Hardware Accelerators for Financial Mathematics - Methodology, Results and Benchmarking

Christian de Schryver #, Henning Marxen *, Daniel Schmidt #

#*Micrelectronic Systems Design Department, University of Kaiserslautern*
Erwin-Schroedinger-Strasse, 67663 Kaiserslautern, Germany
schryver@eit.uni-kl.de
schmidt@eit.uni-kl.de

**Stochastics and Financial Mathematics Department, University of Kaiserslautern*
Erwin-Schroedinger-Strasse, 67663 Kaiserslautern, Germany
marxen@mathematik.uni-kl.de

***Abstract*—Modern financial mathematics consume more and more computational power and energy. Finding efficient algorithms and implementations to accelerate calculations is therefore a very active area of research. We show why interdisciplinary cooperation such as (CM)^2 are key in order to build optimal designs.**

For option pricing based on the state-of-the-art Heston model, no implementation on dedicated hardware is known, yet. We are currently designing a highly parallel architecture for field programmable gate arrays based on the multi-level Monte Carlo method. It is optimized for high throughput and low energy consumption, compared to GPGPUs. In order to be able to evaluate different algorithms and their implementations, we present a benchmark set for this application. We will show a very promising outlook on future work, including dedicated ASIPs, fixed-point research and real-time applications.

***Index Terms*—finance, benchmarking, hardware acceleration**

I. Introduction

Nowadays, financial markets are as vivid as never before. In modern electronic markets, stock prices may change several times within a few milliseconds. Participating traders (that can also be computers) have to evaluate the prices and react very quickly in order to get the highest profit, which requires a lot of computational effort.

In general, running these computations on servers or clusters with standard CPUs is not feasible due to either long run times or high energy consumption. Using general purpose graphics processing units (GPGPUs) as accelerators helps to increase the speed, but still requires a lot of energy. Besides, at the moment energy efficiency becomes more and more crucial for the reason of high energy costs and - even more critical - a limited supply of energy that can be provided. For example, in [16] it is stated that the City of London (with its new financial center Canary Wharf where a lot of leading institutes are located) does not provide additional energy until after the Olympic winter games in 2012, that have higher priority. Financial institutes are currently outsourcing all computing systems not used for pricing computations (such as storage or backup) out of the critical area.

This leads to the dilemma of needing faster computations on the one hand and limited energy resources on the other hand.

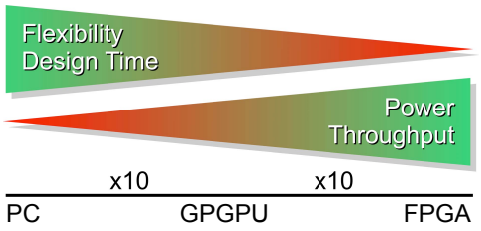


Fig. 1. PC vs. GPGPU vs. FPGA

Moving away from GPGPUs to dedicated hardware accelerators can help to drastically reduce the power consumption at the same or even higher throughput. For different application domains, some comparisons between CPU, GPGPU and programmable hardware units (field programmable gate arrays, FPGAs) have already been shown in [13] and [5], highlighting the enormous potential of energy savings for FPGAs. Figure 1 shows that standard software implementations require the least effort for implementation and can provide the highest flexibility, while dedicated hardware solutions on FPGAs are hard to design and - once finished - not easy to be changed again. From a different view, FPGAs can save up to about 99% of energy compared to a software implementation on a standard PC and allow a much higher throughput. GPGPUs are located between standard PCs and FPGAs. Between each neighboring architectures, one can expect a difference of about one order of magnitude on average for power consumption and throughput [13]. Although most financial institutes are relying on GPGPUs at the moment for the reason of standardized software development toolkits and their flexibility, FPGAs are an interesting alternative because of their higher energy efficiency.

A big challenge is the complexity of many models used to estimate the future price behavior of financial products. In many cases no mathematical closed-form solution exists so that approximation methods like Monte Carlo simulations or the finite difference method must be employed. Though, it is necessary to precisely specify a solution right at the beginning of the design process. Re-designing a nearly finished hardware implementation can require a very high amount of effort. The Center of Mathematical and Computational Modeling (CM)^2

of the University of Kaiserslautern is a perfect forum for an interdisciplinary cooperation to tackle this issue.

