FPGA BASED ACCELERATION OF COMPUTE-INTENSIVE WORKLOADS IN FINANCE

AGENDA

- Trends
- FPGA architecture
- High level design flows
- Finance Library for FPGA
WHERE DO FPGAS FIT IN?

FPGAs enable solving system level data movement issues

Control Plane
- Management
- Security
- Protocol bridging

FPGA / GPU

CPU

FPGA

Storage

Data Flow Processing
- Inline processing
- Pre-processing
- Pre-filtering
- Exception processing
- Cryptography
- Compression
- IO expansion
- Protocol bridging

Storage Acceleration
- Cryptography
- Compression
- Indexing
WHERE FPGAS ADD VALUE

• Networking
  • Low Latency for High Frequency Trading
  • Offload Server or Trading Rules Compliance

• Algorithms
  • Valuation, risk and machine learning
  • Big data collection, curation and analysis
  • Financial Portfolio Risk Management
    • Pre-emptive risk, collateral, liquidity calculations
    • Large financial companies with their own equity/bond/currency trading desks
    • Real-time or near real-time risk and scenario analysis

• Three key financial deployment environments
  1. “Colocation” next to the exchange for high frequency trading
  2. Traditional data center
  3. Quant/data scientist desk-top
COLLOCATED IN EXCHANGES

High Frequency Trading
- Server is collocated with the exchange
- Datapath and/or algo acceleration
- Nano secs. matter!

Nasdaq, CME, ICE, NYSE, TSE, ARCA....

Compliance & risk

Hosted deployment model

Exchange Data Feeds

Feed backbone (multicast)

Order Routing Network

Services Core

Trader Network

Feed handlers

Analytics & Algorithmic Engines

Execution Engines
HIGH FREQUENCY TRADING ACCELERATOR

Round Trip latency with QPI: ~500ns
Round Trip latency with PCIe: ~1000ns
Remote from Exchange

Exchange Data Feeds

Feed backbone (multicast)

Feed handlers

Enable Direct Market Access

Order Routing Network

Services Core

Trader Network

Nasdaq, CME, ICE, NYSE, TSE, ARCA....

Analytics & Algorithmic Engines

Execution Engines

Back-test algos on historical data

Portfolio risk management & Rules Checking

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

For more complete information about compiler optimizations, see our Optimization Notice.
FINANCIAL DATA-STREAMING

OPRA Protocol

• Allows protocols to be described in OpenCL
  • OPRA protocol used for options trading
  • CPU Offload or Low Latency or Compliance
  • Embed a kernel in the middle of an FPGA

<table>
<thead>
<tr>
<th>Demo</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>10Gbit / sec</td>
</tr>
<tr>
<td>Latency</td>
<td>300ns - 1 μs</td>
</tr>
<tr>
<td>Demo</td>
<td>SC12</td>
</tr>
</tbody>
</table>
FINANCIAL NETWORKING ACCELERATION

- NASDAQ
  - CPU Offload
- SolarFlare
  - Low latency 10G
  - CPU Offload

Solarflare Application Onload Engine
FPGA ARCHITECTURE: FINE-GRAINED MASSIVELY PARALLEL

- Millions of reconfigurable logic elements
- Thousands of 20Kb memory blocks
- Thousands of Variable Precision DSP blocks
- Dozens of High-speed transceivers
- Multiple High Speed configurable Memory Controllers
- Multiple ARM© Cores

Let’s zoom in
FPGA ARCHITECTURE: BASIC ELEMENTS

Basic Element

1-bit configurable operation

1-bit register (store result)

Configured to perform any 1-bit operation:
AND, OR, NOT, ADD, SUB
**FPGA ARCHITECTURE: MEMORY BLOCKS**

Can be configured and grouped using the interconnect to create various cache architectures.
FPGA ARCHITECTURE: FLOATING POINT MULT/ADD

Dedicated floating point multiply and add blocks
Blocks are connected into a custom data-path that matches your application.
FPGAS AND SOCS

