Hardware Accelerated High-Frequency Trading System

Final Report

TA:
Mingjia Huo

Team Members:
Siyi Yu(siyiyu2)
Kevin Lim (wzlim2)
Richard Deng (ruichao4)

Dec 2nd, 2022

Contents

1 Introduction

1.1 Problem

Modern electronic markets have been characterized by a relentless drive towards faster decision making. Significant technological investments have led to dramatic improvements in latency, the delay between a trading decision and the resulting trade execution. We describe a theoretical model for the quantitative valuation of latency. A low latency for the communication, which refers to sending the order to exchange and receiving the order from exchange, is critical since sometimes a delay of milliseconds could result in a loss of millions dollars.

“Low latency” in a contemporary electronic market would be qualified as under 10 milliseconds, “ultra low latency” as under 1 millisecond. This change represents a dramatic reduction by five orders of magnitude. To put this in perspective, human reaction time is thought to be in the hundreds of milliseconds.

In the financial market today there is a lot of need to optimize the trading/execution latency to support automated quantitative trading systems. While most of the computer software runs on generic operating systems and the CPU executing the logic has many parts of unnecessary instructions/procedures, the automated trading strategy can be highly optimized with hardware to achieve low-latency and high-frequency.

1.2 Solution

The purpose of this project is to prototype a hardware based trading tool that can perform such a high-frequency and low-latency trading. while being able to communicate with the exchange using basic network protocol.

We plan to build a trading system that uses one PCB to connect to a fake exchange and get/send binary market data and use highly optimized FPGA to consume this market data and make decisions based on the data. Since we will not be able to directly connect to the exchanges without approval, we are also going to simulate an exchange using software. Most of the existing low-latency trading strategy will be implemented using FPGA solely. However, our project will utilize an ESP8266 chip for networking purposes.

The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability. FPGAs which could accomplish this task are pretty high end and relatively expensive. FPGAs tend to offer vastly greater flexibility which we do not need for this project. It will help us bring down the cost while maintaining a relatively good performance of the system.

1.3 Visualization

Figure 1 illustrates the overview of our project design.

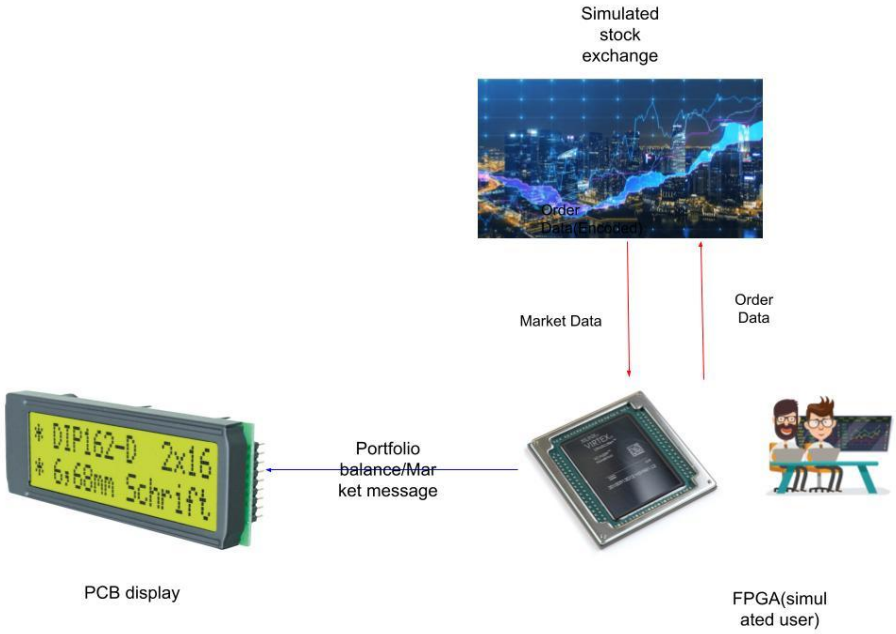

Fig 1: Visualization diagram

1.4 High level requirements

Below is the high level requirements for our each individual subsystem:

- PCB board is able to display the current balance and message
- Exchange is able to correctly process the orders
- Tradeing strategy is able to correctly to make the most sensible decision

We are also aiming to get the network communication between FPGA(simluated user) and simulated exchange working. However, it might be beyond our ability since the complexity of the networking.

