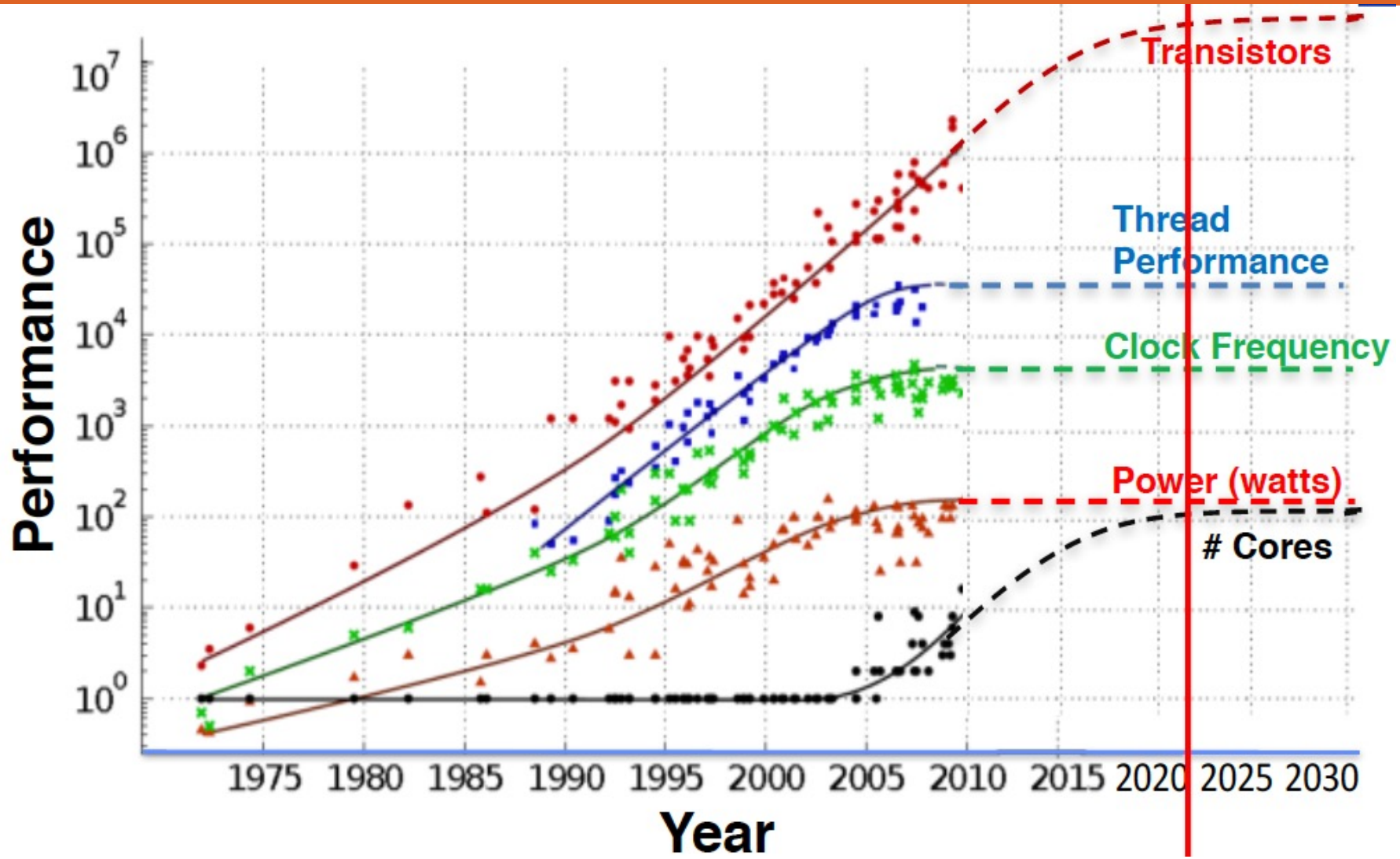
A PRIMER ON FIELD-PROGRAMMABLE GATE ARRAYS (FPGAS)

Senior Research Software Engineer
Research Computing
Bei Wang
Nov 10, 2021

Outline

- Introduction to FPGAs
  - Motivation
  - Architecture
  - Programming FPGA
  - Intel FPGA SDK for OpenCL
- FPGA nodes at Princeton Research
  - Della-fpga
  - Workflow of running an OpenCL application at della-fpga nodes
- Insights: adoption FPGAs for HPC

Moore's Law



Post Moore Scaling

https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/, John Shalf, Digital Computing Beyond Moore's Law, Supercomputing Frontiers, Singapore 2017

Why FPGA for HPC

- Architectural specialization is one option to continue to improve performance beyond the limits imposed by the slow down in Moore's law

- Using application-specific hardware allows more efficient use of the hardware, both in terms of power usage and performance

