CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	A parallel and pipelined implementation of a Pascal-simplex based two asset option pricer on FPGA using OpenCL
Authors	O'Mahony, Aidan T.;Zeidan, Gil;Hanzon, Bernard;Popovici, Emanuel M.
Publication date	2020-10-27
Original Citation	O'Mahony, A., Zeidan, G., Hanzon, B. and Popovici, E. (2020) 'A parallel and pipelined implementation of a Pascal-simplex based two asset option pricer on FPGA using OpenCL', NorCAS 2020, IEEE Nordic Circuits and Systems Conference, 27-28 October, Oslo, Norway, Virtual Conference, (6 pp).
Type of publication	Conference item
Link to publisher's version	https://events.tuni.fi/norcas2020/
Rights	© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works
Download date	2026-11-11 13:49:52
Item downloaded from	https://hdl.handle.net/10468/10682

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

A Parallel and Pipelined Implementation of a Pascal-Simplex Based Two Asset Option Pricer on FPGA using OpenCL

Aidan O Mahony
Department of Electrical and Electronic Engineering
University College Cork
Cork, Ireland
103837793@umail.ucc.ie

Gil Zeidan
Programmable Solutions Group
Intel Corporation
Massachusetts
USA
gilbert.zeidan@intel.com

Bernard Hanzon
Department of Mathematics
University College Cork
Cork, Ireland
b.hanzon@ucc.ie

Emanuel Popovici
Department of Electrical and Electronic Engineering
University College Cork
Cork, Ireland
E.Popovici@ucc.ie

***Abstract*—With the resurgence of hardware for financial technology, several methods for accelerating financial option pricing using reconfigurable hardware have been investigated. This paper presents the first architecture and implementation of a two-asset option pricer based on Pascal's simplex, which takes advantage of the parallelism and pipelining offered by FPGA technology. The theory that this architecture is constructed from is based on a recombining multinomial tree approach which in turn is a generalization of the binomial tree model. Furthermore, we show that while a significant difficulty exists in efficiently maintaining the intermediate values required for the computation, a solution exists in the form of FIFOs.**

Our implementation, on an Intel Stratix 10 GX FPGA, is based on the OpenCL framework and can compute 6250 two asset option prices per second for a time step of 100 and the pipelining of the option value computation show a 25 times improvement when a 50-step pipeline is created.

***Index Terms*—Finance, FPGA, OpenCL, European Option, Field programmable gate arrays, Acceleration, Pricing, Simplex, Multiple Assets**

I. Introduction

Hardware accelerators based on Field-Programmable Gate Arrays (FPGA) have found many applications in finance [1] [2]. The ability to create custom circuits for frequently performed complex operations [3] as well as the ability to parallelize financial applications on FPGAs [4] make them attractive for acceleration purposes. Examples are discussed in [5], [6], and [7].

Financial options are one tool that can be utilized by businesses in increasing the potential for profit [8]. One of the most popular models used for computing the value of an option is the Black-Scholes model [9]. This approach allows for generating a formula which provides an estimate of the price of an option. An alternative model is the *Binomial option pricing model*, created by Cox, Ross and Rubinstein [10]. The two approaches (Black-Scholes and Binomial option pricing model) provide for calculating the value of an option on a single asset. In this paper, we are concerned with calculating the value of an option on multiple assets [7].

The remainder of the paper is as follows: Section 2 discusses related research in this area. In Section 3 we provide an overview of financial options, the Binomial options pricing model, and then we provide a summary of the theory underlying the computation of the price of a derivative on multiple assets using recombining multinomial trees. Section 4 outlines the architecture of the reconfigurable option pricing hardware accelerator as well as detailing the OpenCL kernels required to instantiate the architecture. Section 5 provides details on the test environment as well as results from the implementation. Finally, in Section 6, we review our conclusions.

II. Related Work

Option pricing acceleration has been explored in the past [11]. However, no implementation to date has implemented two-asset option pricing using Pascal's Simplex. In this section, we review the existing research in this area to demonstrate the novelty of our approach.

