High-Frequency Financial Trading Platform Accelerator

高頻金融交易平台加速器

組別：B124　指導教授：馬席彬教授 組員：107061221 陳謙謙 107061140 李孟佳

Abstract

S Speed and accuracy are the most essential elements in High-Frequency Trading(HFT). However, trading nowadays is done by software. Restricted by the sequential execution of a process in software, orders cannot be placed at the best time to make a profit. Thus, Field Programmable Gate Array (FPGA) is chosen for implementation so that we can place the orders in the shortest time when new market data packages are received, benefited from the parallelism and high-speed calculation characteristics of the hardware.

We have done various HFT related simulations, designed algorithms with Verilog-HDL, implemented and verified our designs on the Alveo U50 Data Center Accelerator Card. These were first validated with C/C++ or Python as coding in software languages is relatively easier than in HDL. As we collaborate with VSense Fintech Inc., some details are classified and will not be disclosed. The works of another group (Group CS), will also be discussed briefly to give a full picture of this research. Their parts will be credited in their subsections.

We began this research by understanding the anatomy of Internet packets(Section A) as the market data(inputs) will come in that form. To reinforce our understanding, we built a C/C++ program to analyze the incoming packets and obtain required information, such as futures' prices, and quantities.

Next, we designed and simulated a Futures' Order Book (Section B and C) which updates and stores vital information like futures' prices, quantities, etc., to form a new product, *Synthetic Option*. We also designed its related trading strategy (Section D and E) that predicts if the latest traded future's price aligns logically with the market situation. An order decision will be made based on this information.

Finally, we designed some Trading Strategies for Compound Options and Futures (Section E, F, and G). The objective is to predict a point, *Spot*, which is crucial in developing most HFT-related strategies. It has to be noted that there neither is a "correct" value of *Spot* nor can any financial model predict it flawlessly. However, the value of *Spot* is essential and most of the trading strategies stem from it. Different from the order book strategies, *Spot* enables us to predict the future price of futures, observe unreasonable market data, such that

we can make a profit by placing corresponding orders. By three different methods of approximation and regression to find the *Spot* by programming, we have also designed hardware methods based on the utility of similar triangles and binary search to eliminate the utility of division while calculating. Afterward, with the obtain of *Spot*, and based on the "Call-put parity" principle in economics, we analyzed the best option types and prices for ordering, profiting from the high-frequency trading.

Introduction

Nowadays, financial technology has been mostly performed through software and programming. In other words, the rate of placing new orders or updating orders for our sake to make the most profit is limited by the computations and sequential processes of software. Thus, the company, VSense Fintech Inc. is developing hardware solutions for financial trading purposes.

For the special topic of implementation, we have been working with the company as it is an "academy-industry cooperation". By the implementation of hardware, we can benefit from the parallelism characteristic of hardware and the high speed of computing. Multiple orders can be done speedily and this will increase our real profits from trading in the real-life financial market.

Our hardware developments focus on mainly two kinds of financial products. The first kind of product is the regular order book products in which we aim to place our orders at the 1st rank of the five-ranked order book to be executed with a higher possibility. The other kind of product is the compound options and future. For this kind of product, we have come up with a simple strategy to be implemented and our design of performing division and comparisons by advanced binary search.

Throughout this special topic of implementation, we have mostly performed software simulations in Python, C, C++ then performed hardware design by Verilog-HDL. These logic designs are implemented on the Alveo U50 Data Center Accelerator Card, which is a typical FPGA board designed for massive computing.

Theory Analysis and System Design

Although we have done both software simulations and hardware design, we will focus mostly on the hardware design in this abstract. Since we have been working on this project with another group from the CS department, a proportion of research have been discussed with them and a proportion of implementation was also of their work.

STRATEGY OF THE ORDER BOOK

For order book products, the trading is performed in the way of sorting orders from all sellers and buyers through an order book, in which orders with higher rankings are most likely to be executed. Accordingly, the Ethernet packets we receive from the company show the best 5 ranking prices and quantities for each product.

