

**ISTANBUL TECHNICAL UNIVERSITY**  
**ELECTRICAL-ELECTRONICS FACULTY**

**IMPLEMENTATION OF HIGH FREQUENCY TRADING TECHNOLOGY ON  
FPGA**

**SENIOR DESIGN PROJECT**

**Behiç ERDEM**  
**Rana TİLKİ**  
**Armin ASGHARİFARD**

Uygundur  
Berna Örs Yalçın



**ELECTRONICS AND COMMUNICATION ENGINEERING  
DEPARTMENT**

**JUNE 2023**

**ISTANBUL TECHNICAL UNIVERSITY**  
**ELECTRICAL-ELECTRONICS FACULTY**

**IMPLEMENTATION OF HIGH FREQUENCY TRADING TECHNOLOGY ON  
FPGA**

**SENIOR DESIGN PROJECT**  
**INTERIM REPORT**

**Behiç ERDEM**  
**040170213**

**Rana Tilki**  
**040180741**

**Armin ASGHARİFARD**  
**040190912**

**JUNE 2023**

**ELECTRONICS AND COMMUNICATION ENGINEERING  
DEPARTMENT**

**Prof. Dr. Sıddıka Berna Örs Yalçın**

We are submitting the Senior Design Project Interim Report entitled as “Implementation Of High Frequency Trading Technology On FPGA”. The Senior Design Project Interim Report has been prepared as to fulfill the relevant regulations of the Electronics and Communication Engineering Department of Istanbul Technical University. We hereby confirm that we have realized all stages of the Senior Design Project Interim Report by ourselves, and we have abided by the ethical rules with respect to academic and professional integrity .

**Behiç ERDEM**  
040170213



**Rana TILKI**  
040180741



**Armin ASGHARİFARD**  
040190912



## **FOREWORD**

We are three students in Istanbul Technical University Electronics and Communications Engineering program. The purpose of this thesis is to do comprehensive research on the technologies of High-Frequency Trading in the stock market and how they are implemented to provide very fast automated trading platform.

We sincerely thank Prof. Dr. Berna Örs Yalçın from Istanbul Technical University and İsmail Hakki Topcu from Electra IC company, based in Istanbul, for mentoring and guiding us through this project.

The motivation to start this project rised from the fact that this technology is implemented on FPGA, in communication with a CPU to implement the desired trading strategy and run it directly on the stock exchange server through the FPGA system. FPGA design and embedded programming is what we are interested in and since so much financial profits can be made if the system is successfully implemented, we decided to work and research on this topic.

In this report, after going through an introduction on how this system works, each part of this FPGA system, and the communication and messaging protocols used in this system will be discussed, seperately. The key points to keep in mind for now is that the FPGA system receives live stock market data through ethernet connection with the stock exchange server; this data is processed in real-time and organized into sorted tables, which we call Order Books; The CPU part will receive and process these Order Books and decide on which stock and what quantity will be bought or sold; These decisions will be informed to the FPGA system, and the FPGA system will transfer these orders to the stock exchange server.

The excitement of going through this project is the vast amount of things we learned. We thank all the readers for their interest in our thesis and we invite you to delve into the following chapters.

## TABLE OF CONTENTS

	<u>Page</u>
<b>FOREWORD</b> .....	v
<b>TABLE OF CONTENTS</b> .....	vi
<b>ABBREVIATIONS</b> .....	vii
<b>SYMBOLS</b> .....	viii
<b>LIST OF TABLES</b> .....	ix
<b>LIST OF FIGURES</b> .....	x
<b>SUMMARY</b> .....	xi
<b>ÖZET</b> .....	xii
<b>1. INTRODUCTION</b> .....	1
1.1. Purpose of The Project.....	2
1.2. Literature Review.....	2
<b>2. FPGA Implementation for HFT</b> .....	5
2.1. HFT Protocols.....	6
2.1.1. OUCH Protocol.....	6
2.1.1.1. OUCH Protocol Message Types.....	7
2.1.2. ITCH Protocol.....	10
2.1.2.1. ITCH Protocol Message Types.....	10
2.2. TCP/IP and UDP.....	14
2.3. FPGA Board and Tools Specifications.....	15
2.4. Intellectual Property Cores.....	19
2.4.1. AXI Ips.....	19
2.4.2. AXI 1G/2.5G ethernet subsystem.....	19
2.4.3. AXI interconnect.....	20
2.4.4. AXI direct memory access – DMA.....	21
2.4.5. AXI UARTLite.....	21
2.4.6. Microblaze.....	22
<b>3. RESULT AND ANALYSIS</b> .....	23
3.1. Non-FPGA Environment Experiment.....	23
3.2. Echo Server on FPGA.....	25
3.3. TCP Server on FPGA.....	29
<b>4. REALISTIC CONSTRAINTS AND CONCLUSIONS</b> .....	31
4.1. Application Area of Project.....	31
4.2. Realistic Constraints.....	32
4.3. Social, environmental and economic impact.....	33
4.4. Health and Safety Concerns.....	34
4.5. Standards.....	34
4.6. Cost Analysis.....	36
4.7. Conclusions.....	36
4.8. Future Work and Recommendations.....	37
<b>5. REFERENCES</b> .....	39

## **ABBREVIATIONS**

<b>HFT</b>	<b>: High Frequency Trading</b>
<b>OUCH</b>	<b>: Order, Update, Cancel and Hit</b>
<b>ITCH</b>	<b>: Information Technology Channel</b>
<b>FPGA</b>	<b>: Field Programmable Gate Array</b>
<b>TCP</b>	<b>: Transmission Control Protocol</b>
<b>AXI</b>	<b>: Advanced Extensible Interface</b>
<b>UDP</b>	<b>: User Datagram Protocol</b>
<b>DHCP</b>	<b>: Dynamic Host Configuration Protocol</b>
<b>IP</b>	<b>: Internet Protocol Address</b>
<b>EUR</b>	<b>: Euro</b>
<b>USD</b>	<b>: United States Dollar</b>
<b>TRY</b>	<b>: Turkish Lira</b>
<b>UART</b>	<b>: Universal Asynchronous Receiver/Transmitter</b>
<b>RISC</b>	<b>: Reduced Instruction Set Computer</b>
<b>HDL</b>	<b>: Hardware Description Language</b>
<b>VHDL</b>	<b>: Very High Speed Integrated Circuit Hardware Description Language</b>
<b>LWIP</b>	<b>: Light Weight Internet Protocol</b>
<b>BSP</b>	<b>: Board Support Package</b>
<b>PEP</b>	<b>: Python Enhancement Protocol</b>
<b>BSD</b>	<b>: Berkeley Software Distribution</b>
<b>FIFO</b>	<b>: First in First out</b>

## **SYMBOLS**

<b>MHz</b>	<b>: Mega Hertz</b>
<b>ns</b>	<b>: Nano Second</b>
<b>ms</b>	<b>: Mili Second</b>



## LIST OF TABLES

<b>Table 2.2</b> : ITCH New Order Order Book Directory Message.....	8
<b>Table 2.1</b> : OUCH decoding for new sell order message type.....	13

## LIST OF FIGURES

	<b>Page</b>
<b>Figure 2.1</b> :HFT implementaion block diagram of real environment.....	5
<b>Figure 2.2</b> : Example python OUCH Simulation.....	7
<b>Figure 2.3</b> : Example python ITCH Simulation.....	10
<b>Figure 2.4</b> : Nexys Video Trainer Board.....	16
<b>Figure 2.5</b> : AXI Ethernet Subsystem Block Design.....	20
<b>Figure 3.1</b> : Example order book.....	24
<b>Figure 3.2</b> : Example python order cook. Left column is the quantities, right column is the price. Every new block is represent different timestamp.....	25
<b>Figure 3.3</b> : Complete Block design of the ethernet subsystem.....	27
<b>Figure 3.4</b> : Localhost echoing with hardware.....	28
<b>Figure 3.5</b> : Serial output of Echo server.....	29

## **SUMMARY**

High-frequency trading (HFT) refers to a specific type of algorithmic trading in the financial market that involves rapid trade execution, extensive use of technology, and analysis of high-frequency financial data. It is characterized by its high speeds, frequent trading activity, the utilization of advanced algorithms, and short investment timeframes. HFT employs sophisticated computer algorithms and hardware to rapidly buy and sell stocks or other financial instruments within very short time intervals, often in seconds or fractions of a second.

Having a low-latency technology in implementing HFT is very important. HFT traders aim to capitalize on brief market inefficiencies, leading to not only their own profit but also an overall increase in market efficiency. However, the growing number of participants in the HFT sector attempting to exploit these opportunities means that only the initial traders to execute orders may benefit from a particular opportunity. This underscores the significance of latency, which is the first reason why it matters.

Furthermore, certain HFT strategies like latency arbitrage rely on the ability to access market data and execute orders faster than other investors. Successfully profiting from such activities necessitates real-time reaction to market events using low-latency and low-jitter market access. Even a millisecond reduction in latency can significantly enhance arbitrage profitability. Therefore, HFT strategies demand low-latency market access to receive market data promptly and transmit new orders efficiently.

Due to these reasons, and the hardware acceleration opportunities that FPGA can provide, FPGA is chosen as the primary infrastructure for HFT systems. The parallelism and determinism of FPGA solutions greatly enhance the speed of computing mathematical models and transmitting data to the matching engines of exchanges.

## ÖZET

Yüksek frekanslı ticaret (HFT), finans piyasasında hızlı işlem gerçekleştirmeyi, teknolojinin kapsamlı kullanımını ve yüksek frekanslı finansal verilerin analizini içeren belirli bir algoritmik ticaret türünü ifade eder. Yüksek hızları, sık alım satım faaliyetleri, gelişmiş algoritmaların kullanımı ve kısa yatırım zaman dilimleri ile karakterize edilir. HFT, hisse senetlerini veya diğer finansal araçları çok kısa zaman aralıklarında, genellikle saniyeler veya saniyenin kesirleri içinde hızla alıp satmak için sofistike bilgisayar algoritmaları ve donanımları kullanır.

HFT'nin uygulanmasında düşük gecikmeli bir teknolojiye sahip olmak çok önemlidir. HFT yatırımcıları kısa süreli piyasa verimsizliklerinden faydalanarak yalnızca kendi kârlarını değil aynı zamanda piyasa verimliliğinde genel bir artış sağlamayı amaçlamaktadır. Ancak, HFT sektöründe bu fırsatlardan yararlanmaya çalışan katılımcıların sayısının giderek artması, belirli bir fırsattan yalnızca emirleri gerçekleştiren ilk yatırımcıların faydalanabileceği anlamına gelmektedir. Bu da gecikmenin önemini vurgulamaktadır ki bu da gecikmenin neden önemli olduğunun ilk nedenidir.

Ayrıca, gecikme arbitrajı gibi belirli HFT stratejileri, piyasa verilerine erişme ve emirleri diğer yatırımcılardan daha hızlı gerçekleştirme becerisine dayanır. Bu tür faaliyetlerden başarılı bir şekilde kâr elde etmek, düşük gecikmeli ve düşük titreşimli piyasa erişimi kullanarak piyasa olaylarına gerçek zamanlı tepki vermeyi gerektirir. Gecikme süresindeki milisaniyelik bir azalma bile arbitraj kârlılığını önemli ölçüde artırabilir. Bu nedenle, HFT stratejileri piyasa verilerini anında almak ve yeni emirleri verimli bir şekilde iletmek için düşük gecikmeli piyasa erişimi talep eder.

