University of Kaiserslautern

Department of Electrical Engineering and Information Technology

Microelectronic Systems Design Research Group

Xilinx Open Hardware 2015 Design Contest

Pushing the Limits in Financial Acceleration:

Multidimensional American Option Pricing on Xilinx Zynq

Participant:	MSc. Javier Alejandro Varela
Supervisor:	Prof. Dr.-Ing. N. Wehn
University:	University of Kaiserslautern
Team Number:	XIL-83437
Youtube:	https://youtu.be/EVGVIQayf1o

Contents

1 Introduction

This project is an upgraded version of the MSc. Thesis written by the same author [1], and the follow-up paper presented at the DATE Conference 2015 [2]. It is set to price a computationally intensive financial product called *multidimensional American Option*.
In general terms, an *option* is a contract that gives the right, but not the obligation, to buy or sell the underlying asset at a fixed price and date. In a multidimensional option, there are multiple underlying assets simultaneously. This contract has an associated price, a premimum paid at the moment of purchase. What makes options interesting is that they provide a limited risk (the premium paid at purchase), with unlimited potential gains. And it is precisely the computation of this option price what concerns financial institutions.
The so-called *American style* allows the holder to exercise the option at any time from purchase until the expiry date. It contrasts wtih the European option style, in which the option can only be exercised at a fixed date. The freedom offered by the American style makes its pricing much more challenging, since now an optimal exercise strategy is involved.
The widely used algorithm that tackles this problem is the Longstaff-Schwartz (LS) [3], which employs least-squares linear regression on Monte Carlo (MC) simulated paths. The basic steps of the LS algorithm are described as follows:

1. Generate N independent paths per underlying (stock) at all possible exercise dates, using a chosen random number generator (RNG) and a chosen mathematical model. For multi-dimensional options, the random numbers (RNs) are correlated.

2. Initialize the cash-flow payoff at maturity.

3. Moving backwards one step in time, 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 payoff (if exercised) is compared to the discounted future expected return (if the option is held at this time step), approximated by regression. Least-squares linear regression for user-defined basis functions is applied to obtain the regression coefficients.
   - Cash-Flow update: For every path at the current time step compare the current payoff against the discounted expected return. If the current payoff is larger, updated the cash-flow accordingly, otherwise discount the current cash-flow one step.

   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 model used to describe the evolution of the stocks in the market is the traditional Black-Scholes (BS) [4], particularly suitable for multidemsional problems. For MC simulation, the BS model is discretized into m steps with equal step sizes $\Delta t = \frac{T}{m}$ following the Eurler scheme [5]:

$$\hat{S}_{t_{i+1}} = \hat{S}_{t_i} \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\,\Delta W_i\right), \tag{1.1}$$

with ΔW_i being independent standard normal random variables. The latter random variables are obtained via the Mersenne-Twister [6], and the Box-Muller transformation

[7]. For multidimensional options the normal distributed numbers need to be correlated. In the case of 2 dimensions, the correlation is accomplished following:

$$w \quad = \quad \rho\, z_1 + \sqrt{(1-\rho^2)}\, z_2 \tag{1.2}$$

for correlation coefficient ρ and normal variables z_1, z_2 in the range [0,1], generating the new pair z_1, w.
The improved version of this algorithm is called *Reverse Longstaff-Schwartz* [1, 2], which is particularly suitable for field programmable gate arrays (FPGAs). Once the paths have been generated forward in the traditional way, it recomputes the paths backwards alongside the LS algorithm. To do so the BS model is reversed by changing the sign of the exponent in Eq. (1.1), the Mersenne twister (MT) algorithm is reversed following [8], and the Box-Muller transformation is carefully written in the reverse operation to generate exactly the same sequence of normal random numbers backwards. This reversed operation not only avoids the use of external memory (DRAM), reducing energy consumption, but it also ensures that no blocks become idle during parts of the complete runtime, as explained in [1] and [2].
The work presented in this Xilinx competition upgrades the original architecture by merging the forward and reverse paths generation blocks into flexible (*flex*) blocks, avoiding the use of the partial reconfiguration step presented in [1] and [2]. This version is being currently used as a testbench for undergoing research on further optimizations.