Hence, the objective of the strategy developed here is to form a new product by the best price and best quantity of an order book product. This information is obtained by updating the market data received from the “Taifex Message Protocol”.

To implement order book strategies, we reconstructed the market data to observe the market status. We formed the new product, *Synthetic Option*, by combining CALL (the right to buy a stock) and PUT (the right to sell a stock) to make profits. The best ask/bid price of synthetic future and the best ask/bid quantity of synthetic option for each strike price are the statuses of *Synthetic Option*. *Ask* and *bid* indicates demand and supply respectively. The status will be sent to the *Trading Strategy* modules mentioned in Section G to decide whether to make an order. Detailed block diagrams are shown in Fig C.1.

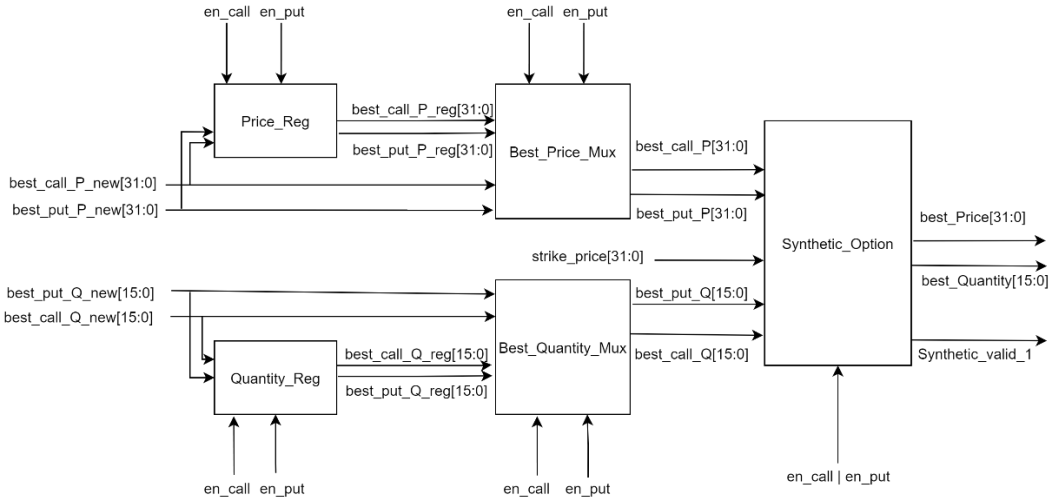

Fig C.1 Modules of Synthetic Option Orderbook

The formulas of calculating the best price and best quantity of ask and bid respectively are derived from the financial characteristics of the market data, shown in formula C.1 & C.2

best price = strike price + best call price – best put price (C.1)

best quantity = min(best call quantity, best put quantity) (C.2)

To perform this task, we choose between the latest market data and the previous best

values to be forwarded to the calculation module and decide the newest best price and best quantity. Hence, the registers and value of multiplexers are updated whenever new market data is sent (*en_call* or *en_put*) and the status of the *Synthetic Option* is also updated. Since the logic of finding the best ask price/quantity and the best bid price/quantity are the same, modules in Fig C.1 are used for both ask and bid.

These modules are used to find the best ask/bid price and best ask/bid quantity for a specific strike price. We can duplicate these modules for each strike price to detect the status of the synthetic option under each strike price in parallel. In this way, we can achieve the efficient calculation of high-frequency trading.

STRATEGY OF THE COMPOUND OPTIONS AND FUTURES

The trading of compound options and futures requires our logic modules to output a package with the "R01 format" mentioned in the "Taifex Message Protocol Regulations Manual"[1], which includes information such as the header, product name, price, quantity, execution type, etc.

The core of our strategy design of compound options and futures consists of two parts: 1. Observe the incoming packages for the latest market situation and try to place our order at the first rank of the market's order book. 2. After our order has been executed, hedging is needed so that we can make a profit.