Bu nedenlerden ve FPGA'nın sağlayabileceği donanım hızlandırma olanaklarından dolayı FPGA, HFT sistemleri için birincil altyapı olarak seçilmektedir. FPGA çözümlerinin paralellliği ve determinizmi, matematiksel modellerin hesaplanma ve verilerin borsaların eşleştirme motorlarına iletilme hızını büyük ölçüde artırır.

## 1. INTRODUCTION

High-Frequency Trading (HFT) has emerged as a prominent trading strategy in financial markets, enabled by rapid advancements in technology and computational capabilities. HFT involves executing a large number of trades in fractions of a second, leveraging sophisticated algorithms to capitalize on market inefficiencies and exploit short-term price fluctuations. This approach requires ultra-low-latency and high-performance systems to process vast amounts of market data and execute trades swiftly.

Field-Programmable Gate Arrays (FPGAs) have gained significant attention in the domain of HFT due to their inherent parallel processing capabilities, low-latency characteristics, and ability to be reprogrammed for specific tasks. FPGA-based solutions offer the potential for enhanced performance, reduced latency, and improved flexibility compared to traditional software-based trading systems.

This thesis aims to explore the implementation of HFT on FPGA platform, focusing on the design, development, and evaluation of a high-performance trading system. The objective is to investigate how FPGA technology can be effectively utilized to enhance the speed and efficiency of HFT algorithms, thereby gaining a competitive edge in the financial markets.

The thesis will then focus on the development and implementation of HFT strategies on FPGA platforms, covering market data analysis and processing, order execution, and risk management. It will address the hardware requirements, design considerations, and challenges encountered during the FPGA implementation process.

Finally, the thesis will discuss the challenges faced during the research, identify limitations, and propose potential avenues for future work in this domain. It will conclude by summarizing the findings, contributions, and implications of the research conducted.

Through this investigation, the thesis aims to contribute to the understanding and advancement of FPGA-based HFT systems, offering insights into the potential benefits and challenges of employing FPGA technology in the dynamic and competitive landscape of high-frequency trading.

### **1.1. Purpose of The Project**

The purpose of using field-programmable gate arrays (FPGAs) in high-frequency trading (HFT) is to achieve faster trade execution speeds. FPGAs are special types of computer chips that can be customized to perform specific tasks, making them more efficient and faster than general-purpose processors like those found in a typical computer. In HFT, FPGAs are used to implement the algorithms that analyze market data and make trading decisions. Because FPGAs can be programmed to perform specific tasks very quickly, they can help HFT firms to execute trades faster than their competitors. By using FPGAs, HFT firms can reduce the latency (or delay) between receiving market data and executing a trade, which can give them an advantage in fast-moving markets.

### **1.2. Literature Review**

An strategy to investing that is still relatively new is high-frequency trading. As a result, it's frequently unclear and unclear how high-frequency trading relates to other, more traditional investment approaches.

Different sorts of HFT modules are used in the current trading market. We intend to apply the Borsa Istanbul (BIST) criteria in our project.

TCP/UDP, OUCH, and ITCH message formats are all used by HFT. BIST uses the 2022 OUCH and ITCH message formats, as well as the SoupBin TCP/Mold UDP standards. But because they run on non-FPGA systems, these protocols have a lot of latency.

HFT is used on FPGA boards in Turkey by the business known as "Matrix Investment." On such project, there are essentially two main tiers. The first layer is the host, which accepts trade orders from clients and notifies them of the process. The second layer is the FPGA board, which communicates with BIST servers and encodes and decodes OUCH and ITCH messages. [1]

Researchers from Brno University in the Czech Republic successfully develop HFT architecture with a 150 MHz clock speed and an average latency of 253 ns on a Xilinx Combo-LXT board. [2] In order to find speedier solutions in 2014, they apply several hashing algorithms on transferred messages. By then, hashing and other algorithms had undergone significant alteration. In contrast to 2014's algorithms, hypothetically newer algorithms offer little latency.

Actually, the project initiative and the Matrix Investment model are extremely similar. To create a client side and an FPGA side. On the client side, it has a user interface to communicate with clients, such as the ability to view current stock market values, place orders, cancel orders, edit orders, and so forth. On the FPGA side, it maintains communication with BIST servers to obtain market data, converts order actions to the desired format OUCH/ITH before transmitting via TCP/UDP to clients between servers, and provides feedback from clients to servers. The ITCH

protocol gathers information from other clients and the market to help clients make decisions about whether to purchase, sell, or cancel orders.

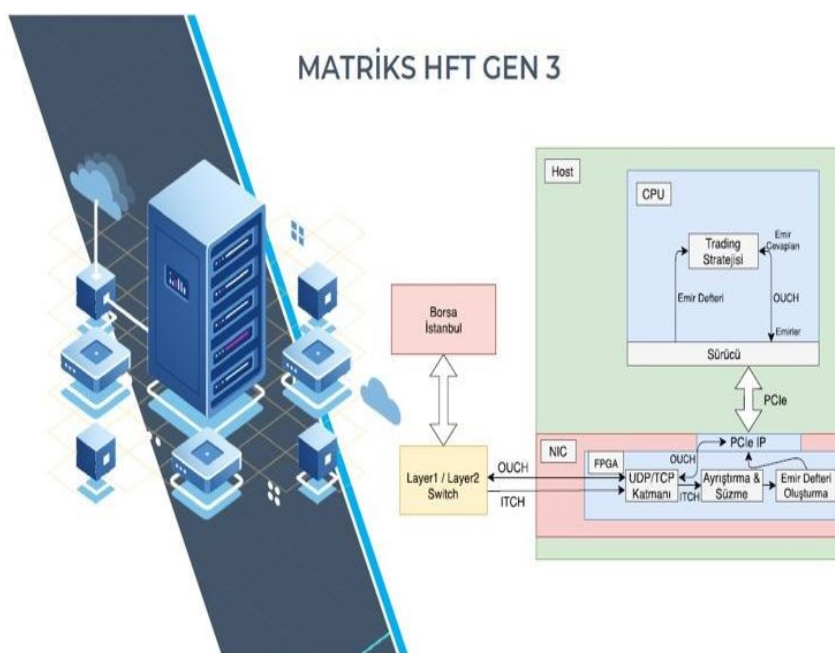
The communication format known as the OUCH protocol consists of requests from clients for services and responses from servers to clients. Due to the undesirable latencies of typical encode/decode operations, the FPGA portion is what delivers low-latency. FPGA processing operations are faster than those of a typical computer because of their nature. Our work is based on hand-written code on FGPA, and this project will be offer a faster method by avoiding the use of FPGA components like DSP, which introduces irregularity into signals, and by streamlining encode/decode functions and fundamental mathematical operations like multiplication.

Alongside with that main goal is the project implement and run all the processes on the FPGA board. All the steps that previously done by now trying to convert to FPGA's HDL language, we prefer to use verilog. Instead of rewriting the modules to be used in the project from scratch, some of the Intellectual Properties (IP) in the Xilinx Vivado program were used. For instance for ethernet module "AXI 1G/2.5G Ethernet Subsystem" IP were used.



## 2. FPGA Implementation for HFT

By their very nature, high-frequency trading systems demand quick, hesitating-free decision-making and execution. In these "mission-critical" trading jobs, properly coded computer systems usually outperform human traders, especially in risky market conditions. As a result, traditional human traders are quickly being replaced by computer trading systems on trading desks all over the world. HFT firms use complex algorithms and high-speed data feeds to identify and act on opportunities to buy or sell securities fractions of a second faster than their competitors. HFT technology consists of both software and hardware parts together, where software part runs on a dedicated CPU, and the hardware is implemented on Field Programmable Gate Array, known as FPGA.



**Figure 2.1** :HFT implementaion block diagram of real environment

Basically aimed HFT system has following functions:

- Receive and classify in-coming orders/quotes

- Perform run-time low latency buy and sell trading signals
- Listens stock market or server confirmation signal belongs to sended order signal.

With this functions aimed HFT system has some differs with traditional HFT systems. Traditional HFT systems has “decide mechanism”. Traditional ones calcute and create meaningful high profitable orders with indicators which belongs to stock market. Some of them has some features to calculate indicators by own.

## **2.1. HFT Protocols**

HFT system has some unique demands to implementation. Generally uses TCP uplink and UDP downlink communication structure. Also uses specific encodind and decoding schemes, “Order, Update, Cancel and Hit – OUCH” and “Intellitest Categorized Historical – ITCH” protocols.

### **2.1.1. OUCH Protocol**

OUCH is a protocol used by financial firms to transmit orders between themselves and other market participants. It stands for Order, Update, Cancel and Hit (match) and is used primarily in the options market. It is a text-based protocol that allows firms to communicate order and trade information in a standardized format. The OUCH protocol is maintained by the Options Clearing Corporation (OCC) and is designed to facilitate the automation of options trading. The abbreviation "OUCH" still refers to "NASDAQ OUCH," the order entry protocol that NASDAQ employs for the submission, modification, and cancellation of orders. This project used OUCH protocol defined at 29 September 2021 via BISTNET.[3]

```

test_server
C:\Users\ASUS\PycharmProjects(SocketPractice\venv\Scripts\python.exe C:\Users\ASUS\Py
Server is starting...
Server socket created.
Server socket is binding...
Socket is bound to server address and is listening on [('172.20.80.1', 5050)].
Waiting for connection request...
New connection from ('172.20.80.1', 56096).
Received username and password from client:
test12 10101010
authorizing username and password
authorized
session created with session id: vlgdat3194
session is active
OUCH message from client [('172.20.80.1', 56096)]:
New Order:
Order Token: 1234567890
Order Book ID: 345
Side: B
Price: 200
Quantity: 50
OUCH message from client [('172.20.80.1', 56096)]:
Replace Order:
Original Order Token: 1234
New Order Token: 5678
Quantity: 480
Price: 23

test_client
C:\Users\ASUS\PycharmProjects(SocketPractice\venv\Scripts\python.exe C:\Users\ASUS\Py
Client is starting...
Client socket created.
Client is connecting to the server...
Successfully connected to [('172.20.80.1', 5050)]
Session is active with session ID: vlgdat3194
Expected Sequence Number: 1
OUCH feedback from server [('172.20.80.1', 5050)]:
Order Accepted; Order Token: 1234567890
OUCH feedback from server [('172.20.80.1', 5050)]:
Order Replaced; Order Token: 1234

```

**Figure 2.2:** Example python OUCH Simulation

### 2.1.1.1. OUCH Protocol Message Types

The OUCH protocol consists of a series of messages that are used to transmit information between firms and other market participants. Some of the types of messages that are defined in the OUCH protocol includes; order messages, these messages are used to transmit orders to the market. They include information such as the symbol, side (buy or sell), quantity, and price; cancel messages, these messages are used to cancel orders that have been previously submitted to the market; replace messages, these messages are used to modify the terms of a previously submitted order; trade messages ,these messages are used to confirm the execution of an order. They include information such as the trade price and the quantity of the trade. Indication of Interest (IOI) messages, these messages are used to indicate a firm's interest in potentially trading a particular security. They do not commit the firm to making a trade, but rather are used to gauge interest in the market. Also OUCH protocol messages consists feedback that consist the information about sended message whether accepted, cancelled, executed.