Mapping Computation on FPGAs

High-level code

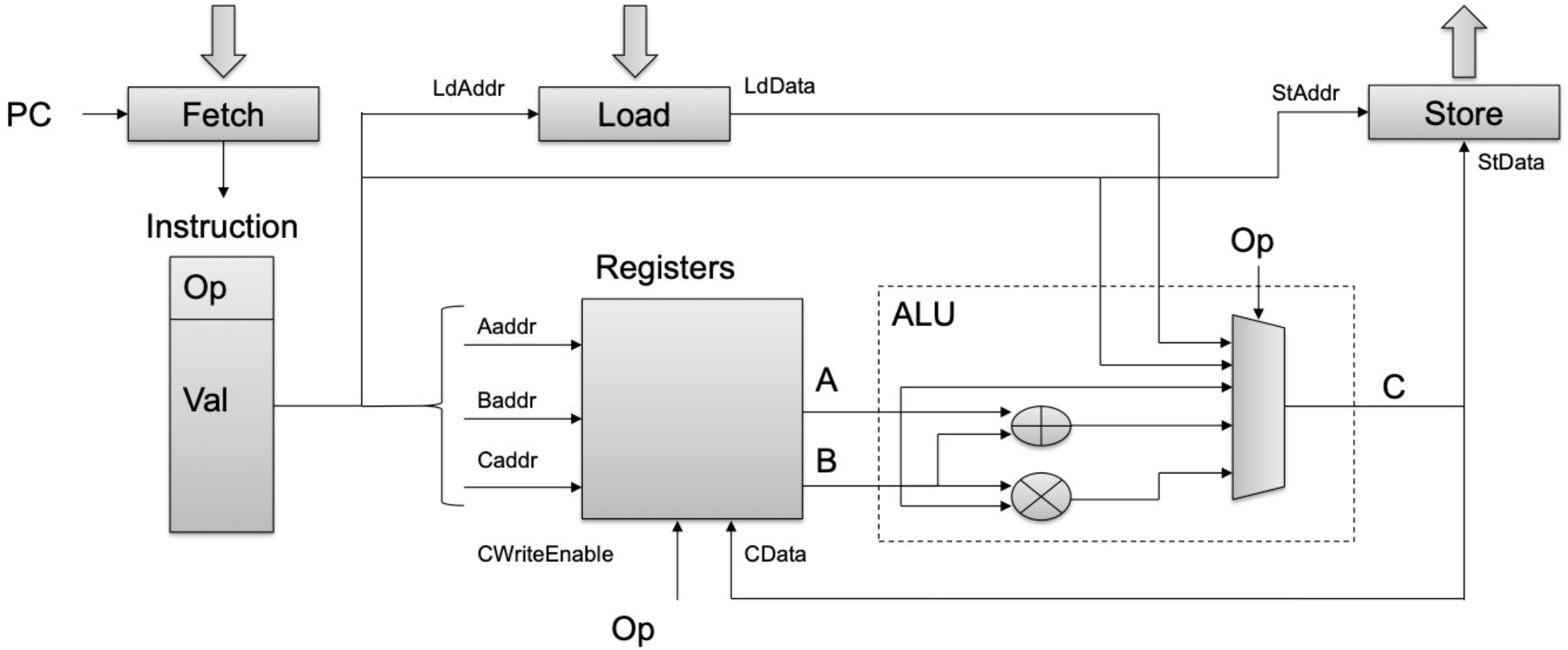
```
Mem[100] += 42 * Mem[101]
```

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]
```

OpenCL for FPGAs, Dmitry Denisenko, Intel Programmable Solution Group

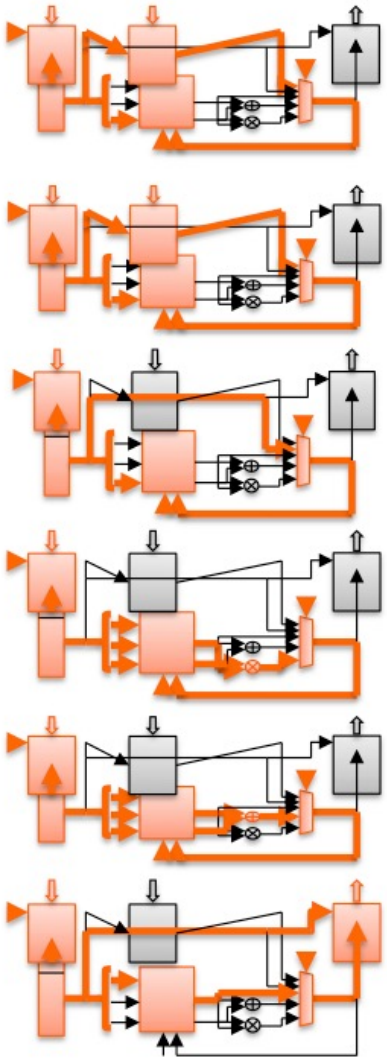
CPU Architecture



- General architecture with data paths covering all cases
- Fixed data width
- Fixed operations

CPU Activities over Time

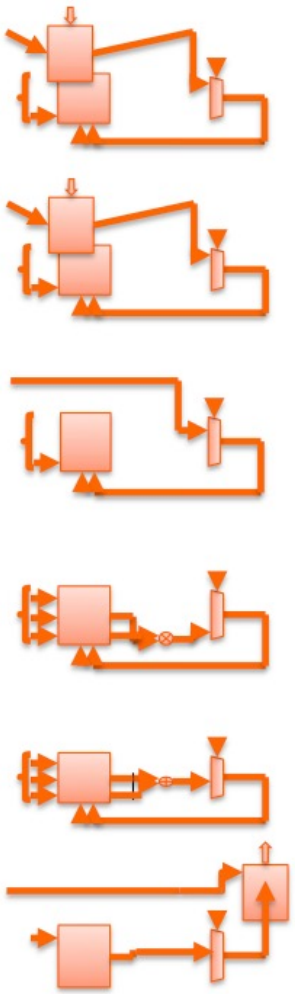
```
R0 ← Load Mem[100]
R1 ← Load Mem[101]
R2 ← Load #42
R2 ← Mul R1, R2
R0 ← Add R2, R0
Store R0 → Mem[100]
```



OpenCL for FPGAs, Dmitry Denisenko, Intel Programmable Solution Group

FPGA Activities over *Space* 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]
```

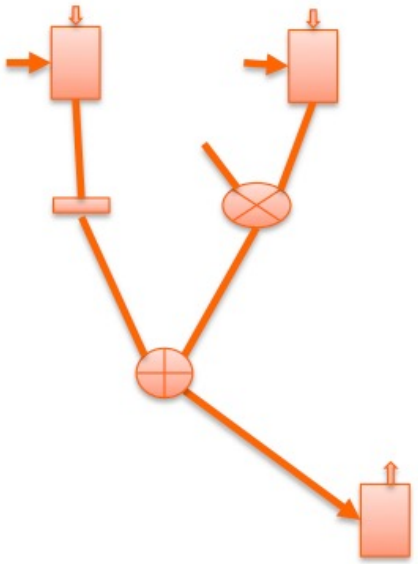


1. Unroll the CPU hardware in space
2. Remove instruction "Fetch" since instructions are fixed
3. Remove unused ALU ops
4. Remove unused Load/Store units

