



FPGA BASED ACCELERATION OF COMPUTE-INTENSIVE WORKLOADS IN FINANCE

AGENDA



Trends

FPGA architecture

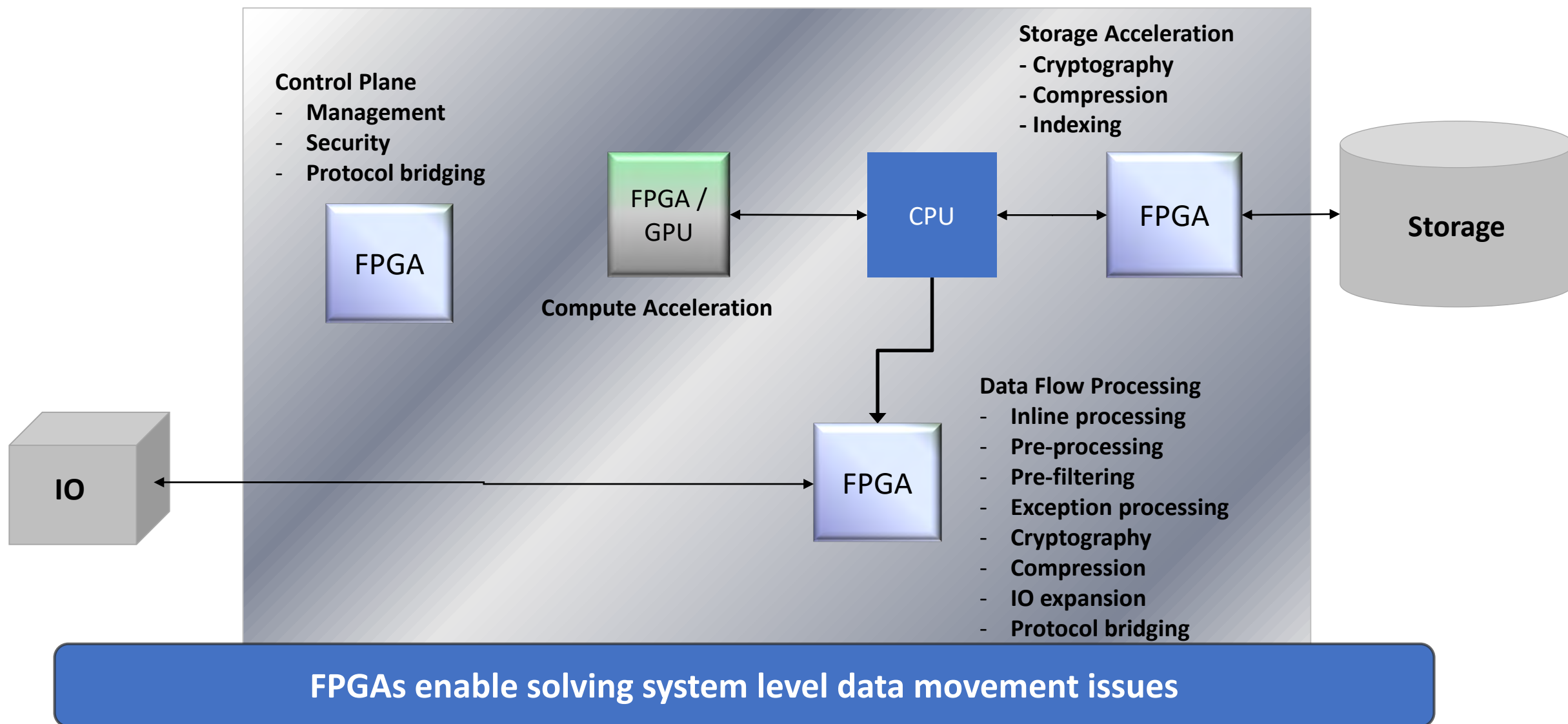
High level design flows

Finance Library for FPGA

WHERE INTEL-FPGAS ADD VALUE



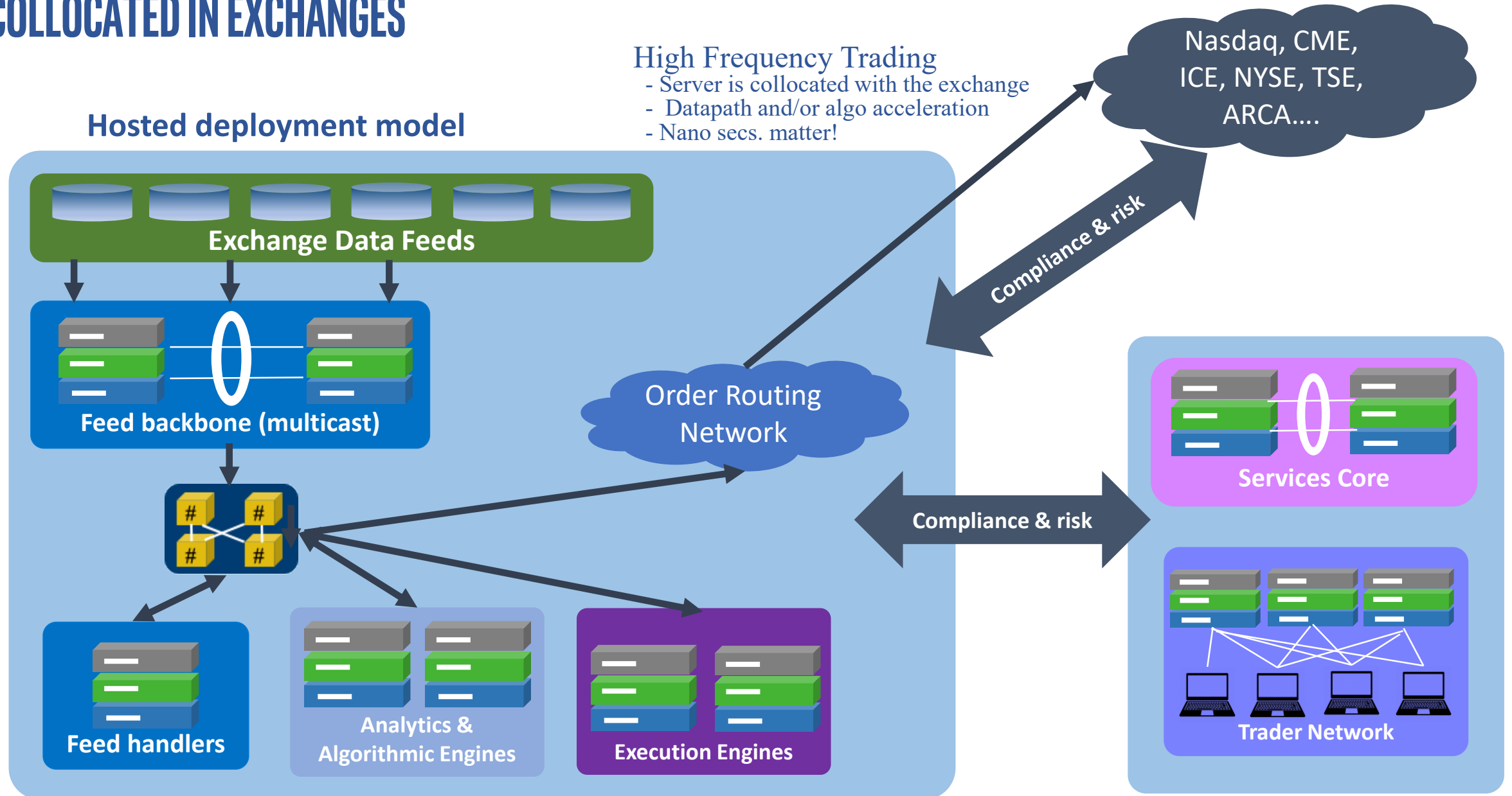
WHERE DO FPGAS FIT IN?



WHERE FPGAS ADD VALUE

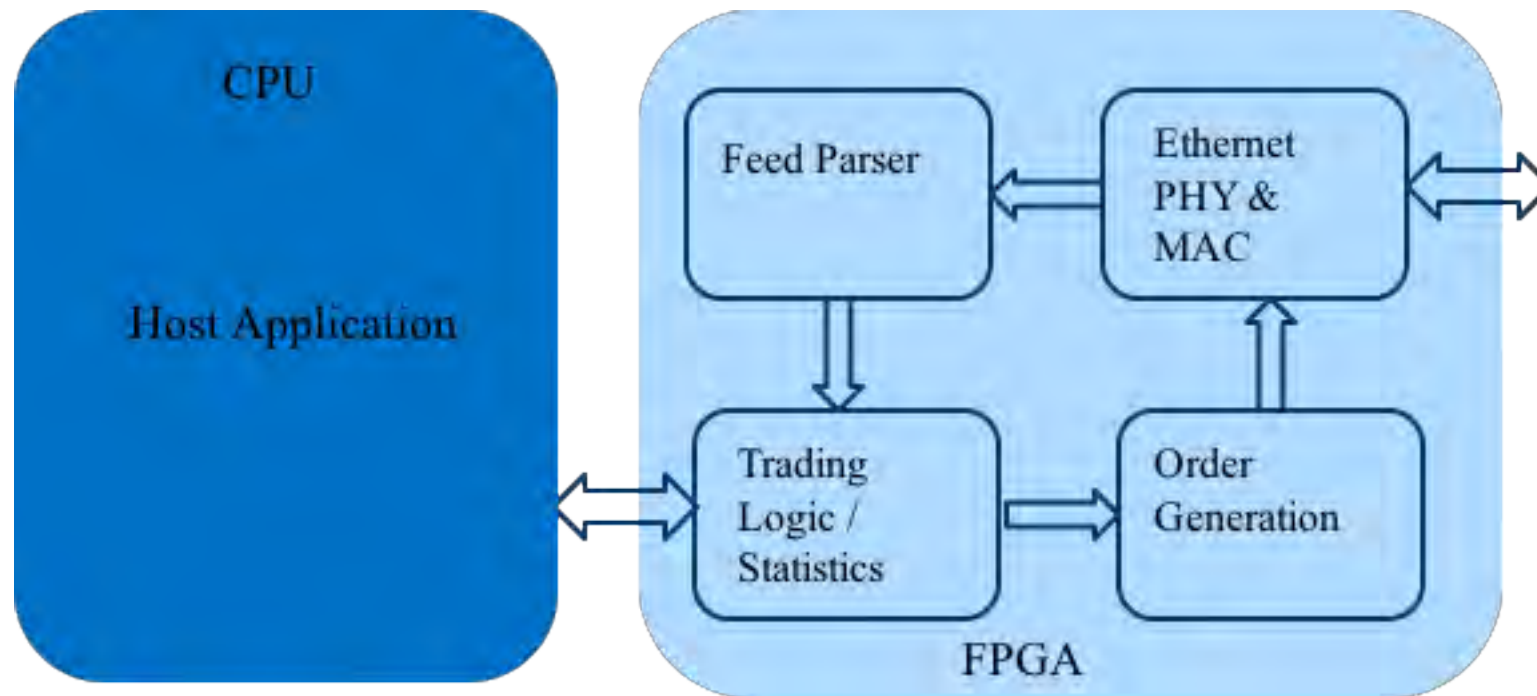
- Networking
 - Low Latency for High Frequency Trading
 - Offload Server or Trading Rules Compliance
- Algorithms
 - Valuation, risk and machine learning
 - Big data collection, curation and analysis
 - Financial Portfolio Risk Management
 - Pre-emptive risk, collateral, liquidity calculations
 - Large 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

