A Runtime Reconfigurable Architecture for Monte Carlo Option Pricing in the Heston Model

HPC on embedded computing devices - a Case Study -

70% of US stock traded via computers
Financial Institution:
- 2000 exotic options / 5 Minutes
- IBM CPU-Cluster 2000 nodes
- Energy consumption per day
  - 20 MWh
  - ~ 1000 households
  - 11 t CO₂

Embedded Computing largely impacts HPC
- Application specific accelerators
- New devices like Zynq
Fast calculations of market prices for all sort of financial products e.g. options

Option: financial contract between buyer and seller
- Buyer: option to buy a share from a seller at a certain time for a certain price
- Finance market: no arbitrage, i.e. making profit with no risk not possible
- Option is not for free ⇒ option price = discounted expected payoff

![Graph showing stock price and payoff](image)

\[ \text{Payoff} \]

\[ \text{Price: 49.25} \]
\[ \text{Profit: 4.02} \]

**Black-Scholes Model:**

\[ dS_t = \mu S_t dt + \sigma dW_t \]

- \( S_t \): price of the asset/share
- \( \mu \): drift, \( \sigma \): volatility, \( W_t \): Brownian motion

![Graph showing stock price timeline](image)

7th July 2014

Price: 49.25
Profit: 4.02
$45.23
Right to buy
Heston market model (1993)

- Volatility in BS model is constant over time
- Crash situations can not be modeled
- State-of-the-art: Heston model
- BUT: in general no closed analytical solution
  Monte Carlo simulations mandatory

\[ \begin{align*}
    \text{d} S_t &= \mu S_t \text{d} t + \sqrt{V_t} S_t \text{d} W^a_t \\
    \text{d} V_t &= \kappa (\theta - V_t) \text{d} t + \sigma \sqrt{V_t} \text{d} W^b_t \\
    \log(S_{t+1}) &= \log(S_t) + (\mu - \frac{1}{2} \sigma^2) \Delta t + \sigma \sqrt{V_t} \sqrt{\Delta t} \text{d} W^a_t \\
    V_{t+1} &= V_t + \kappa (\theta - V_t) \Delta t + \sigma \sqrt{V_t} \sqrt{\Delta t} \text{d} W^b_t
\end{align*} \]

Average Profit: 4.02
$45.23

Complexity Analysis in MC

Price calculation: sample mean of simulated instances of the payoff function \( g(S) \)

\[ \mathbb{E}[g(S)] \approx \frac{1}{N} \sum_{i=1}^{N} g(\hat{S}_i) + \Delta_{\text{statistic}} + \Delta_{\text{discretization}} \]

\[ \text{Cost: } O(N \times k) \]
Multilevel Monte Carlo (2008)

Sequence of MC simulations with different discretization and number of paths
- Trading-off statistic and discretization error on various levels
- Lower levels: coarse discretization, large variance $\Rightarrow$ large $N$ of low complexity
- Higher levels: finer discretization, small variance $\Rightarrow$ small $N$
$\Rightarrow$ MLMC: for same accuracy less computational complexity as SMC

HyPER Platform for Option Pricing

Modular pricing system for wide range of option prize calculations
Combines current trends from technology and computational stochastics
- Heston model, multilevel Monte Carlo, Xilinx Zynq platform
- New hardware oriented optimization: mixed precision
HyPER Architecture

Open platform which is split into Frontend and Backend
- Frontend (HW): MC path generation & path dependent Payoff functions
- Backend (SW): path independent functions

Integrate new products with systematic approach

Flexible HW/SW Split in Hyper
- MLMC level 5: features generated every 1024th clock cycle ⇒ SW
- MLMC level 1: features generated every 4th clock cycle ⇒ HW
HyPER – Static Optimizer

Find optimum platform configuration for given financial products and target platform

Optimization problem: maximize HyPER Performance under constraints

Choices
- Number of HyPER instances
- Number of frontends
- Position of HW/SW split
- Type of CPU/FPGA communication core

Constraints
- FPGA resources
- CPU Load
- Bandwidth CPU/FPGA

⇒ ILP formulation: calculate optimum architecture $H^*_1$ for each level

HyPER Optimization

Characterize basic building blocks

<table>
<thead>
<tr>
<th>Building Blocks</th>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
<th>Act/Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment Generator</td>
<td>301</td>
<td>233</td>
<td>4</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Monotonic Twrce</td>
<td>197</td>
<td>199</td>
<td>4</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Arithmetic Core</td>
<td>226</td>
<td>259</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Pack Generator</td>
<td>513</td>
<td>541</td>
<td>0</td>
<td>45</td>
<td>-</td>
</tr>
<tr>
<td>High-Level Task</td>
<td>145</td>
<td>241</td>
<td>2</td>
<td>38</td>
<td>-</td>
</tr>
<tr>
<td>Parallel Processing</td>
<td>190</td>
<td>198</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Parallel Reduction</td>
<td>899</td>
<td>780</td>
<td>0</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Block Adder (k=5)</td>
<td>392</td>
<td>372</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Block Adder (k=6)</td>
<td>392</td>
<td>372</td>
<td>0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

Hardware Parameters
- FPGA capacity 575
- Clock Speed 100 MHz
- Bandwidth 40 Gbps

Optimized architecture for each level

Static optimization (ILP):
- Bitstreams
  - Reconfigure at runtime
Mixed Precision Format in MLMC

Accuracy: accuracy of the option prize algorithm
Precision: accuracy of the atomic operation

Standard approach: start with available precision of computing hardware, e.g., single/double precision floating point. Then tune your algorithm to achieve the desired algorithm accuracy.

Our approach: start with desired algorithm accuracy. Then tune computation precision to get the desired algorithm accuracy.

⇒ Use customized precision for each level in MLMC

Single-Precision

\[
\text{value} = (-1)^{\text{sign}} \times (1 + \text{fraction}) \times 2^{\text{exponent} - 127}
\]

Example: MC Step generator & RNG

Customized Precision Results

Fast and efficient on-line algorithm to calculate precision for each level for given option prize accuracy

- Based on variance of “exact” MLMC: max 10% deviation

Different precision for each Level
Results

CPU i5-3320M@22nm

Runtime (Energy) in Minutes: pricing 1000 options with 120 Watt at same accuracy

Classical MC

Multilevel MC

3.8x

Multilevel HyPER

Mixed Precision HyPER

FPGA Zynq 7020

12.7x

Classical MC

470 125 9.8 3.4 1.7

For more information please visit
http://ems.eit.uni-kl.de