OpenCL for FPGAs, Dmitry Denisenko, Intel Programmable Solution Group

Further specialization

```
R0 ← Load Mem[100]

R1 ← Load Mem[101]

R2 ← Load #42

R2 ← Mul R1, R2

R0 ← Add R2, R0

Store R0 → Mem[100]
```



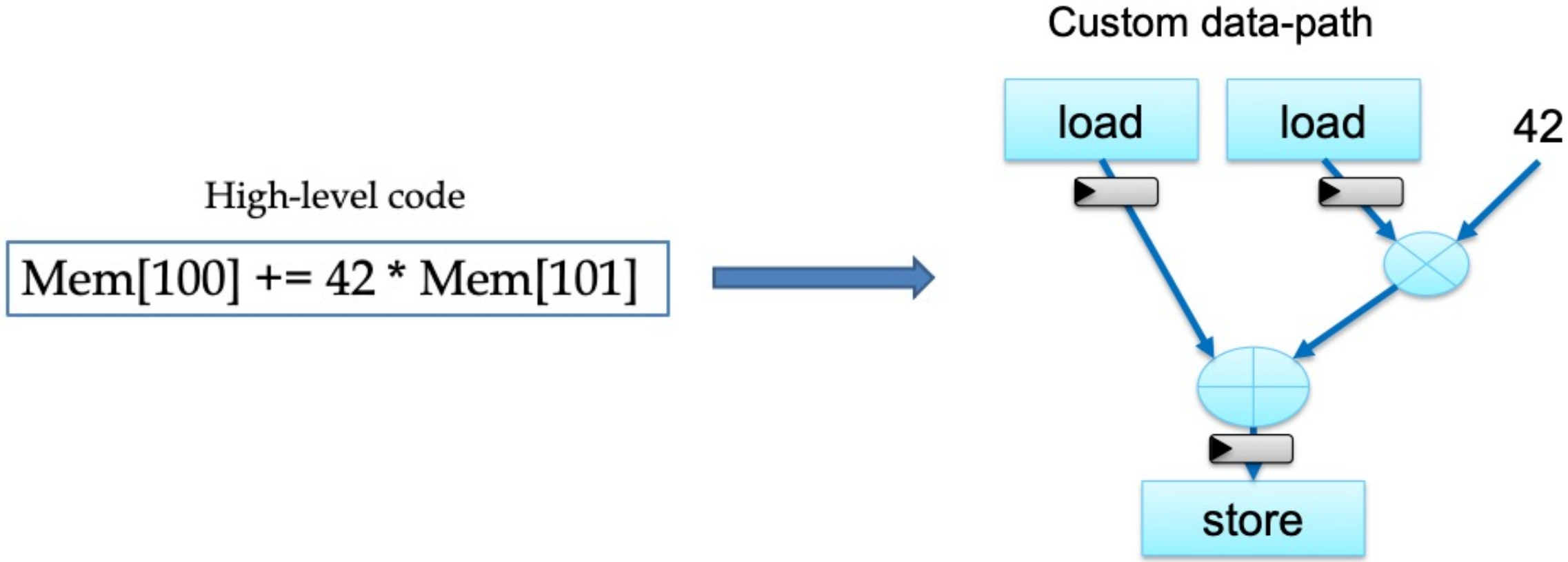
4. Wire up registers properly
5. Remove dead data
6. Reschedule

