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!

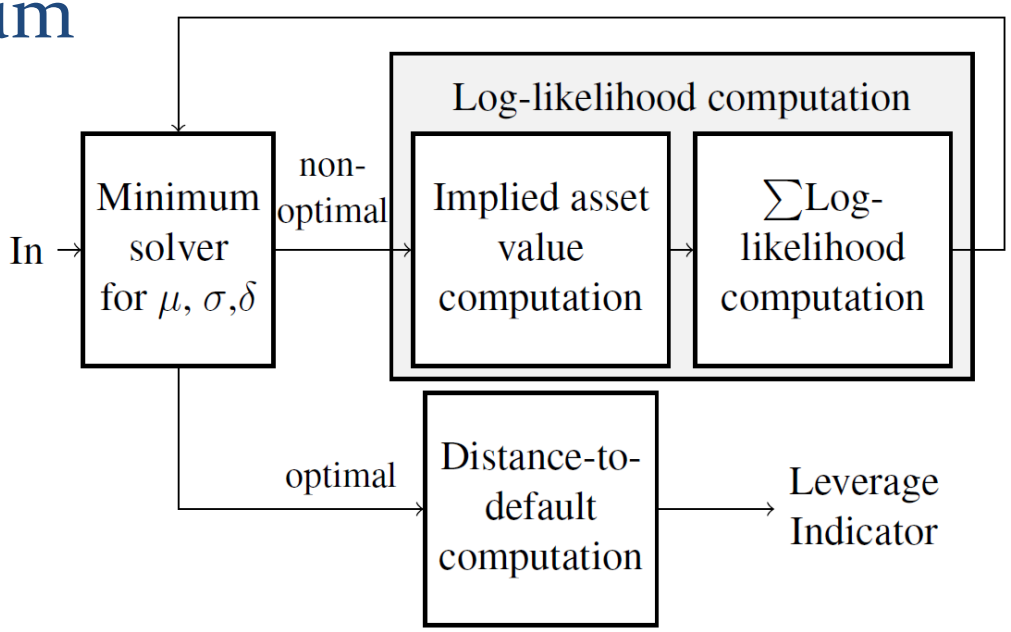
CONTRIBUTIONS

- 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

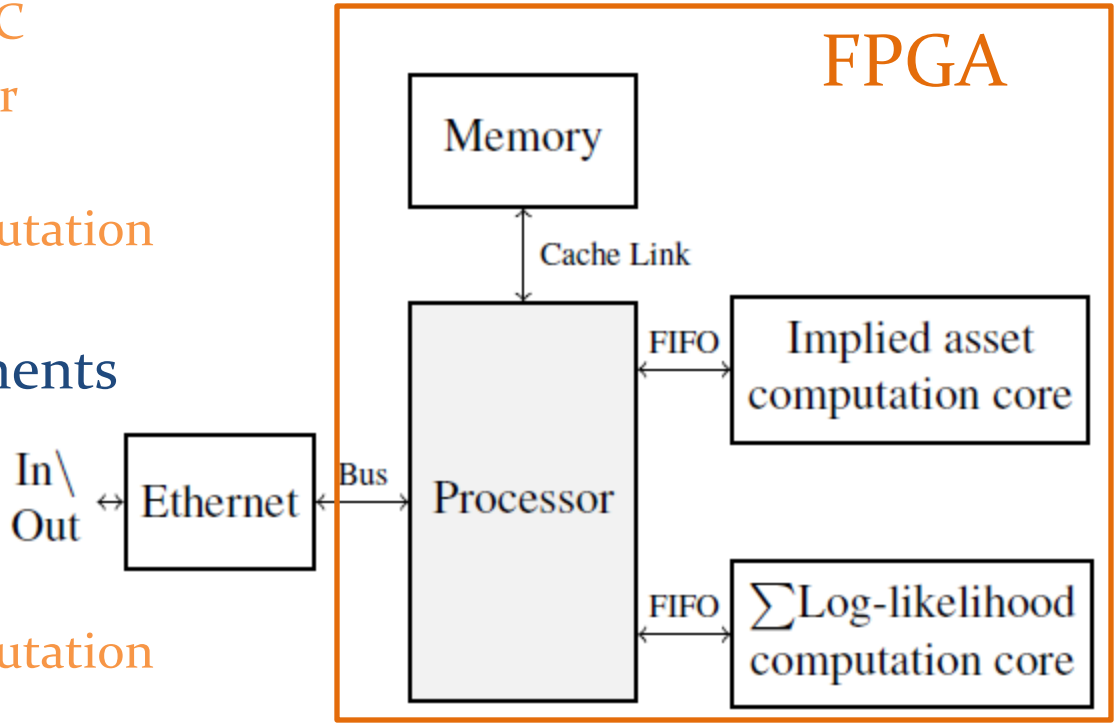
OVERVIEW OF DTD DATAFLOW

- 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

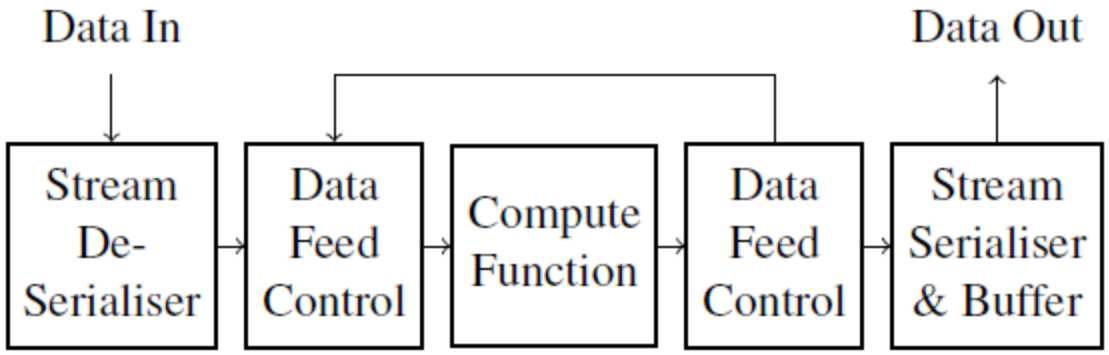


OVERVIEW OF HARDWARE MODULES

NUS
National University
of Singapore

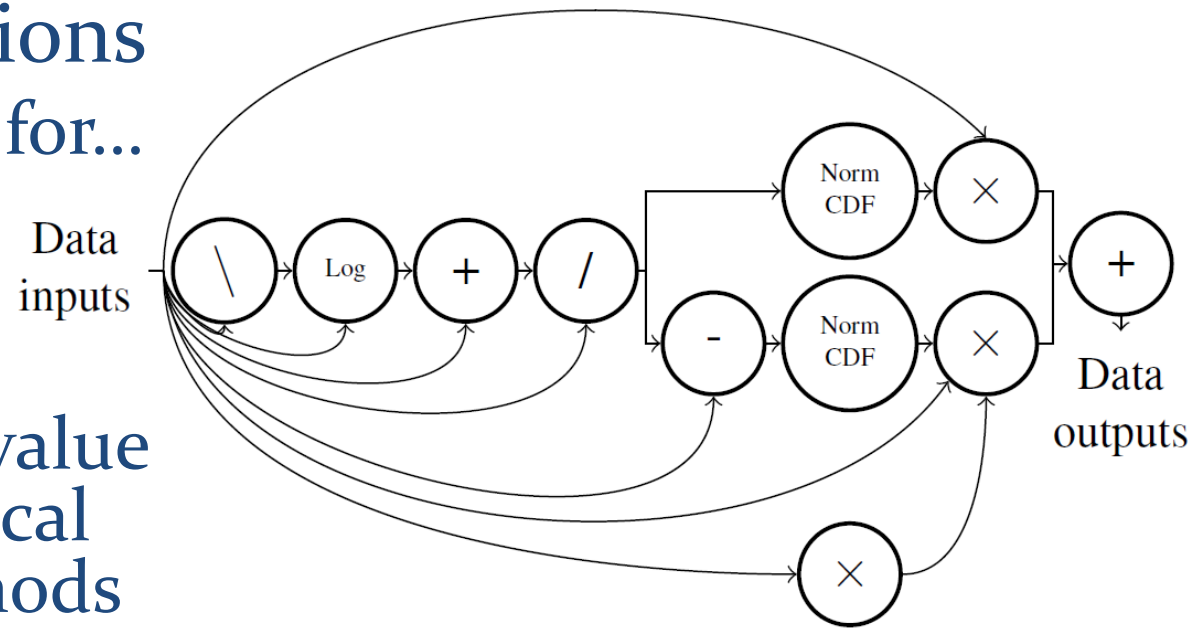
IMPLIED ASSET VALUE

- 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

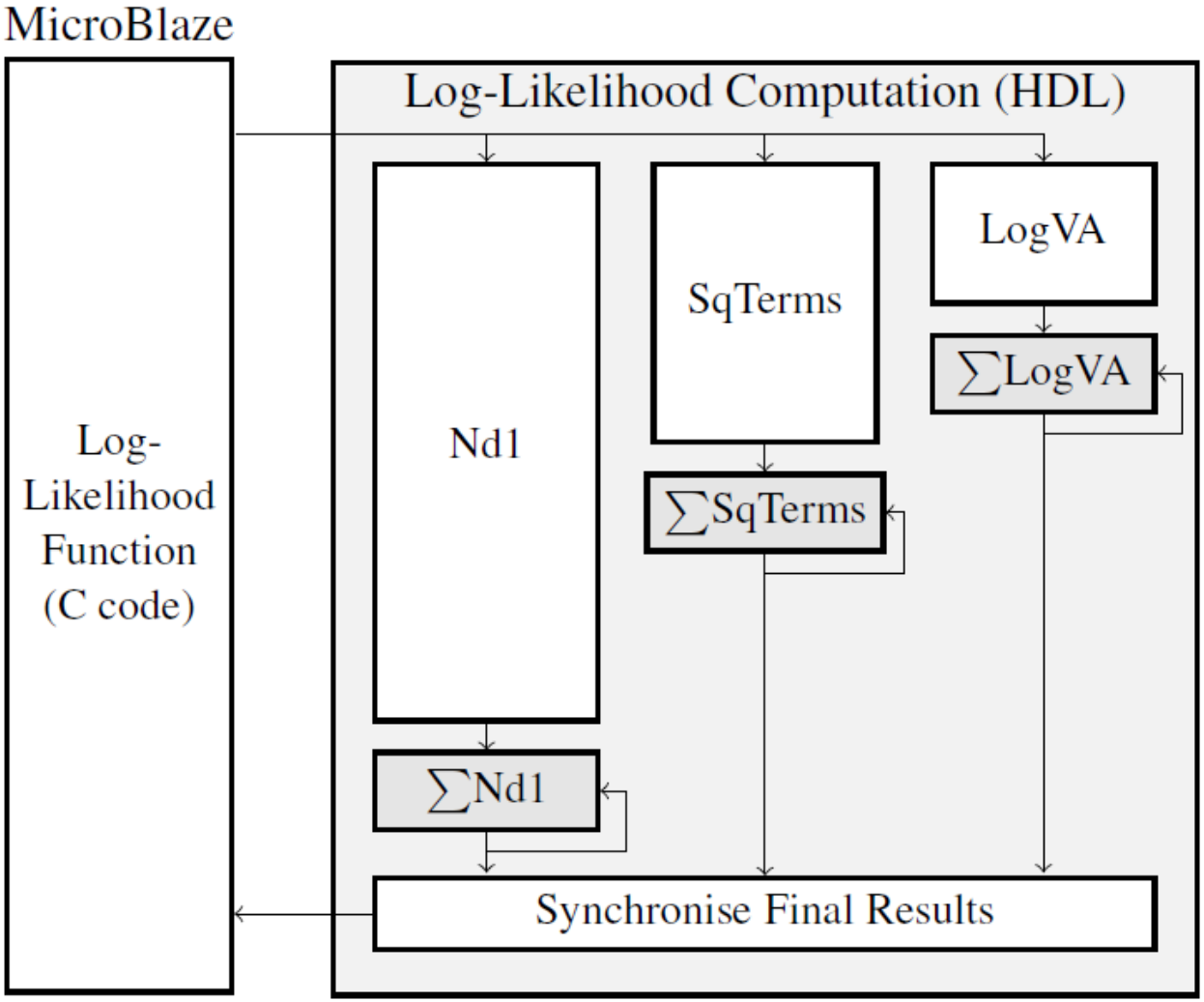


