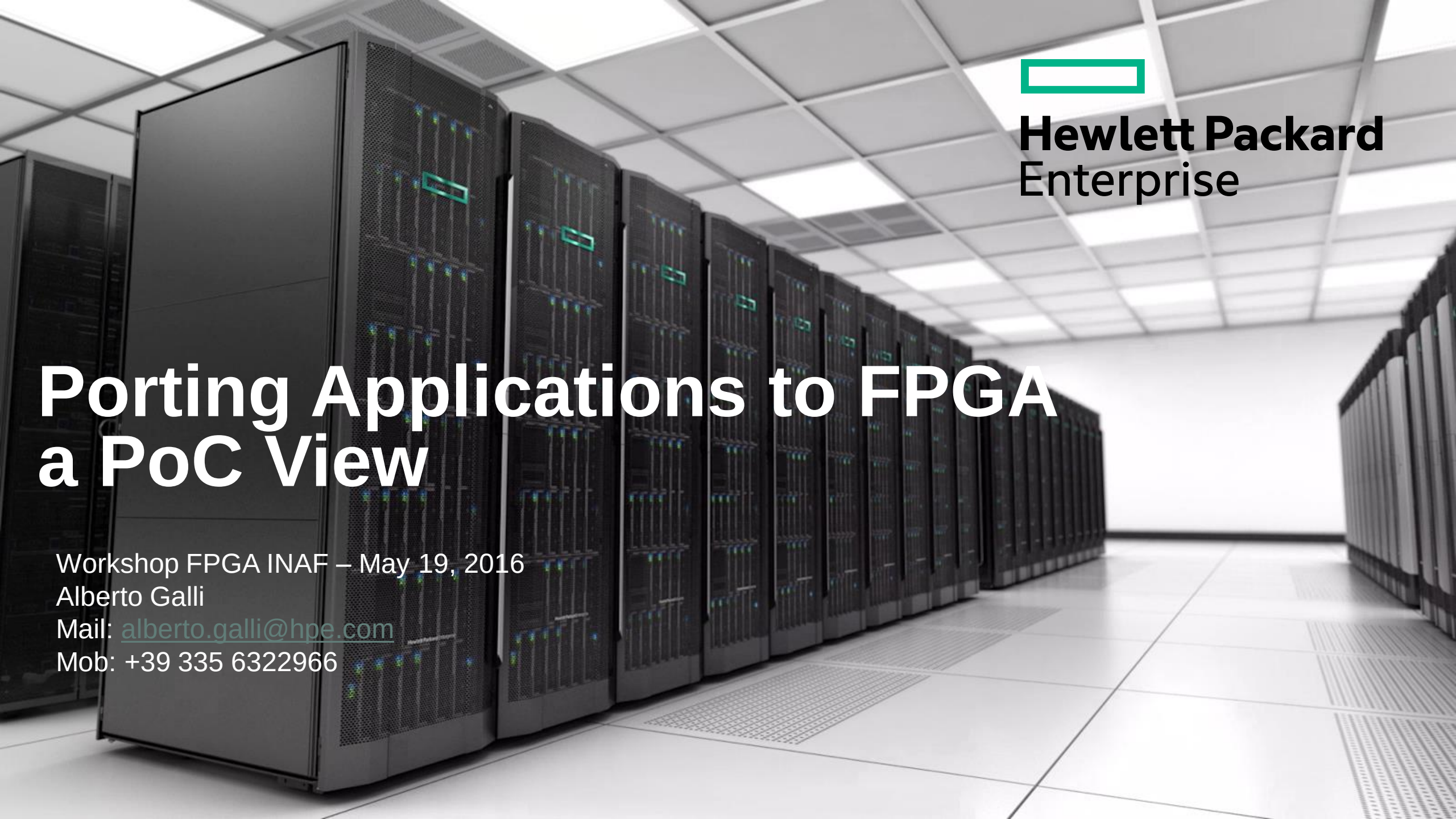


Hewlett Packard Enterprise

Porting Applications to FPGA a PoC View

Workshop FPGA INAF – May 19, 2016

Alberto Galli
Mail: alberto.galli@hpe.com
Mob: +39 335 6322966

Agenda

– Accelerators - FPGA the next hype?

