Reverse Longstaff-Schwartz American Option Pricing on hybrid CPU/FPGA Systems

Christian Brugger,
Javier Alejandro Varela, Norbert Wehn
Microelectronic Systems Design Research Group
University of Kaiserslautern, Germany
{brugger, varela, wehn}@eit.uni-kl.de

Songyin Tang, Ralf Korn
Stochastic Control and
Financial Mathematics Group
University of Kaiserslautern, Germany
{tangs, korn}@mathematik.uni-kl.de

***Abstract*—In today's markets, high-speed and energy-efficient computations are mandatory in the financial and insurance industry. At the same time, the gradual convergence of high-performance computing with embedded systems is having a huge impact on the design methodologies, where dedicated accelerators are implemented to increase performance and energy efficiency. This paper follows this trend and presents a novel way to price high-dimensional American options using techniques of the embedded community. The proposed architecture targets heterogeneous CPU/FPGA systems, and it exploits the FPGA reconfiguration to deliver high-throughput. With a bit-true algorithmic transformation based on recomputation, it is possible to eliminate the memory bottleneck and access costs. The result is a pricing system that is 16x faster and 268x more energy-efficient than an optimized Intel CPU implementation.**

I. Introduction

Harsh competition and stringent regulatory agencies drive financial institutions to constantly increase the complexity of their mathematical market models. Due to their high costs, there is a high incentive to investigate efficient ways to perform pricing and risk management tasks. One approach is to build specialized computing architectures, and in this regard field programmable gate arrays (FPGAs) are starting to be used extensively [1] based on performance and energy-efficiency.

While many numeric algorithms map nicely to FPGAs, there often remain parts that are best executed on central processing units (CPUs). Hybrid devices combine CPU and FPGA fabrics on a single device, delivering the best of both worlds. A key challenge of such heterogeneous computing systems is to carefully balance all aspects of the hardware, including communication, reconfiguration times, memory bandwidth, FPGA area and CPU loads.

In this work we will investigate custom computing solutions for the Longstaff-Schwartz (LS) method [2], a Monte Carlo (MC) method to price American options. Our proposed solution targets hybrid computing systems and is able to perform high-precision and energy-efficient computations. Besides the classical approach, we present a novel algorithmic improvement that does not require to store all intermediate steps for all paths, but recomputes them on the fly. Recomputation is a well known technique in embedded system to avoid energy costly memory accesses [3], [4]. This allows us to reduce the energy consumption by trading-off memory bandwidth with FPGA resources, effectively moving less data across the board.

Our novel contributions are:

- A high-throughput option pricing architecture suitable for high-precision computations.
- A novel Reverse LS algorithm that does not require to store the full intermediate results, reducing the requirements on external memory.
- A thorough analysis and comparison of the conventional method and the new architecture, including area, performance and energy numbers.

II. Background and Related Work

The use of FPGAs for accelerating financial simulations has become attractive since the release of the first devices. For MC methods de Schryver et al. have shown that FPGAs are 33x more energy-efficient in the Heston market model [5]. For the GARCH model Thomas et al. could show speedups of 80x [6], for the Black-Scholes (BS) model they showed speedups of 313x [7]. Sridharan et al. have extended this work to multi-asset options in the BS model [8], with speedups of up to 350x for one FPGA device. All four implementations can only price European options. They cannot be used for American options, for which an optimal exercise strategy has to be found.

As far as we know there is only one FPGA implementation for MC-based American options pricing by Tian and Benkrid [9], based on the LS method. By making use of an efficient fully parallel architecture and an external memory chip to store the simulated MC paths, they have achieved speedups of 20x. While their implementation is similar to ours in some basic points, our proposed architecture provides significant advantages on both the algorithmic and implementation level, being more energy efficient.

A. Black-Scholes (BS) Model

Due to its robustness and easy extensions in practice [10], the BS model is still used widely nowadays for many high-dimensional products, for which more complex models would be too hard to calibrate to, due to overfitting problems. In the BS model the price $S(t)$ of an economic resource is modeled as a stochastic differential equation:

$$\mathrm{d}S(t) = S(t)\mu\,\mathrm{d}t + S(t)\sigma\,\mathrm{d}W(t), \tag{1}$$

with μ being the drift that can be used to model risk-free interest rate and dividends. The constant volatility σ is a

978-3-9815370-4-8/DATE15/©2015 EDAA

measure of the observable fluctuations of the price S and $W(t)$ is the Brownian motion. The fair price of a derivative today can be calculated as $P = \mathbb{E}[g(S)]$, where g is a corresponding discounted payoff function. Although closed-form solutions for simple payoffs like vanilla European options exist, so-called *exotic* derivatives like barrier, lookback, or American options must be priced with compute-intensive numerical methods. A common choice is MC methods.

