FPGA-based Reconfigurable Computing for Pricing Multi-asset Barrier Options

Rahul Sridharan*, Gregg Cooke†, Kenneth Hill*, Herman Lam*, Alan George*

* NSF Center for High-Performance Reconfigurable Computing
Dept. of ECE, University of Florida
Gainesville FL, USA
e-mail: {sridharan, hill, hlam, george}@chrec.org

† UBS AG
Global CTO Applied Innovation Team
Chicago, IL
e-mail: gregg.cooke@ubs.com

Abstract*—Multi-asset barrier contracts are path-dependent exotic options consisting of two or more underlying assets. As the dimensions of an option increase, so does the mathematical complexity of a closed form solution. Monte Carlo (MC) methods offer an attractive solution under such conditions. MC methods have an $O(n^{-1/2})$ convergence rate irrespective of the dimension of the integral. However, such methods using conventional computing with CPUs are not scalable enough to enable banks to realize the potential that these exotic options promise. This paper presents an FPGA-based accelerated system architecture to price multi-asset barrier contracts. The architecture consists of a parallel set of Monte Carlo cores, each capable of simulating multiple Monte Carlo paths. Each MC core is designed to be customizable so that the core for the model (i.e., "model" core) can be easily replaced. In our current design, a Heston core based on the full truncation Euler discretization method is used as the model core. Similarly, we can use different payoff calculator kernels to compute various payoffs such as vanilla portfolios, barriers, look-backs, etc. The design leverages an *early termination* condition of "out" barrier options to efficiently schedule MC paths across multiple cores in a single FPGA and across multiple FPGAs. The target platform for our design is Novo-G, a reconfigurable supercomputer housed at the NSF Center for High-Performance Reconfigurable Computing (CHREC), University of Florida. Our design is validated for the single-asset configuration by comparing our output to option prices calculated analytically and achieves an average speedup of ranging from 123 to 350 on one FPGA as we vary the number of underlying assets from 32 down to 4. For a configuration with 16 underlying assets, the speedup achieved is 7134 when scaled to 48 FPGAs as compared to a single-threaded version of an SSE2-optimized C program running on a single Intel Sandy Bridge E5-2687 core at 3.1 GHz with hyper-threading turned on. Finally, the techniques described in this paper can be applied to other exotic multi-asset option classes, such as lookbacks, rainbows, and Asian-style options.

Keywords-FPGA-based reconfigurable computing; option-pricing; multi-asset Heston model; Monte Carlo simulation

I. INTRODUCTION

Financial institutions are increasingly exploring high performance computing (HPC) solutions to meet computational demands on various fronts. At the heart of any banking operation are increasingly complex mathematical models and the simulation of such models that aims to accurately model market conditions. One important area is the simulation of ***options pricing*** models that predict the future price (and hence potential risk) of every asset that a bank trades. Accelerating complex pricing models enables banks to explore a vast array of underlying assets (stocks, bonds, etc.), resulting in valuations that are most beneficial to their clients. Most important in determining the price (and its risk) is the range over which an asset's value is expected to change (i.e., volatility). Measuring current volatilities, predicting future volatilities, and balancing volatilities against one another are all activities that banks perform constantly. Interestingly, the often-used Black-Scholes model for calculating the price of a simple "vanilla" option treats volatility as a constant parameter. By doing so, it may expose the trader to arbitrage by other traders as long as he/she owns the option contract. Heston [10] proposed a solution to this problem by modeling the volatility in the Black-Scholes model with a random (or stochastic) variable.

The use of a stochastic volatility model becomes even more important for a variation on the vanilla option contract called a ***barrier*** option, a contract whose value is much cheaper than its vanilla counterpart because the trader is only paying for a small range of benefit. A barrier option is very sensitive to volatility. The slightest variation in the value of the underlying asset can cause the price of the option to tip over the barrier, rendering the option worthless. It is therefore important to model the volatility of the asset as accurately as possible. For this reason, the Heston model is often used with barrier options pricing.

Finally, investors in recent years have found ***multi-asset barrier options*** to be particularly attractive because they give the investor a single relatively cheap contract that captures the relationship among related sets of assets and offers some protection against the complicated volatility relationships that make the risk of the contract uncertain. Their complex covariance structure makes the Heston stochastic volatility model particularly suitable as a means to model multi-asset barrier options.

Note that the Heston layered with a barrier option contract has a closed-form solution. However, the additional mathematical complexity is enough to justify the use a Monte Carlo method with the Heston model instead. Furthermore, no closed-form solution has been found when multiple underlying assets are used to construct the barrier option contract. In this case, the mathematics change from using single variables to represent the asset value and volatility, to using matrices of parameters for a set of assets,

related through a correlation matrix. The use of Monte Carlo methods in such cases is required. However, the slow processing times of these methods make them ineffective in meeting market constraints. Research in deriving a reliable closed-form solution continues

In this paper, we present a high-performance reconfigurable computing (RC) solution to a business-relevant financial application that prices multi-asset barrier options based on the Heston stochastic volatility model using Monte Carlo methods. By greatly accelerating the valuation of multi-asset barrier options, banks will be enabled to manage sophisticated contracts more flexibly and precisely. Also, novel financial products can be a reality if the limits to valuating the contract are relaxed. For example, investors would be able to purchase barrier options contracts on indices of their own devising, or options contracts that simulate variations of popular indices or ETFs (exchange-traded funds).

The remainder of the paper is organized as follows. Section II presents the related research in using accelerators for computational finance models and specifically the Heston model. Section III reviews the Heston stochastic volatility model and outlines an algorithm for pricing multi-asset barrier options.

Section IV describes the design of an FPGA-based system architecture and a novel design for mapping a system of stochastic differential equations (SDEs) to this architecture. The architecture consists of a parallel set of MC cores, each capable of simulating multiple MC paths. An important feature is that each MC core is designed to be ***customizable*** in terms of the MC "model" to be used (e.g., in our current implementation, a Heston core based on the full truncation Euler discretization method is used). Similarly, it can be customized using different payoff calculator kernels to compute various payoffs such as vanilla portfolios, barriers, lookbacks, rainbows, Asian-style options etc. Finally, the design leverages an early termination condition of "out" barrier options to efficiently schedule MC paths across multiple cores in a single FPGA and across multiple FPGAs.

