Implementação e Avaliação de Co-Processadores para BlackScholes em FPGA

Walter Sousa[1], **Marcelle Engle Carvalho Sacramento**[1], **Alexandre S. Nery**[1], **Maria Clícia S. Castro**[1]

[1]Departamento de Informática e Ciência da Computação
Instituto de Matemática e Estatística
Universidade do Estado do Rio de Janeiro (UERJ), Brasil

wjunior08,marcelle.engle@gmail.com,anery,clicia@ime.uerj.br

Abstract. *Option pricing is one of the main topics of the financial market. There are different models to perform this calculation, however BlackScholes is one of the most used nowadays. Due to this, the implementation of this algorithm shows an interesting study case for optimization. FPGAs (Field Programmable Gate Array) are boards commonly used in high perfomance computing, with promising results in several cases. Therefore, the main objective of this paper is to implement the Black Scholes formula in FPGA and evaluate its efficiency, using HLS (High level synthesis) and running in a Pynq-Z1 board. The results were compared with an execution in python on the ARM present on the board. The obtained results show an unexpected performance, and somes of possible explanations to this fact.*

Resumo. *A precificação de opções é um dos principais temas do mercado financeiro. Existem diferentes modelos propostos para a realização desse cálculo, porém o BlackScholes é um dos mais usados atualmente. Por isso, a implementação do seu algoritmo se mostra um caso interessante de estudo de otimização. FPGAs (Field Programmable Gate Array) são placas comumente usadas em computação de alto desemepnho, com resultados promissores em diversos casos. Assim, o principal objetivo desse trabalho é implementar a fórmula BlackScholes em FPGA e avaliar sua eficiência, utilizando HLS (High level synthesis) e executando em uma placa Pynq-Z1. O resultado foi comparado com a execução em python no ARM presente na mesma placa. Os resultados obtidos mostram um desempenho não esperado e algumas das possíveis explicações para este fato.*

1. Introdução

No mercado financeiro, opção é um meio de negociar o direito de compra ou venda de um ativo, em uma data futura por um preço prefixado. Ou seja, quem adquire uma opção está adquirindo o direito de comprar ou vender um ativo no futuro, por um preço predefinido que se manterá independente da flutuação desse ativo. Uma opção pode ser utilizada de diversos modos, como especulação, redução dos riscos de se ter um ativo, entre outros. [Ativo-objeto]. As características das opções estão relacionadas ao ativo ao qual se refere. Portanto, seu preço é derivado das variações do ativo objeto.

As opções são vendidas e revendidas por um preço conhecido como prêmio e, são divididas em categorias. As categorias mais comuns são as americanas e europeias. A principal diferença entre essas categorias é o direito de exercer a compra ou venda. Como dito em [Opções americanas e européias] na opção americana o titular pode exercer a compra ou venda até a data de expiração da opção. Já as européias, só podem ser executadas na data de expiração da opção.

Diversos modelos diferentes de precificação de opções têm sido usados no mercado financeiro. Atualmente, os modelos mais comuns são o Binomial e o Blackscholes. Ambos consideram diferentes variáveis no modelo de precificação, mas elas garantem resultados mais precisos. O modelo Blackscholes é relativamente simples e possui alta eficácia, assim ele é muito utilizado para encontrar preços justos [Precificação de Opções].

BlackScholes é um dos modelos mais comuns de precificação de ativos do mercado financeiro para as opções europeias. Fischer Black e Myron Scholes elaboraram o modelo, adaptando uma fórmula física para descrever a precificação de derivativos, no final dos anos 60. Eles publicaram em 1973 um artigo sobre o modelo, e tornaram-se ganhadores do prêmio Nobel, posteriormente, em 1997 [Precificação de Opções]. Segundo [Li 2012], este modelo é um dos mais importantes na teoria financeira quantitativa moderna.

Devido a grande importância dessa fórmula no mercado financeiro, este artigo busca mostrar uma forma mais eficiente para realizar os cálculos do modelo BlackScholes, através da utilização de FPGA.

