

# FPGA Acceleration of Monte-Carlo Based Credit Derivatives Pricing

Alexander Kaganov<sup>1</sup>,  
Asif Lakhany<sup>2</sup>, Paul Chow<sup>1</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, University of Toronto

<sup>2</sup> Quantitative Research, Algorithmics Incorporated

# Increasing Computational Requirements (1/3)

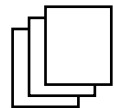
In recent years the financial industry has seen:

1. Increasing contract/model complexity
  - Every year new models are developed
  - Unavailability of closed-form solution
  - Necessitate Monte-Carlo pricing

# Increasing Computational Requirements (2/3)

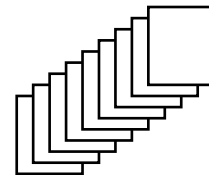
## 2. Increasing portfolio sizes

- Increase in simple instruments
  - Bonds
  - Loans
- Increase in complex derivate security
  - CDO issuance has increased from \$157 billion in 2004 to \$507 billion in 2007 ( $>3x$ )<sup>1</sup>



**N** instruments

**Y** time



**3xN** instruments

**3xY** time (at least)

<sup>1</sup> SIFMA

# Increasing Computational Requirements (3/3)

## 3. Ever-present need to make **real-time decisions**

- Market trends can change quickly
- Instruments traded electronically

**1 ms in Latency is Worth \$100 M in Stock Trading Business Value** (AMD

Analyst Day-26 July 2007)

# Trends in Financial Monte-Carlo Algorithms

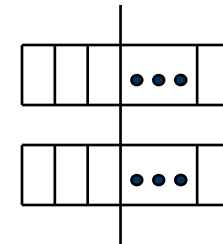
## 1. Computationally intensive

- Converges in  $\frac{1}{\sqrt{N}}$

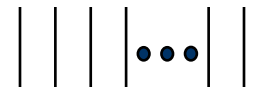
## 2. Highly repetitive

- A large portion of the calculation time is spent in a small portion of the code
  - (~90% of the time is spent in ~10% of the code)

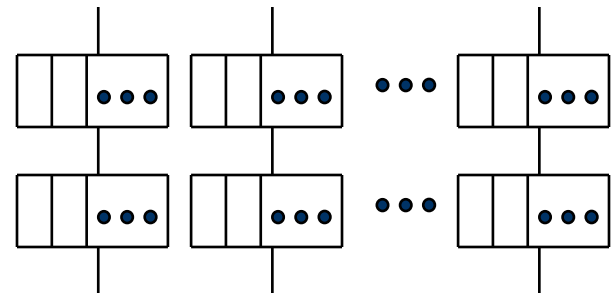
## 3. High degree of coarse and fine-grain parallelism