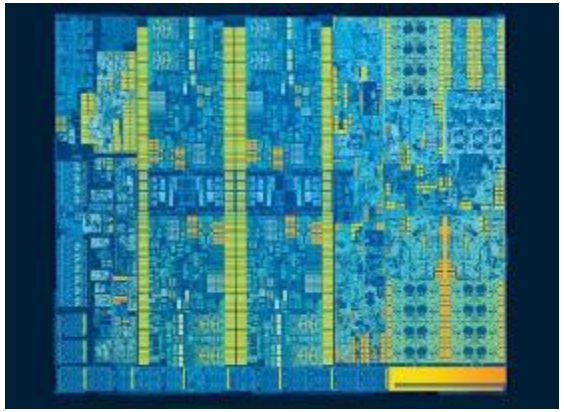
– OpenCL and FGPAs

–FPGA Porting

– Conclusions

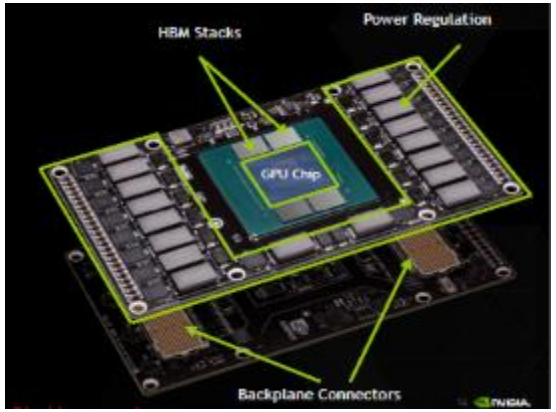
Accelerators

Acceleration Technology Progression



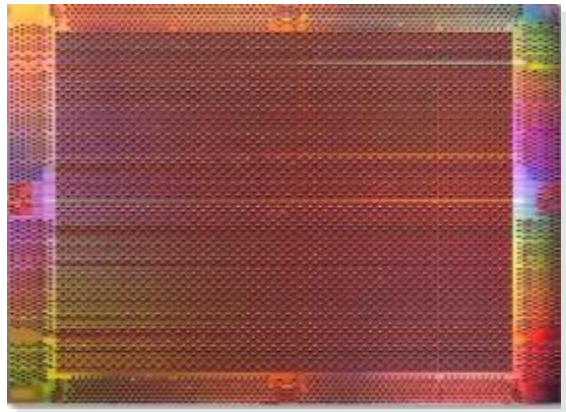
Multicore CPU
Many core (Phi)
X86 CPU

Software/Threads



GPGPU
Parallel Algorithms

CUDA/OpenCL



FPGA
Algorithms in
reconfigurable logic gates

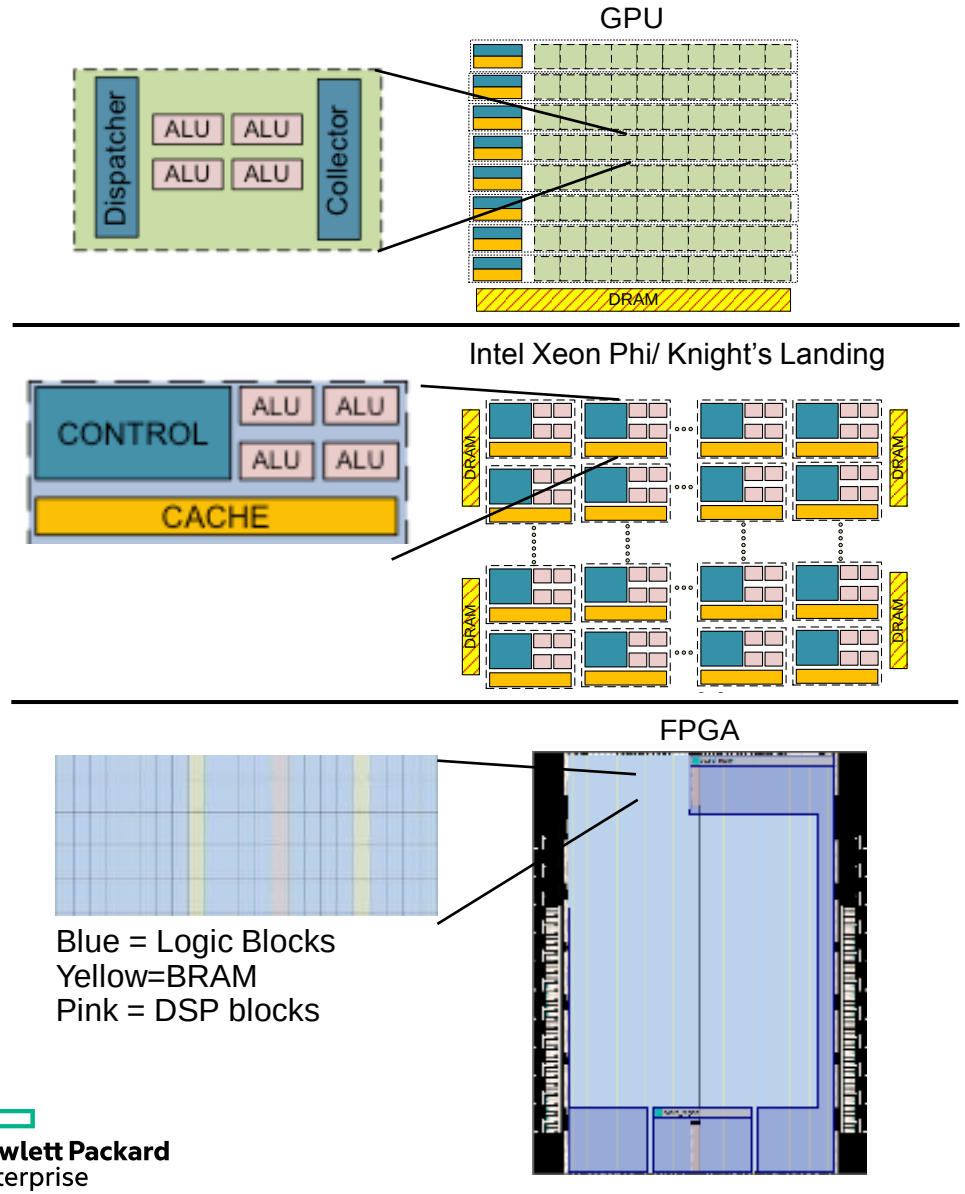
OpenCL/Verilog

SoC
Algorithms in dedicated
logic gates

Verilog/RTL

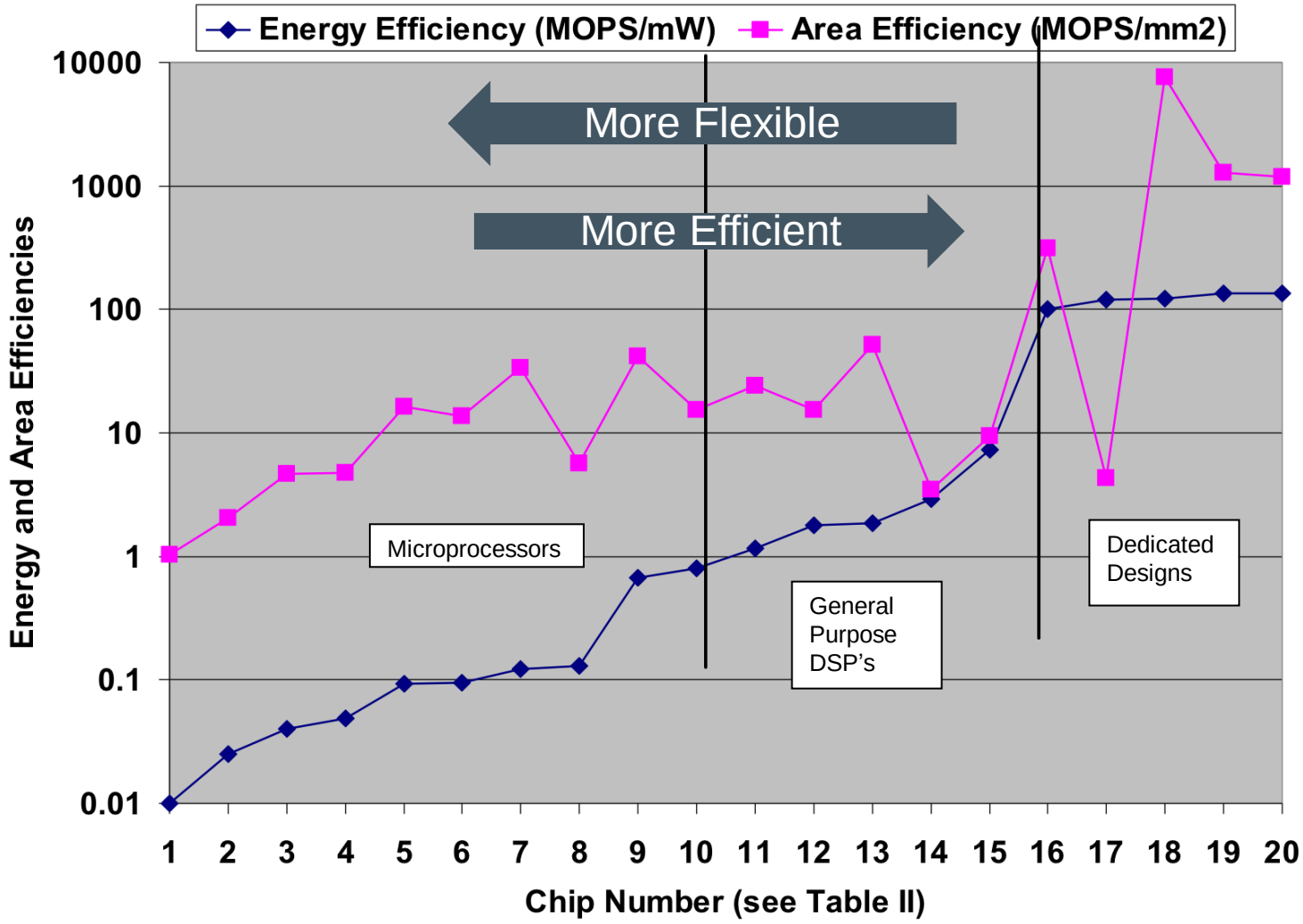
Hewlett Packard
Enterprise

Fundamental building blocks of (popular) accelerators



- GPU
  - Symmetric Multiprocessors (horizontal row) with cache and control logic
  - Each SM (shown in green) consists of a large number of Streaming Processors
  - SP's consist of ALUs, dispatcher, collector
- Xeon Phi/Many-core accelerators
  - Several x86 cores, Vector Processing Units
  - Interconnected in a mesh (NoC)
- FPGA
  - Light Blue, Yellow, Pink stripes are Logic blocks (LUT, Full-Adder, D Flip-Flop), Block RAM and hardened DSP blocks respectively.
  - From a programming perspective, a more basic **"blank-slate"**

