Option Pricing with Multi-Dimensional Quadrature Architectures

Anson H.T. Tse, David B. Thomas, Wayne Luk

Department of Computing, Imperial College London, England
{htt08,dt10,wl}@doc.ic.ac.uk

***Abstract*—Quadrature based methods for numerical integration provide a means of quickly and accurately pricing financial products such as options. These methods can be applied to multi-dimensional products, such as options on multiple underlying assets, but suffer from an exponential increase in computational complexity as the dimension increases. This paper examines the theoretical complexity of quadrature methods for pricing multi-dimensional options, and then relates this to practical performance in contemporary hardware. An automated system for generating hardware architectures for quadrature is used to explore the performance of increasing dimensionality in FPGA implementations, and then compared them to GPU and CPU solutions. We find that a single-precision FPGA can provide 25 times speedup over software in three dimensions, and offers slightly improved performance over a GPU using comparable technology. The latest GPUs are 2.7 times faster than the older technology Virtex-4 FPGA, but the FPGA still provides over 9 times the energy efficiency.**

I. Introduction

Financial services companies continually invent new ways to repackage and modify financial products in order to satisfy the needs of different investors. Many financial derivatives now involve multiple underlying assets instead of just one. As the computation complexity increases exponentially with the number of underlying assets (i.e. the number of dimensions), finding ways to accelerate the option pricing computation becomes a significant challenge.

Quadrature methods are a powerful technique for options pricing, especially in situations where a closed-form solution does not exist. Quadrature methods can be used to price many types of options and are very efficient for pricing path-dependent options.

A single option with one underlying asset priced using quadrature methods can typically be performed in milliseconds on desktop computers. However, the computation time of quadrature methods increases exponentially with the number of dimensions, which is also known as "the curse of dimensionality". This paper shows how to accelerate quadrature computation with multiple dimensions using Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs). The major contributions of this paper are:

- An abstract model of computational complexity for option pricing using quadrature methods, parameterised by dimension and number of time-steps (Section III).
- An automated system for generating hardware architectures and implementation parameterised for a given set of dimensions (Section IV).
- Results showing the performance of designs from our approach, including comparisons with GPU and CPU implementations (Section V).

II. Related work

Numerical techniques have been developed to value complex financial products where a closed-form solution does not exist. These techniques include lattice (binomial and trinomial trees), finite-difference, Monte Carlo and quadrature methods. Among the various methods for hardware acceleration of financial simulation, Monte Carlo methods have been the focus in the past few years.

Some recent studies have shifted the focus to pipelined tree based methods [1], finite-difference methods [2] and quadrature methods [3]. All of these studies have provided solutions for the pricing of certain options (such as American options) which cannot be handled easily by Monte Carlo methods. However, none of the above addresses the issue of pricing options with multiple underlying assets. Reference [3] demonstrates hardware accelerated quadrature option pricing methods. This paper significantly extends it to cover the multiple underlying assets problem and provides the complexity model with the number of dimensions.

III. Model Analysis

A. Option Pricing and Quadrature Methods

To understand option pricing with multiple underlying assets using quadrature methods, we first consider the Black Scholes partial differential equation [4] for an option with all underlying assets following geometric Brownian motion:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}S_iS_j\frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d}(r-D_i)S_i\frac{\partial V}{\partial S_i} - rV = 0 \quad (1)$$

where r is the risk-free interest rate, d is the number of underlying assets, S_i are the underlying asset values, σ_i and D_i are the corresponding volatilities and dividend yields, and ρ_{ij} is the correlation coefficient between underlying asset values S_i and S_j. Note that $|\rho_{ij}| \leq 1$, $\rho_{ii} = 1$ and $\rho_{ij} = \rho_{ji}$. We make the transformations $x_i = \log(S_i)$ to be the chosen nodes

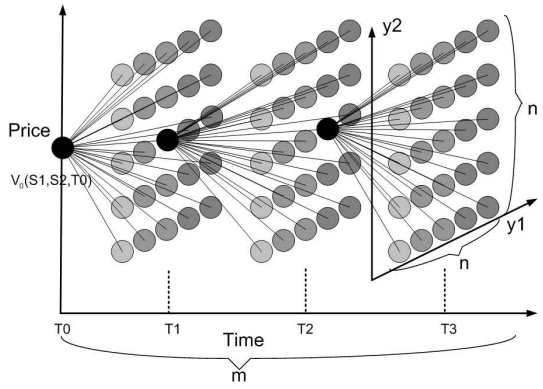

Fig. 1. The iteration process of a 2D barrier option.