Fine-Grain



Coarse-Grain



Typical MC Financial simulation

# **Collateralized Debt Obligation (CDO)**

# CDO

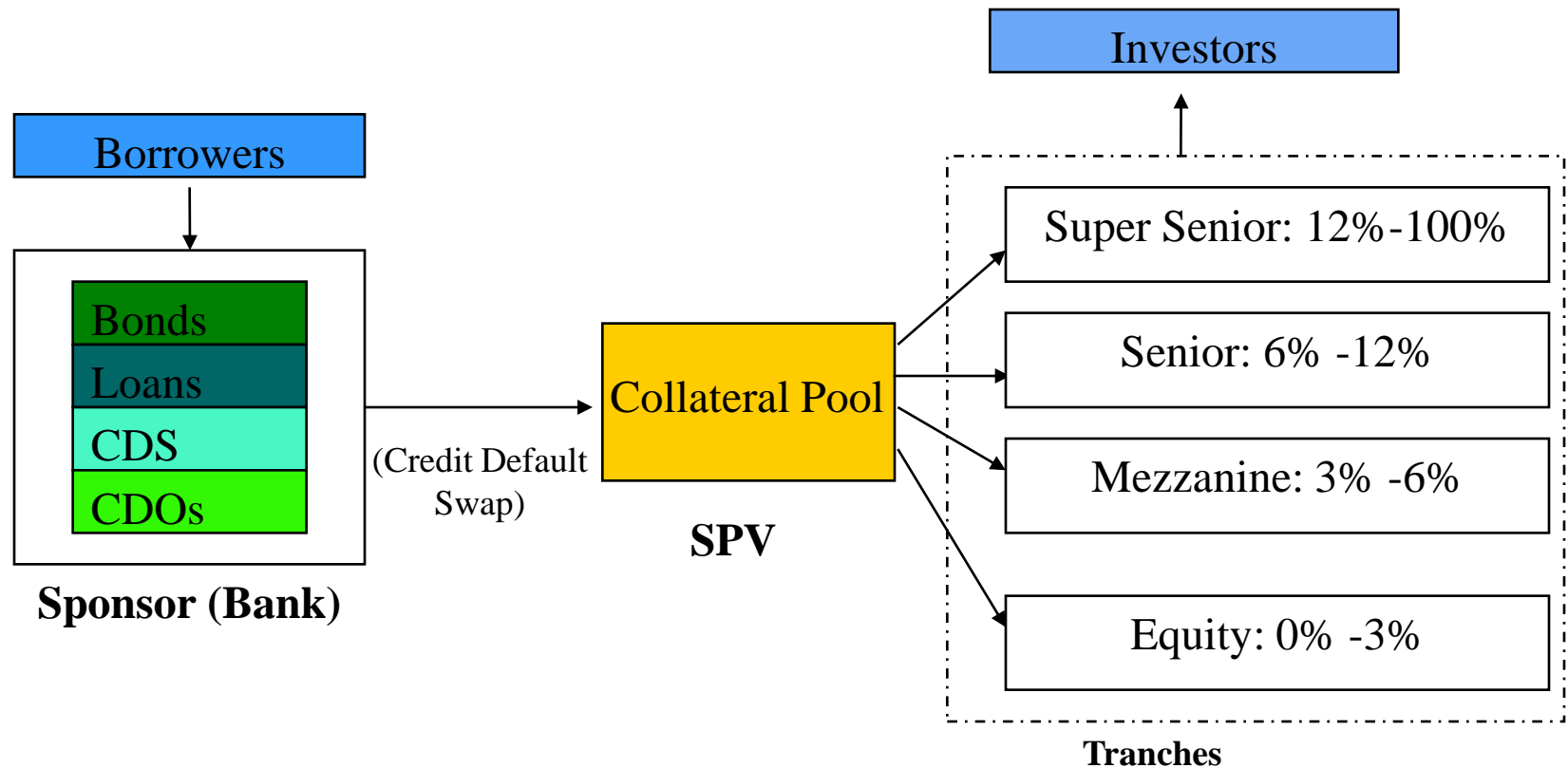
## Problem:

- Banks typically hold portfolios with highly volatile assets.

## Solution:

- Sell assets to an outside entity (SPV), which combines the different assets together into one collateral pool
- Repackage the pool as CDO tranches.
- Sell tranches as form of protection to investors in return for premium payments

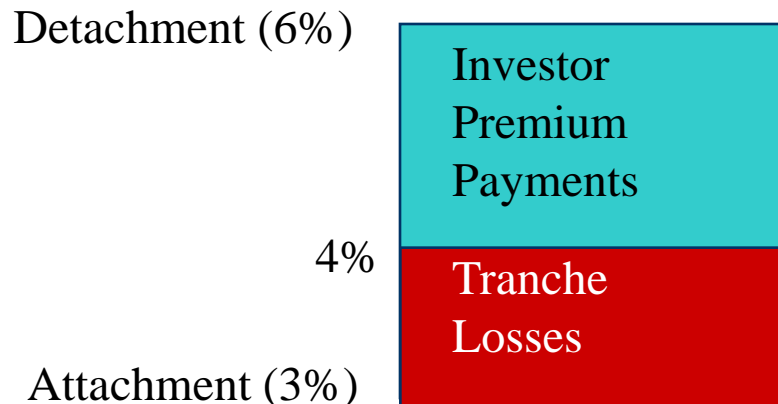
# CDO Structure (1/2)



# CDO Structure (2/2)

- Each tranche has attachment and detachment points
  - Losses below attachment point → the tranche is unaffected
  - Losses above the detachment point → the tranche becomes inactive
- Investor premium is paid based on the tranche width minus tranche losses

