A Multi-Level Monte Carlo FPGA Accelerator for Option Pricing in the Heston Model

Christian de Schryver, Pedro Torruella, Norbert Wehn
Microelectronic Systems Design Research Group
University of Kaiserslautern, Germany
{schryver, wehn}@eit.uni-kl.de

***Abstract*—The increasing demand for fast and accurate product pricing and risk computation together with high energy costs currently make finance and insurance institutes to rethink their IT infrastructure. Heterogeneous systems including specialized accelerator devices are a promising alternative to current CPU and GPU-clusters towards hardware accelerated computing. It has already been shown in previous work that complex state-of-the-art computations that have to be performed very frequently can be sped up by FPGA accelerators in a highly efficient way in this domain. A very common task is the pricing of credit derivatives, in particular options, under realistic market models. Monte Carlo methods are typically employed for complex or path dependent products. It has been shown that the multi-level Monte Carlo can provide a much better convergence behavior than standard single-level methods. In this work we present the first hardware architecture for pricing European barrier options in the Heston model based on the advanced multi-level Monte Carlo method. The presented architecture uses industry-standard AXI4-Stream flow control, is constructed in a modular way and can be extended to more products easily. We show that it computes around 100 millions of steps in a second with a total power consumption of 3.58 W on a Xilinx Virtex-6 FPGA.**

I. Introduction

Modern risk and asset management relies on state-of-the-art financial models and simulation techniques to compute meaningful values with sufficient accuracy in time. The introduction of recent regulation mechanisms in the finance and insurance sector like Basel III or Solvency II has again led to an enormous increase in the demand of permanently available computation power, since important risk numbers have to be evaluated very frequently. At the same time, energy efficiency is becoming more and more important due to the high energy costs. Combining algorithmic optimizations with efficient suitable execution platforms is the key to construct highly efficient hardware accelerators for financial applications. Pricing credit derivatives, in particular options, is one fundamental task in the risk assessment process. In this work we present the first FPGA architecture based on the advanced multi-level Monte Carlo method for European barrier option pricing in the state-of-the-art Heston model in detail.

Banks, financial and insurance companies - and in particular small and medium sized institutes - are forced to move into the high performance computing (HPC) domain nowadays to keep track with the market's and regulatory needs [1] [2]. At the same time, embedded design methodologies, for example the use of specialized accelerator cores and low-power system considerations, are emerging in the HPC sector of the finance domain.

978-3-9815370-0-0/DATE13/©2013 EDAA

Options became very popular at the trading desk after Black, Scholes, and Merton had published the famous *Black-Scholes model* in 1973 that allowed to compute fair option prices for the first time. The Black-Scholes model relies on some simplifications that do not reflect observable market characteristics in many cases [3], and has been extended by Heston in 1993 with a stochastic volatility [4]. Therefore in this paper we model the underlying market dynamics with the state-of-the-art Heston model that is widely accepted in the finance business nowadays [3]. Barrier options in the Heston model in general can only be valuated by using numerical methods [5].

GPUs have already emerged in these sectors over the last years, since they allow a good trade-off for software developers between achievable acceleration of time-critical functions and implementation efficiency [6] [7]. Dedicated hardware accelerators based on FPGAs again increase the possible speedup and energy saving potential by orders of magnitude. In previous work it has already been shown that they can save more than 90% of energy compared to GPUs [8] [7], by providing the same throughput at the same time.

However, to optimize the overall system performance, the algorithm's characteristics need to match the underlying execution platform. Monte Carlo methods are inherently parallel and therefore fit very well to parallel execution platforms, in particular FPGAs where many instances can work in parallel independently.

Although different methods like the finite difference method, binomial or trinomial trees, or the quadrature method can converge faster for specific products, Monte Carlo solvers tackle the widest application range and can be adjusted to many products. They are widely spread in the finance industry to solve high-order problems or - as in our case - path dependent products such as barrier options, where finite difference or tree methods can not be applied easily. The multi-level Monte Carlo method extends the standard single-level Monte Carlo method and allows to achieve more accurate results with the same number of samples compared to the single-level method [9]. To the best of our knowledge, no FPGA implementation based on the multi-level Monte Carlo method has been presented before.