The reason we need a hedging process refers to the "call-put parity" in future trading economics. "Call-put parity" is a principle that defines the relationship between the price of European put and call options of the same class. The same underlying assets, strike price, and expiration date are required to be in the same class. The put-call parity can be determined by an equation $C + PV(x) = P + S$, where: *C = price of the European call option*, *PV(x) = the present value of the strike price (x), discounted from the value on the expiration date at the risk-free rate*, *P = price of the European put*, and *S = spot price or the current market value of the underlying asset.*

To exercise the strategy, we have to find the point, *Spot*, first before getting into making the orders. In Layman's terms, *Spot* can be understood as the logical price in which a buyer and seller agree to trade, and the point *(spot, spoty)* is theoretically believed to exist around the center of all the blue and orange points (red square in Fig 2.). It needs to be noted that no one has ever accurately determined the exact location of this mysterious point *(spotx,spoty)*. However, finding this point is of utmost importance in High-Frequency Trading (HFT) as almost every kind of trading strategy stems from here.

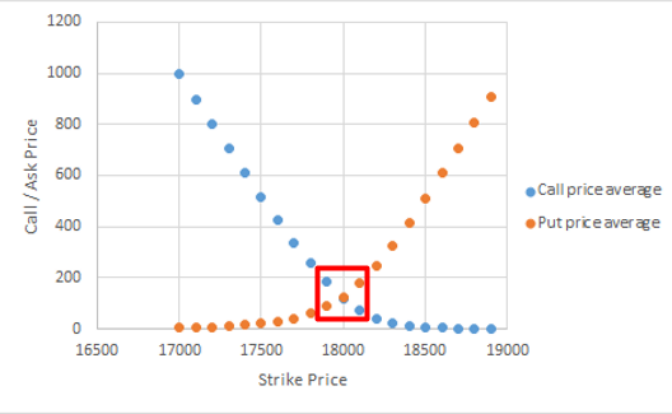

Fig 2. Graph of Call/Put Average Prices Calculated from Market Data

We've applied three methods: 1. Two Linear Equations Approximation, 2. Two Linear Equations and Weighed Correction, 3. Quadratic Regression by Python/C++ to find the *Spot*. Since we have developed the three methods to the extent that the results are similar and all will tolerable error, we chose the "Two Linear Equations Approximation" method to be implemented into hardware logic. Here, to apply our algorithm to hardware-based design, we have developed the "Advanced Binary Search" method to avoid the utility of dividers.

The method of "Advanced Binary Search" is based on the fact that our equation to calculate the *Spot* by division can be retransformed into an equation that only requires multiplication and addition, and in which our *Spot* will be found when the equation equals zero. Hence, we will have to search through all the possible ranges of *Spot* and plug the values into the equation, then find the values whose outcome of the equation is closest to zero. To increase the precision of searching, we divided each range of searching into 16 divisions and calculated *spotx* and *spoty* independently, owing to the good characteristics of the financial parameters.

The method to find the values whose outcome of the equation is closest to zero is based on this idea is called "double elimination" which is often used in real life like competitions and tournaments. By comparing two values, the winner will be forwarded to the winner's brackets while the loser is forwarded to the loser's brackets. In those brackets, two winners from the previous comparisons will meet each other and be compared once again; losers vice versa. Hence, by comparing all the 17 outcomes of the equation with sufficient times, *spotx* and *spoty* can be found. Finally, the required financial parameters *expected call price*, *expected put price*, *the 4 differences of market prices and expected prices*, *call delta*, and *put delta*, etc. are calculated based on the *Spot* and sent to the modules where the ordering strategies are developed. Fig3. shows the top view of the modules to calculate the *Spot* and the financial parameters.