Computing architectures



The Cost of Flexibility in Systems on a Chip Design, Zhang and Broderson

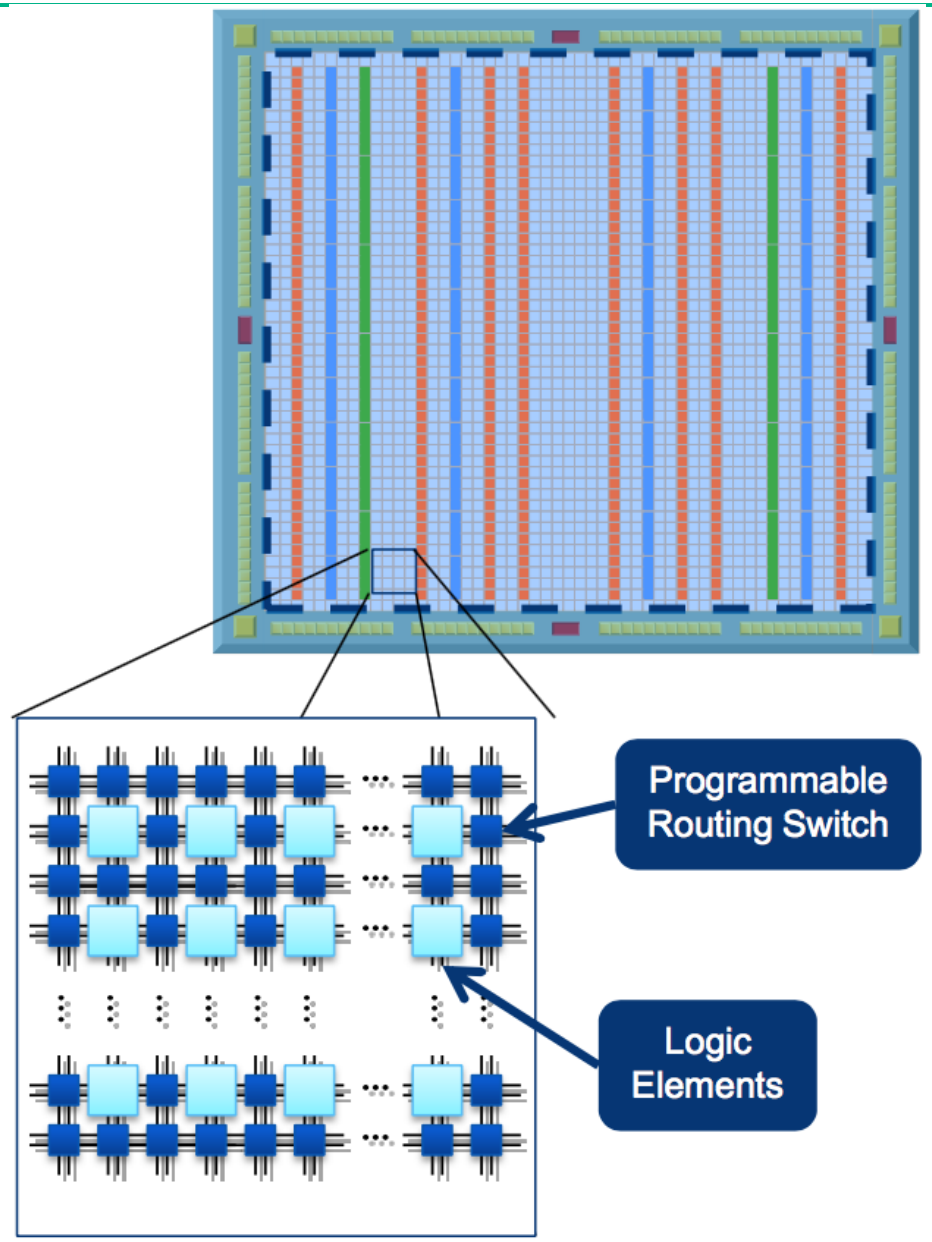
FPGA Architecture

Massive Parallelism

- Millions of logic elements
- Thousands of 20Kb memory block
- Thousands of Variable Presicion DSP blocks
- Dozens of High-Speed transcievers
- Various built-in hardened IP

FPGA Advantages

- Custom hardware
- Efficiency processing
- Low Power
- Able to reconfigure
- Fast time-to-market



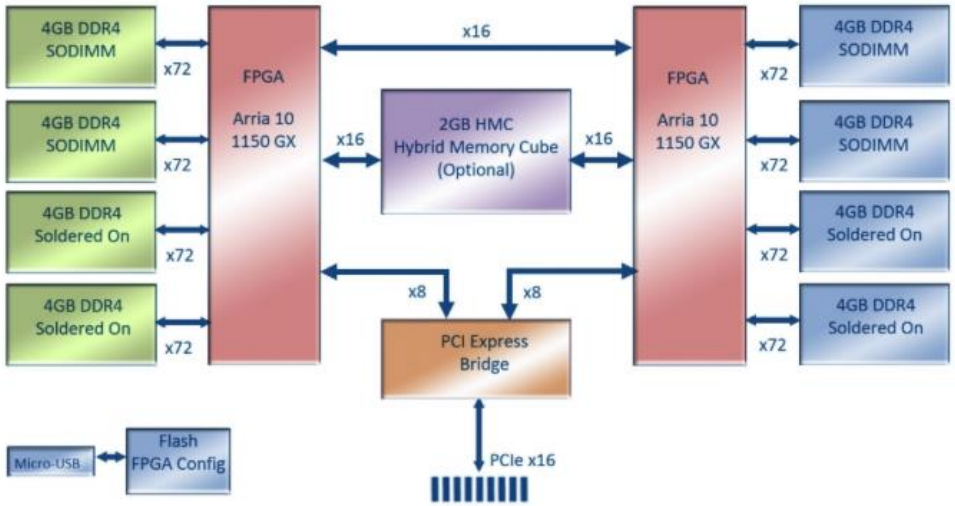
FPGAs (Altera)