For this project, we have developed a design methodology that helps to select a feasible parameter set for a hardware accelerator in question that we present in Section III. In order to make implementations transparently comparable, we propose to use standardized benchmark sets - we elaborate on this in Section IV. By applying this methodology and our benchmark, we have developed some reference implementation designs that we show in Section V, together with the status quo of our research and our contributions up to now. In Section VI we give an overview of open issues and what we plan to examine in the future. Section VII concludes the paper.

II. State-of-the-Art and Related Work

Mathematical finance basically has two different directions. One is concerned with the evaluation of optimal investment strategies under certain market conditions and the other direction is the pricing of derivatives. The basic idea of pricing options is to assume some sort of model for the underlying price process and take the discounted expected value - under a certain measure - of an option as the option price.

A very common problem treated is the calculation of option prices based on the Black-Scholes model from 1973. This model relies on one stochastic differential equation and describes the price development of an option over the time, depending on market parameters such as riskless interest rate, long term drift and a *constant* volatility.

Accelerator design for financial mathematics is a very active research area, and several FPGA implementations have been published in the past. At the FPL 2008 Woods and VanCourt [17] presented a hardware accelerator for multiple, quasi-random, standard Brownian motions suitable for the acceleration of quasi-Monte Carlo simulation of financial derivatives. For credit risk modelling, Thomas and Luk could gain a speed-up of more than 90 times compared to a 2.4 GHz Pentium-4 Core2 [14]. An accelerator for Monte Carlo based credit derivative pricing was developed by Kaganov, Chow and Lakhany [10] in 2008 and showed to be 63 times faster than their software model. Wynnyk and Magdon-Ismail [18] presented an FPGA accelerator for American option pricing based on the Black-Scholes model in 2009 and could achieve a speedup of eleven up to 73 times compared to a software implementation running on a standard PC.

However, nowadays the Black-Scholes model is no longer up to date and does not provide an accurate reflection of modern financial market behaviors, mostly because of the volatility not being constant in reality. Furthermore, closed-form solutions for the Black-Scholes model exist and it only has demonstration purposes to apply stochastic solution methods such as Monte Carlo simulations or the finite difference method [1]. Nevertheless, it is still very common to publish accelerator implementations based on that model, at least in the electrical engineering community.

In 1993, Steven L. Heston presented a more accurate model [9] that extends the model from Black and Scholes by a second stochastic differential equation for stochastic volatility variations. This significantly increases the complexity of the calculation and of the implementation thereof. Nevertheless, the Heston model reflects the real behavior of current stock markets much better and is nowadays widely accepted in the financial mathematics community. But - to the best of our knowledge - no hardware accelerator for that model has been published up to now. For GPGPUs, the first implementations have been presented in the last year. Bernemann, Schreyer and Spanderen from the german bank WestLB [3] showed that they could achieve a speeup of 50 times over CPU by using GPGPUs for simulating the Heston model. Zhang and Oosterlee published a technical report [19] in March 2010 where they even showed speedups of more than 100 times.

The presented speed-ups look very impressive. However, unfortunately we were not able to fairly decide which solution seems to be the most promising for further research and refinements. We will go a bit more into the details of that problem in Section IV.

III. How To Choose The Right Design

For many fields of applications, finding the most efficient design under certain constraints is a difficult job. The main reason for this is a large *design space*. The design space is made up of all possible parameter choices for the design, that means all possible implementation instances.

Most parameters are not adjustable independently, since they are mutually linked. For example, fixing the target architecture to FPGAs one the one hand has a large impact on the selection of suitable algorithms and number systems, and on the other hand affects many performance metrics such as energy consumption, throughput and numerical precision.

Furthermore, the parameters within the design space are in many cases not limited to a single domain of expertise, but require interdisciplinary know-how and decisions. This makes not only the choice of the right values a challenge, but also the evaluation and comparison of different implementations. Besides speedup, more characteristics such as energy efficiency, convergence rate or numerical precision may be very important. This especially holds true for financial mathematics accelerators.