The OUCH protocol is designed to facilitate the automation of options trading by providing a standardized way for firms to communicate order and trade information. It is used by firms to transmit orders to exchanges, as well as to communicate with other firms and market participants.

OUCH New Order Message				
Length (byte)	Name	Type	Value	Binary
1	Message Type	byte	D	01000100
4	Time Stamp	integer	17.31:45'151	00000000 00011101 00010001 11111111
1	Message ID	byte	32	00100000
4	Client Order ID	integer	unique id : 1542454	00000000 00010111 10001001 00110110
8	CCYPair	alpha	EUR/USD	01000101 01010101 01010010 00101111 01010101 01010011 01000100 00000000
1	Order Type	byte	limit order	00000010
1	Side	byte	sell	00000010
8	Quantity	long	150000.54==15000054	00000000 11100100 11100001 11110110
8	Minimum Quantity	long	150000.54==15000054	00000000 11100100 11100001 11110110
4	Rate	integer	1.24==124000	00000000 00000001 11100100 01100000
1	Time In Force	byte	2 = immediate	00000010
Total Message in binary = 01000100 00000000 00011101 00010001 11111111 00100000 00000000 00010111 10001001 00110110 01000101 01010101 01010010 00101111 01010101 01010011 01000100 00000000 00000010 00000010 00000000 11100100 11100001 11110110 00000000 11100100 11100001 11110110 00000000 00000001 11100100 01100000 00000010				

**Table 2.1** : OUCH decoding for new sell order message type

First 3 sections are , highlighted with blue, called message header. Every OUCH message has the same data block even though message type or values are different. Rest of the blocks are changeable due to message or order type.

Client Order Id is unique for every client hence in stock market server many different clients has connected. Their ID determined by the stock markets regulations. It can be based on region, client priority or something else.

Message ID is the value that message number comes from that client since beginning of the trade on current day.

Time Stamp calculated by milliseconds from 17:00.

CCY Pair is exchange pairs. It has some abbreviations, for instance United States Dollar for USD, Turkish Lira for TRY. CCY pair's abbreviations for one pair can only be 4 letters or 4 characters long. If it is shorter than 4 characters, character "slash "/" should be coded based on ASCII. For example for EUR/USD:

First 3 bytes in order E – U – R , 4<sup>th</sup> byte is "/", 5 – 7<sup>th</sup> bytes in order U – S – D and last byte is zero.

Side is order verb it can be buy – 1 or sell – 2 for "new order" message. In this example it is "sell" coded in binary 10.

Quantity is the amount of EUR that sold in exchange of USD.

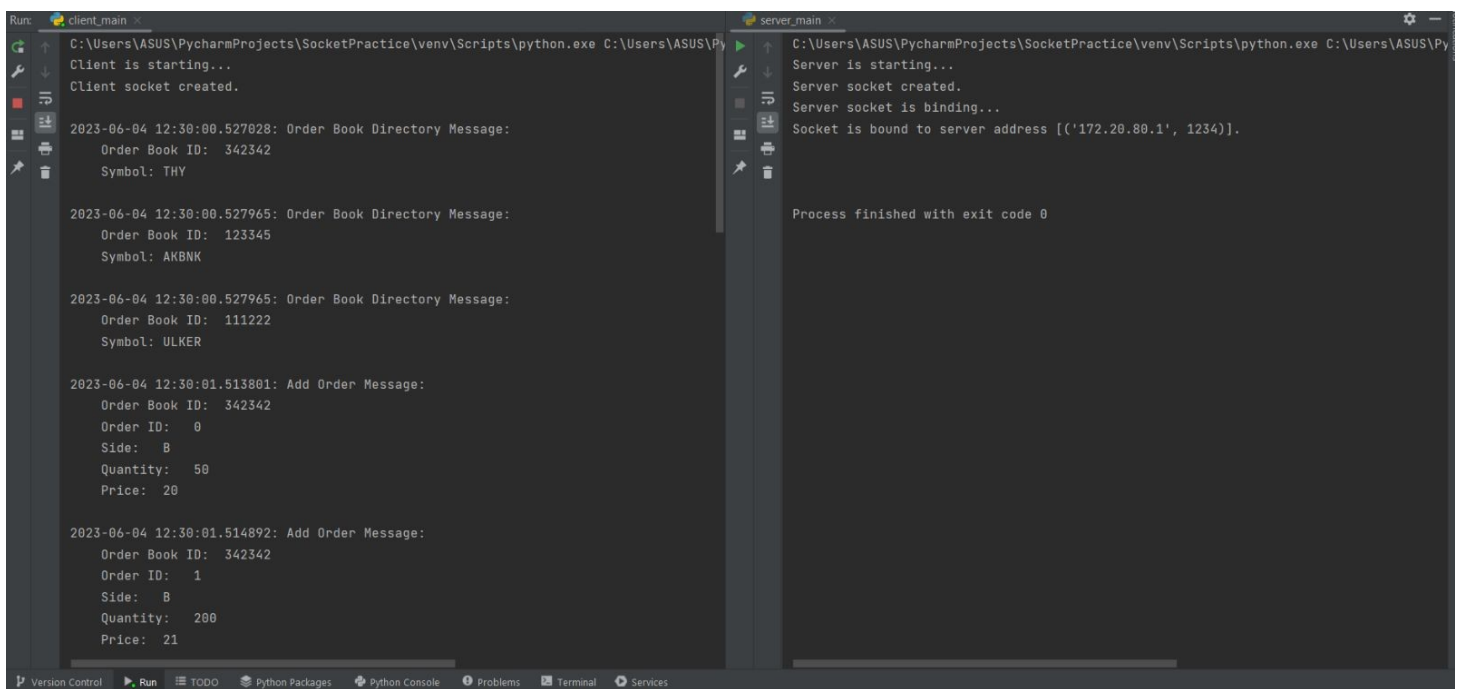
Minimum Quantity is the minimum amount of EUR that sold in exchange of USD. The reason that minimum quantity existence is some days stock markets has vast fluctuations on prices when order received it monitors the price to sell or buy in right amount of pair and price. For instance if current price is 1.24 and new buying order sent with price 1.23 market server waits to price falls to 1.23. At the same time maybe thousands of orders completed and price rise up again 1.24 like this rapid changes stock market look for minimum quantity. If it has enough time to complete order with minimum quantity it executes the order, if it has not enough time to complete order with minimum quantity buy/sell order waits in the order book.

Rate is the value of the CCY pair that client wants to buy or sell.

Time in force is the place act for stock market for placing order to order book.

### 2.1.2. ITCH Protocol

ITCH stands for Information Technology Channel. It is a text-based protocol used by financial firms to communicate information about orders and trades in the market. It is primarily used in the options market and is maintained by the Options Clearing Corporation (OCC). This project used ITCH protocol defined at 20 December 2022 via BISTNET.[4]



```
client_main
C:\Users\ASUS\PycharmProjects\SocketPractice\venv\Scripts\python.exe C:\Users\ASUS\Py
Client is starting...
Client socket created.

2023-06-04 12:30:00.527028: Order Book Directory Message:
Order Book ID: 342342
Symbol: THY

2023-06-04 12:30:00.527965: Order Book Directory Message:
Order Book ID: 123345
Symbol: AKBNK

2023-06-04 12:30:00.527965: Order Book Directory Message:
Order Book ID: 111222
Symbol: ULKER

2023-06-04 12:30:01.513801: Add Order Message:
Order Book ID: 342342
Order ID: 0
Side: B
Quantity: 50
Price: 20

2023-06-04 12:30:01.514892: Add Order Message:
Order Book ID: 342342
Order ID: 1
Side: B
Quantity: 200
Price: 21

server_main
C:\Users\ASUS\PycharmProjects\SocketPractice\venv\Scripts\python.exe C:\Users\ASUS\Py
Server is starting...
Server socket created.
Server socket is binding...
Socket is bound to server address [('172.20.80.1', 1234)].

Process finished with exit code 0
```

**Figure 2.3:** Example python ITCH Simulation

#### 2.1.2.1. ITCH Protocol Message Types

The ITCH protocol consists of a series of messages that are used to transmit information about orders and trades between firms and other market participants. The types of messages that are defined in the ITCH protocol include; time messages,

reference data messages, event and state messages, market by order messages and trade messages.

Timestamps are split into two parts for bandwidth saving reasons: Seconds and nanoseconds. Every second that at least one ITCH message is being created, the message is sent. The message includes the duration of 1970-01-01 00:00:00 UTC, generally known as Unix Time, in seconds.

First reference data message is order book directory. Order book directory messages are sent out for all active securities in the Genium INET Trading system at the beginning of each trading day. Second is Combination Order Book Leg. A mapping between a combination order book and one of the combination leg order books is provided by this message and it may be transmitted throughout the day if new combination order books are integrated into the network. Final order book leg message is Tick Size Table Entry. This message provides details about a tick size for a pricing range. A whole Tick Size Table price range is comprised of all Tick Size messages that have the same order book ID.

Event and State Change Messages includes System Event Message and Order Book State Message. A data or market feed handler event is signaled by the System Event Message type. Information about state changes is communicated through the Order Book State Message.

Market by Order Messages are Add Order Messages and Modify Order Messages. An Add Order Message signifies that the Genium INET Trading system has accepted a new order and added it to the displayable book. An Order ID, which is exclusive to each order book and is used by Genium INET Trading to track the order, is included in the message. The Order ID, Order book ID and Side of the Add Order that the update applies are always included in Modify Order notifications. First Modify Order Messages is Order Executed Message. Every time a book order is executed in

full or in part, this message is generated. Second is Order Executed with Price Message. When an order on the book is executed in full or in part at a price other than the initial display price, a rather uncommon occurrence, this message is received. An other message is Order Replace Message. When a book order is canceled or replaced, this message is sent. The remainder of the original orders amount is no longer usable and needs to be taken out. Order Delete Message which is the final message of Modify Order Message, is sent when a book order is deleted.

