High Frequency Trading Strategies
高頻交易系統策略開發

專題領域：系統組

組　別　：A313

指導教授：馬席彬 教授

組員姓名：陳漢璋

研究期間：2023年 9月初 至 2023年 5月初止，計8個月

報告摘要

高頻交易(High-Frequency Trading, HFT)，是指從那些人們無法利用的、極為短暫的市場變化中尋求獲利的自動交易系統，比如某種證券買入價和賣出價差價的微小變化，或者某隻股票在不同交易所之間的微小價差。高頻交易者以快速反應市場變化，也就是藉由買賣價格的差價來從中獲取利潤 ，因此，延遲時間以及系統吞吐量大幅地影響這類人士的利潤。一般是以電腦買賣盤程式進行非常高速的證券交易，而本次研究是利用現場可程式化邏輯閘陣列(Field Programmable Gate Array, FPGA)加速之高頻交易系統來達到此項目的，並藉由台灣期貨交易過程來賺取差價。

此研究是參考高翊傑學長的論文[1]去實作並試圖加以改良的，而學長研究的基於現場可程式化邏輯閘陣列的高頻交易系統能實現 10G 乙太網路實體收發器、網路分層的封包與解包、部分建立委託簿所必需的台灣期貨市場訊息的編碼與解碼、委託簿的建立，也就是商品五檔價量的管理，以及簡易的高頻交易策略。

乙太網路實體收發器採用 156.25 兆赫時脈(156.25 GHz clock)傳輸 64 位元的資料寬度。接收端能夠辨認並解析位址解析協定、使用者資料報協定與傳輸控制協定的乙太封包，並且提供傳輸控制協定的連線功能。而商品的五檔管理有別於一般是建立每項產品的委託簿，我的研究只支持特定產品的委託簿建立，藉以達到降低交易延遲時間的目的，而高頻交易策略的簡易原因也是如此。

然而，礙於時間不夠充分，目前的成果只到利用C語言來實現網路分層的解包、部分台灣期貨市場訊息的解碼、期貨市場商品五檔價量的管理。最終成果僅到委託簿的建立，尚未完成的包含高頻交易策略的實施、網路分層的封包、市場訊息的解碼，以及最重要的，將C語言轉為硬體語言，並利用FPGA來實施。

若上述提及功能完成了，將使用台灣真實期貨交易環境來驗證系統在市場行情

解析上的正確性以及與期交所連線並策略觸發下單的功能。而整條完整交易路徑的延遲能達到約 500 奈秒，相比於一般高頻交易者使用電腦買賣盤程式進行微秒等級的延遲，效能是百倍有餘。

關鍵字：
高頻交易、硬體加速、場效可程式化邏輯閘陣列、期貨

1. 研究背景

本次研究旨要解決一般所使用基於軟體的高頻交易系統所會發生的問題，例如不可預測的PCI Express(PCIe)傳輸，以及快存遺失等問題。

基於Linux作業系統的典型的軟體高頻交易系統由五個主要基礎架構所組成：網路協議堆疊、金融協議解析、委託簿、客製化交易策略、委託管理系統。而軟體高頻交易系統會在電腦操作網路協議堆疊這個區域時因資料在使用者空間以及核心空間的傳輸而產生大量的延遲。這也是為何本次研究要使用硬體交易系統來達成低延遲的操作。

而軟體交易系統往往伴隨著網路介面控制器(network interface controller, NIC)的使用，當網路介面控制器接收到封包時，它會將封包藉由直接記憶體存取（Direct Memory Access，DMA）來將封包傳送給核心空間。直接記憶體存取會允許外部設備直接進入主記憶體。

第一個議題是當資料傳輸時，記憶體控制中心有可能同時接收到來自中央處理器(Central Processing Unit, CPU)的要求，此時來自中央處理器的記憶體存取速度就會被降低。第二個議題是直接記憶體存取的快取一致性，這可能會導致設備讀取到過時的訊息。