In this work the single-level Monte Carlo architecture for barrier option pricing in the Heston model presented in 2011 [8] is enhanced to multi-level Monte Carlo. We show that a multi-level Monte Carlo architecture for asset path simulations in the Heston model only consumes 3.58 W on a Xilinx Virtex-6 FPGA. Our design achieves a throughput of approximately 100 millions of simulated time steps per second.

The main contributions of this paper are:

- We show the algorithmic advantages of the multi-level Monte Carlo for barrier option pricing in the Heston model compared to single-level Monte Carlo.
- We present the first hardware architecture based on the multi-level Monte Carlo method together with implementation details.
- We give detailed synthesis results, throughput and energy estimations for a Xilinx Virtex-6 FPGA.

II. Related Work

A lot of publications on accelerating option pricing in the Black-Scholes model exist today. They cover not only various option types, but also investigate a wide range of solver algorithms (e.g. Monte Carlo, finite difference, quadrature, trees) on different architectures like CPUs, ASIPs, GPUs and FPGAs.

The first papers investigating the performance of GPUs for option pricing in the multi-asset Heston model have been published by Bernemann et al. in 2010 [10]. They have shown for Monte Carlo simulations that a Nvidia Tesla C1060 GPU can outperform a dual socket Intel Xeon E5620@2.4GHz CPU by factors between 7.5 and 50 with respect to speed, depending on the number of underlyings. In 2011 they have enhanced their work by showing that using quasi-random Sobol sequences instead of pseudo random numbers on a GPU reduces the speedup factor over a CPU system by up to 30% [7].

The impact of different workload splits between CPU and GPU on acceleration Heston pricing has been investigated by Zhang and Oosterlee in 2010 [11]. They have shown for European options that the highest speedup can be achieved if most of the basic arithmetic operations on the high numbers of paths are performed directly on the GPU, avoiding the bottleneck of limited bandwidth between CPU and GPU.

The first FPGA accelerated option pricing solutions in the Heston model have been published in 2011.

Delivorias has compared speedups for Heston simulations on a CPU cluster, two GPUs and the MaxCloud FPGA cluster [6]. He has shown that 4 vectis dataflow engines on the Maxeler MaxCloud service outperform a GPU compute server with 2x Tesla M2090 by a factor of about 1.75 with respect to speed.

Sridharan et al. from the NSF Center for High-Performance Reconfigurable Computing have presented a modular FPGA accelerated system structure for multi-asset barrier option pricing [12]. They use employ Heston cores based on the full truncation Euler discretization scheme that we have already verified to perform best on the algorithmic level [9]. All in all, they show that an 48-FPGA platform can achieve a speedup of more than 7000 compared to an SSE optimized single-threaded C program.

Energy aspects have not been considered in any of these works.

In 2011, it has been shown that for single-level Monte Carlo simulations, a state-of-the-art Tesla C2050 GPU achieves a 5.5x speedup over an 8-core 3GHz Xeon CPU, by only consuming 30% of the energy [8]. Three instances of the Monte Carlo accelerator on one Virtex-5 FPGA running at 100 MHz can achieve the same throughput as the 8-core CPU by only consuming 4% of the energy per simulation.

The potential benefits of using multi-level Monte Carlo methods over single-level implementations for European barrier option pricing have been investigated in 2011 [9], where it has been shown that using multi-level Monte Carlo methods for the underlying path simulations can reduce the computational complexity by up to 40%.