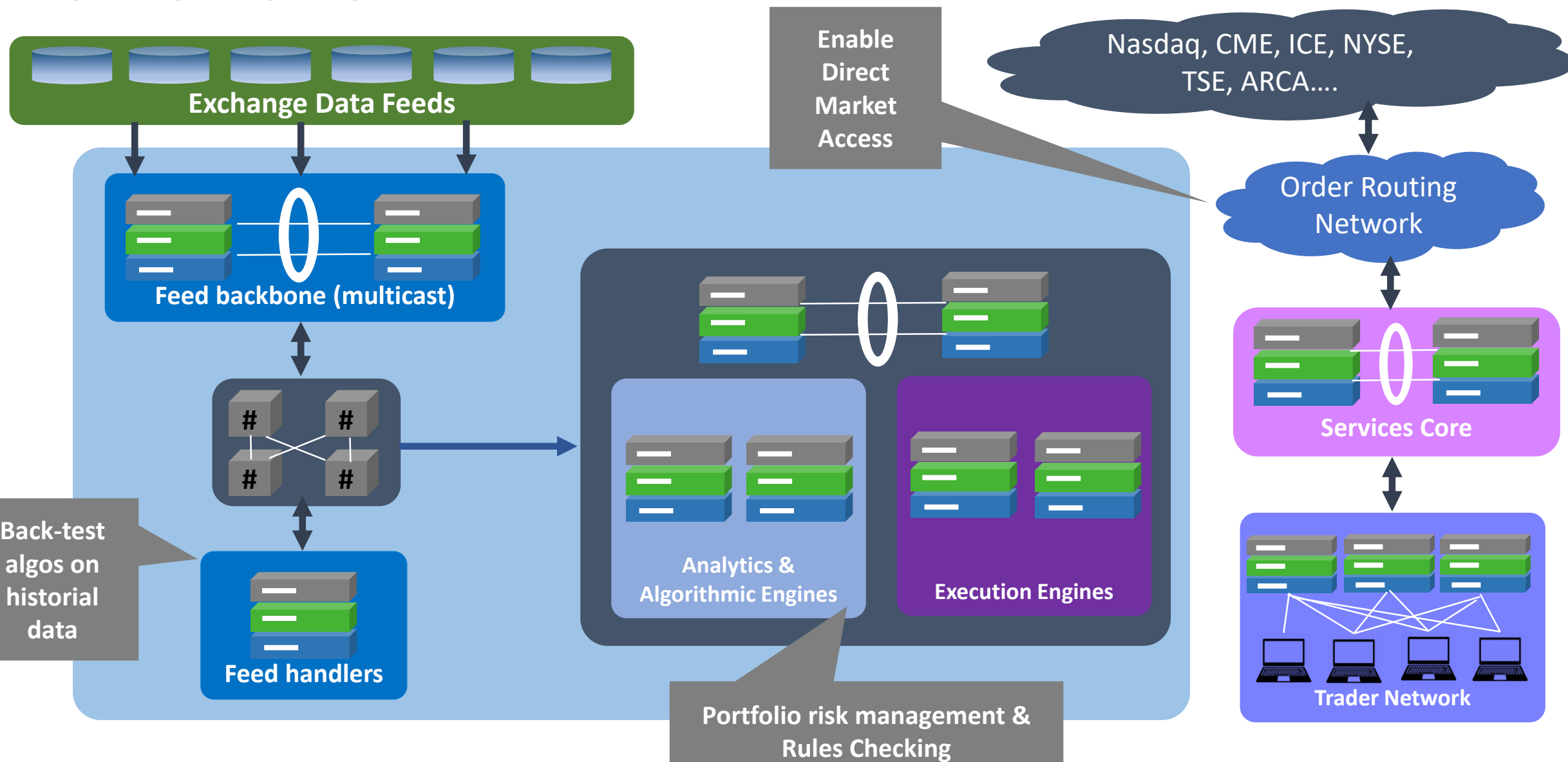


HIGH FREQUENCY TRADING ACCELERATOR

Round Trip latency with QPI : ~500ns
Round Trip latency with PCIe : ~1000ns



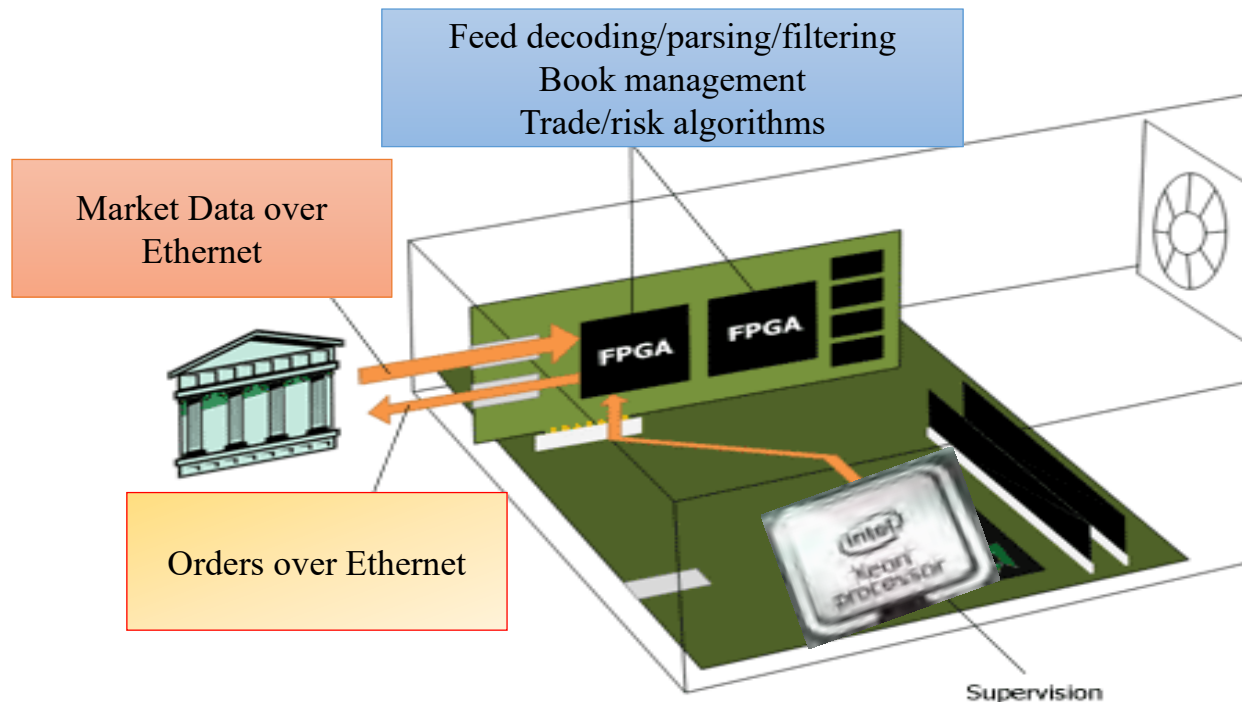
REMOTE FROM EXCHANGE



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



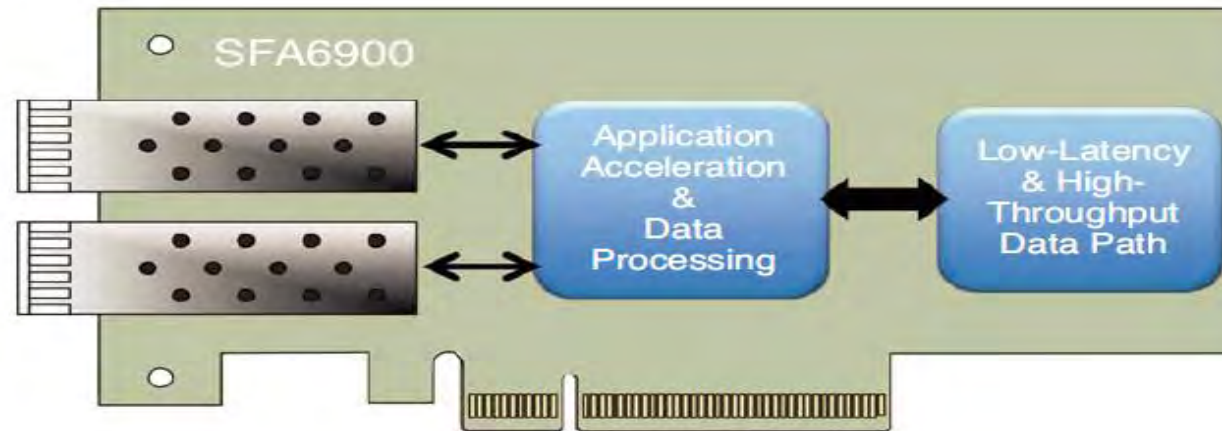
	Demo
Bandwidth	10Gbit / sec
Latency	300ns - 1 μ s
Demo	SC12

FINANCIAL NETWORKING ACCELERATION

What is being announced?

On August 1, 2012, NASDAQ OMX® introduced a new NASDAQ TotalView-ITCH equities depth feed offering using field-programmable gate array (FPGA) technology. This offering provides the same industry-leading performance you expect from the NASDAQ OMX ITCH® feeds with the additional benefit of no message queuing during peak market periods. Because FPGA hardware is more efficient at handling high data rates, it offers consistent and reliable service throughout the trading day, even during market peaks.