In section V, we present our experimental results, including an analysis with respect to a validation of the model configured to price single-asset options as well as performance achieved. The target platform is Novo-G, a reconfigurable supercomputer at the NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida. Novo-G is an FPGA-centric computer which features almost 400 FPGAs in 2012. Our design achieves an average speedup ranging from 123 to 350 on one Stratix IV FPGA as we vary the number of underlying assets from 32 down to 4. For a configuration with 16 underlying assets, the speedup achieved is 7134 when scaled to 48 FPGAs as compared to an SSE2 optimized single-threaded C program running on a single Intel Sandy Bridge E5-2687 core at 3.1 GHz with hyper-threading turned on. Currently, the long valuation times of exotic contracts like multi-asset barriers places a limit on how many such contracts can be sold. Such contracts could be more vigorously marketed if the time to valuate them were reduced significantly. Our design demonstrates that this is indeed possible.

Finally, Section VI concludes this paper, providing a summary of our work and a discussion on future work.

II. RELATED RESEARCH

In recent years, there has been much work in using FPGA-based reconfigurable computing to accelerate the performance of computational finance applications. For example, Zhang et al. developed an interest rate model to price derivatives using the Brace, Gatarek, and Musiela method [25]. Weston el al. used a High-Level Synthesis (HLS) tool to accelerate the computation of portfolios of tranched credit derivatives [24].

In particular, FPGAs have been proven effective in accelerating Monte Carlo methods for a variety of financial processes [11, 15, 17, 19, 20, 21]. Using MC simulation, Kaganov et al. used the copula model to price credit derivatives [11]. Maxeler and J.P Morgan used MC simulation to accelerate the computation of credit and interest rate derivatives. A full portfolio of 480 trades was simulated with a 284 speedup observed against a single-core CPU [15]. Thomas and Luk simulated loan portfolios for credit risk avoidance using an HLS tool Handle-C [20].

More related to our work, Morris and Aubury used the HLS tool HyperStreams to develop a hardware architecture using the Black Scholes model for European options pricing [17]. A speedup varying from 11 to 146 was recorded using a Virtex IV family FPGA using different data types and widths in the datapath. The baseline used for comparison was an optimized C++ code running on a single 2.5 GHz Opteron core. As mentioned earlier, one of the drawbacks of the Black Scholes model is that it treats volatility as a constant parameter. Our work is based on the Heston model [10], which models the volatility in the Black-Scholes model with a random (or stochastic) variable.

In [21], Tian et al. described the use of Monte-Carlo simulation for option pricing using the GARCH model. The GARCH model extends the Black Scholes model by simulating the volatility in a stochastic environment. The target platform was the Maxwell reconfigurable computer at the University of Edinburgh, which contains 64 Virtex IV family FPGAs in 32 nodes. Each of the nodes contains two FPGAs with a 2.8 GHz Xeon processor. The final result of a simulation is the average of all the Monte Carlo paths. Tests were performed using 1, 4, 8, 16, and 32 nodes resulting in a constant speedup of ~340 compared against an equivalent software implementation. Our work extends this approach by having the ability to price *barrier* options.

Schryver et al. used an HLS tool called VisualHDL to implement a Heston model simulating single asset barrier options [19]. A Virtex V family FPGA outperformed a 2 GHz Core 2 Duo processor by a factor of 21. Our approach extends this further by simulating *multi-asset* barrier options using the Heston model algorithm described in [23].

In [22], Tse el al. described their work on multi-asset option pricing using a quadrature method. A Virtex IV family FPGA was used to simulate three underlying assets. A performance gain of ~25x was observed against a 3.6 GHz

Pentium 4 processor. The quadrature method runtime increases exponentially with the number of assets, whereas the Monte Carlo method increases linearly.

III. THE MULTI-ASSET BARRIER OPTIONS ALGORITHM

The Heston stochastic volatility model is central to the algorithm we will use to price multi-asset barrier options. In this section, we briefly review the single-asset Heston model and its extensions to support multiple assets, followed by an outline of an algorithm used to price multi-asset barrier options.

A. Heston model

As mentioned, unlike the Black-Scholes model, the Heston models volatility as a stochastic process. The evolution of the underlying asset price S_t and the variance process V_t is described by the Stochastic Differential Equations:

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

$$dV(t) = -\kappa(V(t) - \theta)dt + \omega\sqrt{V(t)}dW_v(t) \quad (2)$$

where, r is the risk-free interest rate, κ is the rate of mean-reversion of the variance, θ is the long-term mean variance and ω is the volatility of volatility. The terms dWs and dWv are independent Brownian motions which drive the two processes with a correlation co-efficient ρ between them, which is incorporated into ω. The Brownian motions are obtained from random samples of a standard normal distribution.

Different values for ρ affect the skewness of an asset's log-return distribution. The volatility of volatility, ω, affects the peak of this distribution and a higher ω means the volatility is more volatile. K represents the degree of volatility clustering. Thus, it is seen that the parameters of the Heston model affect the implied volatility and can produce different distributions. This overcomes shortcomings of the Black-Scholes model leading to a more accurate pricing of options. [16] provides a detailed discussion on the Heston model and its computation and calibration.

Path-dependent derivatives like barrier options, constructed using multiple underlying assets, are best priced using Monte Carlo methods. MC simulations require efficient discretization of the continuous-time Heston SDEs. The Euler-Maruyama scheme [12] discretizes the continuous-time functions at finite time intervals, simulating the stock and variance process at these discrete intervals. Layering this discretization scheme with the full truncation fix proposed by Lord et al [12] addresses the issue of the volatility process going negative. The scheme produces the least discretization bias of all the Euler schemes suggested by Lord and is given as:

$$S(t+\Delta) = S(t) + r(t)\Delta + \sqrt{\hat{V}(t)^+}\sqrt{\Delta}Z_x \quad (3)$$

$$\hat{V}(t+\Delta) = \hat{V}(t) + \kappa\left(\theta - \hat{V}(t)^+\right)\Delta + \omega\sqrt{\hat{V}(t)^+}Z_v\sqrt{\Delta} \quad (4)$$

where x+ = max(x,0) and equations (3) and (4) represent the asset and volatility motions respectively.

