On Comparing Financial Option Price Solvers on FPGA

Qiwei Jin and Wayne Luk
Department of Computing
Imperial College London
{qj04,wl}@doc.ic.ac.uk

David B. Thomas
Department of Electrical and Electronic Engineering
Imperial College London
d.thomas1@imperial.ac.uk

***Abstract*—A number of different numerical methods for accelerating financial option pricing using FPGAs have recently been investigated, such as Monte-Carlo, finite-difference, quadrature, and binomial trees. However, these papers only compare acceleration of each method against the same method in software, and do not consider a more important practical question, which is to identify the method that provides the best FPGA performance for a given option pricing application, regardless of raw speed-up over software. This paper proposes a framework for comparing the performance of numerical option pricing methods using FPGAs, taking into account both speed (time to solution) and accuracy (quality of solution), and examines how the speed-accuracy trade-off curve varies for each method. We apply the framework to European and American option pricing problems using Virtex-4 parts, and show that the quadrature solver converges fastest for both European and American options, and is also the most accurate in terms of root mean squared error for European options. However, when very accurate American results are needed the finite-difference solver is the most efficient method. Our results also show that the Monte-Carlo solver is at least 100 times less accurate in log scale than those based on other pricing methodologies; this drawback outweighs its benefit of having large raw speed-ups found in previous papers.**

***Keywords*-Finance, Tree, Quadrature, Finite Difference, Monte Carlo, Black Scholes, FPGA, European Option, American Option, Metrics**

I. Introduction

In recent years the application of FPGAs to financial computing has been investigated. For example, recent FPGA option pricing research includes: tree based solvers for American options with 30 times speed up [1]; a finite difference solver for European options with 27 times speed up [2]; a quadrature (numerical integration) solver for European options with 32 times speed up [3]; a stream-oriented Monte-Carlo accelerator for European options with 41 times speed up [4] and a Monte-Carlo simulator for American options using least squared method with 20 times speed up [5]. This approach is now being deployed commercially: for example by using a bespoke FPGA platform and sophisticated synthesis tool chain, an industrial FPGA provided 30 times acceleration over an eight-core Intel CPU [6].

Though it is easy to measure the speed-up of a particular FPGA implementation versus a software implementation of the same algorithm, the existing research does not allow us to compare between two FPGA methods for solving the same problem. Imagine we have two FPGA solvers for a given problem, based on two different mathematical methods and implemented on two different FPGAs; one solver is less accurate but consumes 5% of the total resources on FPGA1 and runs at a 20% higher clock rate than the other solver; the other core is more accurate but consumes 50% of the available resources on FPGA2 and runs at a lower clock rate. Current research into the FPGA acceleration does not give us enough information to determine which method is better, as all that is reported is raw performance, and accuracy is usually not considered in a directly comparable way. If such comparison can be made easy, it will help end users such as investment banks to compare the methods examined by the FPGA research community.

To make a fair comparison, one needs to consider both speed (hardware area, clock rate and intrinsic algorithmic complexity), and accuracy (quality of solution plus convergence with time). This paper presents a novel framework to compare between different iterative FPGA solvers in terms of speed and accuracy, which can be used to compare existing implementations, and predict the accuracy and execution time of methods on other reconfigurable device.

The main contributions of this paper are:

- A methodology for comparing FPGA-based numerical option-pricing applications by considered both speed and accuracy of each method (Section III);
- a comparison of existing FPGA option price solvers using the proposed framework. Tree, finite difference, quadrature and Monte Carlo methods are compared for both European options and American options (Section IV).

II. Financial Option Pricing Methods

There are two main types of options (call and put). For example, a put option is a contract that gives party A the right to sell some asset S to party B at a fixed price K (called the strike price). A number of numerical methods can be used for many types of options, with the most common being trees, quadrature (QUAD), finite difference and Monte Carlo (MC). Within each method there are further choices (for example explicit vs implicit finite difference), but in this paper we choose the main methods as discussed within the FPGA literature.

978-0-7695-4301-7/11 $26.00 © 2011 IEEE
DOI 10.1109/FCCM.2011.30

All these methods can be applied to a huge variety of financial models, but for the purposes of this discussion we will consider models derived from the Black-Scholes method, where underlying assets follow geometric Brownian motion [7]. If the option is a European option, the Black Scholes formula [7] can calculate the result analytically, however for American options there is no closed-form solution. In this paper the Black Scholes formula price is used as reference to other solver prices for European options, which provides a controlled environment to verify the correctness of our proposed methodology. We now introduce the option pricing methods being compared in this paper.

