高頻交易加速器

High-frequency trading accelerator

組別：B157 組員：黃漢霖、卞允辰

指導教授：馬席彬教授

Abstract

本專題題目為高頻交易加速器，研究目的為降低在期貨市場進行高頻交易所遇到的延遲。實作方法為利用 FPGA 進行期貨市場即時行情封包的解碼、運算、以及發送買賣指令，相較於經過電腦軟體運算的程式交易在發出買賣指令前必須經過網路卡、記憶體、CPU、作業系統，直接使用硬體進行解碼並做策略性的運 算會較為快速，使交易者能在瞬息萬變的期貨市場中以最短時間內做出決策，搶占市場先機。

「高頻交易」是指透過自動交易系統利用極為短暫的市場變化中尋求獲利的一種交易方式，雖然每次賺取的報酬極微小，但靠著大量交易累積的利潤也非常驚人。芝加哥聯邦儲備銀行的報告指出，美國股市總體成交量中約有 70%通過「高頻交易」完成，而在市場中進行「高頻交易」的機構數量僅有 2%，因此由此可知隨著科技進步，高頻交易也漸漸成了市場的主流，因此如何在期貨市場中加速資訊的取的、解碼變成高頻交易中一個很重要的課題。

本專題的目的在於如何簡化並加速交易過程，而為了達到研究目的，我們的基本架構為使用 FPGA 板進行即時行情的接收、資料的儲存以及策略的運算，開發方法為使用 C 語言進行程式的撰寫，接著透過 Vivado HLS 將其轉換為 RTL code 使其可以燒錄在 FPGA 板上運作。

Introduction

一、原理分析與系統設計

1.1 原理分析

高頻交易的延遲主要受到網路(network latency)以及軟硬體設備(intenal latency)的速度而定，如果要進行套利，最主要的限制為包含在 internal latency 內的 tick-to-order 時間(註 1)，根據 2019 年的資料，基於軟體開發的高頻交易系統的 tick-to-order 約在 2 微秒，而基於 FPGA 開發的高頻交易系統的 tick-to-order 可以做到 0.3 微秒之內，由此可見基於 FPGA 開發的高頻交易系統較具優勢。

基於軟體開發的高頻交易系統的資料傳遞路徑為網路介面卡接收到來自券商機房的資料封包後，將資料寫入記憶體，之後傳送到作業系統以及交易軟體，再根據交易

軟體內的策略將數據傳送到 CPU 進行運算，運算完後再沿著原路徑返回，透過網路介面卡將資訊以封包的形式送回券商機房下單。而 FPGA 開發的高頻交易系統資料傳遞路徑可以省略掉資料通過作業系統的時間(註 2)，因此本專體選擇以 FPGA 進行高頻交易加速器的開發。

註 1.tick-to-order 時間:從接受到即時行情封包到發出買賣指令的封包所需時間
註 2.以 OSI 七層網路架構來看可以省略掉 session layer、presentation layer、application layer 所需的時間

1.2 系統設計

1.2.1 實作方法

原本規劃的實作方法為透過 C 語言進行委託簿功能的撰寫，主要是針對 I081 的封包進行解析，完成 C 語言的部分後加入優化條件，主要是對 Pipeline 的順序以及變數的位元數進行調整，撰寫完成後透過 Vivado HLS 將其轉換為 RTL code，使其可以燒錄在 Xilinx U50 內進行運算，但經實作嘗試後遇到以下兩點問題:

1. 轉換為 RTL Code 之後可讀性較差，而透過工作站驅動 U50 的 Driver 需要 Fifo_generator 的 IP core 來接收以及回傳資料，因此要接上前後級非常不容易。

2. 同樣功能直接以 Verilog 撰寫僅需要數十行，但先撰寫 C 語言再透過 Vivado HLS 轉換為 RTL Code 後高達上千行，使功能的修改、Testbench 的撰寫、驗證流程都較為困難。

因上述問題，因此最後改以直接撰寫 Verilog 來取代 C 語言轉換為 RTL Code，這樣使得整體的設計流程都較為順利。

主要的程式架構分為四個部分，第一個部分為 Decoder，目的是將接收到的 UDP 封包轉換為 Driver 讀得懂的形式(ASCII)，將封包不重要的部分濾除以及將有 Iteration 的封包拆成不同個數，以保證資料大小的一致性，第二個部分為控制 Driver 的 IP Core(Fifo_generator)，其介面是採用 AXI4-Stream，主要目的是將 Host 傳入的資料傳送至 U50 上，並控制前後級何時開始啟動(註 3)，第三個部分為委託簿，此部分負責解析特定商品 I081 的封包內容並將其依時間順序紀錄，並對委託簿內不同檔位的價格與數量進行修改、新增、刪除，使使用者可以有委託簿作為策略運算的依據，設計時設定每筆傳入 Driver 的資料大小為 32Bytes，其中買賣、檔位、功能以前 12 個 bits 進行控制，而數量及價格分別以 32 個及 40 個 bits 來傳遞。第四個部分與第二個部分一樣，也是使用 Fifo_generator，不同的地方為此部分是將 U50 運算後的資料回傳給使用者。