The main disadvantage of the Euler scheme is the fine level of discretization required to produce sufficiently accurate results. The bias introduced increases as the number of full truncation projections applied to the variance process increases which is the case when the Feller condition is violated [3]. The Quadratic Exponential (QE) scheme proposed by Anderson [3] produces a smaller discretization bias than all the Euler schemes but would result in a complex hardware design. Thus, the Euler scheme is used in this design. However, the system architecture (described in Section IV) for multi-asset options is independent of the underlying discretization technique used and would support future hardware designs for the QE scheme.

B. A multi-asset Heston model

Extension of the single-asset model to multiple dimensions requires the simulation of a system of correlated single-asset SDEs. For a d-dimensional system, we have

$$\frac{d\,S_i(t)}{S_i(t)} = \mu_i\,dt + \sigma_i\,dX_i(t), i = 1, \ldots, d \quad (5)$$

The following additional parameters come into existence while describing a d-dimensional Heston model:

- The cross-correlation between ***d*** Geometric Brownian

The calibration of the multi-asset Heston model with respect to these cross-correlations is beyond the scope of this paper. References [7] and [23] provide a rigorous mathematical model for the calibration of the asset-asset correlation matrix using the full truncation and QE scheme, respectively. In this work, we make an assumption that the cross-correlation between the volatility processes is zero although an important area of research would be to study the effects of this correlation on the evolution of the system of SDEs.

The extension of Equations (3) and (4) to simulate a system consisting of multiple underlings leads us to:

$$S(t+\Delta) = S(t) + r(t)\Delta + \sqrt{\hat{V}(t)^+}\sqrt{\Delta}\varepsilon^s \quad (6)$$

$$\hat{V}(t+\Delta) = \hat{V}(t) + \kappa\,(\theta - \hat{V}(t)^+)\Delta + \omega\sqrt{\hat{V}(t)^+}Z_x\sqrt{\Delta}\varepsilon^v \quad (7)$$

where $\varepsilon^s = \sum_{j=1}^{d} A_{ij}Z_{\Delta j}$, for i= 1, 2, …, d. The matrix A is chosen to be the Cholesky square root of Σ with $AA^T = \Sigma$. The algorithm given in [7] to implement the multi-asset d-dimensional Heston model is as follows:

- Generate a set of Gaussian random numbers $z_1,.., z_d$ at each time step in the time grid
- Compute $\varepsilon_i^s = A\ x\ Z^T$ for each individual asset in the system
- For the variance process compute $\varepsilon_i^v = \rho_i \varepsilon_i^s Z_v \sqrt{1-\rho_i^2}$, where Z_v is an independent Gaussian random variable
- Simulate the next time step for the volatility and asset process according to (6) and (7), respectively

C. *Multi-Asset Barrier Options Pricing*

A barrier option is a contract whose payoff automatically drops to zero when the value of the simulated asset return passes through some pre-defined barrier level, or remains at zero until the value passes through a barrier. Barrier options are also much cheaper than their vanilla counterparts because the trader is only paying for a small range of benefit. Depending on the position of the barrier relative to the initial stock price they can be classified as either "up" or "down" barriers. Upon hitting the barrier an option can either be "knocked out" or "knocked in". A down and out barrier is denoted by the payoff equation:

$$1\{T(b) > T\}\ (S_{(T)} - K)^+, \tag{8}$$

where K is the strike price of the option and $1\{T(b) > T\}$ is an indicator function equal to 1 if the barrier is not hit $\{S(t_i) < B\}$ before the option expires and is 0 otherwise. With multiple underlyings we modify the barrier Equation (8) to calculate to price Worst-of-N call options where N is the number of underlyings.

$$S_{(T)} = \min\{S_1(T), S_2(T), \ldots S_N(T)\} \tag{9}$$

The basic algorithm for pricing multi-asset barriers is as follows:

- Calculate the next value of the option for each individual asset in a system based on (6) and (7)
- Test it against the conditions of the barrier such that all assets in system satisfy the condition
- If barrier is breached, value is zero for rest of path
- Repeat for multiple Monte Carlo paths
- Average the paths to determine the value of the barrier using the payoff Equation (8) and (9)

IV. FPGA-BASED RECONFIGURABLE COMPUTING SOLUTION

This section presents an FPGA-based system architecture to accelerate the pricing of multi-asset barrier option contracts and a novel design for mapping a system of Heston SDEs to this architecture. As shown in Figure 1, the architecture consists of a parallel set of Monte Carlo cores, each capable of simulating multiple Monte Carlo paths. Each MC core is designed to be ***customizable*** so that the core for the model (i.e., "model" core) and/or the "payoff calculator kernel" can be easily replaced. In our current design, a Heston core based on the full truncation Euler discretization method is used is as the model core. If, for example, a smaller discretization bias is required, we can replace the model core with a Heston core based on the QE