## Mezzanine Tranche:



- Losses 1/3 of the principal investment. Paid based on 2/3 of the original investment

# Pricing a CDO

- Default Leg: expected losses of the tranche over the life of the contract
- Premium Leg: expected premiums that the tranche investor will receive over the life of the contract

**CDO Tranche Value = Premium Leg – Default Leg**

$$= E \left[ \sum_{i=1}^T s_i (S - L_i) d_i \right] - E \left[ \sum_{i=1}^T (L_i - L_{i-1}) d_i \right]$$

S =tranche thickness

s<sub>i</sub>= Premium

d<sub>i</sub>= Discount factor

L<sub>i</sub>= Tranche loses at time interval i

# Li's One-Factor Gaussian Copula (OFGC) Model

- Calculate total losses by averaging over all Monte-Carlo (MC) paths
- For each path:

1. Generate:

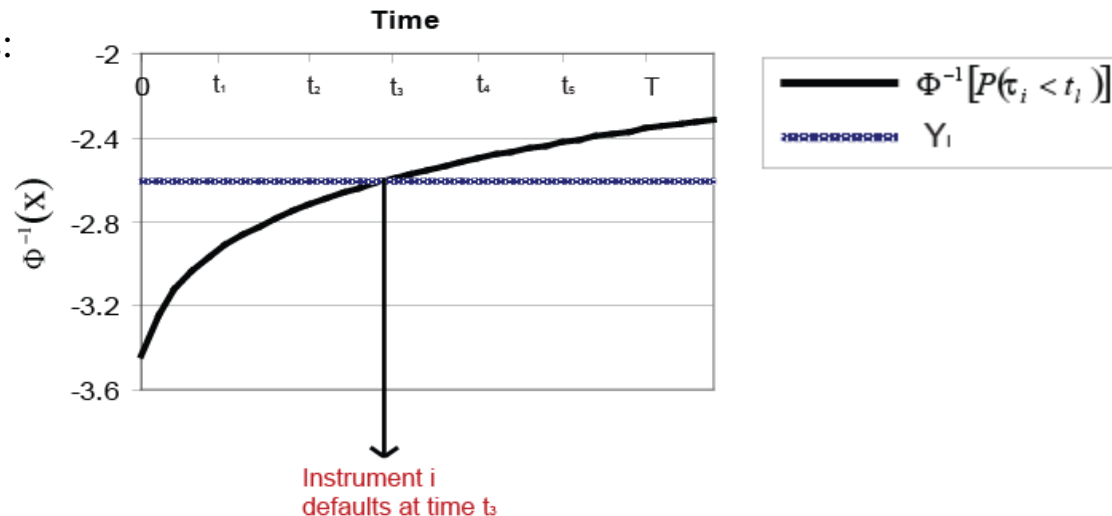
$$Y_i = \alpha_i X + \sqrt{1 - \alpha_i^2} Z_i$$

Systemic Factor      Idiosyncratic Factor

2. Compare:

$$Y_i < \Phi^{-1}[P(\tau_i < t)]$$

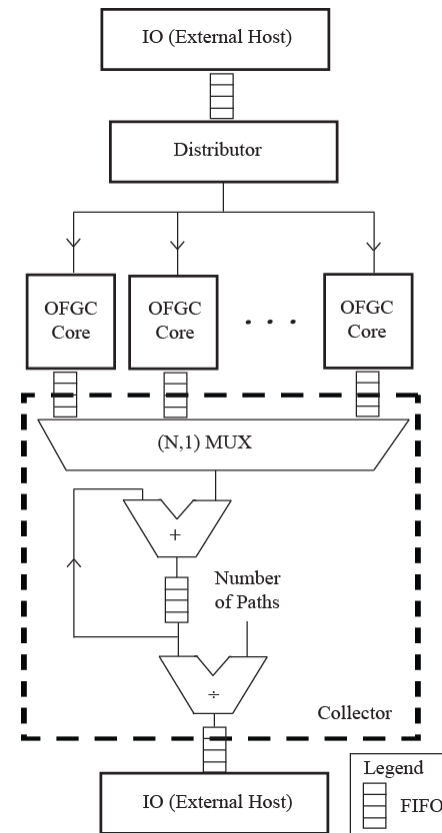
3. Record losses:



# Implementation

# Multi-Core Architecture

- Three portions: Distributor, OFGC pricing cores, and Collector.
- All cores have the same input data except for market scenarios
- Coarse Grain Parallelism: MC paths divided among OFGC cores
- Data transfer occurs in parallel to calculations
  - Double Buffering
- Maximal required data transfer rate of: 24MBytes/sec
  - 1-Lane PCI express- 250 MBytes/sec
  - **Data transfer latency can be hidden**



# OFGC Design

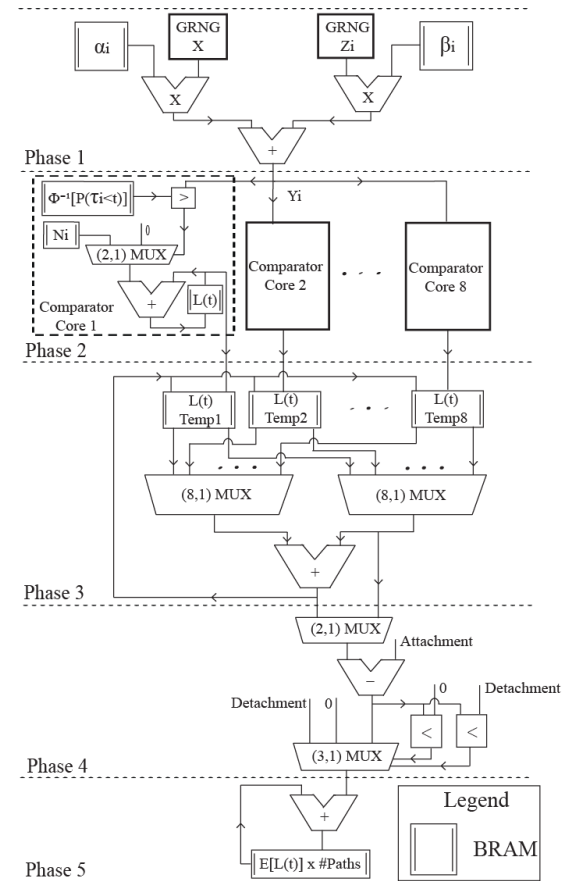
Phase 1: Generate  $Y_i$

Phase 2: Compare  $Y_i < \Phi^{-1}[P(\tau_i < t)]]$ . Record partial losses

Phase 3: Combine the partial sums,  $L(t_j)$ 's.

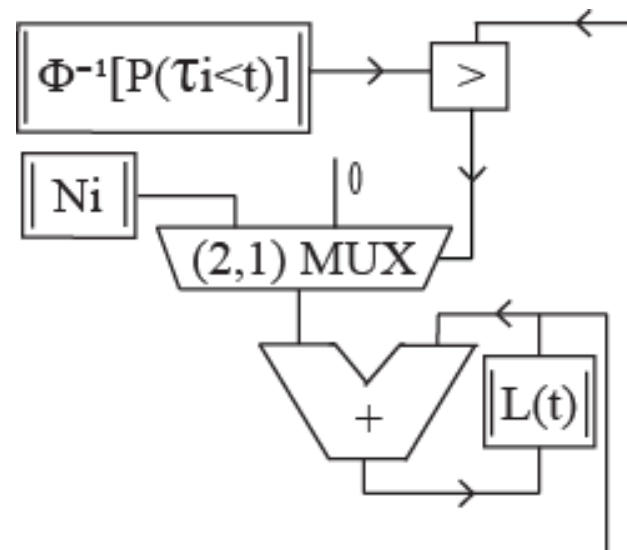
Phase 4: Convert collateral pool losses to tranche losses

Phase 5: Accumulate tranche losses



# Phase 2

- Compare  $Y_i < \Phi^{-1}[P(\tau_i < t)]$ .  
Record Losses
- Fine-grain parallelism:  
parallelize over time
  - 8 replicas
- More replicas  $\rightarrow$  higher speedup (potentially)
  - However, large portions of the hardware become underutilized
- Pipelined adder latency creates multiple partial sums



