ACCELERATION OF DISTANCE-TO-DEFAULT ON FPGA

Izaan Allugundu, Pranay Puranik, Yat Piu Lo, Akash Kumar
DISTANCE-TO-DEFAULT

• Distance-to-Default (DTD) is one of the twelve indicators used in determining a firm’s probability-to-default
• Measures the qualitative “distance” till a firm defaults on its debt
• Also known as the leverage indicator
DISTANCE-TO-DEFAULT (DTD)

• Accuracy depends on the amount of historical data
• More data -> More computation time & costs
• Current setup runs on 200-node cluster using Matlab taking 2 days for over 50,000 firms
• Mostly linear workflow -> easy to pipeline and parallelize
• FPGA is a good candidate!
CONTRIBUTIONS

• Develop a hardware-software accelerated approach to compute the distance-to-default
  – Hardware modules to accelerate the computation of the Implied Asset Value and Log-Likelihood functions
  – Software module to implement the minimum solver
  – Data transfer between data source running solver and hardware accelerator platform
  – Software interface between the hardware and software platforms
OVERVIEW OF DTD
OVERVIEW OF DTD DATAFLOW

- **Stage 1** – Obtain firm’s parameters where log-likelihood is minimum
  - Repeated process through numerical search
  - Evaluates the log-likelihood function
- **Stage 2** – Calculate Distance-to-Default
IMPLEMENTATION OVERVIEW

- Implementation of DTD
  - Dataflow in software
    - MIDACO solver on PC
    - Ethernet data transfer between PC/FPGA
    - Log-likelihood computation on FPGA
  - Dataflow enhancements with accelerators
    - Implied asset value computation
    - Log-likelihood computation
OVERVIEW OF HARDWARE MODULES
IMPLIED ASSET VALUE

• Expected value of the asset given the historic equity value, debt, interest rate with varying volatility and time-to-maturity

• Data-path consists of
  – Stream serialiser and de-serialiser
  – Pipeline loopback and feed control
  – Computation function / model function
OVERVIEW OF IMPLIED ASSET COMPUTATION FUNCTION

• Computation function consist of 4 basic operations
  – Initial search for...
    • Upper bound
    • Lower bound
  – Search for convergence value using numerical analysis methods
    • Bisection
    • Newton-Raphson
OVERVIEW OF LOG-LIKELIHOOD FUNCTION
DESIGN DECISIONS

Hardware Layer

National University of Singapore
DESIGN DECISIONS
AREAS FOR SPEEDUP

• There are several areas where we can optimise FPGA designs over sequential code
  – Simplification and combining of operations
  – Parallel operations
  – Pipelined operations
DESIGN DECISIONS
SIMPLIFICATION AND COMBINING OF OPERATIONS

• Check for a percentage difference in values, i.e. $x/y < e$ or $x/y < 0.1$
• Constant Multiplications
• Negation and absolute
• Multiply or division by 2
• Reduces the need for complex hardware and high latencies for simple operations
DESIGN DECISIONS
PARALLEL OPERATIONS

• Some arithmetic operations can be carried out concurrently
  – Computation of normal CDF values
  – Generation of terminating conditions
  – Computation of Nd₁, Sqterms and LogVA values
  – Addition of these results and storage in accumulators

→ Results in reduced pipeline latency and delay logic
DESIGN DECISIONS

PIPELINED OPERATIONS

• Pipelined operations provides a big boost in performance
• FloPoCo used to generate cores
## DESIGN DECISIONS

### PIPELINED OPERATIONS

<table>
<thead>
<tr>
<th>Operation</th>
<th>MicroBlaze (Speed)</th>
<th>MicroBlaze (Area)</th>
<th>FlopocoCore* + CustomOps</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpadd</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>9 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpsub</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>9 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpmul</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>9 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpconstmul</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>&lt;9 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpdiv</td>
<td>28 cycles</td>
<td>30 cycles</td>
<td>17 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpcmp</td>
<td>1 cycles</td>
<td>3 cycles</td>
<td>1 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpsqrt</td>
<td>27 cycles</td>
<td>29 cycles</td>
<td>-</td>
</tr>
<tr>
<td>fpsquare</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>7 cycles [1 cycle]</td>
</tr>
<tr>
<td>fplog</td>
<td>Depends on CMath Library</td>
<td></td>
<td>9 cycles [1 cycle]</td>
</tr>
<tr>
<td>fexp</td>
<td></td>
<td></td>
<td>24 cycles [1 cycle]</td>
</tr>
<tr>
<td>fpmulti2</td>
<td>4 cycles</td>
<td>6 cycles</td>
<td>1 cycle (C) [1 cycle]</td>
</tr>
<tr>
<td>fdiv2</td>
<td>4 cycles / 28 cycles</td>
<td>6 cycles / 30 cycles</td>
<td>1 cycle (C) [1 cycle]</td>
</tr>
</tbody>
</table>
DESIGN DECISIONS - IMPLIED ASSET VALUE SERIALISERS AND DATA TAGS