New Order Message				
Length (byte)	Name	Type	Value	Binary
1	Message Type	integer	R	1010010
4	Timestamp	integer	2018-10-07T21:37:32.102462105(1538948252102462105)	01011011 10111010 01010010 01101100
4	Order Book ID	integer	161098	00000000 00000010 01110101 01001010
32	Symbol	Alpha	O_USDTRYKE1018P6575	01001111 01011111 01010101 01010011 01000100 01010100 01010010 01011001 01001011 01000101 00110001 00110000 00110001 00111000 01010000 00110110 00110101 00110111 00110101
32	Long Name	Alpha	USDTRY_10/2018_AVR UPA_OPSIYON	01010101 01010011 01000100 01010100 01010010 01011001 01011111 00110001 00110000 00101111 00110010 00110000 00110001 00111000 01011111 01000001 01010110 01010010 01010101 01010000 01000001 01011111 01001111 01010000 01010011 01001001 01011001 01001111 01001110
12	ISIN	Alpha	TROXIST0YMR0	01010100 01010010 01001111 01011000 01001001 01010011 01010100 00110000 01011001 01001101 01010010 00110000



1	Financial Product	integer	1	1
3	Trading Currency	Alpha	TRY	01010100 01010010 01011001
2	Number of decimals in Price	integer	1	00000000 00000001
2	Number of decimals in Nominal Value	integer	0	00000000 00000000
4	Odd lot size	integer	0	00000000 00000000 00000000 00000000
4	Round lot size	integer	1	00000000 00000000 00000000 00000001
4	Block lot size	integer	0	00000000 00000000 00000000 00000000
8	Nominal value	integer	0	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1	Number of legs	integer	0	0
4	Underlying Order Book ID	integer	30026	00000000 00000000 01110101 01001010
4	Strike Price	Price	6575	00000000 00000000 0001100110101111
4	Expiration Date	Date	20181031	00000001 00110011 11110000 00100111
2	Number of decimals in Strike Price	integer	1	00000000 00000001
1	Put or Call	integer	2	00000000 00000010

**Table 2.2 : ITCH New Order Order Book Directory Message**

Finally, Trade Messages are split into Trade Message and Auction Messages. For typical match occurrences involving order types that are not displayable, execution information are provided in Trade Message. Additionally, specific cross trades are published using this message. In auction messages, markets for order dissemination can be configured to be deactivated during auctions. In such instances, each active order will be cancelled by an Order Delete message just before the auction.

## **2.2. TCP/IP and UDP**

TCP/IP is a set of networking protocols that allows devices on a network to communicate with each other. It is the foundation of the internet and most local area networks (LANs). [31,34]

TCP (Transmission Control Protocol) is a connection-oriented protocol that ensures that data is delivered reliably from one device to another. It does this by breaking the data into small packets and sending them to the destination device, where they are reassembled into their original form. If any packets are lost along the way, TCP will retransmit them until they are received correctly.

IP (Internet Protocol) is a connectionless protocol that routes packets of data from one device to another based on the destination device's IP address. It does not guarantee that the packets will be delivered successfully, but it does its best to get them to the destination as efficiently as possible. Together, TCP and IP form the foundation of the internet and are used by almost all devices that are connected to it.

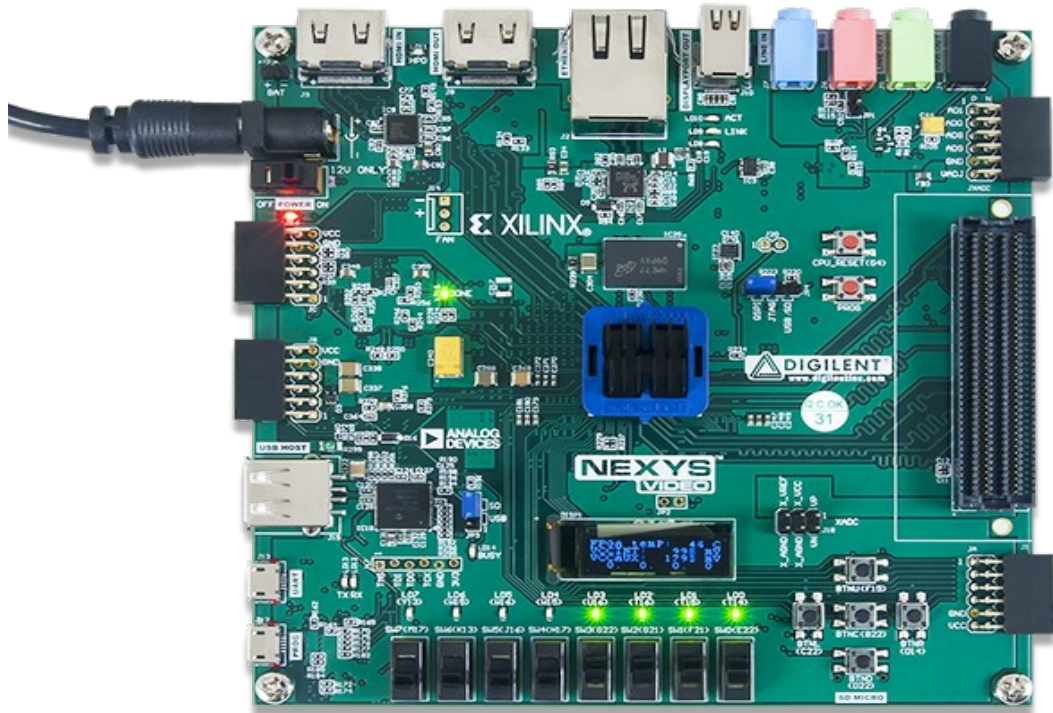
UDP (User Datagram Protocol) is a connectionless protocol that is used to send data between devices on a network. It is an alternative to the more common connection-oriented protocol, TCP (Transmission Control Protocol).

Like TCP, UDP breaks data into small packets and sends them to their destination. However, unlike TCP, UDP does not establish a connection between the devices before sending the packets, and it does not guarantee that the packets will be delivered successfully. This makes UDP faster and more efficient than TCP, but it also means that it is less reliable.

UDP is often used for real-time applications that require low latency, such as online gaming and voice over IP (VoIP). It is also used for tasks that do not require a reliable connection, such as broadcasting data to multiple devices.

### **2.3. FPGA Board and Tools Specifications**

In the project Nexys Video Artix-7 FPGA Trainer board will be used. It is optimized for high data transfers and it has its own ethernet hardware. Also it is supported by Xilinx's ISE® toolset, Vivado ® Design Suite and its licence allows us to use Microblaze units on it.[8]



**Figure 2.4:** Nexys Video Trainer Board

To facilitate communication between an FPGA and an Ethernet network, hardware and software components are referred to as Ethernet subsystems for FPGA boards. The MAC (Media Access Control) layer, TCP/IP (Transmission Control Protocol/Internet Protocol) stack, and physical Ethernet interface are the three components that make up these subsystems most frequently.

The electrical signals produced by the FPGA are transformed via the physical Ethernet interface into signals that may be transferred over an Ethernet network, and the opposite is also true. The TCP/IP stack manages higher-level networking functions including routing, error correction, and the fragmentation and reassembly of data packets, whereas the MAC layer is in charge of controlling the transmission and reception of Ethernet packets.

As understandable designing ethernet subsystem is quite vast task to do therefore AXI ethernet subsystem IP used in the project. Yet standalone AXI Ethernet Subsystem is no sufficient for transferring data and receiving data it is lack of Central Processing Unit (CPU). Microblaze CPU IPs used as well.

We can explain the reason for using a processor as follows; due to its capacity for high-speed data processing and customizability, FPGAs are frequently employed in Ethernet applications. To manage network protocols like TCP/IP and to enable data transfer between the Ethernet controller and other peripherals, a processor is frequently included in an FPGA for Ethernet applications. In addition, hardware accelerators for Ethernet protocols, such as checksum computations and packet filtering, can be implemented using FPGAs, offloading operations from the CPU and enhancing system performance. An FPGA-based Ethernet system can achieve a high level of integration and minimize system complexity while maintaining great performance and flexibility by adding a processor.

A soft processor core called MicroBlaze was created by Xilinx and may be used with their FPGAs. It is a highly configurable, high-performance, low-power CPU that may be altered to satisfy particular design specifications. MicroBlaze can be used to implement a variety of embedded applications, including as networking, motor control, and digital signal processing. It has a tiny footprint, is highly expandable, and can be configured to only have the capabilities and peripherals needed for a particular application.

A tri-mode (10/100/1000 Mb/s) Ethernet MAC or a 10/100 Mb/s Ethernet MAC can be implemented via the Xilinx 1G/2.5G AXI Ethernet Subsystem. This core allows for the connection of a media access control (MAC) chip to a physical-side interface (PHY) chip using the MII, GMII, SGMII, RGMII, and 1000BASE-X interfaces. Additionally, it offers an on-chip PHY for the 1000/2500 BASE-X and 1G/2.5G SGMII modes. PHY Management registers can be accessed using the MDIO interface. The TCP/UDP complete checksum offload, VLAN stripping, tagging,

translation, and expanded filtering for multicast frames features are all potentially enabled via this subsystem.

The Vivado IP integrator block design allows for the addition of the AXI Ethernet Subsystem to the design's canvas. If the subsystem is chosen from the IP catalog in the Vivado Integrated Design Environment (IDE), it can also be used in a register transfer level (RTL) flow. Customization, instantiation inside of a design, output product generation, behavioral simulation, design elaboration, synthesis and implementation, and bitstream generation for the target device are all possible with the catalog. During the system design session, various infrastructure cores are configured and joined to form the hierarchical design block known as the AXI Ethernet Subsystem.

With regard to Ethernet, this subsystem offers more capability and simplicity of use. This subsystem instantiates the necessary infrastructure cores, establishes interface ports based on the configuration, and connects these cores. The Xilinx Tri-Mode Ethernet MAC (TEMAC) and 1G/2.5G Ethernet PCS/PMA or Serial Gigabit Media Independent Interface (SGMII) cores are the infrastructure cores for this subsystem. Also, the AXI Ethernet Buffer core offers additional capabilities such as TX and RX TCP/UDP Partial Checksum offload, IPv4 TX and RX TCP/UDP full checksum offload.

A commercial Ethernet PHY device that supports the BASE-T standard at rates of 1 Gb/s, 100 Mb/s, and 10 Mb/s is attached to the PHY side of the subsystem. Any of the following supporting interfaces, including GMII/MII, RGMII, 1G/2.5G Ethernet PCS/PMA, or SGMII, can be used to link the PHY device.

In this design we use RGMII (Reduced Gigabit Media Independent Interface) as physical interface type. It is essentially a Double Data Rate type of GMII and it allows Ethernet operation at speeds of 10 Mb/s, 100 Mb/s, and 1 Gb/s.

## **2.4. Intellectual Property Cores**

FPGA board has its own physical layer of ethernet system. Able to use it additional tools needed. Most of them are available on Vivado with TEMAC license.