B. Monte Carlo (MC) Methods

Simulating the BS model in Eq. (1) requires the application of an appropriate discretization scheme. In this work we have applied the *Euler discretization*. Discretizing the Eq. (1) into m steps with equal step sizes $\Delta t = \frac{T}{m}$ leads to:

$$\hat{S}_{t_{i+1}} = \hat{S}_{t_i} \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,\Delta W_i\right), \qquad (2)$$

with ΔW_i being independent standard normal random variables.

The classic MC algorithm estimates the price P as the sample mean of simulated instances of the discounted payoff values $g(\hat{S})$. Its complexity depends only linearly on the number of dimensions, which makes such methods an excellent candidate for high-dimensional problems or a method of last resort for options with no other numerical scheme.

C. Challenges with American Options

In contrast to European options with a fixed exercise date, American-style options can be exercised throughout a given period until maturity T. This implies that an optimal exercise strategy needs to be estimated, and several custom MC algorithms have been proposed to this end. The algorithm by Longstaff and Schwartz [2] is the most popular one in practice [11], introduced next.

D. Longstaff-Schwartz (LS)

The LS algorithm uses least-squares linear regression to find the optimal exercise boundary. The basic steps are:

1) Generate N independent paths per underlying (stock) at all possible exercise dates, using a chosen random number generator (RNG) and a mathematical model (Eq. (2)). For multi-dimensional options, the random numbers (RNs) are correlated.
2) Initialize the cash-flow matrix with the discounted payoffs at maturity.
3) Moving backwards one step in time (t_{m-1}), proceed as follows:
   - Linear regression: the goal is to find out whether to exercise the option or to hold it. For this purpose, the current discounted payoff (when exercised) is compared to the value for holding the option (hold price), approximated by regression. Least-squares linear regression with user-defined basis functions is applied to compute the hold price for each time.
   - Cash-Flow matrix update: For every path at the current time step compare the expected return with the current discounted payoff, take the larger one and update the corresponding value of the cash-flow.

   Repeat this process step by step until the initial day.
4) At the initial day, average all values in the cash-flow matrix to obtain the option value.

The challenging part for LS is the choice of basis functions for regression. They highly depend on the exact option being priced and need to be matched well to the characteristics of the payoff function g.

III. Reverse Longstaff-Schwartz

In the formulation of the LS algorithm in Section II-D, first all paths are generated in step 1) and then traversed in reverse order in step 3). That means the value of each stock price at each time step for all paths has to be stored and communicated between these steps. A total of dmN values, with d being the dimension of our derivative. We call this standard approach the *path storage solution*.

For FPGAs, with only limited internal storage of a few MB, this poses a huge design challenge and in general requires to use several external high-speed memory devices, making the design much more complex. We will now present a novel idea based on recomputation to avoid this massive storage of data.

Instead of storing the paths at each time step, we only store the final stock prices at maturity $\hat{S}_{t_m}$ and then recompute all the others alongside step 3) of the LS algorithm. For that to work we need to find a way to compute the stock price $\hat{S}_{t_i}$ based on the future price $\hat{S}_{t_{i+1}}$:

$$\hat{S}_{t_m} \to \hat{S}_{t_{m-1}} \ldots \to \hat{S}_{t_1} \to \hat{S}_{t_0}.$$

The discretized BS equation in Eq. (2) is reversible provided we supply the same RNs, such that:

$$\hat{S}_{t_i} = \hat{S}_{t_{i+1}} \exp\left(\left(\frac{\sigma^2}{2} - \mu\right)\Delta t - \sigma\sqrt{\Delta t}\,\Delta W_i\right).$$

In our work we are using the Mersenne twister (MT) 19937 algorithm [12], which is a linear RNG. This means that its state transition function is invertible, allowing us to generate exactly the opposite sequence of random numbers, starting from the last one. In this regard, we based our reverse MT 19937 implementation on the Fortran code of Hagita [13]. As a result, the Reverse LS method only needs to store and communicate dN values.

