國立清華大學 電機工程學系

實作專題研究摘要

Hardware implementation of high frequency trading

高頻交易的硬體實作

專題領域：系統領域

組　　別：B491

指導教授：翁詠祿 教授

組員姓名：彭俊翔

研究期間：2024年 2月 19日至 2024年 11月 25日止，共8個月

摘要

本專題旨在運用 FPGA 硬體加速，實現高頻交易所需的低延遲和高性能，以克服傳統軟體方案在延遲上的不足。透過使用 Vivado HLS，開發者能快速將 C/C++撰寫的交易策略轉換為硬體描述語言(HDL)，縮短開發週期並優化設計效率，通過比較不同方法，比較 FPGA 的資源利用率與性能表現。此外，也實作台灣期貨交易所 TMP decoder/encoder 硬體的基本功能。

1. 研究背景

高頻交易是一種自動化交易策略，主要依賴高性能硬體、低延遲的網路，以及演算法，以抓住市場上的微小價格變動進行盈利。其主要有以下特色：極快的交易速度、每筆訂單獲利少，交易量大。高頻交易對速度的需求使得傳統的軟體解決方案不足以滿足其要求。FPGA 的超低延遲、可重複程式化，成為高頻交易加速器的首選。

考量到硬體語言的開發時間通常比軟體開發時間長，硬體開發可能跟不上交易演算法的變化速度，因此嘗試使用更有效率的做法: HLS 來降低硬體開發週期。

Vivado HLS（High-Level Synthesis）是由 Xilinx 提供的工具，專門用於將高階語言（如 C、C++、SystemC）編寫的演算法轉換為硬體語言（HDL），如: Verilog。它的主要目的是讓開發者可以更有效率地開發 FPGA，相較於直接使用 RTL 語言，簡化開發過程，提升效率。

2. 研究目的

本專題將使用 Vivado HLS 將交易策略從 C 語言轉換成 RTL code，以及 TMP (Taifex Message Protocol) decoder/encoder 實作。

3. 研究方法

3.1 價差套利

3.1.1 交易策略原理

利用期貨大小台價差大於某個特定的值，買賣特定數量的口數以獲利。以台指期為例，大台指1點=200元，小台指1點=50元，買賣量大台指:小台指=1:4，大台指1口手續費50元，小台指1口手續費25元，期貨交易稅為總價的十萬分之二。假設大台指24000點，一口大台+四口小台的成本:

$$成本 = 24{,}000 \times 0.00002 \times 2 \times 200(稅金) + 50 + 25 \times 4 = 342元$$

所以大小台點數相差兩點以上(> 400元)可以獲利。

同商品流水號	商品代號	資料時間	第1買價	第1買量	第1賣價	第1賣量
361	TXF 2024y7m	08:45:00.058.000	2409000	3	2409100	5
361	MXF 2024y7m	08:45:00.058.000	2407900	1	2408100	6

TXF 第一買價為24090點，MXF 第一賣價為24081點，買 TXF 1口，賣 MXF 4口，即可獲利 (上表買賣價為點數*100)

在 udp_fut_orderbook.csv 這筆資料中，台指期共獲利103761.108元，電子期共獲利2260.060元，金融期則是沒有獲利。

3.1.2 撰寫 C++ code

input data 包含 : 商品流水號、product_code(商品名稱)，如”TXF 2024y 7m”表示大台指2024年7月、時間、第一買價、第一買量、第一賣價、第一賣量。
資料包含 : 大小台指期 TXF MXF 2024年7、8、9、12月、2025年3、6月，大小電子期 EXF ZEF 2024年7、8、9、12、2025年3、6月，大小金融期 FXF ZFF 2024年7、8、9、12月、2025年3、6月。

方法一 : 直接對比商品名稱進行套利運算
將所有商品種類使用 if-else statement 全部列舉出，逐一比對商品名稱，如果商品名稱對應到，就進行相對應的套利判斷和計算，套利計算的部分是獨立存在於每個分支下。
方法二 : 將方法一的程式分切成副程式
先辨別出商品種類，再分別取得相對應的資訊，如:稅率、手續費等。再利用上述資訊進行套利運算。這個方法只需要用到一個套利計算模塊。
需要注意的是，Vivado 不支援有 dynamic memory allocation 功能的 function，如 C++ STL vector、string 等，以及 C 字串等操作。

3.1.3 C simulation

C simulation 確保 C++ code 符合給定的 testbench，testbench 的條件:如果獲利結果跟預期結果相減的絕對值> 0.001，即判定不通過。