2 Design

2.1 Design procedure

2.1.1 Simulated Exchange

This is a simulation of real-world exchange. It can receive orders from market participants and automatically generate trade messages and broadcast it back to market participants.

The complete data flow of the simulated exchange is illustrated in the Fig. 2 below.

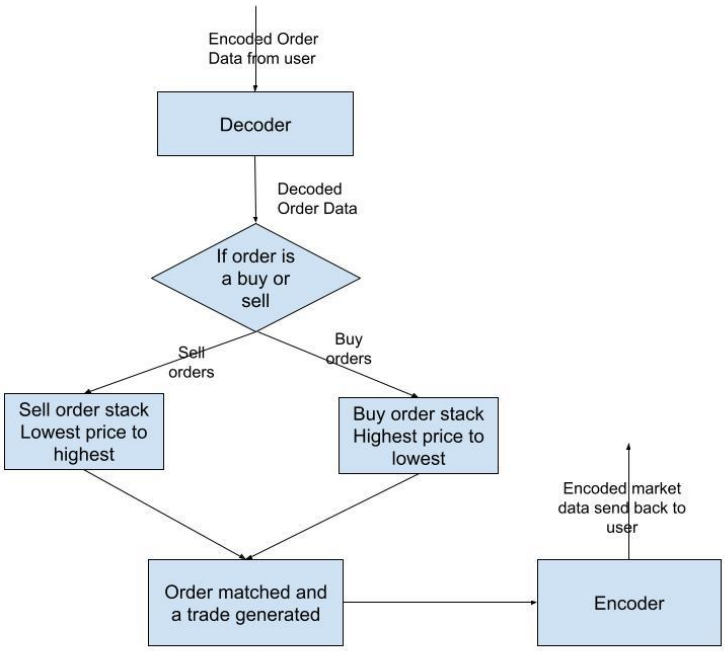

Fig. 2: Data flow of simulated stock exchange

After discussion, we decided to implement it using C++ code. The design of this subsystem has not changed much throughout our development.

2.1.2 PCB

Our initial design is to put a UDP network chip on PCB board and connect it to FPGA to service as a way of communication between FPGA and simulated exchange. However, after seeking advice to professor and TA, we found out it might not be a good option so that we decided to change our design.

Our final PCB design consists of two 4-digit LED lights to display the balance amount and one message board to display any message FPGA generated or received from simulated exchange.

2.1.3 FPGA Trading Strategy

This project is using a market-making strategy based on the paper *An FPGA-Based High-Frequency Trading System for 10 Gigabit Ethernet with a Latency of 433 ns*[8]. Our FPGA design generally consists of three parts:

- A networking part that receives UDP packet and sequence it using the sequence number built-in from the second byte in the message.
- A decision maker that creates the best spread in the market without sending it to the exchange. Once it receives an order that falls in its spread, it is intended to send a counter-offer before anyone else in the market does.
- An encoder that encodes the decision following the protocol specified by the exchange and send it back to the exchange.

2.2 Design details

2.2.1 Simulated Exchange

This is a simulation of real-world exchange. It can receive orders from market participants and automatically generate trade messages and broadcast it back to market participants. This simulated exchange consists of three parts:

I. Order Data Decoder. This decoder will decode the binary order data sent by market participants into internal, more organized, human-readable data and feed it to the orderbook.
II. Orderbook. This is where all the orders from different market participants are stored, and it will generate a trade if the bid side and ask side meets.
III. Market Data Encoder. This encoder will consume the data from the orderbook and encode it into binary market data and broadcast it to market participants.

The simulated exchange will be a piece of software that simulates an exchange in real-time. This simulated exchange will only have one security to trade and will be receiving three types of binary-encoded messages (add order, cancel order, update order) from the market participants using a certain protocol, building a full-depth limit order book based on the order received, automatically trade two orders when the bid side and ask side meet, and re-broadcasting the trade message with the other three client messages in binary encoded form as market data messages.

