2009 International Conference on Computational Science and Engineering

Pricing the American Option using Reconfigurable Hardware

Chris Wynnyk
Electrical Engineering Department, RPI,
110 8th Street, Troy, NY 12180, USA
chriswynnyk@gmail.com

Malik Magdon-Ismail
Computer Science Department, RPI,
110 8th Street, Troy, NY 12180, USA
magdon@cs.rpi.edu

***Abstract*— We present a novel reconfigurable hardware architecture for accelerating American option pricing using the Binomial Lattice algorithm. The architecture provides double precision floating point pricing, evaluating up to $N = 64,000$ time steps in the binomial lattice. Advanced memory management techniques and optimized control logic allow for 4-way parallelism on a single-asset evaluation. These techniques achieve a $73\times$ speedup over an optimized CPU implementation, and a considerable improvement over the best previous reconfigurable hardware implementation. A significant advantage of our approach is that the speed up is on a per asset basis whereas all previous approaches on FPGA and GPU architectures achieve their speed up by evaluating many assets in parallel.**

I. Introduction

This research addresses the acceleration of American option pricing. American options are interesting from a mathematical perspective because they have no closed form solution [1]. This makes them difficult to price both quickly and accurately. The time delay between changes in the underlying stock value and price of the American option creates a potential arbitrage opportunity. Low-latency, accurate pricing can give a significant competitive advantage to arbitrage-based fast trading strategies.

Typical algorithms discretize the underlying continuous pricing equations into N time steps, allowing for a tradeoff between computation time and precision through the choice of N. Real-life option evaluation requires an N = 10,000 to 50,000 steps for adequate precision. For a 2-year option, this corresponds to a discrete point roughly every 15 minutes. All known previous work is severely limited in this respect, reporting a maximum $N = 1,024$ timesteps [2], [3]. This reduced precision is due to fundamental architecture and platform limitations of previous designs.

This paper contributes the first high-precision, accelerated architecture for pricing American options. The architecture provides double precision floating point pricing, evaluating up to $N = 64,000$ time steps in the binomial lattice. Advanced memory management techniques and optimized control logic allow for 4-way parallelism on a single-asset evaluation. These techniques achieve a $73\times$ speedup over an optimized CPU implementation, and a considerable improvement over the best previous reconfigurable hardware implementation. A significant advantage of our approach is that the speed up is on a per asset basis whereas all previous approaches on FPGA and GPU architectures achieve their speed up by evaluating many assets in parallel.

The outline of this paper is as follows: Section II presents the algorithm for pricing the American option. Section III presents our acceleration approach on reconfigurable hardware. Section IV presents implementation details for architecture, memory management, and control strategies. Section V presents results and compares against previous work. Section VI presents conclusions and areas for further research.

II. American Option Pricing

An American option is a financial instrument in which the owner has the right but not the obligation to buy (call option) or sell (put option) a stock for a specified price K (strike price) at time T (expiry). Without loss of generality, we will only consider the put option.

The value of the option is dependent on the price of the underlying stock. In a continuous model, the value of a stock in the risk neutral world is modeled as a geometric Brownian motion (1) with fixed drift r and volatility σ^2.

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1}$$

European options can only be exercised at the maturity T, whereas American options can be exercised at any time up to T. The European option can be priced directly using the Black-Scholes formula [4]. American options are more difficult to price. Pricing requires calculating an optimal exercise strategy, the stock price (at any given time) above which the option is exercised [1].

A. Recombining Binomial Lattice

Rather than a continuous model, the Binomial Lattice approach models the underlying stock price as a discrete binomial lattice. At each time step the stock price goes up by a factor of λ_+, or down by λ_-:

$$S(t+1) = \begin{cases} \lambda_+ S_t, & \text{with probability } \tilde{p}, \\ \lambda_- S_t, & \text{with probability } 1-\tilde{p}. \end{cases} \tag{2}$$

The lattice is indexed by time t and index i. The value of exercising the option is computed using the logarithm (3) to improve numerical stability. The value of holding (4) is the

