OpenCL: FPGA Acceleration for Software Programmers

Marco Mauri

© 2016 Altera—Confidential

Agenda

- FPGA Overview for software programmers
- Technology trends & challanges
  - Why FPGA & OpenCL?
- OpenCL and Altera SDK Overview

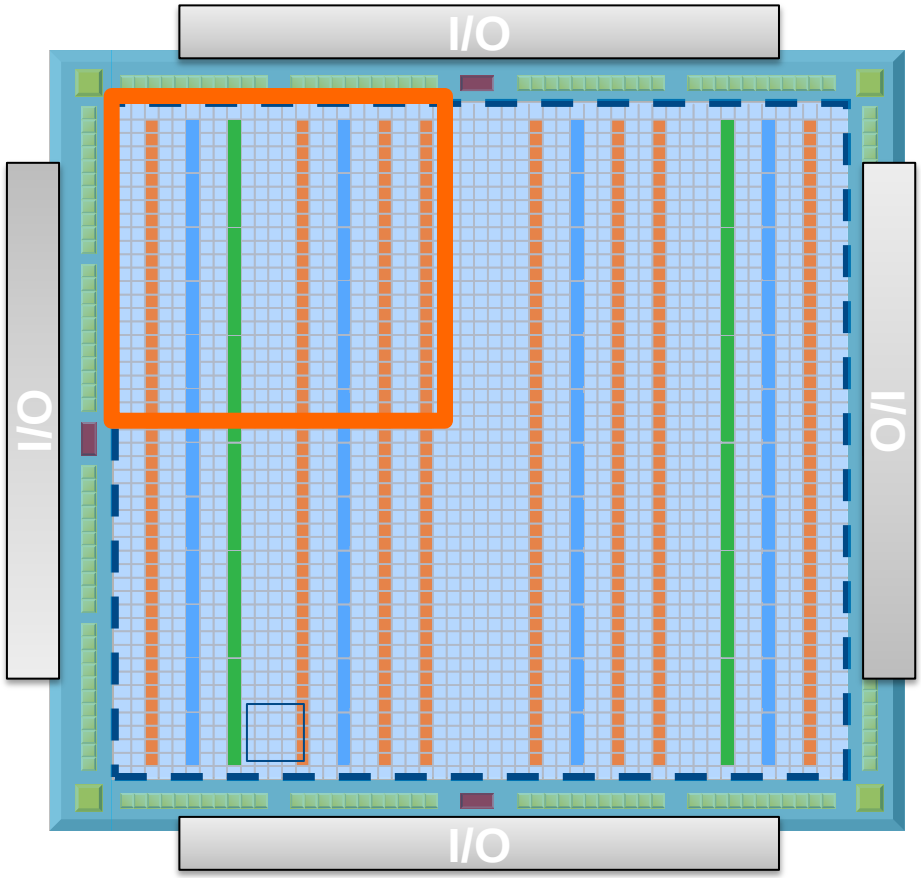
© 2016 Altera—**Confidential**
ALTERA
now part of Intel

FPGA Overview for Software Programmers

© 2016 Altera—Confidential

FPGA Architecture: Fine-grained Massively Parallel

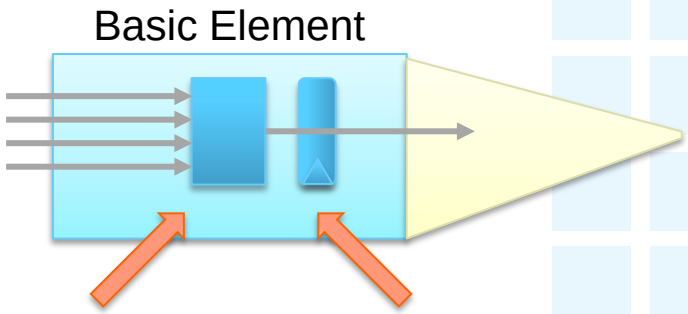
Let's zoom in



- 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

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

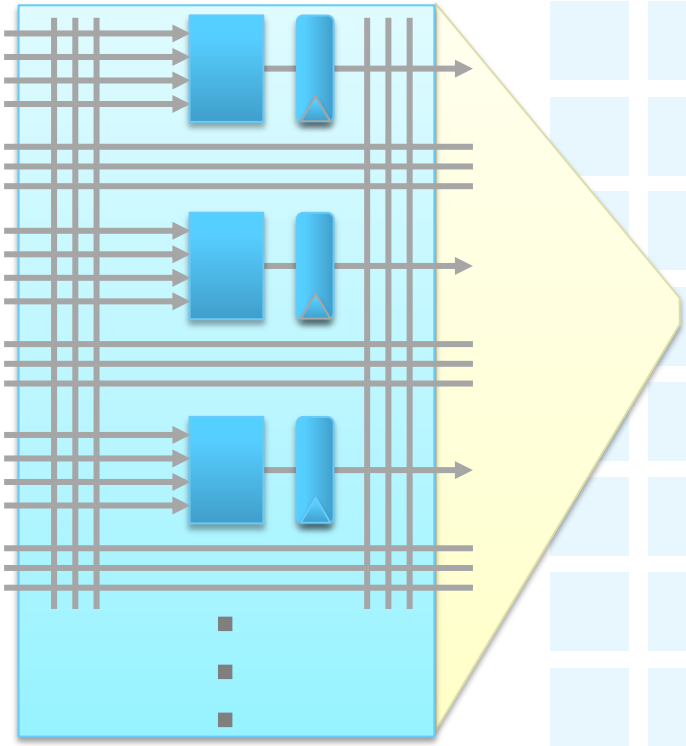
FPGA Architecture: Basic Elements



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

© 2016 Altera—**Confidential**
ALTERA now part of Intel

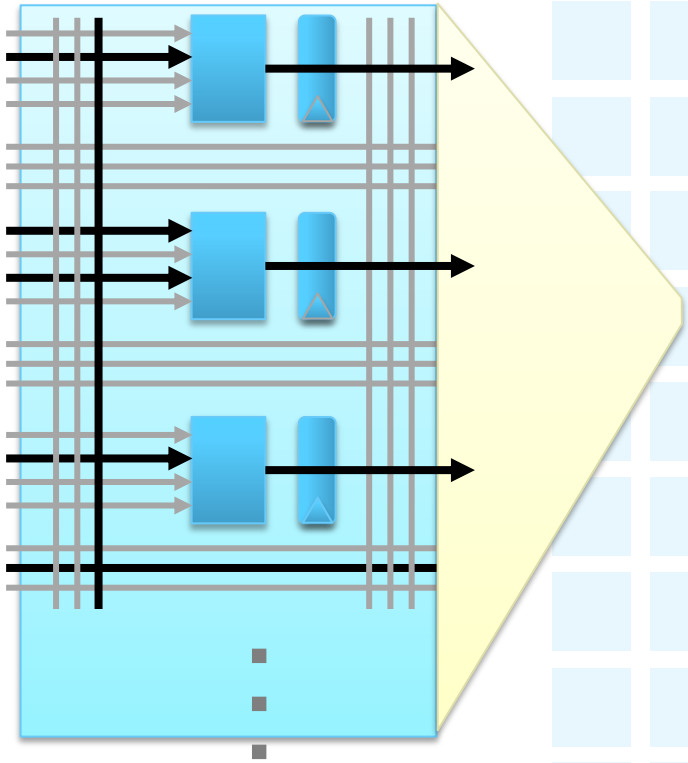
FPGA Architecture: Flexible Interconnect