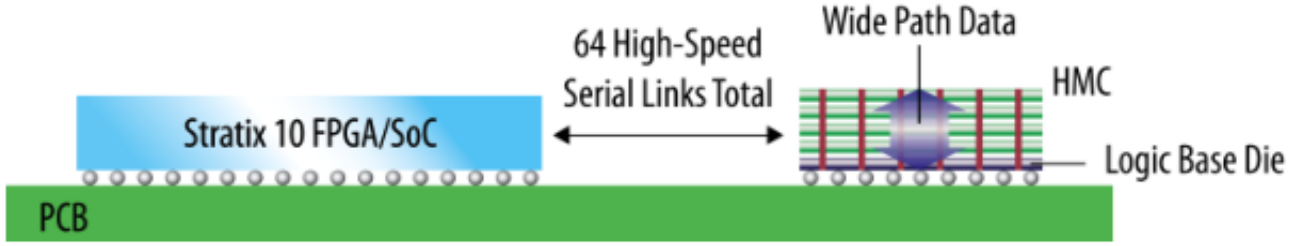
Arria 10

- 660k (SoC) to 1.15M (GX) elements
- Dual Arria 10 + HMC PCIe board
- OpenCL BSP available

Stratix 10

- 5.5M elements
- 3D System-in-Package integration
- Up to 10 TFLOPS (SP)
- High-end version of FPGA
- 330GB/s Memory Bandwidth
- Available End 2016/beginning 2017





Arria 10 Card

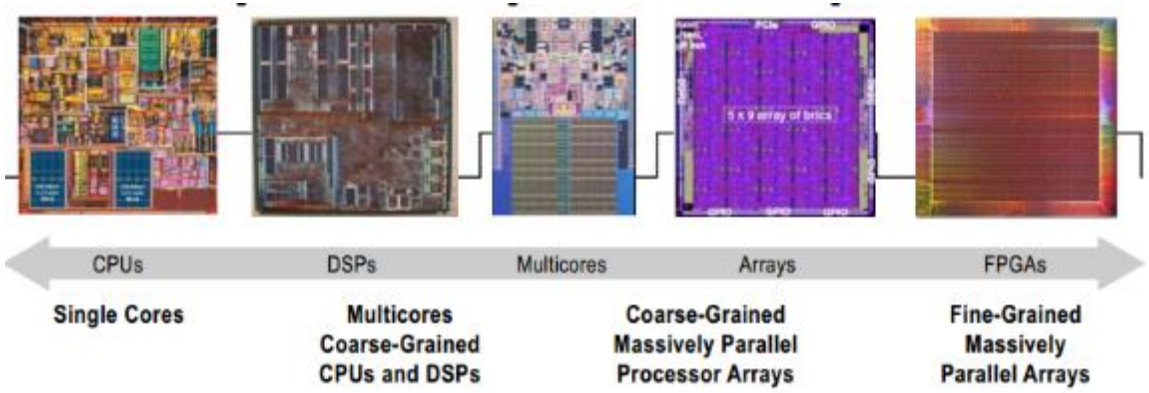
- NIC form factor
- (1) Arria 10 10A1150GX FPGA
  - Active and Passive cooling options
- PCIe Gen3 x 8
- (2) banks of DDR3
- (2) QSFP28 optical network ports for I/O processing and scaling
- OpenCL toolflow

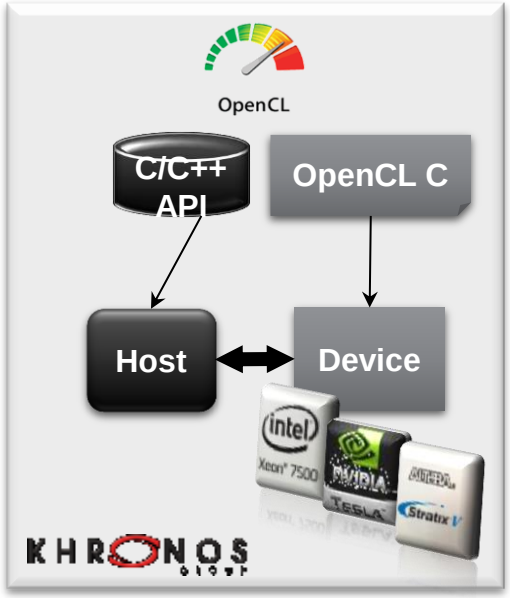
The 510T Accelerator

- GPU form factor
- (2) Arria 10 10A1150GX FPGAs
  - Active and Passive cooling options
- PCIe Gen3 x 16
- (8) banks of DDR4 (4 banks per FPGA)
- Hybrid Memory Cube (HMC) shared memory space between Arria 10 FPGAs
- OpenCL toolflow