In [12] an OpenCL Option Pricer on an Altera Stratix IV is presented which can achieve 2000 options/s for a depth of 1024. To reuse this implementation for two assets requires the computation of all combinations of possible outcomes which potentially results in a computation rate of 30 options/s (see Section III-C for comparison method).

An alternative implementation for computing multi-asset option prices on FPGA is presented in [13] where a Monte Carlo approach is used. While this approach achieved a 123 times speedup on a single FPGA, the accuracy and convergence issues that exist in the Monte Carlo method when compared to the Binomial Model [14] were not addressed. Other Monte Carlo implementations on FPGA are also presented in [3] and [15] without accuracy comparisons however in [16] it is noted that the Monte-Carlo solver is at least 100 times less accurate than other pricing methodologies. Indeed, they remark that their results show that the FPGA based Monte

978-1-5386-5541-2/18/$31.00 ©2020 IEEE

Carlo solver should only be used when there are no other solvers available.

Quadrature Methods for Option Valuation are considered in [17] and [18]. This approach compares well against other methods in terms of accuracy. However, as presented in [16], the quadrature method does not compare favourably with the tree-based approach as the time variable increases.

Other combinations of hardware acceleration of option pricing also exist, e.g. CPU-GPU hybrid [19]. However, none of these alternative approaches allows for multi-asset option pricing.

In the prior research surveyed, CPU implementations generally compare poorly in terms of performance [20], and as a result, we do not consider a CPU implementation in our work. The same survey noted research that concluded that FPGAs provide better performance portability in terms of achieved percentage of device's peak performance compared to NVIDIA GPUs [21], and FPGA based option pricing outperform GPU based option pricing [22] in terms of latency.

Our two-asset option pricer based on Pascals Simplex is shown to be novel based on the current state of the art. Furthermore, based on our survey, the FPGA based Simplex based multi-asset option pricer also has clear differences compared to alternative approaches such as using a single asset option pricer, Monte Carlo based multi-asset option pricer, or Quadrature based multi-asset option pricers.

III. Financial Options

The most common definition of an option is an agreement between two parties, the option seller and the option buyer, whereby the option buyer is granted a right (but not an obligation) to carry out some operation (or exercise the option) at some moment in the future. There are also options with more general payoffs; usually these can be approximated by an appropriate portfolio of call and put options. The predetermined price is referred to as strike price, and the future date is called the expiration date [23]. There are two main types of options. A call option grants its holder the right to buy the underlying asset at a strike price at some moment in the future. A put option gives its holder the right to sell the underlying asset at a strike price at some moment in the future.

The question is, what determines option values? We know the value of an option when it matures, but how do we compute the value initially? Several variables are relevant to these calculations [23], including the price of the asset, the exercise price, delay in paying the exercise price, the volatility of the asset price over time, and the term of the option. These variables are considered in the Binomial Option Price Model.

A. Binomial Option Price Model

The binomial model is a discrete-time model for pricing options in which it is assumed that price changes in the underlying assets occur only after regular time intervals [24]. It involves constructing a binomial tree that represents different possible paths that the price of the underlying asset might follow. Let Z_T be the stock price at the expiration time T and K the strike price. Then the pay-off of a European call option is $c(T) = \max(Z_T - K, 0)$, which reflects the fact that an option might be exercised if $Z_T > K$ and might not be exercised if $S_Z \leq K$. The payoff of a European put option is $p(T) = \max(K - Z_T, 0)$, which reflects the fact that an option might be exercised if $Z_T < K$ and might not be exercised if $Z_T \geq K$.

Consider a non-dividend paying stock whose price is initially Z_0. Divide time into small time intervals of length δt. A time interval will be referred to as a period. Denote by Z the initial stock price at the beginning of a time interval. Assume that in each time interval the stock price moves either to $d_1 Z$ (an "up" movement) or to $d_2 Z$ (a "down" movement). The parameters d_1 and d_2 are equal to one plus the realized return during the time interval. In general, $d_1 > 1$ and $d_2 < 1$. Let q be the probability of an "up" movement and $(1 - q)$ be the probability of a "down" movement.