Basic Elements are surrounded with a flexible interconnect

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

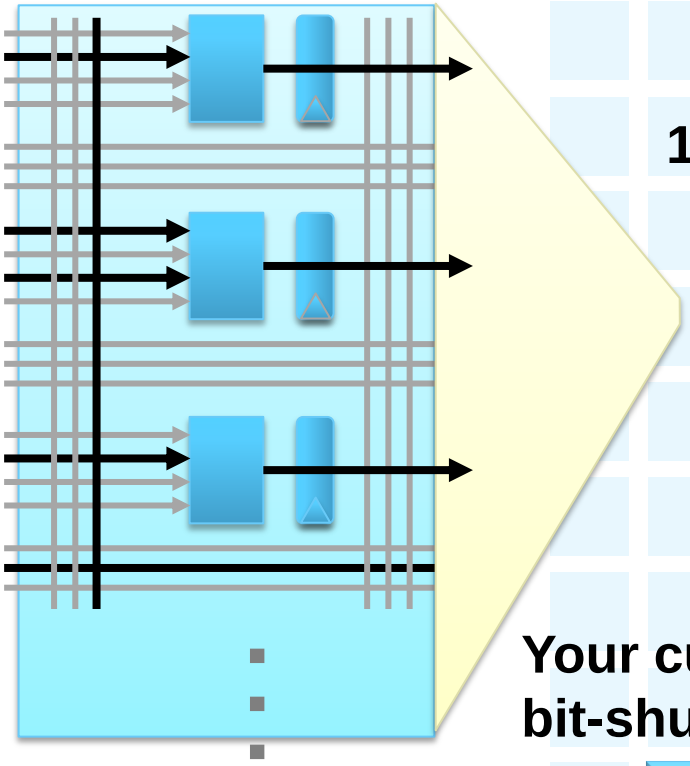
FPGA Architecture: Flexible Interconnect



Wider *custom* operations are implemented by configuring and interconnecting Basic Elements

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

FPGA Architecture: Custom Operations Using Basic Elements



16-bit add

32-bit sqrt

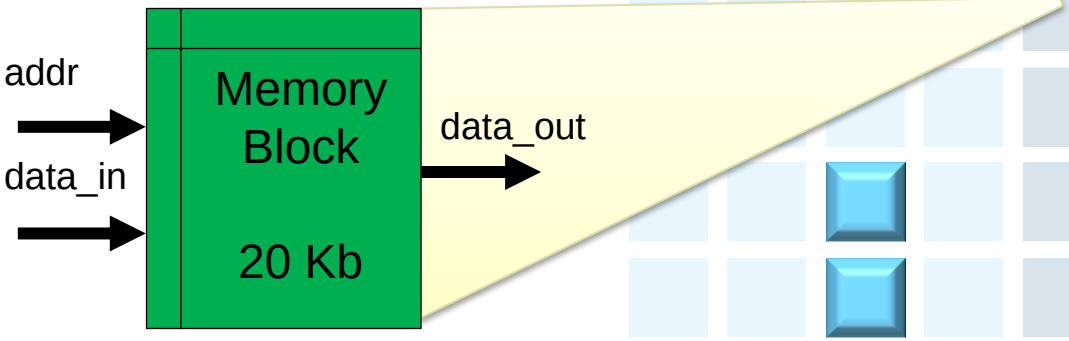
Your custom 64-bit bit-shuffle and encode

Wider *custom* operations are implemented by configuring and interconnecting Basic Elements

© 2016 Altera—Confidential

ALTERA now part of Intel

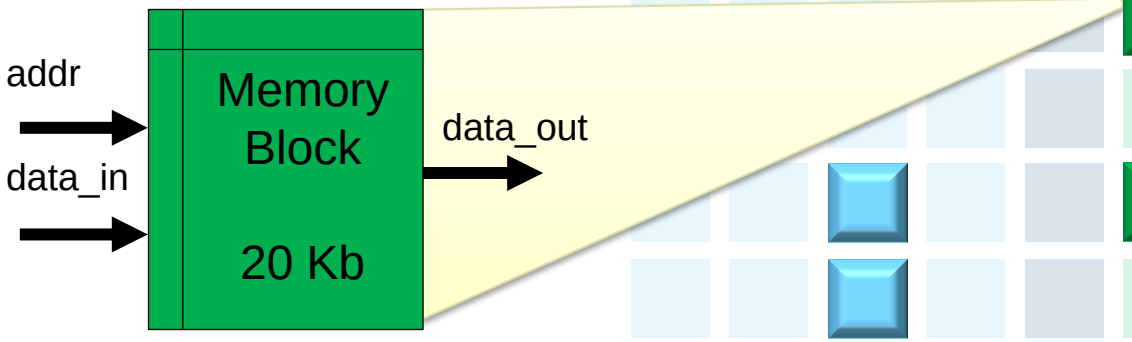
FPGA Architecture: Memory Blocks



Can be configured and grouped using the interconnect to create various cache architectures

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

FPGA Architecture: Memory Blocks



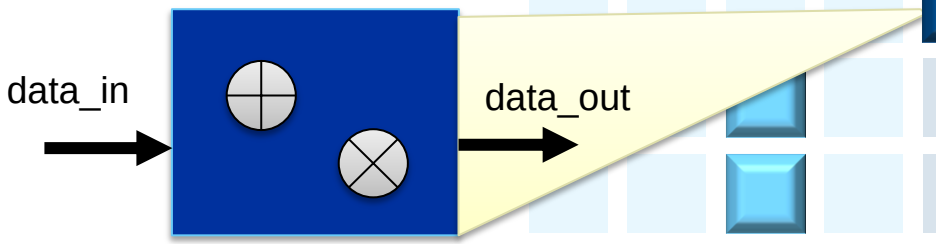
Can be configured and grouped using the interconnect to create various cache architectures

