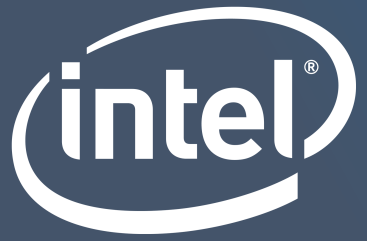


Improving FPGA Use

Financial Library Solutions

Mike Strickland, Intel-PSG

Steve Weston, Intel-PSG

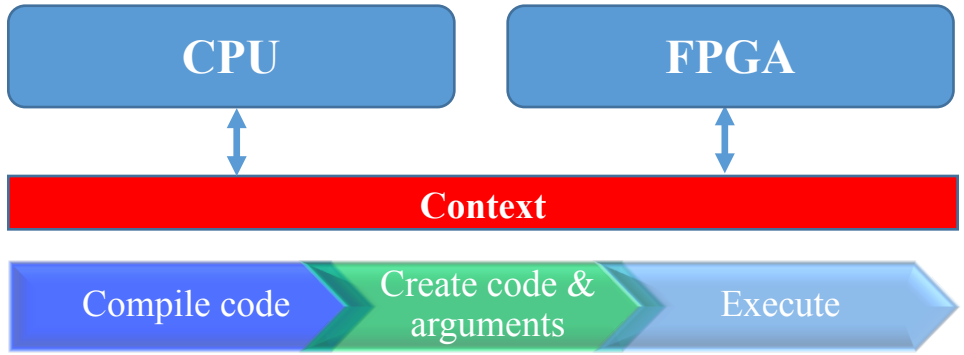
STAC, New York, 07 November 2016

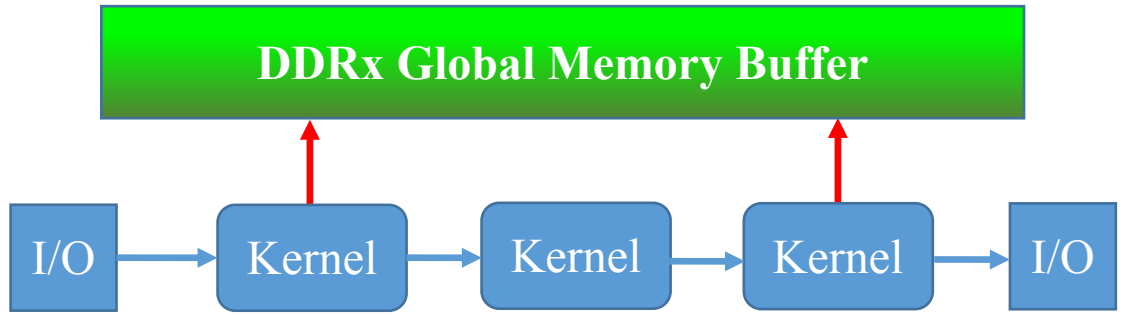
Overview

- Going beyond OpenCL for Ease of Use
- Using Libraries to scale up and out
- Frameworks for Analytics and Machine Learning