1.1 Motivation

The *OTC* (Over-the-Counter) derivatives markets trade nowadays an average anual volume of approximately U$D 700 trillion [9]. This large trading volume and stricter regulations in the market force financial institutions to constantly search for more efficient ways to price their products. Pricing American options, and furthermore the multidimensional version, is a computationally intensive task. Since options are usually a contituent part of trading portfolios to hedge risk, it becomes essential to research on different acceleration alternatives. This work is a part of a much larger undergoing project.

2 Design

A detail of the setup used in this project is presented in Table 2.1. The version of Vivado and Vivado HLS available is 2014.3.

Table 2.1: Setup Table.

Detail	Description
Option style	American
Option type	Call/Put
Option characteristics	maximum
Dimensions	2
Basis Functions type	monomial
Basis Functions detail	$1, max(S_1, S_2), max(S_1, S_2)^2$
Paths per dimension	10K
Steps	365
Data type	single-precision floating point
Total data	27.85 MB
Platform	ZC702 Evaluation Kit
Zynq	XC7Z020-1CLG484C
Operating System	Linux (Linaro)
FPGA clock frequency	100.00 MHz

The overall architecture is presented in Fig. 2.1. Linux Linaro Server runs on the Zynq PS side, and a software code (.cpp) is compiled and executed to take control of the entire operation. The FPGA architecture fully pipelined and has been split into reusable IP blocks connected via hls::streams.
This version of the architecture differs from the design presented in [1, 2] in two ways:

1. No intermediate partial reconfiguration is needed, by making the Paths Generation blocks flexible enough to be able to operate both forward and backward. (This strategy is necessary as a testbench for an ongoing research and paper).

2. The Regression Coefficients block is fully implemented in hardware to reduce runtime. (Placing this block in software is only necessary under certain flexibility requirements that are no longer necessary in the current research).

The operation starts by compiling and running the driver (.cpp file), by sourcing in linux a ./run.sh file. Using the corresponding bitstream the FPGA is then programmed. All IP blocks are setup by the driver and triggered to start operation. The Paths Generation block first generates all paths forward but do not transfer any result to the rest of the architecture, which is in stand-by. Once maturity is reached, the paths are recomputed backwards and transferred to the LS blocks. When the initial timestep is reached, the CashFlow values are transferred to an accumulator. The accumulated value is read by the PS and averaged to obtain the option price. A golden (reference) price is also provided in software.
The user has full control over all variables related to the algorithm, where only the number of paths (pathsMC) are limited (in this project) to a maximum of 32K. The reason is that a larger number of paths implies an increase in the BRAMs used, and 10K paths yield acceptable results in practice.

One important detail about the LS algorithm is the fact that the Regression Coefficients step is a natural bottleneck. It requires all paths at the current step to be processed and accumulated. And the update blocks can only begin once the coefficients have been calculated. In other words, when the Regression Coefficients block is working, the complete pipelined architecture is forced to stall once at each time step. The use of hls::stream greatly simplifies the design allowing the pipelined architecture to regulate itself automatically during such condition.

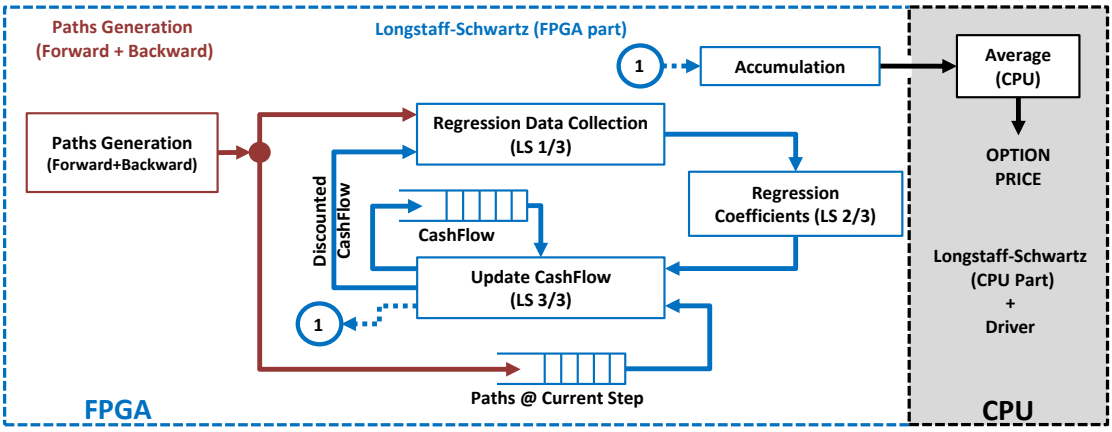

Figure 2.1: Simplified design architecture including Paths Generation and Longstaff-Schwartz, targetting the Zynq platform. Multiple parallel instances of all blocks are possible, except for the Regression Coefficients block that is unique.