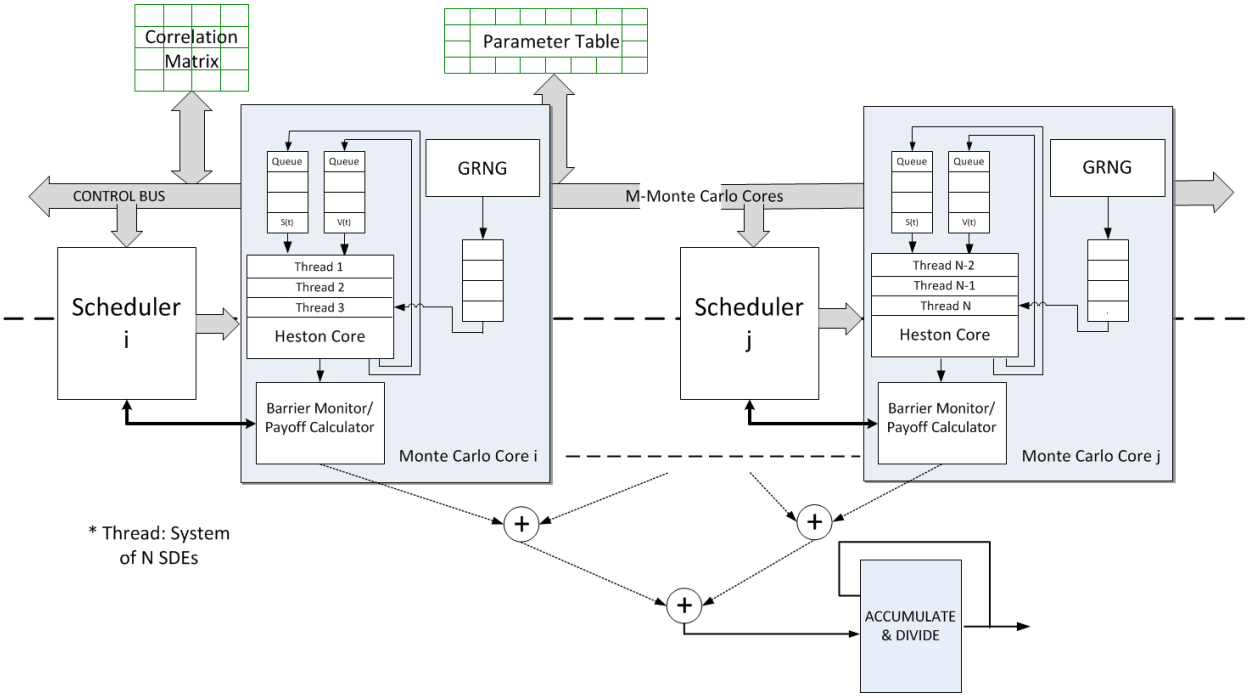

Figure 1. Overall system architecture: multi-asset barrier options pricing

scheme. Similarly, we can use different payoff calculator kernels to compute various payoffs such as vanilla portfolios, barriers, look-backs, etc. Each MC core is associated with a scheduler, which monitors the simulation to perform *early termination* and run-time scheduling of the MC threads (to be described later in this section).

A d-dimensional multi-asset barrier model, as described by the algorithm in Section II, involves the forward simulation of a system of SDEs $S_i(t + \Delta)$ and $V_i(t + \Delta)$ for i=1,2…d. Parallelization of these forward motion simulations is achieved at two levels:

1. Multiple underlying assets belonging to a single system of SDEs are time-multiplexed within a Heston core. In our architecture, such a system of SDEs is defined as a *thread*.
2. Multiple systems of SDEs or *threads* are simulated independently across all the cores of a single FPGA and across multiple FPGAs.

In the section, we describe the design of key components of an MC core, followed by a discussion on our techniques for parallelization and scheduling.

Model core.

The key component in an MC core is the model core. As mentioned, a *Heston core* based on the full truncation Euler discretization method is used as the model core in the current implementation. Figure 2 shows the design and data flow of such a Heston core to perform the forward simulation of Equations 6 and 7 described in Section III. The Heston core has a pipelined architecture simulating the asset and volatility processes as scheduled by the scheduler at each stage of the pipeline. The scheduling process itself is abstracted away from the MC core making it modular enough to support future extensions or be replaced by other model cores.

The Heston core is further optimized to use a two's complement fixed-point representation. MATLAB's fixed-point toolbox [13] was used to perform simulations and determine the bit-widths at each stage for the required level of precision. This is compared against a double-precision floating-point representation to determine the error introduced by the fixed-point design. From these simulations we determined the bit-widths such that the fixed-point error introduced is kept lower than the discretization bias produced by the Euler full truncation scheme.

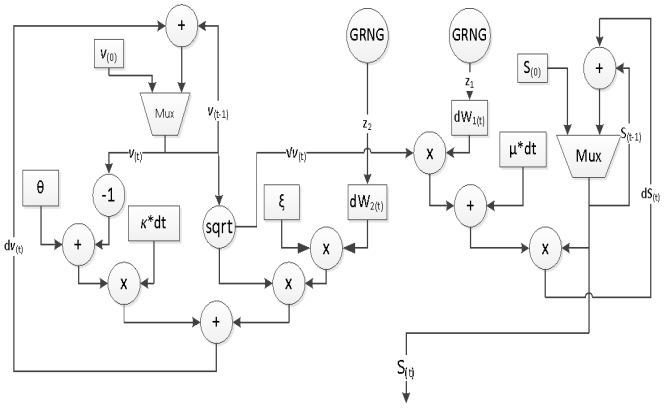

Figure 2. Full truncation Heston model core

Inputs to MC core.

An MC core has two inputs, a *correlation matrix* and a *parameter table*. The cross-correlations described in Section III are calibrated using techniques described in [7] and inputted to the MC cores through the correlation matrix. The resulting vector, ε^s (Equation 6 in Section II), is generated by computing the matrix product between a normally distributed random number vector and the correlation matrix. Each row of ε^s forms an input to one of the assets in a thread. The structure of a cross-correlation vector is shown in Figure 3, where z_1, z_2... z_d represent Gaussian random numbers and $\rho_{1,1}$, $\rho_{1,2}$…$\rho_{1,d}$ represent asset-asset cross-correlations.

The second input to an MC core is a parameter table. Each row of the parameter table represents input parameters corresponding to each of the asset in a thread. The parameter table is implemented as distributed Block RAMs in the FPGA and is also controlled by the scheduler.

Thread scheduler.

Each MC core is associated with a *thread scheduler*, which monitors the simulation to perform *early termination* and run-time scheduling of the Heston paths.

We define a system of multiple underlying assets and their corresponding volatility processes as a *thread*. Thus, the evolution of multiple-assets corresponds to the evolution of threads across multiple MC cores, with each thread representing a single Monte Carlo path. Given a Heston core with a pipeline depth of M and a thread with N assets, we can simulate independently M/N threads per core (the depth M is adjusted so as to fit an appropriate number of threads). For an FPGA implementing K cores, we can thus simulate K × (M/N) threads or MC paths at any given time. For a configuration with 4 underlying assets, we have K= 36, M= 16, and N= 4. Thus, we are able to simulate 144 threads or MC paths at a given time instant.