再來的問題是Linux作業系統在處理TCP訊息的時候也有許多性能上的瓶頸，導致整體延遲時間過長，因此，基於硬體的高頻交易系統對於低延遲的要求來說是近乎必須的。

2. 研究方法

附圖是本次研究的主要研究方法流程，其中包括幾個主要的功能：網路層處理、金融協議解碼器、委託簿、客製化交易策略、金融協議編碼器。首先台灣期貨交易所會將交易所的資料藉由使用者資料包協定（User Datagram Protocol, UDP）串流將資料傳給網路層處理，而網路層處理會再將有效負載資料(payload data)傳出送至金融協議解碼器，金融協議解碼器會再將解碼完的資訊，如傳輸群組流水序號、產品編號、產品價錢、產品檔位等資料傳給委託簿，委託簿建立好以後會再將訊息傳給客製化交易策略，決定是否要套利，若要下單則傳訊息給金融協議編碼器，將訊息包裝回Taifex Message Protocol(TMP)電文格式，並回傳給網路層處理為TCP封包，以建立可靠的下單訊息，並回傳給台灣期貨交易所，完成下單。

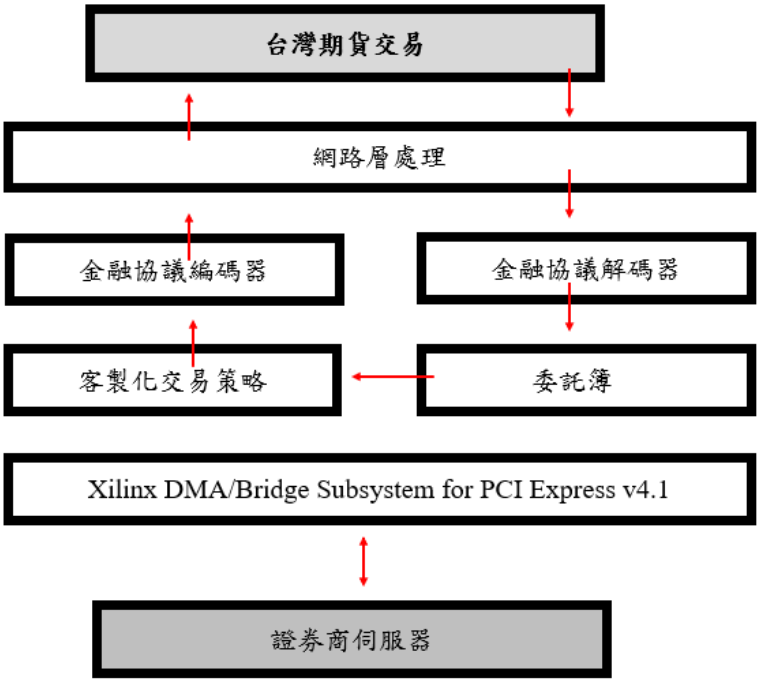

Figure 1: 基於場效可程式化邏輯閘陣列高頻交易系統的架構圖

2.1 網路層處理 (Network Layer)

此部分不是本次研究的重點，因此是全數沿襲[1]中所提到的方法去實作。網路層處理主要可以處理物理層(physical layer)、資料連接層(data-link layer)、網路層(network layer)、傳輸層(transport layer)的資訊。其中包括使用兩個10G Ethernet PCS/PMA v6.0 IP core以及設計32 bits的循環冗餘校正(Cyclic redundancy check, CRC)等功能。最後每個時脈傳輸 64 位元的資料寬度出去給金融交易解碼器，資料內容是以UDP形式傳出，而時脈訊號則是156.25 兆赫的頻率去傳的。

而金融交易編碼器傳進的資料則會設計網路層解碼器來解出TCP的共用檔頭，並搭配其他功能來實行最終的編碼，並將封包編碼回物理層後傳回台灣期貨交易所。

2.2 金融協議解碼器(Financial Protocol Decoder)

資料是藉由乙太網路傳輸，當資料傳進金融協議解碼器時，資料都有共同的IPV4檔頭以及UDP共用檔頭，而這些檔頭的形式都是固定的，或者說位元大小都是固定的數量，因此我可以輕易地用計數器來將個別的資訊都分類出來。

檔頭中唯一重要的資訊只有封包的長度，這個資訊坐落於每個封包的第16-17個位元(從0開始計)，有了長度，那要將每個封包分開是輕而易舉。而接下來則需要解碼的是根據[2]內所規範的逐筆行情資訊傳輸作業手冊的電文格式來進行對照。