2X Core Performance Enabled by 14 nm Tri-Gate + HyperFlex Architecture
30 Gbps Chip-to-Chip Transceivers, Path to 56 Gbps Chip-to-Chip
  Up to 144 Transceiver Channels
  M20K Embedded Memory Blocks
  Up to 10 TFLOPS DSP Performance
Floating-Point Optimized Variable-Precision DSP Blocks
  DDR4 Support up to 2666 Mbps
  PCI Express® (PCIe®) Gen 1, Gen 2, Gen 3, Gen 4*
Lowest Power for Bandwidth-Intensive Applications
  Up to 53 Mbits Block RAM
  Up to 768 IO
  Up to 53 Mbits Block RAM
  Up to 624 IO
  Up to 17.4 Gbps XCVRs
  Up to 660K LEs
  Up to 42 Mbits Block RAM
  Up to 624 IO
HIGH LEVEL TOOL FLOWS
THE SOFTWARE PROGRAMMER’S VIEW

• Programmers develop in mature software environments
  • Ideas can easily be expressed in languages such as ‘C’
    o Typically start with simple sequential program
    o Use parallel APIs / language extensions to exploit multi core for additional performance
  • Compilation times are almost instantaneous
  • Immediate feedback
  • Rich debugging tools
Design Productivity with HLD Tools

Traditional RTL Design Methodology

<table>
<thead>
<tr>
<th>Design Creation</th>
<th>Functional Verification</th>
<th>RTL Synthesis</th>
<th>Place &amp; Route</th>
<th>Gate Level Verification</th>
</tr>
</thead>
</table>

HLS Design Methodology

<table>
<thead>
<tr>
<th>Design Creation</th>
<th>Functional Verification</th>
<th>RTL Synthesis</th>
<th>Place Route</th>
<th>Gate Level Verification</th>
</tr>
</thead>
</table>

RTL vs Untimed C++ Functional Verification Times

<table>
<thead>
<tr>
<th>Design</th>
<th>RTL Sim Time</th>
<th>C Sim Time</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES Encryption</td>
<td>22 mins</td>
<td>46 ms</td>
<td>29,000x</td>
</tr>
<tr>
<td>Huffman Encoding</td>
<td>13 mins</td>
<td>52 ms</td>
<td>15,000x</td>
</tr>
<tr>
<td>Optical Flow</td>
<td>~2 Days</td>
<td>10 seconds</td>
<td>12,000x</td>
</tr>
<tr>
<td>Complex FIR Filter</td>
<td>4.5 min</td>
<td>63 ms</td>
<td>4,200x</td>
</tr>
<tr>
<td>Packet Processing NAT</td>
<td>2.5 min</td>
<td>54 ms</td>
<td>2,700x</td>
</tr>
</tbody>
</table>

10x Faster Functional iteration Cycle
DIFFERENT SOLUTIONS FOR DIFFERENT USERS

For Different Users

“Algorithm” Designer

“Software” Designer

“Embedded” Designer

“Hardware” Designer

Delivery of Different Front-Ends

DSP Builder (Model)

Intel® FPGA SDK For OpenCL (SW)

Intel® HLS Compiler (IP)

HDL Code, Qsys (Schematic)

Key Technology

High Level Design Compiler

HDL

HDL

HDL

HDL

HDL
ACCELERATING HLD TOOL IMPROVEMENTS

Intel® HLS Compiler

C/C++ Front-End

Accelerated Improvement of Quality of Results

Rapid enablement of Intel devices including Stratix 10, Arria 10 and Stratix 10 ARM based SoCs

Compiler Infrastructure

Compiler Optimizations

Front End Tools and Reporting

DSP Builder Front-End

OpenCL Front-End

Actionable feedback and power user control

Accelerated support for advanced features of our products, e.g. Floating-Point

Platform
**OpenCL Use Model**

### Host Code

```c
main() {
    read_data(...);
    manipulate(...);
    clEnqueueWriteBuffer(...);
    clEnqueueNDRange(..., sum, ...);
    clEnqueueReadBuffer(...);
    display_result(...);
}
```

### OpenCL Accelerator Code

```c
__kernel void sum
(__global float *a,
 __global float *b,
 __global float *y)
{
    int gid = get_global_id(0);
    y[gid] = a[gid] + b[gid];
}
```
**OpenCL Tool Flow**