Neste trabalho a implementação foi realizada com HLS (*High Language Syntesis*) para a programação da FPGA, permitindo o uso da linguagem C. Dessa forma, facilitando o entendimento, avaliação e a depuração do código. O ambiente de execução é composto pela FPGA PYNQ-Z1 que possui uma placa com a lógica da família Artix-7 e um processador ARM CortexTM-A9 Dual-Core, onde é possível executar o código python tanto para a chamada do *design*, quanto para comparação de desempenho.

Este artigo está organizado em oito seções. A Seção 2 discute alguns trabalhos relacionados que podem ser encontrados na literatura. A Seção 3 aborda a metodologia usada neste trabalho. As Seções 4 e 5 mostram uma visão geral sobre a fórmula BlackScholes e as FPGAs, respectivamente. As etapas de implementação do algoritmo estão na Seção 6 e os resultados obtidos são discutidos na Seção 7. Finalizamos com nossas conclusões e sugestões de trabalhos futuros na Seção 8.

2. Revisão da Literatura

Por volta dos anos de 2003 e 2004 na pesquisa [Hodjat and Verbauwhede 2004] alguns estudiosos utilizaram uma FPGA para implementar um algoritmo do padrão de criptografia avançado (AES - Advanced Encryption Standard). Este algoritmo é mais conhecido como Rijndael, e foi um padrão adotado pelo governo dos Estados Unidos. O seu maior obstáculo era a velocidade de encriptação e desencriptação dos dados. O processo era muito lento. Por isso, o algoritmo de criptografia AES foi implementado em FPGA, de formas diferentes. Uma das pesquisas, feita por Alireza Hodjat e Ingrid Verbauwhede, utilizou uma abordagem de *pipeline* para a implementação da encriptação, tendo um ren-

dimento de 21.54 Gbits/s. Este resultado é superior aos algoritmos implementados anteriormente sem *pipeline*, que obtiveram rendimento de 17.8Gbits/s e 18.56Gbit/s.

F. Rodríguez-Henríquez, N. A. Saqib e A. Díaz-Pérez realizaram outra pesquisa similar [Rodriguez-Henriquez et al. 2003], que também utilizou a abordagem de *pipeline*. Porém, para um algoritmo de encriptação e desencriptação de dados. Eles realizaram algumas melhorias no desempenho no algoritmo e depois implementaram em FPGA. Foi obtido, então, um rendimento de 4.2Gbits/s, também superior aos algoritmos anteriores de 3.2Gbits/s e 2.2Gbits/s.

Mais próximo desse trabalho temos o artigo [Li 2012] Achieving Superior Performance on Black-Scholes Valuation Computing using Intel® Xeon Phi™ Coprocessors. Escrito por Shuo Li, funcionário da intel, que implementou o Black-Scholes no coprocessador Xeon Phi. Ele compara seu desempenho com as plataformas mais conhecidas, e outros tipos de aceleração, como a paralelização. O resultado mostrou que o coprocessador Xeon-Phi é 3.33 vezes mais rápido em comparação a implementação do BlackScholes totalmente paralelizada no Xeon [Li 2012]. A Figura 1 mostra os *speedups* de obtidos com os coprocessadores Intel Xeon e Xeon Phi.

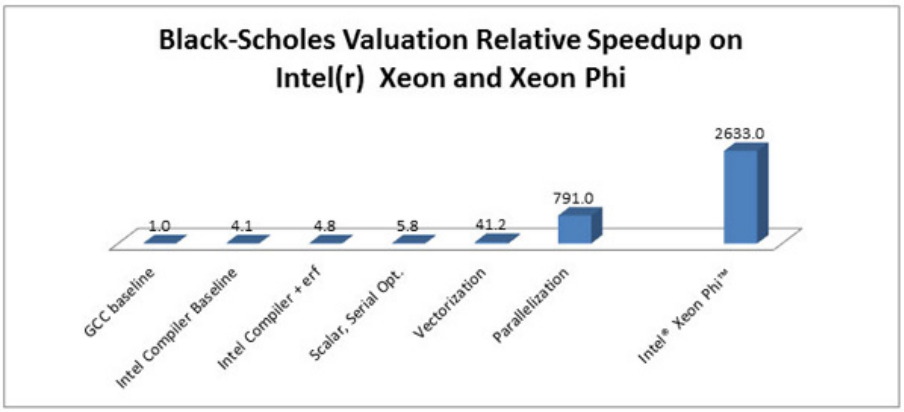

Figura 1. Gráfico dos *speedups* de comparação do processamento do algoritmo BlackScholes nos coprocessadores Intel Xeon e Xeon Phi.