Going beyond OpenCL for Ease of Use

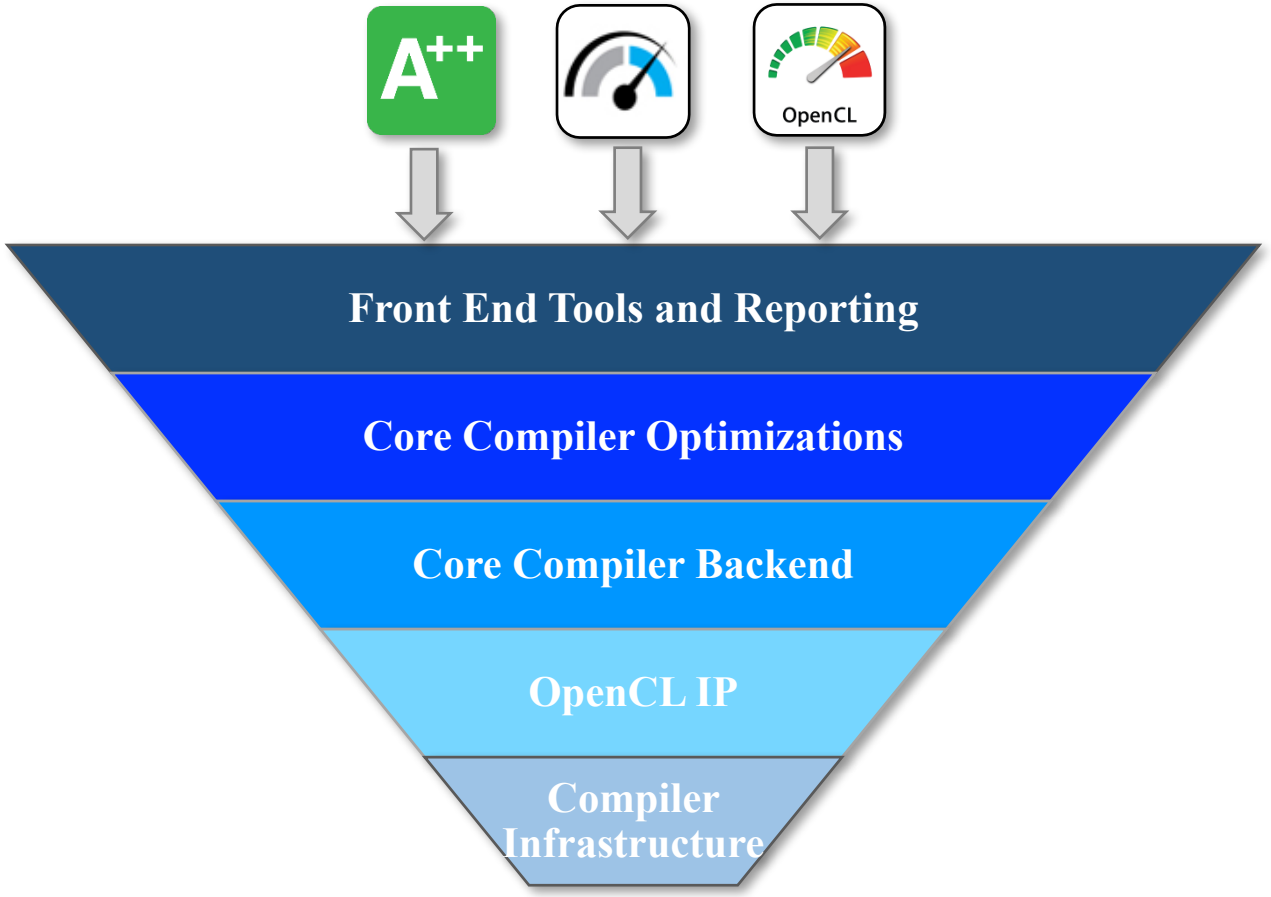
- Software Programmers
  - Need Logic and Data Management - need to write lines of code!
- OpenCL Compiler Benefits
  - Ease of use
  - Scalable
  - Heterogeneous
  - Leverage existing libraries
  - Vendor choice through open standards
- Channels/Pipe Extensions
  - Kernel ➔ Kernel
  - External I/O ➔ Kernel
  - Mix and match HDL and Kernels