2.1 Hardware

The Paths Generation block is described in more detail in Fig. 2.2. All blocks are custom IP, and they are fully pipelined with an initiation interval of 1 (II=1).

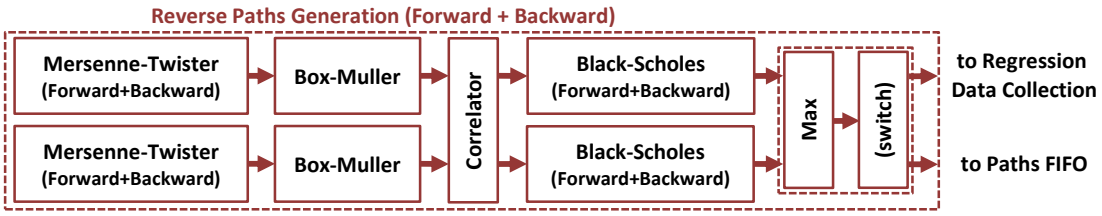

Figure 2.2: Paths Generation: all IP blocks were developed by the team.

- A MT block provides a sequence of uniform random numbers, first forward and then backward. Each block requires a vector of 624 initial states that are generated and delivered by the PS.
- The Box-Muller block transform from uniform into normal distribution. It is carefully coded so that the forward and backward operation yield the same sequence.
- Normal numbers need to be correlated, with coeffient ρ provided by the PS.
- Each BS module simulates the behaviour of each stock in the market. All parameters are delivered by the PS. It only outputs the paths from maturity until the initial day (backward).

- For the given option setup, the maximum value of boths stocks is taken at the current path and step.
- (switch): the paths at maturity are the first values to be delivered by the BS modules. This values are only sent to the Paths FIFO, as it will be explained later. The paths at all other steps are delivered to both outputs. (NOTE: In the current design, max and switch are coded in a single IP block).

The LS section is presented in more detail in Fig. 2.3.

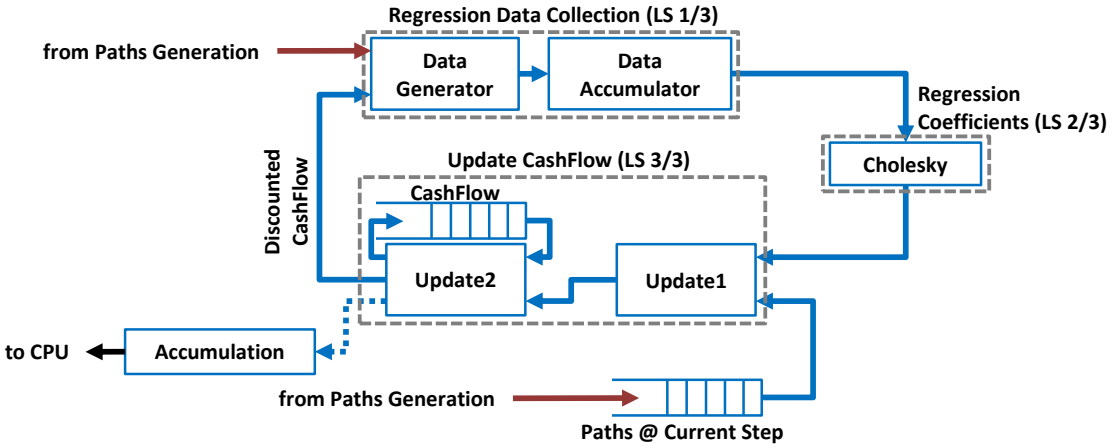

Figure 2.3: Longstaff-Schwartz: the FIFO *Paths @ Current Step* is implemented using *AXI4-Stream Data FIFO (1.1)* the all other IP blocks were developed by the team.