去掉檔頭後，剩餘封包的資料內容其實根據[2]的電文規範，每筆資料也會分成行情訊息共用檔頭以及訊息本體兩部分。至於行情共用檔頭因位元大小是固定的，就如同前面處理的方式一樣，可以輕易地利用計數器將所需資訊分類，包括交易時間、訊息種類、傳輸群組流水序號等。檔頭中的資訊也包含了通知我這是何種訊息，因此我再根據電文規範提供的對照表，可以得出這是何種訊息，再將收到每種訊息時該如何處理的方法寫好，大致上的內容也是利用計數器搭配電文規範的格式將每筆重要資料抓出來，其中包括買賣價錢、大小、檔位等，即完成解碼，並將解碼後的數據傳至委託簿。

本次研究應當用硬體語言來完成，而我先用C語言驗證方法可行性以後，再將語言轉換為硬體程式語言。在C語言中我藉由以下幾個函式來完成解碼：

1. int msg_trans(unsigned char trans_code, unsigned char msg_kind)

   這個函式主要是將行情共用檔頭中的TRANSMISSION-CODE, MESSAGE-KIND兩個資料給抓下來，並根據不同組合來一個一個給予分類，而我的研究只需要用到四種訊號：I081, I083, I002, I010，若為其他種類的訊息則捨棄。

2. int* I081_function(unsigned char buffer[])

   I083訊息是委託簿快照訊息，當我經由msg_trans()這個函式分類出訊息為I083訊息時，將剩餘的資料丟進此函式處理。當接收到委託簿快照訊息時，應將委託簿清空並填入新的值，因此會向委託簿傳出一個重置訊號，並輸出商品的買/賣、價格、口數、檔位這幾個必要資訊。

3. int* I083_function(unsigned char buffer[])

   I081訊息是委託簿揭示訊息，當我經由msg_trans()這個函式分類出訊息為I081訊息時，會根據資料內提供的迴圈數量將電文做數次固定格式的解析，並解析出是要新增/修改/刪除何種買賣資訊，並輸出商品的買/賣、價格、口數、檔位這幾個必要資訊給委託簿。

4. int* I002_function(unsigned char buffer[])

   I002訊息是序號重置訊息，當我經由msg_trans()這個函式分類出訊息為I002訊息時，代表期交所系統發生異常，發生同地或異地系統重啟。若該 CHANNEL 屬即時行情群組則須清空各商品委託簿，並重置該傳輸群組之群組序號，同時重置各商品行情訊息流水序號，並等待接收委託簿各項資訊。

5. int* I010_function(unsigned char buffer[])

   I010訊息是商品基本資料訊息，當我經由msg_trans()這個函式分類出訊息為I010訊息時，利用計數器將所需資料抓出，如DECIMAL-LOCATOR這個資料是告知我們該商品的行情價格小數點要點在哪裡。最後將所需資料傳出給委託簿使用。

2.3 委託簿(Order Book)

委託簿的建立仰賴於金融協議解碼器傳輸的資料。在本次研究中C語言的驗證方面是以長度為20的二維陣列來儲存(寬度則是由要支援多少產品的委託簿來決定)，買方價錢和口數、賣方價錢和口數，根據一到五檔依序排列，因此長度總共為20。當然，到時要轉為硬體語言時則會根據電文規範長度來重新定義委託簿的長度，並將位元長度壓至最低。而在我的設計中，委託簿只支援特定產品的建立，以使延遲時間達到最低。

Bid			Ask		
Level	Size	Price	Price	Size	Level
1	1	9830	10100	4	1
2			10150	2	2
3			10200	4	3
4			10300	2	4
5					5

Figure2: 特定一產品的委託簿範例

委託簿的建立則是由clear_orderbook()以及update_orderbook()兩個函式來實現的，前者是在收到高訊號時，輸出一個空的委託簿；後者則是根據委託簿揭示訊息或者委託簿快照訊息解碼出來的資料來更新委託簿。

2.4 客製化交易策略 (Custom Trading Logic)