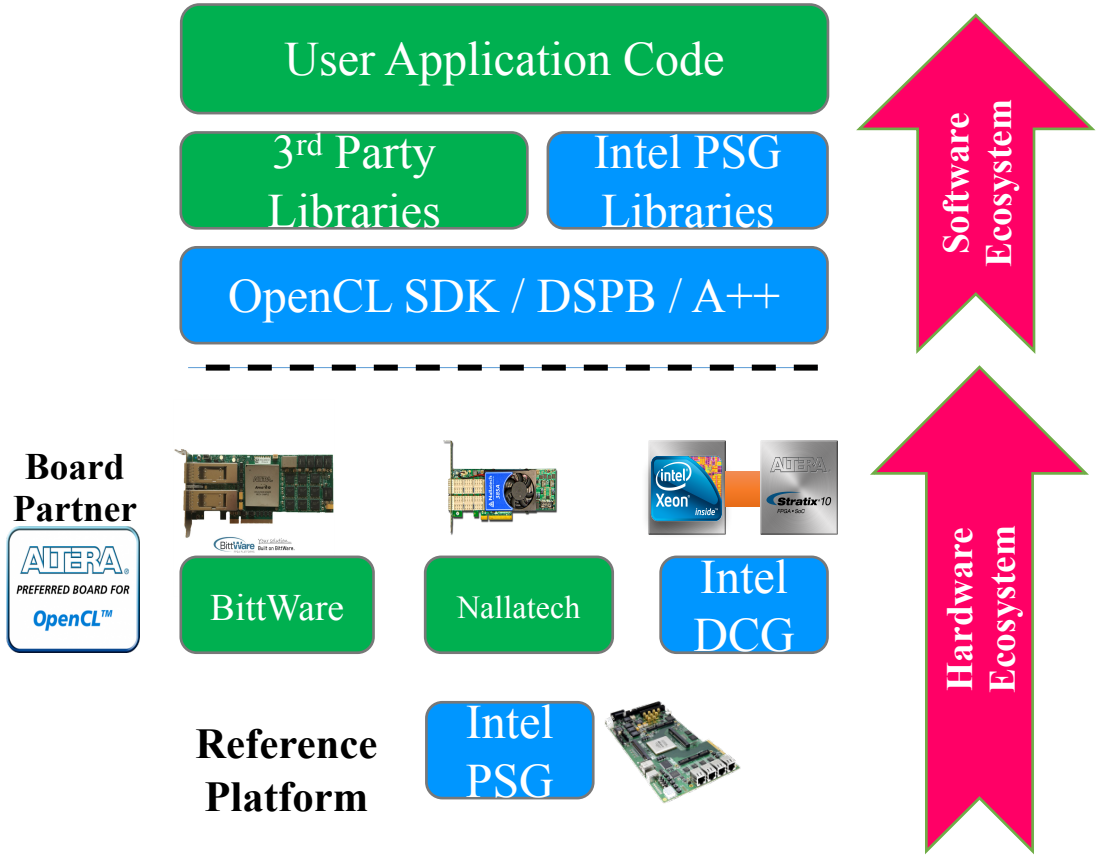


Add Libraries to Existing Programming Interfaces

Today: “C to Gates”, Matlab, OpenCL



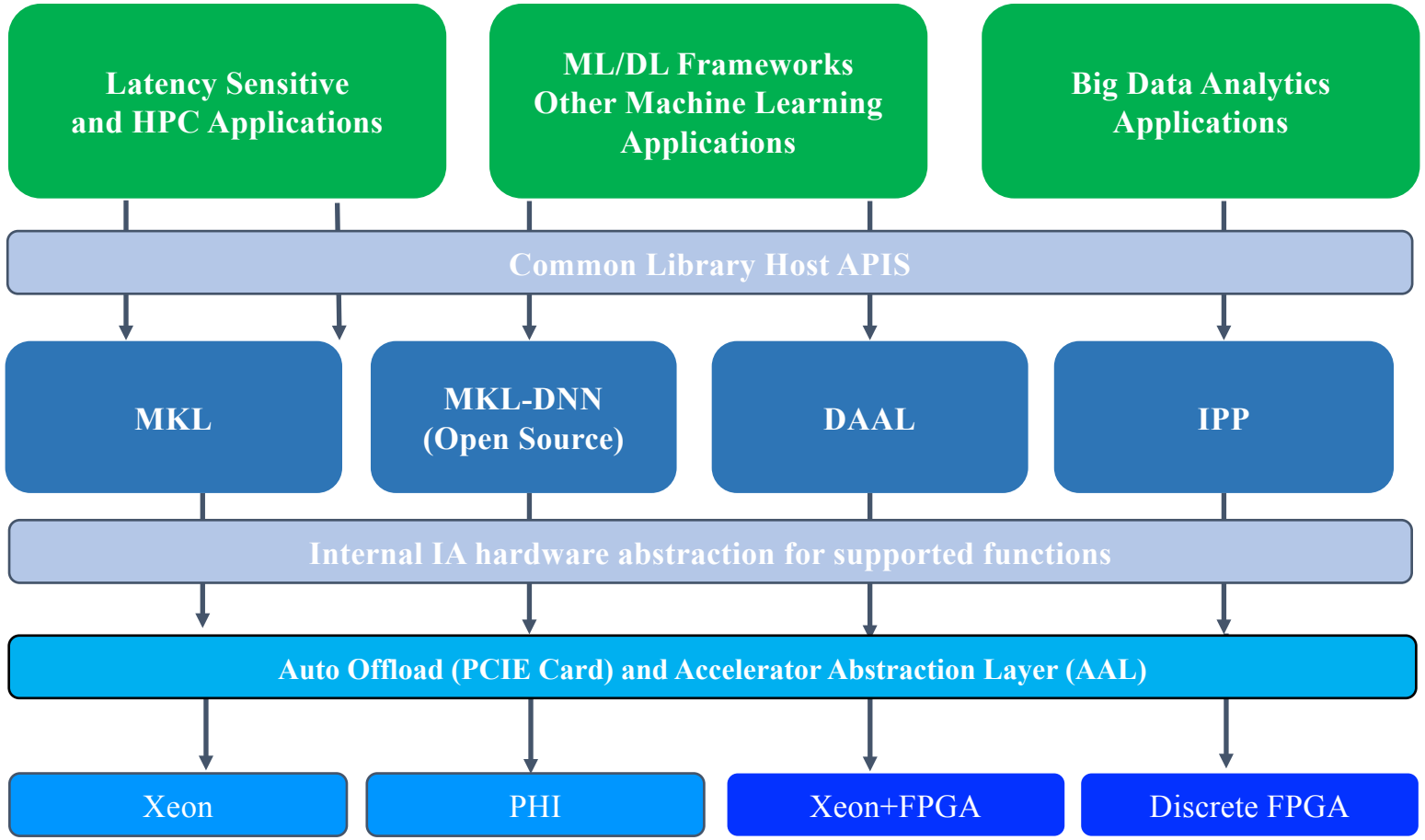
Turnkey Libraries AND
User Defined Libraries