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.496

discounted expected value of the future nodes. The value of the option is the maximum of exercising and holding:

$$v_{ex}(t,i) = K - S_0 * e^{i \log \lambda_+ \ + \ (t-i) \log \lambda_-}, \quad (3)$$

$$v_{hold}(t,i) = e^{-r\Delta t}\Big(\tilde{p}\ v_{opt}(t+1,i+1)\ + \ (1-\tilde{p})\ v_{opt}(t+1,i)\Big), \quad (4)$$

$$v_{opt}(t,i) = \max\left(v_{ex}(t,i),\ v_{hold}(t,i)\right). \quad (5)$$

Lattice nodes are evaluated backwards in time, starting from the boundary condition at expiry T (6). The final option value is $v_{opt}(0,0)$.

$$v_{opt}(T,i) = \max(K - S_{T,i}, 0) \quad (6)$$

The risk-neutral model is calibrated using real-world stock drift μ_R and volatility ${\sigma_R}^2$. Constraining the risk neutral variables (7) makes it possible to cache the exercise values v_{ex} at the start of each pricing. The risk-neutral binomial lattice parameters are then given by (8 - 10). Refer to [1] for algorithm details.

$$\lambda_- = \frac{1}{\lambda_+} \quad (7)$$

$$\lambda_+ = e^{\mu_R \delta t + \sigma_R \sqrt{\delta t}} \quad (8)$$

$$\lambda_- = e^{\mu_R \delta t - \sigma_R \sqrt{\delta t}} \quad (9)$$

$$\tilde{p} = \frac{e^{r\delta t} \ - \ \lambda_-}{\lambda_+ \ - \ \lambda_-} \quad (10)$$

This Binomial Lattice algorithm is well suited for implementation with reconfigurable hardware. The computation of v_{opt} at each node requires only two multiplies, an addition, and a compare. The algorithm requires $O(N^2)$ node evaluations and $3N$ memory space (including the v_{ex} cache).

B. Monte Carlo Simulation

Monte Carlo methods can be used to price European and American options. Pricing is straightforward for European options. Random paths are generated using geometric Brownian motion (1), and each path is evaluated at expiry T (11). The final price is the expected value of v_{opt} over all sampled paths.

$$v_{opt} = \max(K - S_T, 0) \quad (11)$$

Monte Carlo European option pricing is very fast on reconfigurable hardware, with reported speedups of $146\times$ [5], $250\times$ [6], and $340\times$ [7]. For the American option, Monte Carlo methods are complicated by the need to find an optimal exercise boundary. This requires either sorting or regression [1], which consumes significant configurable chip resources [8], [9], [10]. Performing sorting or regression off-chip introduces a communication bottleneck that quickly overwhelms any acceleration gains.

XtremeData's $250\times$ European pricing architecture [6] is highly scalable, since path evaluations are completely independent. American option pricing eliminates this independence, as all paths need to be computed incrementally and simultaneously to estimate the optimal exercise boundary.

Beyond these technical hurdles, Monte Carlo pricing is stochastic, rather than deterministic. The risk of a statistically anomalous result has significant implications when the results are the basis for a trading strategy. For these reasons, the Monte Carlo method is not further considered.

C. Finite Difference Methods

These methods solve the American option pricing equation as a free boundary differential equation. The solution is arrived at directly, through finite differencing. These algorithms are typically highly sequential in nature, and therefore poorly suited for hardware acceleration. Additionally, differencing techniques have the potential to introduce numerical instabilities in certain situations [1], which makes them less desirable for automated trading applications.

III. Hardware Selection

Graphical Processing Units (GPU) are highly-parallel chips originally designed for graphics acceleration. NVidia's CUDA programming language makes GPUs accessible for general-purpose high performance computing. NVidia's Tesla S1070 GPU platform contains up to 120 GPU blocks, with 8 processing cores and 1 double precision floating point unit per block [11]. Each block has 64 Kb of local memory, with access to a much larger shared memory. Speedup is achieved by formulating the problem into a number of threads, which are then dispatched to the cores in parallel.

Field Programmable Gate Arrays (FPGA) are reconfigurable logic chips. FPGA's provide four mechanisms for algorithm acceleration. Logic elements can be pipelined, allowing multiple arithmetic operations to occur during a single cycle. Multiple copies of the pipeline can operate in parallel on a single chip. Local memory is large and highly configurable, reducing stalls from cache misses. Lastly, control logic can be implemented separately from the arithmetic pipeline, allowing for full pipeline utilization.