During our research we have seen a lot of papers that show elaborate implementations of a specific algorithm (see Section II) that is not questioned in the papers anymore. However, we claim that the algorithm itself is in fact not the most important selection. An accelerator should be designed to solve a specific problem - it does not matter which algorithm is used, as long as the result is calculated correctly. We therefore propose to distinguish clearly between three terms:

- the *problem* that is tackled (what to solve)
- the employed *model* (how to solve)
- the *solution* (how to build)

To clarify the situation, we use the *problem* "calculate the price of an option with two barriers for a given duration" as an example. European knock-out barrier options pay a certain amount of money at a fixed maturity time depending on the

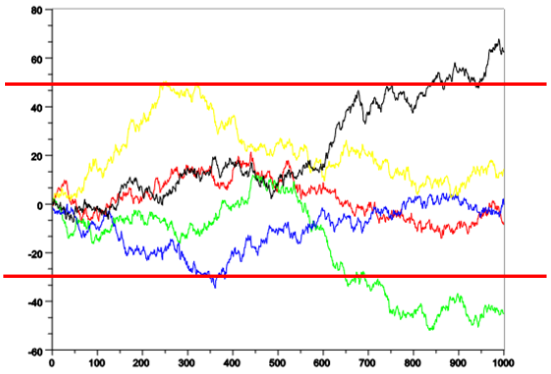


Fig. 2. Barrier testing for a Brownian motion

value of the underlying asset. This amount is only paid when the barrier is not crossed up to the maturity time. If one of the barriers is hit, the option becomes worthless. Thus it needs to be checked whether the barrier was ever hit or not. Figure 2 illustrates the typical random behavior of different realizations of an asset price over the time.

It is obvious that the problem description itself does not yet give any suggestions to the solution. Since the price of an option is tightly coupled to the price of a certain stock at the market, we need a *model* that provides the stock price behavior. For our chosen example, suitable models are for example the Black-Scholes model (outdated nowadays) or the Heston model. The model in general gives a formal and abstract view of (a certain aspect of) the problem.

The *solution* finally is a dedicated approach for solving a (modeled) problem. It is characterized by a specific *algorithm* and its *implementation*. For evaluating the Heston model, for example, finite difference methods or stochastic Monte Carlo simulations can be used. They may be implemented for example on standard PC clusters, GPGPUs or on FPGAs.

The parameters of the design space can basically be divided into two groups: the algorithmic parameters that are mostly selected by mathematicians, and the implementation parameters determined by the hardware designers. However, as mentioned before correlations exist between several parameters, so that the selection should be optimally made by having a generative exchange between experts of both groups.

For the rest of the paper, we will use the problem-model-solution triple "calculate the price of an option with two barriers for a given duration with the Heston model by using Monte Carlo methods" as a showcase. Even for that specific selection, the design space is still very large. An extract of the related design space is shown in Figure 3[1]. We cannot explain every parameter here (for details see [11] [8]), but in a nutshell it is obvious that even for a very specific task a huge amount of possible accelerator implementations may exist. In Figure 3 we see two different views of the same tree. These trees are symbolic for the design space. The left one is the view of a mathematician that has mostly to do with algorithmic and numerical aspects. The right view is the one of an electrical engineer who may not understand the mathematician's concerns in detail, but is wondering what the best decisions with respect to hardware efficiency might be.

[1] http://www.sxc.hu/photo/285734 We thank sxc user vxdigital for sharing this image of the oak tree and allowing the use of it in this paper. He holds all copyrights to this image.

