HARDWARE ACCELERATED HIGH-FREQUENCY TRADING SYSTEM Project Proposal

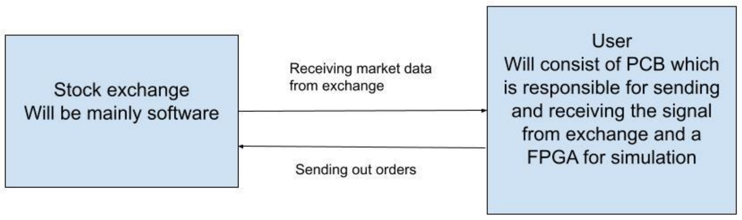
Introduction

Problem

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 run 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

Solution

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. As shown in the diagram D1 below.


Stock exchange
Will be mainly software
Receiving market data from exchange
Sending out orders
User
Will consist of PCB which is responsible for sending and receiving the signal from exchange and a FPGA for simulation

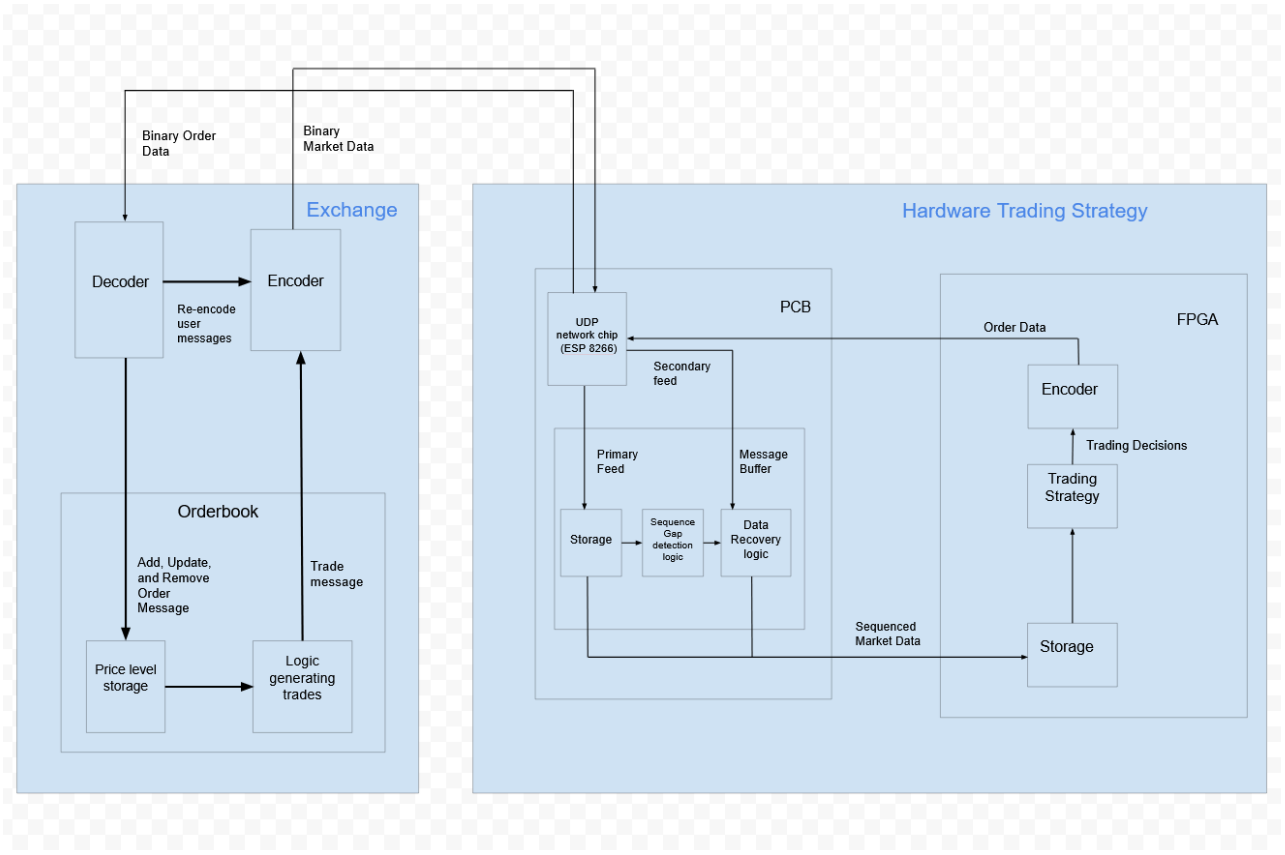
D1

High Level Requirements

1. 100% orders being successfully sent to exchange and received by users
2. 30% faster than using purely software implementation of this system
3. 100% orders being correctly processed by the exchange

Design

Block Diagram
The overview of our block diagram is shown in the diagram D2 below. It contains all the subsystems we intended to complete for this project.



