DEEP LEARNING INFERENCE WITH INTEL® FPGAS

Class 5

What We Discussed in the Previous Lesson

Using the Deep Learning Accelerator Suite to implement CNN networks on the FPGA with the OpenVINO™ toolkit

The different architectures and primitives that are provided as part of the DLA Suite, how they are implemented on the FPGA and how they can be customized

Agenda

Introduction and acceleration stack overview

Developing a SW host application

Getting Acceleration Functions

Getting started with the Acceleration Stack

Objectives

Explain How the acceleration stack enables abstraction of the FPGA in Cloud/Enterprise clusters transparently

Define the process for a host application to discover and configure an FPGA accelerator

Explain where to get accelerated functions for the FPGA

Define the process to create an accelerator function unit for the FPGA

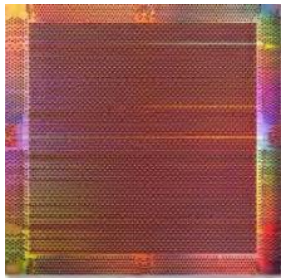
Explain the process for adding an FPGA accelerator card into a cluster and installing the acceleration stack

Relate sample designs to get started

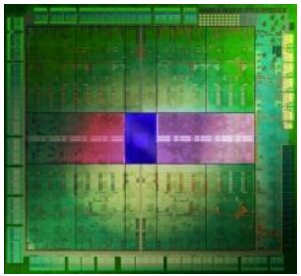
The Future is Heterogenous

Span from CPU, GPU, FPGA to dedicated devices with consistent programming models, languages, and tools

CPUs



GPUs



FPGAs

ASSP

Separation of Concerns

Two groups of developers:

- Domain experts concerned with getting a result
  - Host application developers leverage optimized libraries
- Tuning experts concerned with performance
  - Typical FPGA developers that create optimized libraries

Intel® Math Kernel Library is a simple example of raising the level of abstraction to the math operations

- Domain experts focus on formulating their problems
- Tuning experts focus on vectorization and parallelization

FPGA Enabled Performance and Agility

Workload Optimization: ensure Xeon cores serve their highest value processing

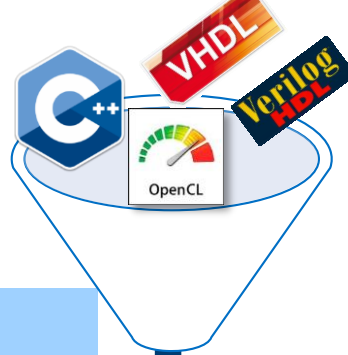
Efficient Performance: improve performance/watt

Real-Time: high bandwidth connectivity and low-latency parallel processing

Developer Advantage: code re-use across Intel FPGA data center products

FPGAs enhance CPU-based processing by *accelerating algorithms* and *minimizing bottlenecks*

Using FPGAs Just Got Easier



Orchestration / Rack Management

User Application

Software Frameworks

Libraries

Open Programmable Acceleration Engine (OPAE)

OS Driver

Increase Abstraction

Increase User Base

User Design

Accelerator Function (AF) (Loadable Workload)

Core Cache Interface

FPGA Interface Manager (Standard I/O Interfaces)