III. Multi-Level Monte Carlo for Option Pricing in the Heston Model

A. Market Models for Option Pricing

In 1993, Steven Heston has generalized the Black-Scholes model with a stochastic volatility process in the so-called *Heston model* [4]. It consists of a system of stochastic differential equations, that model the dynamics of the option's underlying *asset price* S and its *volatility* V over the time t:

$$dS(t) = rS(t)dt + \sqrt{V(t)}S(t)dW^S(t) \quad (1)$$

$$dV(t) = \kappa(\theta - V(t))dt + \eta\sqrt{V(t)}dW^V(t) \quad (2)$$

The randomness of the market is given by the Brownian motions W^S and W^V that are correlated with a factor ρ. All other parameters further specify the setting of the financial market [4]. The Heston model is used to model the behavior of the option's underlying. The desired option price is derived from the discounted expected payoff at the time to maturity, taking into consideration upper and lower bound for the specific option type [13].

The Heston model in its original form is a time-continuous model. Although closed-form and semi-closed form for very special settings and option types exist, they are not sufficient to cover the necessary range of practical use cases [5]. Therefore numerical methods are mandatory to evaluate options in the Heston model in the general case.

The correlation between the price and the volatility process is a key feature of the Heston model to reflect realistic market scenarios. It can be maintained by applying the Euler-Maruyama method to the Heston model and discretizing the time into N equidistant steps [14]. For a process between 0 and the time T the discretization points are:

$$0 = t_0 < t_1 < t_2 < \cdots < t_N = T$$

For $k = 0, ..., N-1$, let $\Delta_k t := t_{k+1} - t_k$ and accordingly for the Brownian motions $\Delta_k W^S := W^S_{t_{k+1}} - W^S_{t_k}$ and $\Delta_k W^V :=$

$W^V_{t_{k+1}} - W^V_{t_k}$. By assuming an equidistant grid over t, $t_k = k\frac{T}{n}$ for $k = 0, ..., N$.

With this notation and the starting conditions $\overline{V}_0 = V_0$ and $\overline{S}_0 = S_0$, applying the Euler scheme to Equation 1 and Equation 2 lead to the discrete version given by Equation 3 and Equation 4.

$$\overline{S}_{t_{i+1}} = \overline{S}_{t_i} + r\overline{S}_{t_i}\Delta_k t + \overline{S}_{t_i}\sqrt{\overline{V}_{t_i}}\Delta_k W^S \tag{3}$$

$$\overline{V}_{t_{i+1}} = \overline{V}_{t_i} + \kappa(\theta - \overline{V}_{t_i}) + \eta\sqrt{\overline{V}_{t_i}}\Delta_k W^V \tag{4}$$

Due to numerical reasons, negative values can occur when simulating the discrete volatility process. Several schemes for dealing with negative volatility values have been proposed up to now, for example absorption, reflection, or full truncation [9]. Numerical results indicate that full truncation performs best on average in single- and multi-level simulations [9], together with continuity correction of the barriers. The step generator architecture shown in Section IV-D is based on this algorithmic selection.

B. Monte Carlo Path Simulations

1) Single-level Characteristics: Monte Carlo methods rely on the law of large numbers that states that an expected value $\mathbb{E}(X)$ of a random variable X can be approximated by averaging over a large number of sample drawn independently from the distribution of X. For that reason Monte Carlo methods show an intrinsic parallelism and are therefore very interesting for the construction of parallel hardware accelerators.

Monte Carlo methods show a very robust behavior for a wide range of applications. However, they have a rather slow convergence rate: the standard deviation of the error in the crude Monte Carlo method is of order $O(1/\sqrt{N})$ with N being the number of samples [15]. That means: in order to increase the mean accuracy of the result by one digit, 100 times more experiments have to be executed.

Variance reduction techniques are basic methods to improve the accuracy in Monte Carlo simulations for fixed sample sizes. In our work we have used the *antithetic variate*: with one random variable Z, we simulate two paths, one with Z and one with $-Z$. It has been shown that antithetic variance reduction can speed up a Monte Carlo option pricing algorithm on CPU and GPU by around 50% [9].

Using Monte Carlo for path simulations always introduces a simulation error. The overall error (here the mean squared error (MSE)) can be partitioned into two parts [14]:

$$MSE = Variance + (Bias)^2 \tag{5}$$

The variance results from the Brownian motions in the SDEs and is also called *stochastic error*. It becomes smaller with increasing numbers of drawn samples in the Monte Carlo setting. The bias is a *systematic error* introduced by the discretization and can be reduced by simulating finer discretization steps.