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



Solarflare Application Onload Engine

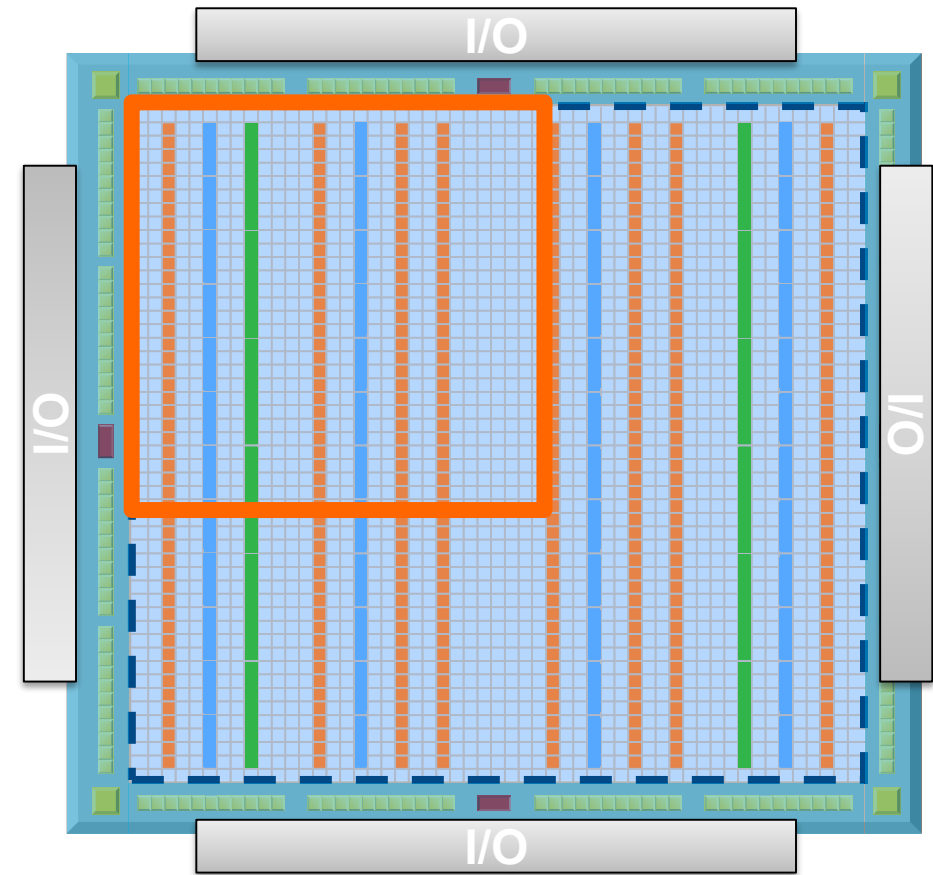
FPGA ARCHITECTURE



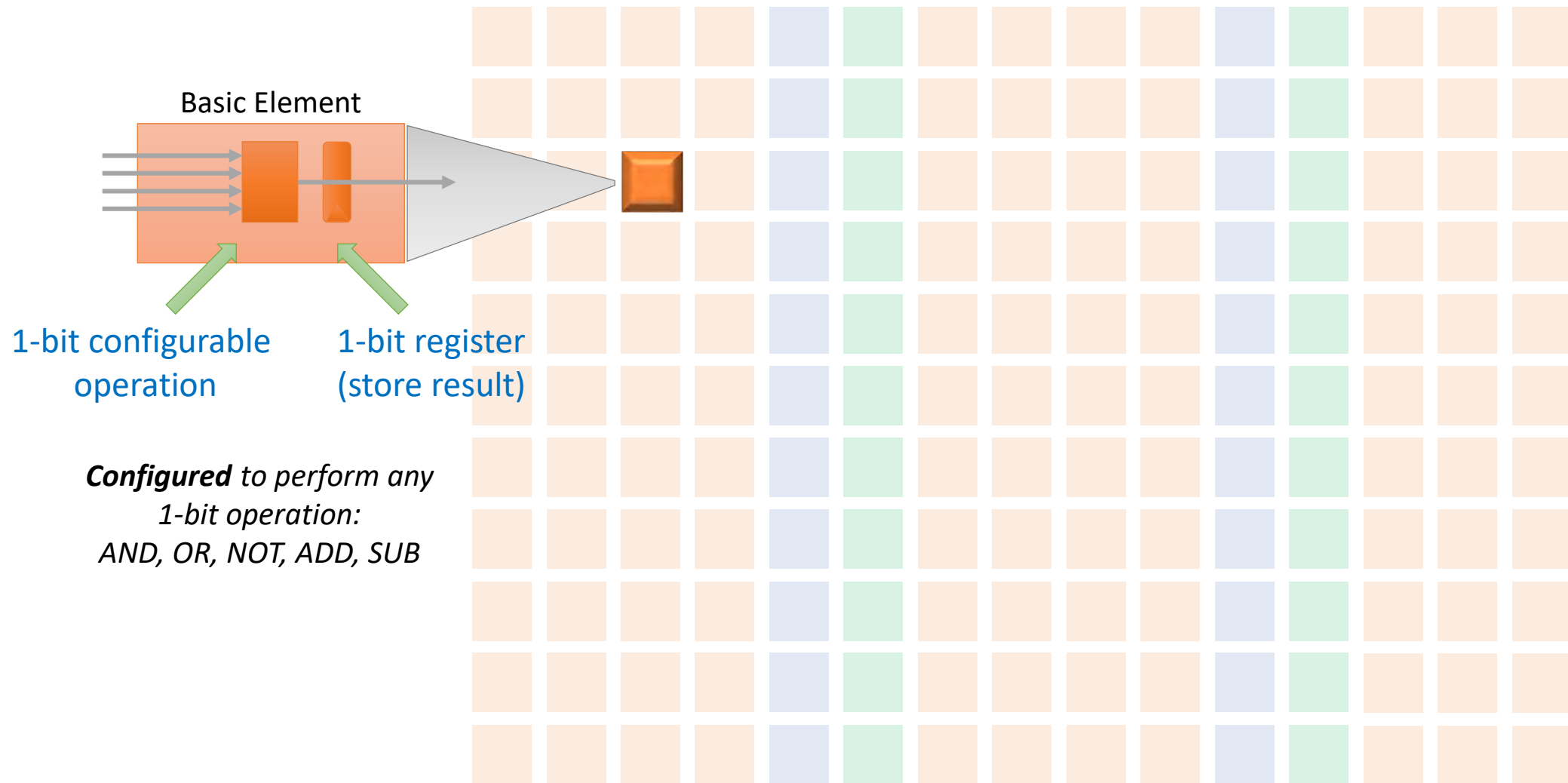
FPGA ARCHITECTURE: FINE-GRAINED MASSIVELY PARALLEL