In 1979, Cox, Ross, and Rubinstein introduced a method to approximate the value of options on a single asset Z via a binomial tree [10]. The goal is to construct a recombining binomial tree with N levels to approximate the price process of an asset Z and the price of an option. In each time-step, the value of the asset can go either up or down (although $0 < d_2 < d_1$ is all that is required), where the asset price at time 0 equals Z_0. Each node in the tree represents a possible outcome of the asset price and has a certain probability.

B. Recombining Multinomial Trees Based on Pascal's Simplex

This method is a generalization to options on several assets of the well-known binomial tree method, which is used for option prices with one underlying asset. An advantage of the method introduced in [25] is that it has a minimal number of successor nodes at each node in the tree. Therefore, the proliferation of nodes in the case of several assets is less severe than in other tree-based approaches (such as using a separate binomial tree for each asset where all combinations of possible outcomes need to be computed).

By exploiting the fact that the simplex-trees have close connections to polynomial algebra (i.e. each node can be associated with a multi-index $\nu = (\nu_1, \nu_2, \ldots, \nu_{k+1})$, with corresponding time step index $t = |\nu| := \sum_{j=1}^{k+1} \nu_j$, which in turn can be associated with a monomial $x^\nu = x_1^{\nu_1} x_2^{\nu_2} \ldots x_{k+1}^{\nu_{k+1}}$ of auxiliary variables $x_1, x_2, \ldots, x_{k+1}$) we can use monomial ordering [26] to help with the bookkeeping of the values attained at the nodes of the tree and the relation with the values at the neighbouring nodes.

Figure 1 presents a visualization of a projection of the random walk of the recombining trinomial tree in three dimensions to two dimensions with three-time steps of length δt on two assets Z_1 and Z_2. The random walk starts at the black node on top, which is projected on the black node on the two-dimensional plane. From this node, there are three possible moves to time δt, which are represented by the red nodes and lines. The moves from δt to $2\delta t$ are represented by the blue lines, and the green lines represent the moves from time $2\delta t$

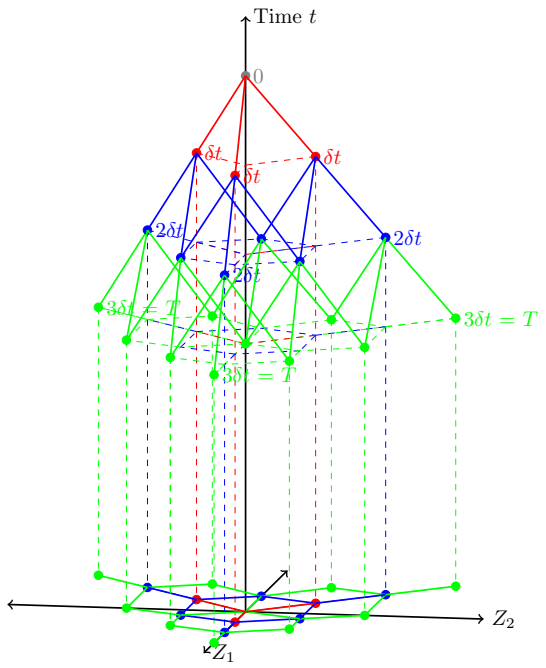

Fig. 1. Example of 3-dimensional to 2-dimensional projection

to time $3\delta t$, and the possible outcomes at time $T = 3\delta t$ are represented by the green nodes and the black node.