OpenCL for FPGAs, Dmitry Denisenko, Intel Programmable Solution Group

Custom Data-path on FPGA

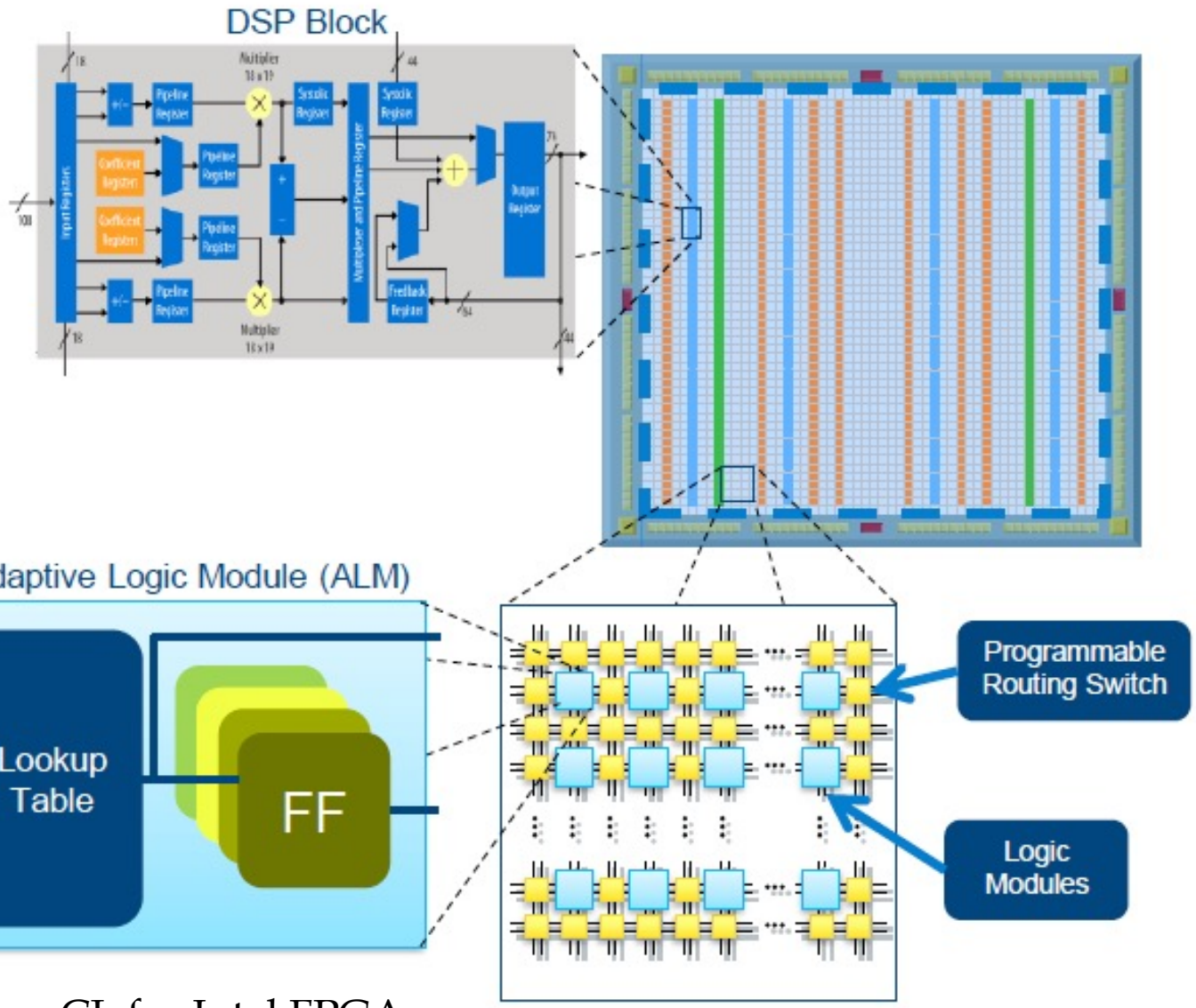
High-level code

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



FPGA Architecture

- Massive Parallelism
  - Millions of logic elements
  - Thousands of embedded memory blocks
  - Thousands of Variable Precision DSP blocks
  - Programmable routing
  - Dozens of High-speed transceivers
  - Various built-in hardened IP
- FPGA Advantages
  - **Custom hardware!**
  - Efficient processing
  - Low power
  - Ability to reconfigure
  - Fast time-to-market



Intel FPGA Technical Training: Optimizing OpenCL for Intel FPGAs

FPGA for HPC

- Roadblocks
  - Traditionally programmed using Hardware Description Language (HDL) mainly Verilog and VHDL
  - Had limited computational capabilities
- Radical changes in recent years
  - OpenCL has been adopted two major FPGA vendors, Altera (now Intel) and Xilinx
  - Intel introduced new Arria 10 FPGA family, which for the first time in the history of FPGAs, included DSPs with native support for floating point operations
- FPGAs are still behind GPUs in terms of both compute performance and external memory bandwidth

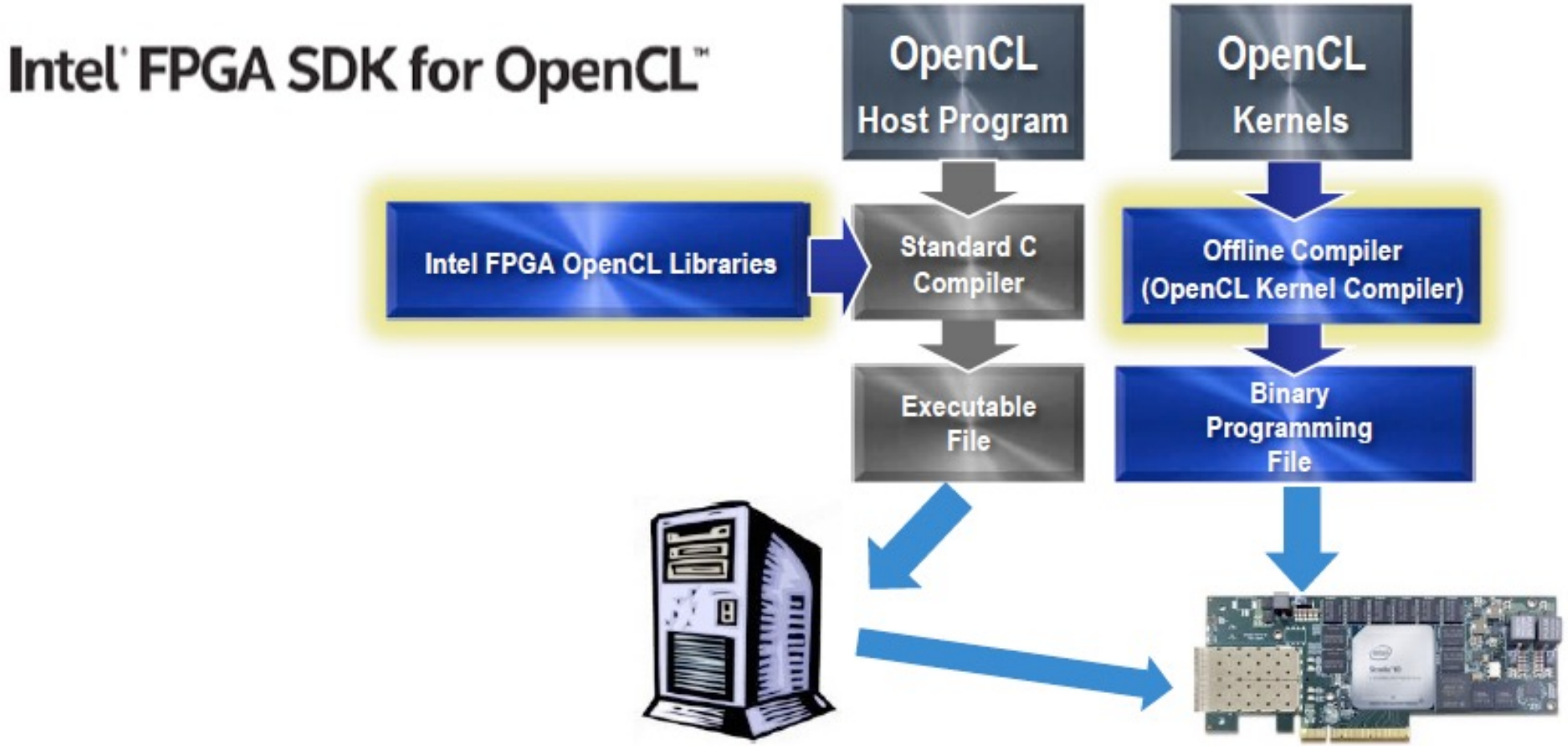
	Peak performance (sp)	Memory bandwidth	Power efficiency
Arria 10 GX 1150 FPGA	1450 GFLOP/s	34.1 GB/s	70 Watts
NVIDIA GTX 980 Ti GPU	6900 GFLOP/s	336.6 GB/s	275 Watts

