around the demands of the other subsystems. Since all are operational and functioning as specified in the Conops, FSR, and ICD, integration will be the next step in our design process.

2. Data Scraping Subsystem Report

2.1 Subsystem Introduction

2.1.1 Overview

The data scraping subsystem for the GBSPS is vital for feeding the Gated Recurrent Unit (GRU) based neural network the information it needs to make a well-informed prediction. The goal of this subsystem is to quickly and accurately retrieve the data that we are interested in for any given stock ticker and append this data to a .CSV file. This subsystem will need to work efficiently as there are lots of data points for each individual stock, it is also of great importance that the data that is retrieved is appended to the correct corresponding date as well.

2.2. Subsystem Details

2.2.1. Main Stock Data

For the main stock data, volume traded, low, high, open, and close price are collected for each trading day. These are the most common data points used for analyzing stocks. This data is sourced and verified from Yahoo Finance using the pandas-data reader library which has stock data from various internet sources readily available for use and cross-verification.

```
#prompt user for stock tickers
userInput = input("Enter stock tickers of interest, one at a time, followed by ↵ (enter q to stop): ")
userInput = userInput.upper() #make stock ticker all capital letters

stockTickerArray = [] #empty list of stock tickers
while (userInput != 'Q'):
    try:
        currentData = pdr.DataReader(userInput, 'yahoo', startDate, endDate) #try to get main stock data from yahoo finance
    except:
        print ("Stock ticker invalid please enter another or press 'q' to stop: ")
    else:
        stockTickerArray.append(userInput) # add valid stock ticker to list of valid stick tickers
        currentFileName = userInput + "_stock_data" + ".csv" #create base file
        with open(currentFileName, 'w', encoding= 'UTF8' ) as f: #this writes the current stock tickers data to a csv file
            currentData.to_csv(currentFileName) #save data retriebed from yahoo finance (dataframe object) to the base csv

    userInput = input() #get next stock ticker from user
    userInput = userInput.upper()
```

Figure 1: User Input Code Capture

2.2.2. Insider Trading, Stock EPS data, and Inflation Rates

As there are no libraries for these data points HTML parsing was used extensively in order to retrieve this data. We decided on using the libraries Requests (sending HTTP requests to get the HTML) and BeautifulSoup (for parsing the HTML) for their ease of use and beginner friendliness.

The amount of insider trading data points available largely depends on the individual company and since there are not that many insider trades happening per day

a lot of the data points are zeros. Since for a majority of the time insiders are not trading shares at all when there is an insider buy or sell it can be a very important indicator on how a particular company is doing behind-the-scenes. The following chart is what was embedded in the HTML of the insider trading source we used (http://openinsider.com/). The data points of interest are the trade type (Trade Type) and amount bought and sold (Qty) for a particular transaction.

X	Filing Date	Trade Date	Ticker	Insider Name	Title	Trade Type	Price	Qty	Owned	ΔOwn	Value	1d	1w	1m	6m
M	2022-04-01 16:45:11	2022-03-30	AMZN	Selipsky Adam	CEO Amazon Web Services	S - Sale	$3,348.08	-69	22,296	0%	-$231,017				
	2022-03-22 16:17:33	2022-03-18	AMZN	Selipsky Adam	CEO Amazon Web Services	S - Sale	$3,200.00	-344	22,365	-2%	-$1,100,800				
D	2022-02-23 16:52:59	2022-02-22	AMZN	Jassy Andrew R	Pres, CEO	S - Sale+OE	$3,009.57	-492	94,729	-1%	-$1,480,708				
D	2022-02-23 16:40:40	2022-02-22	AMZN	Clark David H	CEO Worldwide Consumer	S - Sale+OE	$3,009.57	-137	4,130	-3%	-$412,311				
	2022-02-17 17:45:27	2022-02-15	AMZN	Selipsky Adam	CEO Amazon Web Services	S - Sale	$3,150.92	-679	22,709	-3%	-$2,139,475				
D	2022-02-17 17:29:56	2022-02-15	AMZN	Jassy Andrew R	Pres, CEO	S - Sale+OE	$3,150.92	-848	93,971	-1%	-$2,671,980				
D	2022-02-17 17:23:23	2022-02-15	AMZN	Clark David H	CEO Worldwide Consumer	S - Sale+OE	$3,150.92	-1,027	4,077	-20%	-$3,235,995				
DM	2022-02-17 17:15:05	2022-02-15	AMZN	Zapolsky David	SVP	S - Sale+OE	$3,146.85	-1,199	3,119	-28%	-$3,773,073				
D	2022-02-17 17:08:31	2022-02-15	AMZN	Olsavsky Brian T	SVP, CFO	S - Sale+OE	$3,150.92	-910	2,372	-28%	-$2,867,337				
D	2022-02-17 17:01:17	2022-02-15	AMZN	Reynolds Shelley	VP	S - Sale+OE	$3,150.92	-222	6,122	-3%	-$699,504				
M	2022-01-07 16:24:08	2022-01-05	AMZN	Selipsky Adam	CEO Amazon Web Services	S - Sale	$3,303.34	-69	23,388	0%	-$227,930				

Figure 2: Insider Trading Data

EPS is a common metric to see how well a particular company's stock is doing, it is calculated as a company's net profit divided by the number shares it has outstanding. For this data we sourced my data from https://www.macrotrends.net/ a reputable source when it comes to investment research. An issue with getting this data is that it appears to only have begun being reported at around 2009, so we are unfortunately limited by this but besides this it is calculated and updated every quarter. The following chart is how it appears embedded in the website's HTML and what we had to extract from the webpage for each individual stock requested.

Apple Quarterly EPS	
2021-12-31	$2.10
2021-09-30	$1.23
2021-06-30	$1.30
2021-03-31	$1.40
2020-12-31	$1.68
2020-09-30	$0.74
2020-06-30	$0.65

Figure 3: EPS Example Data

When looking at the GRU predictions based on the initial data we were not accounting for inflation, and it seemed like the prediction was following the trend well but shifted down (think same slope but different intercept). This is when we decided that looking into how inflation rate may play a role in the general trends of a stock price may

be useful. For this data is sourced from https://www.officialdata.org/ which in turn gets their data directly from the Bureau of Labor Statistics. The following chart is how the inflation rate (relative to the previous year) was presented on the website and what we had to extract from the HTML of the webpage.