Early termination ensures that a new thread begins its

$$\begin{bmatrix} Z_1\times\rho_{1,1} + Z_2\times\rho_{1,2} + Z_3\times\rho_{1,3} + \ldots + Z_d\times\rho_{1,d} \\ Z_1\times\rho_{2,1} + Z_2\times\rho_{2,2} + Z_3\times\rho_{2,3} + \ldots + Z_d\times\rho_{d,2} \\ Z_1\times\rho_{3,1} + Z_2\times\rho_{3,2} + Z_3\times\rho_{3,3} + \ldots + Z_d\times\rho_{d,3} \\ \vdots \\ Z_1\times\rho_{d,1} + Z_2\times\rho_{d,2} + Z_3\times\rho_{d,3} + \ldots + Z_d\times\rho_{d,d} \end{bmatrix}$$

Figure 3. Structure of the cross-correlation vector ε^s

simulation immediately upon the termination of one of the current executing threads in a core. The scheduler initiates new thread in the pipeline under either of the following two conditions:

1. A thread reaches its maturity period T without breaching the barrier.
2. One or more assets constituting a thread breach the barrier at time instant *t*.

Figure 4 illustrates the data structure maintained by the thread scheduler. An associative array maps the number of time steps simulated to its corresponding thread in the MC core. The *barrier_hit* signal indicates to the scheduler if the barrier has been breached by an asset in any of the scheduled threads. The *path_counter* register is updated when a new thread is scheduled in a core. This register keeps track of the total number of MC paths simulated and initiates transfer to the host once a desired number of paths have been simulated.

Discrete-time barrier monitoring and pricing.

The output from the Heston core is the forward simulation of an asset value at a certain time instance t. This output is the input to the *barrier monitor/payoff calculator* which checks for the condition when the barrier is breached and calculates the resulting payoff. The barrier monitor component is gated, which enables it to be switched on or off as defined by an application. For example, for a knock-out barrier, when switched on, the monitoring unit asserts a corresponding *barrier_hit* signal in the event of a breach. A breach by any asset in the thread triggers the scheduler to initiate a new thread in the pipeline (*early termination*). In-stream signaling is used to map an asset to its corresponding thread within the barrier monitor and payoff calculator.

Currently, the payoff calculator is designed for vanilla portfolio payoff and the worst-of-N payoff calculations. Equations 8 and 9 in Section III define the functions of the worst-of-N payoff calculator for a knock-out barrier. The calculator is enabled by the scheduler when a thread completes its execution. Again, the MC core is configurable in that different payoff calculators (e.g., from a library) can be used to compute various payoffs such as vanilla portfolios, barriers, look-backs, etc.

Gaussian random number generator (GRNG).

Considerable amount of research has gone into developing efficient hardware-based Gaussian random number generators (e.g., [2], [4], [5], [14]). We adopt a design as outlined in [6] for generating the hardware GRNG. Two inversion-based random number generators, each capable of generating one normally distributed random number per clock cycle, are incorporated into each MC core. The inverse CDF (ICDF) of a uniformly generated random number (Ψ^{-1} (U)) is evaluated to produce a pseudo-random number with the required Gaussian distribution. Piecewise polynomial approximation and hierarchical segmentation

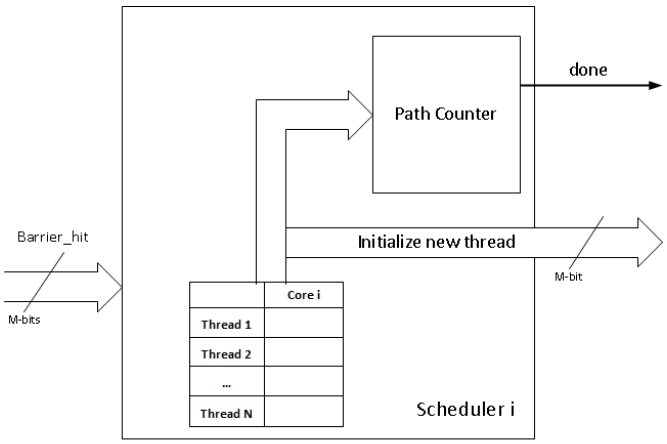

Figure 4. Thread scheduler

using pre-computed lookup tables are used to compute the ICDF of the uniform random number in hardware.

V. Experiments and Evaluation

In this section, the design of our FPGA-based reconfigurable computing solution for pricing multi-asset barrier options is evaluated in two ways:

1. Validating the design and implementation of the Heston model by comparing its output to approximated single-asset as well as multi-asset option prices calculated analytically
2. Comparing the performance of the FPGA implementation against an SSE2 optimized single-threaded C program

The target platform for our experiments is the Novo-G reconfigurable supercomputer, housed at the NSF CHREC Center for High Performance Reconfigurable Computing, University of Florida [8, 9]. Novo-G features almost 400 FPGAs in 2012, which are supported by quad-core Nehalem Xeon E5620 processors, GTX-480 GPUs, 20Gb/s non-blocking InfiniBand, GigE, and ~2TB of total RAM.

Our design currently targets the Stratix IV E530 FPGAs on Novo-G. Being embarrassingly parallel, we partition our design evenly based on the number of paths executed in each FPGA. Outputs from different FPGAs are gathered in a host machine to calculate the value of the discounted payoff.

A. Design Validation and Verification

The ability of our solution to produce market-consistent results is an important factor in determining its business relevance. We perform a series of tests to determine the accuracy of the underlying Heston model as well as the multi-asset barrier pricing model.

Single-asset Heston model.