Ref: Hamid Reza Zohouri, High Performance Computing with OpenCL, Ph.D. thesis, 2018

Programming FPGA

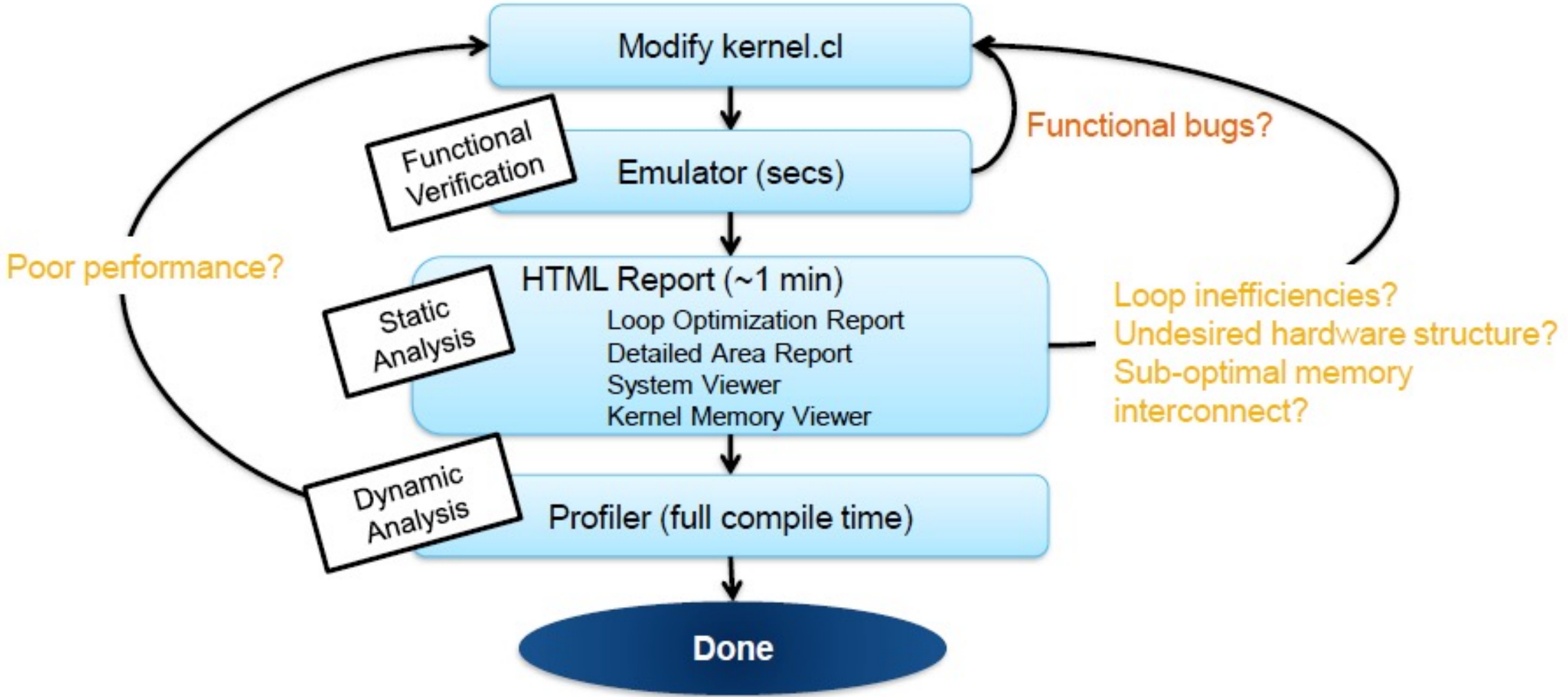
- Hardware description languages (HDL) such as VHDL or Verilog
  - Used by hardware designers only
  - Describe the behavior of the algorithm to crease low level digital circuit
  - Take several months to even years
- High level synthesis (HLS)
  - Makes FPGA usable by software programmers
  - Design at a higher level of abstract by leveraging GNU compatible HLS compiler
- OpenCL
  - Design with C/C++ based software language
  - Makes FPGA acceleration available to software developers
  - Open standard for heterogeneous computing
- OneAPI
  - Based on data parallel C++ (DPC++) programming language and runtime
  - Consists of a set of C++ classes, templates and libraries to express a DPC++ program
  - Develop a clean, modern C++ based application w/o most of the setup associated with OpenCL code

Intel FPGA SDK for OpenCL



Intel FPGA Technical Training: Optimizing OpenCL for Intel FPGAs

Kernel Development Flow and Tools



Intel FPGA Technical Training: Optimizing OpenCL for Intel FPGAs

SDK Components

- **Offline Compiler (AOC)**
  - Translates OpenCL kernel source code into an Intel FPGA hardware configuration file
- Host Libraries
  - Provide OpenCL host and runtime API for host application
- **AOCL Utility**
  - Performs various tasks related to board, drivers, and compile source
- Software Requirements
  - Intel Quartus Prime software, plus license
  - Intel FPGA SDK for OpenCL, plus license
  - C compiler for the host program

Offline Compiler (AOC) Options

Option	Description
--list-boards	Prints a list of available boards
--board	Compile for the specific board
-march=emulator	Create kernels that can be execute and debugged on the host x86 w/o the board
-g	Add debug data to reports
-rtl	Compile and link the kernel or object files w/o the board; Generate compiler optimization report
--report	Print out area estimates to screen
--profile	Enable profile support when generating aocx file
--help or -h	Help for the tool