IV. Hybrid Longstaff-Schwartz Architecture

An optimized schedule where no blocks remain in idle is possible in hybrid CPU/FPGA systems, such as the Xilinx Zynq, where dynamic reconfiguration is used. This approach has led to the architecture shown in Fig. 1, which follows the explanation given in Section II-D.

A. Amortizing the FPGA Reconfiguration

Reconfiguring the FPGA implies certain time and energy consumption which can easily exceed the runtime and energy consumption required when pricing a single option. However, when pricing a large set of options, the combination of the paths storage approach and the novel Reverse LS allows for

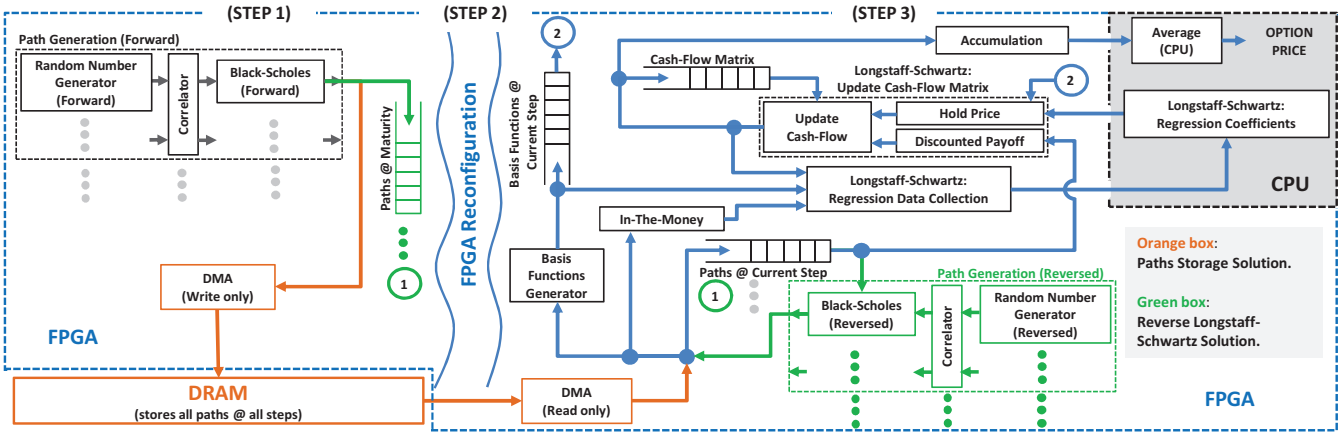

Fig. 1. Architecture: Forward path generation, Reconfiguration, and LS. Both solutions are included: Paths Storage and Reverse LS. Paths are generated via MT 19937. An ICDF module converts the uniform distribution obtained from MT into a normal distribution [14], as required by BS model.

easy amortization of the mentioned reconfiguration. In this case, all paths are generated for every option, but only the ones at maturity are stored in an external memory chip. Once the process is finished, the FPGA is reconfigured only once and the options are priced one by one, initializing the paths from the external memory and recomputing the paths backwards by means of the Reverse LS.

V. Longstaff-Schwartz on Zynq

The Xilinx Zynq-7000 hybrid CPU-FPGA device was chosen to implement the LS. In this regard, all modules in FPGA have been designed in Vivado high-level synthesis (HLS), pipelined and optimized for a clock frequency of 133 MHz. The place-and-route (P&R) report on resources utilization was then fed into the Xilinx Power Estimator [15] in order to obtain power estimations of individual blocks. These estimated values have been cross-verified by means of testbenches on the Xilinx Zynq ZC702 Evaluation Kit. In a similar way, DRAM DDR3 power consumption is based on measured values at different bandwidths on the same board, and extrapolated to the maximum theoretical bandwidth available (4266 MB/s at 533 MHz and 32-bit data bus). Energy consumption is derived from the obtained average power consumption and the required runtime.