# OFGC Design

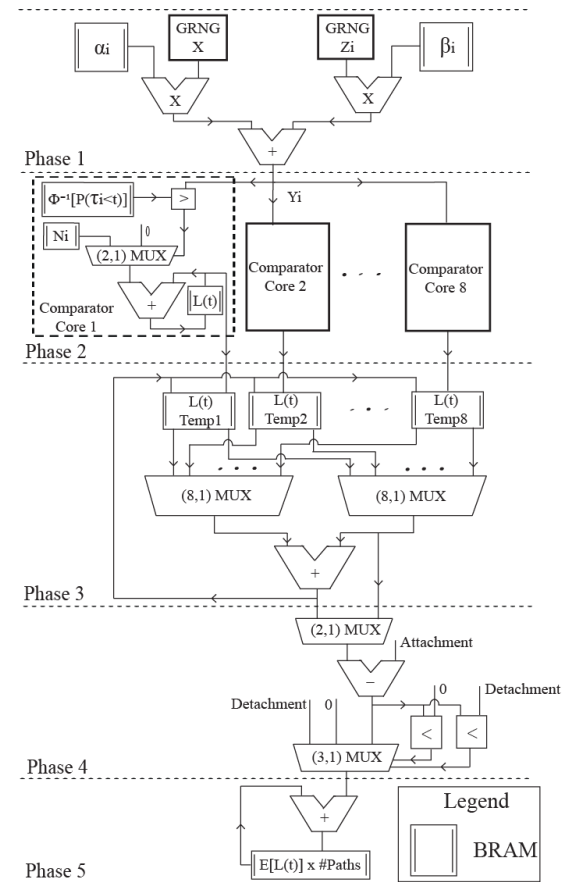
Phase 1: Generate  $Y_i$

Phase 2: Compare  $Y_i < \Phi^{-1}[P(\tau_i < t)]]$ . Record partial losses

Phase 3: Combine the partial sums,  $L(t_j)$ 's.

Phase 4: Convert collateral pool losses to tranche losses

Phase 5: Accumulate tranche losses



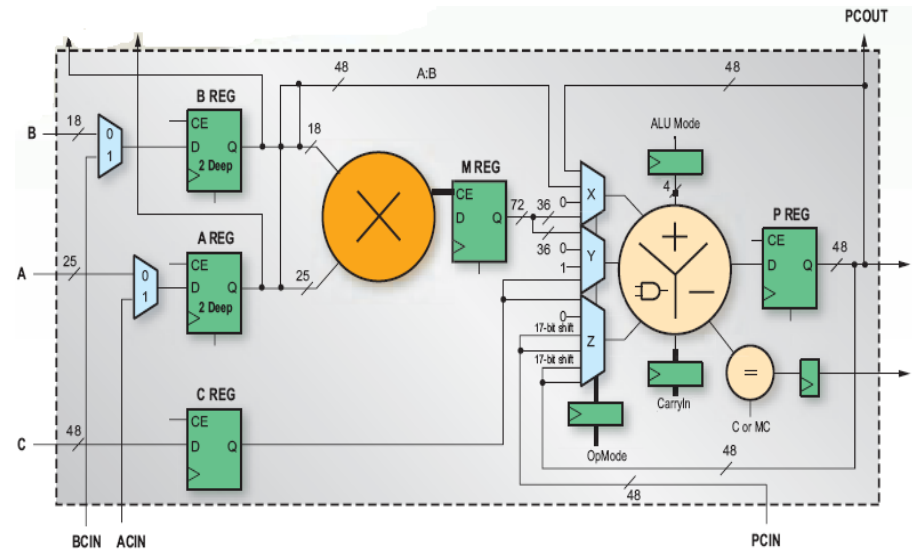
# Experiments and Results

- Three notional representations were explored: floating-point single-precision, double-precision, and fixed-point.
  - Floating-Point DSP exploration
  - Single-Precision/Double-Precision Hybrid
  - Fixed-Point
- Performance Results

# Floating-Point DSP Exploration: DSP48E Background

- Highly optimized slices dedicated to arithmetic operations
- Potential clock frequency 550 MHz
- Support for over 40 operating modes:

- multiplier
- multiplier-accumulator
- three input adder
- barrel shifter
- wide bus multiplexers
- etc