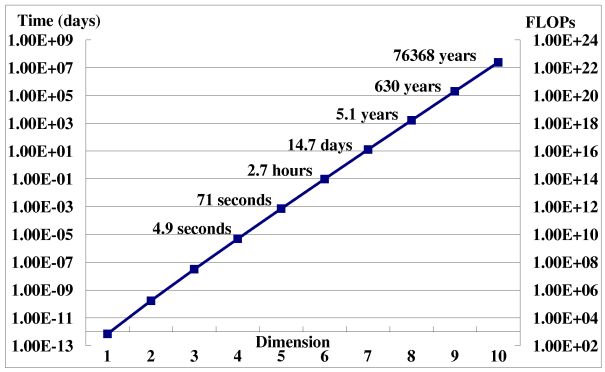

Fig. 2. The time required for the pricing of European options. The y-axis on the right indicates the total FLOPs required. (n = 100)

at t and $y_i = \log(S_i)$ to be the chosen nodes at $t + \Delta t$. Let R be the matrix such that $R(i,j) = \rho_{ij}$. The solution is:

$$V(x_1,\ldots,x_d,t) = C\int_{-\infty}^{+\infty}\ldots\int_{-\infty}^{+\infty} V(y_1,\ldots,y_d,t+\Delta t) E(x_1,\ldots,x_d,y_i,\ldots,y_d)dy_1\ldots dy_d \quad (2)$$

$$C = e^{-r\Delta t}(2\pi\Delta t)^{-n/2}(|R|)^{-1/2}(\sigma_1\sigma_2\ldots\sigma_d)^{-1}, \quad (3)$$

$$E(x_1,\ldots,x_d,y_i,\ldots,y_d) = exp(-\frac{1}{2}\alpha^T R^{-1}\alpha), \quad (4)$$

$$\alpha_i = \frac{x_i - y_i + (r - D_i - \frac{\sigma_i^2}{2})\Delta t}{\sigma_i(\Delta t)^{1/2}} \quad (5)$$

Eq. (2) is the fundamental equation for the option pricing and it contains an integral which cannot be evaluated analytically. $V(y_1,\ldots,y_d,t+\Delta t)$ can be regarded as the option value in the future time $t + \Delta t$. The option value at current time $V(x_1,\ldots,x_d,t)$ is therefore evaluated by integrating $V(y_1,\ldots,y_d,t+\Delta t)E(x_1,\ldots,x_d,y_i,\ldots,y_d)$ for all the possible values for assets $y_1, y_2, \ldots, y_d$. The number of dimensions for this integration is equal to the total number of assets. Quadrature methods are required to evaluate this multi-dimensional integral. There are many different methods for numerical integral evaluation. The most common methods include the trapezoidal rule and Simpson's rule [5].

The number of integrations depends on the type of option. For European options, the required number of integrations is 1. For other options such as American options and Barrier options, multiple steps of integrations are required. The time is divided into many small intervals. For each time interval, all the option values V in different combination of $x_1,\ldots,x_d$ are calculated by integrating the values of V and E in the next time interval and then used as the $V(y_1,\ldots,y_d,t)$ for the calculation in the previous time interval. The current option price is therefore calculated iteratively backward from the end of the time. Fig. 1 shows the graphical representation of the iteration process. We define n as the number of possible values (grid points) for y_1 and assume the number of grid points for all y_i is the same, and we define m as the number of time intervals. As a result, the number of integrations required for American options and Barrier options is mn^d.