### **2.4.1. AXI Ips**

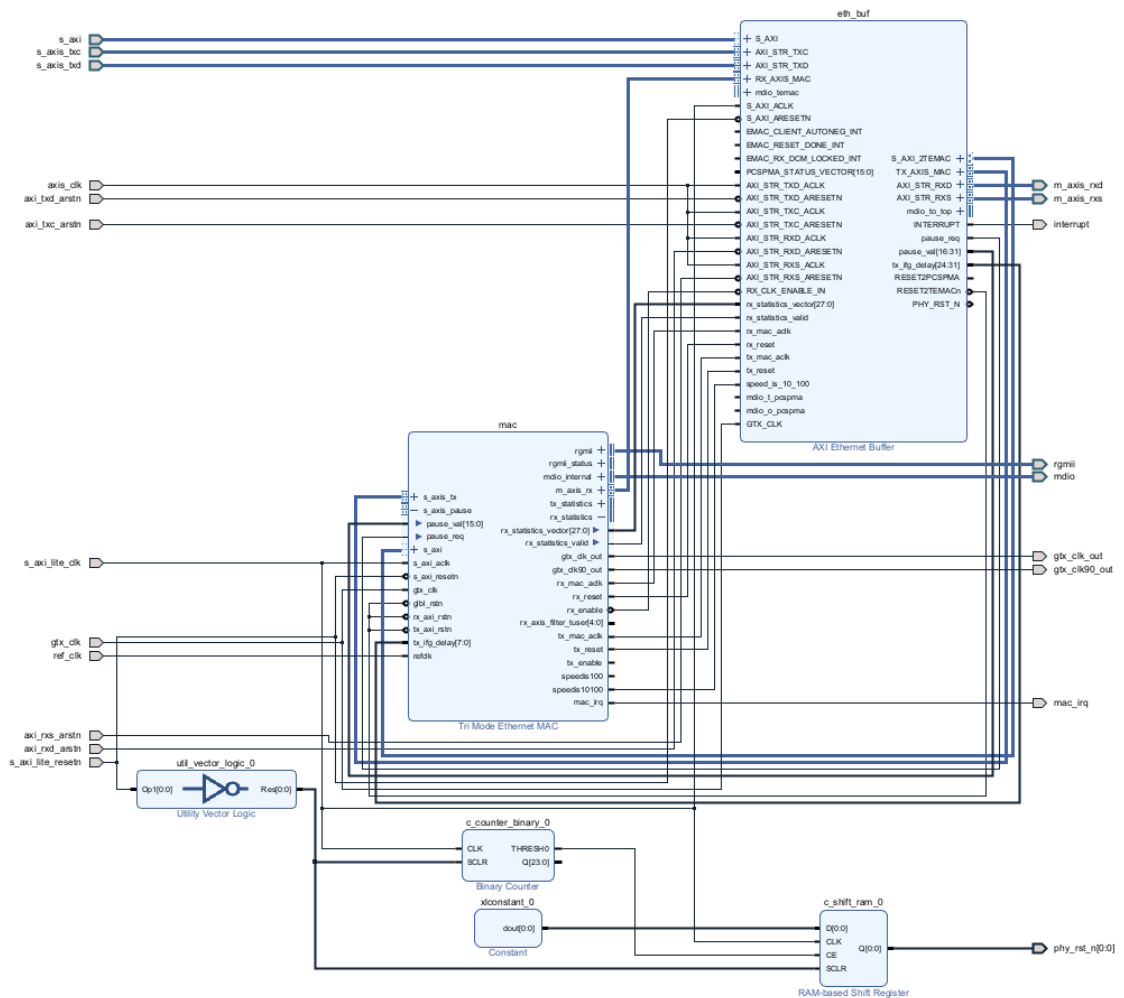
A prominent interface used in Xilinx Vivado, a design suite for creating FPGA (Field-Programmable Gate Array) designs, is AXI (Advanced eXtensible Interface). AXI is a widely used interconnect standard that makes it easier for different IP (Intellectual Property) blocks in an FPGA design to communicate with one another.

For effective data transfer between IP cores, memory controllers, and other parts of an FPGA design, AXI is created to offer a high-performance, low-latency, and scalable interface. It uses a master-slave architecture, in which a master demands data transfers and a slave complies.

### **2.4.2. AXI 1G/2.5G ethernet subsystem**

The AXI 1G/2.5G Ethernet Subsystem IP is a pre-built intellectual property (IP) block provided by Xilinx for implementing Ethernet connectivity in FPGA designs using the AXI (Advanced eXtensible Interface) protocol. It is designed to support high-speed Ethernet communication at 1 Gigabit per second (Gbps) and 2.5 Gbps data rates.

The AXI 1G/2.5G Ethernet Subsystem IP offers a complete solution for integrating Ethernet connectivity into FPGA designs, providing a range of features and capabilities including; Direct memory access, Ethernet Media Access Control, FIFOs and buffers, clock and reset interfaces.[9]



**Figure 2.5: AXI Ethernet Subsystem Block Design**

### 2.4.3. AXI interconnect

The AXI Interconnect is a key component in the ARM AMBA (Advanced Microcontroller Bus Architecture) specification. It functions as a System-on-Chip (SoC) design's system-level interconnect for tying together numerous masters and slaves. The AXI Interconnect makes it possible for the system's IP (intellectual property) cores, memory controllers, and other peripherals to communicate effectively and at high speeds.[10]



The AXI Interconnect regulates the routing of data and control signals between many channels, commonly referred to as "masters" and "slaves," to enable communication.

#### **2.4.4. AXI direct memory access – DMA**

AXI DMA designed to facilitate efficient data transfer between external devices and the FPGA's memory. It employs the AXI protocol for communication, providing high-performance, low-latency, and configurable DMA capabilities.[11]

The AXI DMA IP block acts as a data mover, offloading data transfer tasks from the processor or software, resulting in improved system performance and reduced CPU utilization. It is particularly useful when large volumes of data need to be moved between different memory regions or exchanged with external devices, such as high-speed interfaces, storage devices, or peripherals.

#### **2.4.5. AXI UARTLite**

AXI UARTLite core provides a standardized interface for integrating UART functionality into FPGA designs, allowing for asynchronous serial communication with external devices.

Typically, the AXI UARTLite IP core is used to facilitate serial interface communication between the FPGA and external devices, such as PCs, microcontrollers, or other peripherals. It adheres to the AXI protocol, which offers an interconnect standard that is high-performance, low-latency, and scalable.[12]

#### **2.4.6. Microblaze**

MicroBlaze is a soft processor IP core. It is a configurable, 32-bit RISC (Reduced Instruction Set Computer) processor designed to be implemented on FPGA. [13]

The MicroBlaze IP core offers a flexible and customizable embedded processing solution within an FPGA design. It is highly configurable, allowing designers to tailor the processor's features and resources to meet the specific requirements of their application.

Communication, OUCH and ITCH protocols runs on the microblaze IP. It runs C or C++ code on FPGA.

In addition, the required clock, reset and interrupt handler IPs have also been appropriately added.

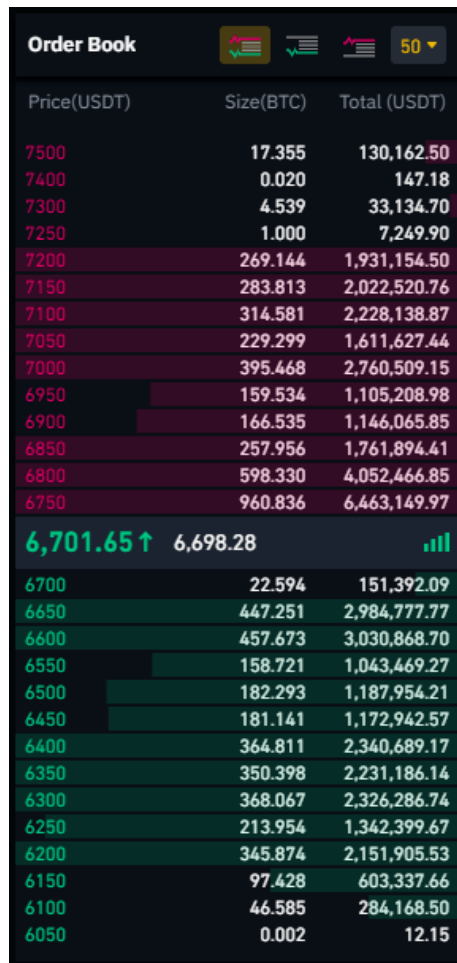
### **3. RESULT AND ANALYSIS**

First part of the project was implementing HFT communication system on local computers with real data using python language coding.

#### **3.1. Non-FPGA Environment Experiment**

As stated experiment made by using Python language and local computers. Real data was obtained from “BIST Educational Support” section. Real data has one day of every information about stock market including; buy orders, sell orders, cancel orders, limit-stop orders, volalities etc. Buy and sell orders quantities and paper names extracted from example data and converted to OUCH or ITCH protocol as demand. OUCH and ITCH protocol specifications, encoding and decoding schemes written by BIST recommendations. [3-4]

Also the order book that a list of buy and sell orders for a particular security or financial instrument that is sorted by price level created to read and observe the coming orders.



**Figure 3.1:** Example order book

In other words, the order book shows the prices at which buyers are willing to purchase the security, as well as the quantities of the security that they are willing to buy at each price level. It also shows the prices at which sellers are willing to sell the security, and the quantities that they are willing to sell at each price.

The order book can be thought of as a snapshot of the current supply and demand for a particular security. It is constantly changing as new orders are placed and existing orders are filled.

```
100 20.75
270 20.6
145 20.5
10 20.4
20 20.0

100 20.75
270 20.6
100 20.5
10 20.4

100 20.75
150 20.6
100 20.5
10 20.4

80 20.75
150 20.6
100 20.5
10 20.4

150 20.6
100 20.5
10 20.4
```

**Figure 3.2:** Example python order cook. Left column is the quantities, right column is the price. Every new block is represent different timestamp.

### 3.2. Echo Server on FPGA

By the name echo servers is type of the network term that system sends back whatever the data sent to it basically echoing the received data.

Example design implemented on Nexys Video A7 FPGA board. Design taken as an example from Digilent's sources "Nexys Video - Getting Started with Microblaze Servers" to create base of the ethernet subsystem. Further developments will be unique for the OUCH, ITCH protocols.[6]

To use AXI Subsystem Ethernet subsystem; we need to implement first Microblaze and run block automation. Then clock and concat blocks will added to microblaze IP. After Microblaze implemented 4 cores will added for AXI, Memory Interface Generator, AXI Uartlite, AXI Ethernet Subsystem, AXI Timer. Then running block automation for them. After automation done ethernet physical interface should be selected in this case it is RGMII. Running block automation again. Concat block takes interrupt inputs and sends them to microblaze controller. After all block design completed HDL wrapper should be obtain and then bitstream should generated.

Using that block design on Figure 3.3 firstly on Vivado bitstream file has been generated. Empty ports and connections setted on constraint file to ignore state due to prevent bitstream generation failure. Using bitstream file and project itself it exported to Vitis on Vivado.

On Vitis there are some ready-to-use application projects like echo server, TCP Client, TCP server etc. using "Lite Weight Internet Protocol" to realize modules. LWIP is an open-source TCP/IP networking stack made primarily for embedded systems with minimal resources. TCP/IP stands for Transmission Control Protocol/Internet Protocol. It seeks to offer a compact and effective TCP/IP implementation, enabling network access in hardware with limited resources, such as embedded systems, microcontrollers, and microprocessors.