3.2 TMP 實作

3.2.1 TMP

Taifex Message Protocol 是台灣期貨交易所（TAIFEX，Taiwan Futures Exchange）為電子交易系統所制定的通訊協議。下表是 Message Type 以及對應的 Input/Output。
其中，L type message 用於啟動/關閉子系統，送出後須等待期交所回覆，再送下一筆，R type message 是在連線至子系統後，開始收發下單、連線確認等相關訊息，可以同時收發，而不需等待期交所回應。

Message Type	Direction	Description
L10	In/Out	Wake up message
L20	Out	Wake up confirm message
L30	In	Login notice message
L40	Out	Login request
L50	In	Activate application
L60	Out	Activate application confirm
L70	Out	Deactivate application
L80	In	Deactivate application confirm
R01	Out	Order placement
R02	In	Order status report
R03	In	Error report
R04	In	Confirm connection message
R05	Out	Connection confirmed message

3.2.2 FSM 設計

一開始，state = IDLE，等待 en 由0->1時，state 變為 L10，TMP 送出 L10 message 後，state 變為 L10r，等待期交所回覆 L10 message，待收到期交所 L10 message 後，state 變為 L20並送出 L20 message，state 變為 L30等待接收 L30，一直到 TMP 送出 L60後，state 變為 ON，此時子系統連線成功，若 off 由0->1，則送出結束訊息 L70，等待期交所回復 L80後，state 最終回到 IDLE。
在子系統連線成功的狀態下，開始收發 R01~05，一開始 msg_state = M_IDLE，不送出任何訊息，若 TMP 收到由期交所送來的 R04，則 TMP 須回覆 R05，待傳送完後回到 M_IDLE，若收到 order，則 TMP 須送出訂單訊息 R01，待傳送完後回到 M_IDLE，若 TMP 正在傳送 R01時收到由期交所送來的 R04時，msg_state 會變為 R01_R05，會繼續送完 R01，然後 msg_state 會變為 R05，傳送 R05，傳送完回到 M_IDLE。

3.2.3 I/O

input clk
input rst_n
input off
input en

input [31 : 0] order_rdata
input order_rvalid
input [3 : 0] order_rkeep
input order_rlast
output order_rbusy

接收從交易模塊所送出的訂單訊息，1 cycle 傳送32 bits data，當有訂單時 valid : 0 -> 1，當傳送一組資料最後32bits 時，last : 0 -> 1，keep 是指保留 data[31:0]的位置，如當 keep = 4'b1000，則保留 data[31:24]。送出 R01訂單時 busy : 0 -> 1。

input [31 : 0] tmp_rdata
input tmp_rvalid
input [3 : 0] tmp_rkeep
input tmp_rlast
input tmp_rbusy

接收從 TCP/IP decoder 的訊息，1 cycle 傳送32 bits data，當有訊息時 valid : 0 -> 1，當接收一組資料最後32bits 時，last : 0 -> 1，keep 是指保留 data[31:0]的位置，如當 keep = 4'b1000，則保留 data[31:24]。busy 代表 TCP/IP encoder 目前無法接收 TMP output

output [31 : 0] tmp_tdata
output tmp_tvalid
output [3 : 0] tmp_tkeep
output tmp_tlast

送出訊息至 TCP/IP encoder，1 cycle 傳送32 bits data，有訊息時 valid : 0 -> 1，當傳送一組資料最後32bits 時，last : 0 -> 1，keep 是指保留 data[31:0]的位置，如當 keep = 4'b1000，則保留 data[31:24]。

4. 研究結果

4.1 價差套利

方法一 ：

HLS 合成結果

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	1695	10.000 ns	16.950 us	1	1695	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	11359	-
FIFO	-	-	-	-	-
Instance	-	14	1450	2692	-
Memory	72	-	1152	180	0
Multiplexer	-	-	-	39611	-
Register	-	-	46810	-	-
Total	72	14	49412	53842	0
Available	730	740	269200	134600	0
Utilization (%)	9	1	18	40	0

方法二 ：

HLS 合成結果

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	146	10.000 ns	1.460 us	1	146	none