- 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

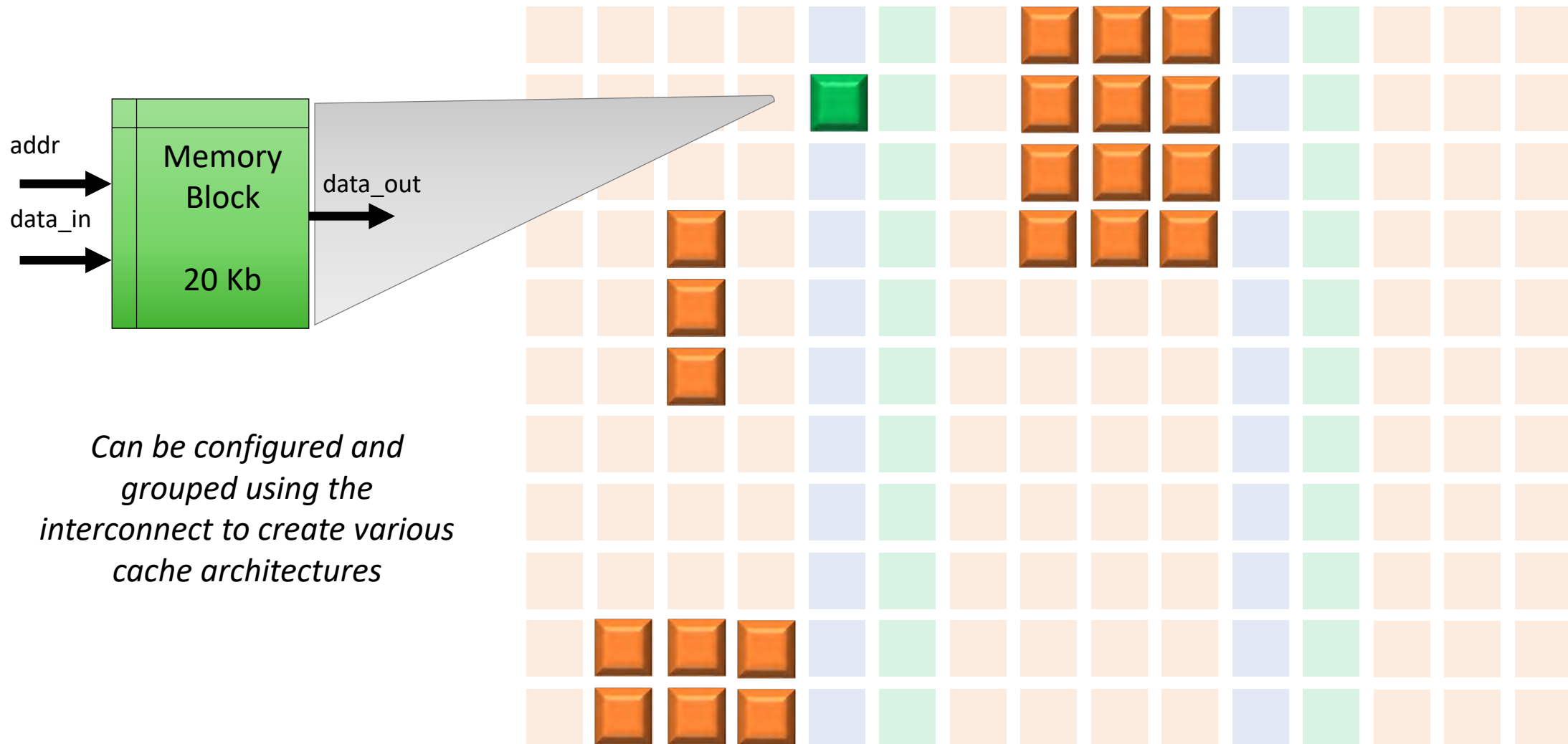
Let's zoom in



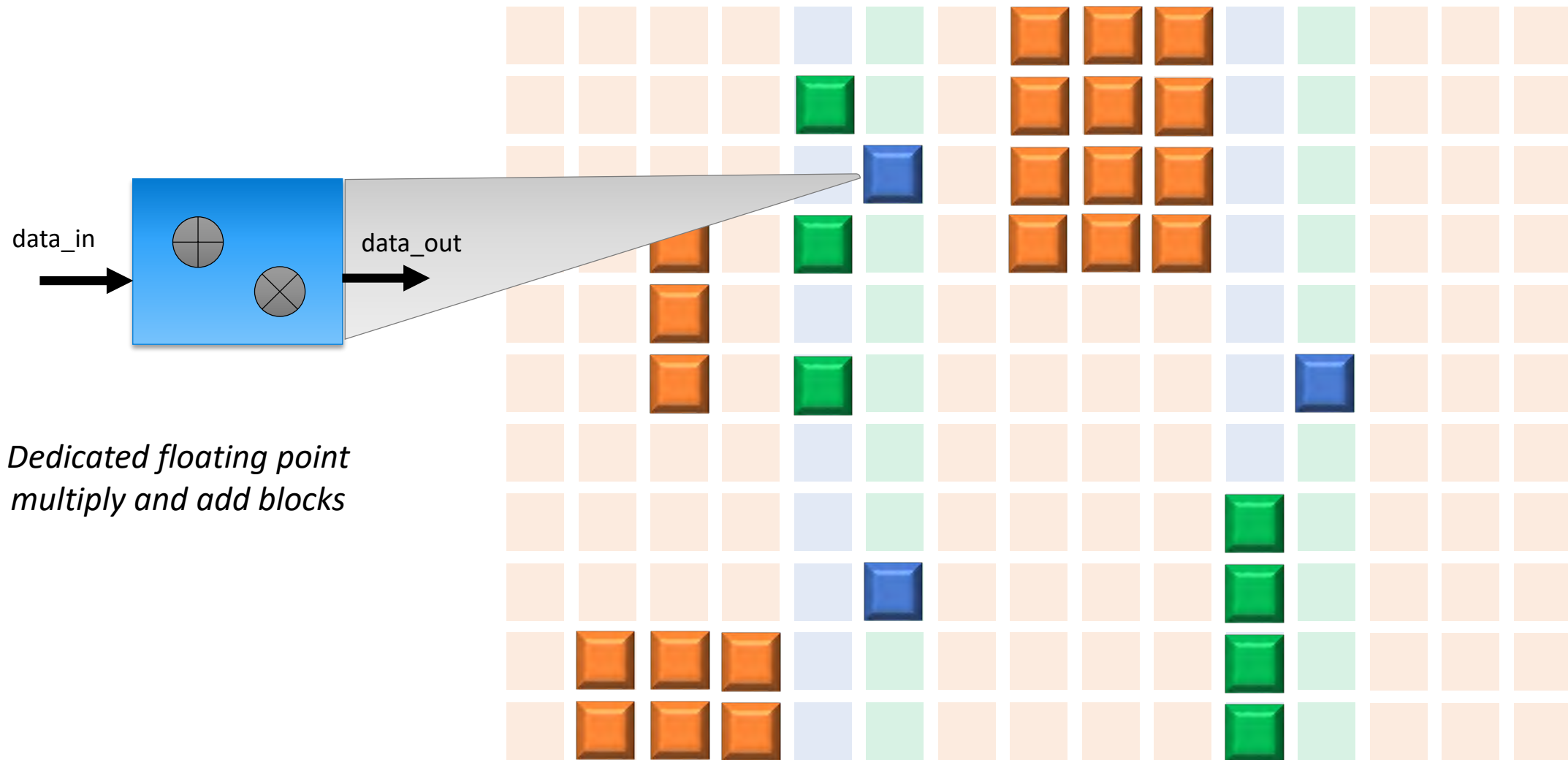
FPGA ARCHITECTURE: BASIC ELEMENTS



FPGA ARCHITECTURE: MEMORY BLOCKS

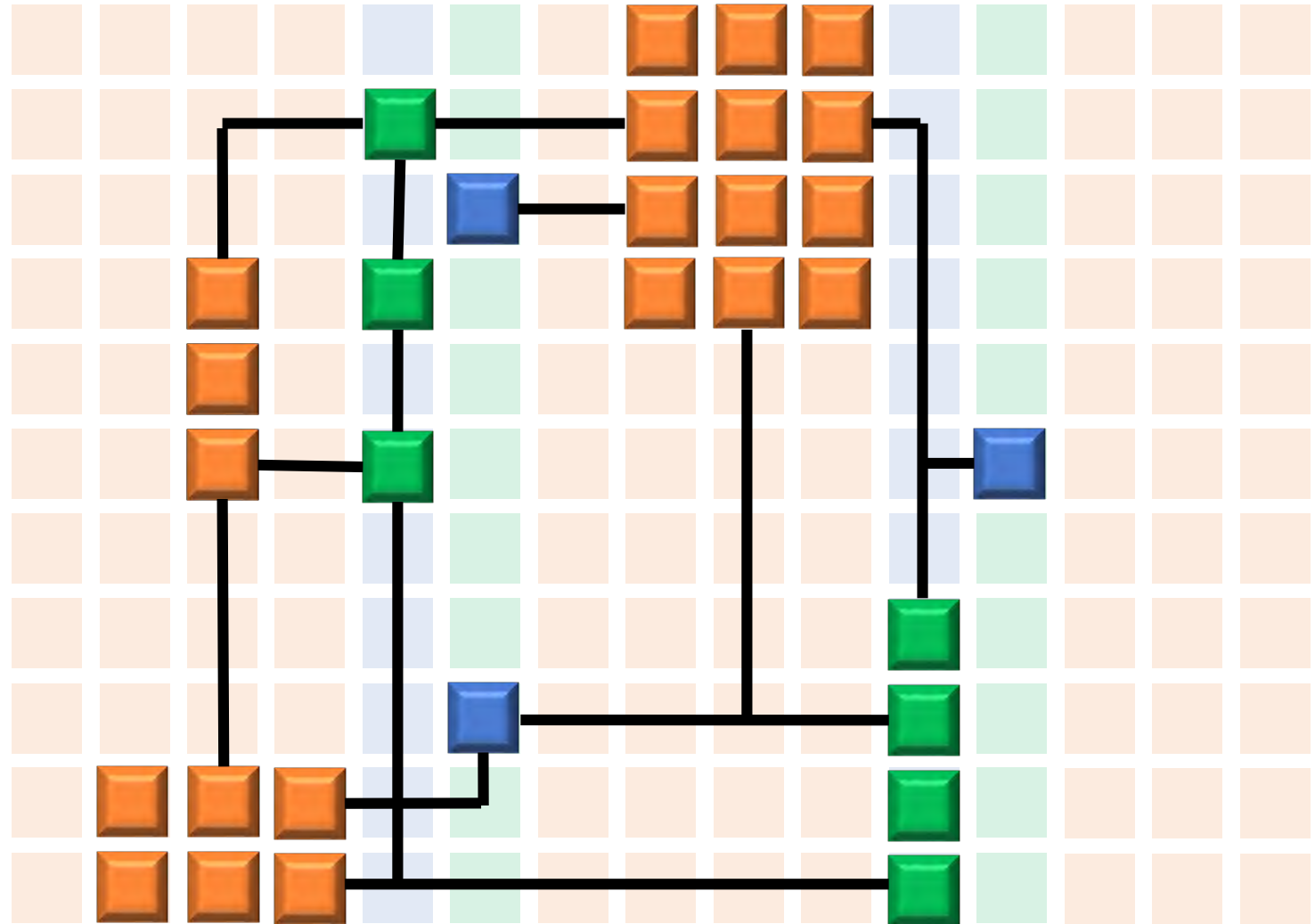


FPGA ARCHITECTURE: FLOATING POINT MULT/ADD

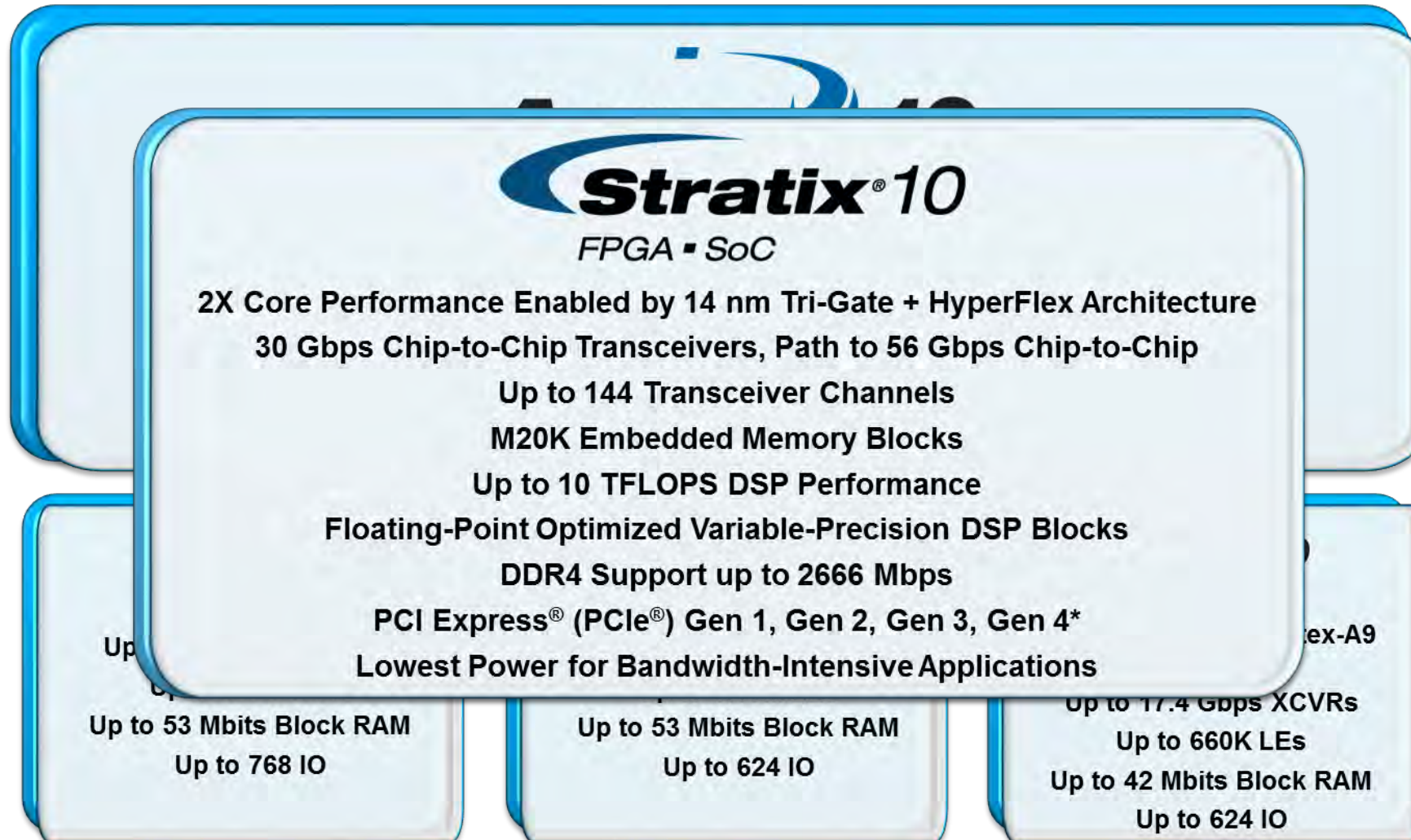


FPGA ARCHITECTURE: CONFIGURABLE ROUTING

Blocks are connected into
a **custom data-path** that matches
your application.