Programmable Accelerator Card

What Acceleration Stack enables

ECOSYSTEM OF FPGA WORKLOADS

End Application User

APPLICATION & FPGA DEVELOPMENT

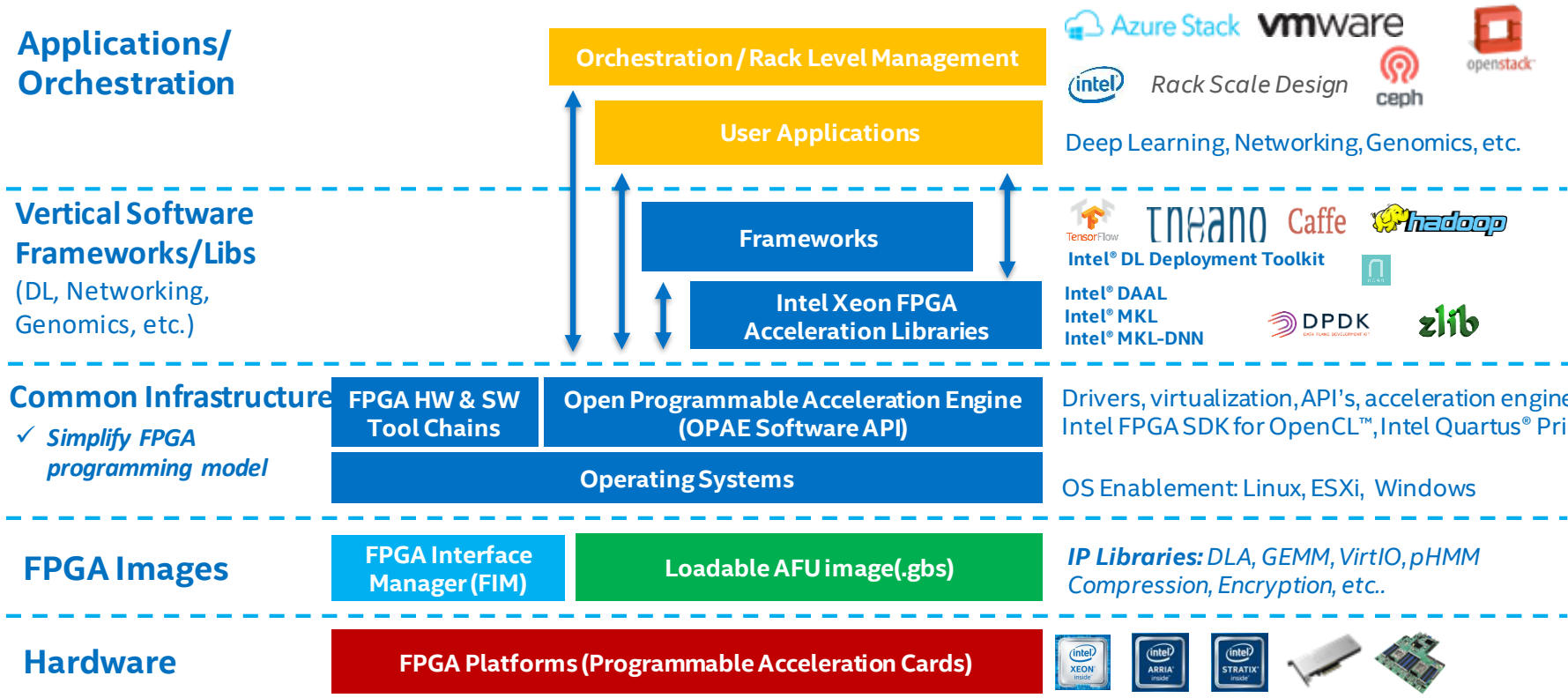
HW & SW Developer

FPGA DEPLOYMENT & MANAGEMENT

Data Center Operator
Integrated Services
Vendors

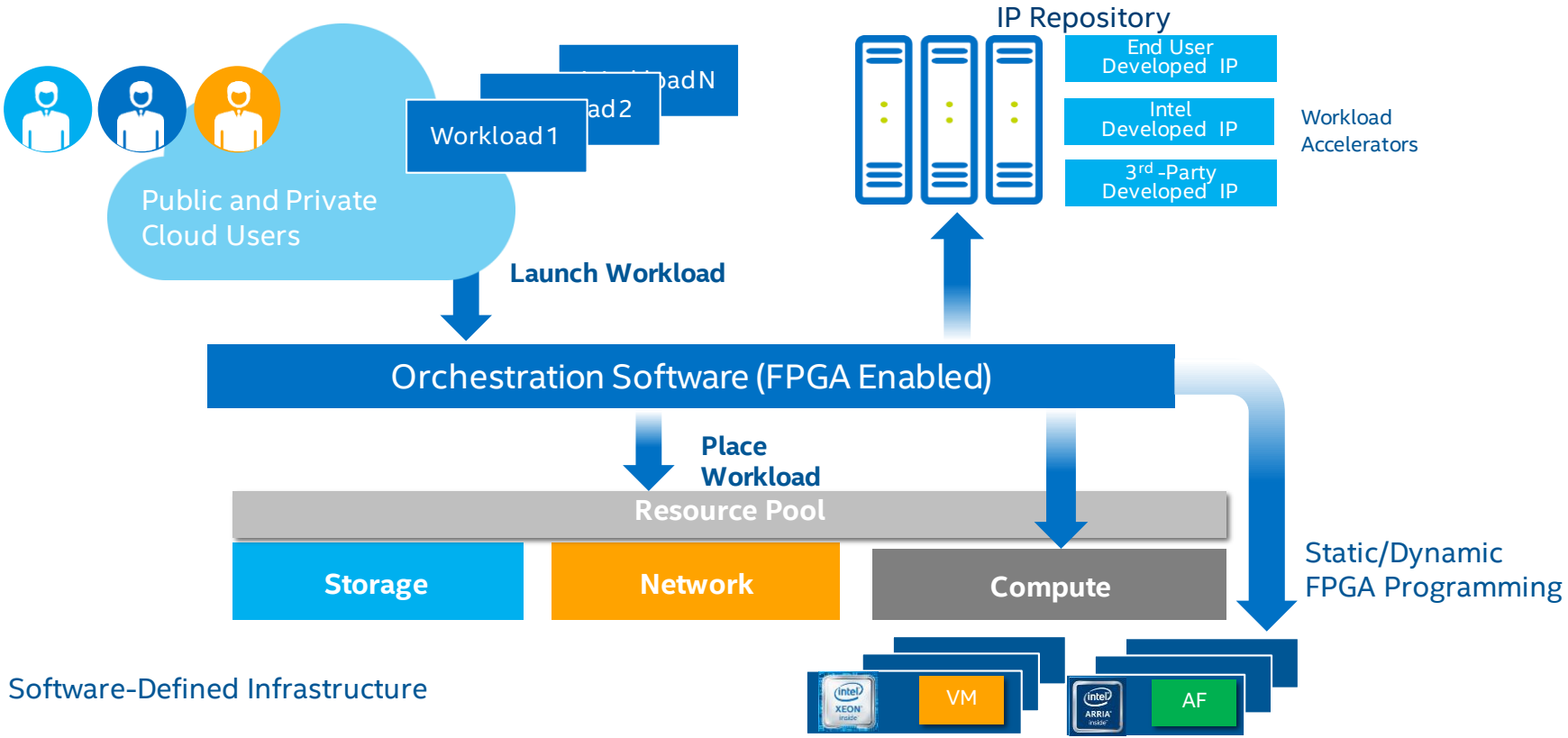
Enabled by

The Acceleration Stack for Intel® Xeon CPU w/ FPGA



*Other names and brands may be claimed as the property of others

Orchestrating FPGA-accelerated applications



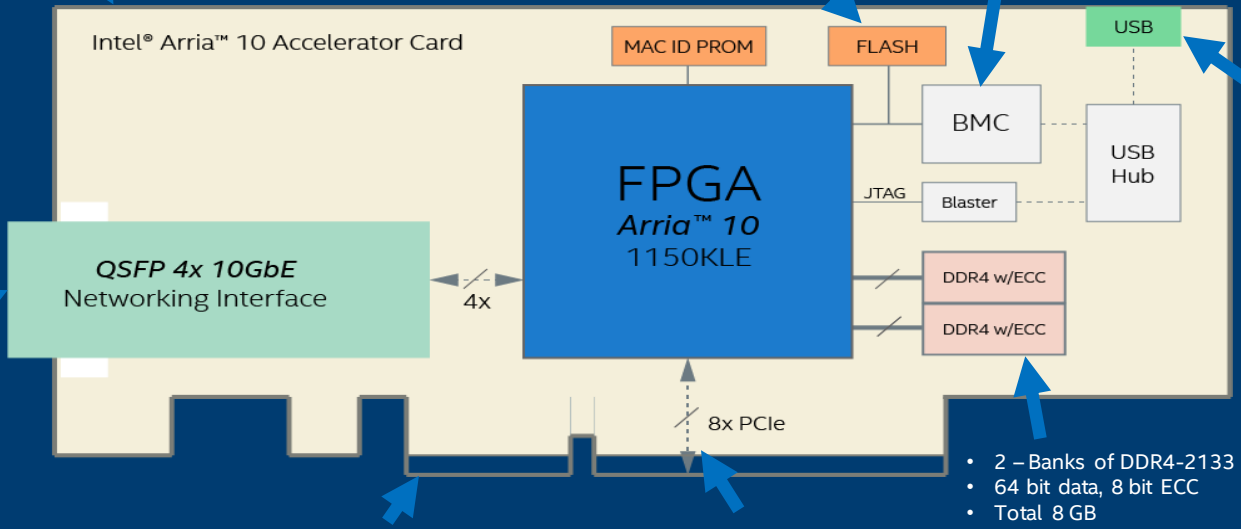
Software-Defined Infrastructure

PROGRAMMABLE ACCELERATION CARD WITH ARRIA 10 FPGA

- Low-profile (half-length, half height) PCIe* slot card
- 168 mm × 56 mm
- Maximum component height: 14.47 mm
- PCIe × 16 mechanical

- 128 MB Flash
- For storage of FPGA configuration

- Board Management Controller (BMC)
- Server class monitor system
- Accessed via USB or PCIe



- USB 2.0 port for board firmware update and FIM image recovery

- QSFP+ Slot accepts pluggable optical modules

- 2 – Banks of DDR4-2133 SDRAM, 4 GB each
- 64 bit data, 8 bit ECC
- Total 8 GB

- Powered from PCIe+12V rail
- 70 W total board power
- 45 W FPGA power

PCIe X 8 Gen3 connectivity to Xeon host

Open Programmable Acceleration Engine (OPAE)

Simplified FPGA Programming Model for Application Developers

Consistent API across product generations and platforms
- Abstraction for hardware specific FPGA resource details

Designed for minimal software overhead and latency
- Lightweight user-space library *(libfpga)*