This project uses FPGAs for algorithm acceleration. Altera's Stratix III EP3SE260 FPGA has sufficient local memory to evaluate up to $N = 64,000$ time step binomial lattices, whereas the S1070 GPU can accommodate at most $N = 2,730$ time steps in local memory. Larger N requires the use of GPU shared memory, which incurs significant synchronization penalties due to pipeline stalls.

IV. Implementation

The American Put Binomial Lattice pricing algorithm is implemented entirely in double-precision floating point representation (IEEE 754-1985). The architecture performs single asset pricing with up to $N = 64,000$ time steps. The algorithm is written exclusively in Verilog, with Altera IP Cores performing floating point operations.

The architecture is optimized and implemented for the Stratix III EP3SE260F1152C2 FPGA featured on Terasic's DE3 development board. The implementation is verified using ModelSim v6.3g simulation, and compiled using Altera's Quartus v9.0 software.

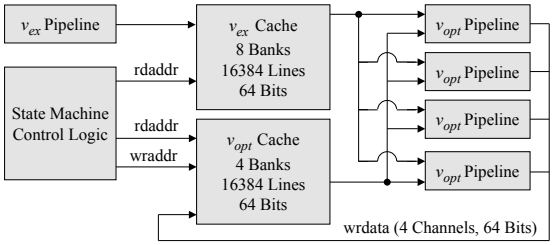

Fig. 1. High Level Architecture. The v_{ex} cache is structured as 8 banks, each with 16384 lines and 64 bits per line, using on-chip memory. Each bank is capable of one read and one write per cycle, from different addresses.

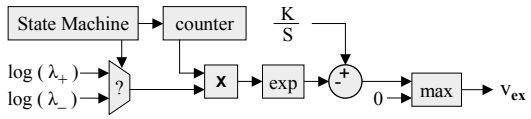

Fig. 2. The v_{ex} pipeline computes one exercise value per cycle, storing results in the v_{ex} cache. Computing the exponent is expensive in hardware, but minimizes communication bandwidth and processor overhead, improving scalability. There is single copy of this pipeline; it runs for $2N + 1$ cycles at the start of each pricing operation.

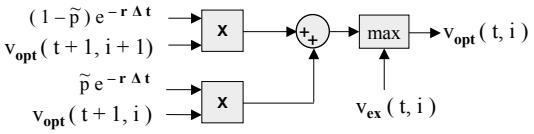

Fig. 3. The v_{opt} pipeline evaluates nodes in the binomial lattice. Each pipeline evaluates one node per cycle, and is replicated 4 times. This pipeline operates for $\frac{N^2}{8}$ cycles to price the option.

A. Architecture

(Fig. 1) shows the top-level architecture. First, v_{ex} values are computed and cached (Fig. 2). Next, the binomial lattice is evaluated (Fig. 3). The lattice evaluation pipeline is based on previous work by Jin [3].

B. Memory

Each node evaluation requires two reads and a write, all in double precision floating point (64 bits). With 4 replications running at 150 MHz, this amounts to 76.8 (Gbits/sec) reading and 38.4 (Gbits/sec) writing. This high-bandwidth memory access requires the use of on-chip memory resources.

Memory is organized into separate banks, with dedicated state machines generating read and write addresses for each memory bank. Node evaluation is performed for a single t, for 4 adjacent values of i. This adjacency simplifies address generation, and allows for the $v_{opt}(t, i + 1)$ necessary to calculate $v_{hold}(t, i)$ to also be used as $v_{opt}(t, i)$ for evaluating v_{hold} for $i = i + 1$. Otherwise, node evaluation would require 3 reads per cycle [3]. The algorithm is evaluated in place; write addresses are the same as read addresses, delayed by 40 cycles to account for pipeline latency.

This precise memory management avoids stalls that would otherwise occur in a computer or GPU implementation, allowing for efficient parallel computation.