OpenCL and FPGAs

OpenCL (Open Computing Language) Overview



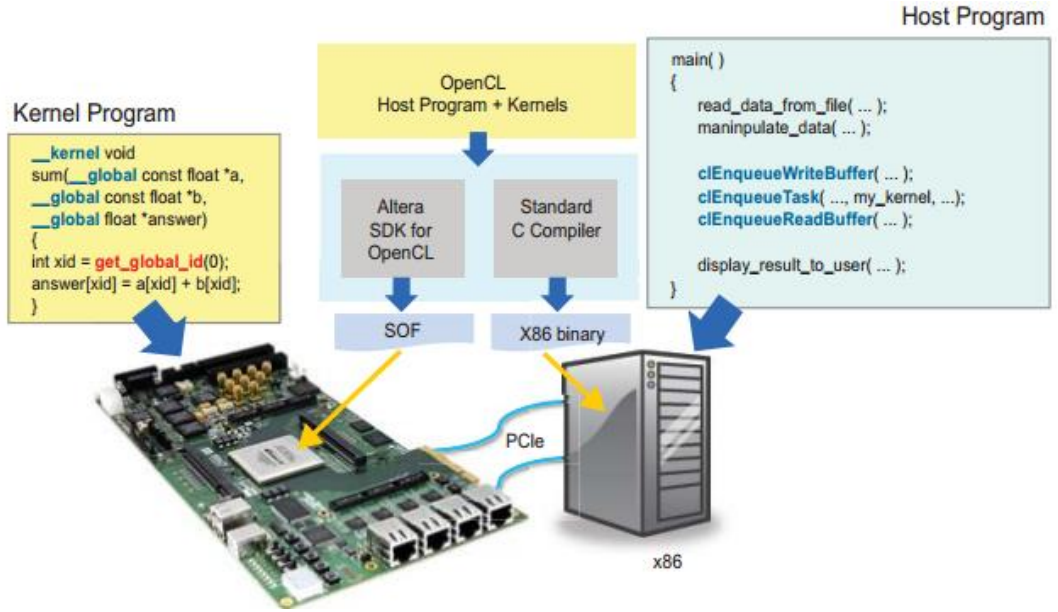


More information: http://www.khronos.org

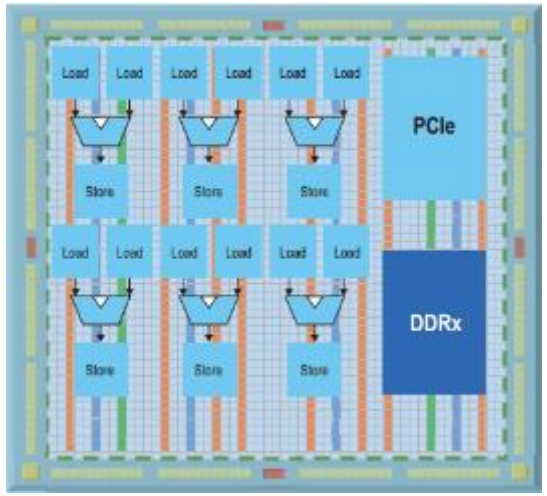
- Software programming model to operate across all devices
  - C/C++ API for host program with *extensions to specify parallelism*
  - OpenCL C for acceleration device
- Provides increased performance with hardware acceleration
  - CPU offload to appropriate accelerator
  - Local Memory
  - Explicit Parallelism
    - Task (SMT)
    - Data (SPMD)
- Portability
  - Existing code for CPU/GPU, DSP, FPGA will run on another
  - Does not mean the algorithm is necessarily optimal for that platform
- Higher level of abstraction than HDL
  - Competing tools to convert sequential programs to HDL

OpenCL on FPGAs

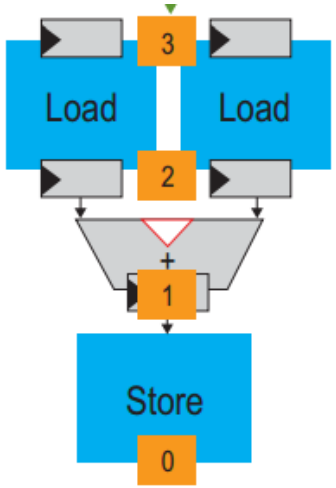
– The codes
  – Kernel: critical parallelizable piece of code
  – Host: buffer management, enqueuing, kernel dispatch
– **Critical** difference in programming principle
  – GPUs, DSPs naturally suited for **thread-level parallelism**
  – FPGAs inherently suited for **pipeline parallelism**
– Compilation, Emulation, Profiling
  – Kernel compilation involves AOC (script) which after converting to object file calls a complete co-design flow: (1) analysis & synthesis, (2) Fitter (Place and Route), (3) Generate programing file
  – Takes hours
– Programing
  – JTAG, Program to SPI, Configuration via Protocol[2]



OpenCL compilation process



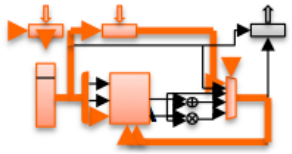
Thread-Level parallelism



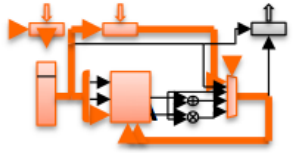
Pipeline parallelism

Just a simple example

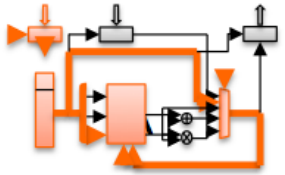
R0 ← Load Mem[100]



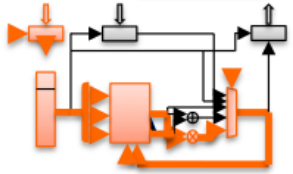
R1 ← Load Mem[101]