The option price computation is performed as follows: at final time T the asset price vector $Z(T)$ has a value on each of the $\binom{T+2}{2}$ (in general $\binom{T+k}{k}$) nodes ν on the tree at the final time. Let values of Z, F on the tree at node ν now be denoted as $Z(\nu), F(\nu)$(F denotes the pay-off function) respectively. We also define $F(v)$ as the value $F(t, Z)$ where (t, Z) are the values of the time and of the process Z at node v. For each ν with $|\nu| = T$ the option value $V(\nu) = F(Z(\nu))$ is calculated ($V(v)$ defined in equation 1). The value (for a European option) at node ν at time $t = |\nu| < T$, given the values at time $t+1$ at the successor nodes $\nu + e_j$, $j = 1, 2, 3$, where e_j stands for the standard basis row vector with entry 1 on position j and zeros elsewhere, is given by

$$V(\nu) = \sum_{j=1}^{3} p_j V(\nu + e_j) e^{-r}, \tag{1}$$

where r stands for the interest rate per time unit. So the task to be carried out by the acceleration device is (1) to generate the asset price vectors $Z(\nu), |\nu| = T$ in case of European options (and $Z(\nu), |\nu| = 0, 1, 2, \ldots, T$ in case of American options, which we do not treat here, however), (2) to compute the option prices $V(\nu) = F(Z(\nu))$ at the final time $|\nu| = T$, and (3) to iteratively compute the option values $V(\nu)$, for decreasing values of $|\nu|$, for all $|\nu|$ with $|\nu| \leq T$, ending up with $V(0)$ which is the desired option value approximation.

C. Comparison against using Binomial Trees for Multi-Asset Computation

The number of nodes required to construct the binomial trees when compared to the simplex based approach has an implication on the computation effort required to calculate the option price. Assuming the case where binomial trees are used there is the need to construct a tree for each of the two shares individually as all combinations of nodes from each tree must be computed at every time step. Assuming a tree depth of N, our trinomial tree requires $\frac{N+3}{3}\frac{1}{2}(N+1)(N+2)$ nodes and a recombining binomial tree consists of $\frac{1}{2}(N+1)(N+2)$ nodes then we need $\frac{2N+3}{3}\frac{1}{2}(N+1)(N+2)$ nodes when using binomial trees for a trinomial problem. As an illustration, consider where $N = 768$. Using multiple binomial trees will require 151881345 nodes, whereas the multinomial tree approach requires 76088705 nodes (half as much).

IV. Architecture

The architecture (illustrated in Figure 2) which achieves pipelining and parallelism is described in this section. The parallelism is possible since the European option price allows us to compute the stock value (the root nodes of the multinomial tree) directly without storing the intermediate values. This is the task (1) as described in section III-B (i.e. generation of the asset price vectors $Z(\nu), |\nu| = T$). For simplicity, we align the tree depth N with the expiration time T, i.e., the tree depth corresponds with the expiration time in our architecture.

Our implementation of the task (2) and task (3) from section III-B iteratively computes the option values $V(\nu)$. This is carried out by the pipelined stages of our architecture. Once the first three leaf nodes ($V(\nu)$) are computed, the pipeline stage can start. The remaining $V(\nu)$ is not required for initial intermediate European value calculations. The pipeline can be extended up to the limits of the hardware. The buffer is required to store intermediate option values until required - this organized as a FIFO.

All red arrows in Figure 2 are "channels" used for passing data between kernels which are analogous to "pipes" in the OpenCL standard. While "pipes" are a part of the OpenCL standard, "channels" are an extension of pipes and are specific to the Intel implementation of the OpenCL standard and therefore are not useable on platforms that do not support the Intel OpenCL SDK.

The required OpenCL kernels are presented in Table I and are described in detail in subsection IV-D. The producer and consumer kernels are not shown in Figure 2. However, these are simply used to interface with the host code running on CPU.

A. Computation of the Direction Vectors and Risk-Neutral Probability Vector

Recall the parameters q, d_1 and d_2 from the Binomial Option Price Model. Conceptually, for the multinomial case, we also have these parameters; however, they do not describe an up and down movement and the probability of those movements in the same fashion as the binomial case. Instead,

