Programming and Benchmarking FPGAs with Software-Centric Design Entries

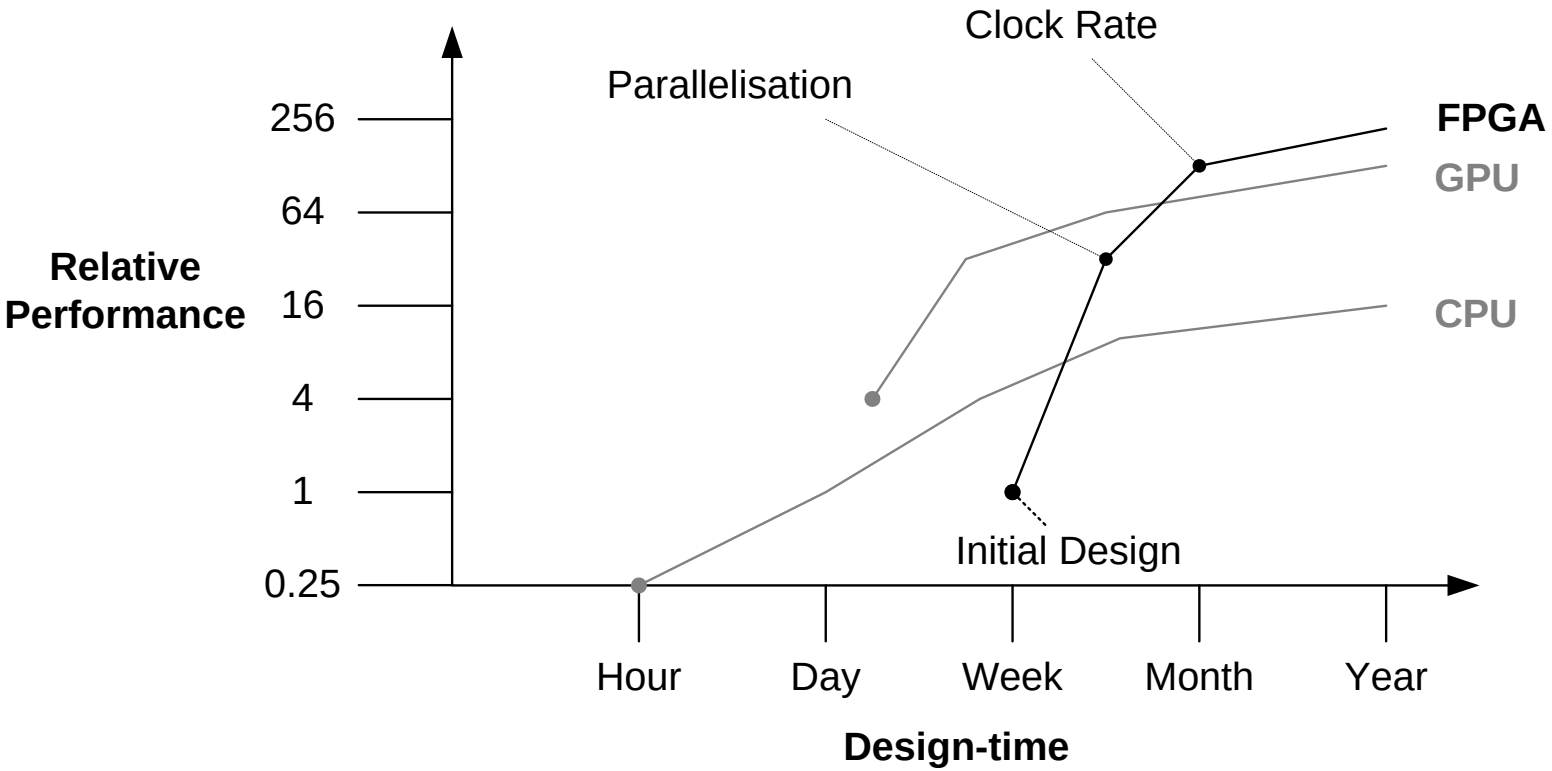
Michaela Blott

Principal Engineer, Xilinx Research

September 2015

Productivity Gap: Application Perspective

(David Thomas, Imperial College, UK)