Em [Eriko Nurvitadhi and Marr 2016] foi realizada uma avaliação em relação a melhoria da eficiência de algoritmos de redes neurais binarizadas *(BNNs - Binarized neural networks)* por meio de aceleração de hardware. O acelerador foi implementado no Aria 10 FPGA e no ASIC de 14 nm. Para a comparação dos resultados, foi utilizado também o software otimizado em uma CPU multi-core de alto desempenho e GPU para servidores na nuvem. Pelos resultados obtidos foi concluído que a FPGA se mostra mais eficiente que a CPU e GPU. Porém, A FPGA não é mais eficiente que a implementação no ASIC. Mesmo assim, foi observado que a FPGA é uma solução atrativa, proporcionando uma mehora de eficiência superior, sem ter que se prender a uma solução ASIC fixa.

3. Metodologia

Neste trabalho, para implementar e avaliar o desempenho do modelo BlackScholes em FPGA é utilizado o software Vivado HLS da Xilinx (https://www.xilinx.com). O algoritmo foi primeiro adaptado para a linguagem C++, de modo a utilizar as bibliotecas disponibilizas pela Xilinx, específicas para suas FPGAs.

O algoritmo também foi implementado em python 3 rodando sobre o PYNQ linux, que é baseado no Ubuntu 18.04, utilizando o ARM presente na placa PYNQ-Z1. Essa execução foi utilizada para comparação dos dados de saída e dos tempos obtidos na execução via FPGA.

As etapas do trabalho consistem das otimizações na implementação, da síntese, e da comparação entre soluções já existentes e as implementadas nesse trabalho.

4. BlackScholes

Atualmente muito conhecido no mercado financeiro, o modelo BlackScholes tem como principal proposta a precificação de opções. Para isso, são utilizadas apenas as seguintes variáveis conhecidas [Precificação de Opções]:

$$d1 = \frac{ln(\frac{s}{x}) + (r + \frac{\sigma^2)}{2})T}{\sigma\sqrt{T}} \tag{1}$$

$$d2 = d1 - \sigma\sqrt{T} \tag{2}$$

Opção de compra:

$$c = S.CND(d1) - X.e^{-rt}.CND(d2) \tag{3}$$

Opção de venda:

$$p = X.e^{-rt}.CND(-d2) - S.CND(-d1) \tag{4}$$

onde as variáveis têm os seguintes significados:

σ = Volatilidade, é uma medida que determina a variação do preço de um ativo;
r = Taxa de juros livre de risco (SELIC no Brasil);
T = Tempo restante para o exercício da opção em anos;
S = Preço atual do ativo que a opção se refere;
X = Preço de exercício, valor que será pago pelo ativo caso a opção seja exercida;
CND = Função de distribuição cumulativa normal.

O modelo é baseado nas seguintes hipóteses [BlackScholes]:

- Os preços das ações se comportam correspondentemente a um modelo lognormal com desvio padrão e média constante;
- Não existem custos de transação;
- Não existe arbitragem possível;
- Contratos são divisíveis;
- São contínuas as transações de títulos e ações;
- A taxa livre de risco é a mesma para todos os investidores;
- A taxa de juros livre de risco no curto prazo é constante.

A fórmula do BlackScholes propõe um cálculo diferente de acordo com o seu tipo da opção, que pode ser de compra ou de venda.

Call é como as opções de compra são denominadas. Elas conferem ao titular o direito de comprar um ativo pelo valor determinado no contrato da opção, garantindo o poder de compra mesmo que o ativo tenha valorizado no momento de exercer

a opção[Opções de Compra]. Já as opções de venda, chamadas de *Put*, permitem que o titular tenha o direito de vender um ativo específico pelo preço presente no contrato da opção [Opções de Venda].

A fórmula BlackScholes mostra-se eficiente no suporte a tomada de decisões referentes a negociação de opções financeiras. Por isso, é pertinente melhorar a velocidade do seu cálculo. Pouco explorada para esse fim, a FPGA, que está descrita na Seção 5, demonstra um bom potencial de aceleração do cálculo da fórmula BlackScholes, e melhorar, assim, as decisões nas negociações financeiras.