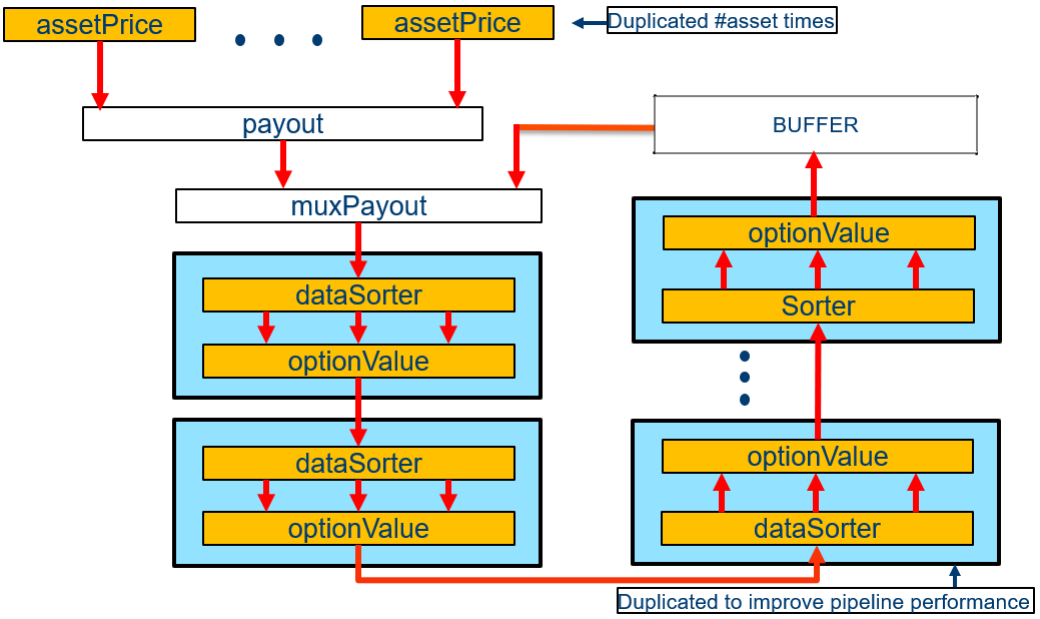

Fig. 2. The block diagram of the proposed system architecture

they need to describe the probabilities of these movements for both assets in a multidimensional movement. Similarly, to the binomial case, however, we can compute this matrix and vector once, in advance of running the computation of the final option value. The computation of the direction matrix d and probability vector q is described in detail in [25] Section 3.3. Furthermore, the elements d_i, where $0 \leq i \leq k$ of d, are called *direction vectors* and the elements in these vectors are addressed as $d_{i,j}$, where $0 \leq j \leq k$.

B. Parallel Computation of Leaf Node Asset Prices

Being able to introduce parallelism into our architecture is only possible because for European options, there is no possibility of an early exercise. Therefore, the intermediate values of the multinomial process are not required. The *assetPrice* computes $Z(\nu)$. This kernel is instantiated for each number of underlying assets.

C. Pipelined Computation of Final Option Value

In Figure 2 we make use of two kernels to compute each node value on the return traversal of the tree, *dataSorter* and *optionValue*. On the return traversal, we need to compute $V(v)$ for decreasing values of $|v|$ (equation 1). As we can see, there is one input to the *dataSorter* kernels, which is the option value from the previous level of the tree. We need three such values to compute the value of the next level, and the *dataSorter* has knowledge of the correct order, i.e. monomial ordering [26], that the previous values must be arranged into. The size of the pipeline has a significant impact on performance and the *dataSorter*/*optionValue* block can be duplicated as many times as the hardware can support.

Kernel	Description
producer	Takes values from host kernel
assetPrice	Asset price calculation $Z(\nu)$ (duplicated, running in parallel)
optionValue	European Value Calculation $V(\nu)$
payout	Computes payout $F(\nu)$
muxPayout	Maintains order of payout values
muxTreeDepth	Maintains tree depth value
dataSorter	Reads values from previous calculations and determines correct usage for value
returnBuffer	Used to buffer intermediate values
consumer	Consumer of final value calculation

TABLE I
OPENCL KERNELS

D. OpenCL Kernels

Table I gives a brief overview of the kernels required to compute the option value, and we describe those kernels in more detail in this section.

1) producer: The *producer* kernel acts as both an interface between the host system and the FPGA and distributes the inputs to the appropriate channels. The inputs provided by the host system to the device are described in Table II.