• Different number of iterations per data set means that output data is not in FIFO order
  – Requires the implementation of a data-tag
  – Stores the index position of the data

• The data between various pipeline stages stored in BRAMs instead of distributed memory
DESIGN DECISIONS - LOG-LIKELIHOOD FUNCTION

- Use of buffer registers for outputs
- Use of 1 module each for Nd1, Sqterms, LogVA

- Software sections of Log-Likelihood function placed in BRAMs to speed up computation
MIDACO SOLVER & DATA TRANSFER

Software Layer
MINIMUM SOLVER

• Solver is used to determine the parameters $\mu$, $\sigma$ and $\delta$.

• Parameters are key to computing DTD

• Established by constrained optimization of log-likelihood function
OLD SYSTEM

• MATLAB function ‘fmincon’ from the Optimization toolbox is used.
  – fmincon attempts to find a constrained minimum of a scalar function of several variables starting at an initial estimate. This is generally referred to as constrained nonlinear optimization or nonlinear programming.
  – Source: www.mathworks.com

• Function requires thousands of evaluations
IMPLEMENTATION

• Mixed Integer Distributed Ant Colony Optimization (MIDACO) was adapted to our system implementation.
• Unique set-up; optimization on PC but each individual evaluation on Hardware
• For 1000 evaluations of function, solver takes 5.83 ms (self-time).
• In our setup, MATLAB takes 257 ms while MIDACO takes 46.63 ms (8000 evaluations).
• Data transferred through Ethernet interface
## AREA UTILIZATION

<table>
<thead>
<tr>
<th>Module</th>
<th>Slices</th>
<th>LUTs</th>
<th>BRAM</th>
<th>DSP48E1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze (reference)</td>
<td>1,685</td>
<td>3,021</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>Normal CDF</td>
<td>3,240</td>
<td>7,976</td>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>Exponential</td>
<td>290</td>
<td>605</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Implied Asset Value</td>
<td>10,434</td>
<td>25,723</td>
<td>18</td>
<td>71</td>
</tr>
<tr>
<td>Log-Likelihood Function</td>
<td>25,258</td>
<td>25,580</td>
<td>11</td>
<td>74</td>
</tr>
</tbody>
</table>
### IMPLIED ASSET VALUE

<table>
<thead>
<tr>
<th>Platform</th>
<th>PC (2.9 GHz)</th>
<th>FPGA (100 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MATLAB</td>
<td>Virtex6 (HDL)</td>
</tr>
<tr>
<td>Sets computed</td>
<td>511</td>
<td></td>
</tr>
<tr>
<td>Sets per Second</td>
<td>21,043</td>
<td>348,619</td>
</tr>
<tr>
<td>Cycles taken</td>
<td>711 e5</td>
<td>1.47 e5</td>
</tr>
<tr>
<td>Cycles per Set</td>
<td>139,050</td>
<td>287</td>
</tr>
<tr>
<td>Relative performance</td>
<td>1.00X</td>
<td><strong>16.6X</strong></td>
</tr>
</tbody>
</table>
## LOG LIKELIHOOD FUNCTION

<table>
<thead>
<tr>
<th>Platform</th>
<th>PC (2.9 GHz)</th>
<th>FPGA (100 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MATLAB</td>
<td>Virtex6 (HDL)</td>
</tr>
<tr>
<td>Outputs per Second</td>
<td>79</td>
<td>25040</td>
</tr>
<tr>
<td>Cycles taken</td>
<td>1080 e5</td>
<td>1.12 e5</td>
</tr>
<tr>
<td>Time taken (s)</td>
<td>0.038</td>
<td>0.0001</td>
</tr>
<tr>
<td>Relative performance</td>
<td>1.00X</td>
<td>317X</td>
</tr>
</tbody>
</table>
CONCLUSIONS

• Compute intensive parts moved to FPGA
• The hardware implementation speed-up over PC
  – Implied asset value: 16.6x
  – Log-likelihood function: 317.17x
• Data communication still the bottleneck
• Students working on such project end up with high-paying jobs in banks ;)

28
FUTURE WORK

• Conversion of the MIDACO solver to HDL to further accelerate the system.
• Using double precision floating point values for greater accuracy.
• Further optimization of the cores and the hardware-software interface to achieve a greater speed-up
THANK YOU