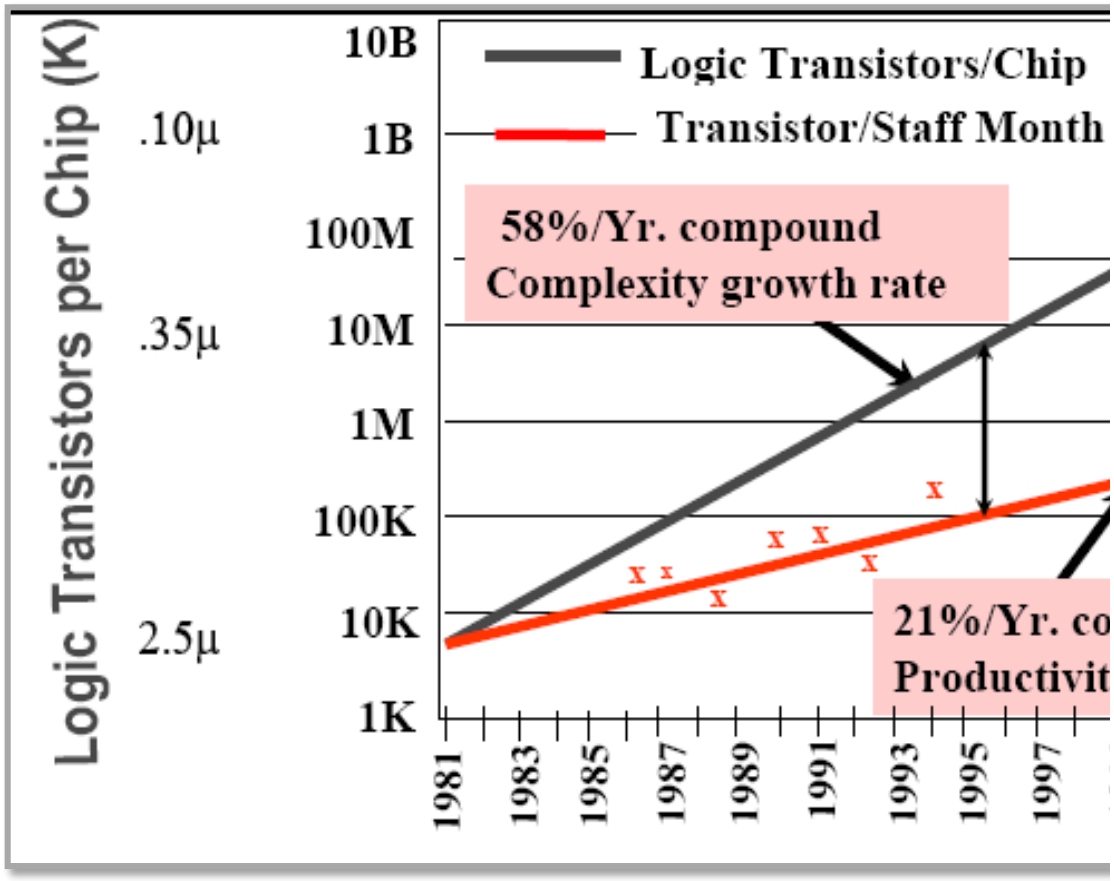


FPGAs provide large speed-up and power savings – *at a price!*

Days or weeks to get an initial version working

Multiple optimisation and verification cycles to get high performance

Productivity Gap: Device Perspective



Normal mortals cannot easily program massively parallel systems

Evolution of Design Environments

Time

Abstraction

Legacy

- ISE, RTL-based design entry with IP library

Embedded CPU integration

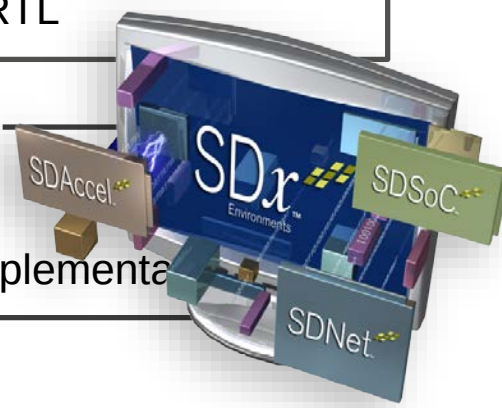
- Microblaze, SDK, EDK

Raised abstraction for accelerators

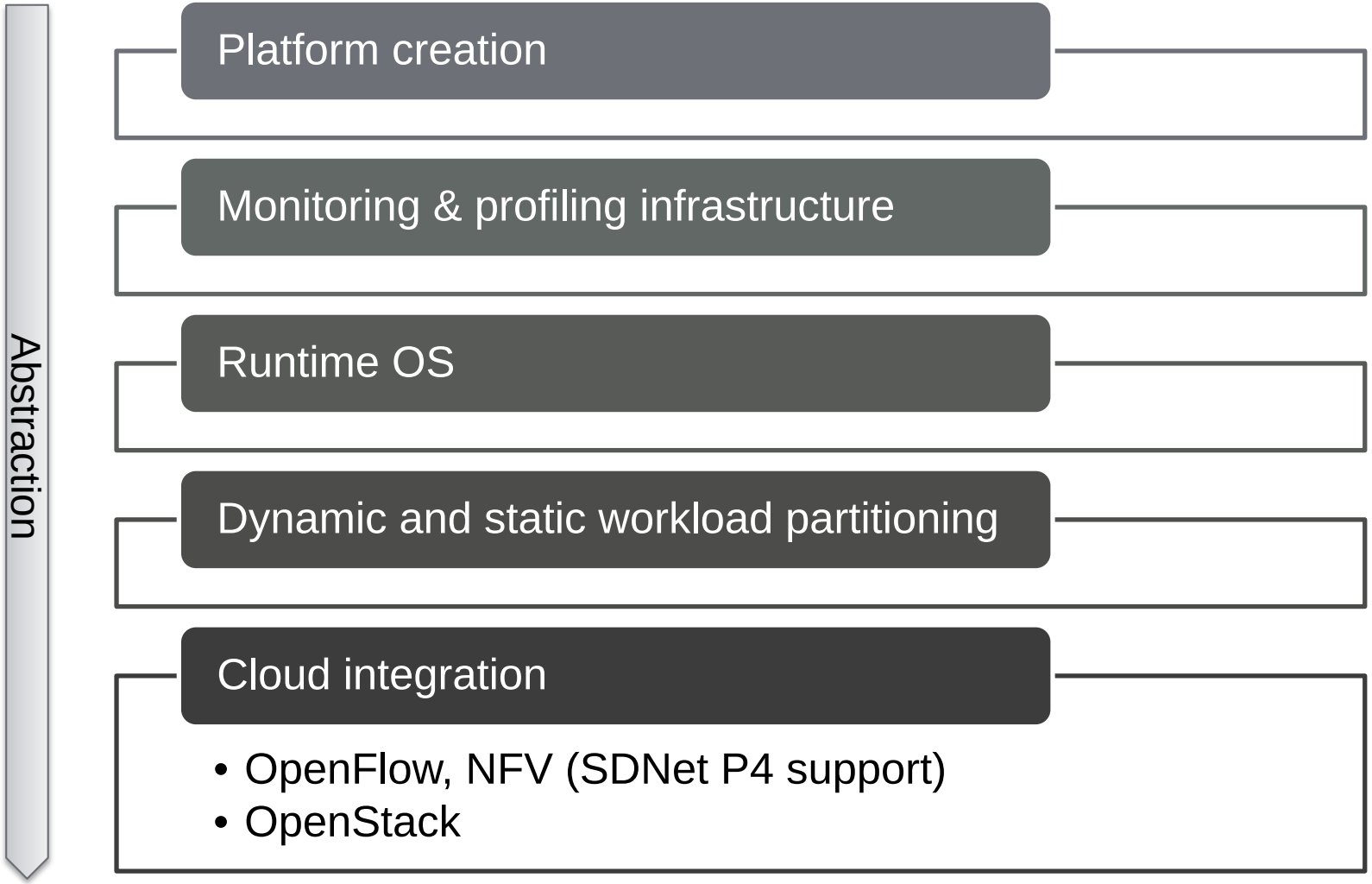
- Vivado HLS
- SDNet (DSL PX)
- Block stitching and manual integration in platform in RTL

Simplified host integration & automated infrastructure creation

- SDSoC, SDNet, SDAccel
- Predefined methods for data transfer & automated implementa