2) assetPrice: The value (or price) of the underlying assets at each node can be computed directly without constructing the entire tree. This kernel computes $Z(\nu), |\nu| = N$ which is the price of an asset at final time T using the precomputed values in the *end_node_values* and *init_step* vectors. For European options this approach is suitable as the storage of the

Kernel	Description
N	Tree depth
max_min	Whether to call/put on max or min (boolean value)
end_node_values	defined as a 2-element vector containing $\left\{(d_{0,0})^N Z_0, (d_{1,0})^N Z_1\right\}$ where the two assets are multiplied by the first two elements in the direction vector to the power of N.
init_step	defined as a 2x2 element matrix containing $\left\{\frac{d_{1,2}}{d_{1,1}}, \frac{d_{1,1}}{d_{1,0}}\right\}, \left\{\frac{d_{0,2}}{d_{0,1}}, \frac{d_{0,1}}{d_{0,0}}\right\}$
mult_coeff	3-element multiplicative coefficient vector containing $(1/R) * q_2$,$(1/R) * q_1$,$(1/R) * q_0$ where R is defined as $e^{r*(1/N)}$ (r is the risk neutral rate)
share_coeff	2-element share coefficient vector containing $\left\{\frac{1}{d_{1,1}}, \frac{1}{d_{0,1}}\right\}$
strike_vector	2-element vector containing the strike prices K.

TABLE II
OPENCL KERNEL INPUTS

intermediate asset prices are not necessary as the possibility of early exercising of the option is not possible for the European case however this direct computation is not possible if we were interested in American options. The asset price calculation uses the *end_node_values* vector and the *init_step* matrix to compute the asset prices $Z(\nu)$.

3) optionValue: This kernel is used to determine the option value $V(\nu)$ at a given node ν by implementing Equation 1. We leverage the *mult_coeff* (containing $\frac{1}{R} * q$) to compute the option value in conjunction with the payout values passed into the kernel.

4) payout: Computes the payout which is defined as

$$F(N) = \begin{cases} max(K - Z_N, 0), & \text{if } European\ option \\ max(Z_N - K, 0), & \text{otherwise} \end{cases} \quad (2)$$

5) muxPayout: Reads payout initially from the asset price computations until all asset price computations are consumed. After this, it only reads values from the option value pipeline.

6) muxTreeDepth: This kernel acts as a *"control"* kernel maintaining the current tree depth during the reverse traversal of the value tree. Once the traversal reaches the root node, this kernel signals the other running kernels to complete.

7) dataSorter: The dataSorter kernel arranges the outputs of each block into the correct order for the optionValue computation using a reverse monomial ordering scheme. This order is optimized such that values that are required for multiple node computations are rearranged for the option value computation just in time.

8) returnBuffer: This buffer kernel is required if the depth of the tree is greater than the number of computing blocks supported by the FPGA. The maximum size of this buffer is defined at compile time with the knowledge of the number of computing blocks.

Tree Depth	Time (usecs) X50	Time (usecs) X25	Time (usecs) X1	Ratio X1/X50	Ratio X1/X25
100	160	190	1220	7.62	6.42
200	580	617	8347	14.39	13.52
300	1441	1639	27296	18.94	16.65
400	2862	3430	63928	22.36	18.65
500	4985	6225	124092	24.89	19.93

TABLE III
LATENCY PERFORMANCE AND PIPELINING IMPACT

	Ours	Tavakkoli [27]	Wynnyk [28]	Jin [6]
Tree Depth	96	96	96	96
Precision	Double	Fix 16.16	Double	Fix 16.16
FPGA	Stratix 10 s10gx	Virtex-4 xc4vsx55	Stratix III ep3se260	Virtex-4 xc4vsx55
Clock (MHz)	320	224	150	82.7
LUTs RAMs DSPs	18% logic 700 (5%) 158 (2%)	85% logic 401 (78%) 11 (3%)	10% logic 84% bits 148 (19%)	22% logic 252 112 (22%)
Latency usecs	133	162.5 (estimated)	507 (estimated)	21970 (estimated)

TABLE IV
PERFORMANCE-AREA RESULTS

9) consumer: When the final option value is computed, this kernel passes the value from the FPGA to the host.