礙於時間因素，本次研究尚未開發到此部分，日後做到這部分時預計沿用[1]所使用的最簡易交易策略，其演算法如下：

if Best ask price of MTX >4 × Best bid price of TX **then**
Buy 4 lot of MTX with its best ask price and
sell 1 lot of TX with its best bid price
else if Best ask price of TX >4 × Best bid pice of MTX **then**
Buy 1 lot of TX with its best ask price and
sell 1 lot of MTX with its best bid price
end

其中MTX以及TX各代表一個產品，而要實現此演算法，需要將前一級委託簿的產品序號”Product_ID”一個對應至TX，再將另一個對應至MTX。而當此演算法觸發套利條件時，立刻下單並傳送資料給金融協議編碼器編碼成TMP封包，再傳回網路層處理，最後將資料傳回台灣期貨交易，完成下單以，完成獲利。

3. 研究結果

在本次研究中我主要的貢獻為利用軟體驗證金融協議解碼器以及委託簿建立的功能正確性，而經過我用C語言設計的金融協議解碼器以及委託簿確定是可行的，日後在繼續進行此研究時即可將程式碼轉成硬體語言，例如與C語言語法相近的Verilog。我利用我的程式碼讀取台灣期貨交易所一整天的開盤前到盤後的資料來做為驗證用的檔案。首先利用實驗室學長已經設計好的網路層處理(Network Layer)，將台灣期貨交易所傳入的資料轉為UDP封包型式，此時資料是以二進制文件的方式在儲存。

為了方便展示驗證結果，我使用一個名為Wireshark的軟體，此軟體是專門在解析封包的，能將二進制檔案轉為以一個位元一個位元十六進制的方式顯現。而在金融協議解碼器的部分，以委託簿快照訊息以及委託簿揭示訊息對於建立委託簿來說最為重要。

這邊以委託簿快照訊息(I083 message)作為範例：

```
0000  01 00 5e 00 8c 8c 5c 45  27 79 f3 03 08 00 45 00   ··^···\E 'y····E·
0010  00 89 7a b7 40 00 3e 11  5a f9 c0 a8 38 7e e1 00   ··z·@·>· Z···8~··
0020  8c 8c b2 47 36 b0 00 75  31 70 1b 32 42 08 37 20   ···G6··u 1p·2B·7
0030  00 30 00 00 01 00 00 04  71 90 01 00 87 4b 45 46   ·0······ q····KEF
0040  47 32 20 20 20 20 20 20  20 20 20 20 20 20 20 20   G2
0050  20 00 00 00 01 76 31 05  30 30 00 00 00 98 30 00    ····v1· 00····0·
0060  00 00 01 01 31 30 00 00  01 01 00 00 00 00 04 01   ····10·· ········
0070  31 30 00 00 01 01 50 00  00 00 02 02 31 30 00 00   10····P· ····10··
0080  01 02 00 00 00 00 04 03  31 30 00 00 01 03 00 00   ········ 10······
0090  00 00 02 04 9e 0d 0a                               ·······
```

Figure 3：利用Wireshark讀出來的封包(16進位)

```
Data :
01 00 5e 00 8c 8c 5c 45 27 79 f3 03 08 00 45 00
00 89 7a b7 40 00 3e 11 5a f9 c0 a8 38 7e e1 00
8c 8c b2 47 36 b0 00 75 31 70 1b 32 42 08 37 20
00 30 00 00 01 00 00 04 71 90 01 00 87 4b 45 46
47 32 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 00 00 00 01 76 31 05 30 30 00 00 00 98 30 00
00 00 01 01 31 30 00 00 01 01 00 00 00 00 04 01
31 30 00 00 01 01 50 00 00 00 02 02 31 30 00 00
01 02 00 00 00 00 04 03 31 30 00 00 01 03 00 00
00 00 02 04 9e 0d 0a
----------------------------
Packet No.14
Message ID : I083, Information time : 8:37:20.3000
Channel ID = 1   , Channel Seq. = 0000047190
----------------------------
This is an I083 message.
Product ID : KEFG2
Product Message Sequence : 176
Number of MD-ENTRIES = 5
```

```
***** Clear the recent orederbook of the product ******
This is an pre-open remaining message.
------------------------------
No. 1 action :bid
    Price : 09830
    Size  : 1
    Level : 1
No. 2 action :ask
    Price : 010100
    Size  : 4
    Level : 1
No. 3 action :ask
    Price : 010150
    Size  : 2
    Level : 2
No. 4 action :ask
    Price : 010200
    Size  : 4
    Level : 3
No. 5 action :ask
    Price : 010300
    Size  : 2
    Level : 4
------------------------------
```