A comparison between the paths storage approach against their recomputation is only possible in a common setup. In this regard, the bandwidth is set equal to the maximum available in a single DRAM interface on Zynq. Then multiple parallel instances of all blocks are set in the FPGA in order to achieve the same bandwidth, and the comparison is carried out in terms of energy consumption at the same bandwidth.

TABLE I
FPGA resources breakdown

Step	Block	LUT	FF	DSP	BRAM
1	Path Generation Forward (4x)	18404	17376	188	88
	Paths @ Maturity (4x)	1752	1648	0	64
2	Reconfiguration	-	-	-	-
3	Longstaff-Schwartz (4x)	28296	33468	212	108
	Path Generation Reversed (4x)	24048	23932	164	88

To evaluate the runtime and energy consumed we price an American maximum call option on two correlated stocks with 365 time steps and 10K paths per stock and basis functions: $\{1, \max(S_1, S_2), \max(S_1, S_2)^2\}$. We cross verified our implementation with a binomial tree implementation[1].The fair option price in this case, and for a strike price K, is given by:

$$P = \sup_{\tau \in [0,T]} \mathbb{E}\left[e^{-r\tau}(\max\{S_1(\tau), S_2(\tau)\} - K)^+\right].$$

To reach the target bandwidth in this setting we instantiate 4 path generation instances, each having 2 BS blocks, and 4 instances of the regression step 3 (blue part in Fig. 1).

A. Results and Comparison

The total amount of stock data adds up to 27.85 MB, which at 4266 MB/s in a fully pipelined block is processed in approximately 6.53 ms. The total runtime, including the communication overhead between CPU and FPGA and excluding the FPGA reconfiguration, equals 16.94 ms. A full reconfiguration on the Zynq would take 50 to 100 ms.

For the given setup, the FPGA resources utilization is summarized in Table I. A note is made on the fact that the minimum Zynq device in which such a setup fits is the Z-7030.

Fig. 2 presents the energy consumption breakdown of the whole architecture when the novel Reverse LS approach is implemented.

When comparing the recomputation of the paths in FPGA against the storage of all paths in DRAM (both when writing and reading data), there is a reduction in energy consumption of 2x, as depicted in Fig. 3. Since the DRAM is already used by the Linux operating system running on the ARM cores of the Xilinx Zynq, we only accounted for the additional dynamic energy required for this comparison.

[1]Reverse LS: $P = 9.92 \pm 0.24$; Binomial Tree (Benchmark): $P = 10.12$; Setup: $S_{1,2}(0) = 100$, $K = 100$, $r = 0.05$, $\mu_{1,2} = -0.05$, $\sigma_{1,2} = 0.2$, $\rho = 0.1$. The chosen basis functions generally deliver good results for general options, however not the best result. This depends on the type and the number of basis functions, which need to be tried and tested.

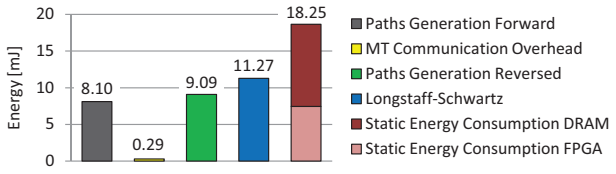

Fig. 2. Energy consumption breakdown of the LS architecture on Zynq.

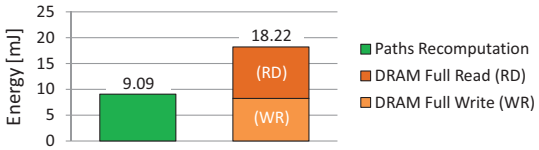

Fig. 3. Energy required to read and write the paths from/to DRAM is two times as high as recomputing them on the FPGA alongside the LS algorithm.

An optimized CPU implementation of the entire algorithm in Matlab on an Intel i5-2450M (2.50 GHz) core with, 6 GB of RAM, requires, for the given setup, 270 ms and an energy consumption of 12.70 J. The latter has been obtained at the power-plug with all unnecessary components in the computer disabled. In contrast, our implementation in Zynq requires 16.94 ms and consumes 47 mJ, providing a speedup of 16x in runtime and 268x in energy consumption. FPGA configuration times are not considered here, based on the amortization techniques depicted in Section IV-A.