B. Complexity Analysis

From Eq. (4), the evaluation of E involves the matrix multiplication of column vector α. The calculation of all α_i from Eq. (5) can be optimized as:

$$\alpha_i = (x_i - y_i + C1_i)/C2_i \quad (6)$$

$C1_i$ and $C2_i$ can be precomputed in a pre-processing stage as the values of all $C1_i$ and $C2_i$ are kept constant throughout the quadrature evaluation.

Table I shows the summary of computation complexity for some example options.

The computation time can be estimated by assuming that a 10 GFLOPs CPU is used (the peak performance of a Pentium 4 3.2GHz CPU is around 6.4 GFLOPs) and it takes the same time for all different floating point operators. Fig. 2 shows that the computation time required is drastically increased with the number of dimensions. The computation time required for a European option with 7 underlying assets is 14.7 days using a desktop computer with peak performance.

IV. Implementation

A. System architecture

The option evaluation system architecture is not designed for pricing a specific type of option. The implementation to support various options with any number of dimensions is considered. The system architecture of the generic option valuation system using quadrature method is shown in Fig. 3. The architecture consists of the following components: (a) a pre-processing block, (b) QUAD evaluation core, (c) a post-processing block, and (d) a main control unit.

Option parameter inputs for the system include $T, S0_1\ldots S0_d, r, D_i\ldots D_d, \sigma_1\ldots\sigma_d, R$, option-type, option-specific-parameters and dimension. The option-type and option-specific-parameters provide the flexibility to support the pricing of other options. For example, we could specify the number of periods (m) and the knock-out/ knock-in prices

TABLE I
THE COMPUTATION COMPLEXITY FOR SOME EXAMPLE OPTIONS.

Option type:	Number of integration	Number of evaluation of E	Number of floating-point operation for evaluating E
European options	1	n^d	$n^d(2d^2+4d+1)$
Barrier options	mn^d	mn^{2d}	$mn^{2d}(2d^2+4d+1)$
American options	mn^d	mn^{2d}	$mn^{2d}(2d^2+4d+1)$

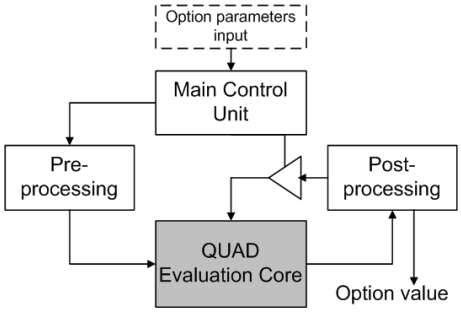

Fig. 3. The system architecture of the generic option valuation system. The arrows indicate the dataflow of the system.

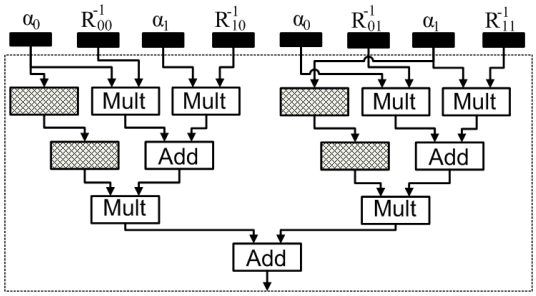

Fig. 4. Fully pipelined $\alpha^T R^{-1}\alpha$ calculation part inside a 2D QUAD evaluation core.

(b) for Barrier options. We could also specify the grid density factor or grid size factor to select the corresponding accuracy.

The most challenging part of the multi-dimensional design is how it supports an arbitrary number of dimensions. For multiple assets design, the hardware evaluation cores are completely different for different dimensions as the underlying logic and the number of pipeline stages are different. As a result, all the possible hardware designs for all possible dimensions are implemented. The resultant bitstream for each dimension is prepared and stored in the system. The hardware evaluation core has to be reconfigured with the bitstream for the corresponding dimension.