Lots of smaller caches

Few larger caches

© 2016 Altera—**Confidential**
ALTERA now part of Intel

FPGA Architecture: Floating Point Multiplier/Adder Blocks

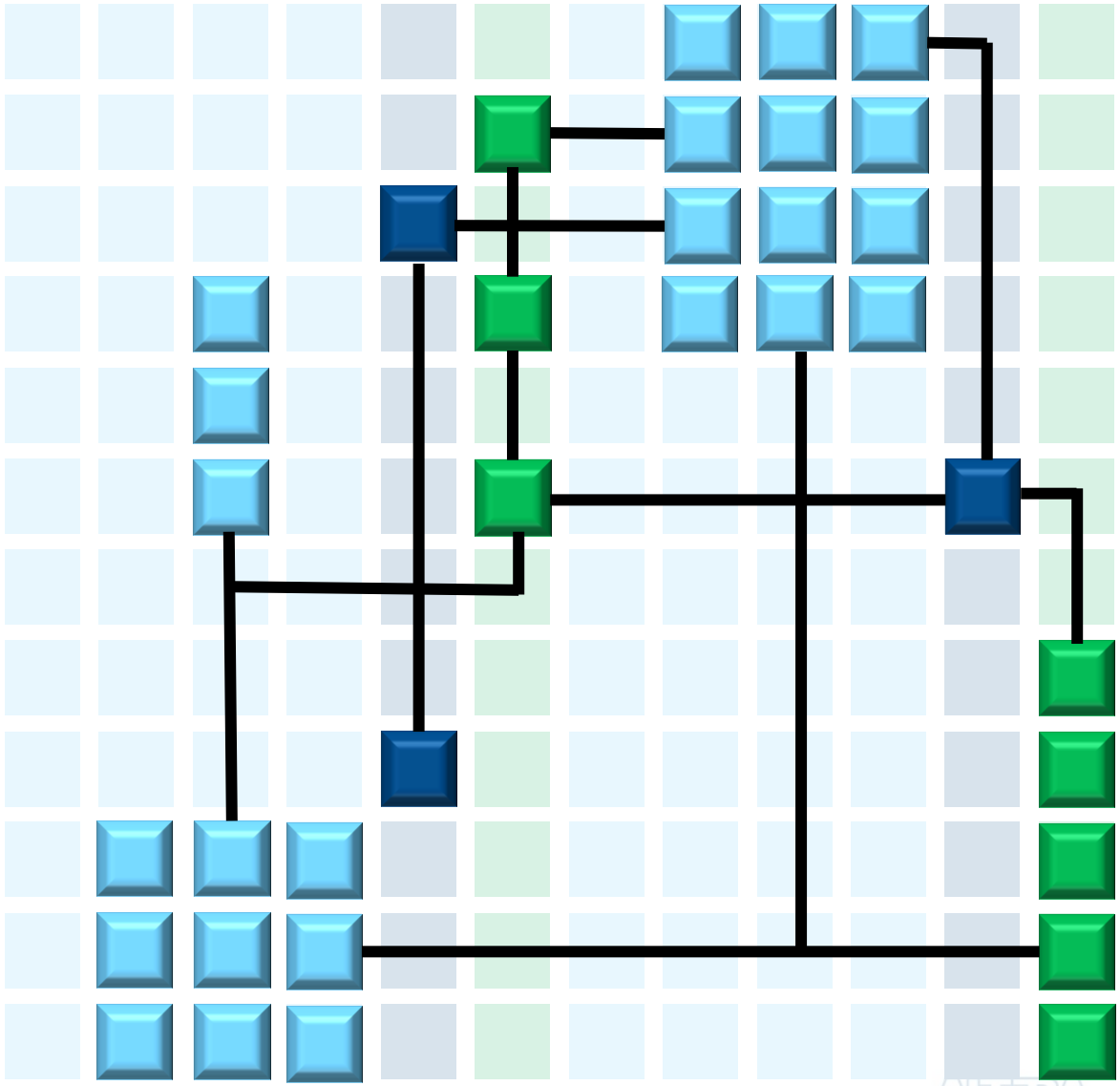


Dedicated floating point multiply and add blocks

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

FPGA Architecture: Configurable Routing

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



© 2016 Altera—**Confidential**
ALTERA
now part of Intel

Mapping a simple program to an FPGA

High-level code

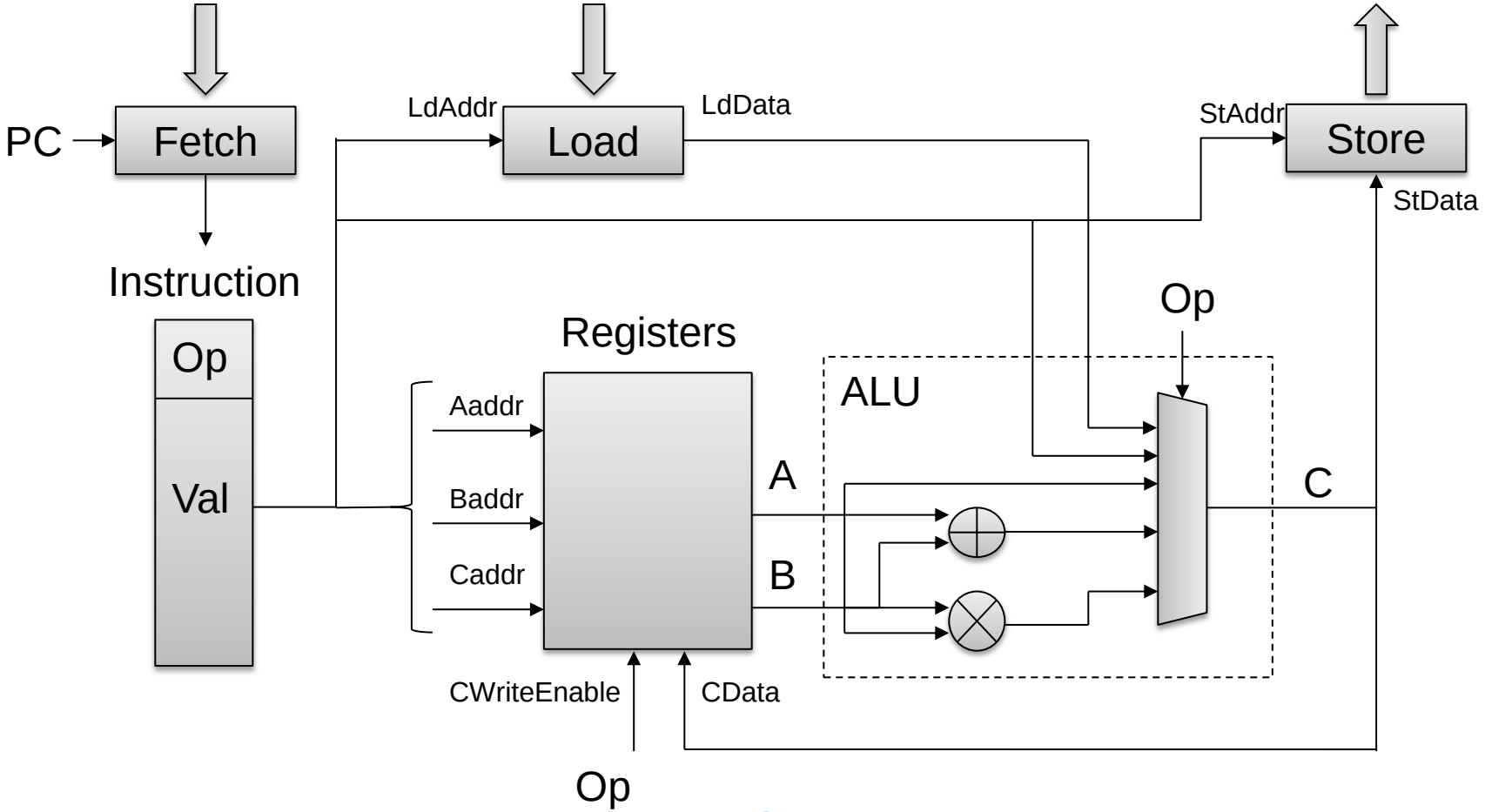
```
Mem[100] += 42 * Mem[101]
```

CPU instructions

```
R0 ← Load Mem[100]
R1 ← Load Mem[101]
R2 ← Load #42
R2 ← Mul R1, R2
R0 ← Add R2, R0
Store R0 → Mem[100]
```