1. Generate intermediate `.aoco` files
2. `aoc -c <kernel filename>.cl`
3. Syntax, resource estimation and optimization report okay?
   - Yes
   - No
4. `aoc --march=emulator <kernel filename>.cl`
5. Functionally Correct?
   - Yes
   - No
6. `aoc -profile <kernel filename>.cl`
7. Performance acceptable?
Library Approach

Intel Provided Building Blocks

- Intel® Xeon® processor library elements
- FPGA intellectual property (IP)
- Intel® Xeon® processor calls FPGA IP

Easier to Use and High Performance / Watt
**BLACK SCHOLES OPTIONS PRICING**

Price 100K to 1M options portfolio
- 8 Black Scholes Engines, 4 DDR IV interfaces
- 5 Inputs, 1 output
- 3.2 Billion option/sec

- Adding Greeks
  - One additional engine per DDR IV
  - 32% increase in resources

Fin-Lib phase 1
- Demo available in Q4 2016

### Models

<table>
<thead>
<tr>
<th>Models</th>
<th>ALMs</th>
<th>RAMs</th>
<th>DSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Scholes wo Greeks</td>
<td>1%</td>
<td>2%</td>
<td>5%</td>
</tr>
<tr>
<td>Black-Scholes with Greeks</td>
<td>1%</td>
<td>2%</td>
<td>8%</td>
</tr>
<tr>
<td>Bachelier</td>
<td>3%</td>
<td>12%</td>
<td>31%</td>
</tr>
</tbody>
</table>
MACHINE LEARNING INFERENCE

- Topology
- Framework
- SW Library
- Run time libraries
- Operating System / Firmware
- PCIe Hardware

Deep Learning Accelerator
- AlexNet, GoogleNet, Customer-developed
- Intel Caffe
- MKL-DNN
- OpenCL
- OS + BSP
- FPGA + CNN IP
ACCELERATING AND OFF-LOADING CHALLENGING FINANCIAL COMPUTATIONS
OVERVIEW

Libraries at 3 levels

Pre-compiled
Callable from C/C++
Uses MKL API

OpenCL
Callable functions
BSP support

IP Cores
VHDL/Verilog
Nested/hierarchical kernels
Financial Library – Phase 1 FinLib demo delivered

- Phase 1 functions all accelerated and included in the newly created FinLib for FPGA, which covers ~95% of exchange-traded options.
- FinLib demo and code shipped to HPe for Super Computing (SC16) in Salt Lake City – strong feedback.
- Demo from DSP Symposium now uses real exchange data from CME and will be available to customers in Swindon datacentre by end 2016.

<table>
<thead>
<tr>
<th>Model</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black-Scholes</td>
<td>European exercise, pricing &amp; risk</td>
</tr>
<tr>
<td>Black-Scholes-FFT</td>
<td>European exercise – market calibration</td>
</tr>
<tr>
<td>Garman-Kohlhagen</td>
<td>European exercise – foreign currency</td>
</tr>
<tr>
<td>Curran</td>
<td>European exercise – arithmetic average</td>
</tr>
<tr>
<td>Cox-Ross-Rubenstein</td>
<td>American exercise – spot and futures</td>
</tr>
<tr>
<td>Bjerksund-Stensland</td>
<td>American exercise – very fast approximation</td>
</tr>
<tr>
<td>Merton</td>
<td>European exercise – on dividend paying stocks</td>
</tr>
<tr>
<td>Kirk</td>
<td>European exercise – lognormal spread</td>
</tr>
<tr>
<td>Bachelier</td>
<td>European exercise – normal spread</td>
</tr>
</tbody>
</table>
FINANCIAL LIBRARIES PERFORMANCE

• FinLib can execute 3.2 billion option calculations/second using ~40% of an Arria10 1150 GX at 300MHz.

• A10 using 4 DDR4 controllers (memory bound problem) – Nallatech 510T card

• 2 Black-Scholes engines / DDR4 interface
  • Core running @ 400MHz (new DSPBA backend)
  • 4 DDR4 interfaces provide 3.2 billion option/s
    • 40% of an Arria 10 1150 GX (FMAX =300Mhz)

• FinLib also generates 5 risk sensitivities for each option at the same time as the option price (major advantage)
  • 1 Engine / DDR4 interface
  • 32% of the resources (less engines more calculations)

CPUs can’t do this at line rate!
SINGLE FUNCTION EXAMPLE FFT

- **FFTLib demo** - existing FFT functionality wrapped up into a library (FFTLib), integrated and exposed at the MKL level.