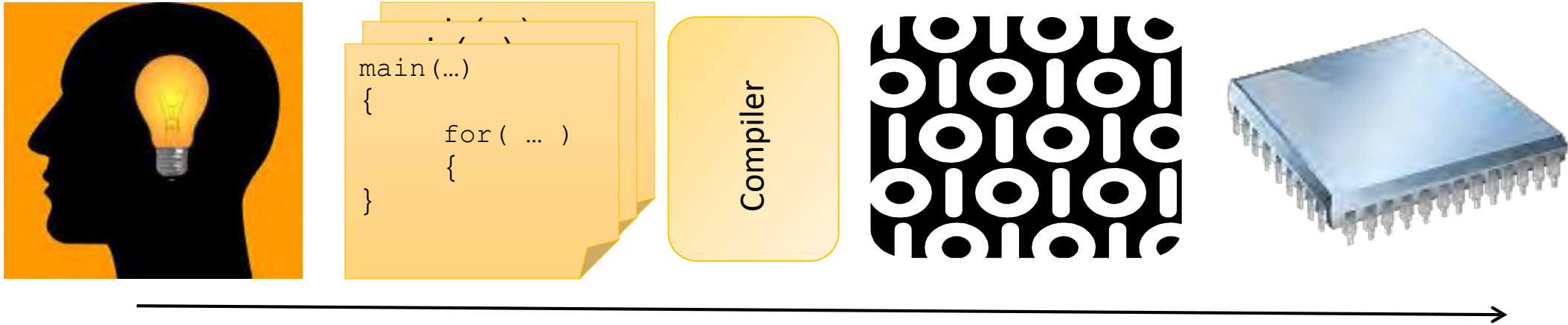
FPGAS AND SOCS



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'
 - Typically start with simple sequential program
 - 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

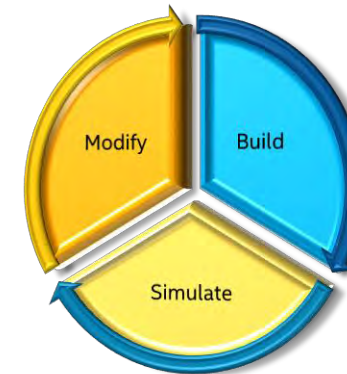
Traditional RTL Design Methodology



HLS Design Methodology



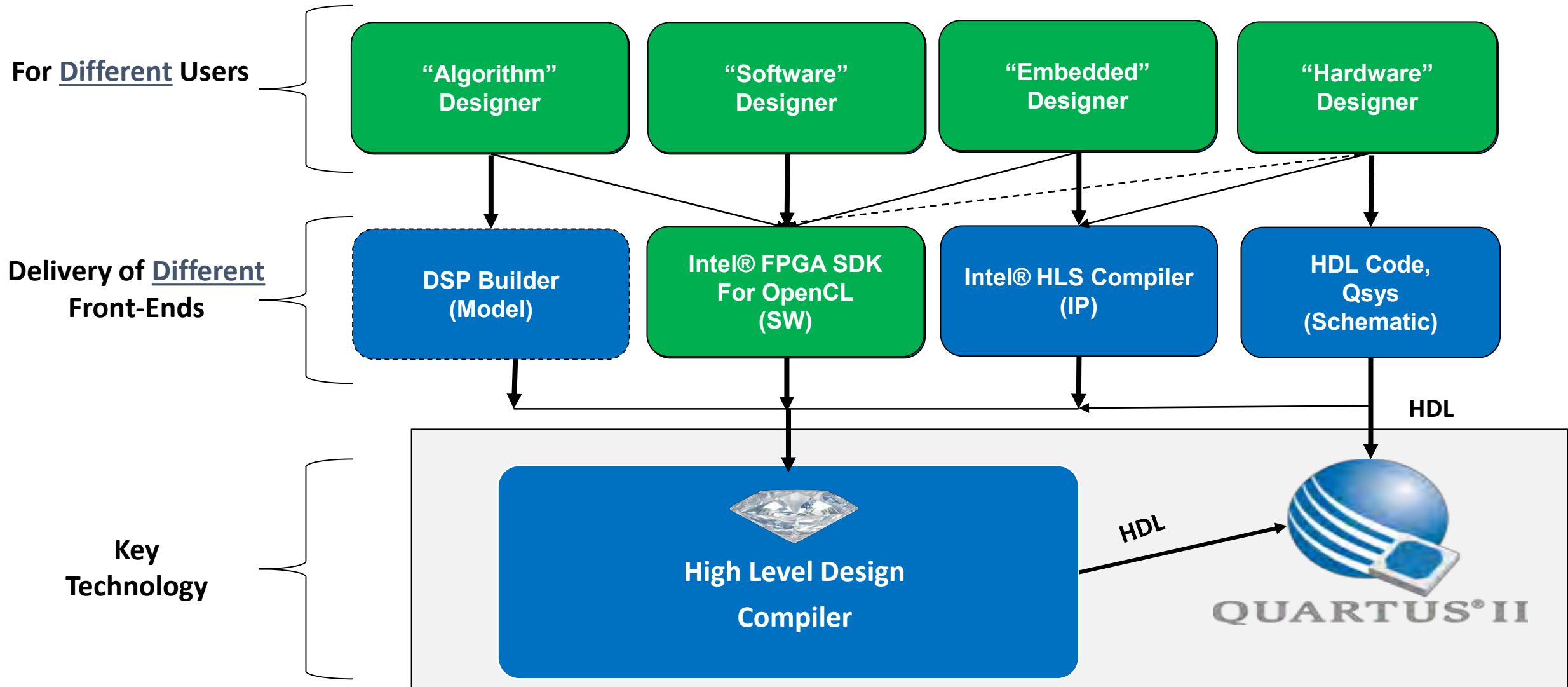
10x Faster Functional iteration Cycle



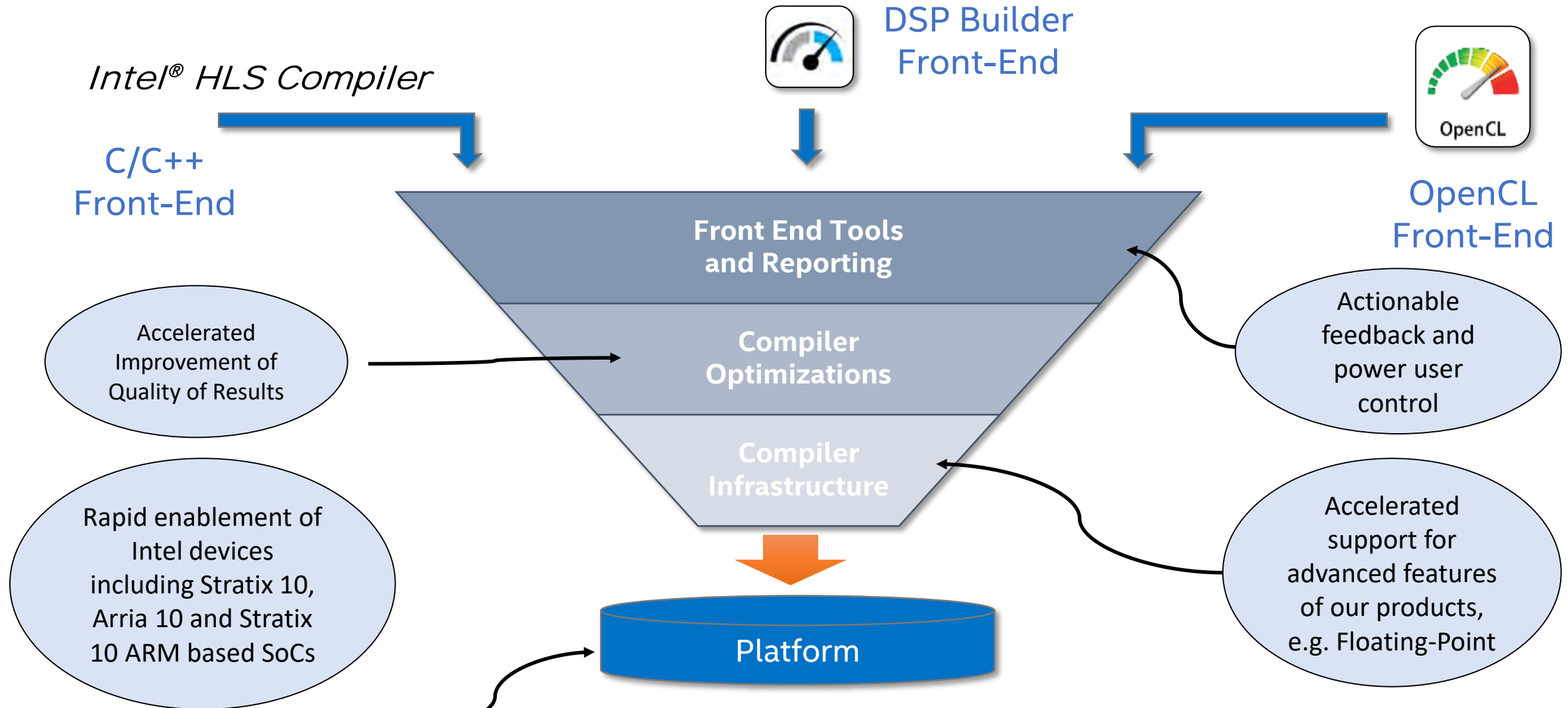
RTL vs Untimed C++ Functional Verification Times