Exposing Libraries via Intel-MKL

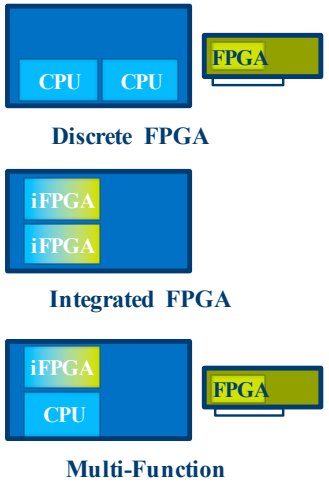
New capability:

- Adding to existing Xeon library functions currently exposed through MKL.
- In future, calls to selected MKL functions will also execute on either Xeon/Phi or FPGA.
- Minimsing changes to users application code.

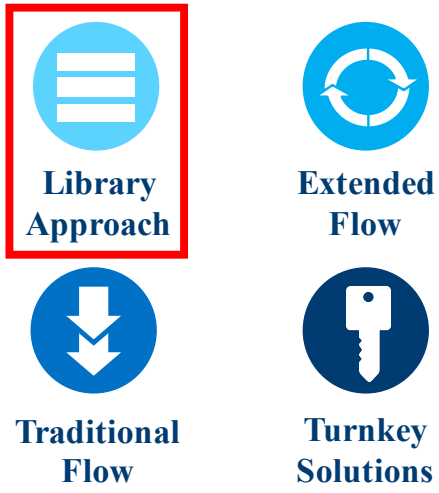


Libraries are key to scaling

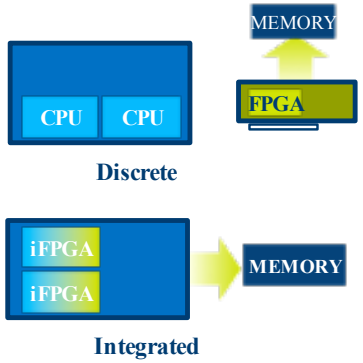
System Arch and Form Factor



Solution and Development



Memory Attachment



Libraries will enable:

1. **FPGAs to be used easily and conveniently by software users.**
2. **Close-coupling with Xeon enables users to choose the right solution for their specific problem.**
3. **Users to scale up the software stack and scale out across multiple devices.**

Libraries will be delivered at several levels:

High Level Algorithms
- Risk management
- Machine learning
- Big data(centre) analytics

Functional Components
- Pricing/valuation functions
- Filters and signal processing
- Statistical functions

Low Level Primitives
- Mathematics
- Arithmetic
- Accuracy and precision

Turnkey financial libraries

Exchange Traded	Payoff	Product coverage	Phase 1
Black-Scholes	European	European exercise options on all underlyings	✓
Bjerksund-Stensland	American	American exercise options on most underlyings	✓
Bachelier	Spread (normal)	European exercise options on normal spreads	✓
Cox-Ross-Rubenstein	American	American exercise options on equities	✓
Curran	Average rate	European exercise on arithmetic average on most underlyings	✓
Garman-Kohlhagen	European	European exercise on currencies	✓
Jewson	European binary	European exercise on heating and cooling days	✓
Kirk/Bjerksund	Spread (lognormal)	European exercise options on log-normal spreads	✓
Merton	European	European exercise on dividend paying single stocks	✓