© 2016 Altera—Confidential
ALTERA now part of Intel

First let's take a look at execution on a *simple* CPU

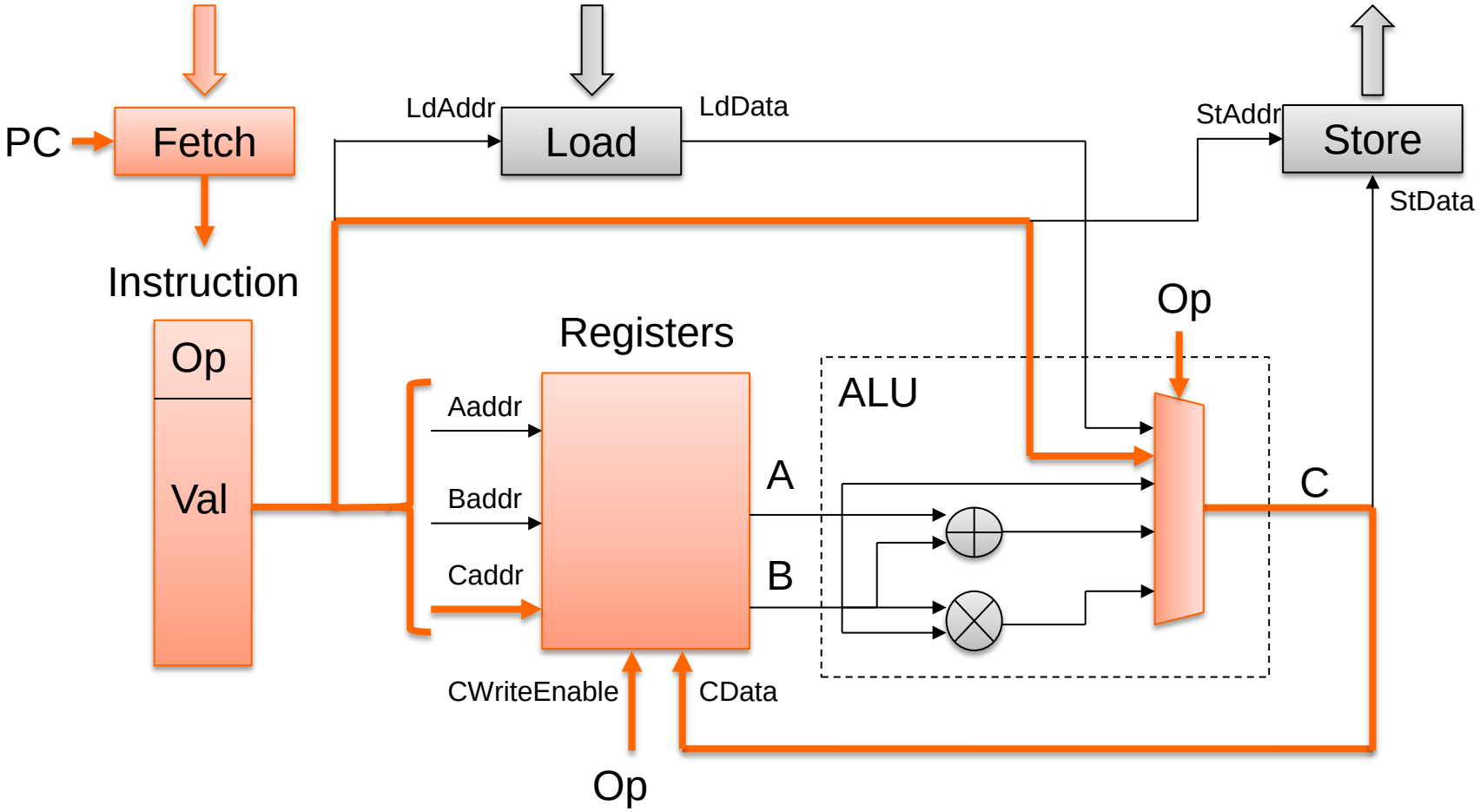


Fixed and general architecture:

- **General "cover-all-cases" data-paths**
- **Fixed data-widths**
- **Fixed operations**

© 2016 Altera—**Confidential**
ALTERA now part of Intel

Load constant value into register



Very inefficient use of hardware!

© 2016 Altera—Confidential
ALTERA now part of Intel

CPU activity, step by step

R0 ← Load Mem[100]

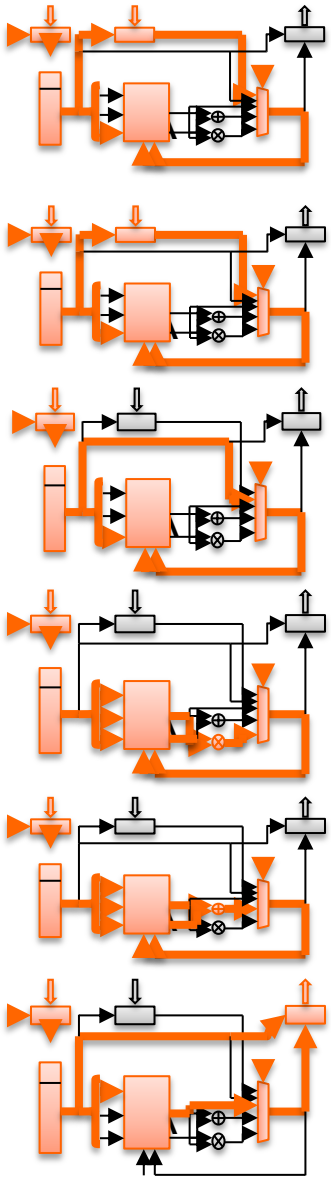
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



Time

© 2016 Altera—Confidential

ALTERA now part of Intel

On the FPGA we unroll the CPU hardware…

R0 ← Load Mem[100]

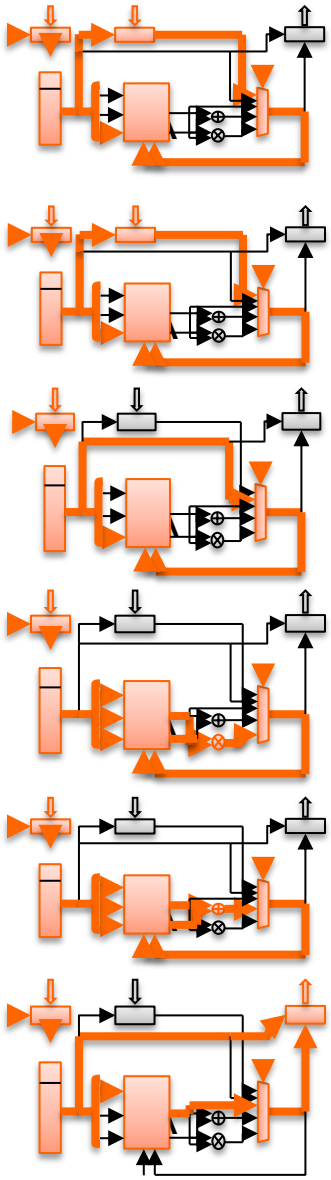
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



Space

© 2016 Altera—Confidential

ALTERA now part of Intel

… and specialize by position

R0 ← Load Mem[100]

