Insper

Curso Engenharia de Computação

Relatório Final de Iniciação Tecnológica

Eiki Luis Yamashiro

Orientador(a): Prof Rafael Corsi

Aceleração de Algoritmos de High-Frequency Trading com FPGA

São Paulo

2022

Resumo

A bolsa de valores oferece diversos benefícios para investidores, com a renda proveniente de um certo nível de risco, de forma a promover o aquecimento de toda cadeia produtiva e econômica dos países. O High-Frequency Trading ou operações em alta frequência permite a análise de uma grande quantidade de informação, à uma alta cadência de transferência de dados. Entre o tempo da requisição até a recepção da ordem, o preço pode variar, ou seja, o valor esperado, pode ser diferente do valor real. Mediante a essa situação, um método para acelerar algoritmos através de FPGAs é proposto, com o objetivo de diminuir o tempo entre a requisição e a recepção. O foco da aceleração está na decodificação do protocolo FIX através da placa FPGA.

Palavras-chave: Computação Embarcada, High-Frequency Trading, Field Programmable Gate Array.

Abstract

The stock exchange offers several benefits to investors, the income comes from a certain level of risk, in order to promote the heating of the entire productive and economic chain of the countries. High-Frequency Trading allows the analysis of a large amount of information and data at a high rate of data transfer. Between the time from the request to the receipt of the order, the price may vary, the expected value may differ from the actual value. Given this situation, a method to accelerate algorithms through FPGAs is proposed, with the objective of reducing the time between request and reception. The focus of acceleration is on decoding the FIX protocol through the FPGA board.

Keywords: Embedded Computer, High-Frequency Trading, Field Programmable Gate Array.

Sumário

1 INTRODUÇÃO 6

2 PROPOSTA 7

3 BANDAS DE BOLLINGER 7

PROTOCOLO FIX 8

4.1 Estrutura da Mensagem 8

4.2 Tipos de Dados 8

4.3 Remoção da Redundância de Dados 9

4.4 Topologia do UDP Datagram 9

5 COMUNICAÇÃO SOCKET 10

6 AMBIENTE DE DESENVOLVIMENTO 11

7 RESULTADOS 13

8 PRÓXIMOS PASSOS 15

REFERÊNCIAS 17

1 Introdução

Desde seu surgimento, a bolsa de valores oferece diversos benefícios, ara investidores, com a renda proveniente de um certo nível de risco e a abertura para realização de investimentos, e para empresas, possibilitando arrecadação de fundos, além de bancos ou financeiras. Dessa forma, a bolsa e valores promove o aquecimento de toda cadeia produtiva e econômica dos países.

Segundo Peter Gomber [1], a dinâmica competitiva do mercado transformou o panorama de processos de compra e venda de ativos, o que resultou em uma nova maneira de execução de conceitos e de infraestrutura acerca da segurança das operações. O processo inteiramente manual, exposto a erros humanos, foi se automatizando e a utilização de ferramentas da computação tornou-se cada vez mais recorrente. Dessa forma, o mercado iniciou sua modernização a partir do uso de Electronic communication networks (ECNs).

Atualmente, o mercado financeiro movimenta bilhões de dólares diariamente, e cada vez mais, se torna acessível para novos investidores. Assim, a importância e quantidade de análises demandadas pelo mercado aumentou. O High-Frequency Trading ou operações em alta frequência permite a análise de uma grande quantidade de informação, de forma a otimizar o desempenho dos modelos, priorizando as tomadas de decisões eficientes, que são essenciais para a obtenção do lucro.

As ordens enviadas para a bolsa de valores são computadas em ordem de chegada. Portanto, além da eficiência, o tempo também é considerado no momento de mensurar o desempenho, cada milissegundo é importante e impacta diretamente no lucro sobre os outros investidores do mercado. Assim, os feeds de dados de mercado destinados às operações de baixa latência são disponíveis via UDP, permitindo que uma única fonte seja compartilhada para vários usuários [2].

Entre o tempo da requisição até a recepção da ordem, o preço pode variar, ou seja, o valor esperado, pode ser diferente do valor real, visto que cada operação possui o potencial de alterar o preço de compra ou venda. Mediante a essa situação, um método para acelerar os algoritmos através de FPGAs é proposto por Ricardo Sabedra [4] para aumentar a velocidade que a ordem é feita, pois quanto menor o tempo entre a requisição e a recepção, menor é a chance de ocorrer uma grande variação entre o valor real e o esperado.

Segundo Leber [3] uma decodificação eficiente do fluxo de dados é essencial, assim a decisão de emitir o pedido deve ser realizada com a menor latência possível. Para atingir esses objetivos, Leber apresentou um mecanismo implementado em FPGA, para acelerar as negociações. Assim, as tarefas de decodificação são descarregadas para blocos de um hardware otimizado. Esse sistema proposto implementa uma pilha de processamentos completa.