Demo using Black-Scholes

$$C(S,t) = N(d_1)S - N(d_2)Ke^{-rt}$$

$$d_1 = \frac{1}{\sigma\sqrt{t}}\left[\log\left(\frac{S}{K}\right) + t\left(r + \frac{\sigma^2}{2}\right)\right]$$

$$d_2 = \frac{1}{\sigma\sqrt{t}}\left[\log\left(\frac{S}{K}\right) + t\left(r - \frac{\sigma^2}{2}\right)\right]$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} e^{-\frac{1}{2}z^2}dx$$

$$\textit{delta:} \quad \Delta_C = \frac{\partial C}{\partial S} = \mathrm{N}(d_1) \ ,$$

$$\textit{gamma:} \quad \Gamma_C = \frac{\partial^2 C}{\partial S^2} = \frac{\mathrm{N}'(d_1)}{S\sigma\sqrt{\tau}} = \frac{K\,\mathrm{e}^{-r\tau}\,\mathrm{N}'(d_2)}{S^2\sigma\sqrt{\tau}} \ ,$$

$$\textit{theta:} \quad \Theta_C = \frac{\partial C}{\partial t} = -rK\,\mathrm{e}^{-r\tau}\,\mathrm{N}(d_2) - \frac{\sigma S N'(d_1)}{2\sqrt{\tau}} = -K\,\mathrm{e}^{-r\tau}\left[r\,\mathrm{N}(d_2) + \frac{\sigma\,\mathrm{N}}{2\sqrt{}}\right.$$

$$\textit{rho:} \quad \rho_C = \frac{\partial C}{\partial r} = \tau K\,\mathrm{e}^{-r\tau}\,\mathrm{N}(d_2) \ ,$$

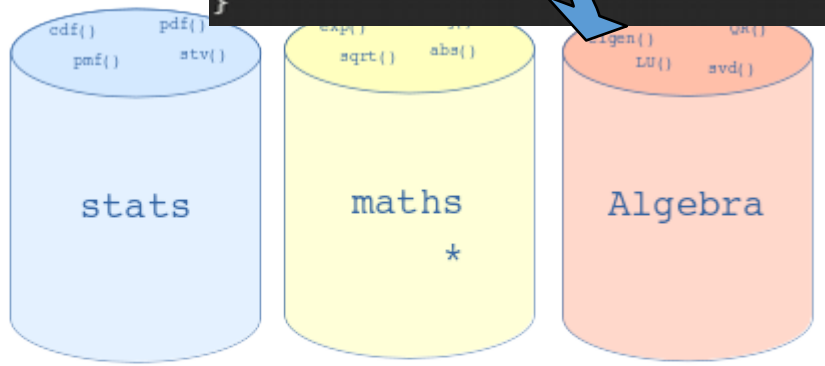
$$\textit{vega:} \quad \mathcal{V}_C = \frac{\partial C}{\partial \sigma} = \sqrt{\tau}S\,\mathrm{N}'(d_1) = \sqrt{\tau}K\,\mathrm{e}^{-r\tau}\,\mathrm{N}'(d_2) \ .$$

```
/// The
/// call
/// call
/// spot
/// stri
/// r =
/// vol = volatility of the stocks return (a number between 0 and 1)
/// t = time to option maturity (in years)
/// N(d) = normal cumulative distribution function
///
float OptionPricer(char callPutFlag, float spot, float strike, float vo
{
    float d1, d2;

    d1 = (log(spot/strike) + (r + vol * vol / 2) * t) / (vol * sqrt(t))
    d2 = d1 - vol * sqrt(t);

    if (callPutFlag == 'c')
        return spot * CND(d1) - strike * exp(-r * t) * CND(d2);
    else if (callPutFlag == 'p')
        return strike * exp(-r * t) * CND(-d2) - spot* CND(-d1);
}
```