IV. Benchmarking - Fairly Comparing Implementations

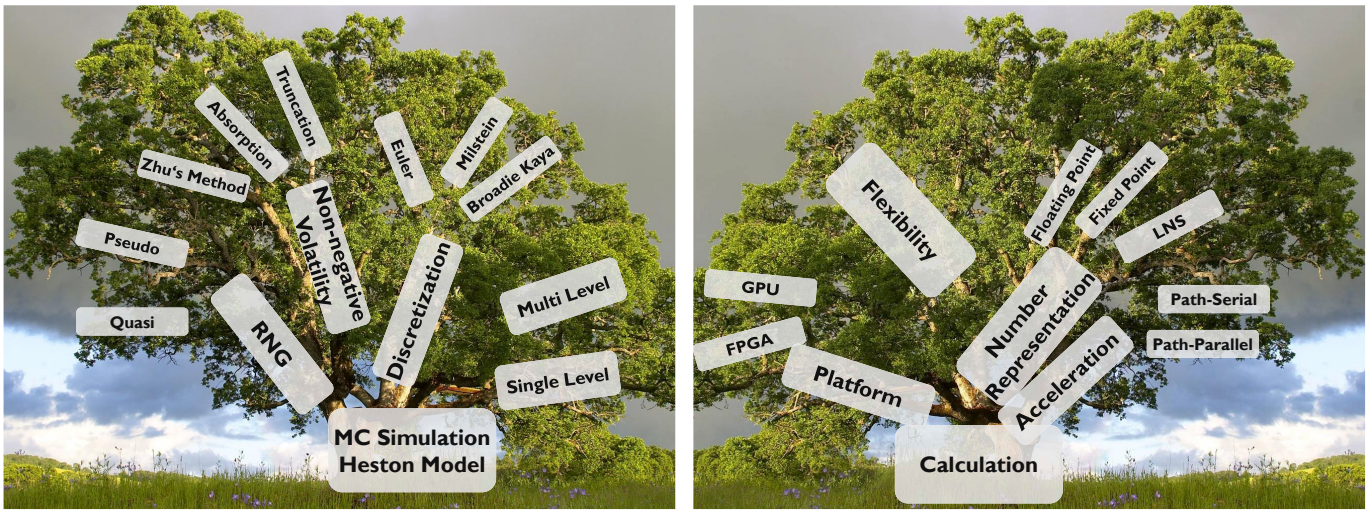
Comparing different implementations is a non-trivial task. Many attributes can be considered, including speed, accuracy and energy consumption. This becomes even more difficult when it is not clear which algorithm was used. Furthermore, in many cases it is not possible to distinct whether a presented algorithm or implementation has the displayed behavior only with the employed example or in a more general setting. Nevertheless, it is important to be able to compare various algorithms and different implementations, also over various target architectures.

Therefore the need for a benchmark set arises. This set should be independent of the algorithm and implementation used. For option pricing in financial mathematics, this need has already been claimed by Morris and Aubury in 2007 [12]. We are not aware of any progress made since that paper was published. We therefore have decided to develop a completely new benchmark that will enable us to fairly compare different algorithms, e.g. multi-level and single-level Monte Carlo, on different hardware. Thus we propose a benchmark based on the *problem/model* combination. In our case it is the pricing of double barrier options in the Heston model. It is clear that independently of the used algorithm and implementation the result must be the same. Therefore the final prices of the different options in the benchmark set have to be provided.

With the benchmark set it is possible to use different metrics, like speed (that is now the real time until the results are available), accuracy and power consumption, for the calculations leading to the right (or approximate) result without actually looking at the implementation and the algorithm itself. This allows a fair and publicly traceable comparison of the *solution* part of the *problem/model/solution* triplet.

The benchmark itself consists of different combinations of parameters for the Heston model and for barrier options, including the prices. The data for the Heston parameters is taken from different recent publications ([2], [11], [20]) and are enlarged by an extremer case. The benchmark parameters span a wide range of possible combinations used in this field. For some options of the benchmark closed form solutions exist that allow to obtain the exact results. This is important to verify that simulations converge to the correct values and makes it easier to compare the results. For the other cases the exact prices are not known and are therefore provided as close approximations.

For further publications we not only encourage the authors to use the presented benchmark but also give details of the algorithm and the implementation used. Thus it is possible to see where an increase in performance comes from. This is essential in order to evaluate the contribution of a certain result and to find ways to improve it even further. To achieve a higher transparency we will publish the code we used to



(a) Algorithmic parameters

(b) Implementation parameters

Fig. 3. Two views of the same solution tree

analyze the algorithms and the one we implemented on the FPGA.

The benchmark was directly used when comparing different Monte Carlo algorithms with the metric of computational complexity. In our special *problem/model* combination there are a lot more adjustments to the algorithms than seen in figure 3(a). Many of them can be combined, what leads to a huge design space that now can be handled by applying the benchmark set, so that we haven been able to choose a specific algorithm to be efficiently implemented on dedicated hardware. The multi-level Monte Carlo method provides a better asymptotic convergence behavior, using our benchmark we checked whether this method is beneficial for our application. We will show our first results in the next section.

V. Status Quo and First Results

We have started this cooperation within $(CM)^2$ about one year ago now. Both participating chairs have experience of more than ten years in their respective field of research, so that we can profit from a lot of knowledge in the areas of efficient hardware design respectively stochastics and financial mathematics.