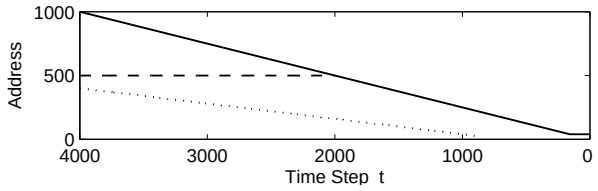

Fig. 4. Optimal Control for $N = 4000$. Since we evaluate across four replications, our address ranges from 0 to 1000. The lattice is evaluated backwards in time from $t = 4000$ to $t = 0$. Runtime is proportional to the area under the line. The solid line shows the niave indexing. The tail on the far right is necessary due to data dependencies. The dashed line shows an optimized maximum address required, when nodes above the strike price are neglected. When the v_{ex} optimization is used, all addresses under the dotted line can be neglected. These two optimizations significantly reduce the area, and thus the computational complexity of analyzing the binomial lattice.

C. Control

The node evaluation pipeline has a 40 cycle delay between reading $v_{opt}(t, i)$, and reading valid data for $v_{opt}(t - 1, i)$. The latency is managed by setting the minimum index i to be greater than the latency. The binomial lattice indexing is constructed such that non-existent nodes can be evaluated without impacting the final result. This stall imposes a small overhead (less than 1000 cycles), and is critical for enabling low-latency parallel pricing of a single option. This is seen as the flat tail in the far right of Figure 4.

Two further control optimizations are possible to accelerate FPGA implementations. These are not implemented for this paper. First, observe that the price of all nodes above the original strike price is always zero. These can be left out of calculations without affecting results, reducing computation time by $N^2/8$ if $K = S_0$ (Fig 4).

Also, observe that the exercise boundary is always non-decreasing with t [12]. When evaluating $v_{hold}(t, i)$ for i below the optimal exercise value, we know that $v_{opt}(t + 1, i)$ and $v_{opt}(t + 1, i + 1)$ can be reduced to $v_{ex}(t + 1, i)$ and $v_{ex}(t + 1, i + 1)$. The v_{opt} only needs to be computed for values of i above the optimal exercise boundary. We can stop evaluating once we reach the optimal exercise boundary. This removes the area below the dotted line (Fig 4). The speedup in standard C code is overwhelmed by the cost of the conditional branching introduced, but this could offer significant gains in an FPGA where conditional branching could be tested in parallel to pipeline operation.

V. Results

A. Performance

Table I shows compilation results for the Stratix III chip. Logic and DSP utilization is low, meaning there is room for more pipeline replications. Initial estimates suggest that we can increase pipeline replication from $4\times$ to $32\times$.

Table II shows a comparison of results. Our CPU benchmark runs on a single core of an AMD Turion 64 bit 1.6 GHz dual-core processor. The benchmark is written in C, and compiled using 'gcc -march=k8 -mfpmath=sse'. The benchmark node evaluations per second decrease as N increases due to cache

TABLE II
PERFORMANCE COMPARISON

	Our Benchmark	**Our FPGA**	**Jin CPU [3]**	**Jin FPGA [3]**	**Jin GPU [3]**	**Pharr CPU [2]**	**Pharr GPU [2]**
Hardware	AMD Turion 64 TL-52	Stratix III EP3SE260F1152C2	Intel Core2 Duo	Virtex 4 FPGA xc4vsx55	Nvidia GPU GeForce 7900	AMD Athlon 64 3200+	Nvidia GPU GeForce 6800
MHz	1600	150	2200	67.3	650	2000	400
Precision	Double	Double	Double	Double	Single	Single	Single
Performance at N = 1000 time steps							
Node Evals/sec	60 M	576 M	4.2 M	202 M	477	60 M	575
Pricings/sec	120	1152	8.4	385	954	120	1150
Speedup	1×	9.6×	0.24×	3.4×	8.1×	1×	9.6×
Performance at N = 10,000 time steps							
Node Evals/sec	16.67 M	599 M	-	-	-	-	-
Pricings/sec	0.3	12	-	-	-	-	-
Speedup	1×	36×	-	-	-	-	-
Performance at N = 50,000 time steps							
Node Evals/sec	10.3 M	600 M	-	-	-	-	-
Pricings/sec	0.0082	0.48	-	-	-	-	-
Speedup	1×	59×	-	-	-	-	-
Performance at N = 64,000 time steps							
Node Evals/sec	8.26 M	600 M	-	-	-	-	-
Pricings/sec	0.0040	0.293	-	-	-	-	-
Speedup	1×	73×	-	-	-	-	-

TABLE I
COMPILATION RESULTS, QUARTUS II V9.0

Hardware	Stratix III EP3SE260F1152C2
Logic	10%
Registers	8% (15,287 / 203,520)
Memory Bits	84% (12,584,289 / 15,040,512)
Multipliers	19% (148 / 768 DSP blocks)
Clock Speed	150 MHz

misses. The FPGA nodes per second increases for larger N, since there are no cache misses and larger N decreases cache calculation overhead.