B. Pipelined Implementation

The hardware implementation of the QUAD evaluation core can be divided into three major parts. The first part is the evaluation of the vector α from Eq. (6). The second part is the matrix multiplication of $\alpha^T R^{-1}\alpha$. The last part is the rest of the integration. *HyperStreams* and the Handel-C programming language are used for the FPGA implementation of the QUAD evaluation core. *HyperStreams* is a high-level abstraction language and library [6]. It produces a fully-pipelined hardware implementation with automatic optimization of operator latency at compile time.

Fig. 4 shows the diagram of $\alpha^T R^{-1}\alpha$ calculation inside a 2D QUAD evaluation core. The grey boxes denote the pipeline balancing registers that are allocated automatically by *HyperStreams*. As the evaluation is pipelined, we can have a partial integral value for every clock cycle. For an FPGA running at 100MHz, the QUAD evaluation core can produce 100M partial integral values per second.

C. Design Automation

Implementing the evaluation core for each dimension manually would be tedious. As a result, an automatic evaluation core generator is designed. The generator accepts 2 input parameters: number of dimensions and precision (single or double). An operator tree is generated and stored in a temporary file. Finally, the operator tree file is parsed and the corresponding *HyperStreams* and Handel-C codes are generated. The Handel-C code could be compiled for simulation or bit-file generation.

Table II shows the FPGA device utilization figures for xc4vlx160 devices of the QUAD evaluation core in different dimensions and precisions. The result indicates that the FPGA device is fully occupied for 2 dimensions under double-precision, and is fully occupied for 6 dimensions under single-precision.

V. EVALUATION AND COMPARISON

In this section, the performance of different implementations of QUAD evaluation core is studied. We choose the pricing of 1,000 European options as the performance benchmark. The performance analysis for pricing 3-underlying assets European options is studied. Simpson's rule is preferable to the trapezoidal rule in our system as the error terms of Simpson's rule decrease at a rate of $(\delta y)^4$ which produces more accurate results with the same hardware complexity.

The performance of both 32-bit single precision and 64-bit double precision is investigated. The acceleration of FPGA implementation and GPU implementations are compared with software implementation on a PC as a reference. The reference PC is a 3.6GHz Pentium 4 processor with 1GB RAM and Ubuntu 8.04.1 operating system. The software implementation is written in C language and compiled with maximum speed optimization options.

The targeted FPGA is Xilinx Virtex-4 xc4vlx160. The designs are compiled using DK5.1 and Xilinx ISE 10.1. The targeted GPU is nVidia Geforce 8600GT with 256MB of on board RAM and nVidia Tesla C1060 with 4GB of on board

	FPGA - Virtex-4 xc4vlx160					
Precision	single					double
Dimension	2	3	4	5	6	2
Slices	27,970 (41%)	38,006 (56%)	51,862 (76%)	67,582 (99%)	67,582 (99%)	100,963 (149%)
LUTs	32,053 (23%)	43,580 (32%)	60,058 (44%)	70,909 (52%)	94,545 (69%)	84,063 (62%)
FFs	22,418 (16%)	30,005 (22%)	41,466 (30%)	54,569 (40%)	72,515 (53%)	67,361 (49%)
DSPs	51 (53%)	75 (78%)	96 (100%)	96 (100%)	96 (100%)	96 (100%)
Clock Rate	91.2MHz	89.7MHz	91.5MHz	88.0MHz	(*)	(*)

TABLE II
THE LOGIC UTILIZATION OF QUAD EVALUATION CORE IN DIFFERENT DIMENSIONS. ASTERISK (*) INDICATES THAT THE PLACE AND ROUTE PROCEDURE CANNOT BE COMPLETED.

	FPGA	GPU			PC
	Virtex-4 xcvlx160	Geforce 8600GT	Tesla C1060		Pentium 4
Arithmetic	single	single	single	double	double
Clock Rate	89.7MHz	1.35GHz	1.3GHz	1.3GHz	3.6GHz
Processing Speed (M values/sec)	89.7	81.7	589.0	241.7	3.6
Acceleration	24.9x	22.7x	163.7x	67.2x	1x
Max Power (Watt)	5.2	43	200	200	115
Processing/Max Energy (M values/J)	17.25	1.90	2.94	1.21	0.03

TABLE III
THE PERFORMANCE COMPARISON OF DIFFERENT IMPLEMENTATION OF 3D QUAD EVALUATION CORE.