- The Data Generator prepares, in a fully pipelined operation (II=1), the values that are to be considered for regression at the current step. It follows the same accumulation strategy described in [10].
- The Data Accumulator accumulates in a fully pipelined operation (II=1) the generated values for the current step. It uses a lightweight version with only one adder and one BRAM to host a vector of 16 partial sums. Once all paths (input values) have been accumulated, the total sum is obtained from the partial sums, using only one adder, in 113 clock cycles (II=7). This small time overhead is negligible when compared to the minimum number of paths at around 10K. Since many accumulators are required (8 in this configuration), the main goal is to achieve a lightweight version that can accumulate the partial sums at II=1.
- Regression Coefficients are computed via Cholesky decomposition and using forward and backward substitution. It currently computes 3x3 matrices. It has been designed for minumum resources utilization, and achieves and initiation latency of 197 clock cycles. (NOTE: see [10] as a reference, although they implemented a direct matrix inverse method to obtain the coefficients. Direct matrix inverse is not advisable due to potential instability).
- CashFlow Update 1 computes the discounted continuation value (using the regression coefficients and the paths at the current step) and the current payoff. At maturity it only reads the values from the paths at the current step, since regression is not possible yet. At all other steps it reads the regression coefficients (once per step) and all corresponding paths. This block is fully pipelined with II=1 for the paths loop.

- CashFlow Update 2 compares both values delivered by the previous block and updates the CashFlow accordingly, either with the current payoff or with the discounted CashFlow value. This block is fully pipelined with II=1. The CashFlow FIFO is always updated. The discounted CashFlow output is available at all intermediate time steps. All new CashFlow values are delivered to the CashFlow accumulator only at step 0 (initial date). NOTE: The IP block incorporates the CashFlow FIFO as a vector inside the block, which makes the design, test and debug easier than using an external FIFO.
- CashFlow Accumulator is responsible for accumulating the CashFlow at the initial date, and to deliver the total sum to the PS when requested. Its design is almost identical to the one used in the Data Accumulator.
- Paths @ Current Step FIFO: it is the only off-the-shelf IP block used (AXI4-Stream Data FIFO (1.1)).

2.2 Software

The driver is written in C language and run under Linux Linaro Server. There is a run.sh script that needs to be sourced for setup and to compile and run the source code. Since all IP blocks are memory-mapped, the code opens a page for each device (IP) and communicates to them as if writting to memory.
First the FPGA is programmed using the corresponding bitstream. The FPGA architecture can be tested for debugging purposes (optional).
Then all IP blocks are setup. For the MT modules the program has to compute and deliver the initial states (seeds), which is done by invoking the corresponding functions in *mt19937ar.h*. All modules are then triggered and the driver waits until the CashFlow Accumulator has the final sum. It then reads the value, and computes the average which yields the option value.
A golden (reference) value is obtained from the benchmark found in *ls_golden.h*. Rounding effects in floating-point operations have a significant impact on the results obtained. Therefore, there are two versions: one is a straightforward implementation, and a second one uses the same strategy used in the Data Accumulation block. This is to provide a much closer result to the one obtained in the FPGA implementation. It is important to take into account that the golden version is not an exact copy of the IP blocks in FPGA, which means that small discrepances are to be expected in the results.
The provided files *mt19937ar.h* and *mt19937ar.cpp* were obtained from a third party [11].
The base setup of the Linux kernel as well as the functions to open a page in linux, were previously prepared at the Microelectonics Systems Design Research Group, University of Kaiserslautern. All other source code delivered in this project is custom-made.

2.3 Design Reuse

This work presents a modular architecture, and is the first fully functional design for American Options (in FPGA) in our research group. Each IP module is self contained and can be reused in later designs. They are all now part of our (growing) IP library.
Regarding off-the-shelf IP blocks, the FIFO was implemented by means of *AXI4-Stream Data FIFO (1.1)*. Although the LogiCORE IP Floating-Point Operator v7.0 is also available and configurable as accumulator, the custom-made lightweight version of the accumulators in this project (with an initiation interval II=1) is a better fit.

All blocks in the Paths Generation step can also be used also in other applications rather than finance. This includes the blocks that work forward and backward, and the ability to deliver exact sequences of random numbers in both directions.
Although all Longstaff-Schwartz blocks are specifically design for such algorithm, several parts are reusable. First, the lightweight accumulators with an initiation interval II=1. Second, how these accumulators are wrapped into a single IP block with 8 accumulators. Third, the Cholesky decomposition (with forward and backward substitution) for 3x3 matrices can be reused and extended.
The IP blocks presented in this project as well as the source code are not available online at the moment. They are currently part of an ongoing research.