The price of the option finally is the discounted expectation value of the Monte Carlo simulated assets in the Heston model under consideration of the option's payoff profile and potential barriers [13].

2) The Multi-Level Monte Carlo Method: The multi-level Monte Carlo method as presented by Heinrich in 2001 [16] and refined by Giles in 2008 [17] can be seen as an extension of the statistical Romberg method from Kebaier [18]. Like the statistical Romberg method, it uses a telescopic sum for the expectation values and allows to reduce the order of the computational complexity from ϵ^{-3} to $\epsilon^{-2} \cdot log(\epsilon)^2$ with ϵ being the bound of the root mean squared error $rMSE = \sqrt{MSE}$ [14].

In contrast to the single-level method, where the number of simulations and time steps is given by the user, the multi-level Monte Carlo method uses an heuristic scheme to determine the amount of simulations needed to achieve a certain accuracy of the result. This is achieved by adapting the number of time steps for every simulation *level*. It is important to emphasize the difference between the algorithm used in this work (that varies the discretization level over the simulated time), and the approach presented by Jin et al. that is called *multi-level* as well, but works with different computational precisions [19].

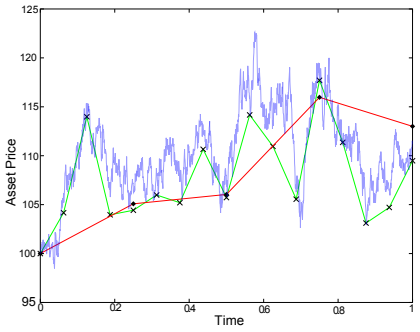

Fig. 1. A continuous asset price and two different discretizations using 4 and 16 discretization steps.

Figure 1 shows a discrete asset path simulation on two different levels, that means with 4 and 16 discretization steps (corresponding level 1 and level 2 in this example). The multi-level constant M is defined as the ratio $M = \frac{number\ of\ fine\ steps}{number\ of\ coarse\ steps}$, in this case $M = 4$. The key is that both levels are simulated simultaneously with *the same* Brownian motions. For the coarse steps, the Brownian increments of the finer level are accumulated to simulate one bigger step. It can be seen that the asset prices of the coarse step simulations are not matching the fine step prices. The variance of the difference for fine and coarse step simulations is evaluated by the multi-level controller to adjust the number of simulations in the next simulation run [17]. The stopping criteria for the simulation is the variance of the difference between fine and coarse step results that has to be below a predefined threshold.

Our proposed architecture strongly corresponds to the algorithmic partitioning by splitting the multi-level control from the path simulator engine as shown in Figure 2.

IV. Hardware Architecture

In Section III-B we have shown why multi-level Monte Carlo simulations have a huge potential for credit derivative pricing. In this section we derive an efficient hardware architecture for multi-level Monte Carlo simulations and present comprehensive architectural details.

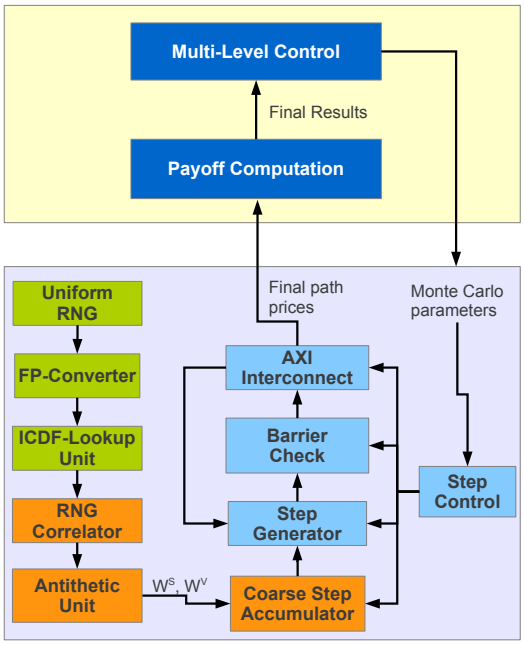

Fig. 2. Structure of the multi-level Monte Carlo path simulator.