- Used same function syntax as existing MKL to simplify ease of use and migration.

- Re-use: FFTLib re-used in Black-Scholes-FFT function - we can nest kernels AND fill up the FPGA to ensure maximum performance
Monte Carlo Simulation of Complex Derivatives

Generalised Monte Carlo path generator
Example option payoff is European style exercise on the average of the spread between two underlyings:

Payoff = Max(CP * (Avg1 - Avg2 - Strike), 0)

/// To find the value of an average rate or Asian option on the spread between two assets:

double MCAvgSpreadOpt(int CP, double S1, double S2, double X, double T, double r, double b1, double b2, double v1, double v2, double rho, long nSteps, long nSims)
Example of a collar trading strategy:
Buy at put at strike price A, sell a call at strike price B:

A strategy for the bullish but nervous, because some part of the expected upside has been given up to ensure potential downside loss is limited - popular strategy after a rise in the price of the underlying when traders want to protect unrealised profits.

```
/// Strategy 1: Collar on exchange traded options
/// Assumes two simple cases:
/// buySell = 1 means buy put sell call (i.e. user supplies put details, function
/// calculates call premium and summary results of the net position.
/// buySell = 2 means buy call sell put (i.e. user supplies call details, function
/// calculates put premium and summary results of the net position

int strat_simple_collar(double *unknownStrike, double *
*netReturn, double *netDelta, double *netGamma, double *
*netVega, int buySell, double fwdPrice, double time,
double Vol, double R, double suppliedStrike, double suppliedPremium)
```
Phase 1 exchange traded options models provide the ability for users to model risk for entire portfolios on-chip.

FinLib scales across multiple devices, enabling risk summarisations to be performed on-chip for peak performance or off-chip using required memory type for even greater scalability.
SCALING UP - OPTIMAL SILICON USE AND RE-USE VIA LIBRARIES

Using the Black-Scholes option pricing model from finance as an example, the diagrams below illustrate how the Libraries Project delivers three vital infrastructure building blocks:

1. **Low-level numerical libraries** for mathematics and statistics.
2. **Intermediate algorithms** built from combining the low-level libraries from Intel-PSG and end-users.
3. **Automated topology generation** that balances performance requirements, power consumption and physical layout considerations.

### Libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLLib</td>
<td>CNN, RNN, SVM, Random Forests, Boosted Trees etc</td>
</tr>
<tr>
<td>FinLib</td>
<td>Pricing, valuation and risk for low and high frequency trading</td>
</tr>
<tr>
<td>LALib</td>
<td>SGEMM/BLAS type functions; tensor methods</td>
</tr>
<tr>
<td>FFTLib</td>
<td>Forward, backward, super-sampling etc</td>
</tr>
<tr>
<td>MALib</td>
<td>Core math functions: Sin, Cos, Tan, Log, Exp etc Core stats functions: univariate and multivariate stats</td>
</tr>
</tbody>
</table>

For Black-Scholes, a statistics library function from MALib is used to compute the cumulative normal which forms a key part of the OpenCL implementation of Black-Scholes.

For the FFT option pricer the new FFTLib has been used to integrate the characteristic function that is needed to generate option values.

Libraries are combined using OpenCL to write the Black-Scholes algorithm which calls library functions. Automated topology generation enables optimal use of chip resources – BRAMs and logic. The finance example shows how topologies can be used scale-up and fill the chip to accommodate different models on-chip at the same time, or several copies of the same model on-chip at the same time.
CODE THAT PERFORMS AND OUTPERFORMS

Download a free, 30-day trial of Intel® Parallel Studio XE 2018 today


AND DON’T FORGET...

To check your inbox for the evaluation survey which will be emailed after this presentation.

P.S.

Everyone who fills out the survey will receive a personalized certificate indicating completion of the training!
LEGAL DISCLAIMER & OPTIMIZATION NOTICE

• INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

• Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

• Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804