2 Proposta Atual

O atual projeto tem como objetivo acelerar o algoritmo das Bandas de Bollinger através de uma implementação em Python e C, com diferentes hardwares e sistemas operacionais. Assim, será possível obter resultados do desempenho de uma Raspberry Pi 3, Laptop com i7 7th gen e FPGA com um Kernel de Linux.

3 Bandas de Bollinger

Aldrige **[6]** apresenta um algoritmo clássico para o *High-Frequency Trading*, conhecido como *Bandas de Bollinger*, esse modelo indica valores máximos e mínimos dos preços de uma posição específica, para qualquer ponto no tempo. Assim, o cálculo fornece duas bandas, a superior e a inferior, que determinam uma área onde, com base no histórico do ativo, é possível identificar se o preço sofrerá uma valorização ou uma queda. Além das bandas é possível determinar a média móvel dos valores de uma ação.

A Figura 2 ilustra o funcionamento do algoritmo. Se a variação do preço corrente (linha vermelha) cruzar a borda superior, de baixo para cima, é enviada uma ordem de venda, caso contrário (variação do preço cruza a banda inferior) é enviada uma ordem de compra.

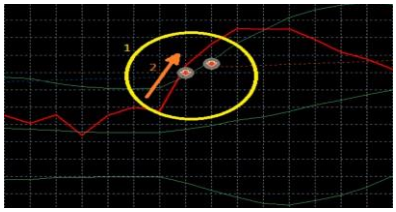


Figura 1 - **[3]** Borda de subida - Bandas de Bollinger

As Equações da média móvel $(\overline{X})$ e do desvio padrão (σ) a seguir, são utilizadas para o cálculo das Bandas de Bollinger:

$$\overline{X} = \frac{\sum_{j=1}^{N} X}{N}$$

Equação 1 – Cálculo da média móvel $(\overline{X})$

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(X_j - \overline{X})^2}{N}}$$

Equação 2 – Cálculo do desvio padrão (σ)

A partir do cálculo do desvio padrão e da média móvel, pode-se determinar as bandas superior, central e inferior:

$$Banda\ superior = \overline{X} + 2\sigma$$

Equação 3 – Cálculo da Banda Superior

$$\underline{Banda\ central = \overline{X}}$$

Equação 4 – Cálculo da Banda Central

$$Banda\ inferior = \overline{X} - 2\sigma$$

Equação 5 – Cálculo da Banda Inferior

Para permitir a comparação da variação do preço corrente, o armazenamento desses valores é feito em filas do tipo *FIFO* (*First in, First out*).

Segundo Babbar **[8]**, o algoritmo das Bandas de Bollinger apresenta uma alta precisão na tomada de decisões de comprar ou vender ações. Assim, esse será o algoritmo escolhido para ser implementado na *FPGA*.

4 Protocolo FIX

O *FIX Market Data Optimization Working Group* desenvolveu o protocolo *FIX Adapted for Streaming* (FAST) para otimizar, em termos eletrônicos, o tráfego de dados financeiros, particularmente para grandes volumes e baixas latências.

O FAST é um algoritmo de compressão de dados, que reduz a largura de banda, que por consequência minimiza a latência entre o envio e o recebimento do pacote. O FAST é uma adaptação do protocolo FIX e utiliza uma formatos e estruturas de dados semelhantes, com o diferencial de remover redundâncias de dados, através de métodos como o Lempel-Ziv.

4.1 Estrutura da Mensagem

Podemos de forma simplificada, compreender a estrutura de uma mensagem FIX da seguinte maneira:

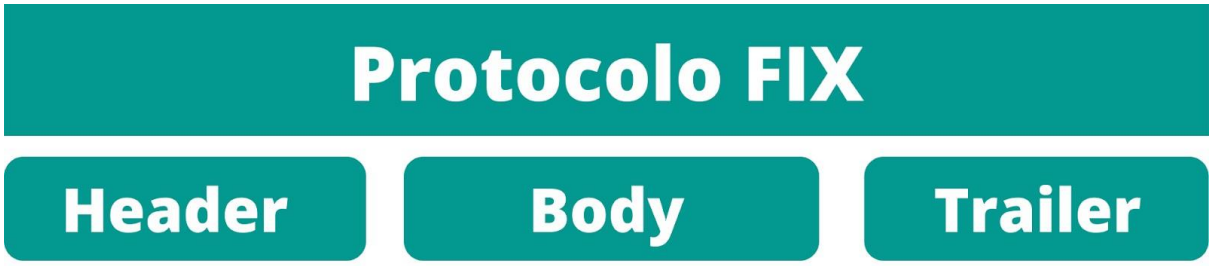


Diagrama 1 – Estrutura do Protocolo FIX

4.2 Tipos de Dados

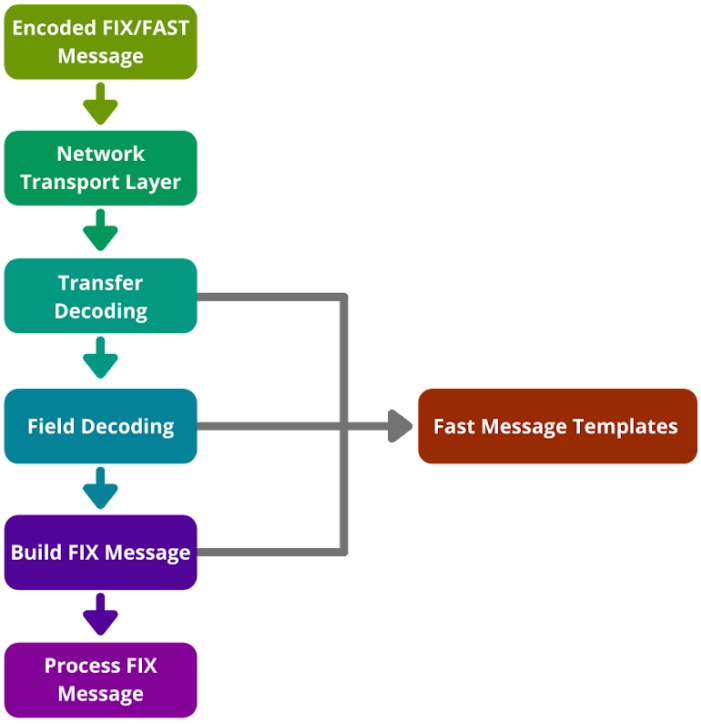
Abaixo é apresentado todos os tipos de dados reconhecidos pelo FAST:

- ASCII (7-bit) strings
- Unicode Strings (UTF-8)
- Byte Vectors
- Decimal Numbers
- Signed Integers (32 e 64 bits)
- Unsigned Integers (32 e 64 bits)

O protocolo FIX possui mais tipos de dados, porém a maioria pode ser mapeada com os dados do FAST. Exemplo: Price -> Decimal, TimeStamp -> Unsigned Integer. O formato do UTCTimeStamp é YYYYMMDD-HH:MM:SS:ssss codificado para um inteiro como YYYYMMDDHHMMSSssss.

4.3 Remoção da Redundância de Dados

A redundância de dados é removida, através da seguinte ideia: Se há um template, nenhuma outra informação precisa ser mandada (como tags e field separators). Algumas partes do package é constante ou possui valores default que podem ser removidos ou omitidos.



3.4 Decodificação do FAST

i. O Encoder FAST traduz a mensagem FIX para uma FAST.

ii. A mensagem é transmitida (UDP / Datagram) e recebida pelo sistema do cliente.

iii. Transfer Decoding: Identifica o template, extrai o código binário e mapeia os bits para o template.

iv. Field Decoding: Operações <copy> / <delta> para determinados os valores em cada campo do template.

v. Construção da mensagem FIX processada. Diagrama 2 – Decodificação do FAST

4.4 Topologia do UDP Datagram

MsgSeqNum	NoChunks	CurrentChunk	MsgLength	FAST message
4 bytes	2 bytes	2 bytes	2 bytes	*MsgLength* bytes

Tabela 1 – Header UMDF

Todos os atributos definidos no header seguem a convenção big-endian.

MsgSeqNum - Este atributo contém o mesmo valor do Tag 34-MsgSeqNum.

NoChunks - Número total de blocos que constituem uma única Mensagem FIX / FAST identificada por MsgSeqNum no canal na sessão de negociação atual.

CurrentChunk - A posição atual do bloco de dados que constitui uma única Mensagem FIX / FAST identificada por MsgSeqNum no canal na sessão de negociação atual.

MsgLength - O comprimento da seguinte sequência de bytes que constitui um bloco de dados.