The **binomial model** works by discretising both time and the price of underlying asset S, and mapping both onto a binary tree [8]. The **trinomial model** is similar to the binomial model, except prices can now go up, down, or stay the same at each time step, where depth of the tree means time [1]. The **quadrature method** (QUAD) overcomes the "distribution error" and "non-linearity error" introduced by tree based and finite difference methods, by using numerical integration at each time step [9]. The **explicit finite difference method** (EFD) solves the Black Scholes PDE by discretising both time and the price of the underlying asset S, and mapping both onto a two-dimensional grid [2]. The **Monte Carlo method** (MC) is usually only used when the problem cannot be handled using non-stochastic numerical methods; it uses a large number of randomly generated simulations to estimate the expected option price [10].

III. Comparison Methodology

A common feature of all these numerical methods, there is a tradeoff between speed to solution and accuracy of solution. The key metrics we can use to evaluate each FPGA solver are: a) **Execution time** (e.g. in seconds); b) **Accuracy** (e.g. relative error to the reference); c) **Resource consumption** (e.g. in LUTs and DSPs). We now discuss the application of the metrics to finance option pricing models.

A. Measuring Execution time

In order to compare each of the option price solvers proposed in Section II, we define a parameter n, the Asset Price Observation (APO) points at option expiry, which is a function of the number of time steps m in the model. The definition of n and the problem size in terms of the total number of iterations for each model are listed in Table I.

Since all the reported FPGA implementations are fully pipelined, one intermediate result is produced per clock cycle. This allows us to link clock rate to execution time, as shown in Equation 1:

$$ExecutionTime = \frac{x}{C \times N} + o \times N \quad (1)$$

where C is the clock frequency of a solver core implemented on the FPGA, N is the number of replicates of the central kernel and o is the total communication overhead.

Table I
The meaning of asset price observation (APO) point n in each pricing method and how to calculate problem size x

Solver	Meaning of n	Num Iterations x
Binomial	Number of leaves in tree	$n \times (n+1)/2$
Trinomial	Number of leaves in tree	$(n+1)^2/4$
EFD	Number of discretised asset price points in the grid	$n^3 \times \sigma^2 \times T$
QUAD	Number of integration points	$n^2 \times m$
MC	Number of paths	$n \times m$

B. Measuring Accuracy

Due to the unpredictable nature of solution convergence, solver accuracy can not be measured accurately in a single run. We propose a Monte Carlo approach to measure the accuracy by running experiments multiple times using randomly generated parameters for the target solver. This minimises the possible biases from individual runs. The framework is comprised of the following components:

- A set of options O, a set of possible option input parameters P and a set of solvers S,
- a function $t(n, O, S) \rightarrow t$ which gives the run-time of the hardware (Equation 1),
- a function $a(n, O, S, P) \rightarrow v$ which gives the result using solver O with n APO points and option inputs P,
- a function $r(O) \rightarrow P$ which takes the type of the option as an input and outputs an option with randomly generated parameters,
- a reference (golden) solver $g(O, P) \rightarrow v$ where $g(O, P) = a(N_G, O, S_G, P)$. This is an analytical solver if one exists, or one of the existing solvers which is known to be stable with very large n.

The accuracy of a given solver is represented in terms of root mean squared error over a random sample of size b.

$$e(n, O, S) = \sqrt{\frac{1}{b}\sum_{i=1}^{b}\{a[n, O, S, r(O)] - g(O, P)\}^2} \quad (2)$$

C. Estimating Hardware Resource Utilisation

Hardware resource utilisation for the option solvers can not be measured accurately until the synthesis phase of the design flow. Rapid analysis methodology has been proposed for mapping arbitrary applications to targeted reconfigurable platforms [11], however this only allows us to explore design space in terms of low level hardware description language. To obtain a good approximation from a high level mathematical description we use the following methodology:

- generate a resource consumption table of all the arithmetical operators in use for the target reconfigurable device;
- calculate the arithmetical operators required per step in each solver based on the model's mathematical description;

Table II
SINGLE PRECISION OPTION PRICE SOLVERS: CLOCK FREQUENCY, NUMBER OF REPLICATIONS AND DEVICE MODEL.