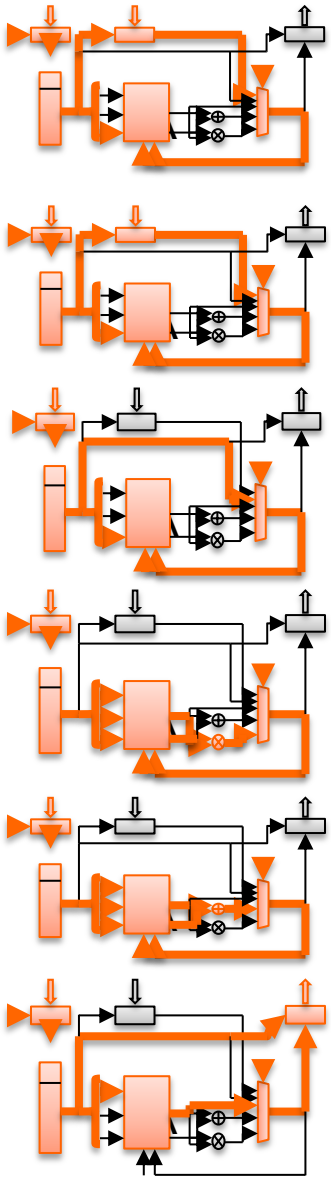
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



1. Instructions are fixed. Remove “Fetch”

© 2016 Altera—Confidential
ALTERA now part of Intel

… and specialize

R0 ← Load Mem[100]

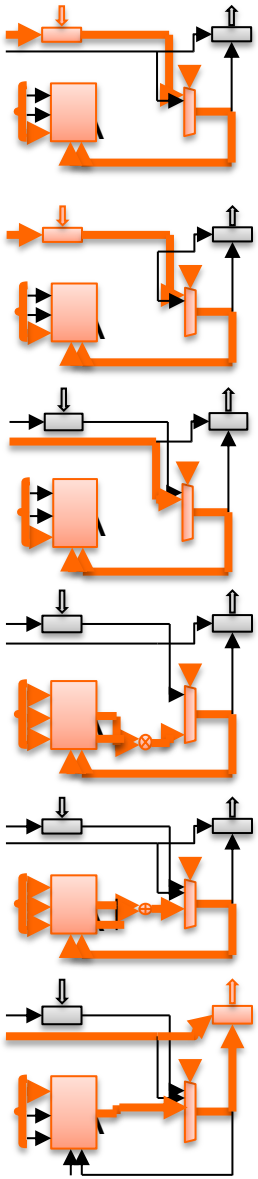
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]

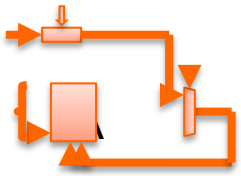


1. Instructions are fixed. Remove “Fetch”
2. Remove unused ALU ops

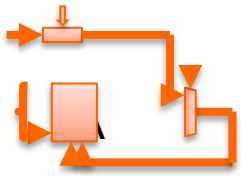
© 2016 Altera—Confidential
ALTERA now part of Intel

… and specialize

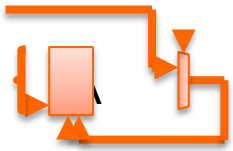
R0 ← Load Mem[100]



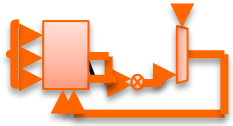
R1 ← Load Mem[101]



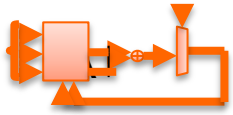
R2 ← Load #42



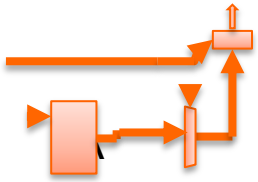
R2 ← Mul R1, R2



R0 ← Add R2, R0



Store R0 → Mem[100]



1. Instructions are fixed. Remove “Fetch”
2. Remove unused ALU ops
3. Remove unused Load / Store

© 2016 Altera—Confidential
ALTERA now part of Intel

… and specialize

R0 ← Load Mem[100]

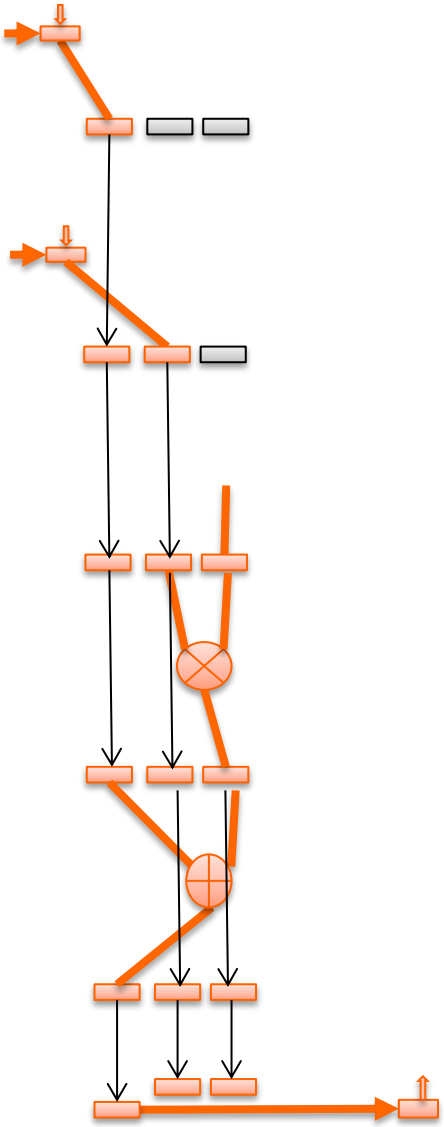
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



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

© 2016 Altera—Confidential
ALTERA
now part of Intel

… and specialize

R0 ← Load Mem[100]

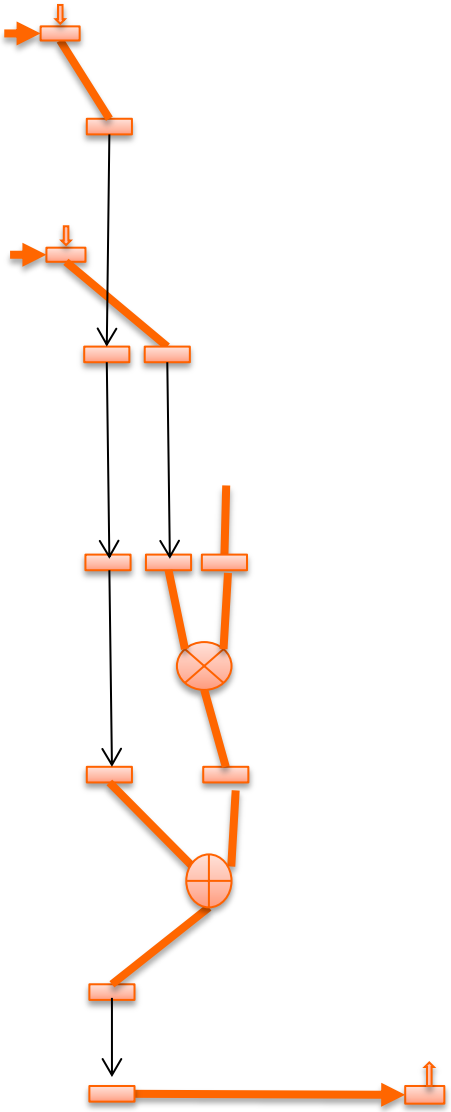
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