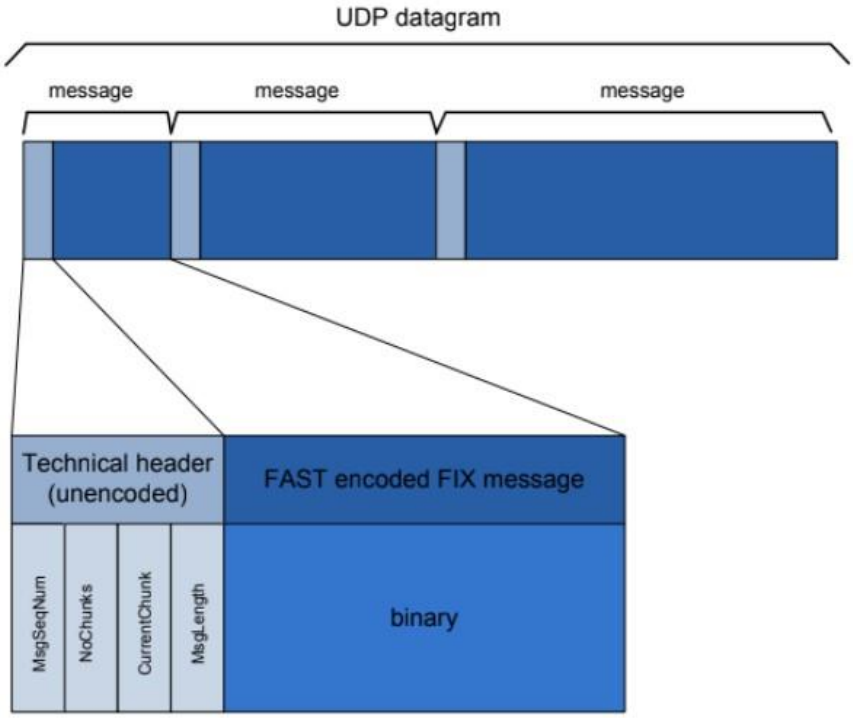


Diagrama 3 – Mensagem Técnica do Header

5 Comunicação Socket

A comunicação em rede no ambiente é feita através de conceitos de sockets. Um socket é uma interface de comunicação bidirecional entre processos. Sockets são representados como descritores de arquivos e permitem a comunicação entre processos distintos na mesma máquina ou em diferentes máquinas, através de uma rede.

A BVMF se comunica com a aplicação de um sistema cliente através do modelo cliente/servidor. Que possui a seguinte característica:

- O cliente conhece o endereço e forma o acesso ao servidor e toma a iniciativa da comunicação.
- O servidor é uma entidade passiva, recebe apenas pedidos dos clientes e responde em tempo real.
- O cliente envia uma requisição de serviço e aguarda uma resposta do servidor.
- As implementações do cliente e do servidor são independentes e autônomas; apenas as sequências de mensagens trocadas durante a comunicação, que caracterizam o serviço, devem ser respeitadas.

5.1 Cliente

O cliente cria um socket e o conecta ao endereço do servidor. O envio e recebimento de dados é feito através do socket com as funções de **write** e **read**. A comunicação pode ser encerrada com a função **close**.

5.2 Servidor

Para se criar um server, é necessário criar um socket e associar um endereço através de uma função de **bind**. O envio e recebimento de dados é feito através de **write** e **read**.

6. Ambiente de Desenvolvimento

Os códigos desenvolvidos estão disponíveis em:

- https://github.com/EikiYamashiro/hft_fpga_acceleration
- https://github.com/EikiYamashiro/bollinger_bands_animation
- https://github.com/EikiYamashiro/bollinger_bands

Para o desenvolvimento do projeto, com a visão de fidelizar o modelo ao mercado financeiro, será utilizado a criação de um ambiente que simule a comunicação da bolsa de valores através de sockets. A BVMF será o servidor, que por sua vez, ao ser criado, deve estar preparado para receber a qualquer momento um convite (handshake) da aplicação cliente para o início da troca de dados. Atualmente, não há um protocolo específico (header + payload + eop) e os dados são mandados a uma frequência de 1 Hz.

O cliente possui o papel de simular um algoritmo de High-Frequency-Trading, dessa forma recebe os dados do servidor, efetua o cálculo para o algoritmo determinado e envia a ordem de buy, hold ou sell, ao depender dos resultados dos cálculos. O servidor deve enviar de forma constante os pacotes com intervalos iguais e com a mínima variação possível, para simular de forma fiel a BVMF. Ao receber a ordem do cliente, efetua as ações de compra/venda se necessário e retorna uma

resposta confirmando que a ordem foi efetuada. O diagrama 4 ilustra uma comunicação entre o server e o cliente.

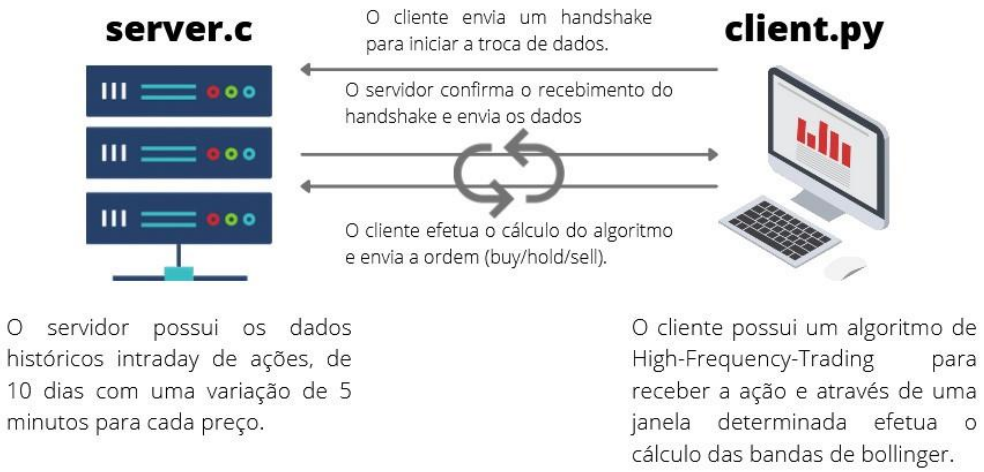


Diagrama 4 – Comunicação Server/Client

6.1 Servidor

O servidor foi desenvolvido em C os dados dos preços a cada 5 minutos das ações estão salvos em um txt. O programa main realiza a leitura linha à linha do txt, onde para cada linha lida, é enviada ao cliente o preço “atual” da ação. Assim, a cada 1 segundo o servidor disponibiliza uma atualização no close price do stock específico.