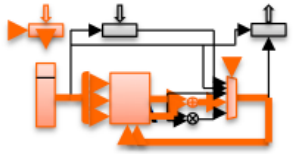
R2 ← Load #42



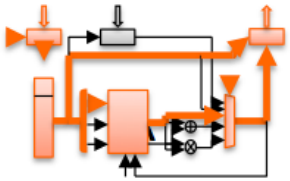
R2 ← Mul R1, R2



R0 ← Add R2, R0

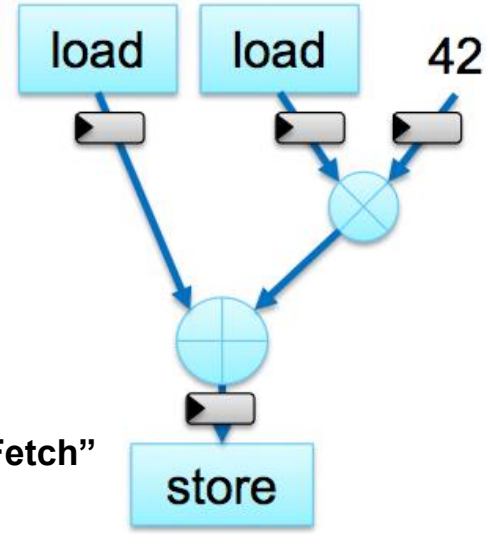


Store R0 → Mem[100]



High-level code

```
Mem[100] += 42 * Mem[101]
```



1. **Instructions are fixed. Remove instruction “Fetch”**
2. **Remove unused ALU ops**
3. **Remove unused Load/Store**
4. **Wire up registers properly! And propagate state.**
5. **Remove dead data.**

The PoC

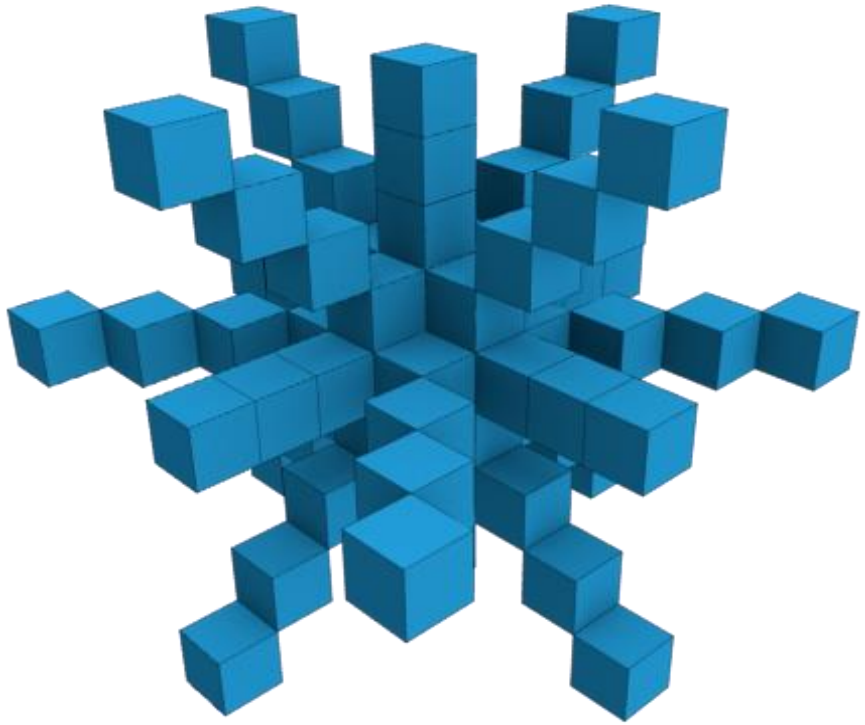

DILBERT
BY SCOTT ADAMS
I NEED YOU TO JOIN ME ON A SALES CALL TO TELL MY CUSTOMER HOW EASY IT WILL BE TO SWITCH TO OUR SOFTWARE.
IT ISN'T EASY.
THIS IS A SALES CALL. ALL YOU NEED TO DO IS SAY EVERYTHING WILL BE EASY.
WHAT HAPPENS WHEN THEY FIND OUT IT ISN'T EASY?
THEY WON'T FIND OUT UNTIL AFTER THEY PAY US.
WHAT WILL YOU DO WHEN THEY COMPLAIN?
I'LL TELL YOUR BOSS YOU MISLED THEM.
NOT IF I WARN HIM FIRST!
TOO LATE. I ALREADY TOLD HIM YOU'RE A LIAR.
DilbertCartoonist@gmail.com
© 2015 Scott Adams, Inc. /Dist. by UNIVERSAL UCLICK
www.dilbert.com
2-27-15