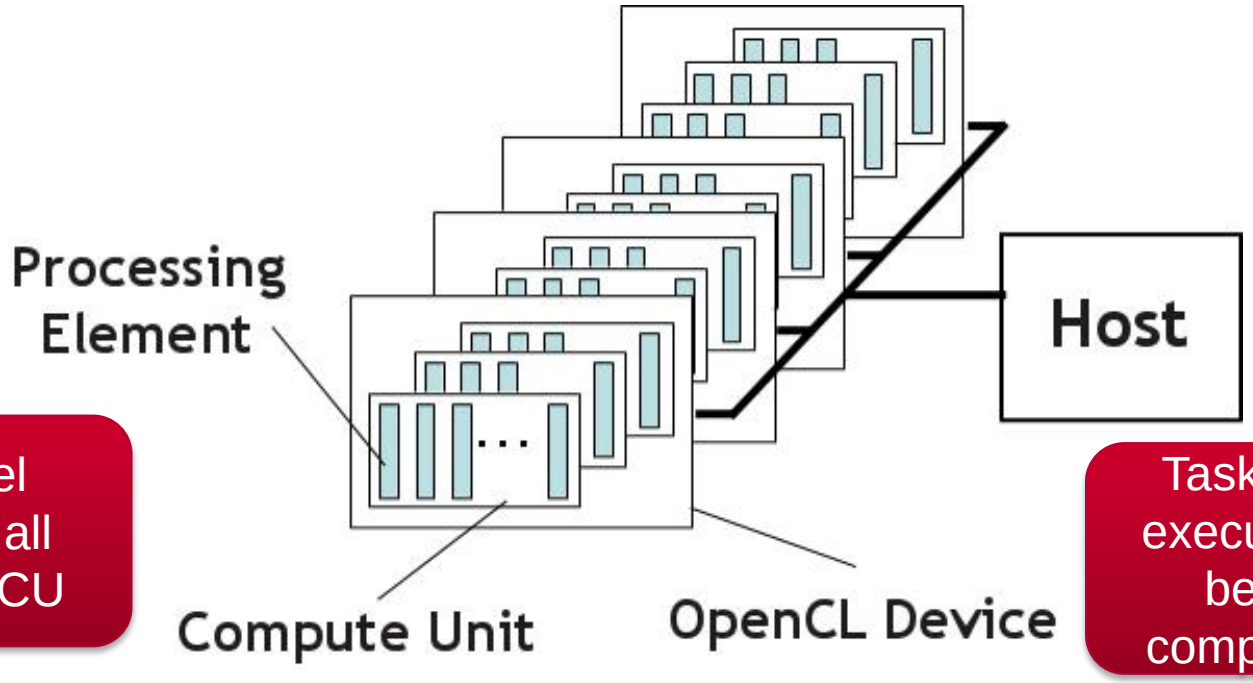


Future



OpenCL

- **Evolving standard since 2008**
- **Defined by Khronos Group**
  - Broad Vendor Adoption : CPU, GPU, DSP, FPGA
  - Application Developers : Adobe, Huawei, Baidu, Fujitsu,
- **Initial GPU-favourable hardware abstractions, now moving towards a more neutral standard**



Data parallel execution of all PEs within 1 CU

Task parallel execution only between compute units

Now that we have platform compatibility*,

…

we can figure out

what application works on what device!

***for a subset of applications that fit the OpenCL execution model, and don't run into resource limitations, and don't use vendor specific extensions**

Challenges in Benchmarking with OpenCL

- **No performance portability**
  - Numerous implementations for each application yield very different results
    - High discrepancy between out-of-the-box and optimized performance
    - Optimizations for different platforms are very different
    - Many benchmarks are biased
    - Running GPU benchmarks on FPGAs is non-optimal
- **Figure of merits/cost functions for a diverse set of hardware architectures**
  - Instruction sets are fundamentally different
  - Different hardware counters, different profiling tools for different platforms
- **Correlating theoretical numbers (hardware-neutral) with measured results**
- **What is an acceptable development effort?**
- **Over what do we normalize: cost, technology node?**

Key Concepts

- **Characterization (broad)**
  - Profile & characterize a set of potential hardware platforms
    - Leveraging and extending UC Berkeley roofline model

    (Cabezas et al "Extending the roofline model: Bottleneck analysis with microarchitectural constraints. IISWC 2014)
  - Analyse and profile a broad set of applications
    - Compute load and memory requirements
  - Correlation yields performance insights
- **Benchmarking (deep)**
  - Out-of-the-box implementation
    - Static code analysis
    - Profiling on all platforms
  - Optimize implementations for all platforms until
    - Optimal performance is reached, or discrepancy to optimal performance is fully understood
    - Static code analysis
    - Profiling on all platforms