Dollar inflation: 1998-2022

Year	Dollar Value	Inflation Rate
1998	$1.00	1.56%
1999	$1.02	2.21%
2000	$1.06	3.36%
2001	$1.09	2.85%
2002	$1.10	1.58%
2003	$1.13	2.28%
2004	$1.16	2.66%
2005	$1.20	3.39%

Table 1: US Dollar Value Inflation Rates

2.2.3. Google Trends Data

For trend/sentimental data, Google Trends is used since their data is easily accessible. Additionally, Google is the most widely used search engine so their search trends cover a large majority of global searches. Since Google does not have an official API for use with their trend data, we had to look around for libraries that were able to retrieve this data. We initially attempted to create a script that went to the webpage of the stock ticker we wanted trend data for and clicked the download button in order to export this data to a .CSV file but unfortunately Google does not allow this on their websites. Fortunately, there were a few libraries that were able to circumvent this problem and the one we ended up going with was PyTrends which was able to get us the data that we needed.

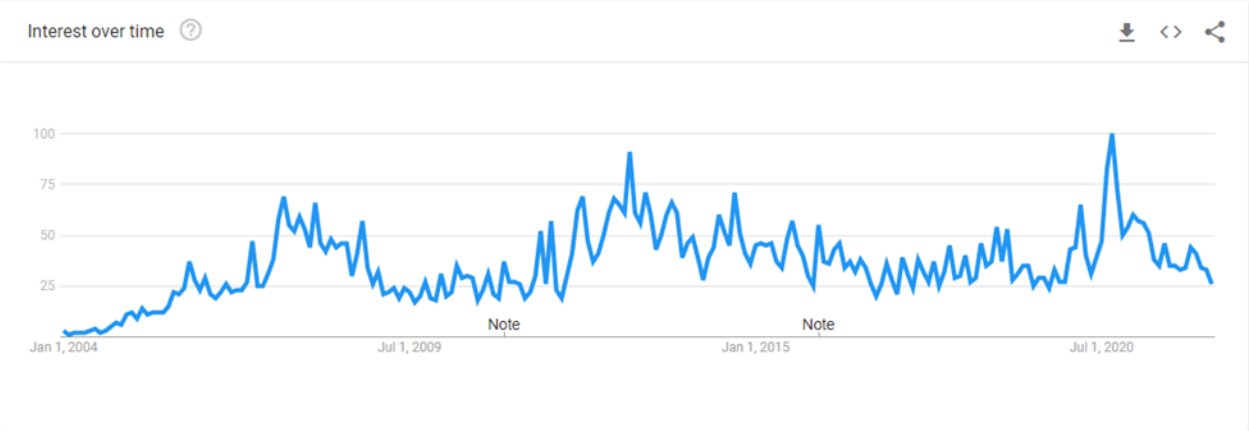


Figure 4: Google Trends Data Example

2.2.4. Future Tasks

Initially we wanted to use a doomsday/market sentiment metric and also feed this to the GRU based neural network, we planned on doing this by analyzing news articles and maybe Tweets and using the VADER (Valence Aware Dictionary and Sentiment Reasoner) library in python. When attempting to gather the large amount of data needed for this task, we ran into the problem of bot detection again from the websites where we were sourcing our articles from. We initially thought it was a cooldown problem but upon further inspection we realized that for the websites we were using any kind of navigation on the page was getting detected and not only that, but script usage was frowned upon in their terms of service. When trying to scrape tweets we also ran into the problem that Twitter frowns upon script usage, but they do offer an API to do this sort of thing. The public access version of this API was quite limited, and we would not have been able to get the number of tweets that are needed for this kind of analysis. They have made available an academic version of their API which we have applied for and are currently waiting to hear back from Twitter regarding this.

Depending on whether some of the data points show correlation with the trajectory of a specific stock's price we may change our data points or add new data points. This will be done in ECEN 404 when further optimizing the GBSPS.

2.3. Validation

2.3.1. Handling Empty Datasets Without Program Interruption

For some stocks, the data publicly available is very limited (especially for insider trading), and for this we initially were running into issues where the insider trading portion of the web scraper was returning errors. After figuring out what the error was, we realized that even when a stock didn't have data published and available on the page the program was still trying to retrieve it. This was leading to erroneous behavior. Upon realizing this issue, we further updated this module to include methods of checking if the data was there or not on the page before trying to retrieve it and if not appending NANs to the .CSV file.

2.3.2. Correct Data Aligned With Corresponding Date In CSV File

Ensuring that the data retrieved was appended to the same corresponding date at which the data was recorded was briefly an issue when implementing the Google Trends module. The problem was in how we were assigning the order of the data points to the list that was to be appended to the file. For some assignments we weren't comparing the full dates which corresponded to the retrieved data to the dates in the .CSV file but only the months and this would cause incorrect data to be appended to wrong dates in the .CSV file. This led to us improving the data assignment in all modules to check the complete date so no erroneous assignments will come about in this manner.

2.3.3. No Duplicate Data Appended to CSV File

Another problem faced was that we were appending duplicate columns to the .CSV file when appending all the retrieved data. In our case the issue was the naming of some of our pandas DataFrames and the ordering of how we appended them to the file (We append everything to the stocks file at the end of every loop). We have since fixed this issue and have validated that this will not be the case anymore.

2.4. Subsystem Conclusion

During the designing and developing of the data scraping subsystem we have primarily used HTML web scraping techniques in order to get the data we need from online sources. The design of this subsystem allows for data retrieval of any generic stock ticker that has information publicly available on the previously mentioned websites. This subsystem works as designed/expected and has the ability to change as we further optimize the GBSPS and find out what attributes correlate more strongly with price trajectory.

3. FPGA Subsystem Report

3.1 Subsystem Introduction:

3.1.1 Overview:

The FPGA subsystem is responsible for creating and maintaining an FPGA on which the other two subsystems will run on. The main technical challenge of this subsystem was answering the question of how to run a GRU on an FPGA. Our answer was to embed a Linux kernel onto a ZCU104 Xilinx FPGA which would allow it to run Python programs. These Python programs will then be able to utilize advanced machine learning libraries thus substantially reducing the time necessary to create a GRU.

3.2 Subsystem Details:

3.2.1 Design Choices:

At the beginning of this project we decided to use a Zybo-Z7 FPGA board which was manufactured by Digilent. We chose this board due to the fact that we were already very familiar with this specific board since we have used it in a previous class. However, we found out that the Z7 utilized an ARMv7 architecture which Python's machine learning libraries could not run on. Thus, we had to choose a new FPGA to embed Linux upon.