5. FPGA

Uma FPGA (Field Programmable Gate Array) é um *chip* baseado numa matriz de blocos lógicos configuráveis (CLBs). Lançada pela Xilinx Inc., as FPGAs podem ser reprogramadas, mesmo depois de serem manufaturadas para assumirem formas de diferentes circuitos [Field-programmable gate array]. Esses *chips* são compostos por blocos lógicos, blocos de entrada e saída e chaves de interconexão, como pode ser visto na Figura 2. Cada um desses blocos está descrito a seguir [FPGA Fundamentals]:

- **CLBs (Configuration Logical Blocks)**: blocos lógicos configuráveis que são a unidade lógica básica de uma FPGA, composto pela reunião de *flip-flops* e *lookup tables* (LUTs);
- **IOB (Input/Output Block)**: *buffers* onde podem ser configurados registros de entrada e saída, possibilitando o acesso ao mundo externo;
- **Programmable Interconnects**: são trilhas usadas para conectar as entradas e saídas dos blocos lógicos.

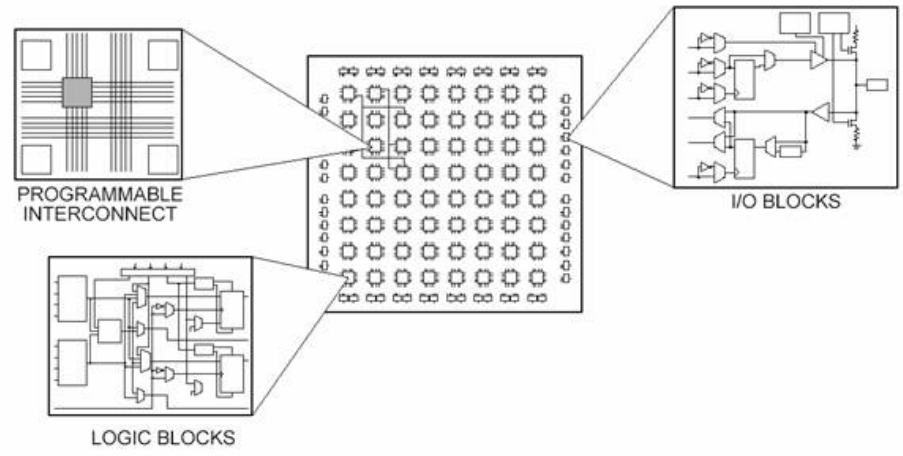

Figura 2. Partes principais que constituem uma FPGA genérica.

Uma FPGA possui uma matriz de portas lógicas reconfiguráveis. Quando a FPGA é configurada, é criado um circuito interno. Este circuito conecta as portas lógicas de forma a criar uma implementação em hardware do aplicativo de software. Diferentemente dos processadores, as FPGAs utilizam um hardware dedicado para o processamento lógico.

Os recursos internos de um *chip* FPGA consistem em uma matriz de blocos lógicos configuráveis (CLBs) cercados por uma periferia de blocos de entrada e saída. Os

sinais são roteados dentro da matriz FPGA por *switches* de interconexão programáveis e rotas de fios.

5.1. Níveis de abstração

Assim como nos computadores, as FPGAs podem ser programadas em diversos níveis de abstração.

Um dos níveis mais comuns é o nível RTL (*Register Transfer Level*), onde se usam linguagens de descrição de hardware, como VHDL e Verilog. Existe também o nível do algoritmo, onde se programa utilizando linguagens de alto nível, como C ou C++. Este nível é conhecido como síntese de alto nível (*High-Level Syntesis*) e facilita a programação das FPGAs.

Neste trabalho foi utilizado o software Vivado HLS (Xilinx), para que a aplicação desenvolvida em linguagem de alto nível C++ fosse compilada para o nível RTL. Sendo, depois, traduzida para a linguagem usada HDL (*Hardware Description Language*).