After evaluating the state-of-the-art, we decided to focus on accelerators for option pricing based on the Heston model - it seems to be a very promising topic since no implementations (either on GPGPUs or FPGAs) have been available one year ago. In contrast to that, the Heston model is already widely spread within the financial community. From former research done in the group of Prof. Korn, multi-level Monte Carlo methods [7] seemed to provide a better convergence behavior than standard single-level Monte Carlo or finite difference methods. Monte Carlo methods also have the advantage of being very flexible. A barrier that is only relevant on a certain time interval to evaluate an option price for example can be easily implemented. Furthermore, multi-dimensional problems can also be solved. This is needed in the case that an option has more than one underlying asset. Nevertheless, for our project we will stick to one asset.

A. A New Random Number Generator for Non-Uniform Distributions

Inherently, Monte Carlo simulations always consume a huge amount of random numbers. To obtain the maximum hardware efficiency for our implementation, we have developed a new random number generator for non-uniform distributions tailored to our application.

For our option pricing accelerators, we need two independent, normally distributed random numbers for each time step of a single simulated stock price path. In general, non-uniformly distributed random numbers are generated in two steps: First, a uniform random number generator creates uniformly distributed values, and in a second step this number is transformed into the desired target distribution.

For the uniform random number generation, a lot of research has already been made leading to efficient and well-proven implementations, such as the Mersenne Twister MT19937 that we use. The three main approaches for obtaining non-uniform distributions are transformation, rejection, and inversion methods [15].

For FPGAs, inversion methods are the usual way to go. They combine many desireable properties: by applying the respective inverse cumulative distribution function (ICDF), they transform every input sample $x \in (0, 1)$ from a uniform distribution to one output sample $y = icdf(x)$ of the desired output distribution by using piecewise polynomial approximation of the ICDF. The works of Woods and VanCourt [17] and Cheung et al. [4] show FPGA implementations of the inversion method.

However, both implementations use fixed-point number representations at the input. This leads to a loss of precision in the tail regions where the probability of a value lying there is very low. But these extreme events can have a large impact, for example for options with barriers it is crucial to know if a

barrier was hit or not, since it completely changes the refund conditions. We have therefore developed a new implementation based on floating-point representation that provides the same precision over the whole ICDF implementation at much lower hardware costs. This work has been presented at the 2010 International Conference on ReConFigurable Computing and FPGAs (ReConFig) in December in Cancún, Mexico [6]. Our random number converter unit requires only about half of the area compared to other state-of-the-art implementations by even higher numerical precision.

To validate our work, it was crucial to develop a new testing methodology, since standardized test suites do not exist for non-uniform distributions. This work has been carried out in cooperation with Elke Korn who has a lot of knowledge in the field of random numbers. The methodology and the validation results for our implementation are also presented in the paper. Our random numbers did not show any noticeable problems in the stochastic tests and also perfectly passed two different application simulations.

B. Fully Parallel Hardware Accelerator

To the best of our knowlege, no hardware implementations of option price accelerators based on the Heston model exist at the moment. We have therefore started with the first implementation, that is nearly finished now. The hardware is fully-parallel, fully-pipelined and designed for high throughput.

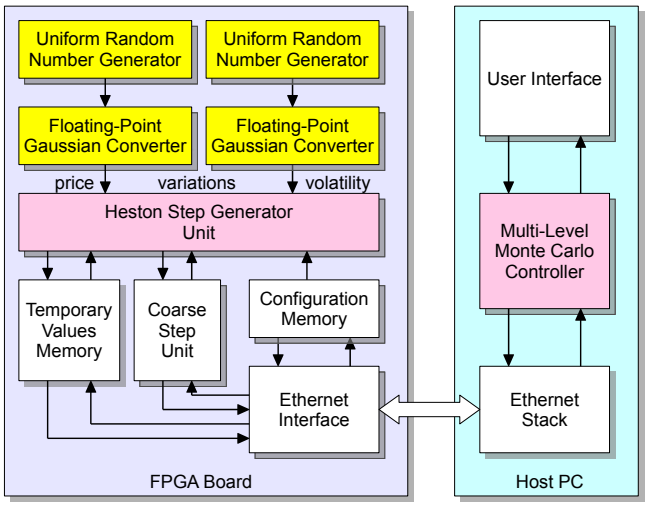


Fig. 4. Fully Parallel Accelerator Structure

Figure 4 shows the structure of one pipelined accelerator circuit. In each clock cycle, our unit consumes two normally distributed random numbers, one for the stock price variation and one for the volatility variation. The Heston step generator unit calculates the price and volatility values for the next time step based on a multi-level Monte Carlo algorithm. The pipeline depth is about 60 stages. In order to maximally utilize the pipelined hardware, it computates one time step for 60 assets in parallel, before moving to the next time step. The values for the respective time steps are stored in a memory temporarily. The coarse step memory holds interim values for a higher step width, this means for a lower Monte Carlo simulation level. In the configuration memory, all model parameters are stored.