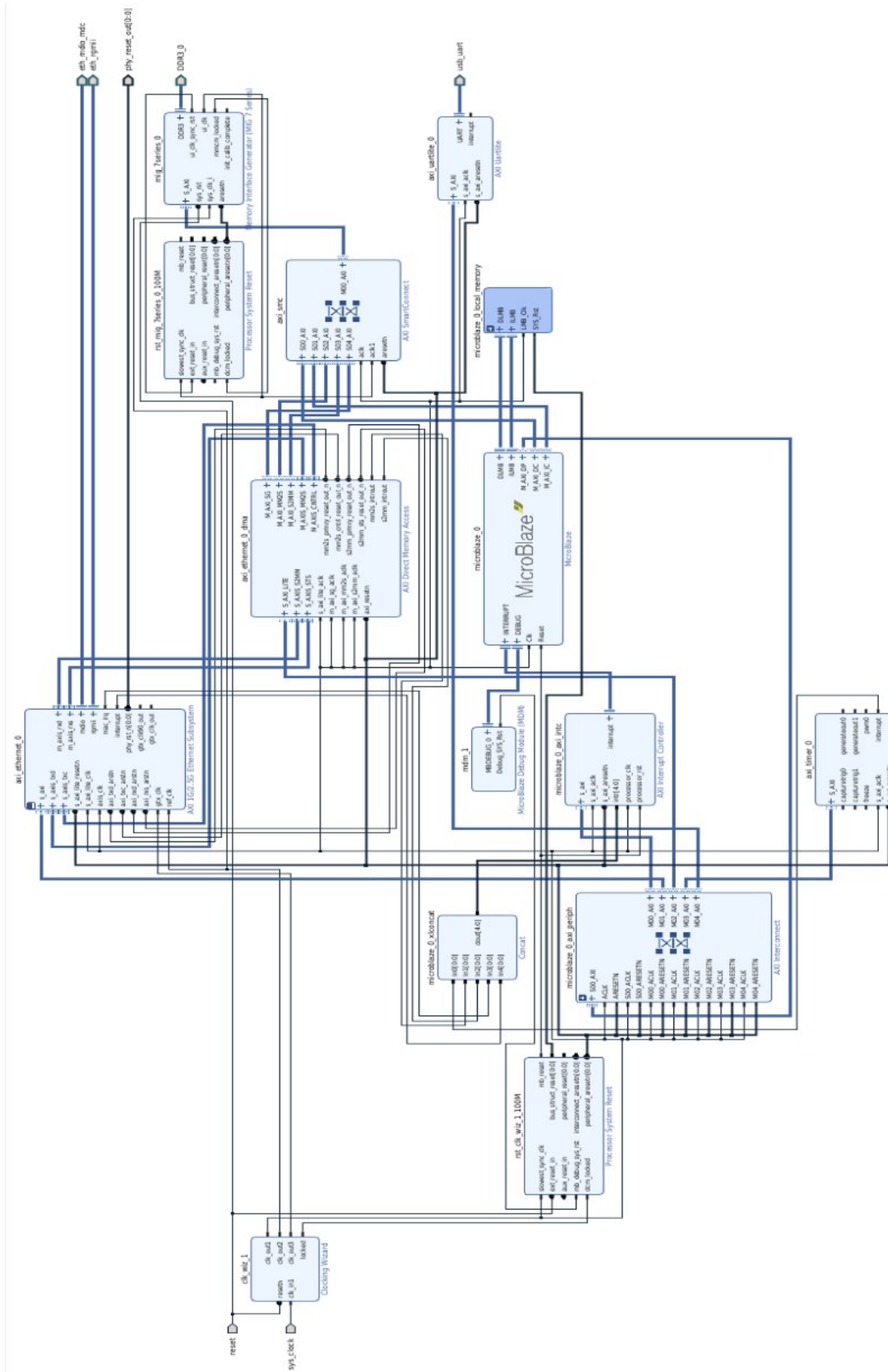


Figure 3.3: Complete Block design of the ethernet subsystem

On the Figure 3.3 complete block design can be seen by all the connections.

LWIP Echo server listens on static IP by default 192.168.1.10 and port 7 it can be changed by demands. In first experiments there are some IP address overlapping because of the given IP address is the already taken by another device on the network thats why at first tries IP address is setted to localhost – 127.0.0.1.

Wireshark is used for the demonstrate the data traffic on the network, capture and analyze whether it is useful or corrupted.[7] At the below figure on localhost (IP:127.0.0.1) send some info packages ,that contains only few bytes long ping requests, to itself (echoing) clearly captured and seen.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	45	58857 → 58858 [PSH, ACK] Seq=1 Ack=1 Win=10233 Len=1
2	0.000049	127.0.0.1	127.0.0.1	TCP	44	58858 → 58857 [ACK] Seq=1 Ack=2 Win=9991 Len=0
3	0.110440	127.0.0.1	127.0.0.1	TCP	45	58857 → 58858 [PSH, ACK] Seq=2 Ack=1 Win=10233 Len=1
4	0.110464	127.0.0.1	127.0.0.1	TCP	44	58858 → 58857 [ACK] Seq=1 Ack=3 Win=9991 Len=0
5	0.209733	127.0.0.1	127.0.0.1	TCP	45	58857 → 58858 [PSH, ACK] Seq=3 Ack=1 Win=10233 Len=1
6	0.209783	127.0.0.1	127.0.0.1	TCP	44	58858 → 58857 [ACK] Seq=1 Ack=4 Win=9991 Len=0
7	0.220727	127.0.0.1	127.0.0.1	TCP	45	58857 → 58858 [PSH, ACK] Seq=4 Ack=1 Win=10233 Len=1
8	0.220883	127.0.0.1	127.0.0.1	TCP	44	58858 → 58857 [ACK] Seq=1 Ack=5 Win=9991 Len=0
9	0.228425	127.0.0.1	127.0.0.1	TCP	45	58857 → 58858 [PSH, ACK] Seq=5 Ack=1 Win=10233 Len=1
10	0.228447	127.0.0.1	127.0.0.1	TCP	44	58858 → 58857 [ACK] Seq=1 Ack=6 Win=9991 Len=0

**Figure 3.4:** Localhost echoing with hardware

After succesfully seen echoes IP addresses on the local network scanned by some tools on Ubuntu like “nmap” and cleared default given address hence LwIP codes on Vitis designed to run server on 192.168.1.10. Clearing another device on local network is easier to changing all IP parameters on the code.



```

-----lwIP TCP echo server -----
TCP packets sent to port 6001 will be echoed back
WARNING: Not a Marvell or TI or Realtek Ethernet PHY. Please verify the initialization sequence
Start PHY autonegotiation
Waiting for PHY to complete autonegotiation.
autonegotiation complete
link speed for phy address 4: 100
DHCP Timeout
Configuring default IP of 192.168.1.10
Board IP: 192.168.1.10
Netmask : 255.255.255.0
Gateway : 192.168.1.1
TCP echo server started @ port 7

```

**Figure 3.5:** Serial output of Echo server

On Vitis LwIP Echo Server there is also DHCP method. But it is unavailable by default by the reason of DHCP protocol has unique hand shake methods to discover empty IP locations on local network. An example code assumes the while designed hardware necessary DHCP modules not added.

### 3.3. TCP Server on FPGA

The BSD license applies to the open source TCP/IP protocol suite known as lwIP. Although it can be used in conjunction with operating systems, the lwIP is a stand-alone stack without any operating system dependencies. The lwIP gives apps access to two APIs:

- RAW API: Offers access to the basic components of the lwIP stack.
- Socket API: Offers a stack interface in the BSD sockets manner.

To use lwip, the hardware system is set first.

The following are some of the hardware system's main parts:

- MAC: Axi\_ethernet, axi\_ethernetlite, and MAC (GigE) and Gigabit Ethernet controller cores are supported by LwIP.
- Processor: MicroBlaze is used.

- Timer: Applications using the lwIP raw API must call specific functions at periodic times in order to maintain TCP timers. This can be accomplished by an application that registers an interrupt handler with a timer.

DMA: In axi\_ethernet-based systems, the axi\_ethernet cores are set up with a FIFO interface or a soft DMA engine (AXI DMA).

After the hardware system is set, the software system is set.

The lwIP library must first be built into the software application in order to use it with lwIP. To do this, first, new Platform Project is initiated in the software application and a new Hardware Platform Specification is created. Then, selection of where the board support project files will be located is made. The suitable hardware platform is chosen for application and the target CPU is picked from the list provided. The type of BSP (board support package) to be created is selected. After that, the creation of the new software platform is finalized and it will be visible in the Vitis Navigator. Automatic building of the board support package is enabled and domain settings are adjusted as needed. The settings are confirmed, the platform is builded, and the settings dialog box is closed. The appropriate domain/BSP is selected from the platform.spr file. Finally, BSP settings are modified, LwIP library is selected from the list of available libraries and it is configured.

## **4. REALISTIC CONSTRAINTS AND CONCLUSIONS**

### **4.1. Application Area of Project**

Equity markets, HFT began in the equity markets and continues to have a strong presence here [14]. Algorithmic strategies rapidly execute trades, exploiting minute price discrepancies in milliseconds [15]. HFT firms can contribute significantly to the overall market liquidity, which can reduce trading costs for other market participants and improve overall market efficiency [16].

Futures and options market, The technological advancements that enabled HFT in the equity markets have also made it possible in futures and options markets. These markets have grown more electronic, enabling HFT firms to play a significant role. HFT strategies can improve price discovery, market depth, and tighten bid-ask spreads [17].

Forex markets, The foreign exchange market, or Forex, has also seen the introduction and growth of HFT. As the largest and most liquid financial market globally, Forex presents substantial opportunities for high-speed, high-volume trading. Market-making strategies have been widely adopted in this space [18].

Fixed income markets, While HFT is less common in fixed-income markets due to these markets' traditionally over-the-counter nature, the introduction of electronic trading platforms has led to increased HFT presence. This trend is particularly noticeable in the U.S. Treasury market [19].

Cryptocurrency markets, The advent and subsequent rise of cryptocurrencies have opened a new frontier for HFT. Their high volatility, along with the 24/7 operating hours of cryptocurrency markets, make them ideal for HFT strategies. HFT can contribute to the liquidity and price discovery process in these markets [20].

## 4.2. Realistic Constraints

Complexity in programming and design, programming an FPGA requires a level of expertise far beyond what is required for coding in high-level languages like Python or Java, commonly used in algorithmic trading. Designing circuits for FPGA often involves hardware description languages (HDL) like VHDL or Verilog, which are significantly more complex to use.

Moreover, while an FPGA offers parallelism to speed up computations, effectively utilizing this feature is a non-trivial task. Designing highly parallel algorithms that correctly and efficiently solve complex trading problems is an area of active research

Cost implications, FPGAs can be quite expensive, not only in terms of hardware cost but also in terms of the development time and resources required. It's necessary to have skilled engineers who understand both the hardware and the trading algorithms to get the most out of an FPGA implementation. Additionally, due to the rapid pace of technological evolution, FPGAs may need to be updated or replaced frequently, contributing to ongoing costs.

Scalability challenges, Scaling FPGA designs to handle more complex algorithms or larger data sets is not as straightforward as it is with CPU or GPU-based systems. This is due to the fundamental difference in architecture, where FPGA solutions are explicitly parallel and require manual management of resources [21].

Lack of flexibility, Once an FPGA is programmed for a particular task, making changes or updates can be more complicated compared to software solutions. The need to reconfigure hardware for every algorithm change can limit the flexibility and

adaptability of FPGA solutions in a market environment that often requires quick adaptation to changing conditions.

Software licenses of programs and tools, due to the this project is belongs to the university and students most of the tools and licenses are free. Therefore it did not create limit while working on the project however in normal circumcitences software and tools licences will create limits for project owners.

### **4.3. Social, environmental and economic impact**

Trading is now much faster and more easily accessible thanks to HFT. There is now a gap between those who can access HFT technology and those who cannot as a result of this. A multiple-tier market is created as a result of companies with access to more capital being able to afford to invest in the newest connectivity and technology.