V. RESULTS

The device used for this experiment was the Intel Stratix 10 GX. The Intel FPGA SDK for OpenCL supplied (version 19.2) with the Stratix 10 supports the OpenCL 1.0 Standard. The host machine was a Dell Precision 7810. We measured the latency of all the running kernels using the OpenCL event profiling information. The pipeline was implemented using 64-bit floating-point precision (double data type).

The pipelining of the *dataSorter* and *optionValue* block is significant regarding the overall latency. We measured three different levels of pipeline sizes: using only one *dataSorter*/*optionValue* block, using twenty-five *dataSorter*/*optionValue* blocks and using fifty *dataSorter*/*optionValue* blocks. In Table III, we present these latencies. The ratio of depth to pipeline size illustrates the benefit of the larger pipeline when computing for deeper trees and the X50 pipeline helps reduce the latency by a factor of 25 when computing two-asset option values for depth 500.

Also noteworthy is the system utilization which was only 18% of flip-flops, 5% of RAMs, and 2% of DSPs for the 50x deep pipeline design and requires an estimated 39.3 Watts (using maximum power characteristics).

The performance demonstrated by our approach when compared to the implementation by Tavakkoli [27] (which is implemented using systolic arrays), Wynnyk [28] (which uses a pipelined architecture, and Jin [6] (which also implements a pipelined architecture using "HyperStreams") as illustrated in Table IV shows that using a single option pricer to achieve the same goal has a higher latency when we consider that using a binomial tree for each share individually requires significantly more computation effort. Note, the estimated latencies of the alternative implementations are solely based on the number of nodes required however they do not account for the complexity of the recombining operation, therefore it is expected the actual latencies will be higher in practice. Furthermore, our approach uses greater precision when compared to fixed precision.

VI. Conclusion

Two-asset option pricing based on Pascal's simplex is an efficient method for computing the fair price of an option based on two underlying assets. Taking advantage of the pipelining and parallel aspects of the computation lends itself to hardware acceleration via FPGA. In this paper we present such an acceleration architecture implemented using the OpenCL framework. Our implementation takes advantage of the channel's extension provided by the Intel FPGA SDK. This approach allowed us to achieve a performance of 200 two-option value calculations per second for a time step of 500 steps.

The underlying theory that our architecture is based on can be extended to greater than two underlying assets. Future work in this area will focus on this extension to where the architecture can support computation using an arbitrary number of assets.

VII. Acknowledgements

This paper has been supported in part by Intel Programmable Solutions Group, Science Foundation Ireland under grant 07/MI/008, and the SFI INSIGHT Centre for Data Analytics.

References

[1] J. Lambert, S. Lee, J. S. Vetter, and A. Malony, "In-depth optimization with the OpenACC-to-FPGA framework on an Arria 10 FPGA," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2020, pp. 460–470.

[2] G. P. M. Virgilio, "When spread bites fast–volatility and wide bid-ask spread in a mixed high-frequency and low-frequency environment," *Research in International Business and Finance*, vol. 51, p. 101066, 2020.

[3] C. de Schryver, P. Torruella, and N. Wehn, "A multi-level Monte Carlo FPGA accelerator for option pricing in the Heston model," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 248–253.

[4] P. Sundararajan, "High-performance computing using FPGAs," *Xilinx White Paper: FPGAs*, pp. 1–15, 2010.

[5] Q. Jin, D. B. Thomas, W. Luk, and B. Cope, "Exploring reconfigurable architectures for tree-based option pricing models," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 2, no. 4, p. 21, 2009.

[6] Q. Jin, D. B. Thomas, and W. Luk, "Exploring reconfigurable architectures for explicit finite difference option pricing models," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. IEEE, 2009, pp. 73–78.