Figure 2 shows the overall structure of the proposed multi-level Monte Carlo accelerator engine. The architecture can be split into two paths: the Monte Carlo path simulator at the bottom, and the payoff and multi-level control at the top. The payoff computation and the multi-level control are dedicated to be executed on an embedded CPU (for example the ARM Cortex-A9 on the recent Xilinx Zynq EPP) and are therefore not further discussed. It is important to note that the main computational load is located in the path simulator presented in this chapter, and that the control flow introduced by the multi-level control is not a critical point in the overall simulation.

The path simulator is in the lower area of Figure 2 is pipelined with a cyclic inner data path that allows the direct feedback of data packets for the iterations of the Monte Carlo simulation. It consist again of several parts: the modules that feed random numbers into the data path (green and orange), and the application's data path based upon different cores that implement a part of the algorithm (blue). All blocks in the data path are connected with AXI4-Stream flow control in order to provide automatic stalling propagation and to reduce the necessary overall control overhead.

First investigations have shown that 32 bit single precision floating point arithmetics provide enough resolution for our application. Therefore from the input of the RNG correlator on, we this data type throughout the design. The floating point units have been created with the Xilinx CoreGen tool with maximum pipelining based on the Xilinx Floating Point IP-Core Library 6.0.

A. AXI4-Stream Based Data Paths

It has already been shown that the hardware needed for this kind of simulations can result in data paths with over 40 stages [8]. Furthermore, the multi-level Monte Carlo method increases the necessary control flow not only in the multi-level controller, but also in the path simulator: the Brownian increments for the coarse step simulations have to be accumulated, and the path simulator has to maintain the correlation between the coarse step paths and the fine path equivalent.

In spite of the increased control flow, we still tackle a strongly data driven application here. Therefore we have decided to equip every core with the recent AXI4-Stream industry interface standard. This approach avoids large overall control units and therefore simplifies the validation to a large extent, since every core can be tested independently out of the system context by employing reusable AXI4-Stream data feeder, receiver and evaluator modules. If a component is approved, it can be integrated in the data path seamlessly.

B. Random Number Generator and Inner Datapath

As shown in Figure 2, the random number generation and preparation can be separated from the cyclic inner data path. For the random number generation (RNG), we use a MT19937 Mersenne Twister together with an ICDF based converter unit that generates normally distributed random numbers [20] (the green components).

The orange blocks prepare the Gaussian random numbers for the use in the Heston path simulation. The RNG correlator unit outputs a pair of random numbers with the correlation ρ (details in Section IV-C). This pair is consumed by the antithetic unit that mirrors the input at the output for one clock cycle, and outputs the negative input in the next clock cycle corresponding to Z and $-Z$ in Section III-B1. The correlator in combination with the antithetic unit allows to provide one pair of random numbers in every clock cycle with only one random number generator.

The coarse step accumulator generates the Brownian increments for the coarse step simulations by accumulating all fine steps of the corresponding fine paths. Since we simulate path parallel, that means the next time steps over all paths in the queue, the accumulator internally stores the coarse step values for all fine paths that are currently processed and outputs the coarse step increments for a coarse step run after every M fine step runs (see Section III-B2).

The cyclic inner data path (blue components in Figure 2) is formed by the step generator (details in Section IV-D), the barrier checker and the AXI interconnect. It uses a *packet* concept to store the current price, volatility, and status of barrier hits (two one-bit flags) for the current time step of every path. During initialization, the data path is filled with packets containing start price, start volatility, and cleared barrier flags. All components are configured with the current option, market,

and simulation parameters over the AXI interconnect that also returns the final prices for all paths. The barrier checker compares the current price against the given barrier values and sets the barrier hit flags if a barrier hit has been detected.