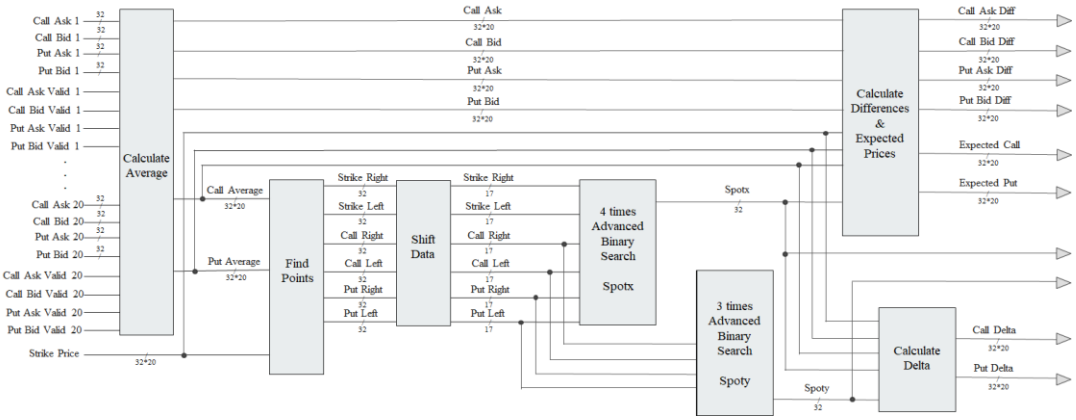

Fig 3. Modules of Processing Financial Parameters

The strategy of making the main order is developed by the following flow: To keep up with the latest market data and adjust our orders, we first observe the trend of future prices. By the price rise or price fall, we will decide whether to bid/ask put/call options by the difference calculated from the previous section. Next, to decide the price of our order, we will first start pricing from slightly higher/lower of the current market option price and take the expected value calculated from the previous section as a reference to see whether the order is worth it. This will give us information on how much an order can profit, whether it should be repriced, or whether it should be canceled if it has been thrown but not executed. Fig 4. shows the top view of the modules to decide the main order. There are more designs under each module but the descriptions of those will not be disclosed in consideration of the classified materials.