3 Results

The target of this project is a fully working implementation of a single instance of the full Reverse Longstaff-Schwartz algorithm in its new combined version (forward+backward). This setup is used as a reference point for other architectures and improvements that are being currently evaluated.
Although the FPGA clock frequency has been set at 100MHz, all modules shown in Table 3.1 have been synthesized and exported with a clock period of 7.5ns (133.33MHz) in Vivado HLS. All IP blocks are fully pipelined with an initiation interval II=1 (with the only exception of Cholesky and the final partial-sums addition inside the accumulators, as explained in the previous sections).

Table 3.1: FPGA P&R resources breakdown per module.

Block	Module	LUT	FF	DSP	BRAM
Paths Generation	Mersenne-Twister Flex	821	814	0	6
	Box-Muller Flex	4697	5447	13	5
	Correlator	693	1135	8	0
	Black-Scholes Flex	2311	2484	23	64
	Max (split)	420	1039	4	0
Longstaff-Schwartz	axis_data_fifo	141	193	0	29
	LS_DataGenerator	1186	1832	17	0
	LS_DataAccumulator	3889	5805	16	8
	Cholesky	2578	2917	5	6
	LS_Update1	1396	2545	15	0
	LS_Update2	791	1149	6	64
	CashFlowAccumulator	433	648	2	1

Table 3.2 presents a detailed analysis of the runtime per module. Even with only one instance of each module in the proposed architecture and a moderate clock frequency of 100MHz, the total runtime including the FPGA configuration is 163.373ms, which implies a 73x improvement over the straightforward implementation in software (Golden) running on Zynq, and 65% faster than a fully utilized Intel i5-2450M (2.50 GHz) core with, 6GB of RAM [2].

Table 3.2: Runtime comparison.

Device	Operation	runtime [ms]	% of total
FPGA	Config. (bitstream)	88.861	54.391%
	Setup	0.002	0.001%
	Execution	74.511	45.608%
	Total	163.373	-
CPU	Zynq - Golden	11938.740	-
	Zynq - Golden (Accum)	12170.005	-
	Intel Core I5 [2]	270.000	-

Measuring the total (static + dynamic) average power consumption on the ZC702 platform using Texas Instruments Fusion Digital Power Designer, and the runtime from

Table 3.3: Energy consumption comparison.

Device	Operation	runtime [ms]	energy [mJ]
FPGA	Config. (bitstream)	88.861	98.560[1]
	Setup	0.002	0.001
	Execution (Forward)	36,400	30.212
	Execution (Backward)	38,111	42.684
	Total	163.373	171.457
CPU	Zynq - Golden	11938.740	5013.96
	Zynq - Golden (Accum)	12170.005	5111.402
	Intel Core I5 [2]	270.000	12700.000

[1] Approximated based on the power consumption between CPU, FPGA and DRAM.

Table 3.2, it is possible to obtain the energy consumption per block in Table 3.3. In order to obtain stable readings, in all power measurements the runtime was set towards 10 seconds, by modifying certain parameters like the number of timesteps.
In terms of energy consumption, the hybrid CPU/FPGA architecture is 35x more energy efficient than the Golden implementation on the Zynq PS, and 74x more energy efficient than the fully utilized Intel Core I5.

Regarding proper numerical operation of the architecture, consider the parameters setup for the two-dimensional American Max option shown in Table 3.4.

Table 3.4: Test setup for a two-dimensional American max option.

Parameter	Value	Description
T	1.0	maturity (1year = 1.0)
S0_1	100.00	spot price (current price of stock 1)
S0_2	100.00	spot price (current price of stock 2)
K	100.00	option's strike price
sigma1	0.30	volatility of stock price
sigma2	0.30	volatility of stock price
r	0.06	risk-free interest rate (1-year interest rate)
q	0.00	dividend yield
rho	0.10	correlation between boths stocks
callPut	-	0=call (buy), any other value=put (sell)
STEPSMC	365	
PATHSMC	10e3	
seedValue1	0	Mersenne-Twister seed for stock 1
seedValue2	12345	Mersenne-Twister seed for stock 2

The results shown in Table 3.5 show the proper operation of the hybrid CPU/FPGA architecture. (NOTE: as mentioned in previous sections, please bear in mind that due to rounding errors in floating-point operations, the numerical results will differ between implementations, since they are not idential copies in terms of coding sequence.)

Table 3.5: Numerical results.