For the internal data handling managed by the step control and in order to keep the system as simple as possible, we make the assumption that all the data in the pipeline is valid and ordered at any point in time. This allows to control the whole data path by using a single counter, without any other complementary logic. After the last steps have been computed, the data is streamed out over the AXI interconnect in order.

C. Correlator Core

The correlator introduces the statistically correlation ρ between the two Brownian motions W^S and W^V in Equation 1 and Equation 2 out of two independent random numbers Z_1 and Z_2 by computing [14]:

$$W^S = Z_1 \quad (6)$$

$$W^V = \rho \cdot Z_1 + \sqrt{1-\rho^2} \cdot Z_2 \quad (7)$$

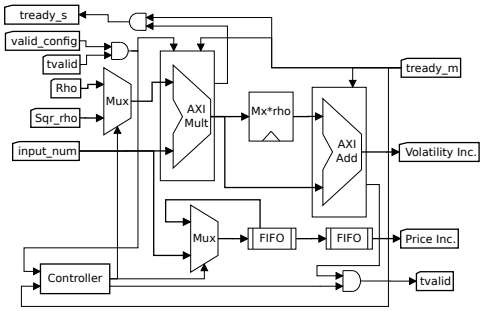

Fig. 3. Detailed block diagram of the correlator core.

Figure 3 shows the detailed structure of the correlator core. It is an exemplary showcase for the other components. Since Z_1 is transmitted across the core and not processed on its pipeline, two FIFOs maintain the integrity of the packets at the output. This way, when a core is stalled the external data will move at the same phase. We did not implement skid registers between the units, because we routed the ready signal from the master interface to all of the cores, making the whole system to respond at once to the ready signal.

D. Step Generator Core

The step generator unit computes the values for price and volatility of the next time step on a path due to Equation 3 and Equation 4. Figure 4 shows the data flow graph of the step generating function. It consists of a large number of arithmetic units, but shows a regular structure. For that reason we have used the Xilinx AutoESL high-level synthesis tool (the Vivado HLS predecessor) to generate the step generator architecture. It invokes the Xilinx CoreGen Tool for generating floating point cores.

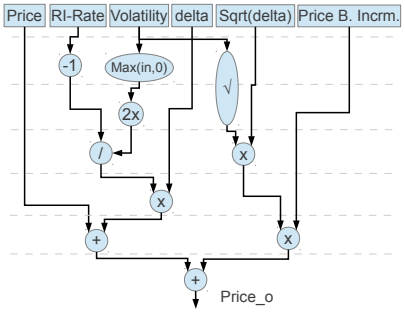

Fig. 4. Data flow diagram of the option price step generator function.

V. Results

We have synthesized our design with the Xilinx ISE 14.2 suite. The target device was the Virtex-6 XC6VLX240T-1FFG1156. The optimization goal was set to speed with extra effort set to high in the MAP and PAR stages. Table V shows the results count after place & route.

The proportion of the resource usage per core is illustrated in Figure 5. We can see that the majority of the resources are occupied by the step generator that contains the vast majority of floating point cores and pipeline registers.

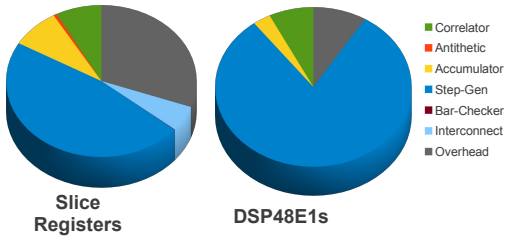

Fig. 5. Resource consumption per core inside the path simulator.

Regarding power consumption, the Xilinx XPower Estimator predicts a chip-only power consumption (including dynamic power) of 3.58 W for our design, being able to operate on ambient temperature up to 80 °C with a standard setup.

The data path of the Monte Carlo path simulation accelerator is able to run up to 120 MHz (timing constraint set to 120 MHz). This relatively low frequency results from a critical path through a DSP48E1 slice in the step generator, that has been introduced by the high-level synthesis. With 120 MHz system clock frequency, our design provides a throughput of around 120 million packets per second.