AOCL Utilities

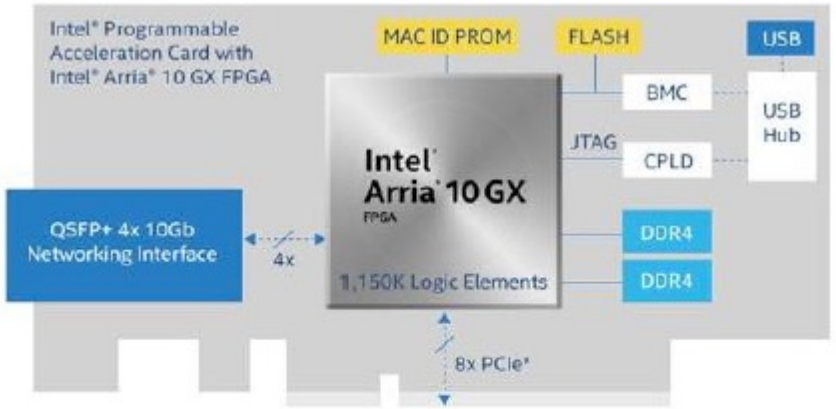
Host Compilation Commands	
aocl compile-config	Displays the compiler flags for compiling your host program
aocl link-config	Shows the link options needed by the host program to link with libraries
aocl makefile	Shows example makefile to compile and link a host program

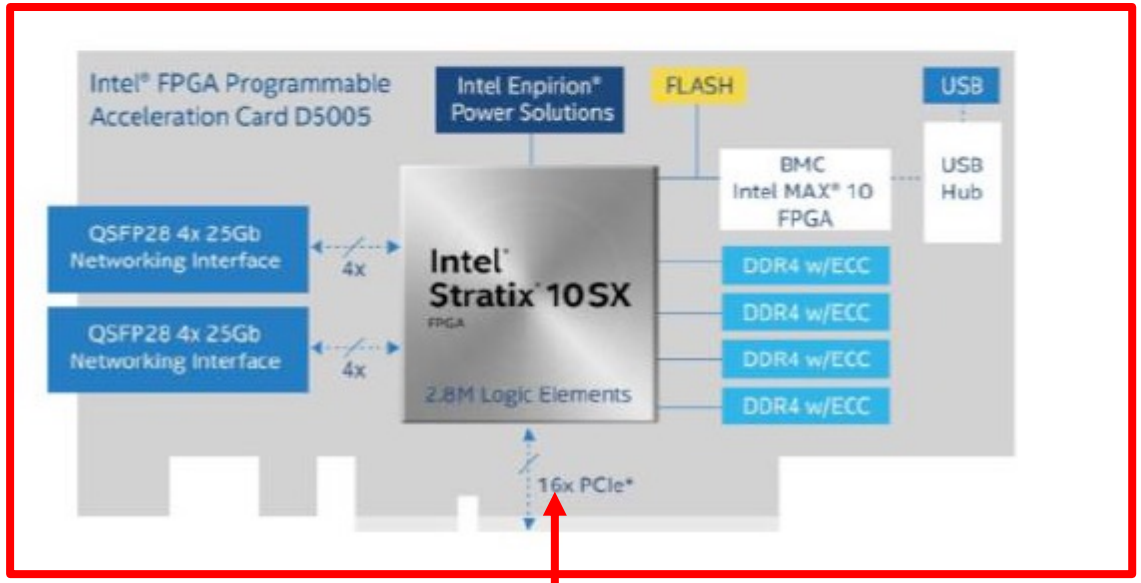
Board Management Commands	
aocl diagnose	Runs the diagnose test program
aocl program	Program the FPGA using the provided aocx file

Others	
aocl report	Displays kernel execution profiler data
aocl env	Displays how the aocx file was compiled

Use “aocl help” for all commands

Intel FPGAs Available





Princeton Research Computing

FPGAs in Research Computing

- Princeton research computing recently installed four FPGAs on the Della cluster
- They can be accessed through:
  - **ssh –l netid della-fpga1.princeton.edu** or **ssh –l netid della-fpga2.princeton.edu**
  - Temporary account is created on della-fpga2 for you. The account will be valid till the end of this week
- Each node has two FPGAs
- There is NO scheduler system installed in these two FPGA nodes

Using OpenCL in Della

- Set up OpenCL environment in della-fpga1 and della-fpga2
  - source /opt/intel/fpga-d5005/inteldevstack/init_env_ocl_20.1.sh

```
export QUARTUS_HOME=/opt/intelFPGA_pro/quartus_19.2.0b57/quartus
export OPAE_PLATFORM_ROOT=/opt/intel/fpga-d5005/inteldevstack/d5005_ias_2_0_1_b237
export AOCL_BOARD_PACKAGE_ROOT=/opt/intel/fpga-d5005/inteldevstack/d5005_ias_2_0_1_b237/opencl/opencl
_bsp
$OPAE_PLATFORM_ROOT/bin is in PATH already
export INTELFPGAOCLSDKROOT=/opt/intelFPGA_pro/20.1/hld
export ALTERAOCLSDKROOT=/opt/intelFPGA_pro/20.1/hld
$QUARTUS_HOME/bin is in PATH already
source /opt/intelFPGA_pro/20.1/hld/init_opencl.sh
```

- Compile on emulation mode on x86 (debugging)
  - aoc –march=emulator kernel_name.cl (option –legacy-emulator is required for compiling using 19.2 OpenCL SDK)
  - Set CL_CONFIG_CPU_EMULATE_DEVICES=<number_of_devices> if using more than 1 devices
- Compile and link w/o building hardware (generating *.aocr file and html report)
  - aoc –rtl kernel_name.cl –report
- Full deployment (generating *.aocx file)
  - aoc kernel_name.cl

Special Setup for Running Emulation Mode at Della-fpga

- The emulator in SDK is built with GCC 7.2.0 and so the libstdc++.so linked to the host have to be at least as new as provided in GCC 7.2.0 which is libstdc++.so.6.0.24
- The devtoolkit provided at RHEL 7 system at della-fpga does not provide the required libstdc++ version
- Fortunately, anaconda carries libstdc++.so.6.0.26 which is from GCC 9.1.0
- To link to that library, we need to run the host as:

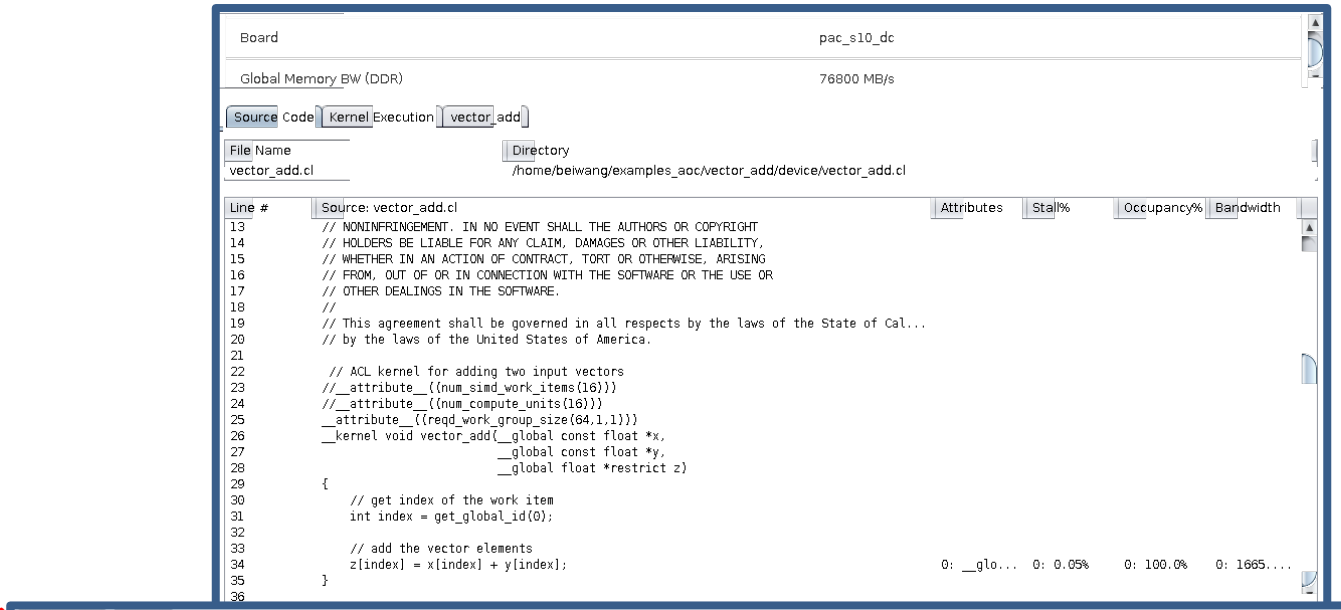
```
env LD_LIBRARY_PATH=/usr/licensed/anaconda3/2020.7/lib:$LD_LIBRARY_PATH
./host
```

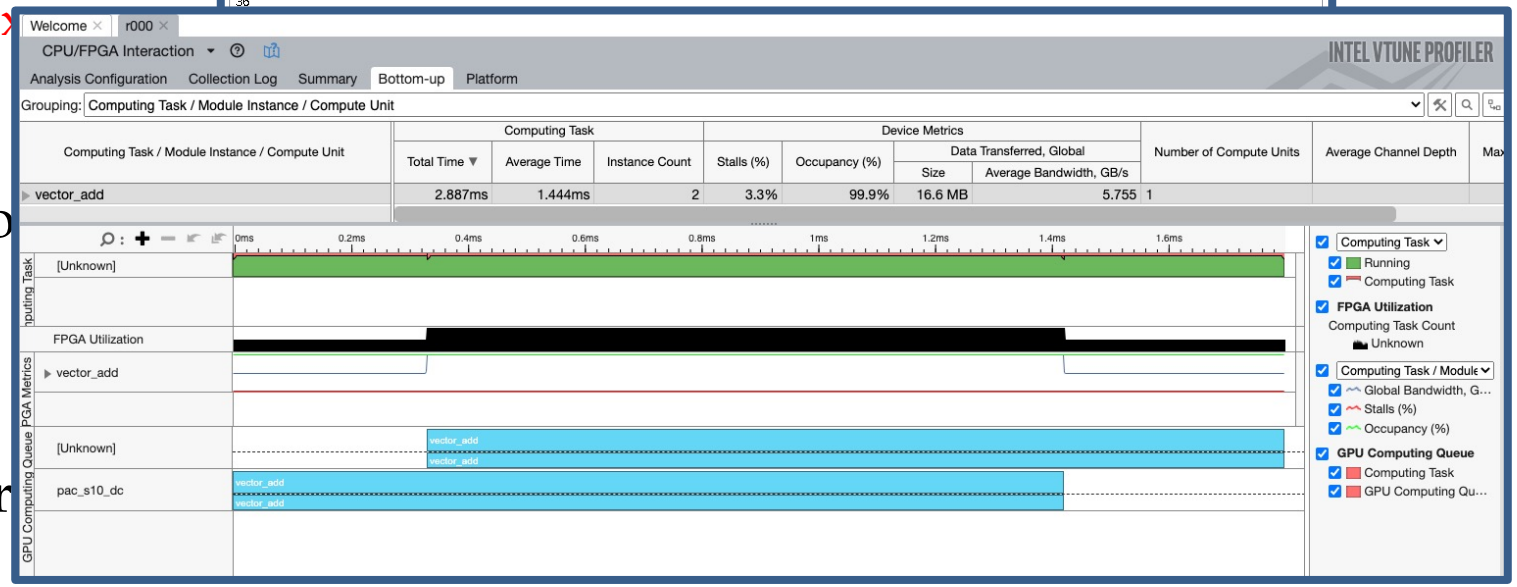
My .bashrc at della-fpga

```
# Setup the env variables to use Quartus 19.2 and FPGA SDK 19.4 version
#source /opt/intel/fpga-d5005/inteldevstack/init_env_ocl_19_4.sh
#source /opt/intel/fpga-d5005/inteldevstack/init_env.sh
source /opt/intel/fpga-d5005/inteldevstack/init_env_ocl_20.1.sh
# Display memory transfer information in aocl profile
export ACL_PROFILE_TIMER=1

# Uncomment the following line if you want to compile on emulation mode on FPGA
#export CL_CONFIG_CPU_EMULATE_DEVICES=1

module load rh/devtoolset/9 cmake #anaconda3
export LD_LIBRARY_PATH=/usr/licensed/anaconda3/2020.7/lib:$LD_LIBRARY_PATH
```

Profiling

- To instrument the OpenCL kernel pipeline with performance counters, include –profile option of the *aoc* command when compiling the kernel
- The counter information is saved in a profile.mon monitor description file and can be converted into a readable profile.json file
  - aocl profile ./host -x kernel_filename.aocx –s kernel_filename.source
- Use the Intel FPGA Dynamic Profiler for OpenCL *report* utility command to launch the profiler GUI
  - aocl report kernel_filename.aocx profile.mon kernel_filename.source
- Alternatively use Intel VTune Profiler to open the profile.json file





Live Demo

Insights: adoption FPGAs in HPC

- The main source of performance bottleneck in current-generation FPGAs is external memory bandwidth.