Taking into account what the Z7 was lacking we chose the Xilinx’s ZCU104 FPGA to embed Linux on. We chose this FPGA for a multitude of reasons, the first and most important being that it utilized an ARMv8 architecture. ARMv8 is an architecture that many devices today utilize and thus supports many of Python's ML (Machine Learning) libraries which we used to create the GRU. Secondly, the ZCU104 was chosen because it was the most cost-effective FPGA available for purchase. All other boards that utilized an ARMv8 architecture were vastly more expensive than the ZCU104 with the second cheapest FPGA the ZCU106 having a total cost of around $3,300. Finally, the last reason for using the ZCU104 was that it was created by Xilinx. Xilinx is the leading manufacturer for FPGAs and thus would be automatically compatible with PetaLinux, unlike the Z7.

Figure 5: ZCU104 Board

3.2.2 Procedures Used to Embed Linux:

Embedding Linux firstly requires a hardware description file in order for PetaLinux to create the boot image. To create this file first, create a project in Vivado and select the FPGA board that is being used. Then create a block design of the FPGA in Vivado.

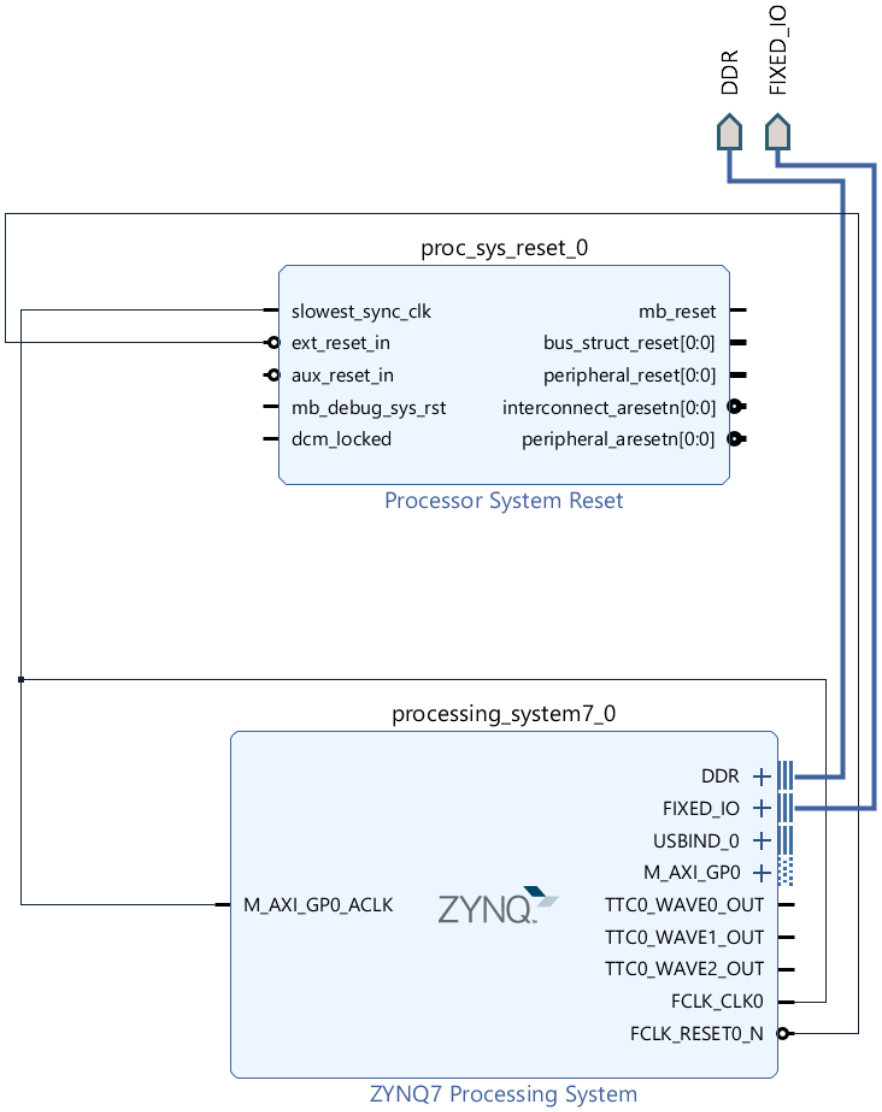


Figure 6: FPGA Block design

Once the block diagram was created, next use Vivado to generate the bitstream. With the bitstream generated then export the hardware to a .xsa file (note: this required Vitis Vidado not the Vivado standalone). The .xsa file is the hardware description file that PetaLinux uses to generate the custom boot image. With the .xsa file in hand, then create a PetaLinux project and set the hardware configuration to the file created. Next, configure the PetaLinux project to our exact specifications. Then build the PetaLinux project and create the boot image.

To use the boot image take a micro SD card and write two partitions onto it. The first partition is the boot partition where the boot image will be placed. This boot partition is formatted in the FAT32 file system format and would be read when the FPGA was turned on to start the boot process. Next, create a second partition formatted to ext4 file system this will be the root file system (rootfs) partition.

This partition is where the FPGA will write and read from and dictates which version of Linux the FPGA will be used. We chose to not use the default rootfs that PetaLinux automatically generates because it is extremely limited in what programs can be installed since it can only utilize packages and programs that Xilinx includes. Thus, we chose to use an aarch64 version of Debian 11.3 which allows the embedded Linux kernel to install packages that Xilinx's Linux does not have i.e: Python's virtual environment package. Once the rootfs was chosen then transfer it to the second partition.

With the rootfs and the boot partitions now completed the micro SD card can now be used to run Linux on an FPGA.

3.2.3 Testing Embedded Linux:

The embedded Linux runs on the ZCU104 and successfully runs simple python scripts as can be seen in Figure 7 below.

```
print("hello world")
x = 5
y = 4
print(x)
print(y)
z = x + y
print(z)
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(proj) debian@arm:~$ ls
fpgadown  proj  projls  simp.py
(proj) debian@arm:~$ python3 simp.py
hello world
5
4
9
(proj) debian@arm:~$
```

Figure 7: Testing Simple Python 3.7 Scripts

Not only has Linux been successfully embedded onto an FPGA but it can now also run simple Python scripts. However, running simple Python scripts is not what needs to be accomplished. Thus, small test scripts from the GRU subsystem have also been run on the FPGA to see whether the FPGA can handle any machine learning scripts.