The highest utilization rate for the selected FGPA device is 8% for the DSP48E1 slices. We have been able to synthesize eleven instances of the presented system with the available resources of one XC6VLX240T-1FFG1156 device, what results in a total throughput of around 1.1 billions of time step computations per second.

A fair comparison with our base architecture from 2011 [8] is not feasible, since the target devices differ and the presented architecture provides a much higher functional range.

Functional Unit	Correlator	Antithetic	Accumulator	Step Generator	Barrier Checker	AXI Interconnect	Data Path total	Data Path total
Slice-Registers	934	66	996	5644	7	652	11949	3%
Registers-as FF	934	66	996	5638	7	652	11943	
Slice LUT's	724	67	520	7178	95	415	10288	6%
LUT - FF pairs	812	67	1057	8903	96	681	14842	
DSP48E1's	5	0	2	55	0	0	68	8%
BRAMs (36E1)	0	0	0	0	0	64	64	6%
Max Freq.[MHz]	294.2	984.25	232.5	101.9	144.19	211.01	120.00	

TABLE I
RESOURCE UTILIZATION COUNT FOR EACH CORE AND THE COMPLETE PATH SIMULATOR.

VI. CONCLUSION

Monte Carlo methods are widely spread in finance and insurance industry, since complex or path dependent products in the finance domain can only be evaluated by running Monte Carlo simulations. The recent regulations and the need to compute prices and risk numbers very frequently nowadays have led to a dramatic increase in computation demands for finance simulations in the past. This is further supported by the trend to employ more realistic market models like the Heston model. FPGA accelerators have a huge potential for providing high simulation throughputs at very low power consumption.

In this work we present the first multi-level Monte Carlo FPGA architecture for FPGAs. The multi-level method iteratively runs Monte Carlo simulations by adjusting the simulation numbers and precisions, and has shown reduce the computational complexity compared to single-level methods by up to 50%. For European double barrier option pricing, we present the Heston discretization and the derivation of the hardware architecture from the multi-level algorithm in detail, together with specific implementation characteristics. For a Xilinx Virtex-6 FPGA we show that our proposed design can simulate up to 100 millions of time steps in an asset path simulation with less than 3.6 W. This clearly highlight the suitability of FPGAs for accelerating finance products with an impressive energy efficiency.

ACKNOWLEDGMENT

We gratefully acknowledge the partial financial support from the Center of Mathematical and Computational Modelling $(CM)^2$ of the University of Kaiserslautern and from the German Federal Ministry of Education and Research under grant number 01LY1202D. The authors alone are responsible for the content of this paper. Furthermore, we thank Steffen Omland for his helpful mathematical support.

REFERENCES

[1] S. Weston, J. Spooner, J.-T. Marin, O. Pell, and O. Mencer, "FPGAs Speed the Computation of Complex Credit Derivatives," *Xcell Journal*, no. 74, pp. 18–25, Mar. 2011.

[2] S. Weston, J.-T. Marin, J. Spooner, O. Pell, and O. Mencer, "Accelerating the Computation of Portfolios of Tranched Credit Derivatives," in *High Performance Computational Finance (WHPCF), 2010 IEEE Workshop on*, Nov. 2010, pp. 1–8.

[3] R. Lord, R. Koekkoek, and D. van Dijk, "A comparison of biased simulation schemes for stochastic volatility models," *Quantitative Finance*, vol. 10, no. 2, pp. 177–194, 2010.

[4] S. L. Heston, "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options," *Review of Financial Studies*, vol. 6, no. 2, p. 327, 1993.

[5] S. A. Griebsch and U. Wystup, "On the Valuation of Fader and Discrete Barrier Options in Heston's Stochastic Volatility Model," *Quantitative Finance*, vol. 11, no. 5, pp. 693–709, May 2011.