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

© 2016 Altera—Confidential
ALTERA
now part of Intel

… and specialize

R0 ← Load Mem[100]

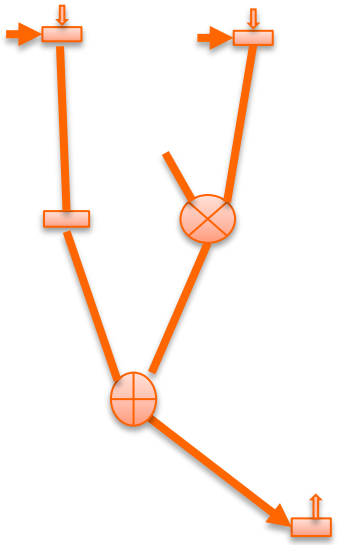
R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]



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

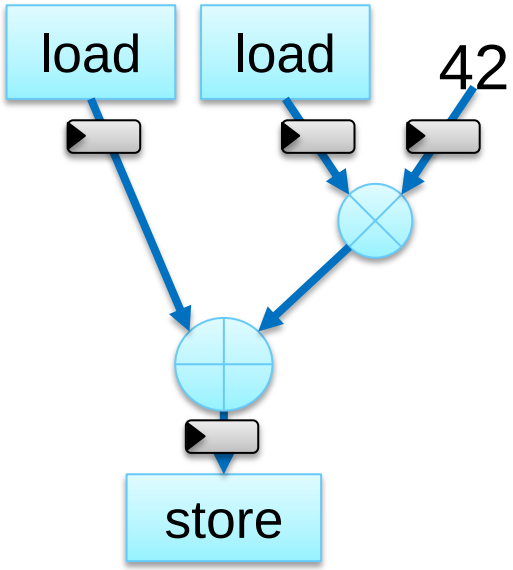
© 2016 Altera—Confidential
ALTERA
now part of Intel

Custom data-path on the FPGA matches your algorithm!

High-level code

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

Custom data-path



Build exactly what you need:

Operations

Data widths

Memory size & configuration

Efficiency:

Throughput / Latency / Power

© 2016 Altera—**Confidential**
ALTERA now part of Intel

Technology Trends & challanges

© 2016 Altera—Confidential

High Level Market Trends

- Application complexity and compute requirements is ever increasing
  - Larger FPGAs (5.5 Million LEs) require longer development times for FPGA developers
  - Software development resources are growing while hardware engineers are diminishing
  - Faster to develop in software and accelerate in hardware
- Distributed computing required to solve growing data requirements
  - SoC and embedded computing is becoming common place in the industry

© 2016 Altera—**Confidential**
ALTERA
now part of Intel

General Challenges

- Distributed computing challenged by data transfers among the nodes and datacenter
  - infrastructure / data center bottleneck
- Maintaining coherency throughout scalable system
- Overloading CPUs capabilities
  - Frequencies are capped
  - Processors keep adding more cores
  - Need to coordinate all the cores and manage data
- Product life cycles are long
  - SW Updates to existing HW
  - GPUs lifespan is short
    - Require re-optimization and regression testing between generations
- Power dissipation of CPUs and GPUs limits system size

© 2016 Altera—Confidential
ALTERA now part of Intel

OpenCL and FPGAs Address Markets Challenges

- Power efficient acceleration
  - Typically 1/5 power of GPU and orders of magnitude more performance per watt of CPU
- FPGA lifecycle over 15 years
  - GPUs lifespan is short
    - Require re-optimization testing between generations
  - FPGA OpenCL code retargeted to future devices without modification
- Our OpenCL flow abstracts away FPGA hardware flow
  - Puts FPGA into software engineers hands
- Shared virtual memory to keep data coherency among kernels
  - IBM CAPI and Intel QPI
- Our OpenCL SDK allows for streaming IO channels and kernel channels
  - Data movement without host involvement
  - Low latency data transmissions to accelerator

© 2016 Altera—Confidential
ALTERA now part of Intel

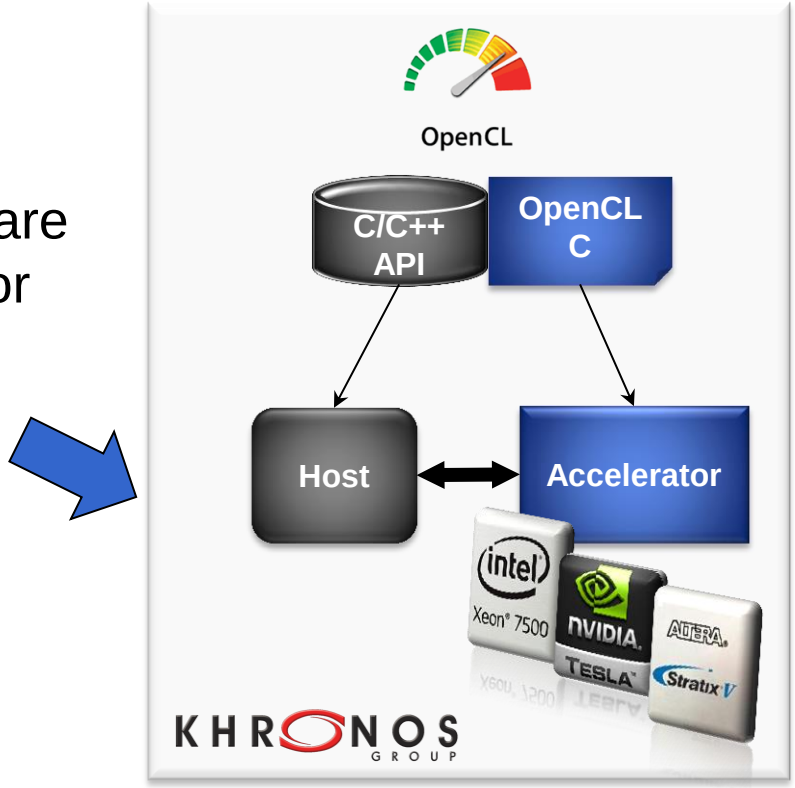
OpenCL and Altera SDK

Overview

© 2016 Altera—Confidential

What is OpenCL?