In order to validate the Heston model, we configure our application to compute option prices for a single underlying asset. We first calibrate the Heston model parameters using a non-linear least-square fit of the parameters to observed

TABLE I. Single-asset Heston model verification

Spot Price	Strike Price	Initial Vol.	Time to Mature	True Value	Hardware Output
100	100	0.04	1 year	10.3009	10.2923
123.4	123.4	0.01994	1 year	13.85	13.8211
100	100	0.04	10 years	13.0847	13.0915
100	100	0.09	5 years	34.9998	34.9814
100	100	1	1 year	39.725	40.3231

market data [18]. Then we generate a parameter table containing exactly similar parameters for all the underlying assets. The correlation matrix we use is an identity matrix which ensures that the simulation of each underlying asset acts as an independent MC path. The payoff function is configured to compute a vanilla call option. We validate our model using 5 test cases as shown in Table I. The table compares the hardware output against the approximated or true value of the option (calculated analytically). A degree of bias can be expected from the hardware output resulting from the discretization error of the full truncation scheme. We consider the model to be validated as long as this error is within the limit of the implied volatility. Replacing the underlying discretization scheme with more sophisticated techniques such as the QE scheme, and improving the parameter calibration algorithm, will result in the Heston model pricing options with a higher degree of accuracy.

Multi-asset Heston Model.

The calibration of a Heston model is made difficult by its bivariate nature; the difficulty is compounded by expanding the model to multiple assets. The correlation between assets becomes important and measuring this correlation is a separate and challenging task. In this paper, we assume that the asset-to-asset correlations are already available and that there is no correlation between volatilities of different assets. This is a useful simplifying assumption but it adds a degree of error to our system; [7] describes a method by which these correlations can be calculated more precisely.

Validation of a Heston model, by which the output is compared to known prices under similar input parameters, is easily done if the model is used to calculate vanilla options, or even simple barrier options, as these prices are obtainable from a variety of sources. More exotic options, like our multi-asset barrier contract, are far more difficult to validate because there is no easily observable market for them. These contracts are customized at the time of sale for each investor and so do not show up in available market data feeds. However, they can be approximated using analytical solutions for similar contracts. See [1] for one such approximation algorithm, which uses a tree of vanilla options to simulate a basket option. The current hardware design is verified against an equivalent software baseline. However, the validation of the multi-asset option feature is left as future work.

B. Performance evaluation on Novo-G.

We evaluate the performance of our FPGA design in terms of speedup against an SSE2 optimized C-based program running on an Intel Sandy Bridge E5-2687 processor at 3.1 GHz with hyper-threading turned on. Our baseline includes a single-threaded version running on a single core as well as a multi-threaded version running on a server node consisting of two 8-core E5-2687 processors. A multi-asset worst-of-N barrier contract is run for 1,000,000 paths as the baseline. Furthermore, the Box-Muller transform is used to generate normally distributed random numbers in software. Speedup evaluations are performed first on a single Stratix IV FPGA and then scaled up to multiple FPGAs on Novo-G. Our FPGA-based design is compiled for a clock frequency of 125 MHz

Four different design configurations (4, 8, 16, and 32 underlying assets) are considered while evaluating performance. We also consider the effect of increasing time to maturities while pricing multi-asset barriers. The options are priced considering a discrete time grid Δt of 1 day. Such a fine level of discretization is required to minimize the bias produced by the Euler scheme as well as to compute discrete-time barrier contracts.

Increasing the number of underlying assets results in a larger correlation matrix and parameter table. This leads to an increase in the number of computations while evaluating the asset-asset cross correlations as defined in Section III, making the design bound by the logic resources on an FPGA. In general, an ***O(m×n)*** relationship exists between the number of underlying assets and the number of multiplications required to calculate these cross-correlations, where ***m*** is the number of MC cores and ***n*** is the number of

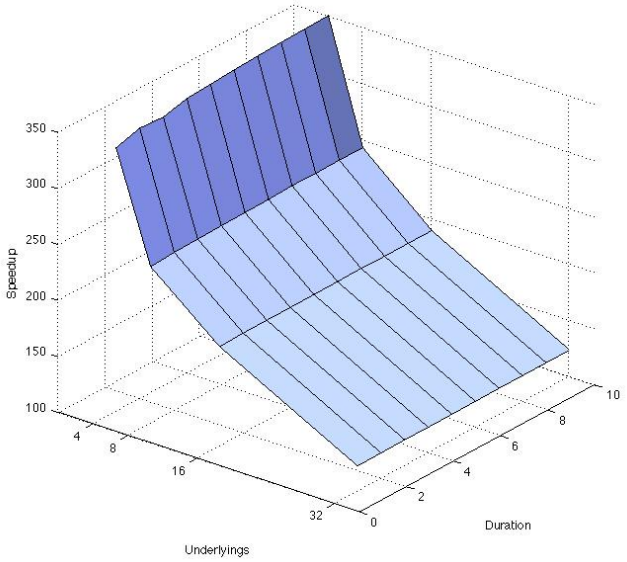

Figure 5. Single FPGA Spedup : Speedup vs. Duration vs. Underlyings

TABLE II. Single-FPGA speedup for different asset configurations

No. of Underlyings	Threads /Core	Cores /FPGA	Speedup† /FPGA
4	4	36	350
8	3	26	242
16	2	20	189
32	2	12	123

† w.r.t. software baseline executed for duration = 10 years

underlying assets. This results in a tradeoff between the number of assets in a system and the total number of cores in an FPGA and hence the overall speedup achieved. Table II shows the number of threads per MC core and the total number of cores for each of the four configurations.

Figure 5 shows the speedup comparison of a single FPGA against the single-threaded version running on one CPU core (*speedup vs. duration (in years) vs. # underlying assets*). Note that for duration = 10 years, the speedup reduces from **350** to **123** as we increase the number of underlying assets in a thread from 4 to 32. This design is limited by the number of DSP multipliers in the FPGA which is determined by the number of underlying assets. The number of cores that can fit in an FPGA is reduced with an increase in the number of assets. For an average case of 16 assets per thread with a maturity period of 10 years, the observed speedup on a single Stratix IV FPGA is **189**.

Each Monte Carlo path simulation is an independent process within the model core, governed by a dedicated scheduler. This design choice makes the application embarrassingly parallel, amenable to both partitioning and scaling across multiple FPGAs. Figure 6 shows the speedup achieved as compared to a single CPU core when the designed is scaled across multiple FPGAs in a server and across multiple servers of Novo-G. The average case of 16 assets per thread with a maturity period of 10 years is used. It is seen that scaling the design across 48 FPGAs results in an overall speedup of **7134**, as compared to software run time of 4415s.