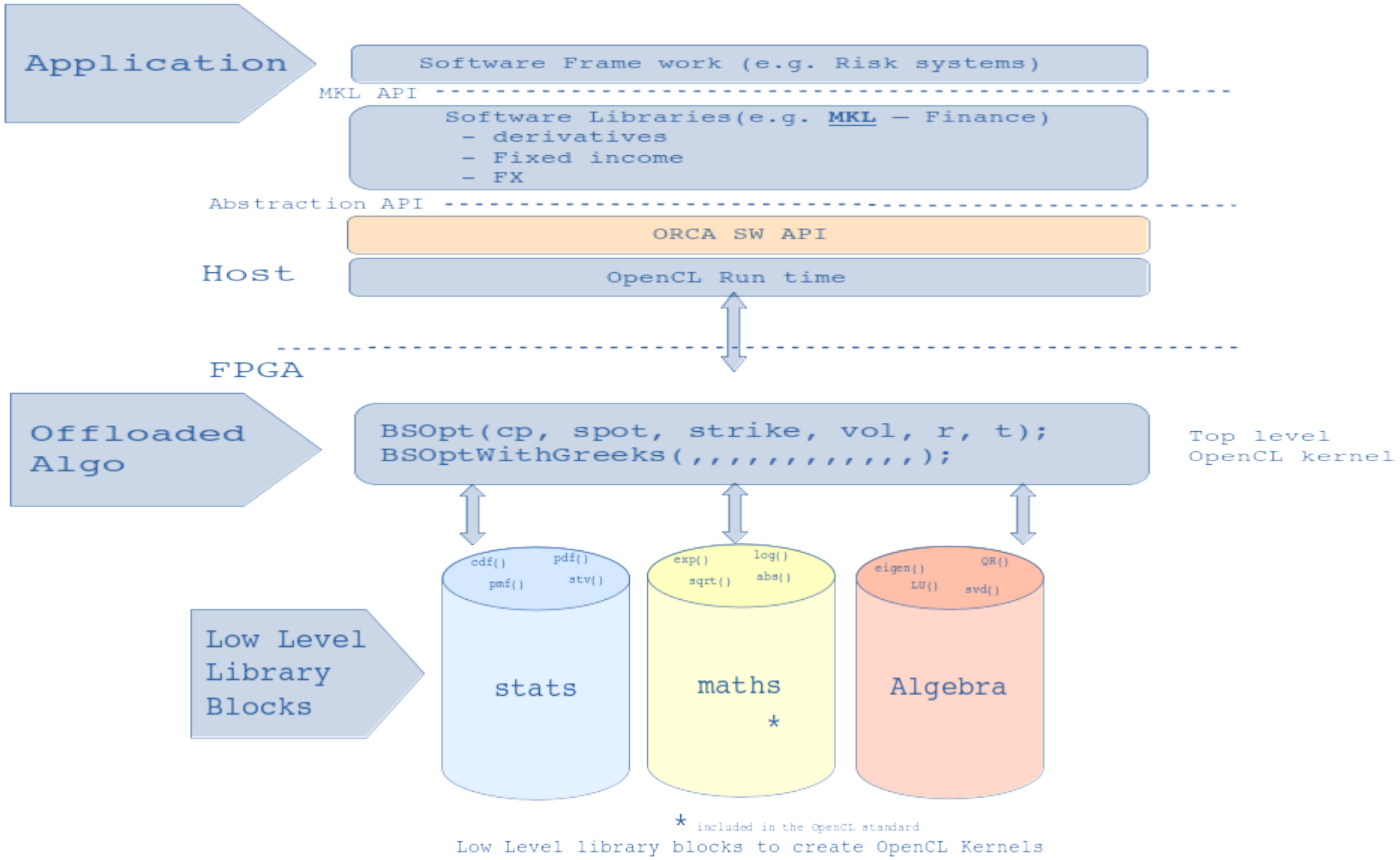
Low Level Library Blocks



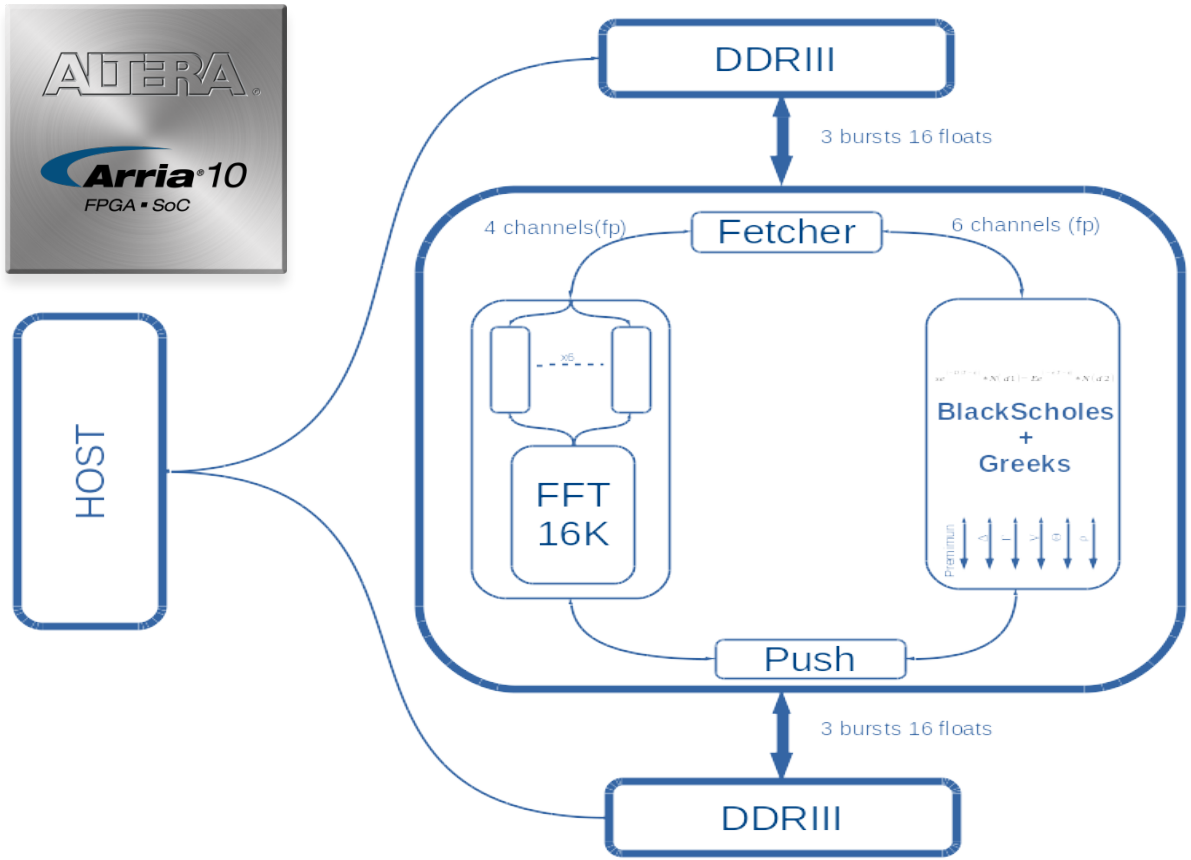