5.2. Placa FPGA PYNQ-Z1

Os resultados expostos nesse trabalho foram obtidos com a placa PYNQ-Z1 (Figura 3). Essa placa possui um processador ARM Cortex™-A9 Dual-Core 650MHz que inclui um servidor *web* que hospeda o ambiente de *design* do Notebook Jupyter, pacotes e núcleo IPython, linux e biblioteca de hardware e API para FPGA. Possui também um controlador de memória DDR3 com 8 canais DMA e 4 porta AXI3 de alta performance. Ela foi projetada para ser usada com PYNQ (Python Productivity for Zynq), um *framework open-source* que facilita os projetistas de sistemas embarcados explorarem os benefícios exclusivos dos APSoCs (All Programmable System-on-Chip) em seus aplicativos, sem ter que usar ferramentas de projeto no estilo ASIC (Application Specific Integrated Circuits) para projetar circuitos lógicos programáveis. [Digilent PYNQ-Z1]

O protocolo de comunicação AXI (*Advanced Extensible Interface*)[Xilinx 2012] viabiliza a troca de informações entre o processador e a área programável. Esse protocolo foi definido pela AMBA (*Advanced Microcontroller Bus Architecture*). Ele é um padrão aberto de especificação de conexão de *chip* para conexão e gerenciamento de blocos funcionais. Suas interfaces são: AXI4-Lite, AXI4 (ou AXI-Full) e Axi-Stream [Xilinx 2011]. A interface mais simples é a AXI4-Lite, geralmente ela é usada mais para sinais de controle e um fluxo de dados pequeno. Já a interface AXI-Full é um pouco mais complexa, sendo utilizada para uma maior transferência de dados, possui rajadas maiores, tendo um limite de 256 dados de 32 bits por envio, no máximo. A interface AXI-Stream é usada para grandes volumes de dados contínuos, sem limite de rajadas curtas.

6. Implementação

6.1. Otimizações

O pseudocódigo da função CND original, que implementa a função distribuição acumulada, pode ser viso no Algoritmo1. Nele é possível verificar que são utilizadas 4 potências da variável K, que são as potências 2, 3, 4 e 5. Para reduzir o tempo do cálculo das potências é possível reutilizar os resultados de uma potência anterior para calcular as demais. O Algoritmo2 é uma versão otimizada da versão original. Ela gerou uma melhoria de 0,4ns (redução de 6,141ns para 5,740ns) na estimativa de tempo entregue pela síntese do HLS.

```
   entrada: X
   saída  : w
1  a1 = 0.31938153;
2  a2 = -0.356563782;
3  a3 = 1.781477937;
4  a4 = -1.821255978;
5  a5 = 1.330274429;
6  L = abs(X);
7  K = 1.0 / (1.0 + 0.2316419 * L); w = 1.0 - 1.0 / sqrt(2 * Pi) * exp(-L *L / 2) *
     (a1 * K + a2 * pow(K,2) + a3 * pow(K,3) + a4 * pow(K,4) + a5 *
     pow(K,5));
8  se X ¡ 0 então
9  |   w = 1 -w;
10 fim
```

Algoritmo 1: Pseudocódigo a função CND original.

```
   entrada: X
   saída  : w
1  a1 = 0.31938153;
2  a2 = -0.356563782;
3  a3 = 1.781477937;
4  a4 = -1.821255978;
5  a5 = 1.330274429;
6  L = abs(X);
7  K = 1.0 / (1.0 + 0.2316419 * L);
8  k2 = K*K;
9  k3 = K*k2;
10 k4 = k2*k2;
11 k5 = k4*K;
12 w = 1.0 - 1.0 / sqrt(2 * Pi) * exp(-L *L / 2) * (a1 * K + a2 * k2 + a3 * k3 + a4
     * k4 + a5 * k5);
13 se X ¡ 0 então
14 |   w = 1 -w;
15 fim
```

Algoritmo 2: Pseudocódigo a função CND otimizado.