Virtex 5 DSP48E Slice Diagram<sup>1</sup>

<sup>1</sup> Diagram taken from Xilinx website

# Floating-Point DSP Exploration: Results

	Floating-Point Single-Precision			Floating-Point Double-Precision	
	Without DSP	With DSP		Without DSP	With DSP
Flip-Flops	7097	6530 (-8.0%)	Flip-Flops	10454	9910 (-5.2%)
LUTs	8660	7052 (-18.6%)	LUTs	13548	13325 (-1.6%)
BRAMs	15	15	BRAMs	31	31
DSP48Es	9	29 (+222%)	DSP48Es	10	40 (+300%)
Frequency	235.2	248.8 (+5.8%)	Frequency	187.3	190.9 (+1.9%)
Average Error (%)	0.39 [1.07]		Average Error (%)	0	

Single-Precision is 1.5 to 2 times smaller but has an accuracy error

# Single-Precision/Double-Precision Hybrid

- Combine the accuracy of the double-precision and resource utilization of single-precision
  - Single-precision notionals and double-precision accumulator at phase 5

	Single Precision	Hybrid
Flip-Flops	6530	6721 (+2.9%)
LUTs	7052	7599 (+7.8%)
BRAMs	15	15
DSP48Es	29	30 (+3.4%)
Frequency	248.8	244.8 (-1.6%)
Average Error (%)	0.37 [1.07]	3.02E-5 [5.27E-5]

# Fixed-Point

- 42-bit notionals, 54-bit final accumulator matches the accuracy of a double-precision design
- Each additional notional bit requires 62 Flip-Flops and 74 LUTs.

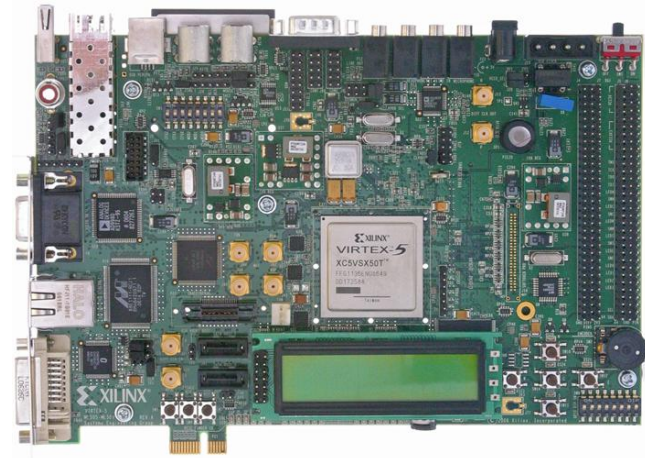
	Single Precision	Fixed-Point
Flip-Flops	6530	4906 <b>(-24.9%)</b>
LUTs	7052	5224 <b>(-25.9%)</b>
BRAMs	15	15
DSP48Es	29	7 <b>(-75.9%)</b>
Frequency	248.8	268.2 <b>(+7.8%)</b>
Average Error (%)	0.37 [1.07]	0

# Performance: Benchmarks

#	Based on Data From	# of Assets	# of Time Steps	# of Default Curves
1	CDX.NA.HY	100	15	5
2	CDX.NA.IG	125	35	5
3	CDX.NA.IG.HVOL	30	19	4
4	CDX.NA.XO	35	22	4
5	CDX.EM	14	6	4
6	CDX.DIVERSIFIED	40	23	5
7	CDX.NA.HY.BB	37	13	4
8	CDX.NA.HY.B	46	26	4
9	Semi-homogenous	400	24	2