As can be seen in Figure 8 the FPGA has successfully run the test scripts that utilize the machine learning libraries that the GRU uses. But, it seems that these scripts also require large amounts of memory that may in the future go beyond the FPGA's capabilities. Which means in the future we may need to increase the amount of swap space to avoid running out of memory in the future for more demanding scripts.

```
# coding: utf-8

# In[1]:


import tensorflow as tf

mnist = tf.keras.datasets.mnist # 28x28 images of hand written digits 0-9

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = tf.keras.utils.normalize(x_train, axis=1)  #scaling data between 0 to
1, this is not necessary but makes training easier
x_test = tf.keras.utils.normalize(x_test, axis=1)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten()) #

model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu ))  #128 neurons ann
d using relu = rectified linear layer 1 tf.nn.relu
model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))  # layer 2
@@@
(proj) debian@arm:~/fpgadown$ python3 pakr.py
2022-04-29 00:11:00.605856: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 188160000 exceeds 10% of free system memory.
Epoch 1/3
1875/1875 [==============================] - 20s 10ms/step - loss: 0.2640 - accuracy: 0.9232
Epoch 2/3
1875/1875 [==============================] - 18s 10ms/step - loss: 0.1068 - accuracy: 0.9662
Epoch 3/3
1875/1875 [==============================] - 19s 10ms/step - loss: 0.0732 - accuracy: 0.9767
313/313 [==============================] - 3s 6ms/step - loss: 0.1005 - accuracy: 0.9693
0.10053471475839615 0.9692999720573425
(proj) debian@arm:~/fpgadown$
```

Figure 8: Testing ML Libraries

3.3 Subsystem Validation:

3.3.1 Validation:

	2/28/22	3/7/22	3/14/22	3/21/22	3/28/22	4/4/22	4/11/22	4/18/22	4/25/22	ECEN 404
Fully install and test the embedded Linux										
Start Designing case and install cooling system										
Fully design and print case and mount onto FPGA										

Table 2: FPGA ECEN403 Execution Plan

Table 3 is the current validation plan for the FPGA subsystem. As can be seen, from the table the most important task of embedding Linux has been completed. With that said some of these tasks have been neglected which may prove detrimental in the future. This is due to the fact that the tasks that have been neglected are the ones that keep the FPGA safe from damage.

3.3.2 Future Tasks:

During the next semester scripts will be taken from both the GRU and data scraping substem and implemented on the FPGA itself. Implementing these substems will be challenging because of the FPGA's limited resources. However, as has been previously mentioned the FPGA has proven that it can run at least a basic version of these scripts. Once the scripts are implemented on the board training of the GRU on the FPGA as is instructed by our sponsor. While the GRU is being trained we will also use our expertise to prevent the FPGA from overexerting and prevent it from damaging itself. Finally, during the next semester, we will start designing a case for the FPGA.

3.4 Subsystem Conclusion:

In conclusion, the FPGA subsystem is on track to be able to start training the GRU by next semester. Although the late switching of boards has delayed the project the FPGA is fully embedded with Debian using PetaLinux. The embedded Linux has also proven that it can utilize the necessary libraries that the GRU needs. Thus, next semester training the GRU on the FPGA can start immediately.

4. GRU Architect Subsystem Report

4.1 Subsystem Introduction:

The GRU Architect subsystem is designed to receive the data available from the Data Scraper Subsystem about any selected stock and process the data into a usable format for model training. That processed data is then fed into the machine learning model to train multiple variations of both model structure and data attributes. The implementation of machine learning being used is a Gated Recurrent Unit (GRU), which is a recurrent neural network. The subsystem will save and use the trained GRU model that produced the most accurate predictions during its operation to produce predictions for the user. The key feature of this subsystem is its ability to change model parameters and data attributes so that it can find an optimally trained model for any given stock.

4.2 Subsystem Details

This GRU Architect Subsystem is written in Python 3.7.4 and makes use of an available GRU unit from the Python machine learning library Keras. The subsystem faces two main challenges: limited data, and restricted generalization. After processing the available data the subsystem is often left with an average of 2,300 data points usable for training. Due to this restricted amount of data the system is prone to overtraining, which can result in predictions that are skewed in a given direction per the characteristics used. Additionally, the operation of this subsystem must run with minimal processing overhead as to not strain the FPGA while remaining within its operating conditions/bounds.

4.2.1 Subsystem Operation

The high-level operation of the code is depicted below in Figure 9.