Solver	European		American	
	Frequency MHz×Size	Device (XC4V)	Frequency MHz×Size	Device (XC4V)
Binomial	79×7	SX55	76×6 [1]	SX55
Trinomial	77×5	SX55	69×4 [1]	SX55
EFD	106×8 [2]	LX160	89×7	LX160
QUAD	100×3 [3]	LX160	90×2	LX160
MC	61×3 [4]	LX160	76×4 [5]	FX100

- apply optimisation to eliminate unnecessary arithmetical operators to minimise hardware resource consumption;
- calculate primitive hardware utilisation u in terms of

$$u = \sum_{i=1}^{n} num(i) \times util(i) \tag{3}$$

where $num(i) \rightarrow \mathbb{N}$ is a function to return the number of the operator used given the index of the operator; $util(i) \rightarrow \mathbb{N}$ is a function to return the resource consumption given the index;
- if the method is Monte Carlo based, use Equation 3 to calculate its resource consumption u_{mc} and add that to the original u;
- Address resources consumed by peripheral and glue logic for each solver core (f_1), and the percentage of total FPGA slice resource to handle device-host communication, necessary number format conversion and reasonable waste of resource (f_2), the estimated core duplications μ on a particular device can then be calculated:

$$\mu = \frac{(1 - f_2)L}{f_1 \times (u + u_{mc})} \tag{4}$$

where L is the total resource available on the reconfigurable device.

IV. RESULT

In this section we use our framework to compare existing option price solvers using the methodologies proposed in Section III. The accuracy comparison is done in double precision, although the actual FPGA implementations are in either single precision or fixed point numbers. This is to rule out possible biases introduced by none-IEEE compliant FPGA floating point library adopted in existing designs. The model parameters m and n are set for good efficiency and accuracy, the details of the setup is out of the scope of this paper and will be included in a future journal paper.

Table II shows the clock frequencies, the number of core duplications and the types of target devices for single precision solvers. Based on Table I and Table II , execution time against n is plotted in Figure 1. It is interesting to see that as n becomes larger, the execution time of the explicit finite difference solver surpasses that of the Monte Carlo solver. This means that the problem size of the explicit finite difference solver increases the fastest amongst all the solvers.

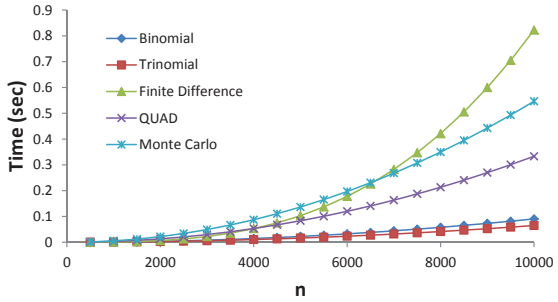

Figure 1. Execution time against asset price observation points n for European options, based on Table I and Table II

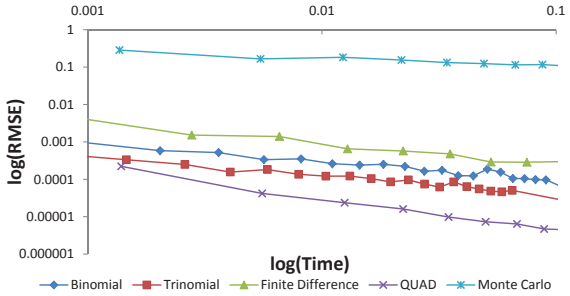

Figure 2. Root mean squared error against time for European option in log scale.

In our experiment the solver accuracy is measured by 10K trials of randomly generated option parameters for a range of APO points from 0.5K to 10K. The option parameters are generated by the following criteria: $S = [50, 100], K = S \pm [0, 10], V = [0.1, 0.3], R = [0.02, 0.2], T = 1.0$. Each pricer price is compared to a reference price considered to be true: we use the Black Scholes formula price for European options and binomial tree price with $n = 50K$ for American options. This is because the error bound for binomial tree at $n = 50K$ is negligible compared to the solver price.

Figure 2 depicts the root mean squared error (RMSE) against time (sec) for European option solvers. It can be seen that the quadrature solver gives the result closest to the reference and is the fastest; the binomial and trinomial solvers have a similar RMSE at all times and the explicit finite difference solver comes last. It can also be seen that the Monte Carlo solver has relatively large RMSE at the beginning and converges slowly with time, it will take more than 0.6 seconds before RMSE drops under 0.05; the convergence degradation is orders of magnitude higher than the speed up gain. It can be concluded that the FPGA Monte Carlo solver should only be used when no other solvers are available.