Abstractions

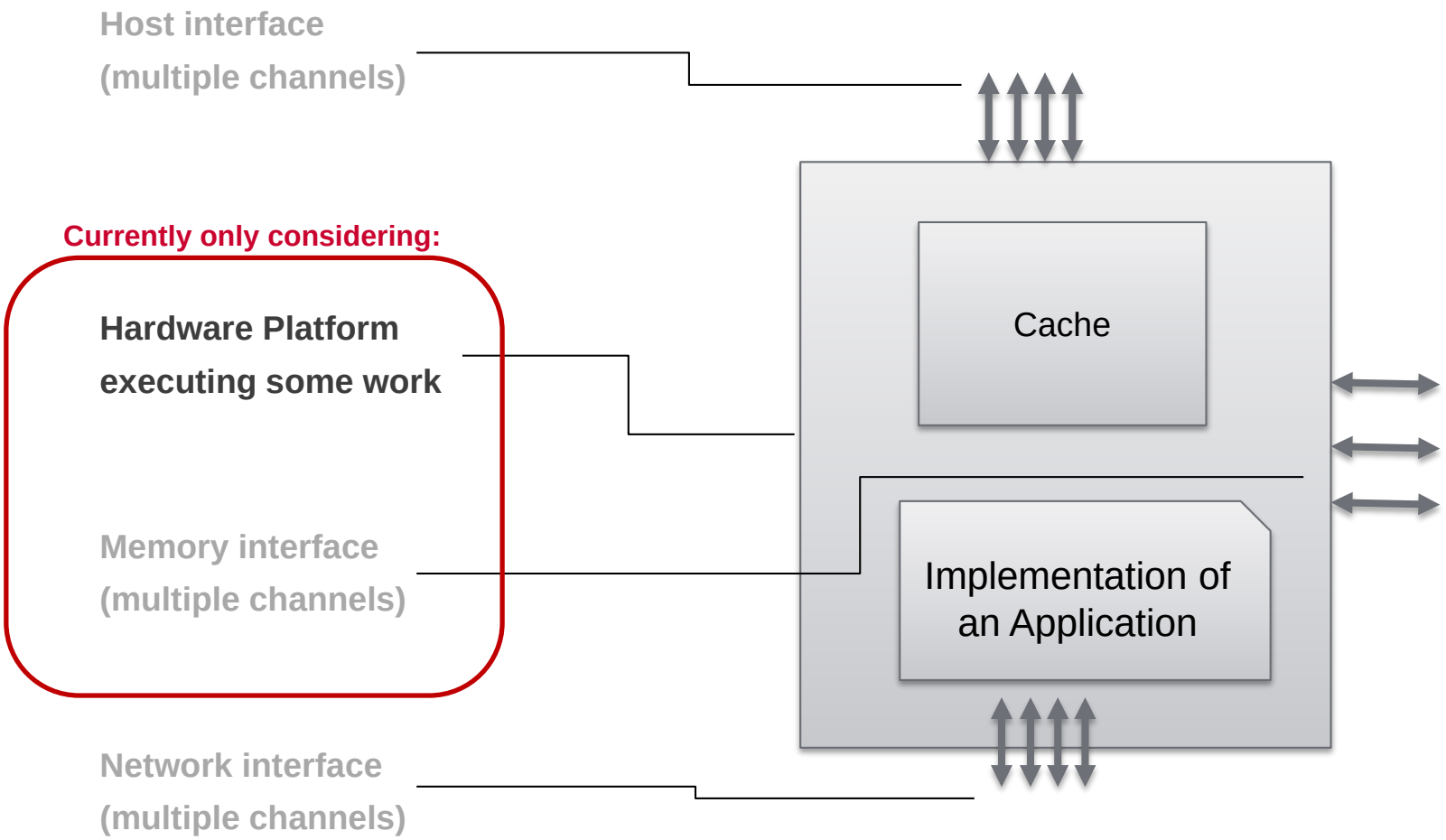


Figure of Merits

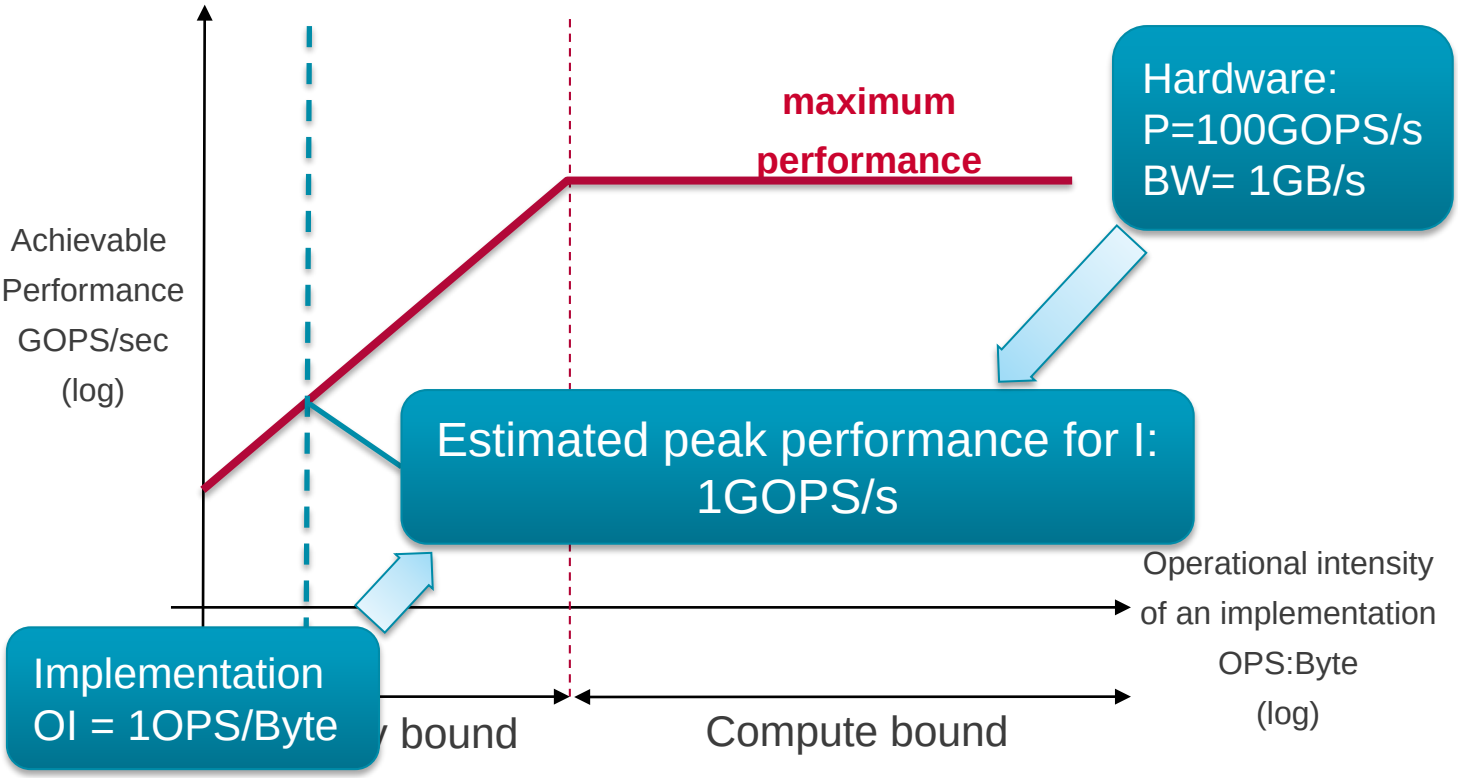


- **Implementation I of application A**
  - Work W [OPS], float and non-float
  - Device memory traffic DMT [Bytes]
  - Operational Intensity OI=W/DMT
  - Execution time T [ms] (memory, accelerator, memory)
  - Performance P [OPS/s]
  - Power consumption PWR [Watt]
  - Energy consumption E [Joules]
  - Ratio R of out-of-the-box : optimized
  - Number of design revisions REV
- **Hardware Platform**
  - Theoretical and measured peak performance P [OPS/s], float and non-float*
  - Theoretical and measured peak memory bandwidth BW [Bytes/s]
  - Peak power consumption TDP [Watt]