[7] V. M. Morales, P. Horrein, A. Baghdadi, E. Hochapfel, and S. Vaton, "Energy-efficient FPGA implementation for binomial option pricing using OpenCL," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[8] K. J. Leslie and M. P. Michaels, "The real power of real options," *The McKinsey Quarterly*, no. 3, pp. 4–23, 1997.

[9] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973.

[10] J. C. Cox, S. A. Ross, and M. Rubinstein, "Option pricing: A simplified approach," *Journal of financial Economics*, vol. 7, no. 3, pp. 229–263, 1979.

[11] B. Zhang and C. W. Oosterlee, "Acceleration of option pricing technique on graphics processing units," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 9, pp. 1626–1639, 2014.

[12] V. M. Morales, P.-H. Horrein, A. Baghdadi, E. Hochapfel, and S. Vaton, "Energy-efficient FPGA implementation for binomial option pricing using opencl," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[13] R. Sridharan, G. Cooke, K. Hill, H. Lam, and A. George, "FPGA-based reconfigurable computing for pricing multi-asset barrier options," in *2012 Symposium on Application Accelerators in High Performance Computing*. IEEE, 2012, pp. 34–43.

[14] S. Fadugba, C. Nwozo, J. Okunlola, O. Adeyemo, and A. Ajayi, "On the accuracy of binomial model and Monte Carlo method for pricing European options," *International Journal of Mathematics and Statistics Studies, European Centre for Research Training and Development*, vol. 1, pp. 38–54, 2013.

[15] X. Tian, K. Benkrid, and X. Gu, "High performance Monte-Carlo based option pricing on FPGAs." *Engineering Letters*, vol. 16, no. 3, 2008.

[16] Q. Jin, W. Luk, and D. B. Thomas, "On comparing financial option price solvers on FPGA," in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*. IEEE, 2011, pp. 89–92.

[17] A. H. T. Tse, D. B. Thomas, and W. Luk, "Accelerating quadrature methods for option valuation," in *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, 2009, pp. 29–36.

[18] H. Anson, D. B. Thomas, and W. Luk, "Option pricing with multi-dimensional quadrature architectures," in *2009 International Conference on Field-Programmable Technology*. IEEE, 2009, pp. 427–430.

[19] N. Zhang, E. Lim, K. L. Man, and C.-U. Lei, "Cpu-GPU hybrid parallel binomial american option pricing," *Lecture Notes in Engineering and Computer Science*, 2012.

[20] X. Tian and K. Benkrid, "High-performance quasi-Monte Carlo financial simulation: FPGA vs. GPP vs. GPU," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 4, pp. 1–22, 2010.

[21] U. I. Minhas, R. Woods, and G. Karakonstantis, "Exploring functional acceleration of OpenCL on FPGAs and GPUs through platform-independent optimizations," in *International Symposium on Applied Reconfigurable Computing*. Springer, 2018, pp. 551–563.

[22] B. Betkaoui, D. B. Thomas, and W. Luk, "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 94–101.

[23] R. A. Brealey, S. C. Myers, F. Allen, and P. Mohanty, *Principles of corporate finance*. Tata McGraw-Hill Education, 2012.

[24] H. U. Gerber, "Mathematical fun with the compound binomial process," *ASTIN Bulletin: The Journal of the IAA*, vol. 18, no. 2, pp. 161–168, 1988.

[25] D. Sierag and B. Hanzon, "Pricing derivatives on multiple assets: recombining multinomial trees based on Pascal's simplex," *Annals of Operations Research*, pp. 1–27, 2017.

[26] D. Cox, J. Little, and D. O'Shea, "Ideals, varieties, algorithms: An introduction to computational algebraic geometry and commutative algebra, utm," 1992.

[27] A. Tavakkoli and D. B. Thomas, "Low-latency option pricing using systolic binomial trees," in *2014 International Conference on Field-Programmable Technology (FPT)*, 2014, pp. 44–51.

[28] C. Wynnyk and M. Magdon-Ismail, "Pricing the american option using reconfigurable hardware," in *2009 International Conference on Computational Science and Engineering*, vol. 2. IEEE, 2009, pp. 532–536.