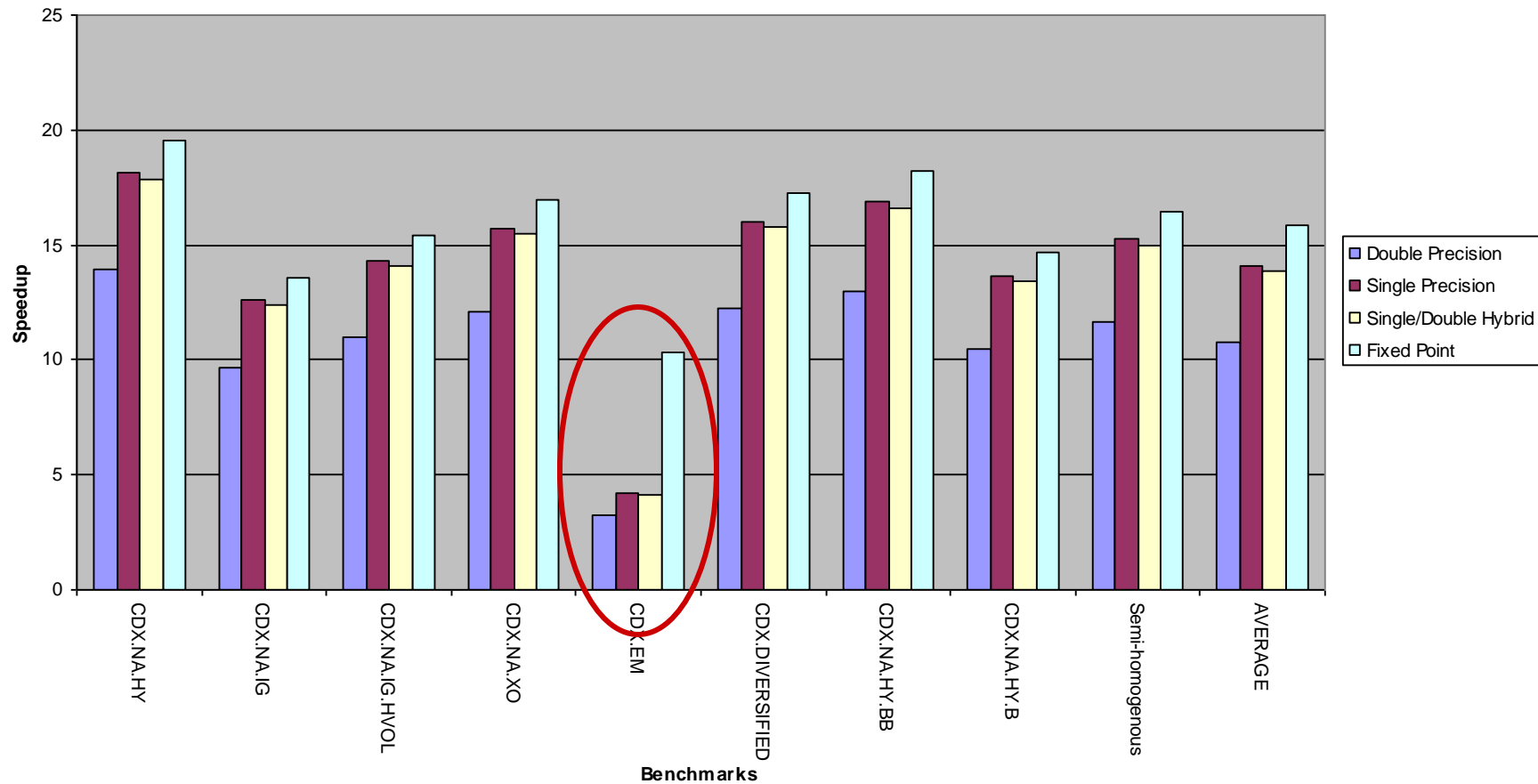
- Credit rating and number of instruments are based on Dow Jones CDX
- Notionals obtained from Moody's, range from \$600,000 to \$6.6 billion
  - $\alpha$ : uniformly distributed in  $[0, 1]$
  - Recovery rate: Normally distributed,  $N(0.4, 0.15)$
  - # of Time Steps: Normally distributed,  $N(20, 10)$

# Processor vs. FPGA setup



- 3.4 GHz Intel Xeon Processor
- 3GB RAM
- C++ program
- 100,000 Monte-Carlo paths
- Virtex 5 SX50T speed grade -3
- Connected to host through PCI express
- 100,000 Monte-Carlo paths

# Performance: Single Core Results (1/2)



# Performance: Single Core Results (2/2)

Single Core Average Acceleration:

Double Precision: 10.6 X

Single Precision: 13.9 X

Single/Double Hybrid: 13.6 X

Fixed Point: 15.6 X

# Performance: Multi-Core

- Monte-Carlo paths independence allows for a linear speedup as more pricing cores are incorporated.