**Max of LINPACK, SHOC L0, CLPeak*

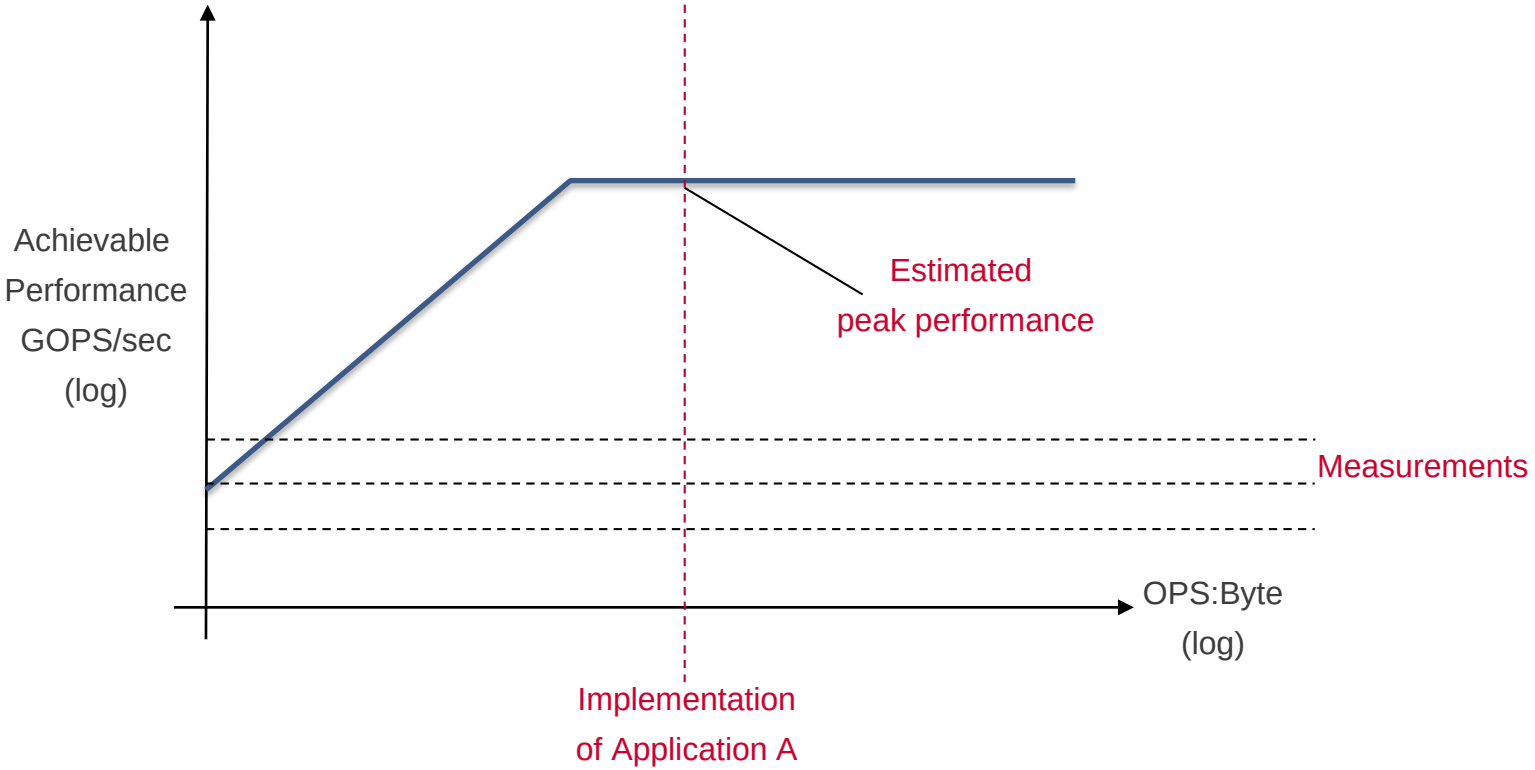
Rooflines for Hardware Platforms

- **Peak performance as a function of operational intensity**
  - *P = min{ OI*BW; P}*



Applications in Rooflines

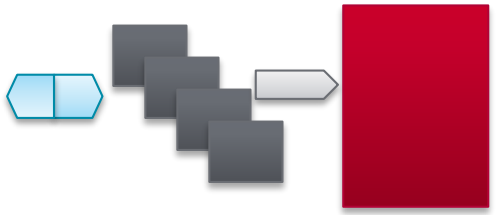
- **Allows performance estimates, tracking of optimizations**



Considered Platforms

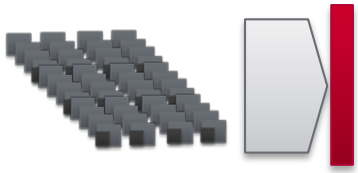
Considered Hardware Platforms

- **Xeon processors (E5-2660 Ivybridge)**
  - Few beefy cores at high rates & ooo execution (10cores @ 2.2GHz)
  - Large memory @ medium bandwidth (64GB @ 60GBps
  - Fully coherent memory subsystem (automatically managed)
  - 256b vector units per every core