- A software programming model for software engineers and a software methodology for system architects
  - First industry standard for heterogeneous computing



- Provides increased performance with hardware acceleration
  - Low Level Programming language
  - Based on ANSI C

- Open, royalty-free, standard
  - Managed by Khronos Group
  - Altera active member
  - Conformance requirements
    - V1.0 is current reference
    - V2.0 is current release
  - http://www.khronos.org

© 2016 Altera—Confidential

OpenCL Use Model: Abstracting the FPGA away

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];
}
```

Host

Standard gcc Compiler

Altera Offline Compiler

EXE

AOCX

Verilog

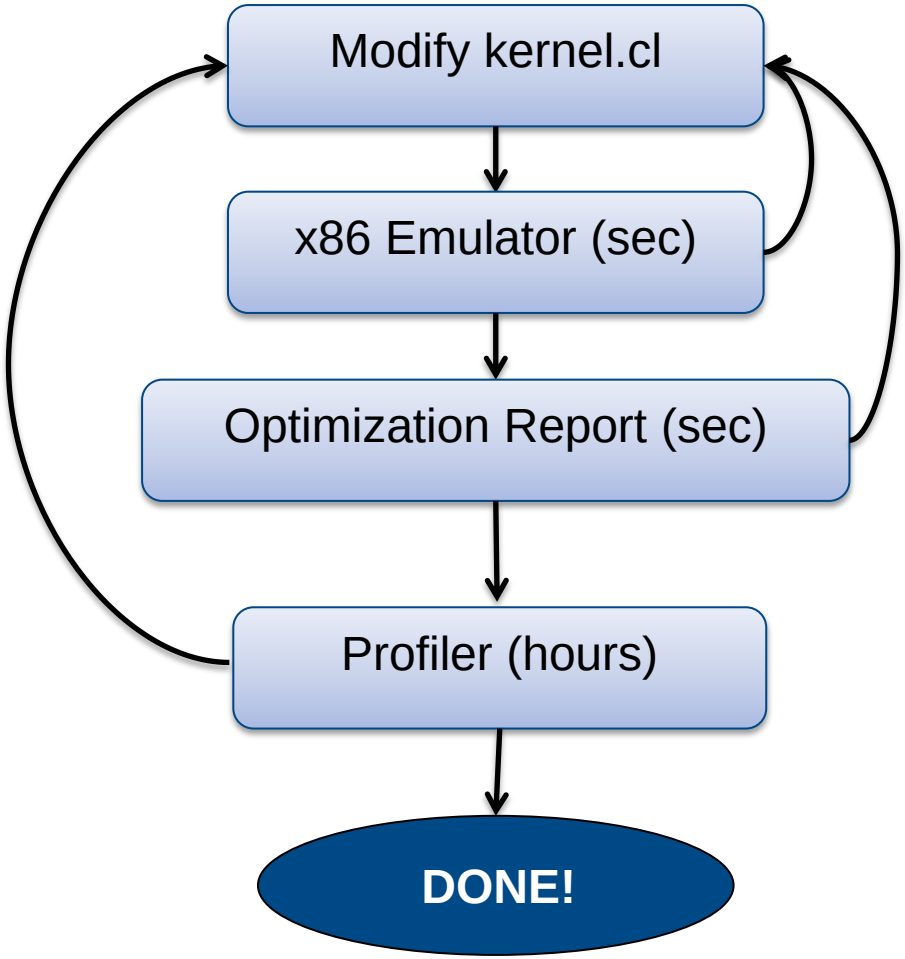
Quartus II

Accelerator

© 2016 Altera—Confidential

ALTERA now part of Intel

Superior Software Development Flow



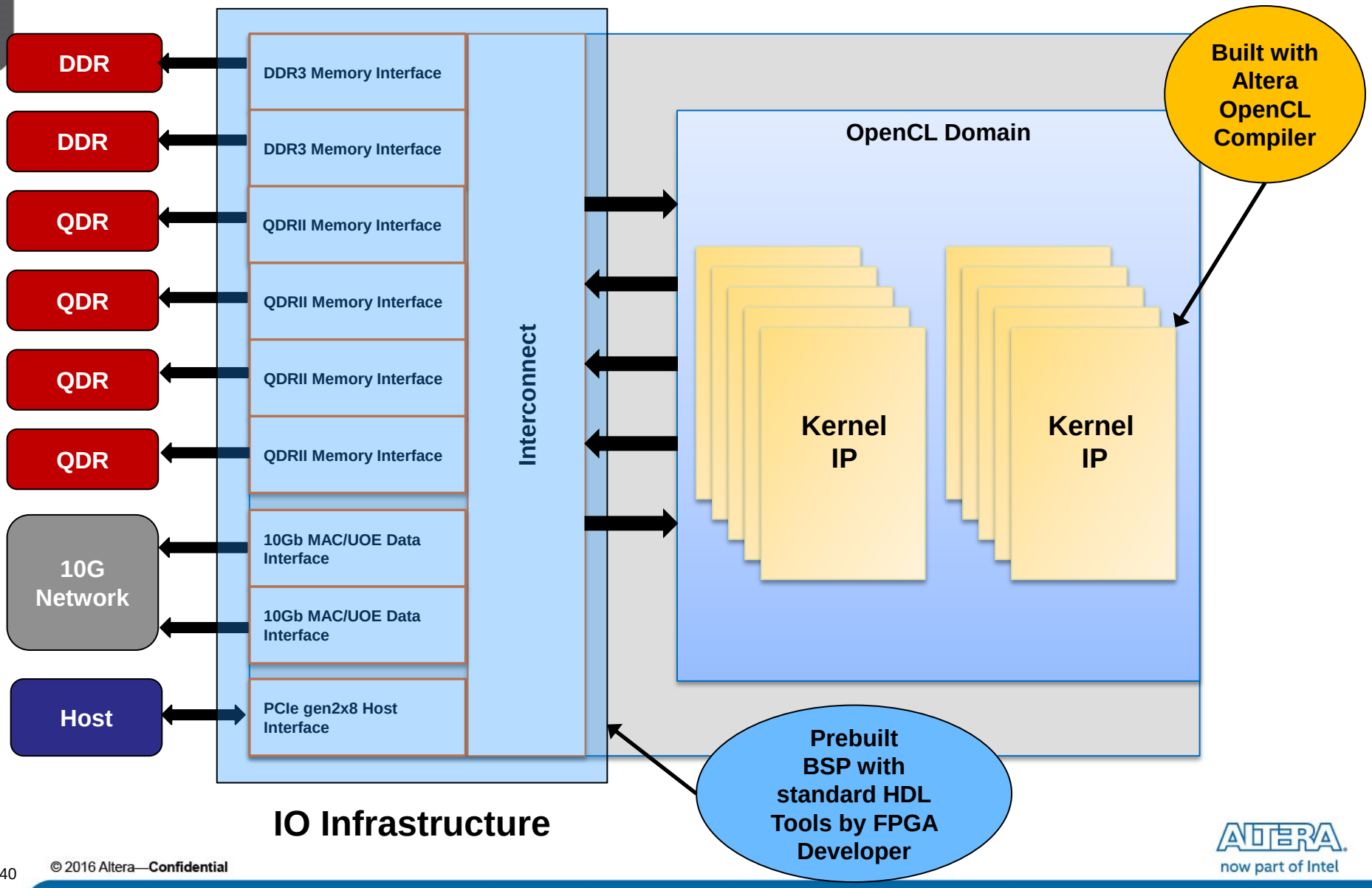
Functional Bugs?

Stall-free pipeline?
Memory coalesced?

System performance met?

© 2016 Altera—Confidential
ALTERA now part of Intel

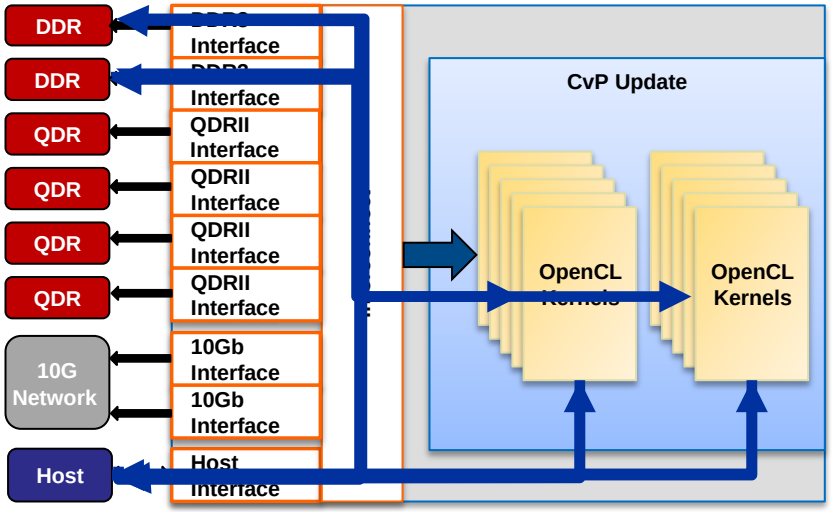
The Only Custom Accelerator Solution: Platforms



© 2016 Altera—Confidential
ALTERA now part of Intel

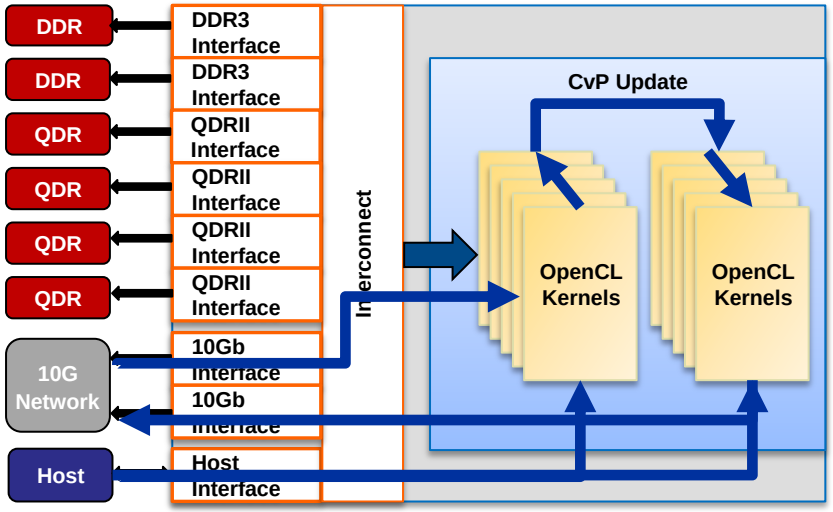
Channels Advantage

Standard OpenCL



Altera Vendor Extension

IO and Kernel Channels