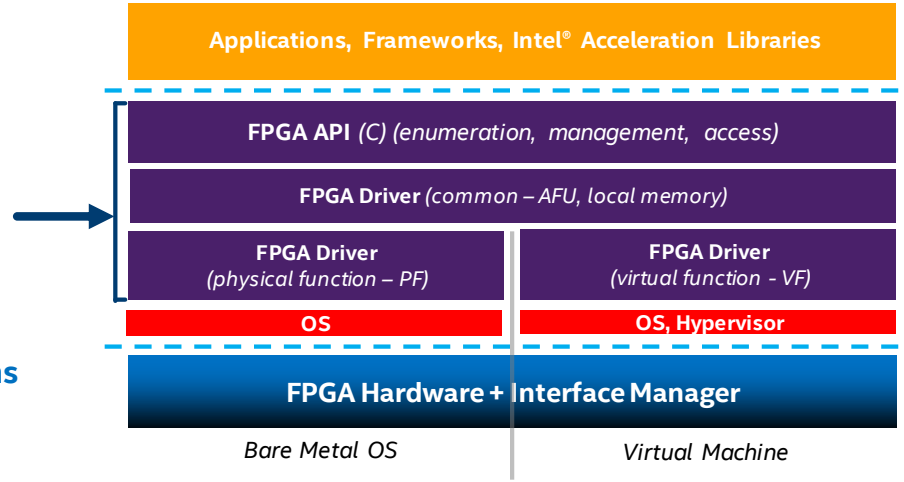
Open ecosystem for industry and developer community
- License: FPGA API (BSD), FPGA driver (GPLv2)

FPGA driver being upstreamed into Linux kernel

Supports both virtual machines and bare metal platforms

Faster development and debugging of Accelerator Functions with the included AFU Simulation Environment (ASE)

Includes guides, command-line utilities and sample code



Start developing for Intel FPGAs with OPAE today: http://01.org/OPAE

FPGA Accelerator Appears as PCIe Device

From the OS's point of view

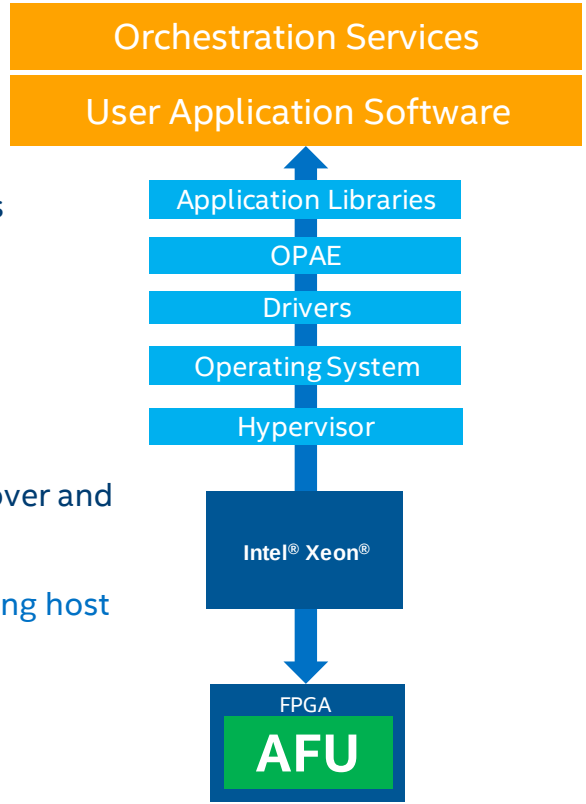
- FPGA hardware appears as a regular PCIe device
- FPGA accelerator appears as a set of features accessible by software programs running on host

Unified C API model

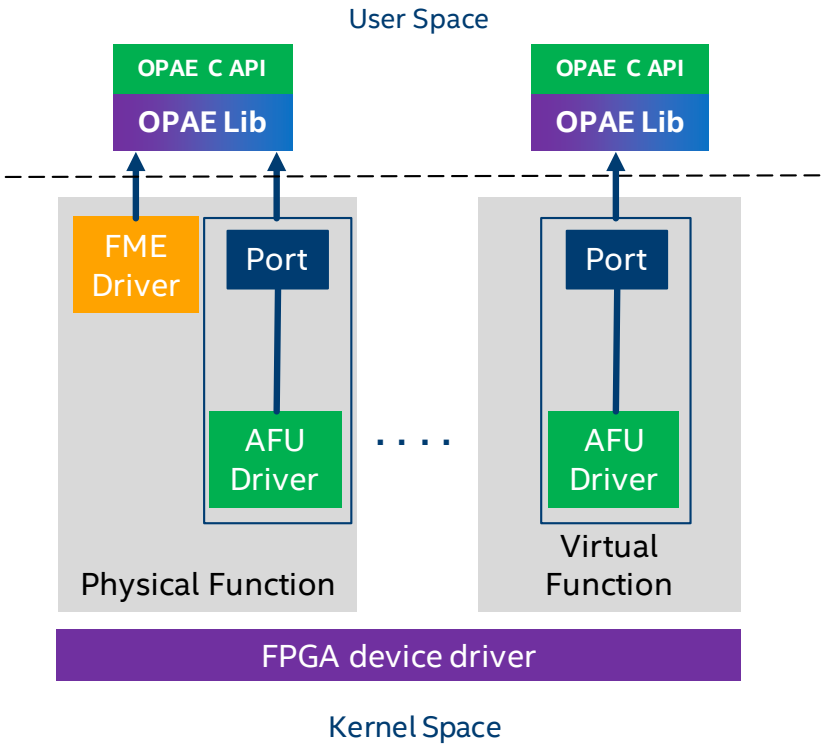
- Supports different kinds of FPGA integration and deployment. (E.g.: A single application can use the FPGA to accelerate certain algorithms)
- Resource management and orchestration services in a data center use to discover and select the FPGA resources and organize them to be used by the workloads

Architecture supports Single Root I/O Virtualization (SROIV) PCIe extension enabling host software to access the accelerator:

- Via a hypervisor/VMM (Virtual Function)
- Bypassing the VMM/Hypervisor Physical Functions



FPGA Driver Architecture



FME: FPGA Management Engine Driver

- Static circuits for power/thermal management, reconfiguration, debugging, error reporting, performance counters, etc.