1.2.2 Decoder

由於是透過 Driver 來驅動 U50，因此封包必須先轉換為 Driver 的資料格式。Driver 再讀取資訊時是以 ASCII Code 的形式讀取(註 4)，因此須先將接收到的封包解碼，留下特定商品的 I081 封包後，再將封包內重要的資訊進行轉換，主要留下買賣、檔位、功能、數量、價格五項資訊進行轉換。

圖(一)第一行為封包內原有資訊，第二行紅框內為轉換後資訊

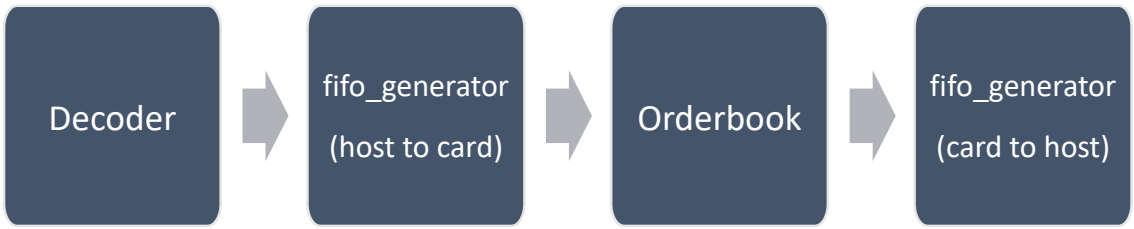
ASCII Code 的前 32 個字元為控制字元，是無法透過鍵盤打出及顯示的，因此數量及價格的部分都是以每一個字元的前四個 bits 來控制，但也因此浪費了不少資料儲存空間。

LSB _ _ _ _ _ _ _ _ - _ _ _ _ _ _ _ _ - _ _ _ _ _ _ _ _ - _ _ _ _ _ _ _ _ .... _ _ _ _ _ _ _ _ MSB 每一個 Input 封包為 256bits

- Input[0]用來控制 Buy/Sell，0 代表 Buy，1 代表 Sell
- Input[3:1] 用來控制 Level，001 代表 Level 1，010 代表 Level 2 依此類推
- Input[11:10] 用來控制功能、00 代表新增、01 代表修改、11 代表刪除
- Input[19:16]、Input[27:24] … Input[75:72]用來代表此封包內商品的量
- Input[83:80]、Input[91:88] … Input[155:152] 用來代表此封包內商品的價格

註 3.透過 Valid-in 及 Valid-out 訊號控制下一級開始運作或是重置資料註 4.假設實際資料為 04 01(hex)，必須將其轉換為字元 A，其 ASCII Code 為 0100 0001

1.2.3 整體架構圖

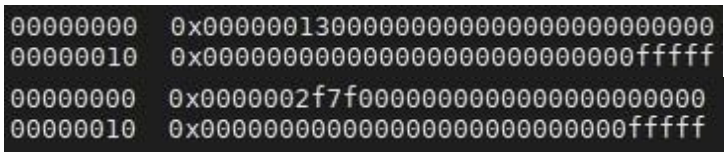


二、實驗結果

2.1 新增功能驗證

2.1.1 第一步

當輸入為 BACA@@@@@@OGOB@@@@@@@@@@@@@@@@@@@@ 時，代表為 Buy 的 Level 1，Size 為 19，Price 為 12159，回傳如下:

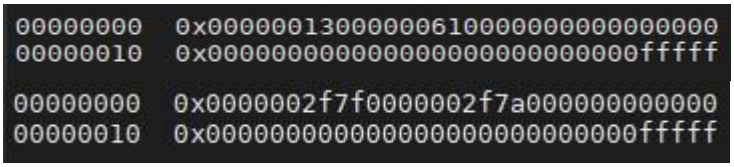


圖(二)新增功能回傳圖

前八個 bits 為 Level 1 的 Size，19 以十六進制表示為 13，12159 以十六進制表示為 2F7F，由此可知第一步正確。

2.1.2 第二步

當輸入為 DAAF@@@@@@JGOB@@@@@@@@@@@@@@@@@@@@時，代表為 Buy 的 Level 2，Size 為 97，Price 為 12154，回傳如下:

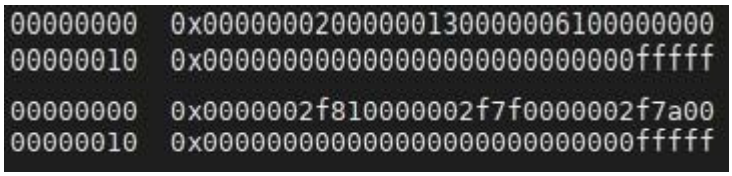


圖(三)新增功能回傳圖

Level 2 的 Size 97 以十六進制表示為 61，Price 12154 以十六進制表示為 2F7A，由此可知第二步正確。

2.1.3 第三步

當輸入為 BAB@@@@@@@AHOB@@@@@@@@@@@@@@@@@@@@ 時，代表為 Buy 的 Level 1，Size 為 2，Price 為 12161，回傳如下:



圖(四)新增功能回傳圖

新增於 Level 1 的 Size 2 以及 Price 12161 使得原本在 Level 1 及 Level 2 的 Size 與 Price 皆

向下移動一個檔位，也就是說對於此商品出價高的排在優先成交的位置，由此可知新增的功能正確。

2.2 刪除功能驗證

當輸入為 DD@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@時，代表為刪除 Buy 的 Level 2，回傳如下:

```
00000000  0x00000002000000610000000000000000
00000010  0x000000000000000000000000000fffff
```

```
00000000  0x0000002f810000002f7a000000000000
00000010  0x000000000000000000000000000fffff
```

圖(五)刪除功能回傳圖

刪除 Buy 的 Level 2 使得 Level 3 的 Size 與 Price 都往上移了一個檔位，代表出價高者取消了該委託單，因此出價較低者成交順位向前一個檔位，由此可知刪除的功能正確。

2.3 修改功能驗證

當輸入為 DLAE@@@@@@KGOB@@@@@@@@@@@@@@@@@@時，代表為修改 Buy 的 Level 2 的 Price 與 Size，將 Price 修改為 12155，將 Size 修改為 81，回傳如下:

```
00000000  0x00000002000000510000000000000000
00000010  0x000000000000000000000000000fffff
```

```
00000000  0x0000002f810000002f7b000000000000
00000010  0x000000000000000000000000000fffff
```

圖(六)修改功能回傳圖

此封包將原本 Level 2 的 Size 從 97 修改為 81，以十六進制表示為 51，也將 Level 2 的 Price 從 12154 修改為 12155，以十六進制表示為 2F7B，由此可知修改功能正確。

2.4 結果簡述

專題研究結果為設計好基本的委託簿架構，基本的功能(新增、修改、刪除) 上都能順利完成，另外還設計了一個適當的編碼方式以及編碼/解碼器可於使用此 Driver 情形下供此委託簿使用，唯一較不方便的地方為此 Driver 回傳一次的

Bandwidth 僅有 256 bits，因此如果依照原本設計，Size 與 Price 無法同時出現，因此未來也許可以將 Price 與 Size 的資料大小縮小，不僅可以節省處理的時間，也省下許多暫存的資源。

心得

卞允辰：

經過了這次的專題，我學會了如何實際運用課程上所學的知識。結合計網概以及程式設計等課程來實現這個專題。另外也在過程中學習了許多新的知識，像是學習使用 HLS 等以前沒學過的東西。我從這次的專題之中學到了很多，非常感謝教授以及學長的教導，協助解決我們遇到的種種問題，最後感謝我的組員黃漢霖的合作與幫忙，使得我們能順利完成。

黃漢霖：

這九個月以來的實作專題，除了讓我對高頻交易、HLS 開發流程、封包的傳遞原理以及 Driver 的運用有更多的認識，最重要的是學習到如何從零開始學習並開發一項專案。在一開始，花了蠻多時間在學習基本的背景知識以及工具的使用，尤其是在 C 語言轉換為 RTL Code 的時候以及了解 Driver 的運用，但最終還是克服了困難。

我特別要感謝的人是馬席彬教授，除了耐心並詳細地指導我們研究進行的方向外，還要容忍我與組員常常遇到 Bug 而導致進度延誤，除了教授以外，也很感謝趙德昊學長不厭其煩的解決我們的問題，常常讓我們問到半夜一兩點，最後也要感謝我的組員卞允辰，這一年來的相互合作以及彼此互相學習使得我能完成實作專題。

Reference

[1] Christian Leber, Benjamin Geib, Heiner Litz, “High Frequency Trading Acceleration using FPGAs”, University of Heidelberg, Germany, September 2011

[2] Andrew Boutros, Brett Grady, Mustafa Abbas, P. Chow , “Build fast, trade fast: FPGA-based high-frequency trading using high-level synthesis”, 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)

[3] John W. Lockwood, Adwait Gupte, Nishit Mehta, Michaela Blott, Tom English, Kees Vissers, “A Low-Latency Library in FPGA Hardware for HighFrequency Trading (HFT)”, 2012 IEEE 20th Annual Symposium on HighPerformance Interconnects