OVERVIEW OF IMPLIED ASSET COMPUTATION FUNCTION

- 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

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

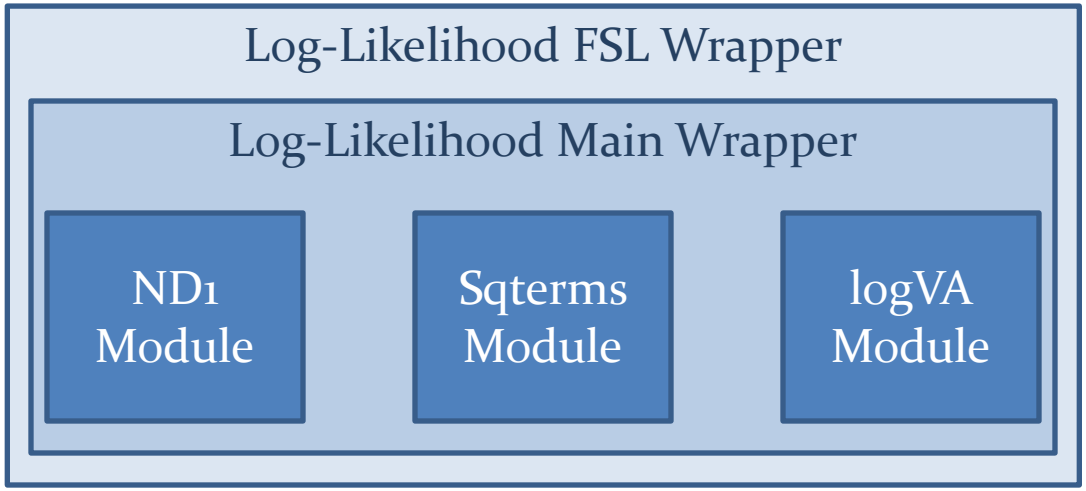
Operation	MicroBlaze (Speed)	MicroBlaze (Area)	FlopocoCore* + CustomOps
fpadd	4 cycles	6 cycles	9 cycles [1 cycle]
fpsub	4 cycles	6 cycles	9 cycles [1 cycle]
fpmul	4 cycles	6 cycles	9 cycles [1 cycle]
fpconstmul	4 cycles	6 cycles	<9 cycles [1 cycle]
fpdiv	28 cycles	30 cycles	17 cycles [1 cycle]
fpcmp	1 cycles	3 cycles	1 cycles [1 cycle]
fpsqrt	27 cycles	29 cycles	-