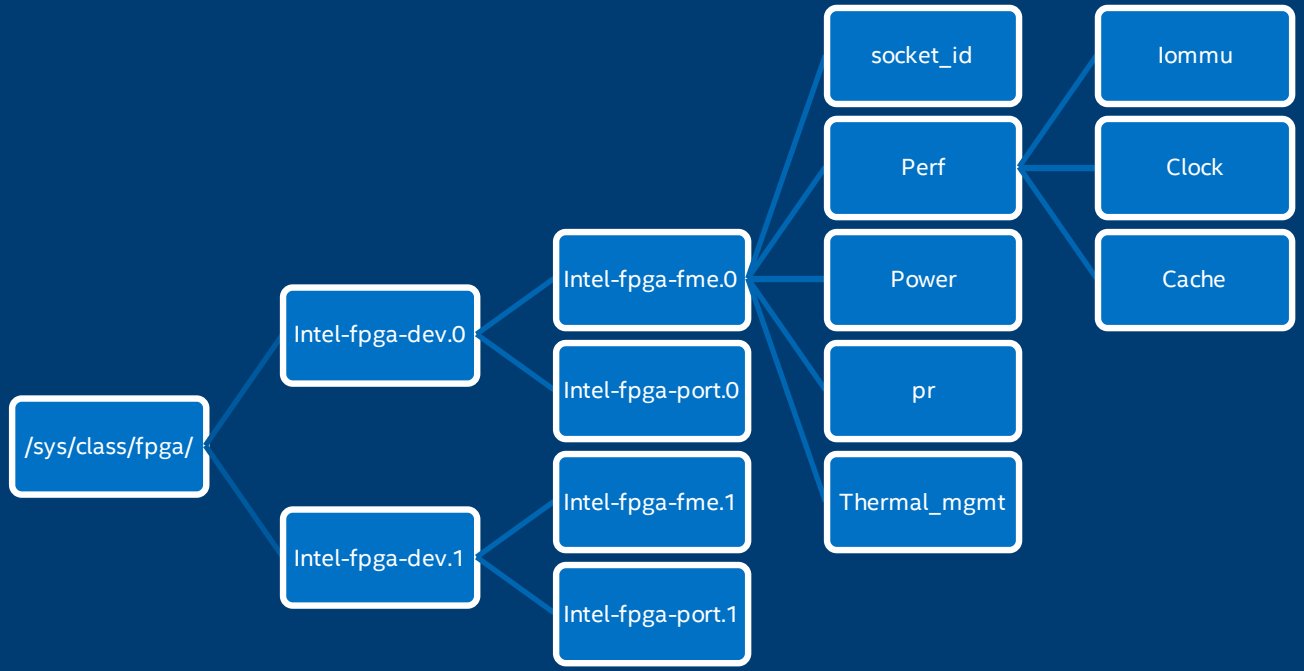
Port:

- Interface between the static (FIM) and the reconfigurable Acceleration Function (AF) region
- Controls communication from software to the accelerator
- Expose features such as reset and debug
- There may be multiple ports exposed through a VF

AFU: Accelerator Function Unit Driver

- Exposes a 256KB region as control registers through Port
- Reconfigurable circuits for application specific functions
- User process can share memory buffers with AFU

HOW HOST APPLICATIONS ENUMERATE THE FPGA DEVICE: SYSFS



Ex:

2 Intel(R) FPGA devices are installed in the host

Each FPGA device has one FME and one Port (AFU)


SW APPLICATION DEVELOPMENT
FOR ACCELERATION STACK

The OPAE Library at a Glance

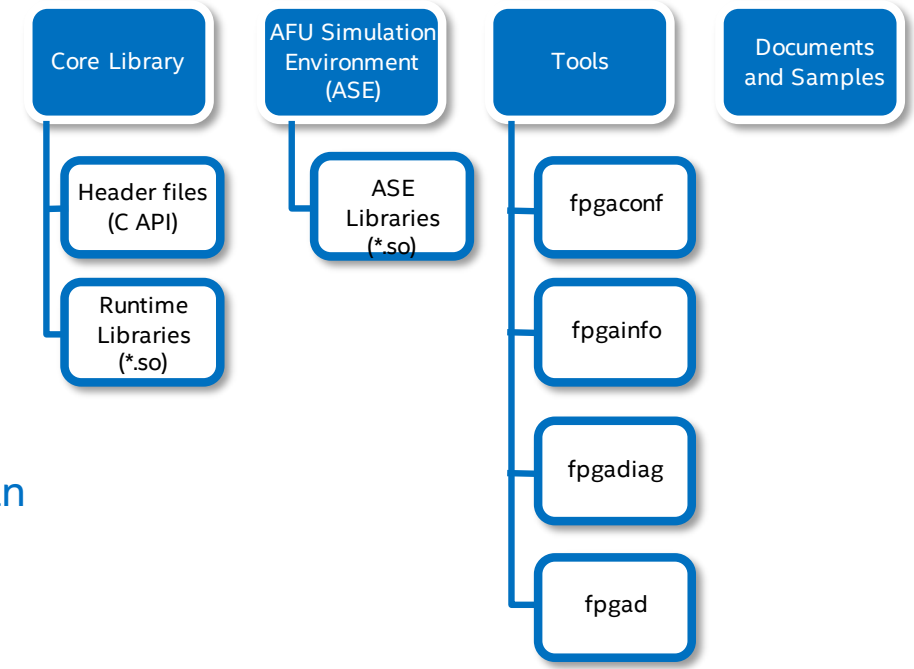
Enumerate, access, and manage FPGA resources through API objects

A common interface across different FPGA form factors

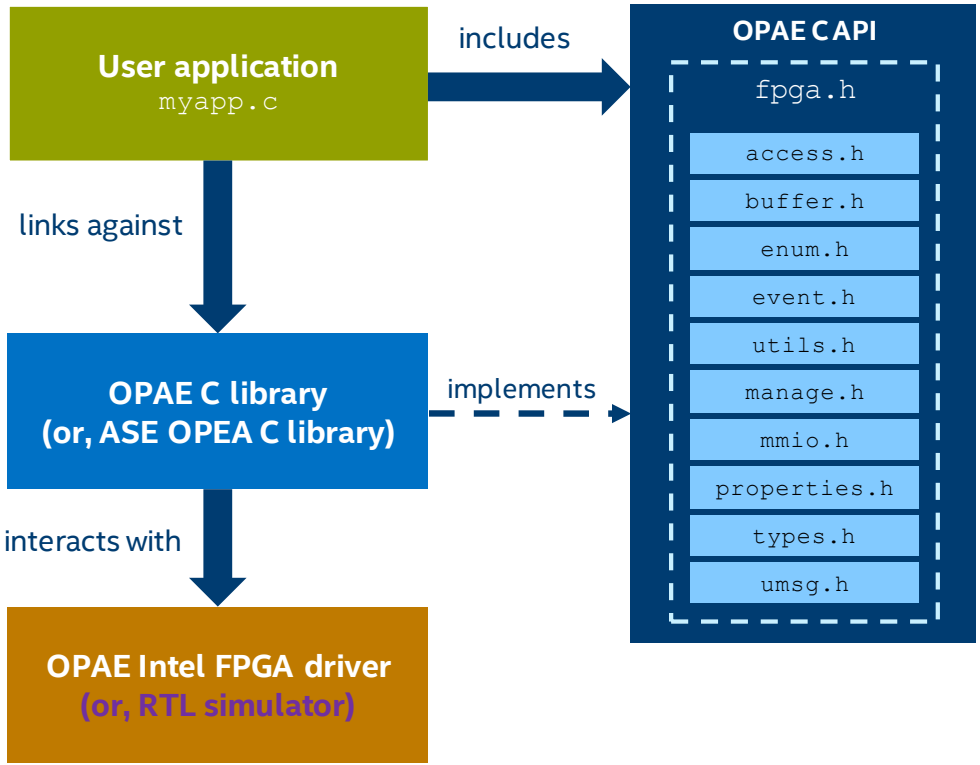
C API designed for extensibility

AFU Simulation Environment (ASE) allows developing and debugging accelerator functions and software applications without an FPGA