Figure 3 depicts RMSE against execution time (sec) for

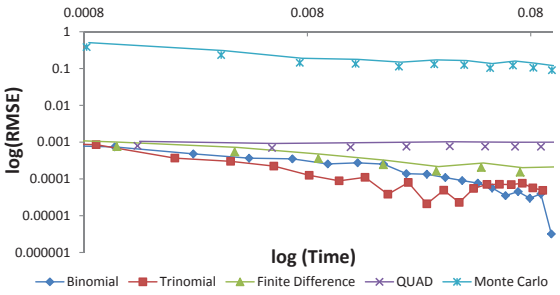

Figure 3. RMSE against time for American options in log scale.

all solvers. The figure shows that the quadrature result converges to a stable value under 0.01 second and the other result converge in just over 0.02 seconds. All the solvers can produce a good enough to use result within 2 micro seconds, with RMSE smaller than 0.001. The QUAD RMSE is a horizontal line since it converges to a value different from the binomial reference. This difference is model-dependent and hence cannot be improved by increasing the number of APO points. It can be seen that the RMSE of the result produced by Monte Carlo solver is still over 0.5 after 0.1 seconds, which is much larger than other solvers.

From the result obtained we are able to conclude that the quadrature method implemented on an XC4VLX160 device produces the most accurate result in terms of RMSE the fastest for European options and produces the most stable result over time for American options. The trinomial tree tends to be more accurate than binomial tree on a XC4VLX160 device for both European and American options, but they tend to converge to the same result when number of APO points gets larger. They have less convergence over time comparing to quadrature method, however, for European options, explicit finite difference solver provides a more accurate result than the tree based method when number of APO points reaches 10K. In practice the quadrature solver should be used if applicable. Otherwise the tree based solvers should be used if the result is time critical and the explicit finite difference method should be used if the result is accuracy critical. On the other hand, the Monte Carlo solvers for European and American options, implemented on a XC4VLX160 device and a XC4VFX100 device, demonstrate orders of magnitude less accurate result even with a much longer execution time. In addition, after a threshold the convergence of Monte Carlo methods becomes negligible, therefore for best result the Monte Carlo method should not be used if other methods are available, even if they are accelerated by FPGA. In other words, Monte Carlo methods should only be adopted as a last resort even though they achieve large speed-ups in hardware.

V. Conclusion and Future Work

This paper presents a novel framework to compare different iterative solvers for FPGA financial applications. Various mathematical option pricing models implemented on FPGAs are compared in terms of result accuracy and execution time, based on reported results. Our results show that the FPGA based Monte Carlo solver should only be used when there are no other solvers available; the degradation in convergence is orders of magnitude higher than the speed up gain, leading to a longer execution time for a desired accuracy.

Future work includes applying the framework to other types of option solvers such as barrier options, Asian options and those involving two underlying assets instead of one. We also plan to include power and energy consumption estimations for designs on different FPGAs.

Acknowledgement: The research leading to these results has received funding from J.P.Morgan, EPSRC, Alpha Data, Xilinx and the European Union Seventh Framework Programme under grant agreement number 248976 and 257906

References

[1] Q. Jin, D. B. Thomas, W. Luk, and B. Cope, "Exploring reconfigurable architectures for tree-based option pricing models," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, pp. 21:1–21:17, September 2009.

[2] Q. Jin, D. Thomas, and W. Luk, "Exploring reconfigurable architectures for explicit finite difference option pricing models," in *Int. Conf. on Field Programmable Logic and Applications*, 2009, pp. 73–78.

[3] A. H. Tse, D. B. Thomas, and W. Luk, "Accelerating quadrature methods for option valuation," in *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, 2009.

[4] G. Morris and M. Aubury, "Design space exploration of the European option benchmark using Hyperstreams," in *Proc. Int. Conf. on Field Programmable Logic and Applications*, 2007, pp. 5–10.

[5] X. Tian and K. Benkrid., "American option pricing on reconfigurable hardware using least-squares monte carlo method," in *Proc. Int. Conf. on Field-Programmable Technology*, 2009, pp. 263 –270.

[6] S. Weston, J. T. Marin, J. Spooner, O. Pell, and O. Mencer, "Accelerating the computation of portfolios of tranched credit derivatives," in *IEEE Workshop on High Performance Computational Finance*, 2010, pp. 1–8.

[7] J. Hull, *Options, Futures and Other Derivatives*, 6th ed. Prentice Hall, 2005.

[8] Q. Jin, D. B. Thomas, W. Luk, and B. Cope, "Exploring reconfigurable architectures for binomial-tree pricing models," in *Proc. Int. workshop on Applied Reconfigurable Computing*, 2008, pp. 245–255.

[9] A. D. Andricopoulos, M. Widdicks, P. W. Duck, and D. P. Newton, "Universal option valuation using quadrature methods," *Journal of Financial Economics*, vol. 67, no. 3, pp. 447–471, 2003.

[10] P. Glasserman, *Monte Carlo Methods in Financial Engineering*. Springer, 2003.

[11] C. Reardon, E. Grobelny, A. D. George, and G. Wang, "A simulation framework for rapid analysis of reconfigurable computing systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 3, pp. 25:1–25:29, November 2010.