B. Comparison

There are several previously published results for acceleration of American option binomial lattice pricing [2] [3]. Table II compares our work with these previous implementations. To the author's knowledge, there has been no published work using hardware accelerated with Monte Carlo or finite difference methods to price American options.

Our implementation provides three improvements over previous work. First, it can evaluate lattices of up to $N = 64,000$ time steps, whereas previous work only reported up to $N = 1,024$ time steps [2] [3]. It is unclear how previous methods would scale to large N.

Second, our results are computed entirely in double precision floating point, providing improved precision over previous GPU implementations. We compute exercise values using the logarithm (3) to provide improved numerical stability.

Lastly, previous implementations require large portfolios (up to several thousand options) to be priced simultaneously to achieve speedup. Our method accelerates single asset pricing, providing low-latency performance important for trading applications.

VI. CONCLUSION

This paper presents a scalable architecture for accelerating American option pricing using reconfigurable hardware, for binomial lattices with up to $N = 64,000$ time steps. We achieve a 72× speedup over a CPU benchmark, and a significant improvement over previous reconfigurable hardware implementations.

Future work will increase the pipeline replication from 4× to 32×, and implement control optimizations for additional speedup. Other future research will develop this into an industrial implementation, by including capability for dividend paying stocks and estimation of the Greek sensitivity parameters. One further improvement could be to develop an Ethernet based C-interface to allow the acceleration to be accessed as a network resource. It is envisioned that pricing arguments could be passed as a URL, and the FPGA, acting as a web server, would return the price. This would allow for data-center scaling of the resource.

REFERENCES

[1] P. Brandimarte, *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction*, 2nd ed. Hoboken, NJ: Wiley-Interscience, 2006.

[2] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.

[3] Q. Jin, D. B. Thomas, W. Luk, and B. Cope, "Exploring reconfigurable architectures for binomial-tree pricing models," in *ARC '08: Proceedings of the 4th international workshop on Reconfigurable Computing*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 245–255.

[4] F. Black and M. S. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–54, May-June 1973. [Online]. Available: http://ideas.repec.org/a/ucp/jpolec/v81y1973i3p637-54.html

[5] G. Morris and M. Aubury, "Design space exploration of the european option benchmark using hyperstreams," Aug. 2007, pp. 5–10.

[6] N. A. Woods, "Fpga acceleration of european options pricing," April 2008. [Online]. Available: http://www.xtremedatainc.com

[7] X. Tian and K. Benkrid, "Design and implementation of a high performance financial monte-carlo simulation engine on an fpga supercomputer," Dec. 2008, pp. 81–88.

[8] J. Harkins, T. A. El-Ghazawi, E. El-Araby, and M. Huang, "Performance of sorting algorithms on the src 6 reconfigurable computer," in *FPT*, 2005, pp. 295–296.

[9] A. Beechick, S. Casselman, and L. D. Yarbrough, "Internal sorting and FPGA," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, J. Schewel, P. M. Athanas, V. M. Bove, and J. Watson, Eds., vol. 2914, Oct. 1996, pp. 66–71.

[10] M. van der Horst and R. Mak, "Multi-dimensional parallel rank order filtering," Oct. 2007, pp. 627–632.

[11] B. Oster, "Nvidia advanced cuda: Optimizing to get 20x performance," 2008, nVISION 2008 Presentation.

[12] X. Chen, J. Chadam, L. Jiang, and W. Zheng, "Convexity of the exercise boundary of the american put option on a zero divident asset," vol. 18, no. 1, 2008, pp. 185–197. [Online]. Available: http://dx.doi.org/10.1111/j.1467-9965.2007.00328.x