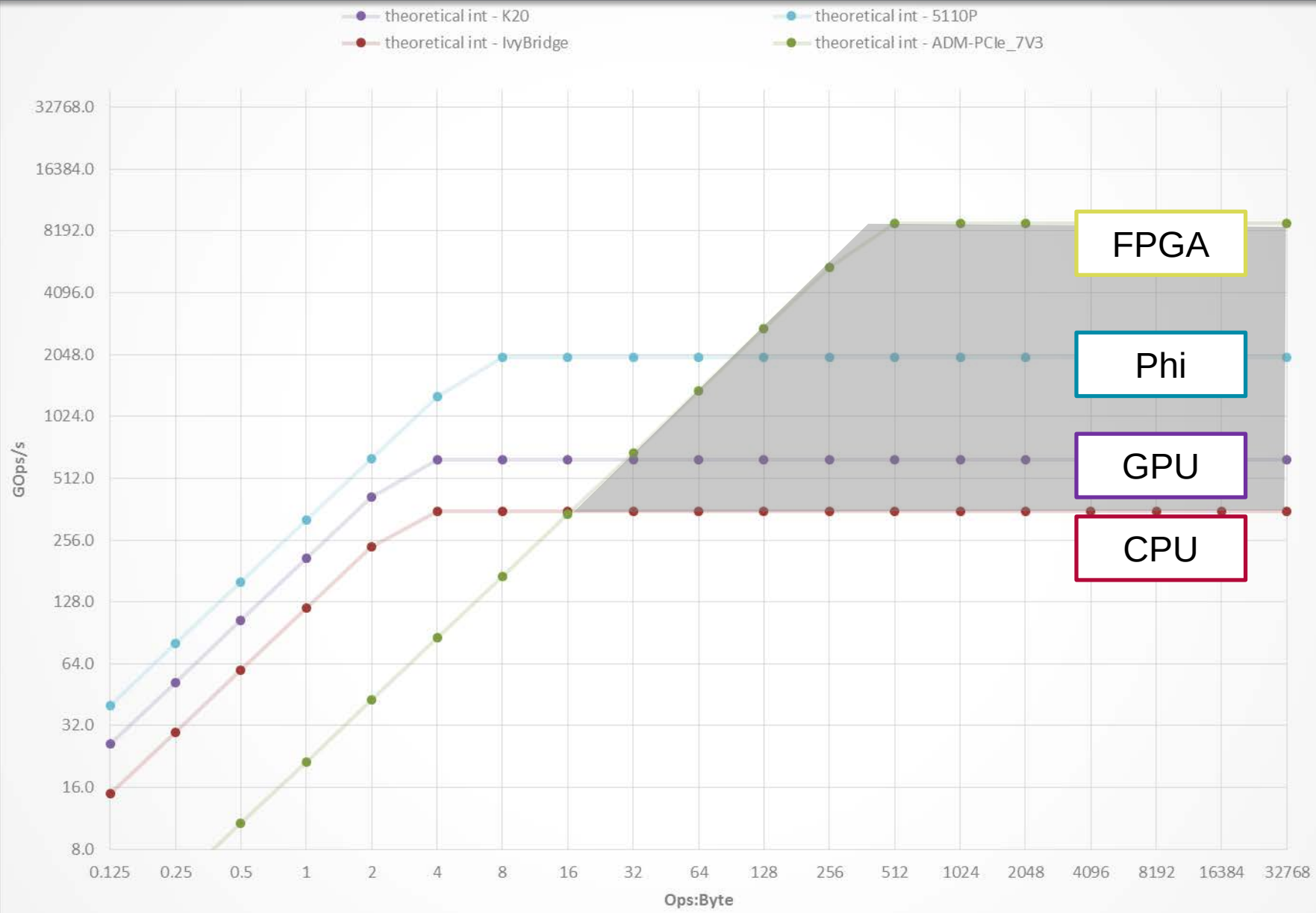
- **Xeon Phi (5110P)**
  - More beefy (in-order processing) cores (60 cores @ 1.053GHz)
  - Fast external memory with lower density (8GB @ 320GBps)
  - Fully coherent memory system
  - 512b vector processing units per core

- **Nvidia K20x**
  - Huge amount of light-weight threads SIMT(13*192 @ 732MHz)
  - Fast external memory with lower density (6GB @ 250GBps)
  - Specialized hardware DP units