Figure 4：利用我的程式碼讀出來的資料

藉由Wireshark讀檔結果和我讀出來的資訊一一比對，可以發現封包讀檔內容正確無誤，與此同時，我還能將封包內的重要訊息給抓出來，包括交易時間、傳輸群組序號等，以及最重要的是封包的分類，此時我已分類出此封包為I083類型的訊息，因此做相對應的處理。

當接收到委託簿快照訊息時，首先確認它的種類是試搓剩餘委託訊息，以此個訊息為例，它是試搓後剩餘委託訊息，當收到此封包後將委託簿清空，並根據NO-MD-ENTRIES的數字大小決定要接收幾個迴圈數量的訊息進來，在本次範例中為五個，依序將訊息接收進來後傳至委託簿。第一個動作是新增價格為09830的買入價，口數為一，第一檔；第二個動作是新增價格為010100的賣出價，口數為4，第一檔，以此類推，直到完成此商品的委託簿。而小數點的位置則是根據商品基本資料訊息(I010)來判斷。

```
This is an orderbook of the product ID : KEFG2
                    Bid          |           Ask
-------------------------------------------------------------------
Level   Size         Price        Price        Size         Level
1       00000001     0000098.30   0000101.00   00000004     1
2       00000000     0000098.30   0000101.50   00000002     2
3       00000000     0000098.30   0000102.00   00000004     3
4       00000000     0000098.30   0000103.00   00000002     4
5       00000000     0000098.30   0000000.00   00000000     5
```

Figure 5：接收來自解碼器的訊息完成委託簿的建立

至此驗證完委託簿建立以及金融協議解碼器的功能正確。本次研究尚未完成的部分：在即時交易的狀況中，完成委託簿的建立以後，委託簿會再將訊息傳給客製化交易策略去處理，並完成後續的是否要下單的動作。

4. 總結

委託簿的建立

藉由軟體驗證的方式，利用我的方法來實施委託簿的建立確實是可行的，只需要將軟體語言轉換為硬體語言，並微調部分的輸入以及輸出即可，最後再將讀取記憶體檔案改成直接從台灣期貨交易所進行實時的輸入輸出便是目標。

降低交易時間延遲

本次研究並未完成硬體語言描述的部分，故也沒有成功達成降低交易時間延遲的部分，但委託簿的建立完成後再加上套利演算法離實際進行交易的目標也不遠了。

5. 參考資料

[1] Kao, Yi-Chieh, An FPGA-based High-frequency Trading System on Taiwan Futures Market, 2021.

[2]“Taiwan Futures Exchange (TAIFEX),” https://www.taifex.com.tw/enl/eIndex.

心得感想

本次實作專題總共做了兩個學期，不過對我來說實際上大約只有實作一個多月的時間。身為一個完全沒有相關背景知識的人，我花了一個學期學習關於計算機網路概論的基本知識，並同時實作邏輯設計實驗，並與此同時開始讀前人所留下的論文，就這樣一路到今年三月多為止才大致上弄懂整體架構，並實際看到封包。對於C語言不熟的我又繼續摸索如何開啟檔案以及如何宣告函數等等，在四月底時總算是懂整個實作的架構，並開始付諸實行，開始解封包、抓資料、了解每個種類的訊息功用以及電文格式的規範，最後建立委託簿。其實講實話我認為我的專題進度正在要起飛的過程，我認為再給我一兩個月的時間專心做一定能做完，只能說時間規劃不太成功，在做專題的初期過於怠惰，沒有以高效率的模式補充完先備知識導致最後時間如此倉促甚至沒做完專題。雖然競賽成績在此已經訂下，但我還是會繼續將專題做完，畢竟我也想看到自己實作出一套能對生活有幫助的系統出來，那個成就感是我所追求的。