D2: Block Diagram

Subsystem Overview
Our design will consist of two main subsystems. One is the simulated stock exchange and the other one is the simulated user.

Inside the exchange part, a decoder is used for translating the binary code received by the stock exchange to the human readable data which is price, quantity and ticker symbol in our case. It will pass the info to the order book and encoder.

Encoder is responsible for translating the order and market info into binary code and sending it to the user.

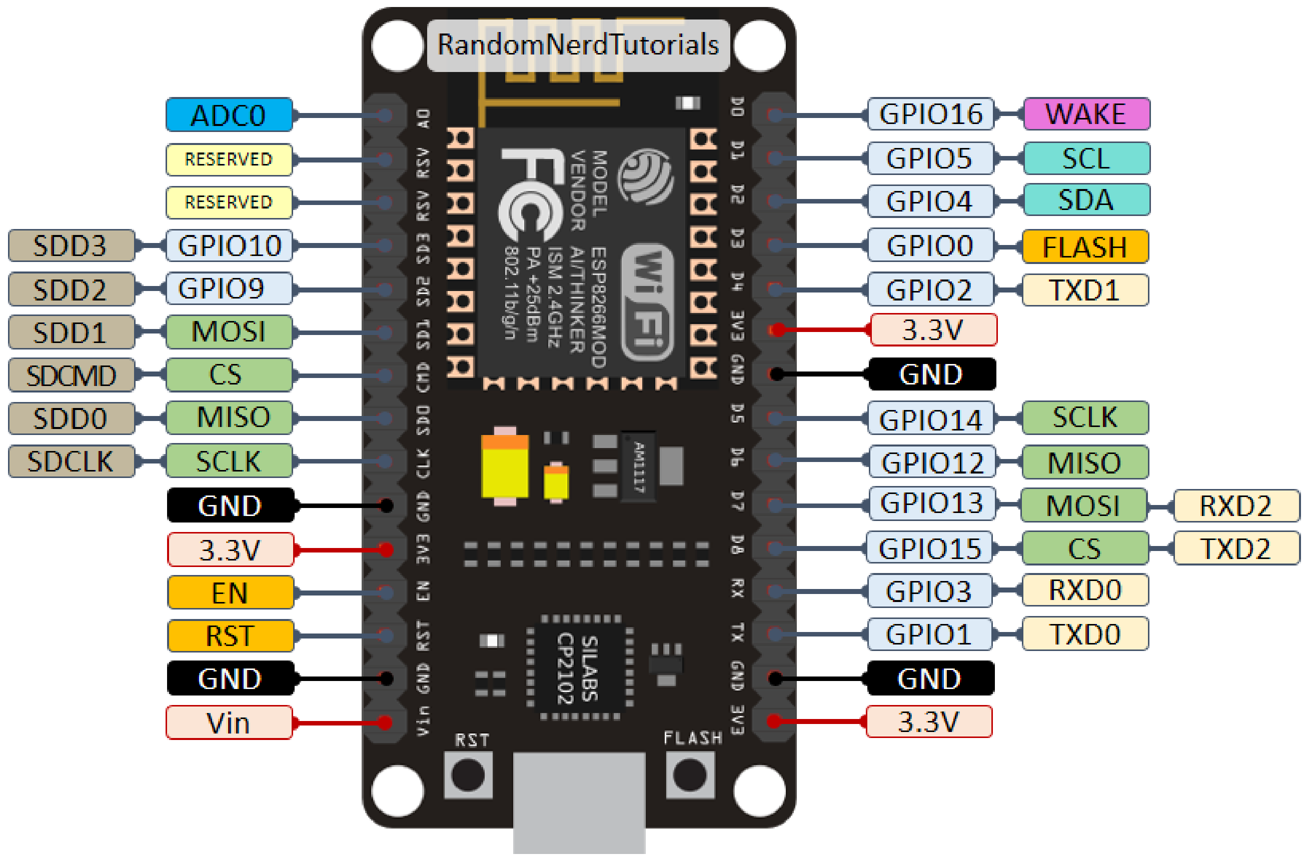
Orderbook where the matching orders are kept. Apart from keeping the orders, it will also match sell orders to its corresponding buy orders so that a deal could be made. After it generates trades inside the orderbook logic, it will send this trade information back to users after being encoded as binary data.