6.2 Cliente

O cliente foi desenvolvido em python e após a criação do socket conectado ao servidor, começa a receber a cada 1 segundo respostas do server. A cada pacote recebido é chamada uma função que calcula, através do algoritmo das bandas de bollinger, e retorna se a ordem que será enviada ao server é de compra, venda ou hold. O gráfico abaixo exemplifica o cálculo das bandas de bollinger, através do histórico das ações da Apple.

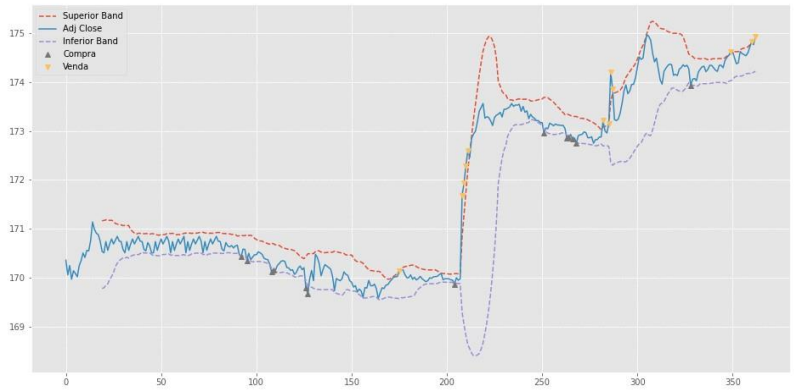


Gráfico 1 – Bandas de Bollinger, ações da Apple

6.3 Controle da Comunicação

Com o objetivo de se obter uma melhor interpretação dos dados e do desempenho da comunicação entre o cliente e o servidor, utilizou-se o software wireshark, que é uma ferramenta que permite a captura e intermediação das comunicações da rede, tanto para o protocolo TCP, quanto para o UDP. Assim, além do roteador que fornece a conexão da rede para o servidor e cliente, foi adicionado um switch. Onde a porta 4 foi configurada para espelhar a comunicação da porta 1 (servidor). Como apresentado na figura abaixo:

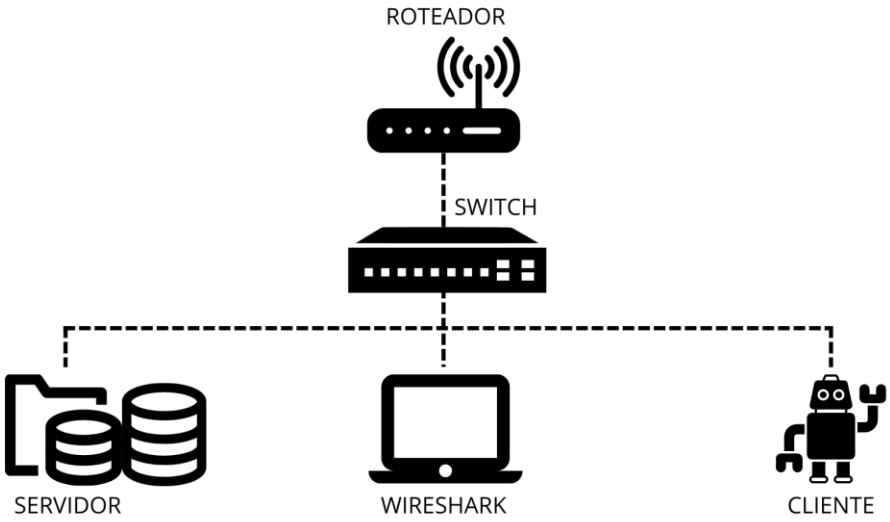


Figura 2 – Funcionamento do Ambiente de Simulação

7. Resultados

O modelo de FPGA escolhido foi a DE10-Standard, fabricada pela Terasic, que possui um core Intel Cyclone V – SoC, com grande capacidade de memória e diversas células lógicas, além do microcontrolador FPGA e ARM (responsável pelo funcionamento do kernel linux embarcado).

Com o intuito de validar o funcionamento do ambiente de desenvolvimento, o primeiro passo foi a implementação do algoritmo das Bandas de Bollinger em Python, com os cálculos matemáticos feito com pacote de cálculo do próprio Python e as listas foram utilizadas como arrays da biblioteca NumPy. A comunicação foi feita através da biblioteca sockets. O algoritmo foi desenvolvido de forma a receber na porta 5000 TCP o valor da ação e retornar uma ordem de *Buy*, *Sell* ou *Hold*. Assim, o arquivo python foi embarcado na Raspberry Pi 3, na FPGA, no Laptop e conectados no switch do ambiente de desenvolvimento para a recepção dos dados das ações, como apresentado na Figura abaixo:

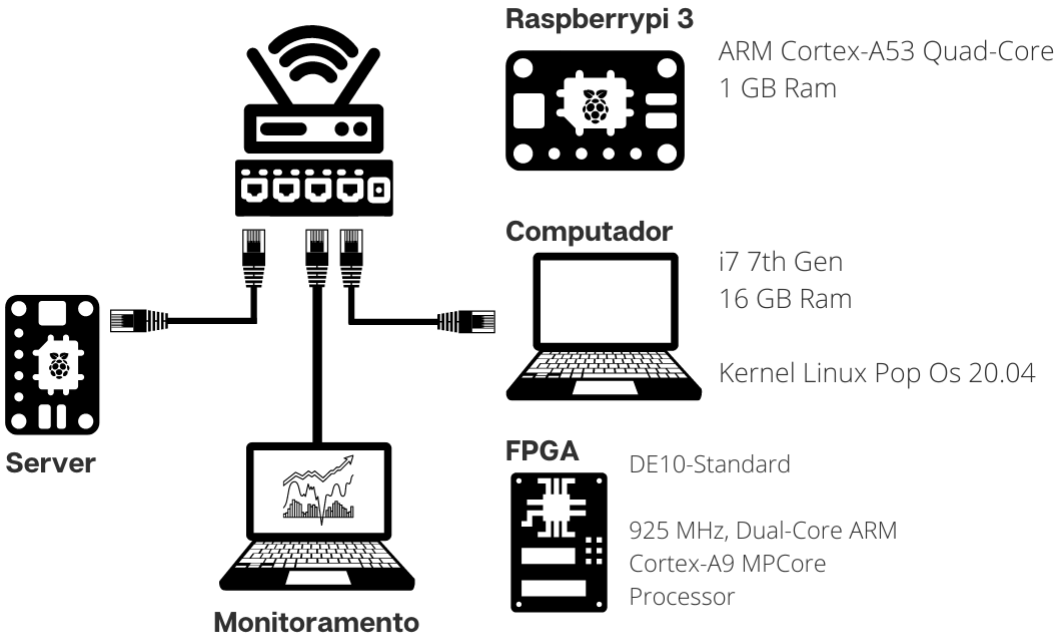


Figura 3 – Ambiente de Desenvolvimento e os Hardwares do Cliente.

Em um primeiro momento, foi realizado um comparativo entre o Intel i7, a RaspberryPi 3 e a FPGA DE10-Standard, com o algoritmo python. O resultado da comparação pode ser visualizado no Gráfico abaixo:

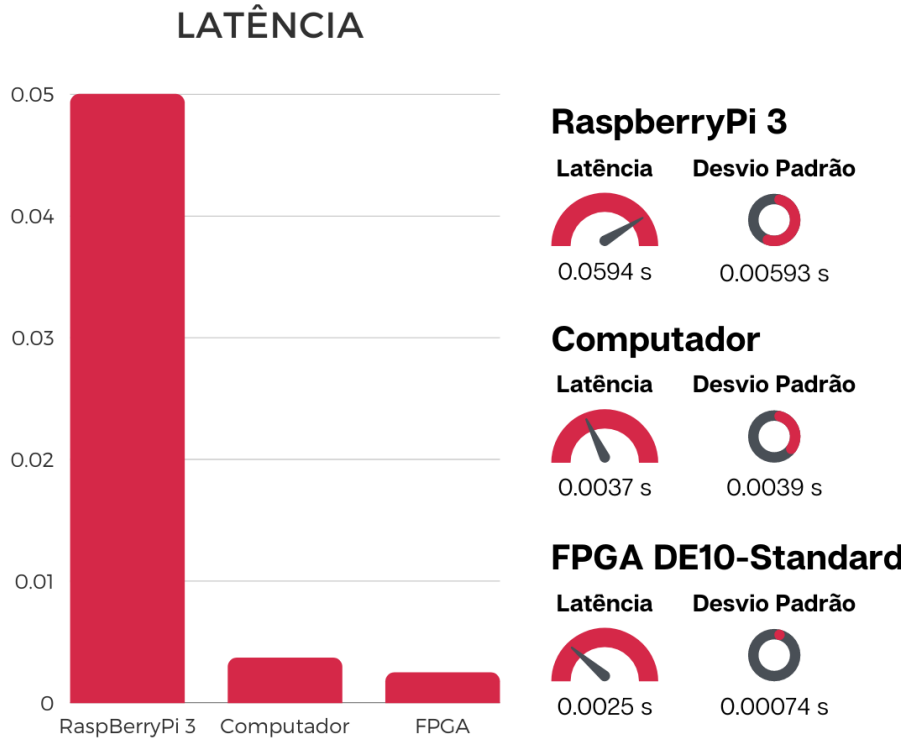


Gráfico 2 – Resultados da Aplicação em Python

Como visto no Gráfico acima, A FPGA de fato, apresentou um desempenho superior. Assim a aplicação python foi transferida para C, substituindo a utilização dos arrays do NumPy para arrays em C, com a vantagem da liberdade que a linguagem C permite, favorecendo a personalização do tamanho que os arrays ocupam na Memória através das funções *malloc* e *free*. O programa C foi compilado com o gcc do Linaro que disponibiliza o compilador com um conjunto de ferramentas para gerar um executável personalizado para rodar em ARM, arquitetura do chip da DE10-Standard.