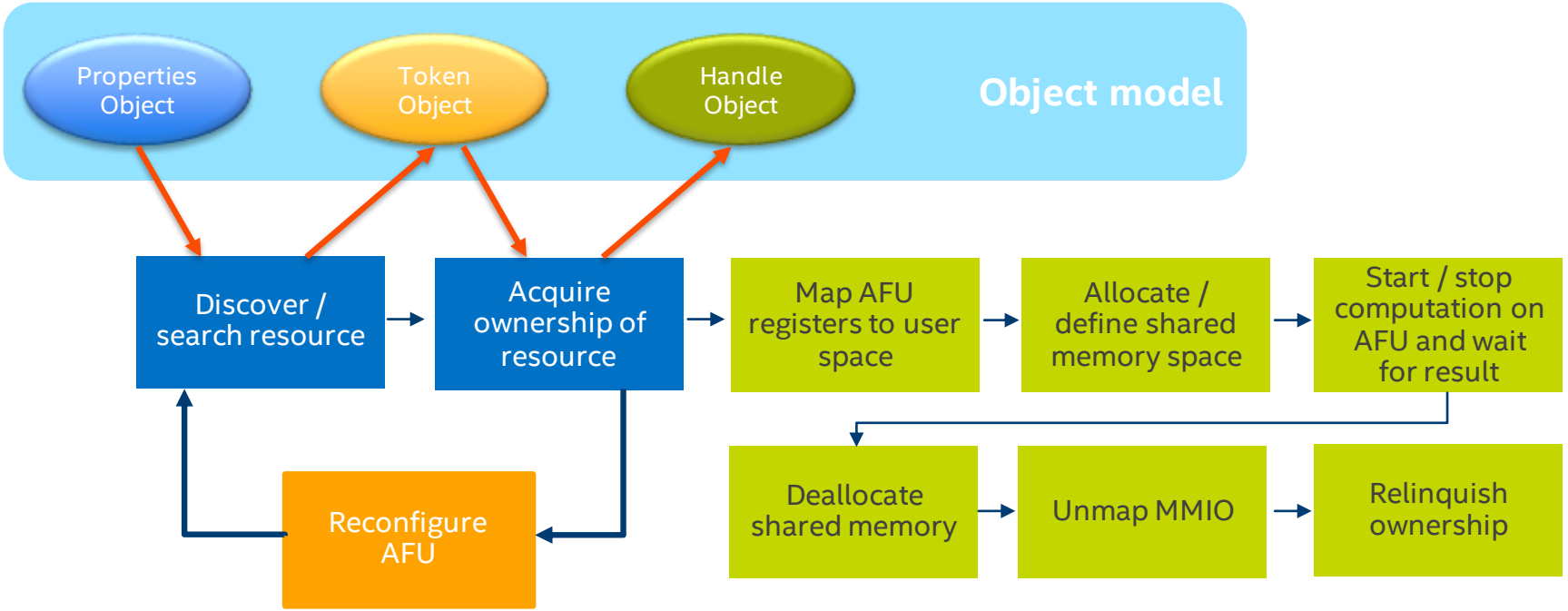
Tools for partial reconfiguration, FPGA hardware information, error reporting, etc.



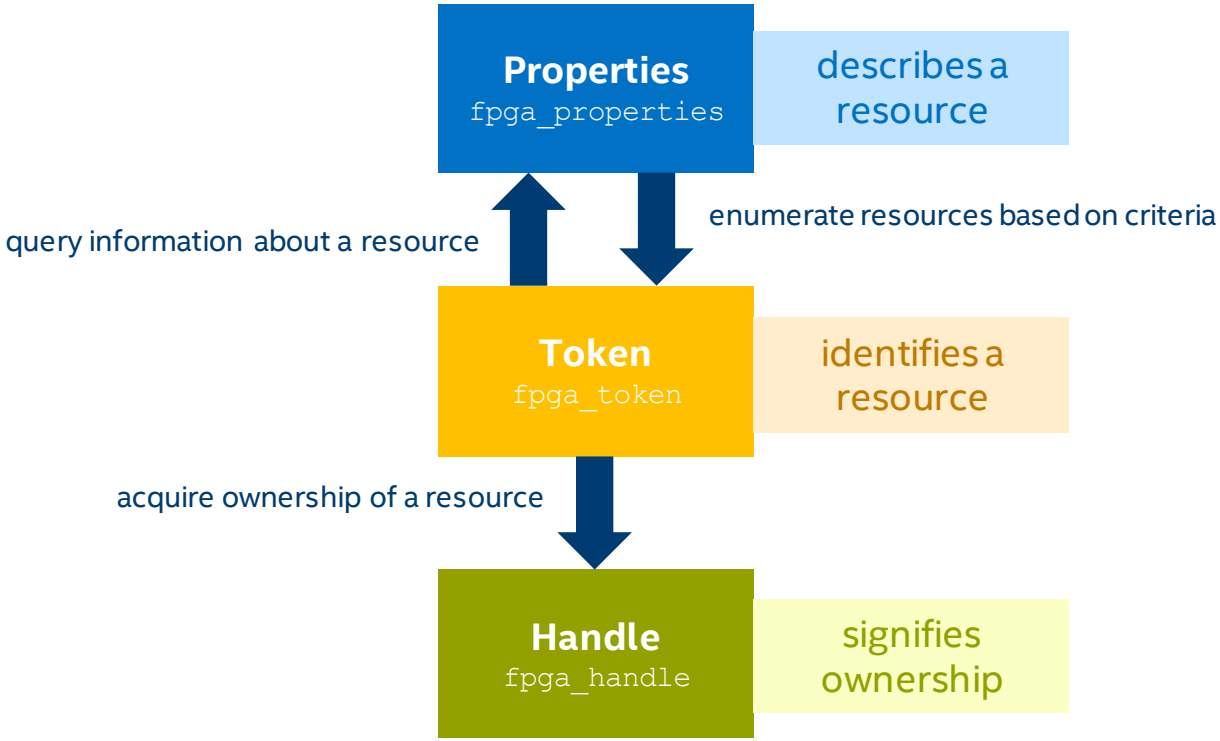
Application Development with OPAE



The OPAE Library Programming Model



The OPAE Object Model



fpga_properties Object

An opaque type for a properties object used by application to query and search for appropriate resources

2 Object types for FPGA resources

- FPGA_DEVICE
  - Corresponds to physical FPGA device
  - Can invoke management functions
- FPGA_ACCELERATOR
  - Represents an instance of an AFU

Defined in types.h file

Property	FPGA*	Accelerator*	Description
Parent	No	Yes	fpga_token of the parent object
ObjectType	Yes	Yes	The type of the resource: either FPGA_DEVICE or FPGA_ACCELERATOR
Bus	Yes	Yes	The bus number
Device	Yes	Yes	The PCI device number
Function	Yes	Yes	The PCI function number
SocketId	Yes	Yes	The socket ID
DeviceId	Yes	Yes	The device ID
NumSlots	Yes	No	Number of AFU slots available on an FPGA_DEVICE resource
BBSID	Yes	No	The FPGA Interface Manager (FIM) ID of an FPGA_DEVICE resource
BBSVersion	Yes	No	The FIM version of an FPGA_DEVICE resource
VendorId	Yes	No	The vendor ID of an FPGA_DEVICE resource
Model	Yes	No	The model of an FPGA_DEVICE resource
LocalMemory Size	Yes	No	The local memory size of an FPGA_DEVICE resource
Capabilities	Yes	No	The capabilities of an FPGA_DEVICE resource
GUID	Yes	Yes	The Global Unique Identifier of an FPGA_DEVICE or FPGA_ACCELERATOR resource
NumMMIO	No	Yes	The number of MMIO space of an FPGA_ACCELERATOR resource
NumInterrupts	No	Yes	The number of interrupts of an FPGA_ACCELERATOR resource
Accelerator State	No	Yes	The state of an FPGA_ACCELERATOR resource: either FPGA_ACCELERATOR_ASSIGNED or FPGA_ACCELERATOR_UNASSIGNED