fpsquare	4 cycles	6 cycles	7 cycles [1 cycle]
fplog	Depends on CMath Library		9 cycles [1 cycle]
fpexp			24 cycles [1 cycle]
fpmulti2	4 cycles	6 cycles	1 cycle (C) [1 cycle]
fpdiv2	4 cycles / 28 cycles	6 cycles / 30 cycles	1 cycle (C) [1 cycle]

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 μ, σ and δ.
- 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

RESULTS & CONCLUSIONS

AREA UTILIZATION

Module	Slices	LUTs	BRAM	DSP48E1
MicroBlaze (reference)	1,685	3,021	20	5
Normal CDF	3,240	7,976	1	27
Exponential	290	605	1	1
Implied Asset Value	10,434	25,723	18	71
Log-Likelihood Function	25,258	25,580	11	74

IMPLIED ASSET VALUE

Platform	PC (2.9 GHz)	FPGA (100 MHz)
Type	MATLAB	Virtex6 (HDL)
Sets computed	511	
Sets per Second	21,043	348,619
Cycles taken	711 e5	1.47 e5
Cycles per Set	139,050	287
Relative performance	1.00X	16.6X

LOG LIKELIHOOD FUNCTION

Platform	PC (2.9 GHz)	FPGA (100 MHz)
Type	MATLAB	Virtex6 (HDL)
Outputs per Second	79	25040
Cycles taken	1080 e5	1.12 e5
Time taken (s)	0.038	0.0001
Relative performance	1.00X	317X

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 ;)

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