```
channel int DataChannel;

kernel producer(...) {
  write_channel_altera(DataChannel, value);
}

kernel consumer(...) {
  value = read_channel_altera(DataChannel);
}
```


© 2016 Altera—Confidential

Altera SDK for OpenCL Competitive Differentiator

- Altera's SDK for OpenCL has proven to be a powerful solution for many vendors
  - Won ***design tool and development software Elektra*** award in Europe
  - Won ***Ultimate P***
  - Actively being ***d***

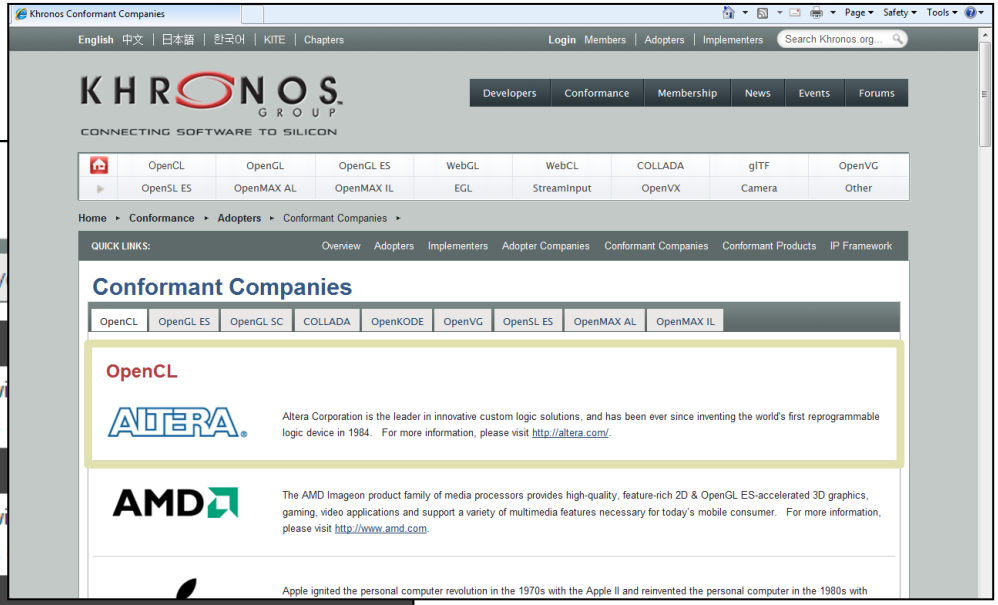
"I was extremely happy to get a great performance with such low effort. I was so impressed with how powerful the Altera tool was! ☺"

--- Senior Engineer,
Altera OpenCL Customer

© 2016 Altera—Confidential

Conformant to OpenCL

- OpenCL v1.0 specification
  - >8500 Programs tested
- Supports Arm Host
  - CV and AV SoC



Conformant Products

OpenCL	OpenGL ES	OpenGL SC	OpenKODE	OpenV
Intel 2013-09-11 OpenCL_1_2				
Windows 8.1 64-bit pre-release build running on Confidential Product				x86_32 wi Windows
Intel 2013-09-11 OpenCL_1_2				
Product: Windows 8.1 32-bit pre-release build running on Confidential Product				x86_32 wi Windows
Intel 2013-08-23 OpenCL_1_2				
Windows 8 64-bit 9200 (RTM) running on Confidential Product				x86_32 with minimum requirement of SSE 4_2 Windows 8 64-bit RTM
Altera Corporation 2013-08-03 OpenCL_1_0				
Nallatech PCIe-385N (Altera Stratix V A7)				Altera Stratix V FPGA
Intel 2013-07-23 OpenCL_1_2				

http://www.khronos.org/conformance/adopters/conformant-companies
http://www.khronos.org/conformance/adopters/conformant-products

© 2016 Altera—Confidential
ALTERA now part of Intel

Additional Altera OpenCL Collateral

- White papers on OpenCL
- OpenCL online demos
- OpenCL design examples
- Instructor-Led training
  - Parallel Computing with OpenCL Workshop by Altera – (1 Day)
  - Optimization of OpenCL for Altera FPGAs Training by Altera – (1 Day)
- Online training
  - Introduction to Parallel Computing with OpenCL
  - Writing OpenCL Programs for Altera FPGAs
  - Running OpenCL on Altera FPGAs
  - Single-Threaded vs. Multi-Threaded Kernels
  - Building Custom Platforms for Altera SDK for OpenCL
- OpenCL board partners page

© 2016 Altera—Confidential
ALTERA now part of Intel

Thank You

© 2016 Altera—**Confidential**