*FPGA and Accelerator state whether or not the property is available for the FPGA or Accelerator objects

Creating fpga_properties object


```
fpga_result fpgaGetProperties(fpga_token token, fpga_properties *prop)
```

Initializes memory pointed at by *prop* to represent properties object

Populates with properties of the resource referred to by *token*

- Passing NULL *token* creates empty properties object which would match all FPGA resources in the enumeration query
- Refine query criteria using fpgaPropertiesSet*() functions

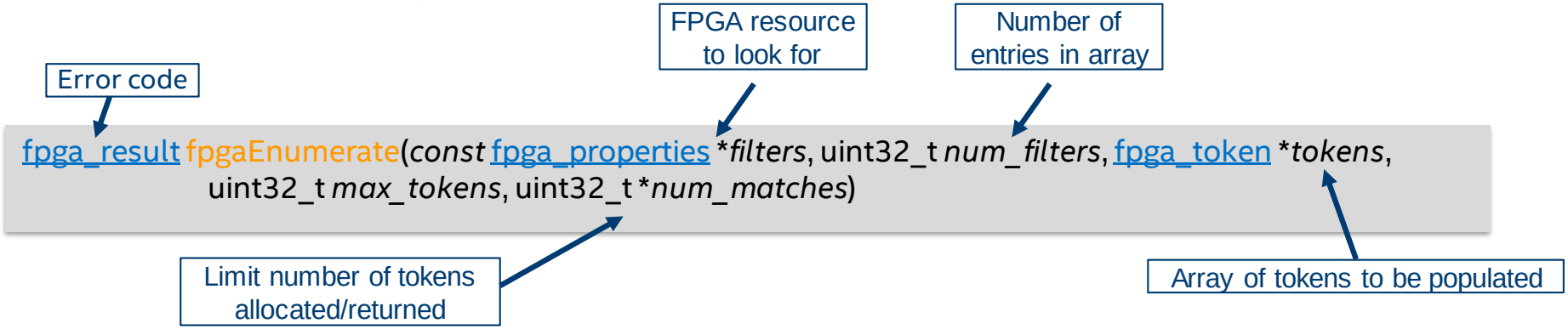
Individual properties can be queried using fpgaPropertiesGet*() accessor functions

Destroy fpga_properties object using fpgaDestroyProperties() function

Located in properties.h file

* Target property from fpga_properties object (i.e. fpgaSetObjectType()

Create an fpga_token



Function that searches for FPGA resources in system that match criteria (may be more than one)

- All accelerators assigned to a host interface, all FPGAs of a specific type, etc.

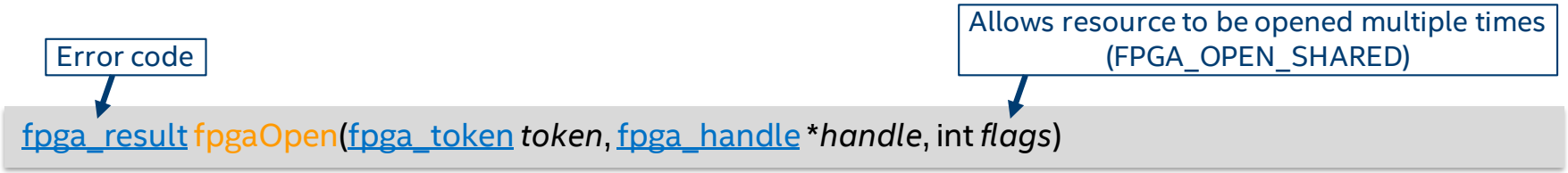
Creates fpga_token objects and populates the array with these tokens

- Number of tokens in the returned tokens array, either *max_tokens* or *num_matches* whichever is smaller

Free the memory with tokens no longer needed using the fpgaDestroyToken() function

Located in enum.h file

Receive fpga_handle



Acquires ownership of FPGA resource referenced by *token*

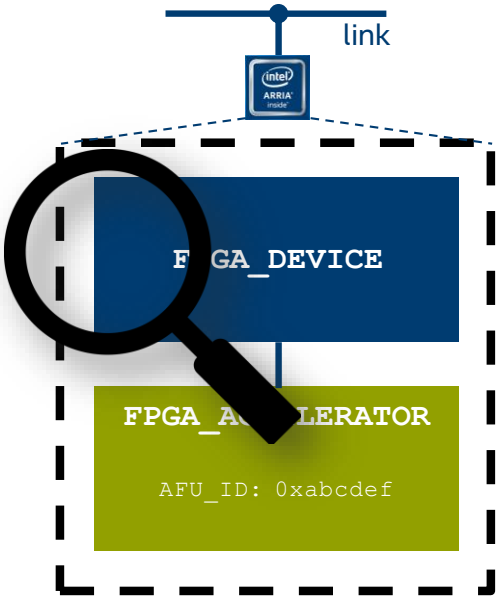
Ownership required to interact with accelerator function

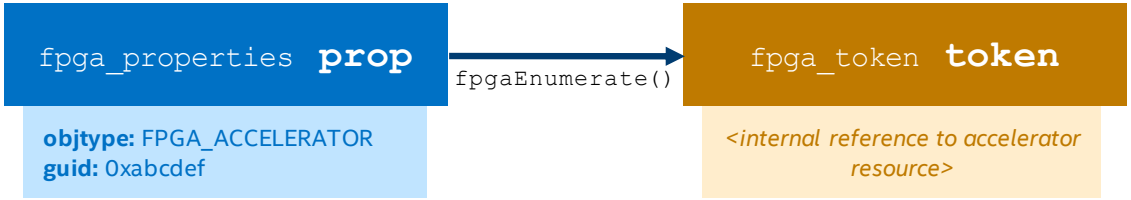
Remains open until fpga_Close() function called or process terminates

- Can also reset accelerator using fpga_Reset() function

Located in access.h file

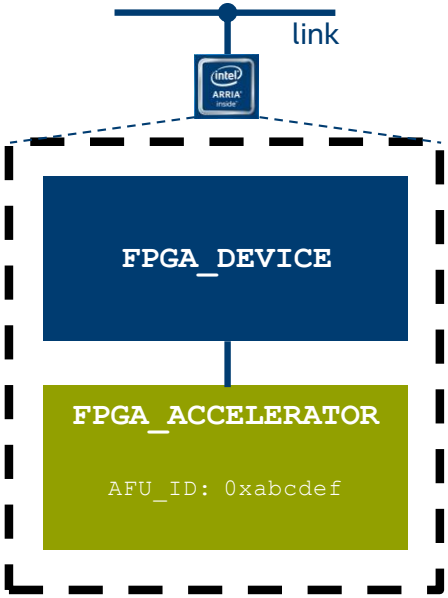
Enumeration and Discovery

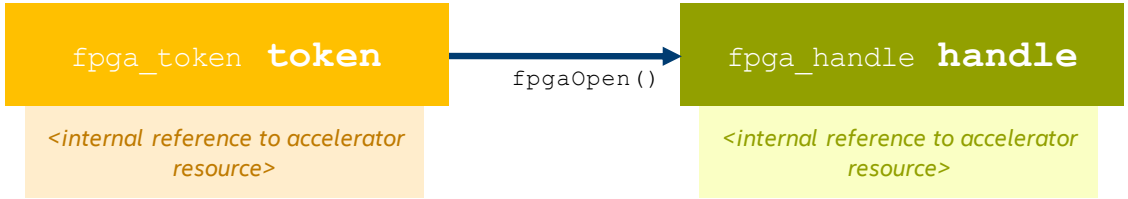





```
fpga_properties prop;
fpga_token      token;
fpga_guid       myguid;   /* 0xabcdef */

fpgaGetProperties(NULL, &prop);

fpgaPropertiesSetObjectType(prop, FPGA_ACCELERATOR);
fpgaPropertiesSetGUID(prop, myguid);

fpgaEnumerate(&prop, 1, &token, 1, &n);

fpgaDestroyProperties(&prop);
```