Device	Peak Perf (GFLOP/s)	Peak Bandwidth (GB/s)	TDP (Watt)	Year
Tesla V100	15,700	897	300	2017
Stratix 10 GX	8,640	76.8	200	2018

- A big portion of HPC applications rely on double-precision (or even higher) computation which cannot be efficiently realized on current FPGAs
- Placement and routing time on FPGAs is a major limiting factor in performance evaluation of these devices
- Lack of libraries and open-source projects significantly hinder the ability of a large part of the community in adopting FPGAs

Hamid Reza Zohouri, High Performance Computing with OpenCL, Ph.D. thesis, 2018

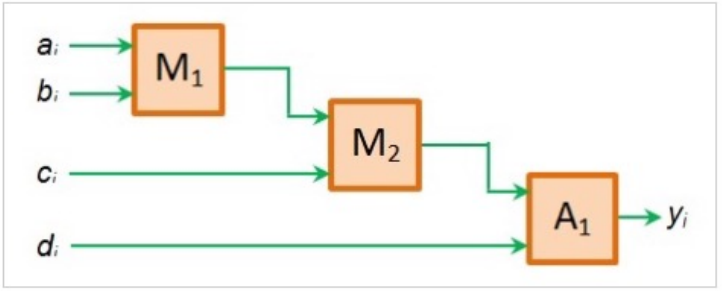
References

- Intel FPGA SDK for OpenCL Pro Edition: Getting Started Guide
- Intel FPGA SDK for OpenCL Pro Edition: Programming Guide
- Intel FPGA SDK for OpenCL Pro Edition: Best Practices Guide
- Free Intel FPGA OpenCL online training
  - Introduction to OpenCL for Intel FPGAs
  - Optimizing OpenCL for Intel FPGAs
- OpenCL for FPGAs, Emitry Denisenko, High-Level Design Team, Intel Programmable Solutions Group
- High Performance Computing with FPGAs and OpenCL, Hamid Reza Zohouri, Ph.D., Thesis, Tokyo Institute of Technology

FPGA Pipeline Parallelism

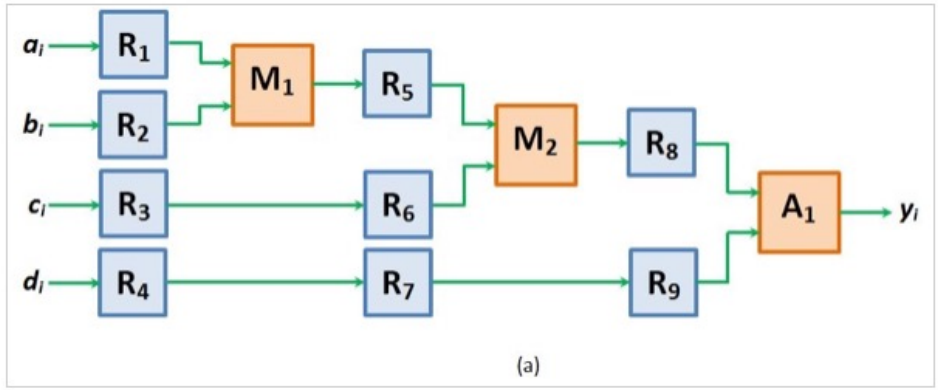
- An example: $(a_i \text{x } b_i \text{x } c_i) + d_i$

Non-pipelined Design



Clk	1	2	3	4	5	6	7	8	9	10	…
M_1	$a_1 \times b_1$	-	-	$a_2 \times b_2$	-	-	$a_3 \times b_3$	-	-	$a_4 \times b_4$	
M_2	-	$a_1 \times b_1 \times c_1$	-	-	$a_2 \times b_2 \times c_2$	-	-	$a_3 \times b_3 \times c_3$	-	-	
A_1	-	-	$a_1 \times b_1 \times c_1 + d_1$	-	-	$a_2 \times b_2 \times c_2 + d_2$	-	-	$a_3 \times b_3 \times c_3 + d_3$	-	
			y_1			y_2			y_3	…	

Pipelined Design



Clk	1	2	3	4	5	…
R_1	a_1	a_2	a_3	a_4	…	
R_2	b_1	b_2	b_3	b_4	…	
R_3	c_1	c_2	c_3	c_4	…	
R_4	d_1	d_2	d_3	d_4	…	
M_1	$a_1 \times b_1$	$a_2 \times b_2$	$a_3 \times b_3$	$a_4 \times b_4$	…	
R_5	-	$a_1 \times b_1$	$a_2 \times b_2$	$a_3 \times b_3$	$a_4 \times b_4$	…
R_6	-	c_1	c_2	c_3	c_4	…
R_7	-	d_1	d_2	d_3	d_4	…
M_2	-	$a_1 \times b_1 \times c_1$	$a_2 \times b_2 \times c_2$	$a_3 \times b_3 \times c_3$	$a_4 \times b_4 \times c_4$	…
R_8	-	-	$a_1 \times b_1 \times c_1$	$a_2 \times b_2 \times c_2$	$a_3 \times b_3 \times c_3$	…
R_9	-	-	d_1	d_2	d_3	…
A_1	-	-	$a_1 \times b_1 \times c_1 + d_1$	$a_2 \times b_2 \times c_2 + d_2$	$a_3 \times b_3 \times c_3 + d_3$	…
			y_1	y_2	y_3	…

https://www.allaboutcircuits.com/technical-articles/why-how-pipelining-in-fpga/