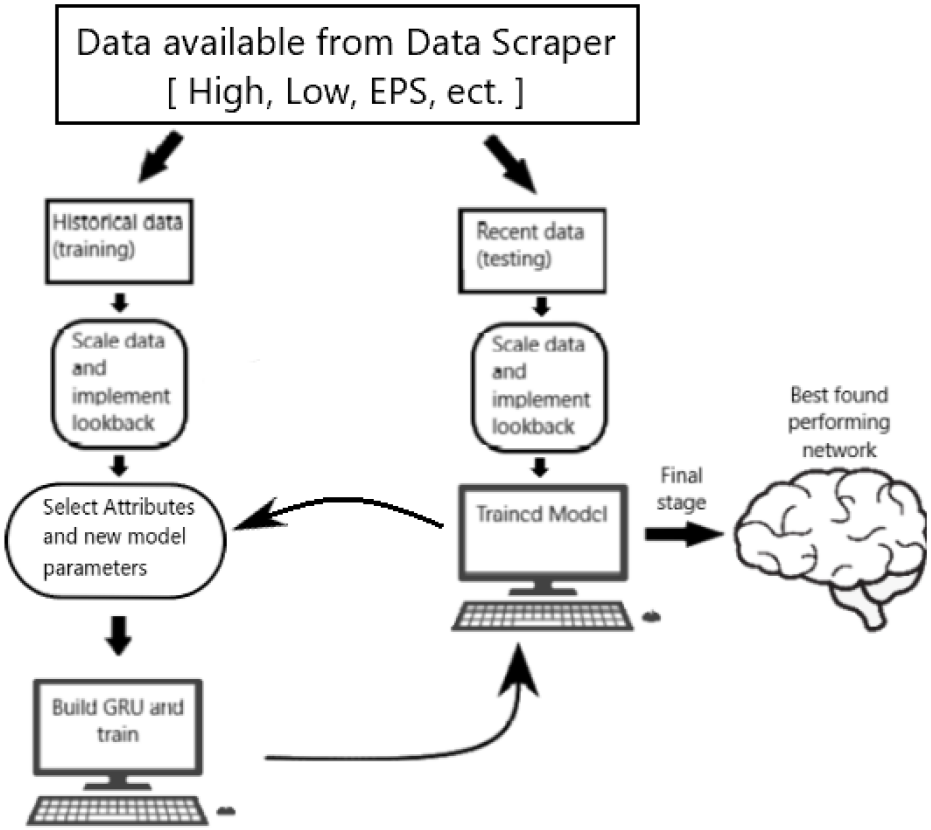


Figure 9: GRU Architecture Code Flow Chart

The operation of the subsystem begins with the data received from the Data Scraper Subsystem which is categorized into two sets, one for training and one for testing. This split is done at the most recent year so that a substantial amount of the data is available to train with. A scaling factor will then be determined that is used for scaling and descaling both the training and testing sets of data. The newly scaled data sets then have a lookback set applied to them, this is a required feature in order for GRU models to be able to appropriately learn. The number of timesteps applied in this lookback stage is the previous 54 days per data point. Thus the input array size of the model varies by the format: (Size, Lookback days, Number of Attributes). From this point the code begins its model building and training phase in a brute force manner to find the best combination of data and model structure to predict the closing price of the selected stock. This brute force loop engages in selecting differing data attributes, learning rates, and neurons to build and train the model. The current operation has 20 different combinations of data attributes to train with. In the testing of each model and attribute combination two metrics are used: root-mean-square error (RMSE) and Directional Accuracy. The best performing model is then saved to be used for predictions.

4.2.2 Subsystem Data Attributes:

The following attributes are those that are currently available to the GRU Architect from any given stock:

[High, Low, Open, Close*, EPS*[1], Day, %Change*[2], Inflation Rate[*3], Insider Trading*[4]]

*Close is used as the value to predict and is used as the Y value in training

*[1] : EPS - Earnings per share in a 54 day period (1 trading quarter)

*[2] : %Change - The percentage increase/decrease in a stocks value per day

*[3] : Inflation Rate - The percentage of inflation of the US dollar

*[4] : Insider Trading has not been implemented/tested yet, but is available

All of these attributes must be scaled between 0 to 1 or -1 to 1 in order to properly train the GRU model and not disrupt its gradient descent calculations. Scaling is a very impactful factor in a GRU models ability to generalize its predictions, if a model encounters a situation in which its predictions are being made using a scaled dataset that goes over 1, it is prone to make predictions much lower or higher than the real value and will often plateau over periods in which the values are above this threshold. To mitigate this, the scaling method implemented leverages the lowest and highest price in the training dataset to calculate an appropriate scaling factor. This method accounts for the historic growth to make a potential maximum of the stock to scale the dataset attributes by. This scaling method is used only on price related attributes (High, Low, Open, Close) of the stock. The calculation is done by the following:

$$\text{Maximum Value} = \frac{\left(\frac{Max(High)}{Min(High)}\right)}{Number\ of\ Years\ in\ Dataset} \cdot Max(High)$$

$$\text{Scaled datapoint} = \frac{Datapoint - Minimum}{Maximum\ Value - Minimum} \text{ ; Minimum = 0}$$

```
MaxData = dataset[:Year]['High'].max() #grba the max value out of the high
#print(MaxData)
MinData = dataset[:Year]['High'].min() #grab the min value out of the high
increase_max = (MaxData/MinData)/Years # This aids to find a "max" value the stock can exist at, this is needed for scaling
#this is calculation is the high growth potential from min to max over train data years.
Factor = increase_max*MaxData #save this calucation to use for the sclaers
```

```
#The Scaler will take in the selected columns and scale them between 0-1 using the data normalization formula
def Scaler(arr, ADJMAX):
    minimum = 0  #All stocks can level off at worthless ($0.00)
    maximum = ADJMAX  #This will be changed so that an imperical estimation of growth max can be made.
    dummy = 0
    scaled_list = []
    for x in range(len(arr)):
        dummy = (arr[x] - minimum)/(maximum-minimum)
        scaled_list.append(dummy)
    #print(scaled_list)
    return scaled_list

#The Decaler will take in the selected columns and descale them out of 0-1 using the data normalization formula inverse
def Descaler(arr, ADJMAX):
    minimum = 0
    maximum = ADJMAX
    dummy = 0
    descaled_list = []
    for x in range(len(arr)):
        dummy = (arr[x]*(maximum-minimum)) + minimum
        descaled_list.append(dummy)
    return descaled_list
```

Figure 10: Data Scaling Code Capture

Percentage change (%Change) is a calculated attribute with the available data. This metric makes use of the close price on a given day in comparison to the previous day's close. This metric is purposely left in decimal form so that it fits the constraint of 0 to 1 for the GRU. This calculation is done by the following:

$$\%Change = \frac{(Close[x] - Close[x-1])}{Close[x-1]} \text{ ; x = Day}$$

```
def percIncr(data): #this will measure the percentage increase/decrease day to day
    percents = []
    for z in range(1,len(data)):
        per = (data[z]-data[z-1])/data[z-1]   # per[%] = (Day2 - Day1) / Day1
        percents.append(per)
    return percents
```

Figure 11: Percent Change Code Capture

The attribute of day, after its usage in other calculations, is converted into a sinusoidal signal so that it is constrained between -1 to 1. This method of scaling is most appropriate since the day of the year is a cyclical attribute without the potential to ever go beyond the bounds. The calculation is done as follows:

$$\text{Day} = sin(\ (day\ of\ year)\ \cdot \frac{2\pi}{365.24}\ )$$

The program implementation of the formula is seen in Figure 12.

```
day_sin_signal = np.sin(dataset['Day']*(2*np.pi/365.24)) #convert the days into a proper signal
#plt.plot(day_sin_signal) #verify sinusoidal day list
#plt.show()
```

Figure 12: Sinusoidal Days Code Capture

4.2.3 Subsystem Prediction Accuracy Metrics:

In the testing step of the trained models, two accuracy metrics are used: root-mean-square error (RMSE) and Directional Accuracy. In the subsystem root-mean-square error is used to measure the differences between the values the GRU predicted and the actual values of the stock. Using this accuracy metric reveals the aggregated magnitude of errors made over a predicted dataset. The closer this value is to zero, the better the predictions that were made. RMSE is sensitive to scale and thus the predictions and actual values must be input in their descaled forms. Calculation of RMSE is done using the following:

$$RSME = \sqrt{\frac{\Sigma_{i=1}^{N}(x_i - \widehat{x_i})^2}{N}}$$; x_i = actual value , $\widehat{x_i}$ = predicted value , N = data points