Além da imagem do Kernel Linux distribuído pela Terasic, também foi utilizado para comparação um Kernel Real Time, fixado com 1 core, onde o algoritmo C possui a prioridade mais alta. Os resultados são apresentados no Gráfico 3 e 4:

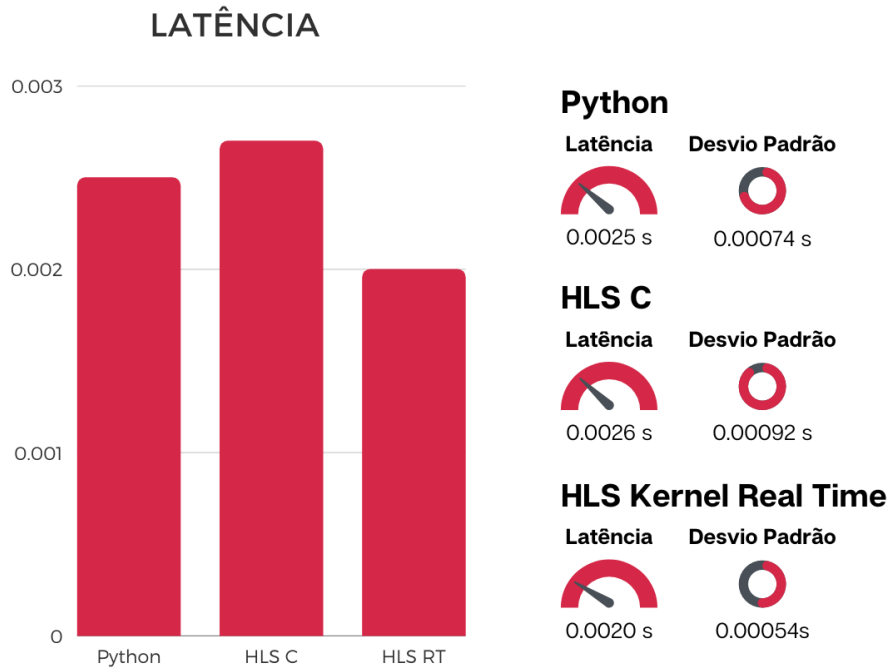


Gráfico 3 - Resultados das Aplicações na DE10-Standard

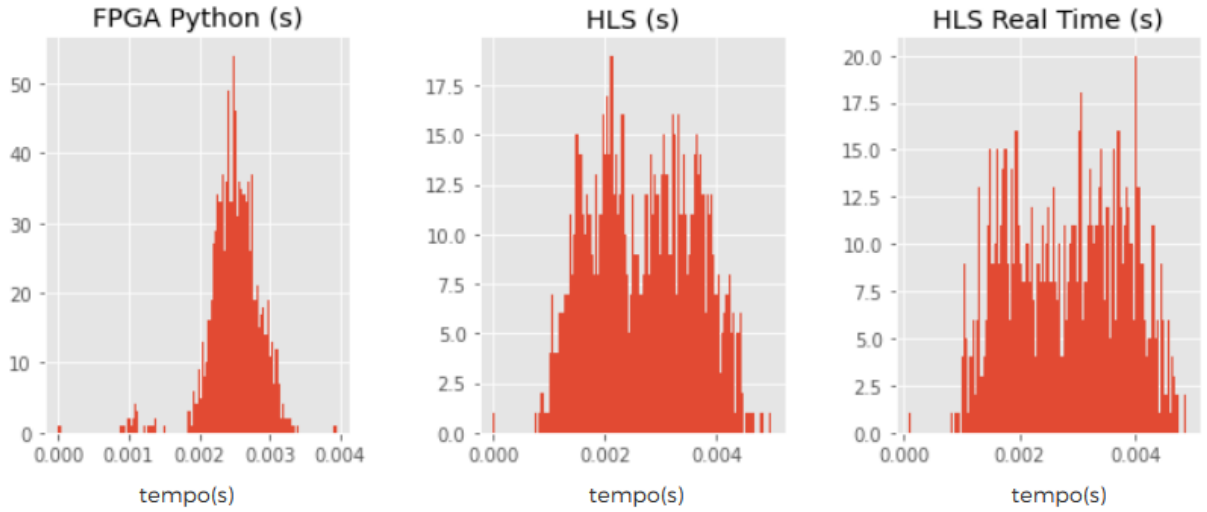


Gráfico 4 – Histograma das Latências

Embora o Python seja uma linguagem de programação interpretada e não compilada, em alguns casos onde há somente cálculos matemáticos, é possível que apresente um desempenho superior à um executável C, conforme o Gráfico 3 apresenta. Porém, ao utilizar o Kernel Real Time, obteve-se uma melhora de 25% 37,03% no desvio padrão. na latência em relação ao Python.