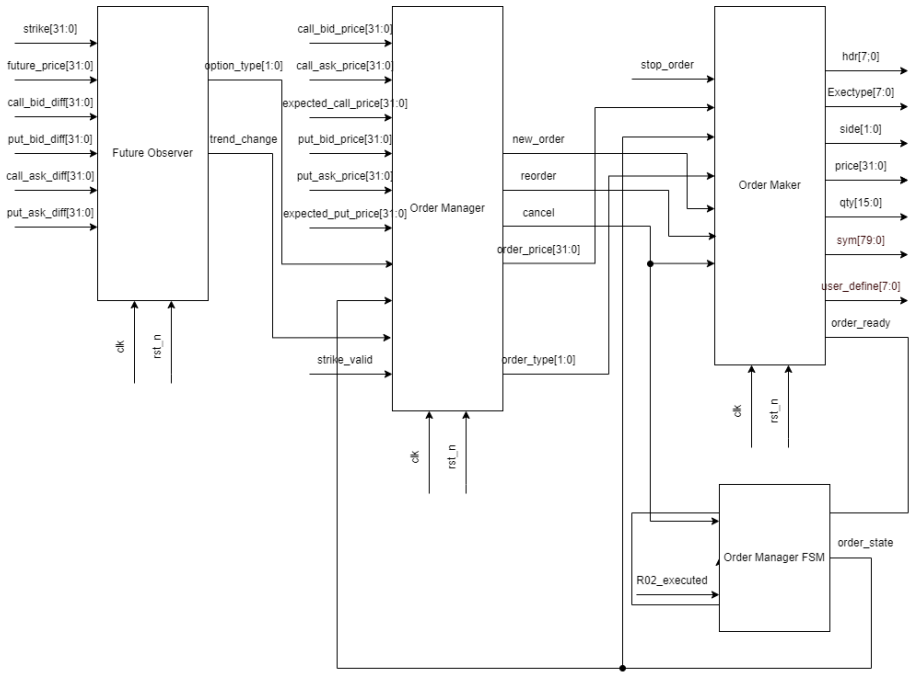

Fig 4. Modules of the Main Order Strategy

The strategy of making the hedge order is developed by the following flow: If an order has been executed, we will have to do the hedging option to make a profit. The option of hedge order will be decided by the opposite of the executed order and related information will be stored in the hedge order book. The price decision will be also done by first starting slightly near the current market price then increased for *bid* or decreased for *ask* if the market price has changed. Once the hedge order is executed, the process of trading is fully completed and thus the order will be removed from the hedge order book. Fig 5. shows the top view of the modules to decide the hedge order. There are more designs under each module but the descriptions of those will not be disclosed in consideration of the classified materials.

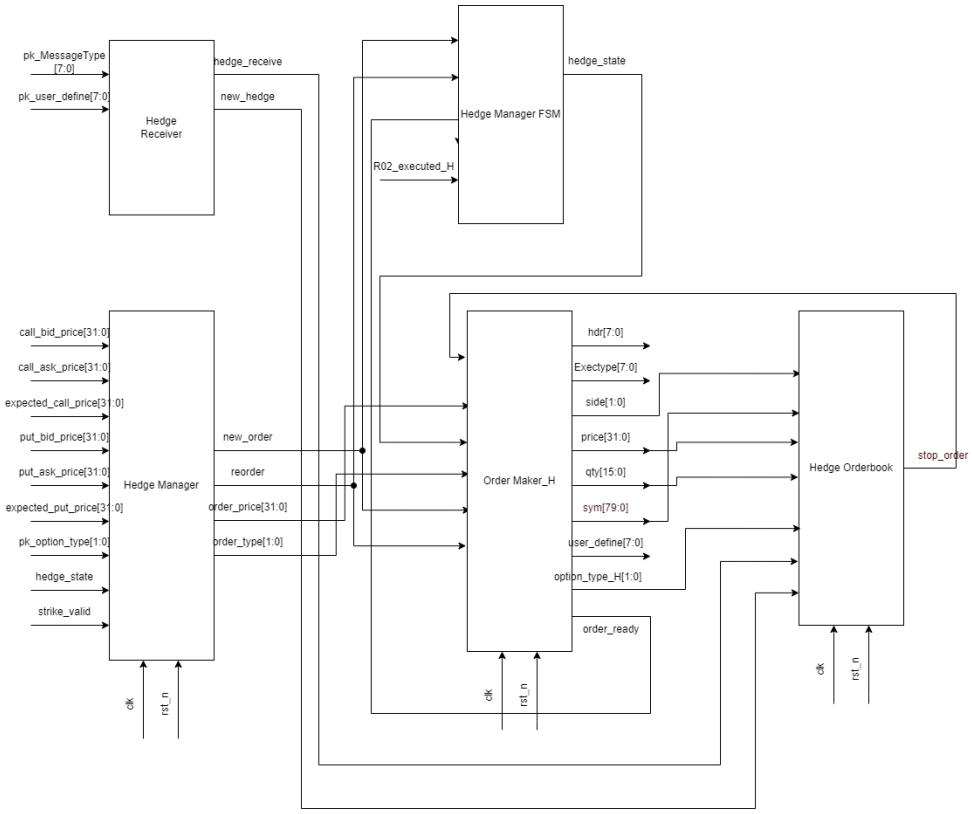

Fig 5. Modules of the Hedge Order Strategy

Results

Owing to the workload division, the results in this section only cover the implementations that include the work from our group.