The non-linearity associated with this increase in speedup is attributable to the synchronization overheads associated with memory transfers and post processing. The large degree to which these overheads are seen in the presented results is attributable to the adequate problem size on a single FPGA being reduced to a small problem when partitioned across multiple FPGAs (i.e., total overhead time as a percentage of total execution time is much larger for 16 FPGAs as compared to a single FPGA).The datasets we use for speedup evaluation represent single option values. Using larger datasets representing multiple multi-underlying options as well as multiple runs of the model would result in a smaller impact of these overheads and a near linear increase in speedup.

Figure 7 shows the speedup achieved speedup by a single server of Novo-G consisting of 16 FPGAs when compared to a multi-threaded baseline running a single

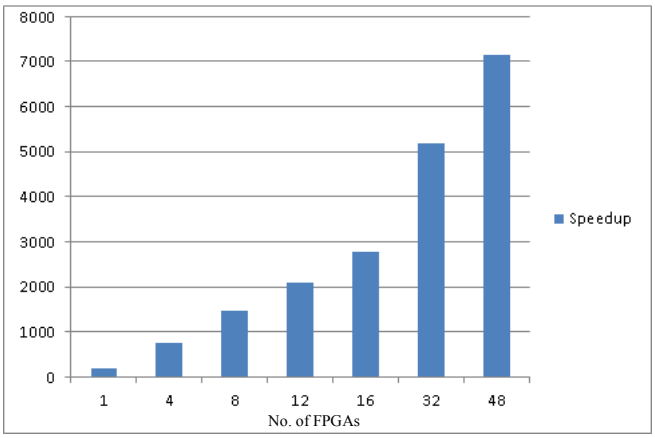

Figure 6. Speedup: Scaling design to multiple FPGAs

server consisting of 16 E5-2687 cores. As discussed, the speedup achieved reduces with increasing the number of underlying assets. Within each underlying asset, the observed speedup increases with the duration. This can be attributable to the smaller problem size for smaller values of duration. As the duration increases, the amount of time spent by the FPGA performing useful work increases compared to the communication overhead. The non-linearity associated with scaling reduces as the number of underling assets increases.

The speedup levels observed would result in an increased ability to explore various types of multi-asset contracts during an intraday trading period. Increasing the FPGA computational density and reducing associated parallelization overheads would further improve performance, allowing us to compute higher dimension options in a relatively flexible manner.

VI. CONCLUSION

In this paper we describe a hardware design of Heston's stochastic volatility model, computed using the Monte Carlo techniques, and use it to calculate multi-asset barrier options. We resolve several of the common challenges that arise when using this type of calculation, including implementing an effective discretization scheme, managing multiple assets as a single collective option, and modularizing the payoff function and the Monte Carlo engine to accommodate various flavors of options. This last feature is particularly interesting to financial engineers because the class of multi-asset options is still considered "exotic" and hence must be customized for each investor who wishes to purchase such a contract. A library of payoff functions would expand our system to many other classes of exotic options.

The speedup achieved by the hardware design for a single FPGA ranges from **123** to **350** depending on the configuration for the number of underlying assets. We show that a tradeoff exists between the number of underlying assets and the speedup achieved, performance being bound by logic resources as we increase the number of underlying

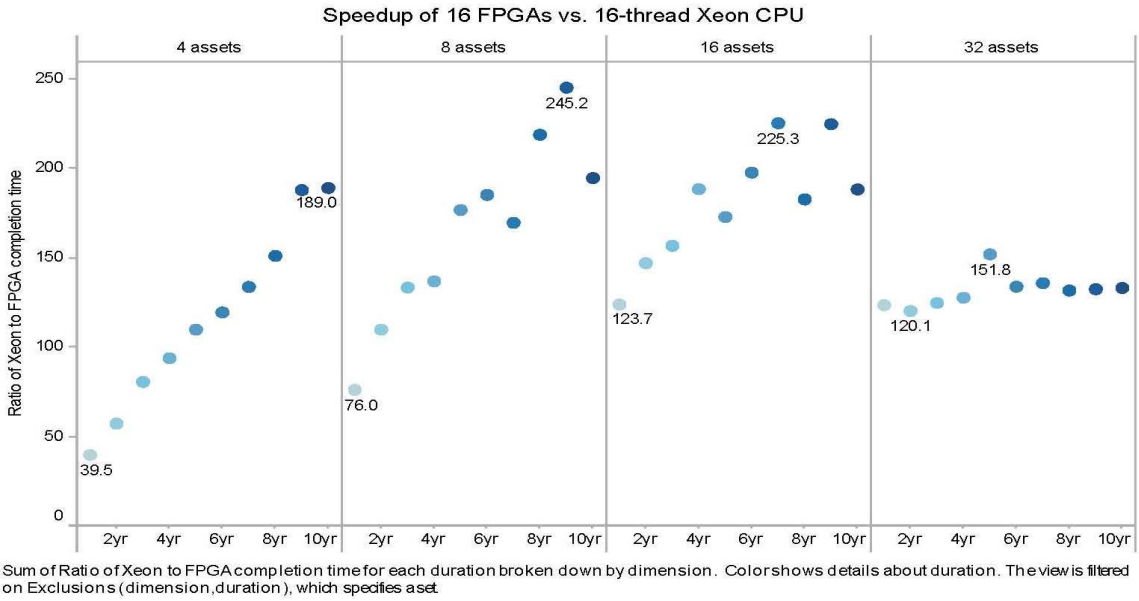

Figure 7. Speedup: 16 FPGAs vs 16 –threaded Xeon CPU

assets. However, a machine like Novo-G enables the design to be scaled to a much larger number of FPGAs. We achieved a speedup of **7134** when a design with 16 underlying assets is scaled to 48 FPGAs.