Due to the inherent parallelism of Monte Carlo simulations, it is not only feasible but self-evident to instantiate as many of these circuits as possible on an FPGA in order to increase the simulation throughput.

The accelerator is implemented on a Xilinx ML-605 evalulation board equipped with a Xilinx Virtex-6 FPGA. The board is connected via Gigabit ethernet to a host PC running the user interface and the program that calculates and sets the configuration values for the accelerator based on the retrieved simulation results.

Synthesis and benchmarking results will be available soon. We are currently supporting single- and double-precision floating-point computations and are working on a fixed-point implementation as well.

VI. Outlook and Future Work

Also for the intended future work a close cooperation between financial mathematics and electrical engineering will be mandatory, since we are planning to research aspects out of both fields.

One characteristic of the Monte Carlo method is the inherent capability to parallelize the calculation. It therefore makes no difference whether several calculations are done in parallel but slowly or only one calculation is done at high speed, as long as the number of calculations remains equal altogether. At the moment, our parallel implementation allows simulating one time step on many assets simultaneously. The whole procedure of calculating one time step is fixed on the FPGA and limited to a single algorithm that is set at design time.

One possible way to improve this is to sequentially compute the basic calculations needed for one time step on a Application-Specific Instruction-Set Processor (ASIP) within the FPGA, i.e. it is runtime-programmable. This procedure can reduce the required area and allows to calculate various algorithms since the functionality is defined in a corresponding program. It is therefore sufficient to load a different program without changing the hardware. The ASIP will occupy much less area than the parallel implementation presented in Section V-B, therefore many ASIPs can be instantiated in parallel. We are currently investigating the necessary instruction set.

To increase the speed of the implementation even more, the floating-point computation can be replaced by fixed point computations. In order to do so, errors resulting from the use of fixed-point calculations have to be approximated. This task will also require both theoretical and practical expertise.

Besides working on the implementation, the benchmark is very important to evaluate the designs. It will also allow to research fixed-point solutions with respect to necessary precision. Further steps will be to publish the analysis of the algorithms with the benchmark in a journal. To increase the transparency even more, we are currently setting up a web site offering all the program code used to create the analysis. It will contain an implementation of the multi-level and the

crude Monte Carlo algorithm with a focus on the calculation complexity rather than the implementation.

To provide more benchmarking results, also for different architectures, we are working on a GPGPU implementation. It provides more flexibility and less implementation work than hardware designs do, but also requires higher energy consumption. In order to verify this assumption the implementation is required. The Monte Carlo simulation algorithm for the Heston model is currently carried out as a student work.

Moreover, we are currently researching on real-time acceleration of financial calculations. This means that hardware or GPGPU accelerators are linked to real-time data streams. This approach seems to be very promising for keeping track with the prices changing quickly in high-frequency trading.

VII. Conclusion

The financial world is running faster and faster and the importance of energy consumption increases drastically. To address this challenge the question of pricing double barrier options in the Heston setting is faced. As the model is more complex than the famous Black-Scholes model and these types of options are path dependent, the algorithms for the calculations are more distinct and also the implementation thereof. In order to be able to cope with the strong connection between the algorithm and the implementation, a combined mathematical and electrical engineering view is needed. $(CM)^2$ provides a perfect framework to do so.

To approximate the pricing process Monte Carlo simulations are used. For a good implementation a fast algorithm with an adjusted implementation thereof is needed. In order to distinguish the different algorithms we have created a benchmark set for double barrier options. This benchmark allows to fairly analyze and compare the diverse algorithms and designs, which is a very important issue due to the big differences in the convergence speed of these algorithms.

For the target architecture, using FPGAs is the hardware of choice if implementation time is not considered. It allows fast computation with low energy consumption. Nevertheless, optimal FPGA designs require deep understanding of the FPGA characteristics and the calculations need to be optimized for it. One commonality of all the Monte Carlo algorithms is the use of (pseudo-)random numbers. In the Heston setting standard normal random numbers are used. There are procedures to create these running efficiently on GPGPUs and CPUs. A new method was presented which is very efficient for an implementation on a FPGA.