RAM. The summary of the performance comparison of 3-dimensional QUAD evaluation core is shown in Table III.

From the result, it can be seen that the FPGA implementation on xc4vlx160 achieved 24.9 times acceleration. For the performance of GPUs, a speedup of 22.7 times is achieved by Geforce 8600GT. A speedup of 163.7 times is achieved by Tesla C1060 under single-precision. In double-precision, Tesla C1060 has shown a 67.2 times speedup over the reference PC. There is no support of double-precision for Geforce 8600GT.

It can be seen that xc4vlx160 outperforms Geforece 8600GT by 9.7%. It is not surprising that Tesla C1060 GPU outperforms xc4vlx160 by 6.6 times, as Tesla C1060 is the latest GPU based on 65nm technology while Virtex-4 xc4vlx160 is an older model of FPGA based on 90nm technology. Future work will compare Tesla C1060 with the latest FPGAs such as Virtex-5 from Xilinx or Stratix IV from Altera.

It is interesting that the xc4vlx160 FPGA demonstrates better performance than the GPUs when energy consumption is taken into account. Xc4vlx160 can compute around 9 times more options than Geforce 8600GT and Tesla C1060 per unit of energy usage. Further performance improvement for FPGAs can be achieved using more up-to-date devices such as Virtex-5. As Virtex-5 has 4 times more slices than Virtex-4 and with higher clock frequency, potentially 4 times more speedup can be achieved without further optimization.

VI. CONCLUSION

This paper explores hardware accelerated option pricing models based on quadrature methods in multiple dimensions. An abstract model of computational complexity for option pricing using quadrature methods, parameterised by dimension and number of time-steps is developed. We implement the design for different dimensions in both FPGA and GPUs. An automated system for generating hardware architectures and implementation parameterised for a given set of dimensions is developed.

The performance of different implementations in higher dimensions is compared. Our implementation on FPGA can generally run 24.9 times faster than a Pentium 4 3.6GHz processor, 10% times faster than a GPU in comparable technology and 2.7 times slower than the latest GPU for 3-dimensional option pricing. From the energy consumption perspective, the FPGA is up to 9 times more efficient than GPUs and 575 times more efficient than the CPU.

ACKNOWLEDGMENT

The support of the Croucher Foundation, UK EPSRC, AlphaData, Celoxica and Xilinx is gratefully acknowledged.

REFERENCES

[1] Q. Jin, D. B. Thomas, W. Luk, and B. Cope, "Exploring reconfigurable architectures for binomial-tree pricing models," in *Proceedings of the 4th international workshop on Applied Reconfigurable Computing*. LNCS 4943. Springer-Verlag, 2008, pp. 245–255.

[2] Q. Jin, D. B. Thomas, and W. Luk, "Exploring reconfigurable architectures for explicit finite differece option pricing models," in *Proc. Int. Conf. on Field-Programmable Technology*. IEEE, 2009.

[3] A. H. T. Tse, D. B. Thomas, and W. Luk, "Accelerating quadrature methods for option valuation," in *Proc. IEEE Symp. on FPGAs for Custom Computing Machines*, 2009.

[4] A. D. Andricopoulos, M. Widdicks, D. P. Newton, and P. W. Duck, "Extending quadrature methods to value multi-asset and complex path dependent options," *Journal of Financial Economics*, vol. 83, no. 2, pp. 471 – 499, 2007.

[5] E. Sueli and D. F. Mayers, *An Introduction to Numerical Analysis*. Cambridge University Press, 2006.

[6] G. Morris and M. Aubury, "Design space exploration of the European option benchmark using hyperstreams," in *Proc. Int. Conf. on Field Programmable Logic and Applications*. IEEE, 2007.