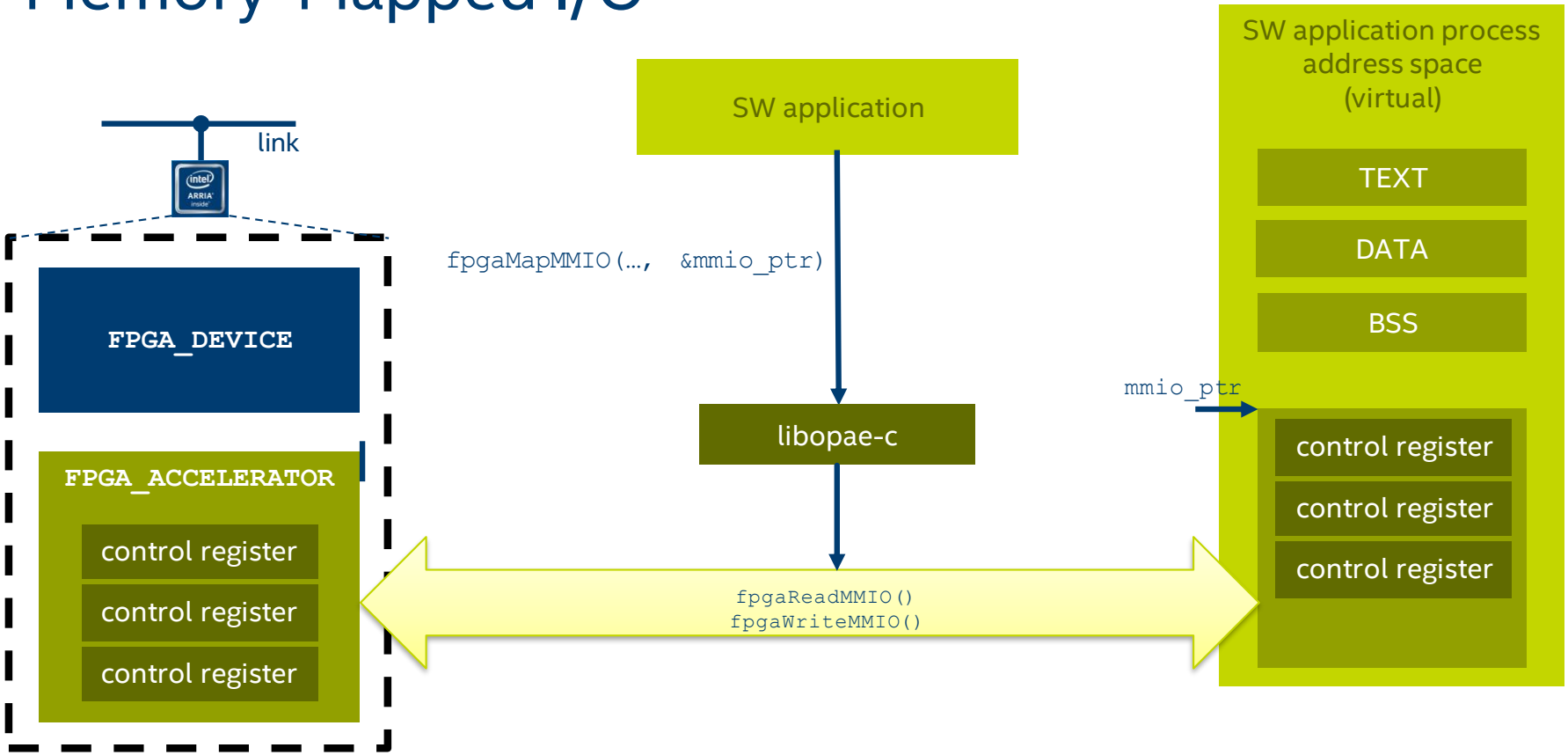
Acquire and Release Accelerator Resource



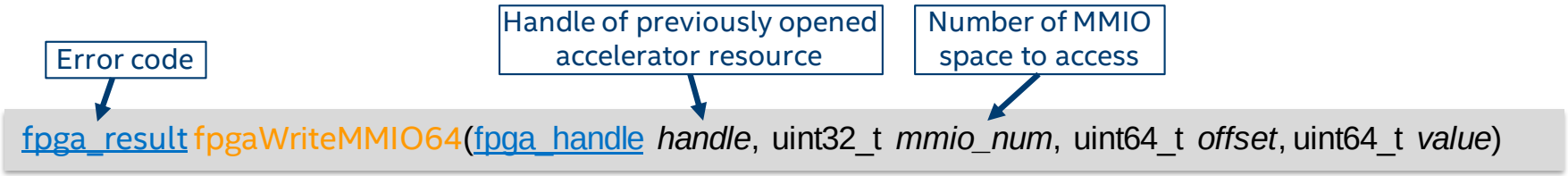



```
fpga_token token;
// ... enumeration ...
fpga_handle handle;

fpgaOpen(token, &handle, 0);
.
.
.
fpgaClose(handle);
```