STRATEGY OF THE ORDER BOOK

Call_valid_ask_1 Call_price_ask_1 Call_qty_ask_1 Synthetic_valid_ask_1 Synthetic_price_ask_1 Synthetic_qty_ask_1
Call_valid_bid_1 Call_price_bid_1 Call_qty_bid_1 Synthetic_valid_bid_1 Synthetic_price_bid_1 Synthetic_qty_bid_1
Put_valid_ask_1 Put_price_ask_1 Put_qty_ask_1
Put_valid_bid_1 Put_price_bid_1 Put_qty_bid_1
Strike

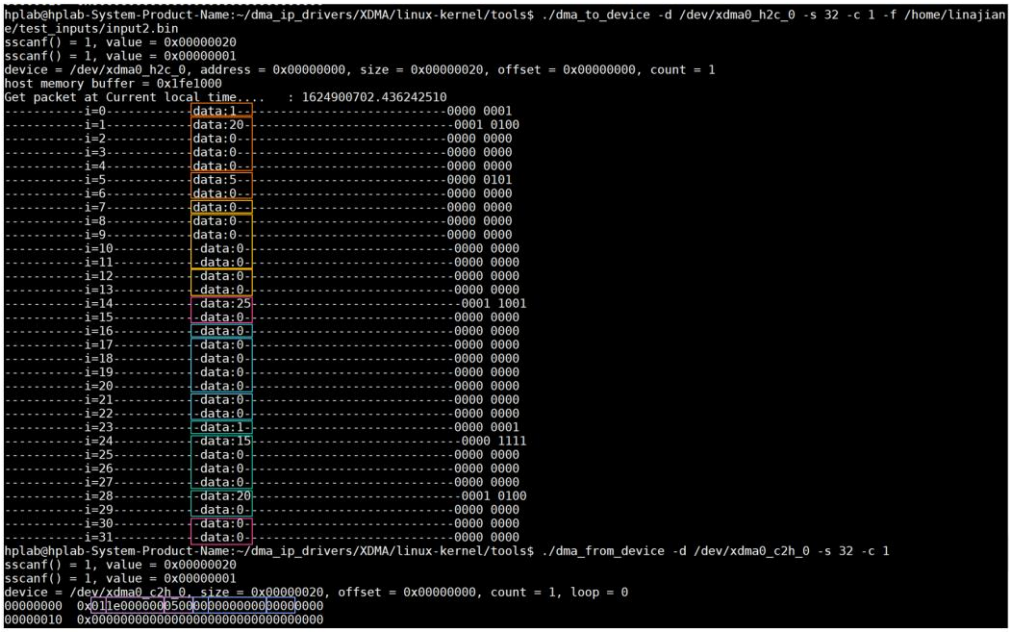

Fig 6. Implementation results of the *Synthetic Option* module

Fig 6. shows the implementation results of the *Synthetic Option* module on the Alveo U50 Data Center Accelerator Card. The inputs and the outputs are boxed in the colors mentioned in Fig R.2.1 and Fig R.2.2 respectively. Since the outputs are in hexadecimal representation and placed in the Little-Endian order, we expect that *Synthetic_price_ask_1* = 30 be represented as 1e000000 and *Synthetic_qty_ask_1* = 5 be represented as 0500. We can see from the above the mentioned outputs are as expected.

STRATEGY OF THE COMPOUND OPTIONS AND FUTURES

Table 1.

Time	Linear (E1)	Weighted Linear (E2)	Quadratic Regression (E3)	TXFG1 price
09:50	17572.87	17573.05	17573	17573
09:55	17559.59	17558.81	17559	17559
10:00	17569.47	17569.92	17570	17570
10:05	17566.82	17566.75	17567	17567
10:10	17571.06	17571.11	17571	17571
10:15	17553.78	17553.9	17554	17554

Table 1. shows the *spotx* obtained by the three different methods discussed. The inputs are real market data on 7/21/2021 from 09:50~10:15. The last column shows the theoretical value of the predicted *spotx.* As we can see, the results obtained from different methods are similar and each with tolerable error.