The second accuracy metric used in the subsystem is Directional Accuracy. This metric quantifies the number of times in a percentage over a dataset that the model predicted

a stock's value would go a certain direction and the actual value did correspond correctly to this movement. The pseudo code of this metric's calculation can be seen below:

```
For Number of Datapoints:
        if (Predicted > Previous Day Predicted) and (Actual > Previous Day Actual)
                Correct += 1
        else if (Predicted < Previous Day Predicted) and (Actual < Previous Day Actual)
                Correct += 1
        else if (Predicted = Previous Day Predicted) and (Actual = Previous Day Actual)
                Correct +=1
        else if (Predicted =  Actual)
                Correct +=1
```

Directional Accuracy [%] = $\frac{Correct}{Number\ of\ Datapoints} \cdot 100$

4.2.4 GRU Model Building:

The building of each model allows for variations in the number of: lookback, learning rate, and the neurons of the internal layers. The GRU layers use the hyperbolic tangent function as the activation because the internal design of a GRU neuron demands it. Dropout is used for each layer in order to mitigate the risk of overtraining. The optimizer used is Adam with a loss metric of mean squared error. The function that builds the GRU models is seen below:

```
def GRUModelBuild(arrX, arrY, arrDem, daysback, LR, NeuronsIntern):
    # The GRU architecture
    ModelGRU = Sequential()

    # First GRU Layer with Dropout regularisation
    ModelGRU.add(GRU(units=50, input_shape=(daysback,arrDem), return_sequences=True, activation='tanh'))
    ModelGRU.add(Dropout(0.1))   # activation Tanh is used as that is native to the GRU model, returning seq will keep
                                 # the tensors shapes consistent between inputs

    # Second GRU Layer
    ModelGRU.add(GRU(units=NeuronsIntern, return_sequences=True, activation='tanh' ))
    ModelGRU.add(Dropout(0.2))

    # Third GRU Layer
    ModelGRU.add(GRU(units=NeuronsIntern, return_sequences=True, activation='tanh' ))
    ModelGRU.add(Dropout(0.2))

    ModelGRU.add(GRU(units=NeuronsIntern, activation='tanh'))
    ModelGRU.add(Dropout(0.1))

    # The output layer
    ModelGRU.add(Dense(units=1)) # The final output is a 1D array of the predicted price point.
                                 #A relu activation has been tested here before and proves to give better occasionally better

    opt = keras.optimizers.Adam(learning_rate=LR)
    ModelGRU.compile(optimizer=opt , loss='mean_squared_error')
    #Loss of MSE is best in this scenario as it is a measure of how far off the system is from the real value,
        # it acts similar to an averaged standard deviation. Closer to zero is better
    return ModelGRU
```

Figure 13: GRU Model Builder Code Capture

4.3 Subsystem Validation:

This section will present each step of operation of the subsystem from the previously shown flow chart and give code print outs to show the validation of each step operating appropriately. The values shown are dependent on the stock being predicted and are subject to change.

The ability of the program to properly organize and prepare the data to be used for model training is the first step of validating proper operation. If the data is not prepared properly, the training of the models will fail.

Date	High	Low	Open	Close	EPS	IR	Day
2008-12-31	3.133571	3.047857	3.070357	3.048214	0.06	0.0384	366
2009-01-02	3.251429	3.041429	3.067143	3.241071	0.06	-0.0036	2
2009-01-05	3.435000	3.311071	3.327500	3.377857	0.06	-0.0036	5
2009-01-06	3.470357	3.299643	3.426786	3.322143	0.06	-0.0036	6
2009-01-07	3.303571	3.223571	3.278929	3.250357	0.06	-0.0036	7
...	...	...	...	...	...	...	...
2021-12-23	176.850006	175.270004	175.850006	176.279999	2.10	0.0470	357
2021-12-27	180.419998	177.070007	177.089996	180.330002	2.10	0.0470	361
2021-12-28	181.330002	178.529999	180.160004	179.289993	2.10	0.0470	362
2021-12-29	180.630005	178.139999	179.330002	179.380005	2.10	0.0470	363
2021-12-30	180.570007	178.089996	179.470001	178.199997	2.10	0.0470	364

Table 3: Data Received GRU Code Print Out

Date	High	Low	Open	Close	EPS	IR	PercentChange	Day
2009-01-02	0.015867	0.014842	0.014967	0.015816	0.06	-0.0036	0.063269	0.034422
2009-01-05	0.016763	0.016158	0.016238	0.016484	0.06	-0.0036	0.042204	0.085965
2009-01-06	0.016935	0.016102	0.016722	0.016212	0.06	-0.0036	-0.016494	0.103102
2009-01-07	0.016121	0.015731	0.016001	0.015861	0.06	-0.0036		
2009-01-08	0.016234	0.015692	0.015760	0.016156	0.06	-0.0036		
...	...	...	...	...	...	...		
2021-12-23	0.863013	0.855303	0.858133	0.860231	2.10	0.0470		
2021-12-27	0.880434	0.864087	0.864184	0.879995	2.10	0.0470		
2021-12-28	0.884875	0.871211	0.879165	0.874920	2.10	0.0470		
2021-12-29	0.881459	0.869308	0.875115	0.875359	2.10	0.0470		
2021-12-30	0.881166	0.869064	0.875798	0.869601	2.10	0.0470		

Table 4: Data Processed GRU Code Print Out

The processed dataset seen in Table 5 verifies the following: NaNs removed, data scaled, Inclusion of Sinusoidal Days and Percentage Change. Following this, the verification that the days have been converted into a sinusoidal signal can be seen in Figure 14.

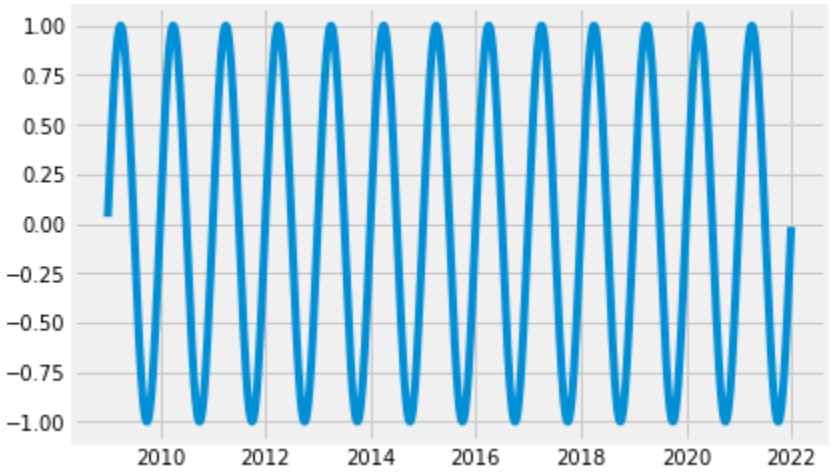


Figure 14: Verification of Sinusoidal Days Attribute

The next step in operation the code builds a dataset with selected attributes, currently the code creates twenty different combinations of attributes to train models with. One of these attribute combinations can be seen below in Table 6.