To test our strategy we will simulate the market data received from market participants, constantly adding limit ask/bid at the same price level in high frequency. We will expect 100% of the orders to be processed correctly. This means an ask order and a bid order with the same price will not be existing in the order book, since it will generate a trade. This part of the project is designed to be run completely by software so there is no power supply unit involved in the subsystem.

2.2.2 PCB

Our PCB board is responsible for visualizing some of the market data. We select two variables to display: portfolio balance and market message. The reason we made these two choices is based on the feasibility and time consumption to implement. Portfolio balance could be easily display using two LED board so that become our first choice. We also notice that market message is also very important for stock trading so we decide to display these information on PCB board.

The overall PCB design schema is show in the Fig. 3 below.

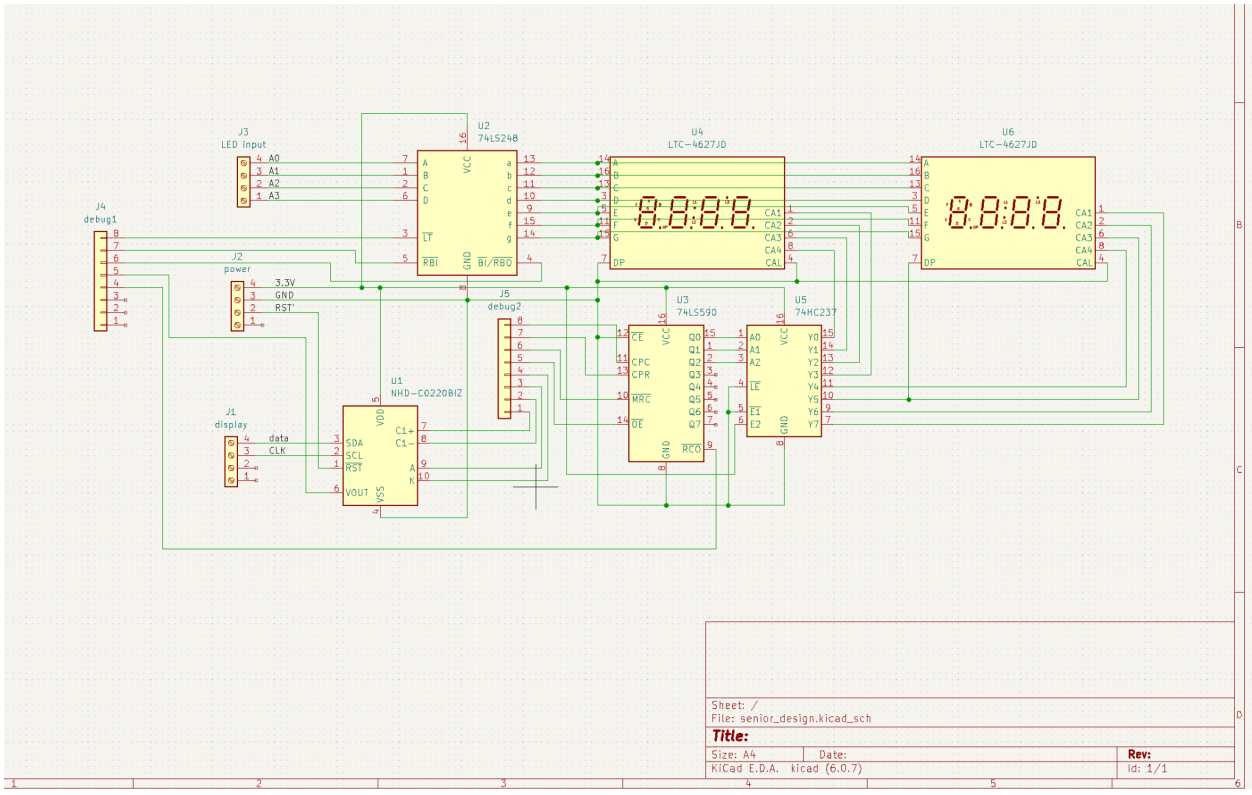

Fig. 3: PCB design schema

The PCB design can be splited into two part: digit LED screen part and market message display part.
Digit LED screen is consisted of the following components:

- Two four-digit LED screens which is responsible for displaying the numbers
- A 74LS590 counter which is responsible for the syncrnization between two LED screens
- A 74HC237 MUX which is responsible for directing each digit to the corresponding position on LED
- A 74LS248 Seven Seg Decoder to translate the input digit to the signal which LED could understand

Message board is being driven by Arduino. It will display waiting when we initialized the system and it will display 'Order Sent' when there is an order being generated.

2.2.3 FPGA Trading Strategy

This part is an FPGA board that consumes market data from the networking hardware and generates decisions based on the market data. This consists of two parts:

I. Decision maker (Strategy). This part is the core trading strategy that takes in market data in real-time and makes decisions based on some logic that directly manipulates the binary market data to optimize speed.
II. Order Encoder. This part will take in the decisions that have been made from our strategy, and encode it to binary and send it back to the networking chip, which will eventually be sent to the exchange.

In order to simplify the process(since trading algorithm is not the key part of this project), we plan to implement two simple strategies:

1. High-frequency market-making: This is a liquidity-providing trading strategy that simultaneously generates many bids and asks for a security at ultra-low latency while maintaining a relatively neutral position. When put into implementation, it will make a spread of $a ~ $b by sending limit bid at $a and limit ask at $b, adjust the spread based on Best Bid Offer (BBO) market data. This part is to test how fast our trading system can be adjusted based on real-time information changes.
2. High-filling rate limit order: A limit order is used to buy or sell a security at a predetermined price and will not execute unless the security's price meets those qualifications. A High-filling rate limit order trading strategy sends a limit bid order whenever there is a limit ask order at $a. This part is designed to test the

latency of our trading system and is extremely useful in options trading where the bid-ask spread is very large and has a low order filling rate.

These two trading algorithms will be sufficient enough for us to mock the real world trading cases and allow us to test the system.

3 Verfication

3.1 RV table and results

3.1.1 Simulated exchange

RV table for simulated exchange is shown in Table 1 below.

Requirements	Verification	Fulfilled
80% accuracy for order data decoding	Encode several messages according to the protocol, send it to the exchange and check whether the data we sent is in the system	Yes
Reject binary data that doesn't follow the protocol	Send some random binary data to see whether it's rejected by the exchange	Yes
orders are being stored in orderbook and be able to modify	Send orders to the exchange to see whether it's in our system. Modify it later and see whether it's modified in our system.	Yes
Trade is generated automatically and properly	Send random orders and make sure best bid price and best ask price never meets	Yes

Table 1: RV table for simulated exchange

3.1.2 PCB

RV table for PCB board is shown in Table 2 below.

Requirements	Verification	Fullfilled
Digit boards correctly display portofolio balance	Feed decoder with changing numbers and check if two screens are displaying correct numbers and syncrizing	Yes
Message screen	Feed driver with random message to see if the display board will display the corresponding	Yes

correctly display market message received	message	

Table 2: RV table for PCB board

3.3 FPGA Trading Strategy

RV table for FPGA Trading strategy is shown in Table 3 below.

Requirements	Verification	fullfilled
Correctly process the data	● Implement same strategy using python, using the same input test data ● Verify whether the python strategy outputs the same decision as the FPGA ones	Yes
30% faster than software strategy	● Implement same strategy using python, using the same input test data ● Verify whether the FPGA strategy is 30% faster than the python one's	Yes

Table 3: RV table for FPGA Strategy

4 Cost

4.1 Labor

Name	Hourly Rate($)	Total Hours	Total
Ricahrd Deng	33	144	4752
Kevin Lim	33	144	4752
Siyi Yu	33	144	4752
Total	99	432	14,256