The FPGA Porting

POC Problem Size

volume ~ 400x400x400

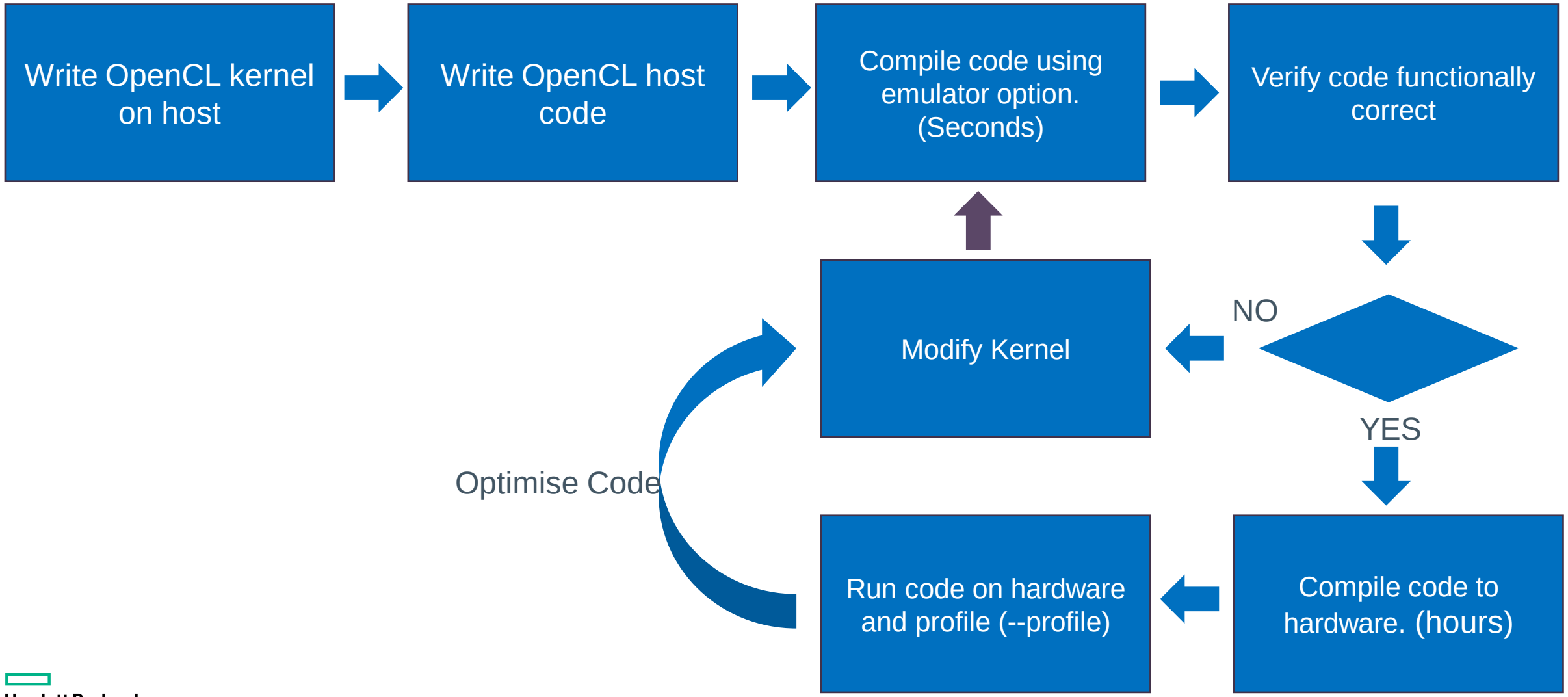
- Three input buffsets P, Q and Constant data
  - P & Q buffer input ~ 0.5 GBytes
  - Constants (6 floats) ~ 1.5 GBytes
  - Total input 2.0 GBytes.
- Output P & Q
  - 0.5 GBytes
- Total memory = 2.5 – 3.0 Gbytes



OpenCL FPGA implementation

- **Key modifications**
  - Create pipelined implementation of code using sliding windows to minimise global memory bandwidth.
    - Sliding window allows full 9x9x9 stencil data to be stored locally in FPGA block memory.
  - Volume is sub divided due to limited block memory. (Can’t create sliding window for larger volumes).
- **SIMD**
  - Process 4 cells in parallel (Global memory bandwidth limit)

Tool Flow


Write OpenCL kernel on host
Write OpenCL host code
Compile code using emulator option. (Seconds)
Verify code functionally correct
NO
Modify Kernel
YES
Optimise Code
Run code on hardware and profile (--profile)
Compile code to hardware. (hours)

What we have obtained - Performance

- 8 elements per FPGA each processing 200 floating point operations
- Kernel clock 200 MHz
- Efficiency of algorithm 80%
- Bandwidth of 510T 68GBytes/sec/FPGA
- Sustained Flops = 512 GFlops/Sec
- **MElem Per FPGA = 1280**
- **MElem Per 510T= 2560**

FPGAs – Final Comparison (power included)

- FPGA power consumption Nallatech 510T
  - DDR4 memory banks 4 per FPGA
  - PCIe switch
  - 150 Watts
  - In the Table Below the power/rack refers to a rack with 40 servers with E5-280v3 (2x120W TDP) each equipped with 2 GPUs/FPGAs

CARD	510T MEASURED	510T ESTIMATION	K80	M60
TDP of 1 card (W)	150	Around 200	300	300
Cards/Rack	80	80	80	80
Power/RACK(W)	21600	25600	33600	33600
Performance per Card	2560	2980	2004	1416
Perf 4 cards	10240	11928	7803	5331
Performance/Rack	204800	238560	156079	106627
Performance/Power	9481	9318	4645	3173

Conclusions

Conclusions

- When it comes to HPC, the accelerator market is largely dominated by GPUs, which are known for floating point performance with encroaching competition from Xeon Phi.
- FPGAs, while still a third-place contender, might have a rather remarkable year ahead in terms of their reach into wider markets
- As any technology is ready to reach and pass all **the 3 Ps** .

Price.Performance.Programability.

- While a great deal of the more recent adoption of FPGAs has been centered on hyperscale and commercial work, there are still hot areas in HPC that are set to benefit from the expanded ecosystem around the accelerators

Hewlett Packard
Enterprise

Thank you