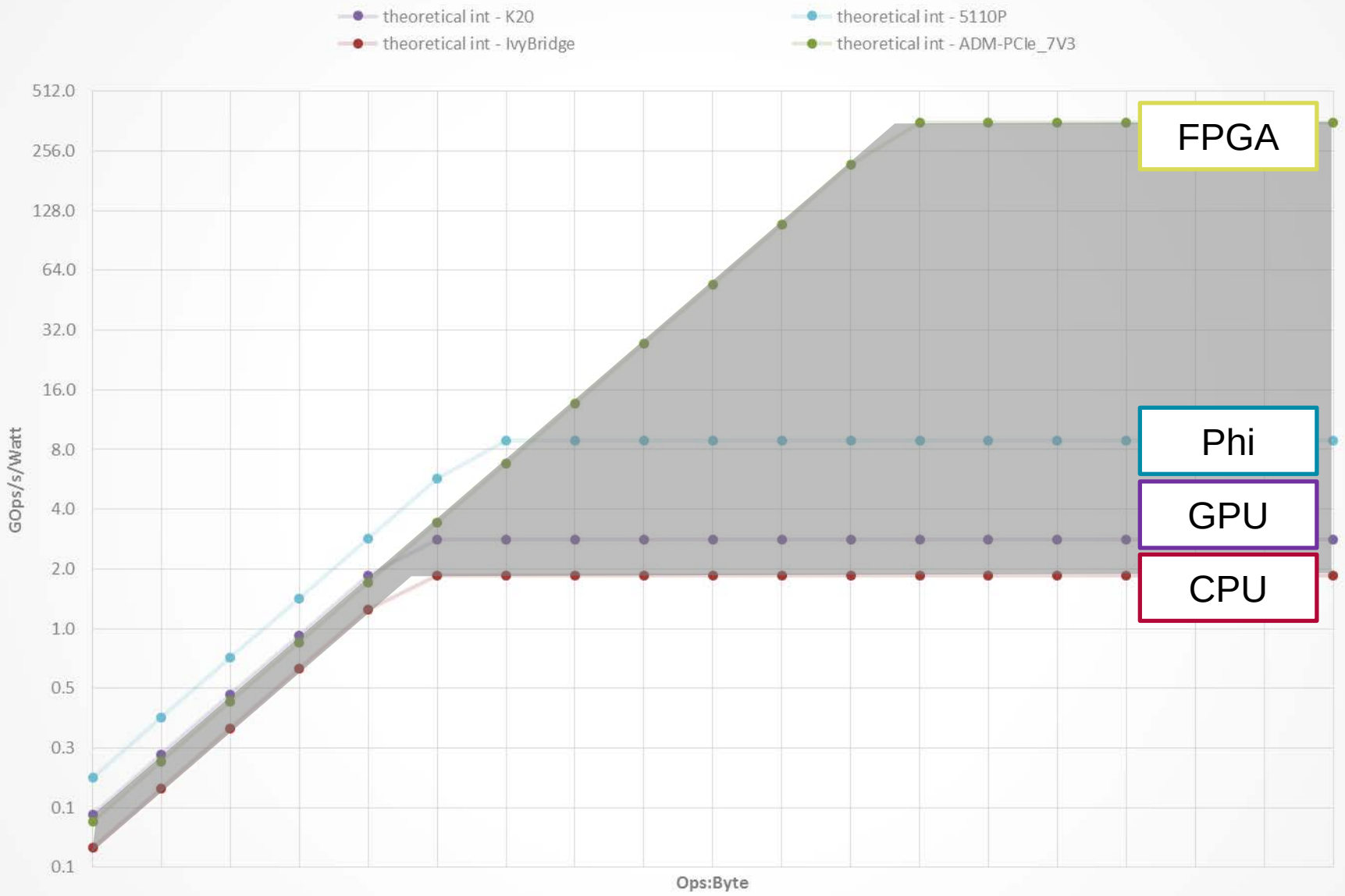


- **FPGA ADM_PCIe_7V3**
  - Massively parallel and fine grained architecture
  - Medium memory at slow speed (16GB @ 21.3GBps)
  - High power efficiency (25W)
  - Massive network connectivity

Roofline for Non-Float. Theoretical Peak Performance



Roofline for Non-Float. Theoretical Peak Performance/Peak Power



The Applications

Characterization of Application (out-of-the-box)

Initial Estimates

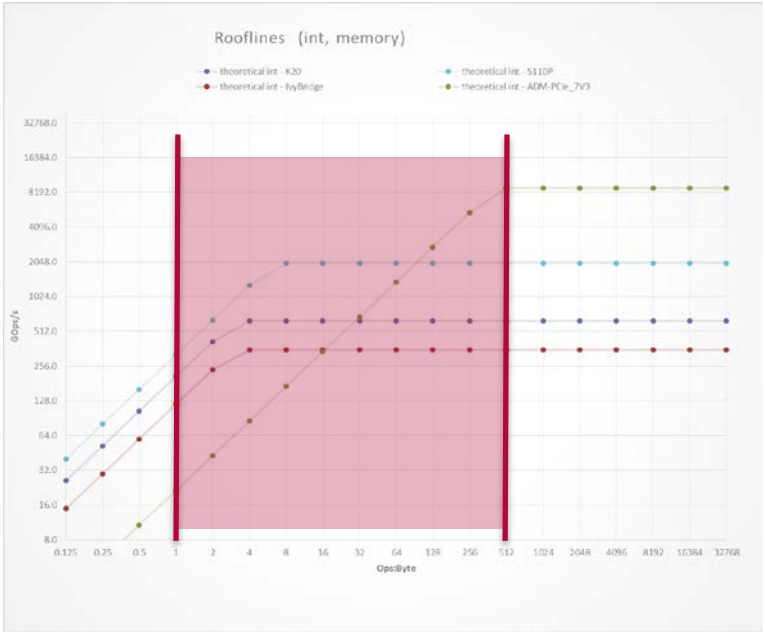
Application	Type	Operational Intensity (min)	Operational Intensity (max)
Video Scalar	Fixed point	Vtaps*Htaps*sh*sv/4 *(sv*sh+1)= 1 (1,1,1,1)	7*Vtaps*Htaps*sh*sv/4 *(sv*sh+1) = 446 (16,16,16,16)
BobJenkins Hash	Non-float	3.1	4.5
Memcached	Non-float	3.65	300
....			

Example 1: Polyphase Video Scaler

- **Generic Image Filtering Equation**

$$Pix_{out}[x, y] = \sum_{HTaps\_1}^{i=0} \sum_{VTaps\_1}^{j=0} Pix_{in}[x - (HTaps/2) + i, y - (VTaps/2) + j] \times Coef[i, j]$$

- **Theoretical Operational Intensity:**
  - OI=7*Vtaps*Htaps*sh*sv/4*(sv*sh+1)



- **Results**
  - Complexity independent of image size
  - Complexity increases with larger window and scaling factor
  - Well suited for FPGA acceleration

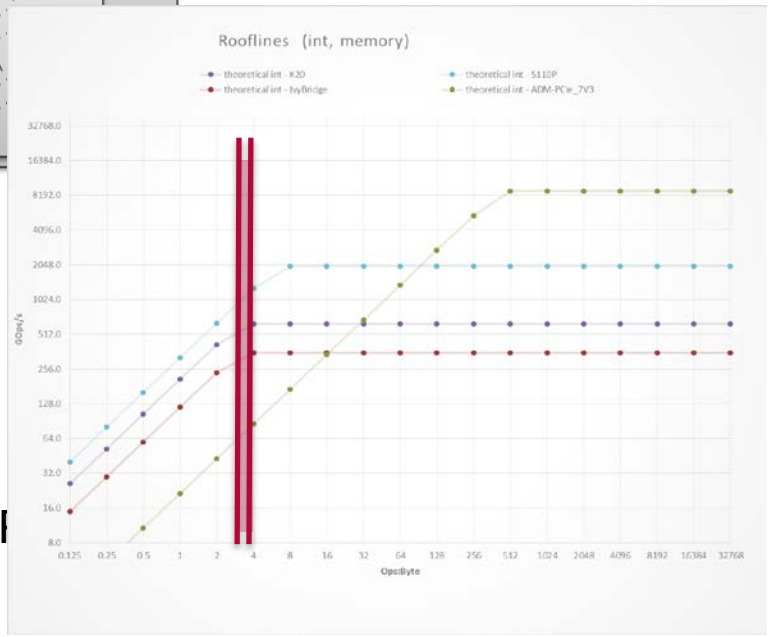
Sv,sh,vtaps, htaps	TOI
1,1,1,1	1
4,4,4,4	26
16,16,16,16	446
Beyond that, this doesn't matter …	

Example 2: Bob Jenkins Hash Algorithm

```
a=b=c=deadbeef + ((unint32_t)length<<2) + initval;

while (length>3) {
          a += k[0];
          b += k[1];
          c += k[2];
          mix(a,b,c);
          Length -= 3;
          k += 3;
}

switch (length) {
          case 3: c += k[2];
          case 2: b += k[1];
          case 1: a += k[0];
          case 0: break;
}

final (a,b,c);
return c;
```

```
#define mix(a,b,c) \ {\
   a -= c;  a ^= rot(c, 4);  c += b;\
   b -= a;  b ^= rot(a, 6);  a += c;\
   c -= b;  c ^= rot(b, 8);  b += a;\
   a -= c;  a ^= rot(c,16);  c += b;\
   b -= a;  b ^= rot(a,19);  a += c;\
   c -= b;  c ^= rot(b, 4);  b += a;\
}
```