Version	**Put**	**Call**
FPGA	4.3104	24.2904
CPU Golden	3.9115	23.9779
CPU Golden (Accum)	4.3051	24.2414

4 Conclusion

The proposed architecture is a hybrid CPU/FPGA design, where the CPU governs the operation of the complete flow. Even though the major part of the complete algorithm runs on FPGA, the initialization of the seeds and generation of interal states for the Mersenne-Twiter blocks is run on CPU. Also the final average of the accumulated Cash-Flow is executed in software.
It has also been presented that there is a considerable advantage in terms of runtime and energy consumption compared to other solutions in CPU. (A comparison to equivalent hybrid CPU/GPU devices like the NVIDIA Tegra K1 is not included in this report).
Furthermore, pricing a two-dimensional American maximum option was the setup used in this work, which provides the basis for an expansion to higher-dimensional options with minimum effort.
Regarding the major challenges to this project, two are particuarly important and they are related to the fully pipelined operation with an initiation interval II=1:

- This fully pipelined architecture stalls regularly once per time step for a certain number of clock cycles (while the Regression Coefficients block is working). This is a natural behaviour of the Longstaff-Schwartz algorithm. However, this architecture also includes a feedback loop, which make the operation more challenging. In the original design it was found that the LS part of the design was stalling at a particular point in time. The cause was found to be that the *flush* option in merged nested *for* loops seemed not to work. Why this happens is still under investigation, but the design was very easily modified to avoid the merging operation (which is in fact not necessary).

- How to efficiently start and stop the operation of a pipelined feedback loop is not always a trivial task (refer to Fig. 2.1). This design only needed three small considerations. First it incorporates the *switch* block at the end of the Paths Generation block to assist when maturity is reached and the LS block starts to operate. Second, the *CashFlow Update-1* module is aware that at maturity no regression coefficients are available. Third, the *Cashflow Update-2* module is also aware that the discounted CashFlow output will not be read when the initial time step is reached (and stops writing data to this output accordingly). Nevertheless, all other blocks in the architecture were design as in a normal pipelined operation.

A details regarding the BoxMuller block is also worth noticing. In Vivado HLS version 2014.3 the inferred instance *uitofp* delivers negative values when the most significant bit of the uint32_t input value is 1. This was posted in the Xilinx HLS forum, and the response was that it was solved in the latest version. The BoxMuller delivered in this project checks and corrects negative values.
A final remark is made on the overall project regarding current and future work. Based on the pricing of a two-dimensional American option presented in this report, there is an undergoing research on further optimizations and larger financial problems in which a hybrid CPU/FPGA device like the Xilinx Zynq is particularly suitable.

5 Appendix

For source code, please refer to the attached .zip file:

- IP blocks in folder HLS
- SDCard image and files in folder SDCard
- Driver source code and bitstream in folder Ccode_and_bitstream.

References

[1] Javier Alejandro Varela. Embedded Architecture to value American Options on the Stock Market. Master's thesis, Microelectronic Systems Design Reseach Group, Department of Electrical Engineering and Information Technology, University of Kaiserslautern, aug 2014.

[2] Christian Brugger, Javier Alejandro Varela, Norbert Wehn, Songyin Tang, and Ralf Korn. Reverse Longstaff-Schwartz American Option Pricing on hybrid CPU/FPGA Systems. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1599–1602, mar 2015.

[3] Francis A Longstaff and Eduardo S Schwartz. Valuing American options by simulation: A simple least-squares approach. *Review of Financial studies*, 14(1):113–147, 2001.

[4] John C. Hull. *Options, Futures, And Other Derivatives*. Pearson, 8th edition, 2012.

[5] Ralf Korn, Elke Korn, and Gerald Kroisandt. *Monte Carlo Methods and Models in Finance and Insurance*. Boca Raton, FL: CRC Press., 2010.

[6] Makoto Matsumoto and Takuji Nishimura. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[7] G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[8] Katsumi Hagita, Hiroshi Takano, Takuji Nishimura, and Makoto Matsumoto. Reverse Generator MT19937. Fortran source code, June 2000. last access 2014-09-16.

[9] Monetary and Economic Department. Statistical release: OTC derivatives statistics at end-December 2013. Technical report, Bank for International Settlements, may 2014.

[10] Xiang Tian and K. Benkrid. American Option Pricing on Reconfigurable Hardware Using Least-Squares Monte Carlo Method. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 263 –270, December 2009.

[11] Makoto Matsumoto. Mersenne Twister Home Page. `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`, January 2007. last access 2014-07-02.