Date	High	Low	Open	EPS	Day
2009-01-02	0.015867	0.014842	0.014967	0.06	0.034422
2009-01-05	0.016763	0.016158	0.016238	0.06	0.085965
2009-01-06	0.016935	0.016102	0.016722	0.06	0.103102
2009-01-07	0.016121	0.015731	0.016001	0.06	0.120208
2009-01-08	0.016234	0.015692	0.015760	0.06	0.137279
...	...	...	...	...	...
2021-12-23	0.863013	0.855303	0.858133	2.10	-0.137279
2021-12-27	0.880434	0.864087	0.864184	2.10	-0.068802
2021-12-28	0.884875	0.871211	0.879165	2.10	-0.051620
2021-12-29	0.881459	0.869308	0.875115	2.10	-0.034422
2021-12-30	0.881166	0.869064	0.875798	2.10	-0.017213

Table 5: Example of Selected Attributes Code Print Out

The implementation of lookback is done by the following step, this will rebuild the array so that each given attribute has a selected number of previous days values attached to it so that the GRU has timesteps available. Below is the verification that the lookback

array is correctly built. This output demanded that the array has 54 lookback days on 5 attributes.

```
Array Size: (2463, 54, 5)
[ [[0.07354532 0.07874358 0.07456265 0.02027027 1.        ]
  [0.08609127 0.08511256 0.07986808 0.02027027 0.        ]
 [0.0840003  0.09010639 0.08617721 0.02027027 0.00824176]
                          ...
 [0.01088761 0.00665848 0.00329796 0.02027027 0.2032967 ]
 [0.0242267  0.02156763 0.01577285 0.02027027 0.20604396]
 [0.0263177  0.0279366  0.03111557 0.02027027 0.20879121]]

 ……..

[[0.85348615 0.84946079 0.8382564  0.66216216 0.76923077]
 [0.86386903 0.85242817 0.86571554 0.66216216 0.77197802]
 [0.83264831 0.80610847 0.8257098  0.66216216 0.77472527]
                          ...
 [0.87857805 0.85836284 0.85209345 0.66216216 0.98351648]
 [0.89061939 0.87059416 0.87503579 0.66216216 0.98626374]
 [0.90352582 0.90041258 0.89661602 0.66216216 0.98901099]] ]
```

Figure 15: Array with Lookback Structure Verification

The final stage of the subsystem builds a GRU model that is trained and then produces a graphic visualization of the closing price predictions it made against the actual values. It will also provide the RMSE and directional accuracy of each model attribute combination for a given model.

The model constructed to produce the following predictions is seen in Figure 16, this model is just one of many that is built throughout the operation of the subsystem.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 54, 50)            8550