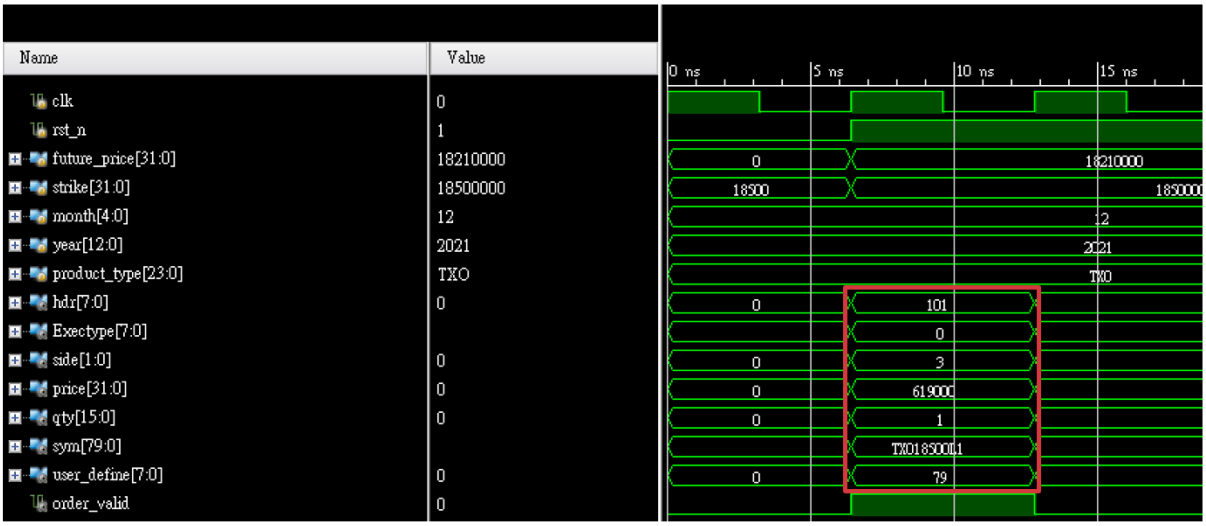

Fig 7. Verilog-HDL Simulation Results of Making the Main Order

Fig 7. shows an example of an order placed. The order shown is a new order with order price = 619000 (619 in real-market), quantity = 1, and product name = TXO18500L1, indicating that this product is a TXO-type “call” product with the strike price being 18500 and placed in December of the year 2021.

Conclusion

We have designed and simulated various HFT-related algorithms in this research. The outputs of our modules indicated that there are no apparent flaws in our current design. This also justifies that we are on the right track. Shortly, we will continue to test our programs and try to optimize them if possible. Additionally, based on the current estimated *Spot* and ordering strategies, we will continue to implement our strategies of hedge orders so that the “Call-Put Parity” principle can be realized and the whole trading becomes more complete and practical for the financial trading market of options and futures.

References

[1] 期交所 TCPIP_TMP_v2.13.0 (Taifex Message Protocol Regulations)

(The references for HFT and its trading strategies are scarcely available as they were mostly developed by private companies.)

Reflections

Before working on this implementation project, we have not ever heard of compound options or the financial terms related to the trading of order book products and futures. Hence, it was quite a big challenge for us to learn about the related financial principles and process of placing orders. From the receipt of market data to placing orders, there are a lot of details and prototypes in between which we need to consider.

It is also our very first time to learn that our knowledge in electrical engineering can be applied to fields like financial technology to solve some computation speed problems in the current financial world. By the process of programming to verify our thoughts and designing hardware logic to realize the trading strategies we desire, we've enhanced our capability in software verification to hardware implementation. We believe that these are essential skills to become an engineer in the field.

Nonetheless, through this implementation project, we got the opportunity to work and discuss with the company VSense Fintech Inc., thus increasing our experiences of communicating with the industry and solving real-life problems. With our implementation project only covering a small portion of the financial products and possible trading policies, there is still quite a long way to go until hardware solutions of each financial trading method are developed and utilized in the real-life markets.