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

DIFFERENT SOLUTIONS FOR DIFFERENT USERS



ACCELERATING HLD TOOL IMPROVEMENTS



OPENCL USE MODEL

Host Code

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

OpenCL Accelerator Code

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

Standard
gcc Compiler

Intel FPGA
Offline
Compiler

Verilog

EXE

AOCX

Quartus® Prime
Design Software

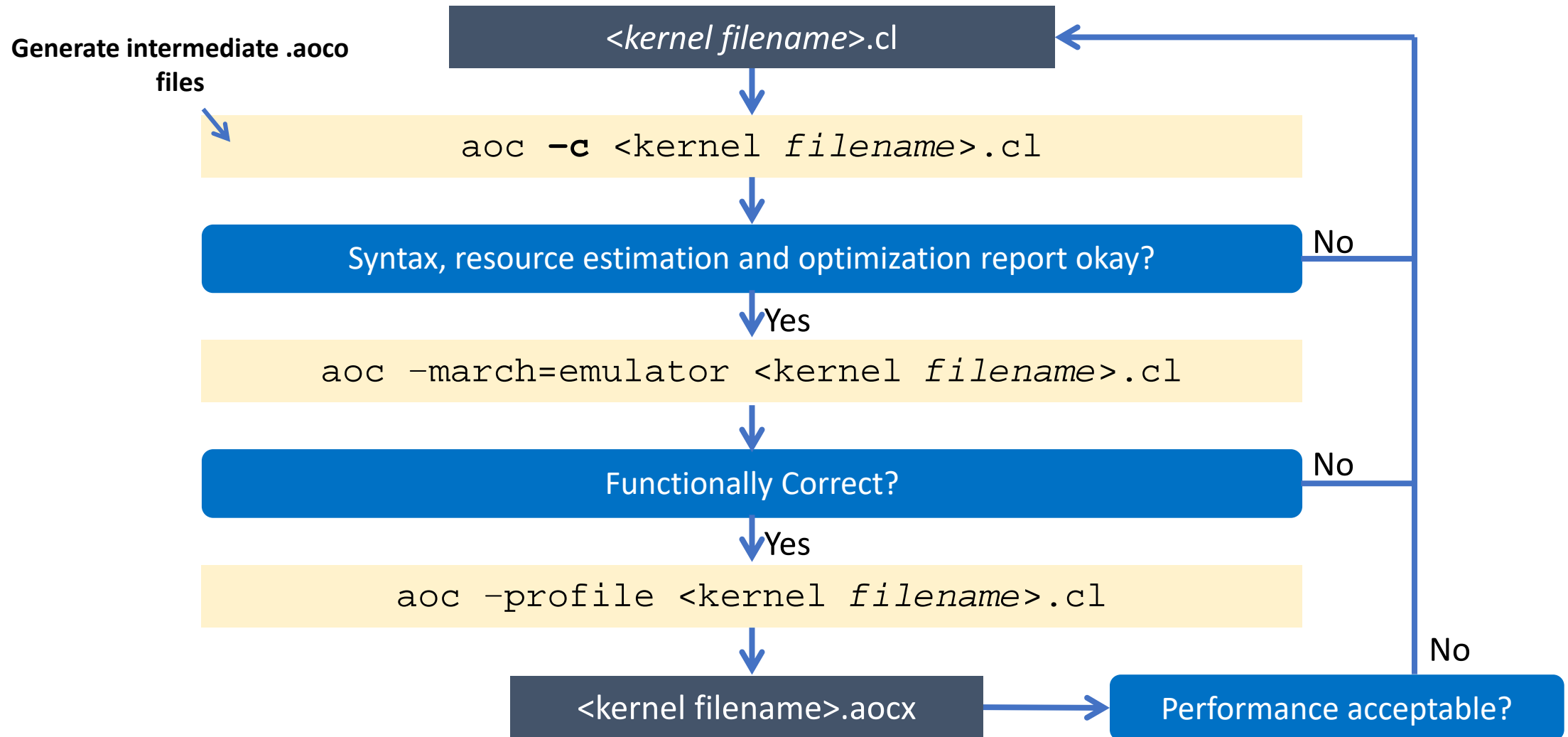


Host



Accelerator

OPENCL TOOL FLOW



COMPILATION REPORT

Applications ▾ Places ▾ Firefox Web Browser ▾ Tue 10:05

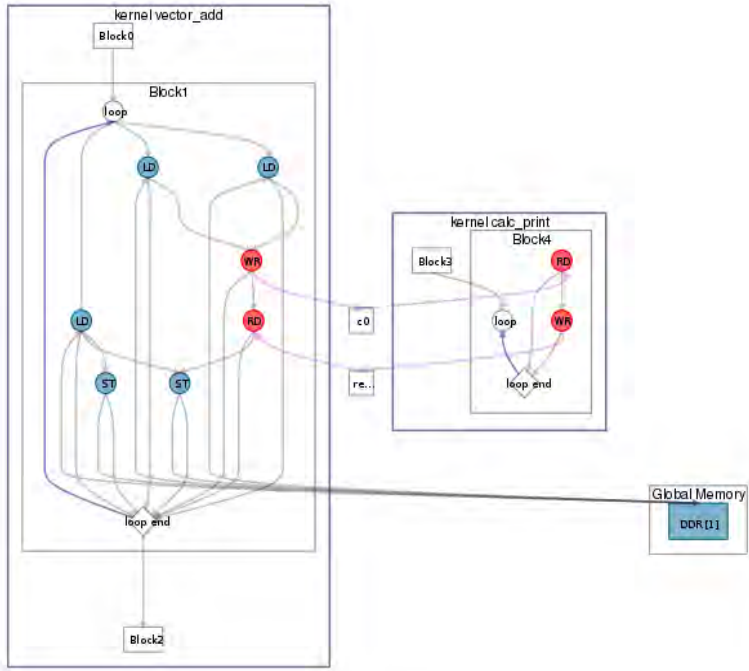
HLD Report: vector_add - Mozilla Firefox

HLD Report: gru78hp... * Vector Add: Intel FPGA®... * HLD Report: vector_a... *

file:///mnt/hgfs/SharedWithVM/projects/vector_add_channel/vector_add_channel/vector_add/reports/report.html#view5

HLD FPGA Reports (Beta) View reports...

System viewer Reset Zoom Clear Selection ☒ Control ☒ Memory ☒ Channels



vector_add.cl

```
19 // This agreement shall be governed in all respects by the laws of the State of California and
20 // by the laws of the United States of America.
21
22 // ACL kernel for adding two input vectors
23
24
25
26 typedef struct {
27     int index;
28     float a;
29     float b;
30 } data2add;
31
32 channel data2add c0;
33 channel float resch;
34
35 __kernel void vector_add(__global const float *x,
36                          __global const float *y,
37                          __global float *restrict z)
38 {
39
40     // get index of the work item
41     //int index = get_global_id(0);
42     data2add infos;
43     float res;
44
45     for(int i=0; i < 10; i++) {
46
47         infos.a = x[i];
48         infos.b = y[i];
49         infos.index = i;
50
51         // add the vector elements
52         //z[index] = x[index] + y[index];
53
54         write_channel_intel(c0, infos);
55         mem_fence(CLK_CHANNEL_MEM_FENCE);
56         res = read_channel_intel(resch);
57         printf(" the result for index = %d is %f\n", infos.index, z[i]);
58         z[i] = res;
```