 dropout (Dropout)           (None, 54, 50)            0

 gru_1 (GRU)                 (None, 54, 80)            31680

 dropout_1 (Dropout)         (None, 54, 80)            0

 gru_2 (GRU)                 (None, 54, 80)            38880

 dropout_2 (Dropout)         (None, 54, 80)            0

 gru_3 (GRU)                 (None, 80)                38880

 dropout_3 (Dropout)         (None, 80)                0

 dense (Dense)               (None, 1)                 81

=================================================================
Total params: 118,071
Trainable params: 118,071
Non-trainable params: 0
_________________________________________________________________
```

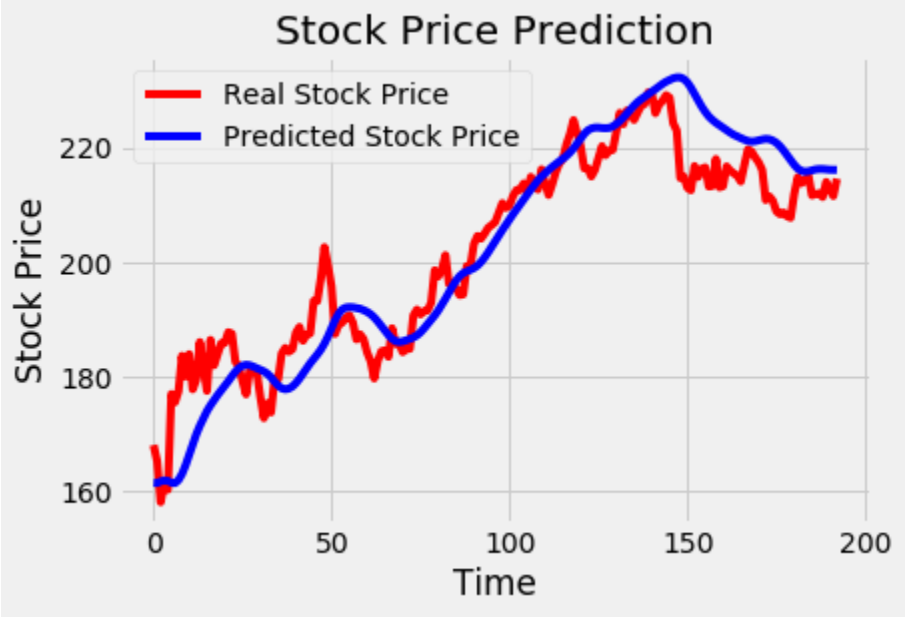
Figure 16: Example Build of GRU Model Print Out

McDonalds (MCD) Stock		
Attributes	RMSE	Directional Accuracy [%]
H, L, O, EPS, Day	5.547	54.69
H, L, O, Day	7.994	55.21
H, L, Day	6.7	56.25
H, L, O	5.809	55.21
H, L	17.342	55.21
H, L, EPS, Day	7.315	54.17
H, L, EPS, Day, IR	6.158	54.69
H, L, IR	6.627	55.73
H, L, IR, Day	14.53	55.73
H, L, O, EPS, %Change	6.552	54.17
H, L, O, Day, %Change	5.022	50.52

H, L, EPS, Day, IR	6.989	57.81
H, L, Day, %Change	5.722	51.56
H, L, O, %Change	7.747	50.00
H, L, Day, %Change, IR	6.129	51.55
H, L, O, Day, %Change, IR, EPS	6.174	53.65
H, L, O, Day, %Change, IR	6.226	53.65
H, L, EPS, Day, %Change, IR	6.696	55.73
H, L, Day, %Change, IR	4.353	49.48

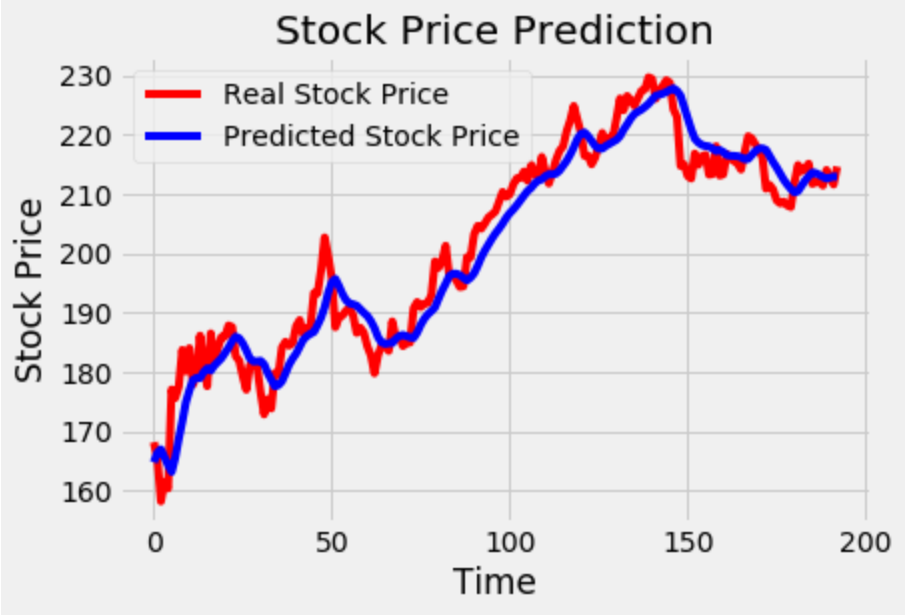
Table 6: Output Data From Model Training on MCD

The data highlighted is the attribute combination of this model that can be considered the best performing and will be saved for prediction usage if another later model structure does not outperform it. Below, two of the visualizations of these predictions can be seen with their corresponding attributes and performance in Figures 17 and 18.



The root mean squared error is 6.988665041397858.
The directional accuracy is 57.81%
Attributes: ['High', 'Low', 'EPS', 'Day', 'IR']

Figure 17: Prediction Visualization 1



The root mean squared error is 4.3538526259066845.
The directional accuracy is 49.48%
Attributes: ['High', 'Low', 'IR', 'PercentChange']

Figure 18: Prediction Visualization 2

4.4 Subsystem Conclusion:

All parts of the subsystems functioned correctly. The subsystem proficiently generated predictions and found optimal models to use for future predictions of any given stock. When interfaced with other subsystems it will be able to handle any incoming data sets to produce predictions while running onboard the FPGA. The current state of the system still needs more attributes available to it and an improvement in training methods. It can be seen that the models fail to adjust their predictions when large swings in price occur, including times in which the price oscillates such as in Figure 19.

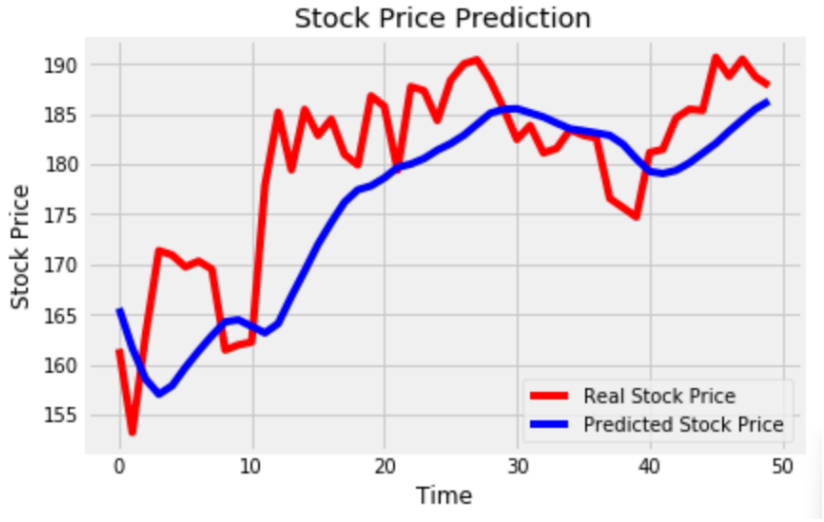


Figure 19: Prediction Visualization 3

Seen in Figure 20 below the sudden large drop in price, known as a “falling knife”, causes the GRU to make heavily skewed initial predictions after the drop. This falling knife is an occurrence seen market wide during March of 2020 due to Covid-19 lockdowns, in order for the GRU to better train for this type of situation the attributes of market sentiment and trade volume will need to be added when fully available.

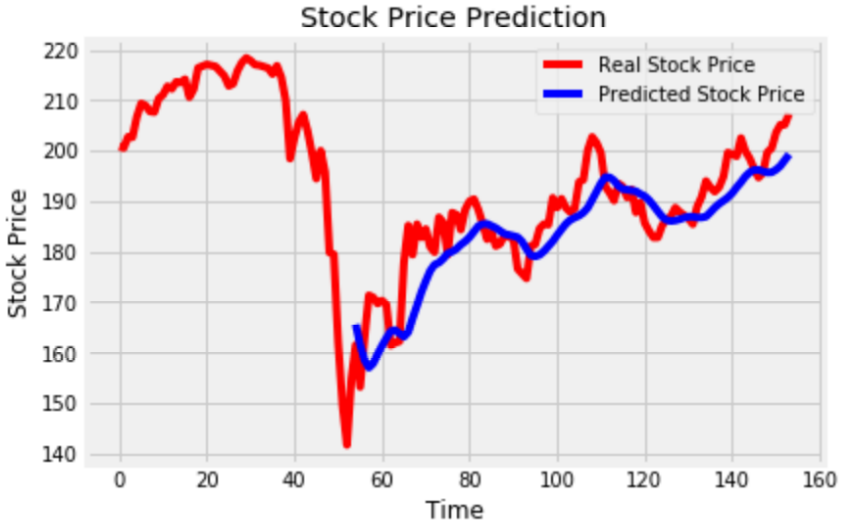


Figure 20: Prediction Visualization 4

To further improve the subsystem it will soon be able to leverage data about market sentiment in its prediction. As well as this, a future version will use reinforcement learning with an agent that can either buy, sell, or hold a stock in order to learn when to make decisions in a beneficial manner.