With suggestions that HFT may contribute to an unfair market structure where HFT businesses can use their speed to the detriment of slower traders, concerns about market fairness have also surfaced. [22].

Due to the robust servers and cooling systems required to support HFT operations, HFT has been linked to rising energy consumption. Data centers' environmental impact increases along with their energy usage, which adds to carbon emissions and climate change. [23].

By supplying liquidity and lowering bid-ask spreads, HFT can improve market efficiency. It enables the quick integration of information into prices, improving price discovery.

HFT, on the other hand, can raise systemic risk in the financial markets. This was particularly evident during the "Flash Crash" of May 2010, when a sudden sell-off caused the Dow Jones Industrial Average to plummet significantly before quickly rising again [24].

It has been hypothesized that HFT will increase economic inequality. While it produces substantial profits for some businesses and people, it may also lead to wealth inequality and a widening income disparity.

#### **4.4. Health and Safety Concerns**

There are no relative work about HFT and its health and safety concerns directly however, the stress and mental health impact of the financial industry more broadly have been studied, as the high pressure, high stakes nature of the work can contribute stress and related issues.

One might also infer potential indirect health and safety considerations. For example, the immense energy consumption and subsequent environmental impact of large-scale computing operations, including HFT, might be considered a health and safety concern from an environmental perspective. Yet, this is not specific to HFT.

#### **4.5. Standards**

In the United States, the Securities and Exchange Commission (SEC) and the Commodity Futures Trading Commission (CFTC) are the primary regulatory bodies that oversee financial trading, including HFT. There are rules in place regarding fraud, market manipulation, and disruptive practices. For example, the SEC's Rule 15c3-5 (the Market Access Rule) requires brokers to have risk controls in place before providing customers with access to the market [25].

In the European Union, the Markets in Financial Instruments Directive II (MiFID II) has specific rules about HFT, such as requiring firms to store time-sequenced records of their algorithmic trading systems and strategies for at least five years [26].

There are also market-specific regulations that HFT firms must follow. For example, the New York Stock Exchange (NYSE) and NASDAQ have rules regarding manipulative trading practices [27].

However, there is ongoing debate about whether existing regulations are sufficient to handle the unique challenges posed by HFT, and new rules and standards may be introduced in the future.

The standard for Verilog is maintained by the IEEE (Institute of Electrical and Electronics Engineers) under the standard [28].

Ethernet is a network communication standard used to establish a local area network (LAN). The IEEE 802.3 standard defines the rules for configuring an Ethernet network [29].

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are core protocols of the Internet protocol suite. They are defined in the Internet Engineering Task Force's (IETF) RFC 793 and RFC 768, respectively[30-31].

Python is an interpreted, high-level, and general-purpose programming language. The Python Software Foundation is the organization behind Python and they provide and maintain the standard implementation of Python, known as CPython. While there isn't a formal "standard" like IEEE or IETF standards, the language's evolution follows a community-driven process where new features, improvements, or changes

are proposed, discussed, and approved via Python Enhancement Proposals (PEPs) [32].

#### **4.6. Cost Analysis**

Mainly project based on human work and open source implementation. However AXI Ethernet Core property requires TEMAC licence from Xilinx Vivado yet it can be purchased via student account for free for limited days. FPGA board is selected and given by advisor Prof. Dr. Berna Örs Yalçın normally it cost approximately 600 USD. Apart from these overall cost can be calculated via adding all parameters such as working times, simulation computer time etc.

#### **4.7. Conclusions**

In the previous sections implementing HFT on FPGA subject explained detailed by achieved works.

First of all gathered information about HFT and how it works. In the same way research has been done for technical aspects of HFT and requirements.

Secondly, a local dry-run HFT environment created with python code. All the necessary encoding-decoding schemes, communication protocols and command line interface established successfully. Then local environment run by real life data obtained from BIST.

Lastly, python work transformed step by step to hardware environment, FPGA. First simple echo server created on board. Then monitor its data flow, time responses and overall answers to the sent data. After adequate observation done simple TCP client



built on FPGA. However TCP Client is not working as expected. Due to the lack of time it can not be corrected.

#### **4.8. Future Work and Recommendations**

Fully integrated HFT module on hardware could be implemented. Furthermore with enough resource and suitable FPGA board buy and sell orders can be created with using artificial intelligence software. This would allow faster transactions. If AI trained well and error rates set properly orders and transactions would yield more profit to users. In the event of all these steps achieved successfully HFT hardware design can be run on directly on to stock market servers, it will reduce latency between stock market and HFT hardware to minimum and yields more profit to users.



## 5. REFERENCES

- [1] Matriks HFT. Matriks Bilgi Dağıtım Hizmetleri - Matriks Data. (n.d.). Retrieved November 27, 2022, from <https://www.matriksdata.com/website/kurumsal-urunler/matriks-hft>
- [2] Dvorak, M., & Korenek, J. (2014). Low latency book handling in FPGA for high frequency trading. *17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*.  
<https://doi.org/10.1109/ddecs.2014.6868785>
- [3] Ouch protocol specification - bursa istanbul. (n.d.). Retrieved November 27, 2022, from [https://www.borsaistanbul.com/files/OUCH\\_ProtSpec\\_BIST\\_va2414.pdf](https://www.borsaistanbul.com/files/OUCH_ProtSpec_BIST_va2414.pdf)
- [4] *Itch protocol specification - bursa istanbul*. (n.d.). Retrieved January 5, 2023, from <https://www.borsaistanbul.com/files/bistech-itch-protocol-specification.pdf>
- [5] AMD Adaptive Computing Documentation Portal. (n.d.). Retrieved April 27, 2023, from <https://docs.xilinx.com/r/en-US/pg138-axi-ethernet/AXI-1G/2.5G-Ethernet-Subsystem-v7.2-Product-Guide>
- [6] K, S. (n.d.). *NEXYS video - getting started with microblaze servers*. Nexys Video - Getting Started with Microblaze Servers - Digilent Reference. Retrieved April 27, 2023, from <https://digilent.com/reference/learn/programmable-logic/tutorials/nexys-video-getting-started-with-microblaze-servers/start>
- [7] *Wireshark*. Wireshark user's guide. (n.d.). Retrieved April 27, 2023, from [https://www.wireshark.org/docs/wsug\\_html\\_chunked/](https://www.wireshark.org/docs/wsug_html_chunked/)
- [8] Martha. (n.d.). *NEXYS Video Reference Manual*. Nexys Video Reference Manual - Digilent Reference. Retrieved April 27, 2023, from

[https://digilent.com/reference/programmable-logic/nexys-video/referen  
ce-manual?redirect=1](https://digilent.com/reference/programmable-logic/nexys-video/referen<br/>ce-manual?redirect=1)

- [9] AXI 1G/2.5G ethernet subsystem. (n.d.). Retrieved from  
<https://docs.xilinx.com/r/en-US/pg138-axi-ethernet/IP-Facts>
- [10] AXI interconnect. (n.d.-b). Retrieved from  
<https://docs.xilinx.com/r/en-US/pg059-axi-interconnect>
- [11] AXI Dİrect Memory Access. (n.d.-b). Retrieved from  
[https://docs.xilinx.com/r/en-US/pg021\\_axi\\_dma](https://docs.xilinx.com/r/en-US/pg021_axi_dma)
- [12] AXI UARTLite. (n.d.-d). Retrieved from [https://docs.xilinx.com/v/u/2.0-English/axi\\_uartlite\\_ds741](https://docs.xilinx.com/v/u/2.0-English/axi_uartlite_ds741)
- [13] Microblaze IP Core. (n.d.-e). Retrieved from  
<https://docs.xilinx.com/r/en-US/ug1579-microblaze-design/Instantiate-MicroBlaze-IP-Cores>
- [14] Chlistalla M., 2011, High Frequency Trading. Better than its Reputation?,  
Deutsch Bank Research, Research Briefing, February
- [15] Aldridge, I. (2013). High-frequency trading: A practical guide to algorithmic  
strategies and trading systems. Hoboken: Wiley.
- [16] Brogaard, J., Hendershott, T. J., & Riordan, R. (2013). High frequency trading  
and price discovery. *SSRN Electronic Journal*.  
doi:10.2139/ssrn.2341037
- [17] Hasbrouck, J., & Saar, G. (2011). Low-latency trading. *SSRN Electronic  
Journal*. doi:10.2139/ssrn.1695460

- [18] Chaboud, A. (2009). *Rise of the machines: Algorithmic trading in the Foreign Exchange Market*. Washington, D.C.: Board of Governors of the Federal Reserve System.
- [19] Mizrach, B., & Fleming, M. J. (2008). The microstructure of a U.S. treasury ECN: The brokertec platform. *SSRN Electronic Journal*.  
doi:10.2139/ssrn.1107933
- [20] Hu, A., Parlour, C. A., & Rajan, U. (2018). Cryptocurrencies: Stylized facts on a new investible instrument. *SSRN Electronic Journal*.  
doi:10.2139/ssrn.3182113
- [21] Parvez, Husain, and Habib Mehrez. “Asif: Application Specific INFLEXIBLE FPGA.” *Application-Specific Mesh-Based Heterogeneous FPGA Architectures*, 26 Oct. 2010, pp. 77–101, doi:10.1007/978-1-4419-7928-5\_5.
- [22] Aldridge, Irene. *High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*. Wiley, 2013.
- [23] Lewis, Michael. *Flash Boys: A Wall Street Revolt*. W.W. Norton & Company, 2015.
- [24] Kirilenko, Andrei, et al. “The Flash Crash: High-Frequency Trading in an Electronic Market.” *The Journal of Finance*, vol. 72, no. 3, 21 Apr. 2017, pp. 967–998, doi:10.1111/jofi.12498.
- [25] “Rule 15C3-5 - Risk Management Controls for Brokers or Dealers with Market Access.” *Small Entity Compliance Guide: Rule 15c3-5 - Risk Management Controls for Brokers or Dealers with Market Access*, 2011, [www.sec.gov/rules/final/2010/34-63241-secg.htm](http://www.sec.gov/rules/final/2010/34-63241-secg.htm). Accessed 05 June 2023.

- [26] “MIFID II: European Securities and Markets Authority.” *MiFID II | European Securities and Markets Authority*, [www.esma.europa.eu/publications-and-data/interactive-single-rulebook/mifid-ii](http://www.esma.europa.eu/publications-and-data/interactive-single-rulebook/mifid-ii). Accessed 05 June 2023.
- [27] *New York Stock Exchange LLC, NYSE ARCA, Inc., NYSE MKT LLC F/K/a NYSE ...*, [www.sec.gov/litigation/admin/2014/34-72065.pdf](http://www.sec.gov/litigation/admin/2014/34-72065.pdf). Accessed 05 June 2023.
- [28] "IEEE Standard for Verilog Hardware Description Language," in *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)*, vol., no., pp.1-590, 7 April 2006, doi: 10.1109/IEEESTD.2006.99495.
- [29] "IEEE Standard for Ethernet," in *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, vol., no., pp.1-7025, 29 July 2022, doi: 10.1109/IEEESTD.2022.9844436.
- [30] Postel, J. “Transmission Control Protocol.” *RFC Editor*, 1 Sept. 1981, [www.rfc-editor.org/rfc/rfc793](http://www.rfc-editor.org/rfc/rfc793). Accessed 05 June 2023.
- [31] Postel, J. “User Datagram Protocol.” *RFC Editor*, 1 Aug. 1980, [www.rfc-editor.org/rfc/rfc768](http://www.rfc-editor.org/rfc/rfc768). Accessed 05 June 2023.
- [32] “What Is Python? Executive Summary.” *Python.Org*, [www.python.org/doc/essays/blurb/](http://www.python.org/doc/essays/blurb/). Accessed 05 June 2023.
- [33] “LwIP 2.1.1 Library.” *AMD Adaptive Computing Documentation Portal*, [docs.xilinx.com/r/2021.2-English/oslib\\_rm/LwIP-2.1.1-Library-v1.6](https://docs.xilinx.com/r/2021.2-English/oslib_rm/LwIP-2.1.1-Library-v1.6). Accessed 05 June 2023.
- [34] Specification of Internet Transmission Control Program, [www.rfc-editor.org/info/rfc675](http://www.rfc-editor.org/info/rfc675). Accessed 05 June 2023.