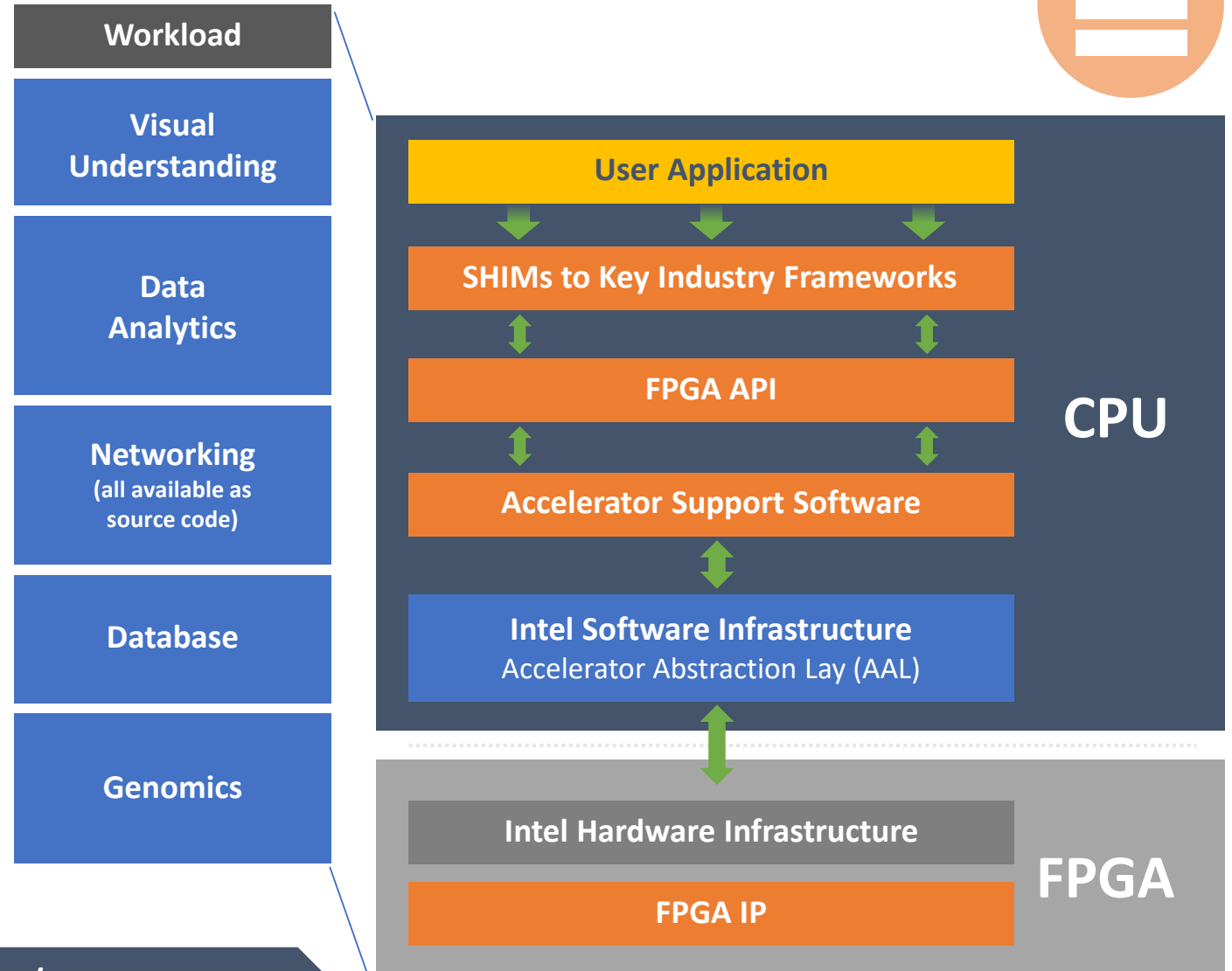

LIBRARIES



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

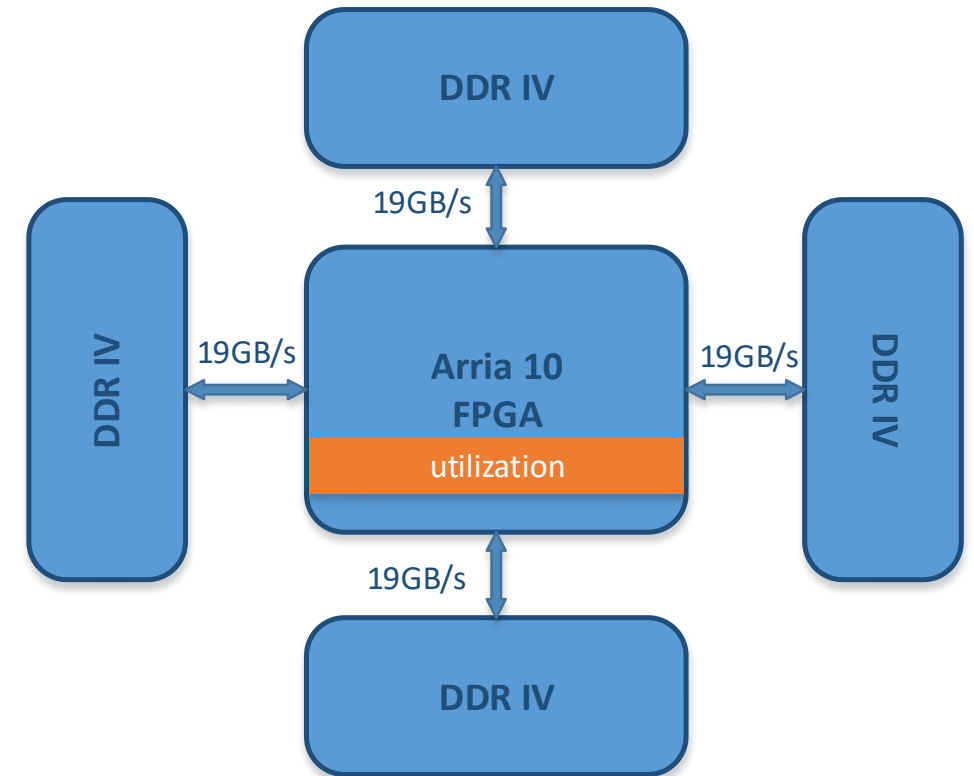
BLACK SCHOLES OPTIONS PRICING

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

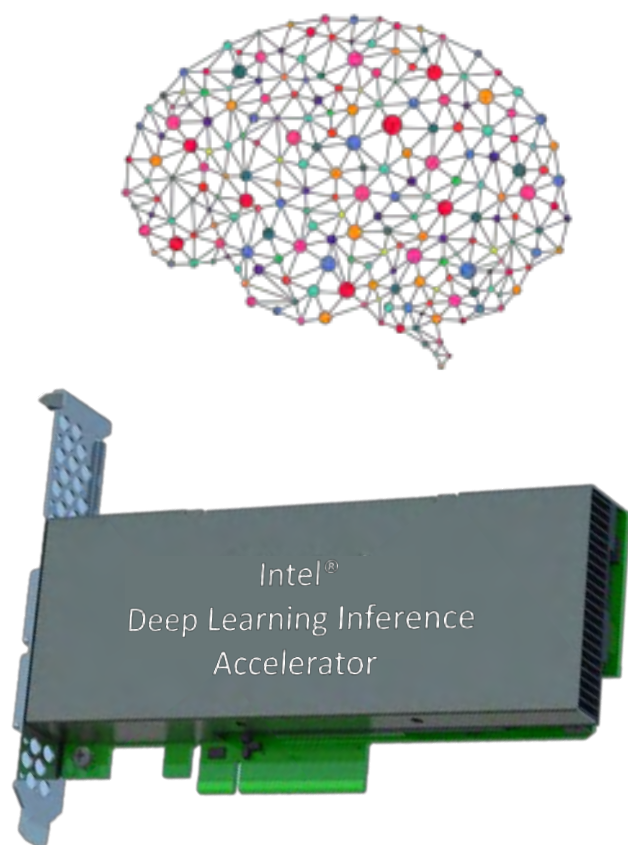
Fin-Lib phase 1

- Demo available in Q4 2016



Models	ALMs	RAMs	DSPs
Black-Scholes wo Greeks	1%	2%	5%
Black-Scholes with Greeks	1%	2%	8%
Bachelier	3%	12%	31%

MACHINE LEARNING INFERENCE



Topology

Framework

SW Library

Run time libraries

Operating System / Firmware

PCIe Hardware

Deep Learning Accelerator

Customer interface

AlexNet, GoogleNet,
Customer-developed

Intel Caffe

MKL-DNN

OpenCL

OS + BSP

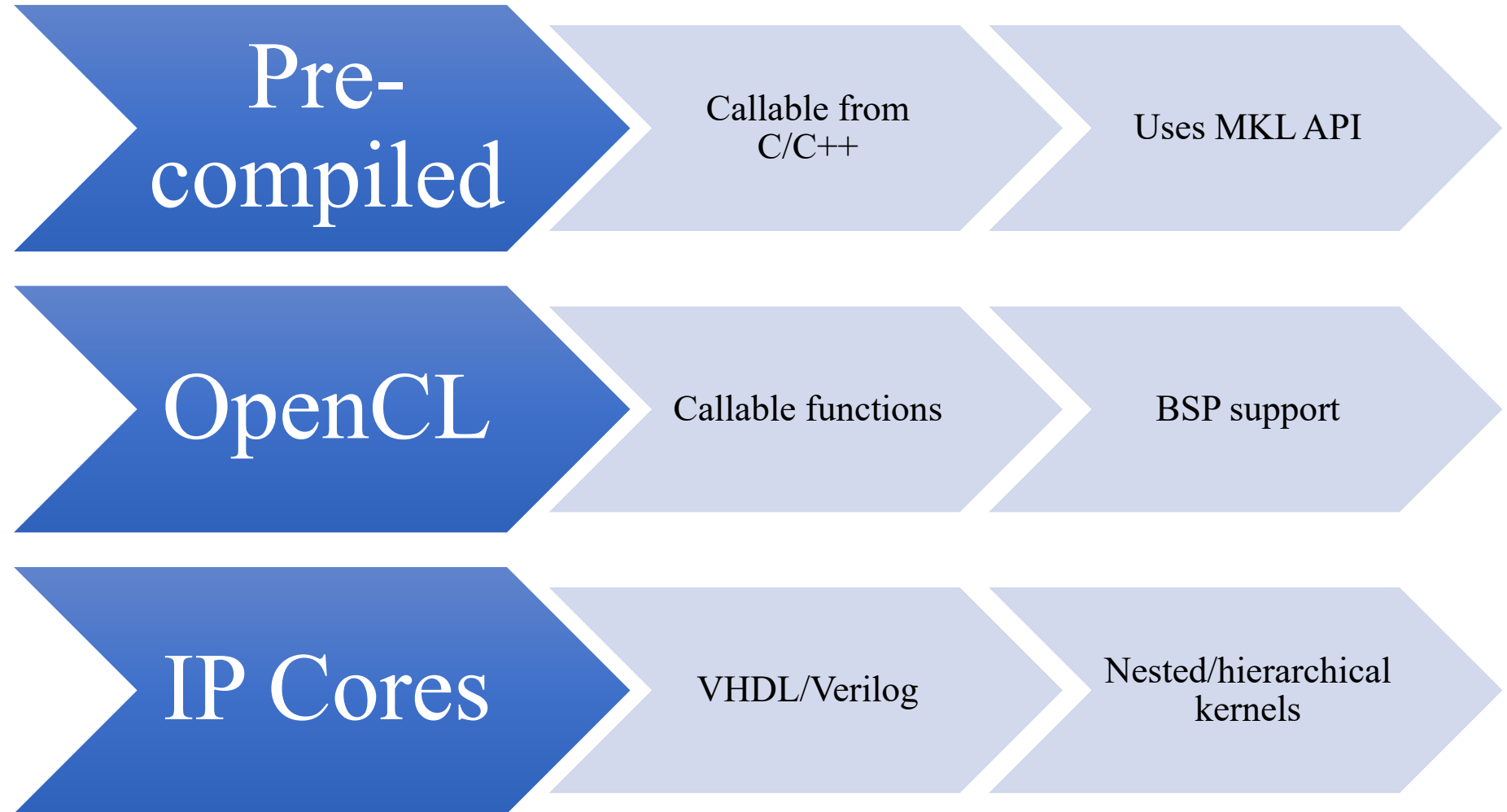
FPGA + CNN IP

ACCELERATING AND OFF-LOADING CHALLENGING FINANCIAL COMPUTATIONS



OVERVIEW

Libraries at 3 levels



FINANCIAL LIBRARY PHASE 1

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.

FinLib Phase 1, v0.9 Models

Model	Product
Black-Scholes	European exercise, pricing & risk
Black-Scholes-FFT	European exercise – market calibration
Garman-Kohlhagen	European exercise – foreign currency
Curran	European exercise – arithmetic average
Cox-Ross-Rubenstein	American exercise – spot and futures
Bjerkstrand-Stensland	American exercise – very fast approximation
Merton	European exercise – on dividend paying stocks
Kirk	European exercise – lognormal spread
Bachelier	European exercise – normal spread