Currently, the long valuation times of exotic contracts, like multi-asset barriers, places a limit on how many such contracts can be sold. Such contracts could be more vigorously marketed if the time to valuate them were reduced significantly. By greatly accelerating the valuation of multi-asset barrier options, banks will be enabled to manage sophisticated contracts more flexibly and precisely. Also, novel financial products can be a reality if the limits to valuating the contract are relaxed. Our design demonstrates that this is indeed possible.

ACKNOWLEDGMENT

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422.

REFERENCES

[1] C. Alexander and A. Venkatramanan, "Analytic Approximation for Multi-Asset Option Pricing," Math. Finance, DOI: 10.1111/j. 1467-9965.2011.00481.x.

[2] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "On the Efficiency and Accuracy of Hybrid Pseudo-Random Number Generators for FPGA-Based Simulation," IEEE Int. Symp. on Parallel and Distributed Process., Miami, FL, April 14-19, 2008.

[3] L. Andersen, "Efficient Simulation of the Heston Stochastic Volatility Model," Social Science Research Network, Dec. 12, 2006.

[4] S. Bank, P. Beadling, and A. Ferencz, "FPGA Implementation of Pseudo Random Number Generators for Monte Carlo Methods in Quantitative Finance," 2008 Int. Conf. on Reconfigurable Computing and FPGAs, Cancun, Mexico, Dec. 3-5, 2008.

[5] S. Chandrasekaren and A. Amira, "High Performance FPGA Implementation of the Mersenne Twister," 4th IEEE Int. Symp. on Electronic Design, Test and Applications, Hong Kong, China, Jan. 23-25, 2008.

[6] R. Cheung, D. Lee, W. Luk, and J. Villasenor, "Hardware Generation of Arbitrary Random Number Distributions from Uniform Distributions via the Inversion Method," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 8, pp. 952-962, Aug., 2007.

[7] G. Dimitroff, S. Lorenz, and A. Szimayer, "A Parsimonious Multi-Asset Heston Model: Calibration and Derivative Pricing," Int. J. of Theoretical and Applied Finance, vol. 14, no. 8, pp. 1299-1333, 2011.

[8] A. George, H. Lam, and G. Stitt, "Novo-G: At the Forefront of Scalable Reconfigurable Supercomputing," Computing in Sci. and Eng., vol. 13, no. 1, pp. 82-86, 2011.

[9] A. George, H. Lam, A. Lawande, C. Pascoe, and G. Stitt, "Novo-G: A View at the HPC Crossroads for Scientific Computing," Proc. of the Int. Conf. on Eng. of Reconf. Sys. and Algs. (ERSA), NV, 2010.

[10] S. Heston, "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options," The Review of Financial Studies, vol. 6, no. 2, pp. 327-343, 1993.

[11] A. Kaganov, P. Chow, and A. Lakhany, "FPGA Acceleration of Monte-Carlo based Credit Derivative Pricing," Int. Conf. on Field Programmable Logic and Applications, Heidelberg, Germany, Sept. 8-10, 2008.

[12] R. Lord, R. Koekkoek, and D. Dijk, "A Comparison of Biased Simulation Schemes for Stochastic Volatility Models," Quantitative Finance, vol. 10, no. 2, pp. 177-194, 2010.

[13] MathWorks. (2012). Fixed-Point Toolbox [Online]. Available: http://www.mathworks.com/products/fixed

[14] J. McCollum, J. Lancaster, D. Bouldin, and G. Peterson, "Hardware Acceleration of Pseudo-Random Number Generation for Simulation Applications," Proc. Of the 35th Southeastern Symp. on Systems Theory, Morgantown, WV, March 16-18, 2003.

[15] O. Mencer and S. Weston, "Computational Acceleration of Credit and Interest Rate Derivatives," Global Derivatives Trading and Risk Management, Paris, France, May, 2010.

[16] N. Moodley, "The Heston Model: A Practical Approach," Faculty of Science, Univ. of the Witwatersrand, Johannesburg, South Africa, 2005.

[17] G. Morris and M. Aubury, “Design Space Exploration of the European Option Benchmark using Hyperstreams,” Int. Conf. on Field Programmable Logic and Applications, Amsterdam, Netherlands, Aug. 27-29, 2007.

[18] M. Ncube. (2010). Heston Model Calibration and Simulation [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/29446-heston-model-calibration-and-simulation

[19] C. Schryver, I. Shcherbakov, F. Kienle, N. When, H. Marxen, A. Kostiuk, and R. Korn, “An Energy Efficient FPGA Accelerator for Monte Carlo Option Pricing with the Heston Model,” 2011 Int. Conf. on Reconfigurable Computing and FPGAs, Cancun, Mexico, Nov. 30-Dec. 2, 2011.

[20] D. B. Thomas and W. Luk, “Credit Risk Modeling using Hardware Accelerated Monte-Carlo Simulation,” 16th Int. Symp on Field-Programmable Custom Computing Machines, Palo Alto, CA, April 14-15, 2008.

[21] X. Tian, K. Benkrid, and X. Gu, “High Performance Monte-Carlo Based Option Pricing on FPGAs,” Eng. Letters, vol. 16, no. 3, EL_16_3_24, Aug. 2008.

[22] A. Tse, D. Thomas, and W. Luk, “Option Pricing with Multi-Dimensional Quadrature Architectures,” 2009 Int. Conf. on Field-Programmable Tech. , Sydney, Australia, Dec. 9-11, 2009.

[23] W. Wadman, “An Advanced Monte Carlo Method for the Multi-Asset Heston Model,” M.S. thesis, Inst. of Appl. Math. , Delft Univ. of Tech., Delft, Netherlands, 2010.

[24] S. Weston, J. T. Marin, J. Spooner, O. Pell, and O. Mencer, “Accelerating the Computation of Portfolios of Tranched Credit Derivatives,” 2010 IEEE Workshop on High Performance Computational Finance, New Orleans, LA, Nov.14, 2010.

[25] G. Zhang, P. Leong, C. Ho, K. Tsoi, C. Cheung, D. Lee, R. Cheung, and W. Luk, “Reconfigurable Acceleration for Monte Carlo based Financial Simulation,” 2005 IEEE Int. Conf. on Field-Programmable Tech., Dec. 11-14, 2005.