[6] C. Delivorias, "Case Studies in Acceleration of Heston's Stochastic Volatility Financial Engineering Model: GPU, Cloud and FPGA Implementations," Master's thesis, The University of Edinburgh, Aug. 2012. [Online]. Available: http://www.hpcfinance.eu/sites/www.hpcfinance.eu/files/Christos_Delivorias_0.pdf

[7] A. Bernemann, R. Schreyer, and K. Spanderen, "Accelerating Exotic Option Pricing and Model Calibration Using GPUs," WestLB et al., Herzogstrasse 17 Düsseldorf 40217 Germany, Feb. 2011. [Online]. Available: http://ssrn.com/abstract=1753596

[8] C. de Schryver, I. Shcherbakov, F. Kienle, N. Wehn, H. Marxen, A. Kostiuk, and R. Korn, "An Energy Efficient FPGA Accelerator for Monte Carlo Option Pricing with the Heston Model," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, Dec. 2011, pp. 468–474.

[9] H. Marxen, A. Kostiuk, R. Korn, C. de Schryver, S. Wurm, I. Shcherbakov, and N. Wehn, "Algorithmic Complexity in the Heston Model: An Implementation View," in *Proceedings of the fourth workshop on High Performance Computational Finance (WHPCF '11)*. ACM New York, NY, USA, Nov. 2011, pp. 5–12, iSBN 978-1-4503-1108-3.

[10] A. Bernemann, R. Schreyer, and K. Spanderen, "Pricing Structured Equity Products on GPUs," in *High Performance Computational Finance (WHPCF), 2010 IEEE Workshop on*, Nov. 2010, pp. 1–7.

[11] B. Zhang and C. W. Oosterlee, "Acceleration of Option Pricing Technique on Graphics Processing Units," Delft University of Technology, Tech. Rep. 10-03, Feb. 2010.

[12] R. Sridharan, G. Cooke, K. Hill, H. Lam, and A. George, "FPGA-based Reconfigurable Computing for Pricing Multi-asset Barrier Options," *Proceedings of Symposium on Application Accelerators in High-Performance Computing PDF (SAAHPC)*, Jul. 2012.

[13] J. C. Hull, *Options, Futures, And Other Derivatives*, 8th ed. Pearson, 2012.

[14] H. Marxen, "Aspects of the Application of Multilevel Monte Carlo Methods in the Heston Model and in a Lévy Process Framework," Ph.D. dissertation, University of Kaiserslautern, 2012.

[15] R. Korn, E. Korn, and G. Kroisandt, *Monte Carlo Methods and Models in Finance and Insurance*. Boca Raton, FL: CRC Press., 2010.

[16] S. Heinrich, "Multilevel Monte Carlo methods," *Large-Scale Scientific Computing*, pp. 58–67, 2001.

[17] M. B. Giles, "Multilevel Monte Carlo path simulation," *Operations Research-Baltimore*, vol. 56, no. 3, pp. 607–617, 2008.

[18] A. Kebaier, "Statistical Romberg Extrapolation: A New Variance Reduction Method and Applications to Option Pricing," *The Annals of Applied Probability*, vol. 15, no. 4, pp. 2681–2705, 2005. [Online]. Available: http://www.jstor.org/stable/30038520

[19] Q. Jin, D. Dong, A. Tse, G. Chow, D. Thomas, W. Luk, and S. Weston, *Reconfigurable Computing: Architectures, Tools and Applications: 8th International Symposium, Arc 2012, Hongkong, China, March 19-23, 2012, Proceedings*. Springer-Verlag New York Incorporated, Mar. 2012, vol. 7199, ch. Multi-level Customisation Framework for Curve Based Monte Carlo Financial Simulations, pp. 187–201.

[20] C. de Schryver, D. Schmidt, N. Wehn, E. Korn, H. Marxen, A. Kostiuk, and R. Korn, "A Hardware Efficient Random Number Generator for Nonuniform Distributions with Arbitrary Precision," *International Journal of Reconfigurable Computing (IJRC)*, vol. 2012, Mar. 2012, article ID 675130, 11 pages.