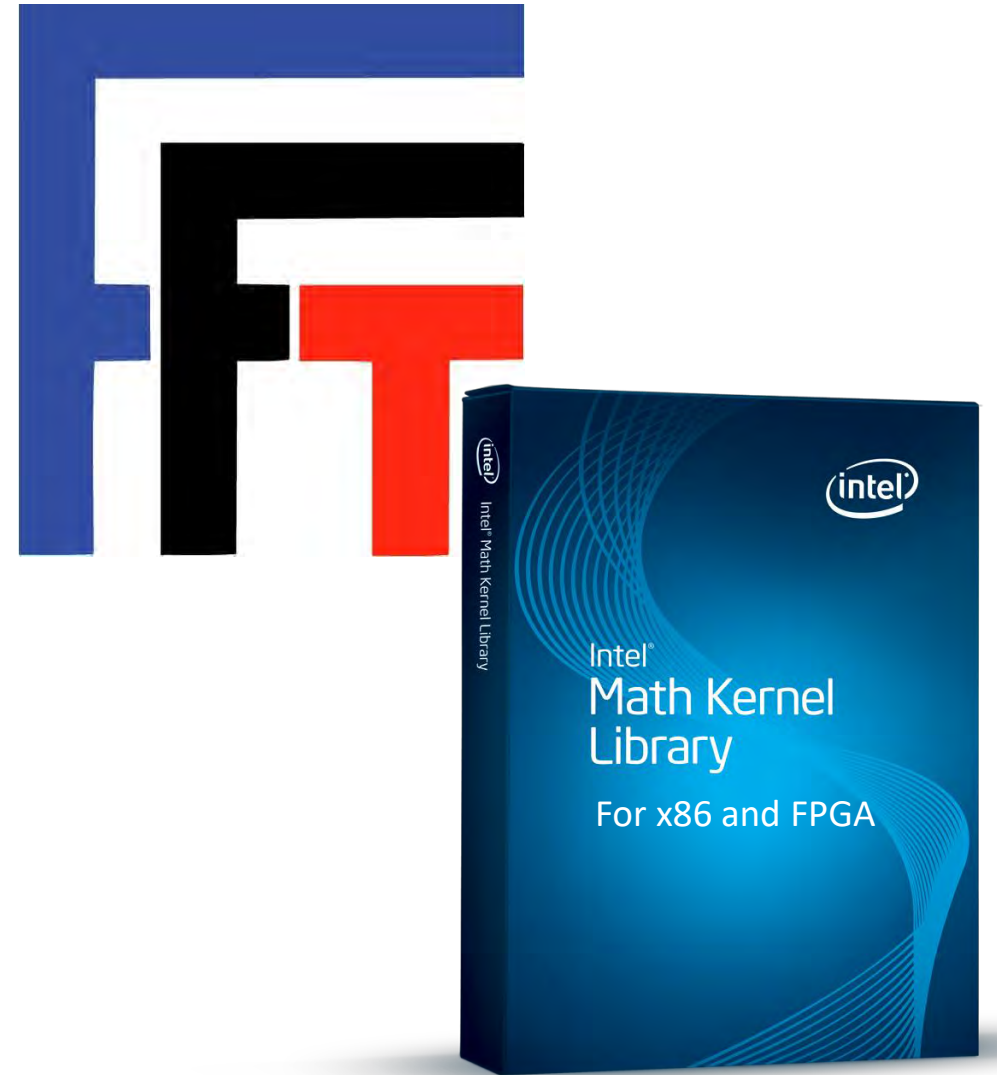
FINANCIAL LIBRARIES PERFORMANCE

- 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

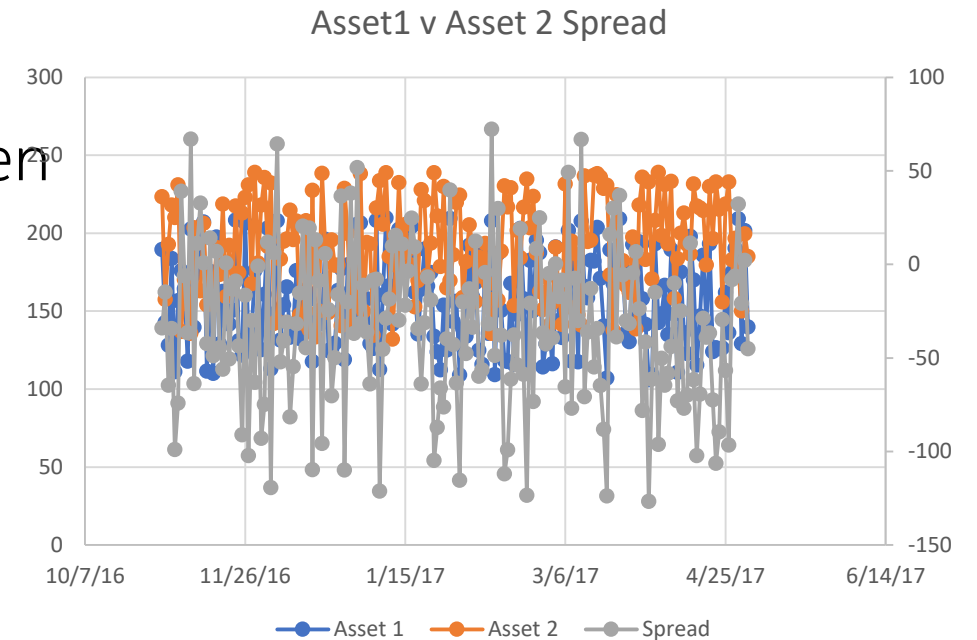
Generalised Monte Carlo path generator

Example option payoff is European style exercise on the average of the spread between two underlyings:

$$\text{Payoff} = \text{Max}(\text{CP} * (\text{Avg1} - \text{Avg2} - \text{Strike}), 0)$$

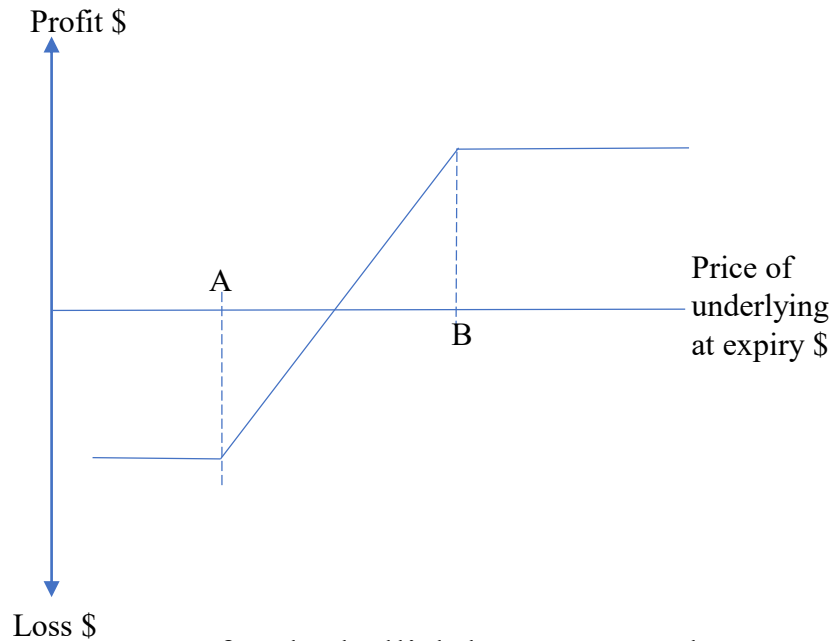
/// To find the value of an average rate or Asian option on the spread between two assets:

```
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)
```



RAPID DEVELOPMENT OF ALGORITHMIC TRADING STRATEGIES

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

Collar

	Buy Put	Sell Call
Spot price (S)	1,320.00	1,320.00
Strike price (X)	1,250.00	1,436.61
Time to maturity (T)	0.5000	0.5000
Risk-free rate (r)	2.00%	2.00%
Volatility (s)	30.00%	30.00%
Forward price	1,333.2662	1,333.2662
Option Value	71.7324	71.7324

```
/// 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.
```

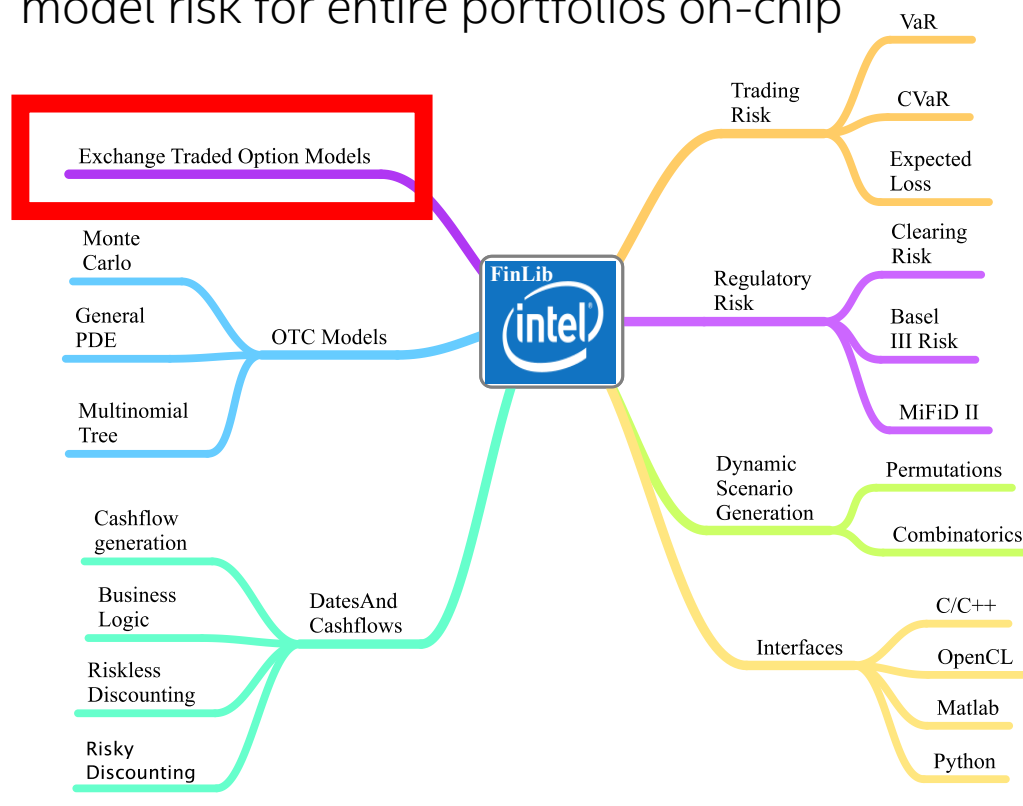
```
/// buySell = 2 means buy call sell put (i.e. user supplies call details, function
```

```
/// 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)
```

SCALABLE PORTFOLIO RISK

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.

1

MLLib

CNN, RNN, SVM, Random Forests, Boosted Trees etc

FinLib

Pricing, valuation and risk for low and high frequency trading

LALib

SGEMM/BLAS type functions; tensor methods

FFTLib

Forward, backward, super-sampling etc

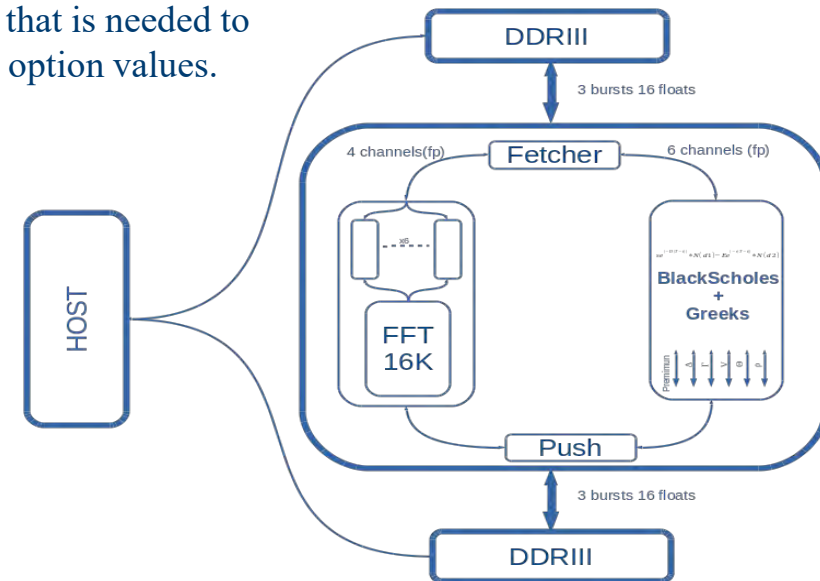
MALib

Core math functions: Sin, Cos, Tan, Log, Exp etc
Core stats functions: univariate and multivariate stats

2

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

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



3

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

<https://software.intel.com/en-us/intel-parallel-studio-xe/try-buy>

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!



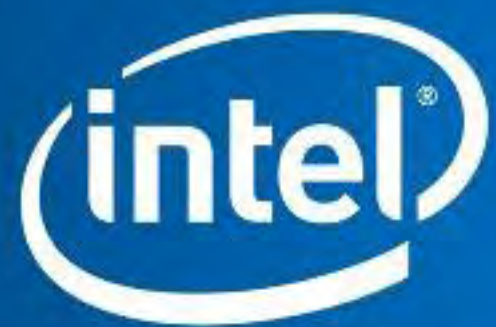
LEGAL DISCLAIMER & OPTIMIZATION NOTICE

- 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



Software