Detail

Instance

Loop

Utilization Estimates

Summary

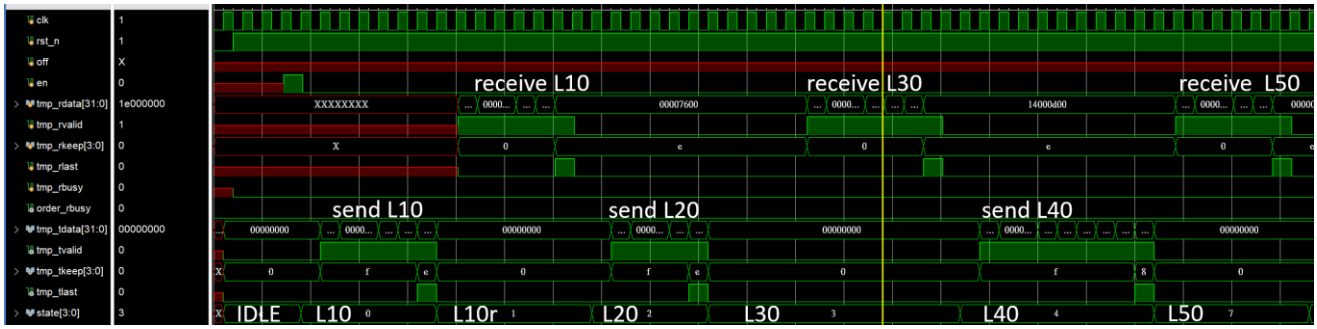
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	4	0	1304	-
FIFO	-	-	-	-	-
Instance	2	14	2904	4256	0
Memory	10	-	0	0	0
Multiplexer	-	-	-	1065	-
Register	-	-	1183	-	-
Total	12	18	4087	6625	0
Available	730	740	269200	134600	0
Utilization (%)	1	2	1	4	0

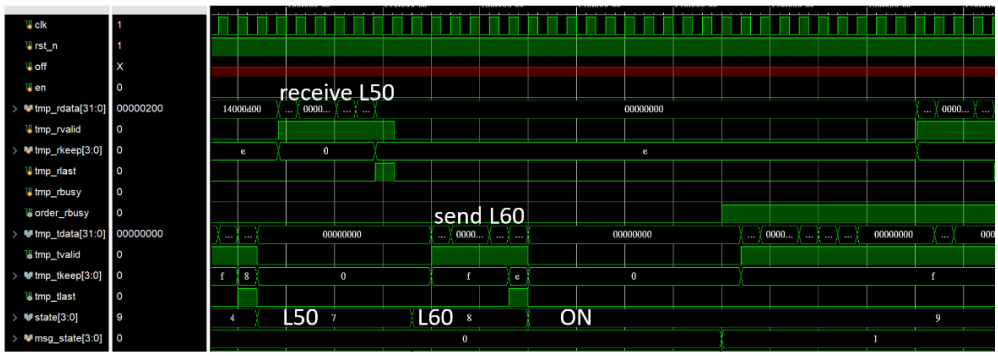
比較方法一及方法二 Utilization Estimates 的部分，可以發現 BRAM、FF、LUT 的使用率，方法二的結果都比較低，推測方法一的寫法沒有進行分割，所以在合成時，HLS 會將每一種可能性全部合成出來，方法二的寫法有將一部分分割出去，在合成時有進行 resource sharing，所以整體資源使用率較低。從 Co-simulation 的結果來看，方法二的平均延遲也較低。

4.2 TMP

Simulation Result

從 state = IDLE 到完成啟動子系統





5. 總結

5.1

HLS 相較直接撰寫 RTL，的確可以縮短交易策略的開發週期。只要有 C++ code，HLS 就可以將其轉為 RTL，但相對也有限制，如 HLS 不支援 C/C++的部分功能，對於不同 C/C++的 coding style，對於 HLS 合成出來的 RTL，其性能(如:資源占用率、延遲)影響也非常大。另外，HLS 會自動進行 pipeline 設計，對於延遲有高要求的情況下，直接撰寫 RTL，自己設計 pipeline 的分割點，對延遲有更多控制權，可能是比較好的方法。

5.2

實做 TMP 的基礎功能:期交所子系統啟動、關機，以及下單、連線確認功能。

6. 心得

透過這次的專題實作，對於高頻交易有更深入的認識，包含網路層、交易訊息協定、交易策略等等。自己也重頭開始摸索 Vivado HLS 的使用方式，並使用 HLS 合成出 RTL，有別於一般的 RTL programing。此外，實作 TMP decoder/encoder，了解到期貨商與期交所的 financial protocol，以及委託單方面的知識。感謝教授、學長的指導，也感謝學長提供資料。