* included in the OpenCL standard

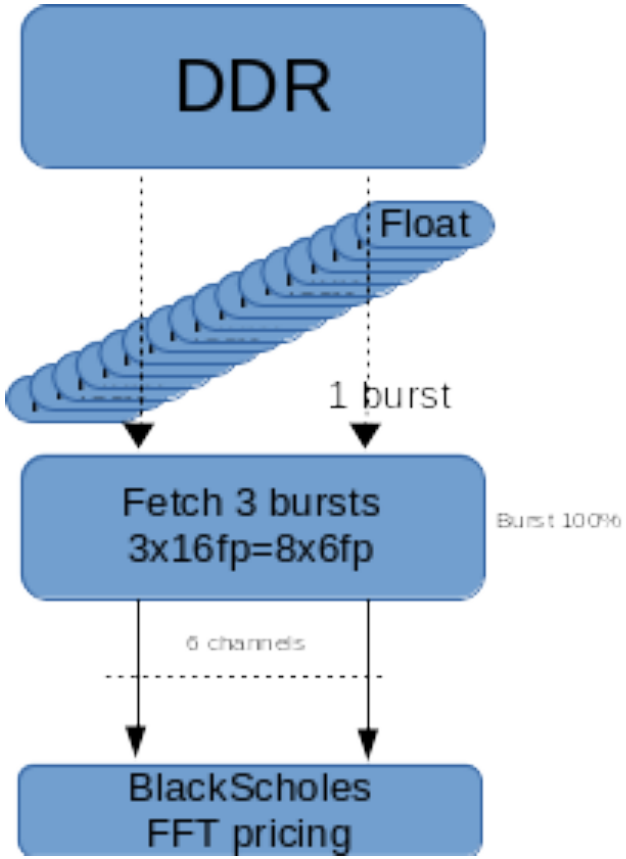
Low Level library blocks to create OpenCL Kernels