B. Comparison to Related Work

The reference work [9] presented a dedicated FPGA implementation targeted for one specific option and setting, and 26/32-bit fixed-point operations. Our proposed architecture targets high-dimensional options, and further uses single-precision floating-point operations, so that the user does not have to take care of the accuracy of the solution.

A direct comparison proves hard to be accomplished. In terms of energy efficiency, porting their work [9] to the same Xilinx Zynq device would yield, according to XPE, 2.46 mJ dynamic energy with one DRAM chip. In turn, our downscaled architecture to one-dimension and the same number of paths and steps only requires 1.85 mJ dynamic energy, being a 33 % improvement while also providing higher precision.

VI. Conclusion

American option pricing is a computational challenge for financial institutions, which operate huge clusters. In this paper we propose a high-throughput and energy-efficient pricing system for American options. Compared to the state-of-the-art, this is the first FPGA-based implementation targeting the full range of multi-dimensional American options.

Our main contribution is Reverse Longstaff-Schwartz, a bit-true algorithmic transformation where we exploit recomputation. Instead of storing all paths, we simply recompute them later, removing any bandwidth limitation and significantly improving energy-efficiency. By additionally exploiting runtime reconfiguration and utilizing an optimized scheduling to amortize the reconfiguration times, we are able to deliver higher energy-efficiency. In this regard, the resulting architecture is 16x faster and 268x more energy-efficient than an optimized Intel i5 implementation in Matlab.

Acknowledgment

We gratefully acknowledge the partial financial support from the Center of Mathematical and Computational Modelling $(CM)^2$ of the University of Kaiserslautern, from the German Federal Ministry of Education and Research under grant number 01LY1202D and from the Deutsche Forschungsgemeinschaft (DFG) within the RTG GrK 1932 "Stochstastic Models for Innovations in the Engineering Sciences", project area P2. The authors alone are responsible for the content of this paper.

References

[1] M. Feldman. (2011, Jul.) JP Morgan Buys Into FPGA Supercomputing. HPCwire. Last access: 2014-09-16. [Online]. Available: http://archive.hpcwire.com/hpcwire/2011-07-13/jp_morgan_buys_into_fpga_supercomputing.html

[2] F. A. Longstaff and E. S. Schwartz, "Valuing American options by simulation: A simple least-squares approach," *Review of Financial studies*, vol. 14, no. 1, pp. 113–147, 2001.

[3] M. Kandemir, F. Li, G. Chen, G. Chen, and O. Ozturk, "Studying storage-recomputation tradeoffs in memory-constrained embedded processing," in *Design, Automation and Test in Europe, 2005. Proceedings*. IEEE, 2005, pp. 1026–1031.

[4] H. Koc, M. Kandemir, E. Ercanli, and O. Ozturk, "Reducing off-chip memory access costs using data recomputation in embedded chip multi-processors," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 224–229.

[5] 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 *Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec. 2011, pp. 468–474.

[6] D. B. Thomas, J. A. Bower, and W. Luk, "Hardware Architectures for Monte-Carlo based Financial Simulations," in *Proc. IEEE Int. Conf. Field Programmable Technology FPT 2006*, 2006, pp. 377–380.

[7] A. H. Tse, D. B. Thomas, K. H. Tsoi, and W. Luk, "Efficient Reconfigurable Design for Pricing Asian Options," *SIGARCH Comput. Archit. News*, vol. 38, no. 4, pp. 14–20, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1926367.1926371

[8] 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)*, 2012.

[9] X. Tian and K. Benkrid, "Implementation of the Longstaff and Schwartz American Option Pricing Model on FPGA," *Journal of Signal Processing Systems*, vol. 67, no. 1, pp. 79–91, 2012.

[10] F. Black and M. Scholes, "The Pricing of Options and Corporate Liabilities," *The Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, Mar. – Jun. 1973.

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

[12] M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, Jan. 1998.

[13] K. Hagita, H. Takano, T. Nishimura, and M. Matsumoto, "Reverse Generator MT19937," Fortran source code, June 2000, last access 2014-09-16. [Online]. Available: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/VERSIONS/FORTRAN/REVmt19937b.f

[14] 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 2010 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec. 2010, pp. 190–195.

[15] Xilinx, "XPower Estimator (XPE)," Jun. 2014, last access: 2014-09-16. [Online]. Available: http://www.xilinx.com/products/design_tools/logic_design/xpe.htm