4.2 Parts

Parts	Part#	Quantity	Unit Cost($)	Cost($)
FPGA	DE10-Nano	1	$135	$135
PCB	-	1	$0	$0
OR gate	SN74HC32N	20	$0.5	$10
AND gate	CD4085BE	20	$0.5	$10
Register	HCF40100BE	20	$2.5	$50.0
LED lights	LTC-4627JD	4	$2	$8
Message board	NHD-C0220	2	$8	$16
Counter	74LS590	2	$1	$2
Decoder 7 Seg	74HC237	2	$1	$2
1 to 4 Decoder	74LS248	2	$1.5	$3
Bread board	-	1	$0	$0

5 Conclusion

5.1 Future Improvements

We faced many challenges while developing our project. First of all, we made mistakes while designing the PCB. Specifically, some components didn't fit into the PCB as expected, including the two LED digits display. Since we only submitted one PCB to order, we couldn't connect all the components without using a breadboard, so our final product had quite a few components on the breadboard. One future improvement then is to fit all components in a well designed PCB.

Another future improvement is implementing the message board. The message board was introduced to our design relatively late, so we faced some difficulties setting up the message board and getting it to work before the deadline. We burned one of our message boards after having smoldered it to the PCB already, so we had to use another message board and connect it to the system using a breadboard. We also discovered that it was quite difficult to implement the message board driver with an FPGA, so we decided at the last minute to use an Arduino instead to program the message board. In addition to the PCB and the message boards, we also felt that the network communication between exchange and FPGA can be improved. A more robust communication system would have yielded faster response times.

5.2 Ethics & Safety

As we finalized our design in the design document, we identified 3 major concerns of ethics from the Association for Computing Machinery (ACM) Code of Ethics and Professional Conduct for our project.

Respect privacy (ACM 1.6)

- Our design may enable users to collect and trade personal information, which violates the privacy of the user. We shall protect the privacy of our users and make sure that no one is able to access personal information.

Honor Confidentiality (ACM 1.7)

- Our design may process highly valuable information such as trade secrets, financial information, and client data. We must not share this data with anyone, and we must not access the information ourselves.

Design and Implement Systems that are Robust and Usably Secure (ACM 2.9)

- A fragile or easily breakable design will likely lead to data leakage among other problems. We should make sure that our design is robust and all information is secure.

One concern regarding safety was the power supply. If not handled properly, there is a risk of overcharging which could damage the PCB. We also discovered while developing the project that smoldering can be quite dangerous if not done properly. We made sure to use a wet sponge to cool down and turn off the smoldering power supply after we are done.

In developing the project, we were aware of possible ethical and safety concerns and made sure to follow the code of ethics from ACM and the Institute of Electrical and Electronic Engineers (IEEE).

6 References

[1] Ultra Low Latency Networking with FPGA, 2020.
https://www.youtube.com/watch?v=32cK_yDcouQ

[2] UDP - ESP8266 Arduino Core, 2017.
https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-examples.html

[3] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct”, 2018. [Online]. Available: https://www.acm.org/code-of-ethics.

[4] Institute of Electrical and Electronic Engineers, “IEEE Code of Ethics”, 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html.

[5] 8-Bit Binary Counters With Output Registers datasheet, 1988.
https://www.ti.com/lit/ds/symlink/sn74ls590.pdf?ts=1667257886873&ref_url=https%253A%252F%252Fwww.google.com%252F

[6] High-Speed CMOS Logic, 3- to 8-Line Decoder/Demultiplexer with Address Latches datasheet, 2003.
https://www.ti.com/lit/ds/symlink/cd74hc137.pdf?ts=1667207383378&ref_url=https%253A%252F%252Fwww.google.com%252F

[7] BCD-to-Seven-Segment Decoders/Drivers datasheet, 1988
https://www.ti.com/lit/ds/symlink/sn74ls247.pdf?ts=1667237244274&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74LS247

[8] Y. -C. Kao, H. -A. Chen and H. -P. Ma, "An FPGA-Based High-Frequency Trading System for 10 Gigabit Ethernet with a Latency of 433 ns," 2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), 2022, pp. 1-4, doi: 10.1109/VLSI-DAT54769.2022.9768065.