Inside the Hardware Trading Strategy part, the binary market data will be received by PCB and using a sequence gap detection and buffering method (described below), it will generate a sequenced market data stream and send it to our FPGA. Our FPGA will use this data to generate trading decisions, and then send it back to the exchange after encoding it to binary followed by the exchange's protocol.

Subsystem Requirements

We will develop a simulated stock exchange. This will be a piece of software that simulates an exchange in real-time. This fake 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 following the same protocol. 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 all the sell orders should be matched to the corresponding buy orders. This part of the project is designed to be run completely by software so there is no power supply unit involved in the subsystem.

Networking hardware will be responsible for the communication between users and the stock exchange. It will be made of a PCB with ESP8266(as shown in the picture P1 below) chip on it and other parts/ports to communicate with the simulated exchange through the network. We plan to use UDP since it is what most exchanges will use. This part will be optimized to reduce the connectivity latency between the fake exchange and the trading system and feed the data to the FPGA. Without this component, there will be no way for users and stock exchanges to communicate with each other. Since ESP8266 requires a power supply for it to be functional, we would need a power supply for this subsystem. The ESP8266 requires a 3.3V power supply and 3.3V logic levels for communication and a maximum of 170mA current.



P1

We will also need to in order for the system to simulate users and their trading strategy like shown in the picture P2 below. This will be realized by an FPGA which is optimized for trading strategy, making decisions based on the binary data received from the Networking Hardware, sending the decision back to the networking PCB and which will send it back 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.

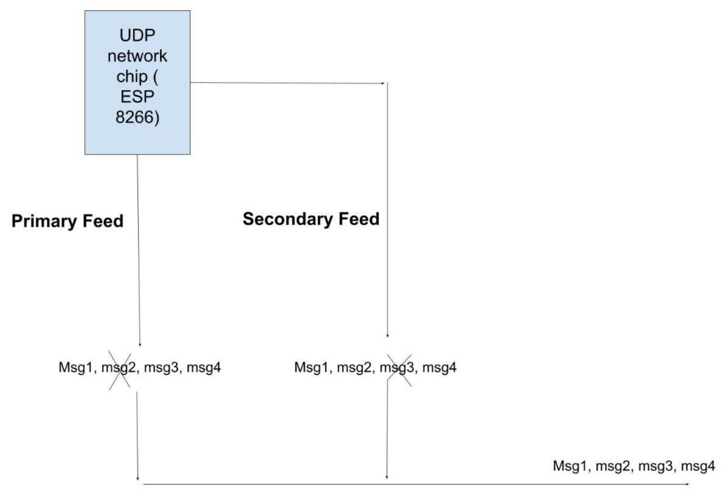
These two trading algorithms will be sufficient enough for us to mock the real world trading cases and allow us to test the system. Since this system makes use of FPGA, we will need a power supply unit for this subsystem.

P2

Tolerance Analysis

Since we are using UDP as our network communication protocol, it might fail to deliver some of the packets. It is very critical for us that every single packet is received, otherwise we cannot process the market data correctly. However, we designed a sequence gap detection and recovery method built in the PCB, only when both primary data feed and secondary data feed fails on the same packet we will experience a failure, thus our system has high tolerance for network failure.

A diagram shown below demonstrates when message 2 drops on primary feed and message 3 drops on secondary feed, we can still recover all 4 messages after combining them.



Ethics and Safety

We have identified 3 major concerns of ethics and safety 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 will need to 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.

There are also minor concerns that are not covered by these 3 topics. We will make sure to follow all ethics and safety guidelines.