	Double	Single	Single/Double Hybrid	Fixed - Point
Single Core Acceleration	10.6X	13.9X	13.6X	15.6X
Maximum # of Instantiations	2	4	4	5
Multi-Core Acceleration	15.7X	46.5X	46.8X	63.5X

# Summary

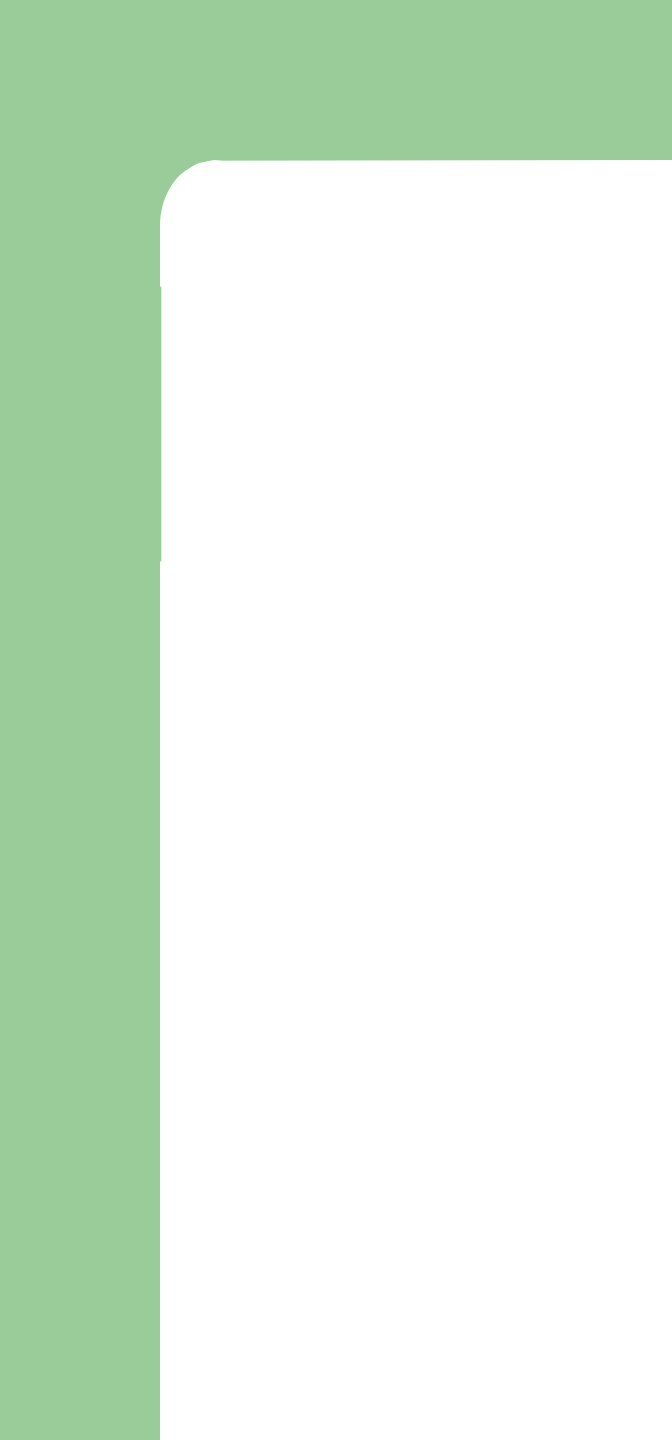
- Presented a hardware architecture for pricing Collateralized Debt Obligations using Li's model
- Demonstrated the advantages of using DSP48Es in terms of resource utilization and frequency
  - Especially evident for single precision
- Established that either a single/double hybrid or fixed-point representations could be used to balance resource utilization and accuracy
- Fixed-point hardware design is over 63-fold faster than a corresponding software implementation

# Future Work

1. Expand to Multi-Factor model

$$Y_i = \sum_{j=1}^m (a_{ij} X_{ij}) + \beta_i Z_i$$

2. Attempt the algorithm on a different accelerator architecture
  - GPU



Thank You  
(Questions?)