Example using Black-Scholes

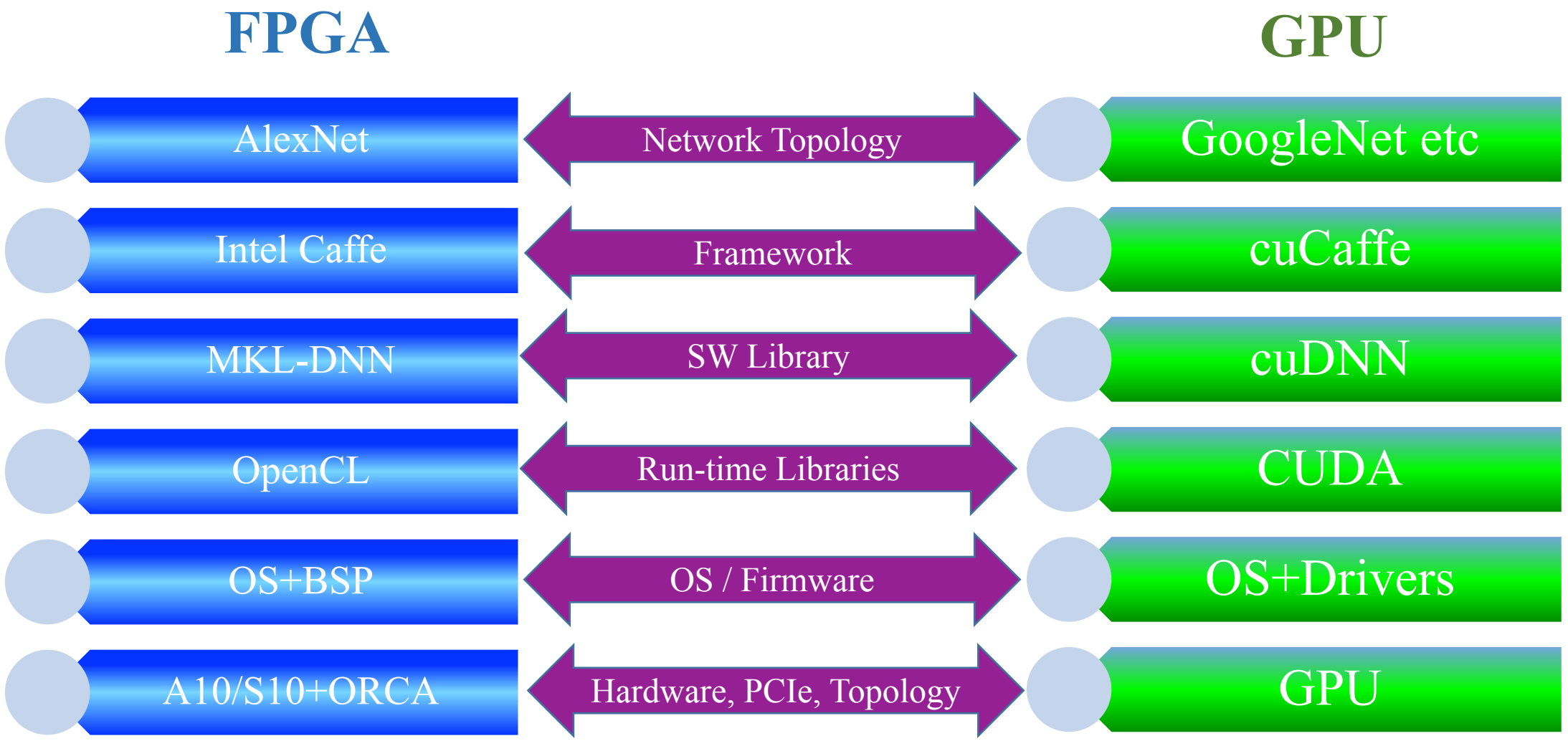


Example using Black-Scholes





Enabling a more generic framework



Thank you

If you have any questions
please stop by the Intel stand!