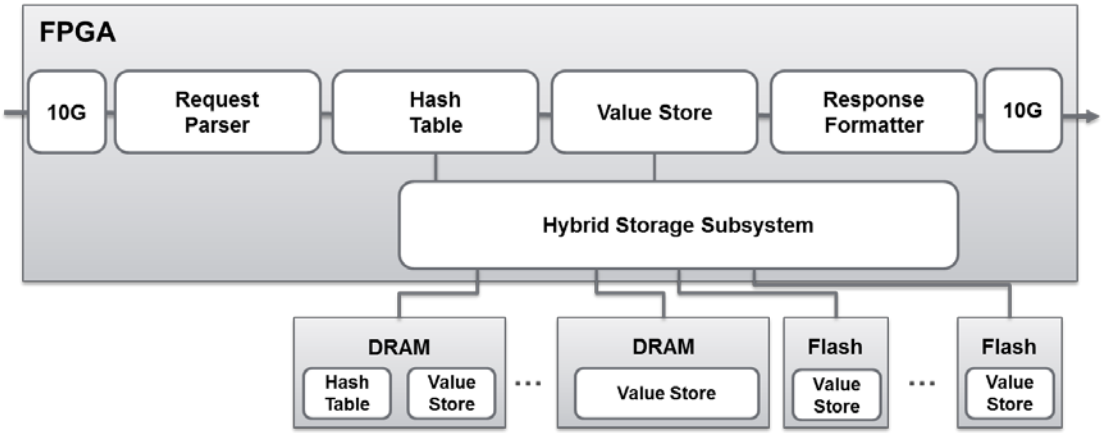
```
#define final(a,b,c) \ {\
          c ^= b; c -= rot(b,14);\
          a ^= c; a -= rot(c,11);\
          b ^= a; b -= rot(a,25);\
          c ^= b; c -= rot(b,16);\
          a ^= c; a -= rot(c,4);\
          b ^= a; b -= rot(a,14);\
          c ^= b; c -= rot(b,24);\
}
```



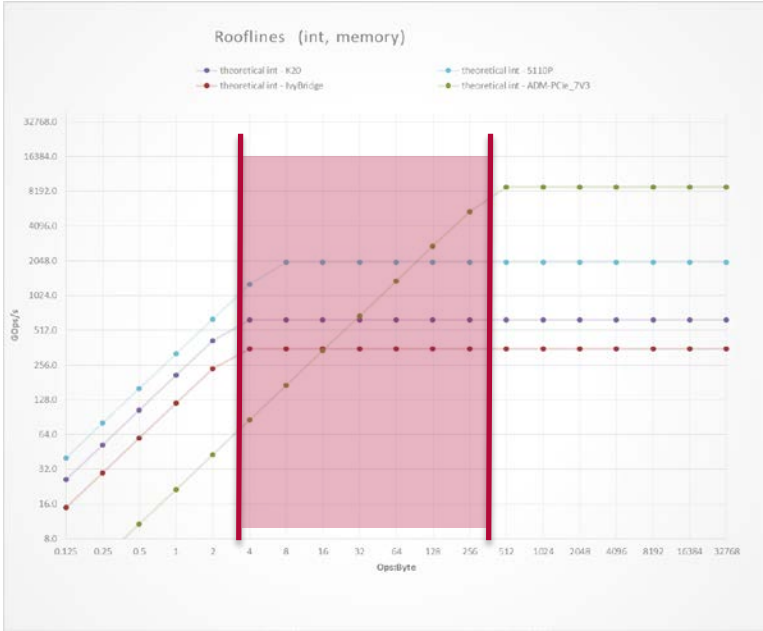
- **Theoretical Operational Intensity:**
  - OI = 4.5 (for maximum key size)
- **Results**
  - Memory bound => poor performance on F
  - Isolated function, not application

Example 3: Key-Value Stores

HotStorage 2015, Scaling out to a Single-Node 80Gbps Memcached Server with 40Terabytes of Memory



- **Streaming application**
- **Network interface, SSD interface**
- **Theoretical Operational Intensity: (per packet, GET operation)**
  - OI = [3.65, 300]
- **Results**
  - 80Gbps performance
  - 40TBytes



Observations

- **Preliminary results within constraint OpenCL1.2 execution model**
  - FPGAs excel at highly compute intensive applications with non-float dominating performance
  - Applications that cannot reach peak performance on other platforms
    - For example for applications with data-flow compute model, high branch conversion, asymmetric data types
  - Video scaler, CNN, Smith Waterman
- **Many other applications outside this model**
  - 80Gbps memcached with 40TBytes of object storage
  - 100Gbps+ network processing (NFV, DPI, IDS, traffic management)

Shortcomings & Next Steps

- **Operations are not directly comparable on different platform**
- **Integrating network and host interface bottlenecks and local memory characteristics into the roofline**
- **Fairness of power measurements**
- **Normalization**
- **Extension to other design entries**

Summary

- **New high-abstraction design flows are emerging for FPGAs**
  - Raise abstraction for accelerators and simplify host integration
  - In the future: run-time APIs, cloud API integration

- **Benchmarking diverse hardware architectures poses numerous challenges**
  - Community effort needed:

Heterogeneous High-performance Reconfigurable Computing - Workshop H²RC 2015

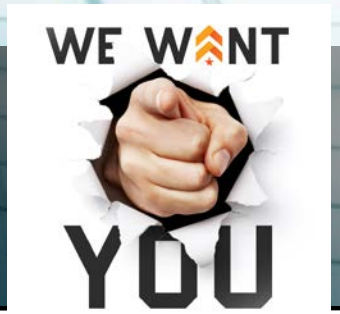
- **We propose a benchmarking methodology with cost functions to**
  - Quantify and measure work, performance, design productivity & complexity
  - Across a diverse range of hardware platforms
  - Characterization of applications across these platforms

XILINX
ALL PROGRAMMABLE™

ICHEC
Irish Centre for High-End Computing

Thank You!

mblott@Xilinx.com



We are seeking great research scientists at all levels!