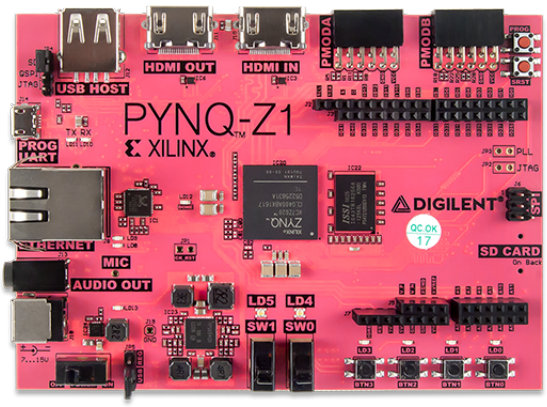

Figura 3. Placa PYNQ-Z1.

6.2. Síntese

Para que fosse feita a síntese do código em C para uma linguagem de descrição de hardware foram necessárias poucas modificações:

- Mudança da biblioteca math.h para hls_math.h;
- Adição dos pragmas para indicar os tipos de interfaces usadas para entrada e saída.

A síntese desse código teve como resultado um *design* com tempo de execução de aproximadamente foi de 5,740 ns.

O código em linguagem de descrição de hardware é integrado a outros circuitos, como o controle de *clock*, que garantirão o funcionamento da entrada e saída, criando um *design* que pode ser carregado na FPGA, como pode ser visto na Figura 4.

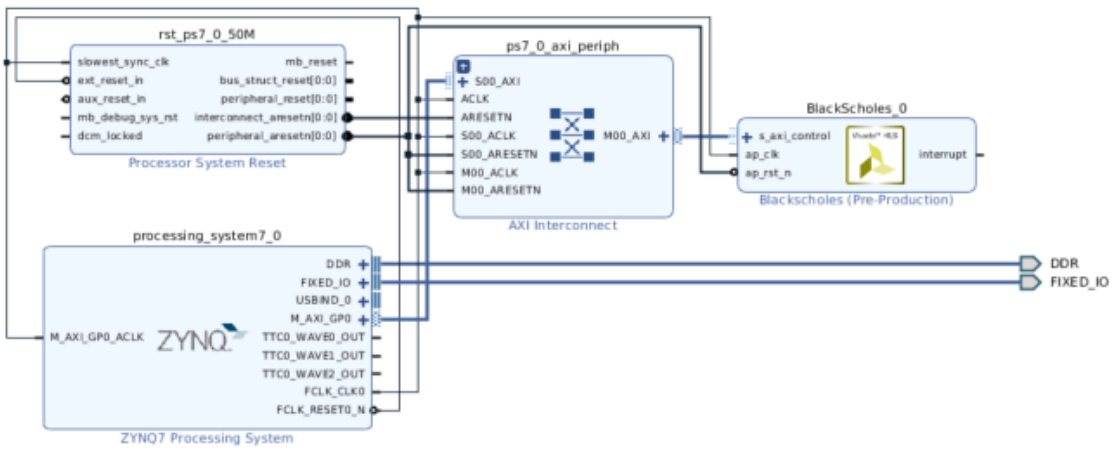

Figura 4. Visão geral da arquitetura.

7. Análise dos Resultados

Foram feitos experimentos com um conjunto de 1.000.000 entradas na FPGA com dados do PARSEC [Bienia 2011]. A Tabela 1 apresenta os resultados para 10 execuções do algoritmo implementado somente em Python e utilizando a FPGA, o tempo foi contabilizado em segundos e não contempla a leitura dos dados.

Tabela 1. Resultados obtidos para 1 Milhão de opções distintas com dados do PARSEC.

Execução	Python	FPGA
1	55,44	328,04
3	55,59	328,09
4	55,39	328,09
5	55,50	328,01
6	55,57	328,08
7	55,55	327,96
8	55,52	328,01
9	55,55	328,11
10	55,48	328,22
Média	55,52	328,08
Desvio Padrão	0,06	0,59

O programa Vivado HLS, ao final da síntese faz uma estimativa de tempo para uma execução no design feito. O tempo estimado para o design do algorítimo do BlackScholes foi de 5,740 ns para cada dado executado, com uma imprecisão de 0,75 ns.

A média dos valores da execução aferidos diretamente em pyhton foi de 55,52 segundos para 1 milhão de execuções, com 0,06 segundos de desvio padrão. Deste modo, para cada opção o tempo foi em média foi de 55524 ns. Este valor é 9673 vezes maior que o estimado pelo HLS. Porém, a execução dos cálculos utilizando a FPGA teve, para o mesmo número de execuções, uma média de 328,08 segundos com desvio padrão de 0,59 segundos. Isto resulta em 328087 ns por opção. Sendo assim, a execução na FPGA se mostrou mais lenta que a realizada com python.