The detailed analysis of the diverse algorithms was used to make an efficient implementation. Therefore the algorithm is implemented on an FPGA. This should allow a fast computation with low energy consumption. As far as we know, this will be the first implementation of a Monte Carlo simulation in the Heston model on a FPGA. Furthermore it will be the first implementation of the multi-level Monte Carlo method on this hardware. Thus this work expands the field of implementations of financial mathematical problems on dedicated hardware in several ways as new concepts are taken into consideration.

Acknowledgements

We gratefully acknowledge the partial financial support from Center of Mathematical and Computational Modeling $(CM)^2$ of the University of Kaiserslautern.

References

[1] R. L. Akers, I. Bica, E. Kant, C. Randall, and R. L. Young. SciFinance: A Program Synthesis Tool for Financial Modeling. *AI Magazine*, 22(2):27, 2001.

[2] L. Andersen. Efficient simulation of the heston stochastic volatility model. 2006.

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

[4] R. C. C. Cheung, D.-U. Lee, W. Luk, and J. D. Villasenor. Hardware Generation of Arbitrary Random Number Distributions From Uniform Distributions Via the Inversion Method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(8):952–962, Aug. 2007.

[5] B. Cope, P. Y. Cheung, W. Luk, and S. Witt. Have GPUs made FPGAs redundant in the field of Video Processing? In *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, pages 111 –118, Dec. 2005.

[6] C. de Schryver, D. Schmidt, N. Wehn, E. Korn, H. Marxen, and R. Korn. A New Hardware Efficient Inversion Based Random Number Generator for Non-Uniform Distributions. In *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 190–195, Dec. 2010.

[7] M. B. Giles. Multilevel Monte Carlo path simulation. *Operations Research-Baltimore*, 56(3):607–617, 2008.

[8] A. V. Haastrecht and A. Pelsser. Efficient, almost exact simulation of the Heston stochastic volatility model. *International Journal of Theoretical and Applied Finance*, 13(1):1–43, 2010.

[9] S. L. Heston. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *Review of Financial Studies*, 6(2):327, 1993.

[10] A. Kaganov, P. Chow, and A. Lakhany. FPGA Acceleration of Monte-Carlo based Credit Derivative Pricing. In *Proc. Int. Conf. Field Programmable Logic and Applications FPL 2008*, pages 329–334, Sept. 2008.

[11] R. Lord, R. Koekkoek, and D. van Dijk. A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, 10(2):177–194, 2010.

[12] G. W. Morris and M. Aubury. Design Space Exploration of the European Option Benchmark using Hyperstreams. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 5 –10, Aug. 2007.

[13] D. B. Thomas, L. Howes, and W. Luk. A Comparison of CPUs, GPUs, FPGAs, and Massively Parallel Processor Arrays for Random Number Generation. In *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '09, pages 63–72, New York, NY, USA, 2009. ACM.

[14] D. B. Thomas and W. Luk. Credit Risk Modelling using Hardware Accelerated Monte-Carlo Simulation. In *Proc. 16th Int. Symp. Field-Programmable Custom Computing Machines FCCM '08*, pages 229–238, Apr. 2008.

[15] D. B. Thomas, W. Luk, P. H. Leong, and J. D. Villasenor. Gaussian Random Number Generators. *ACM Comput. Surv.*, 39(4):11, Oct. 2007.

[16] P. Warren. City business races the Games for power. *The Guardian*, May 2008.

[17] N. A. Woods and T. VanCourt. FPGA Acceleration of Quasi-Monte Carlo in Finance. In *Proc. Int. Conf. Field Programmable Logic and Applications FPL 2008*, pages 335–340, 2008.

[18] C. Wynnyk and M. Magdon-Ismail. Pricing the American Option using Reconfigurable Hardware. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pages 532 –536, Aug. 2009.

[19] B. Zhang and C. W. Oosterlee. Acceleration of Option Pricing Technique on Graphics Processing Units. Technical Report 10-03, Delft University of Technology, Feb. 2010.

[20] J. E. Zhang and J. Shu. Pricing s&p 500 index options with heston's model. In *Proc. IEEE Int Computational Intelligence for Financial Engineering Conf*, pages 85–92, 2003.