8. Próximos Passos

Para os próximos passos, dado que o ambiente de desenvolvimento já foi validado para comunicação socket via TCP, é necessário implementar um sistema UDP de forma a permitir que a FPGA, o Computador e Raspberry possam rodar de forma simultânea e verificar se de fato a FPGA envia a ordem de compra primeiro.

Outra modificação seria formatar os pacotes que são enviados para que sigam o protocolo FAST FIX, de modo a favorecer a credibilidade do ambiente desenvolvido em relação ao mercado financeiro do Brasil. Assim, também seria possível explorar o tempo de decodificação do pacote FIX e verificar se seria possível uma aceleração somente com a decodificação feita a nível de hardware.

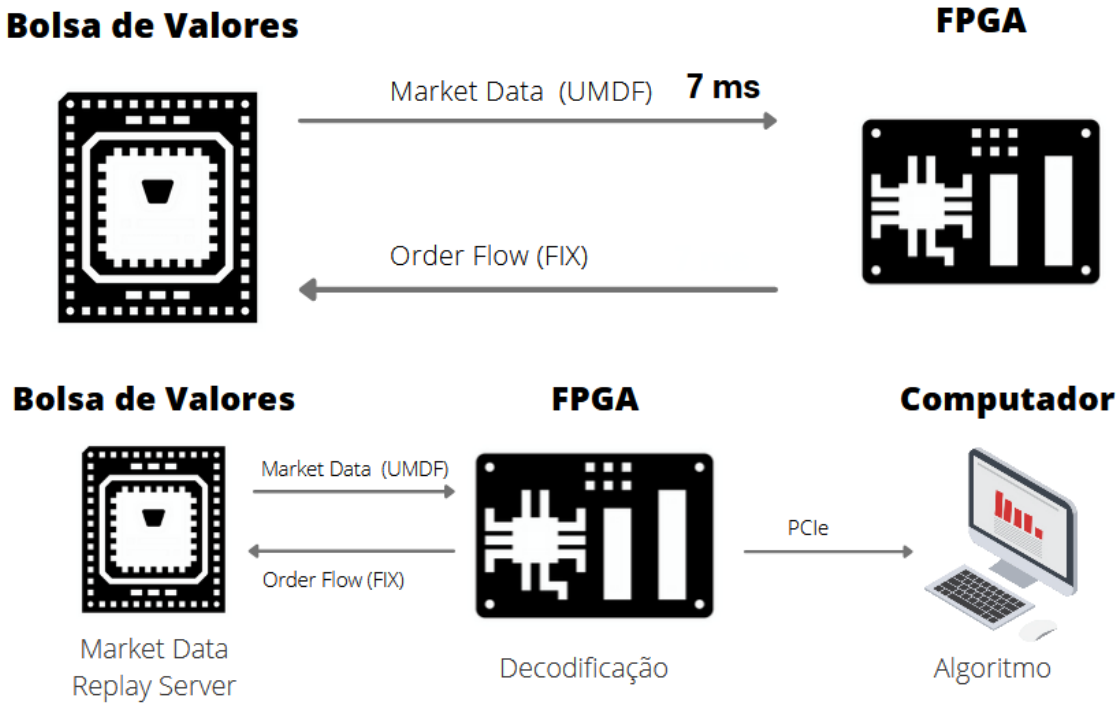


Figura 4 – Próximos Passos, Protocolo FIX e Decodificação em FPGA.

Referências

[1] P. Gomber, B. Arndt, M, Lutat, T. Uhle, **High Frequency Trading**. E-Finance Lab. Abril, 2011.

[2] G.W Morris, D.B. Thomas, and W. Luk, **FPGA Accelerated Low-Latency Market Data Feed Processing**, 2009 17th IEEE Symposium on High Performance Interconnects. 2009.

[3] C. Leber, B. Geib, H. Litz, **High Frequency Trading Acceleration using FPGAs**, University of Heidelberg. 2011.

[4] R. S. Sabedra, **Implementação em FPGA de algoritmo para análise de ativos financeiros na bolsa de valores**. Universidade Federal do Rio Grande do Sul, Instituto de Informática, 2020.

[5] J. A. Brogaard, **High Frequency Trading and Its Impact on Market Quality**. Northwestern University School of Law. Julho, 2010.

[6] Aldrige, I. **High-Frequency Trading – A Practical Guide to Algorithmic Strategies and Trading Systems**. [S.l.]: John Wiley Sons, Inc., 2010.

[7] N. Fujita, **Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming**. University of Tsukuba, Graduate School of Systems and Information Engineering, Tsukuba, Japan. 2019.

[8] BABBAR, R. **A study of performance of moving average, Bollinger bands and relative strength index in selected stocks and stock indices**. 2011.

[9] COSTA, C. R.; ROSA, L. d. S.; BONATO, V. **Integrando o metatrader5 com aceleradores fpga via opencl named pipes**. In: Workshop em Computação Heterogênea – WCH.[S.l.]: SBC, 2018

[10] A. Wilen, J. Schade, R. Thornburg. **Introduction to PCI Express, A Hardware and Software Developer's Guide**. Intel Press. 2003.