## Kişisel Bilgiler

Doğum Tarihi: 01/01/1999 Konya – Türkiye  
Askerlik: Tecilli – 2027  
Ehliyet: B  
Medeni Durum: Bekar  
Adres: İstanbul/ Türkiye

## İletişim

E-Posta: [erdembe17@itu.edu.tr](mailto:erdembe17@itu.edu.tr)  
Github: [github.com/erdembehic](https://github.com/erdembehic)  
LinkedIn: [@behic-erdem](https://www.linkedin.com/in/erdembe17)  
Telefon: +90 542 247 44 84

## Dil

Türkçe Ana Dil  
İngilizce İleri – B2-C1  
Almanca Başlangıç A1.1

## Hobiler

Tenis oynamak  
Bisiklet sürmek  
Tüplü dalış  
Psikoanaliz

## Katıldığım Yarışmalar

2022  
Formula Student Czech  
Teknofest Robotaksi  
Teknofest Sağlıkta Yapay Zeka  
Teknofest Ulaşımında Yapay Zeka  
2021  
Formula Student East( 19. takım)  
Teknofest Robotaksi (En yüksek puan)  
2019  
Formula Student Italy  
2018  
ODTU Robot Günleri

## Behiç Erdem

İstanbul Teknik Üniversitesi Elektronik Haberleşme (%100 İngilizce) Mühendisliği bölümünde okuyorum. Çok hareketli bir öğrencilik hayatı geçirdim. Lisede okul başkanlığı yaptım. Üniversiteye başlayınca ise hem sosyal olarak hem de yabancı dilimi geliştirmek için birçok Model Birleşmiş Milletler konferanslarında hem delege hem de organizatör olarak yer aldım. Ana konusu Formula Student yarışmaları olan ve elektrikli otonom & içten yanmalı şekilde araç ürettiğimiz İTÜ Racing takımında 1 sene Gömülü Sistemler ekip üyesi 2 sene de Otonom Ekip liderliği yaptım. Web tasarım, görsel dizayn ve 3D modelleme bilgim mevcut. Lisans bitirme tezi konum HFT sisteminin FPGA kartında gerçekleşmesi. Bu proje için HDL, python ve C yazılım dilleri kullanıyorum. Kişisel olarak kendimi geliştirirken bir yandan da proje yapmayı severim. Python ve AI kullanarak yazdığım birkaç projem mevcut akıllı sistemler, otomasyon ve tarım alanında AR-Ge çalışmaları yapıyorum. Şu anda Argela Yazılım ve Bilişim Teknolojileri firmasında yarı-zamanlı olarak çalışıyorum. Bulut bilişim sistemleri, linux, docker, kubernetes, openstack vb. alanlarda çalışmalar yapıyoruz. Analitik düşünen, gelişime açık, hızlı karar veren, çevreye uyum sağlayabilen ve liderlik yapabilen birisiyim.

## Eğitim

- 2017 - Mezuniyet 2023 Haziran İstanbul Teknik Üniversitesi/ Elektronik ve Haberleşme Mühendisliği (100% İngilizce)  
→ 2015 – 2017 Konya Özel Form Kampüs Anadolu Lisesi (Tam Burs)  
→ 2013 – 2015 Konya Meram Fen Lisesi

## Kurslar

- 2023 Preperation to Openstack COA Exam Udemy – Kris Kelmer  
TCP/IP Training Course Udemy – Infinite Skills  
→ 2022 Intel FPGA Designer Intel  
Certified Kubernetes Admin (CKA) Udemy – Mumshad Mannambeth  
Kubernetes Hand-On Beginner Udemy – Mumshad Mannambeth  
Docker & Kubernetes Complete Guide Udemy – Stephen Grider  
Docker for Absolute Beginners Udemy – Mumshad Mannambeth  
Linux Edx – Linux Foundation  
→ 2021 Embedded System Essentials with ARM - 1 & 2 Edx – ARM Education  
Robot Operating System 1 & 2 Udemy – Aanis Kouba  
→ 2020 Python ve Veri Bilimi Udemy – Engin Demiroğ  
→ 2019 Biyometrik Yüz Tanıma Haliç Akademi  
NLP ( Neuro Linguistic Programming ) Haliç Akademi  
HTML & CSS Edc – World Wide Web Consortium  
→ 2018 C/C++ kursları Youtube

## Yetenekler

C & C++	Orta – İleri	MS-Office	İleri
Python	Orta – İleri	Adobe Design Tools	Orta
Assembly	Orta	Vivado	İleri
Verilog	İleri	Intel Quartus Prime	Orta
HTML/CSS	İleri	Matlab	Orta
Bash	Orta	CST Studio	Orta
TCL	Orta	LT Spice	Orta
		AWR	Orta


## Çalışma Hayatı ve Tecrübeler


- 2022 Mayıs – günümüz Yarı-zamanlı Bulut Orkestrasyon ve Sistem Mühendisi Argela Yazılım ve Bilişim  
2021 Yaz Stajyer Baykar Savunma – Elektronik Sistem Geliştirme  
2021 Yaz Stajyer Tübitak BİLGEM – Tüm Devre Elektronik Tasarım Laboratuvarı  
2020 Yaz Stajyer Baykar Savunma – RF Mikrodalga Ekibi  
2020 – 2022 İTÜ Racing Takımı Otonom Ekip Lideri  
2019 – 2020 İTÜ Racing Takımı Elektronik Ekip Üyesi  
2018 ve 2019 yılları arasında ITU MUN, IKUMUN, İTÜ MDK DFS Zirvesi, MarmaraMUN gibi etkinliklerde IT Koordinatörü ve organizatör olarak yer aldım.  
2017 Mayıs -2019 Peer Mentor ProBro Akademi


# Rana Tilki




## PERSONAL

 Name  
Rana Tilki

 Phone  
0535 024 34 90

 E-mail  
ranatilki@hotmail.com

 Birthday  
18.03.1999

## Languages

English ●●●●○  
French ●○○○○  
German ●○○○○

## Education

- 2018 - present Electronics and Communication Engineering (100% English)  
Istanbul Technical University, Istanbul  
Senior student  
Current GPA: 3.26 / 4.00
- 2017 - 2018 Medicine  
Aydın Adnan Menderes University  
Drop out GPA: 2.75 / 4.00
- 2014 - 2017 Izmir Science Highschool  
GPA: 95.66 / 100

## Experience

- June 2021-July 2021 TUBITAK BİLGEM  
Digital Design Engineering Intern
- August 2021-September 2021 Arçelik  
Hardware Engineering Intern
- October 2021-June 2022 ITU Racing Autonomous Team  
Team Member
- August 2022-September 2022 Procenne  
Embedded Software Engineering Intern
- October 2022-February 2023 BAYKAR  
Digital Design Engineering Intern
- March 2023-Present BAYKAR  
Digital Design Engineer  
Part-time

## Significant Courses Taken

- VLSI Circuit Design II (This Term)
- Satellite Communications (This Term)
- Digital System Design Applications
- Data Communications
- Introduction to Embedded Systems
- Microprocessor Systems
- Analog Electronics
- Digital Electronics
- Introduction to Logic Design
- Introduction to Sci & Eng Comp. (C)
- Data Structures & Programming (C++)
- Digital Signal Processing
- Analog Communications
- Digital Communications
- Electromagnetic Waves
- Object Oriented Programming

## Skills

- C
- Verilog
- C++
- MATLAB
- LTspice





# Armin Asgharifard

Student

## 👤 Profile

A motivated and detail-oriented Electronics and Communications Engineering student, currently in the fourth year of my Bachelor's degree at Istanbul Technical University. Skilled in software programming, with a strong focus on mobile app development, and experienced in Java and Android app development. Additionally, experienced in digital hardware design, including FPGA, ASIC, and embedded network design. Possessing excellent time management, planning, and team work skills, I am eager to apply my skills and knowledge to real-world projects in a dynamic and challenging work environment. I am confident in my ability to quickly learn new technologies and work towards achieving my career goals.

## 🎓 Education

**Electronics and Communications Engineering , Istanbul Technical University, Istanbul**

September 2019

Bachelors degree

GPA: 3.46/4

## 📁 Work Experience

**Internship at Istanbul Technical University Embedded System Design Laboratory, Istanbul**

June 2022 — July 2022

FPGA design with VHDL

## ★ Projects

**Comparator Circuit in FPGA**

2023

As an FPGA design project with Verilog, an ALU and a Control Unit were designed and connected together to build a digital circuit, in which given two inputs A and B, the circuit compares 2A and B and produces the corresponding output.

**RAM Data Swapper Using Assembly**

2023

As an embedded network design project, a PicoBlaze microcontroller was programmed to swap the contents of two Block RAMs.

**Developing a Game Using Assembly**

2022

A game called 'Stacker' was developed using Assembly programming language, where you put the blocks on top each other to make a high building.

## Details

Istanbul

Turkey

+90 552 661 04 75

[asgharifard19@itu.edu.tr](mailto:asgharifard19@itu.edu.tr)

## Links

[Linkedin](#)

## Skills

Leadership

Effective Time Management

Planning

## Hobbies

Piano playing

Running

## Languages

Azerbaijani

---

Persian

---

Turkish

---

English

---

## **Dadda Tree Multiplier in FPGA**

2022

As an FPGA design project with Verilog, a Dadda Tree Multiplier was implemented to multiply two inputs.

## **Sequence Detector in FPGA**

2022

As an FPGA design project with Verilog, a sequence detector circuit was designed, in which two different 4-bit sequences can be detected, including overlapping.

## **Image Filtering in FPGA**

2022

Using Verilog, a circuit was designed to receive the bits of an image, apply smoothing filter on it, and produce the bits of the resulting image.

## **🦋 Activities**

**Active Member of ITU IEEE ComSoc Club at Istanbul Technical University**

2019 – 2020

## **🎯 Courses & Certifications**

**Beginner and Intermediate Java Course, with a specialization in Android app development, Tabriz Institute of Technology, Tabriz, Iran**

June 2019 – September 2019

**First place in Iran High Schools Music Competition Classical Piano Category, Iran Culture and Art Organization**

2017

## **★ Computer Skills**

Java

VHDL/Verilog

C/C++

Assembly

Python