8. Conclusões e Trabalhos Futuros

Este artigo mostrou uma implementação do algoritmo BlackScholes utilizando uma FPGA.

Pelos resultados obtidos, pode-se concluir que o cálculo do algoritmo do BlackScholes implementado em uma FPGA PYNQ-Z1 utilizando a interface AXI-Lite não teve ganho de desempenho se comparado com o algoritmo em Python executado no ARM Cortex™-A9 presente na mesma placa. Assim, a solução implementada na FPGA se mostrou ineficiente. Comparando o tempo de cálculo estimado na síntese do HLS e a média obtida por execução, é possível afirmar que o tempo total para cada entrada é muito maior que o tempo de cálculo. Dessa forma, podemos considerar este fato como a maior causa possível para a baixa eficiência dos resultados obtidos. Isto é, o tempo de comunicação com a FPGA.

Pretendemos continuar este projeto utilizando outras formas de comunicação com a FPGA, de modo a reduzir o tempo de comunicação dos dados de entrada. Esta redução pode ser atingida implementando o mesmo algoritmo na interface AXI-Full utilizando rajada de dados maiores. Uma outra forma de obter um desempenho mais satisfatório seria implementar o cálculo da volatilidade. Assim, se reduziria o volume de dados de entrada no sistema para cada cálculo.

Como trabalhos futuros podemos também considerar uma implementação que envolvesse um *trading* de ações. Para lucrar com as operações de curto prazo no mercado financeiro, um *trader* é um investidor que busca o momento certo para realizar as operações de acordo com a volatilidade do mercado. A FPGA pode auxiliar e facilitar o trabalho de um *trader*, analisando e calculando o lucro de uma operação naquele determinado momento, de acordo com o preço do ativo em tempo real. Realizando assim, um trabalho rápido e preciso para a tomada de decisão do investidor.

Referências

Ativo-objeto. Ativo-objeto. https://www.bussoladoinvestidor.com.br/abc_do_investidor/ativoobjeto/.

Bienia, C. (2011). Benchmark ing modern multiprocessors.

BlackScholes. Black-Scholes: O Conhecido Modelo De Precificação De Opções. https://www.sunoresearch.com.br/artigos/black-scholes/.

Digilent PYNQ-Z1. Digilent PYNQ-Z1. https://www.xilinx.com/products/boards-and-kits/1-hydd4z.html.

Eriko Nurvitadhi, David Sheffield, J. S. A. M. G. V. and Marr, D. (2016). Accelerating binarized neural networkss: Comparison of fpga, cpu, gpu, and asic.

Field-programmable gate array. What is an FPGA? https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.

FPGA Fundamentals. FPGA Fundamentals. http://www.ni.com/white-paper/55015/en/#toc1.

Hodjat, A. and Verbauwhede, I. (2004). A 21.54 gbits/s fully pipelined aes processor on fpga. *IEEE Xplore Digital Libary*.

Li, S. (2012). Achieving superior performance on black-scholes valuation computing using intel® xeon phi™ coprocessors. *Intel Software Developer Zone*.

Opções americanas e européias. Opções americanas e européias: diferenças não se devem à posição geográfica. https://www.infomoney.com.br/educacao/guias/noticia/380248/opcoes-americanas-europeias-diferencas-nao-devem-posicao\-geografica.

Opções de Compra. O que é uma opção de compra (call)? https://br.advfn.com/investimentos/opcoes/opcao-de-compra.

Opções de Venda. O que é uma opção de venda (put)? https://br.advfn.com/investimentos/opcoes/opcao-de-venda.

Precificação de Opções. Precificação de Opções: Modelo Black e Scholes. https://blog.bussoladoinvestidor.com.br/precificacao-black-scholes/.

Rodriguez-Henriquez, F., Saqib, N., and Diaz-Perez, A. (2003). 4.2 gbit/s single-chip fpga implementation of aes algorithm. *IEEE Xplore Digital Libary*.

Xilinx (2011). Axi reference guide.