Memory-Mapped I/O



MMIO Read/Write



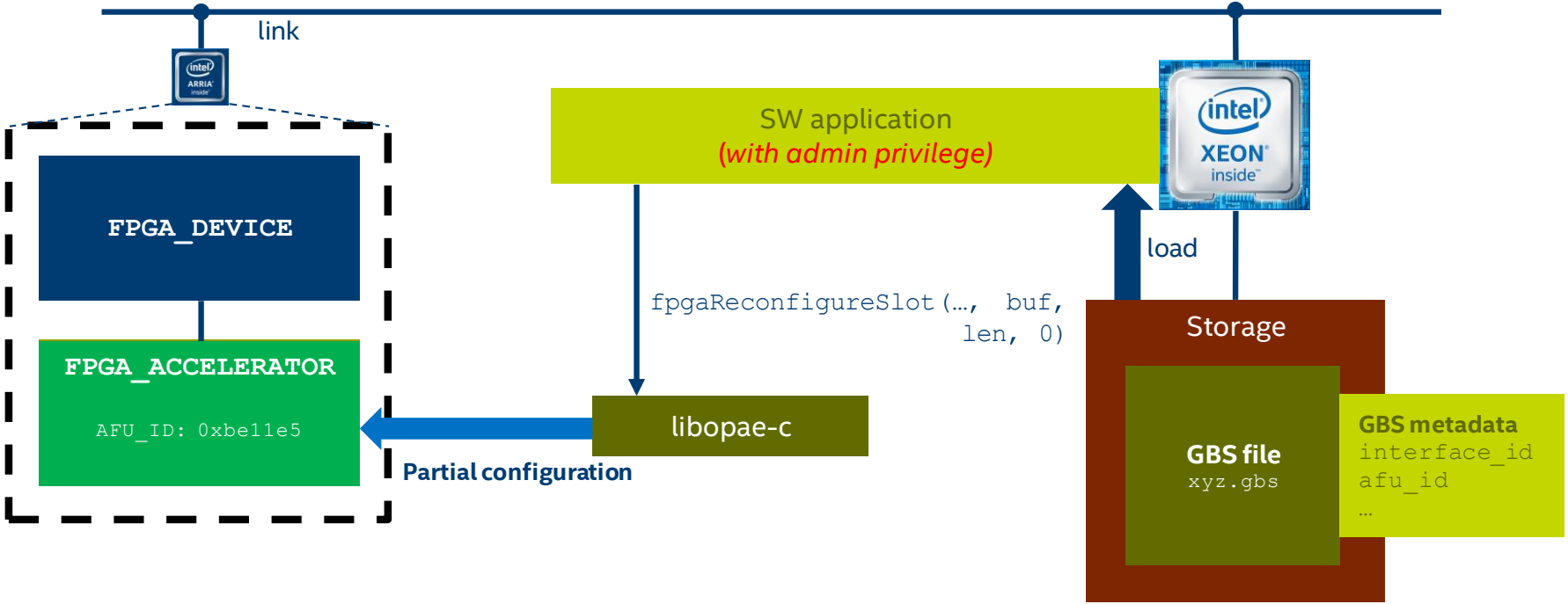
Performs 64bit MMIO space write of value to specified byte offset

- Also supports 32 bit MMIO writes using fpgaWriteMMIO32() function

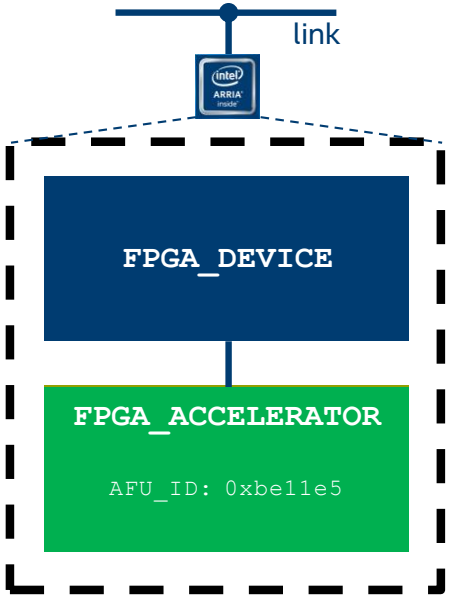
Reads use fpgaReadMMIO64() or fpgaReadMMIO32() functions

Located in mmio.h file

Management and Reconfiguration

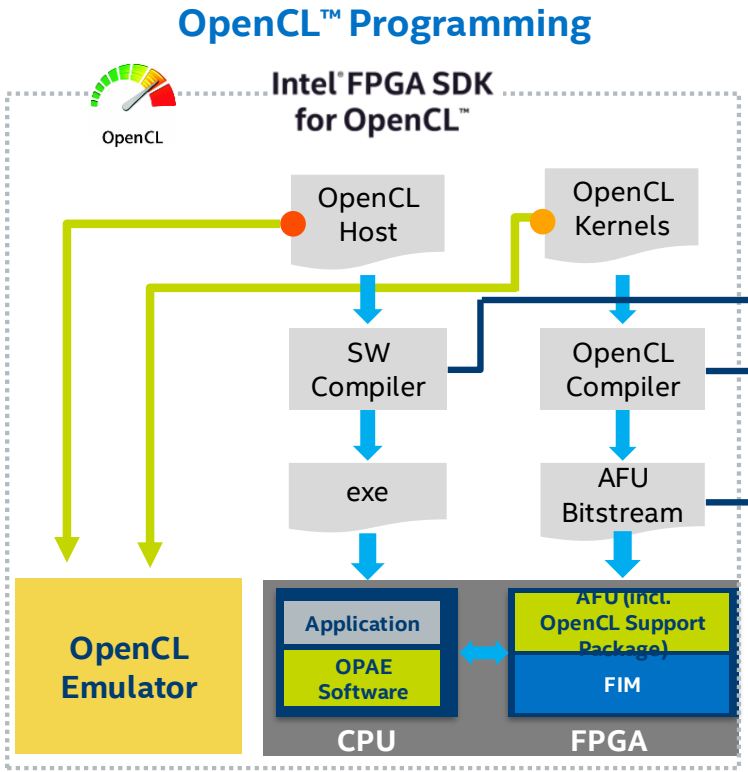


Management and Reconfiguration




```
fpga_handle  handle;          /* handle to device */
FILE        *gbs_file;
void        *gbs_ptr;
size_t       gbs_size;

/* Read bitstream file */
gbs_ptr = malloc(gbs_size);
fread(gbs_ptr, 1, gbs_len, gbs_file);

/* Program GBS to FPGA */
fpgaReconfigureSlot(handle, 0, gbs_ptr, gbs_size, 0);
/* ... */
```

OpenCL Development Approach



Standard C/C++ Compiler linked with

- Intel® FPGA OpenCL Library
- OpenCL Support Package for Acceleration Stack Library
- OPAE Library

Intel® FPGA Kernel Compiler (`aoc`)

- Compiles OpenCL kernel code
- Generates RTL
- Invokes Intel® Quartus Prime Design Software

FPGA Bitstream(`<file>.aocx`)

- Contains AFU .gbs file
- And other OpenCL kernel related information

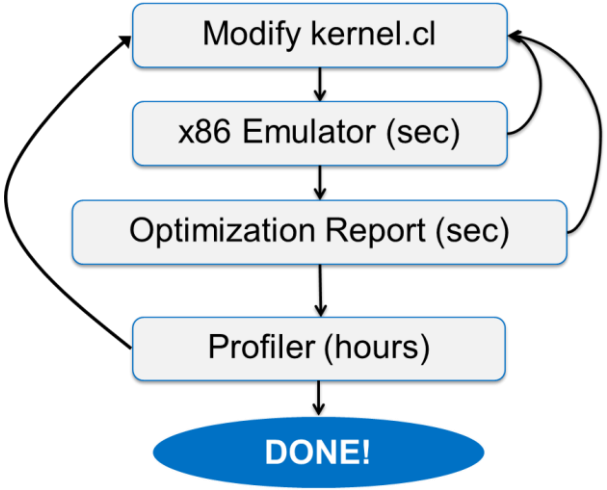
Software Application Flow using OpenCL™

No different from traditional OpenCL™ flow

- C based development and optimization flow to create AFUs and Host Application
- Standard OpenCL™ FPGA application using the Intel® FPGA SDK for OpenCL
  - FPGA OpenCL™ debug and profiling tools supported
- More information on using OpenCL with FPGAs

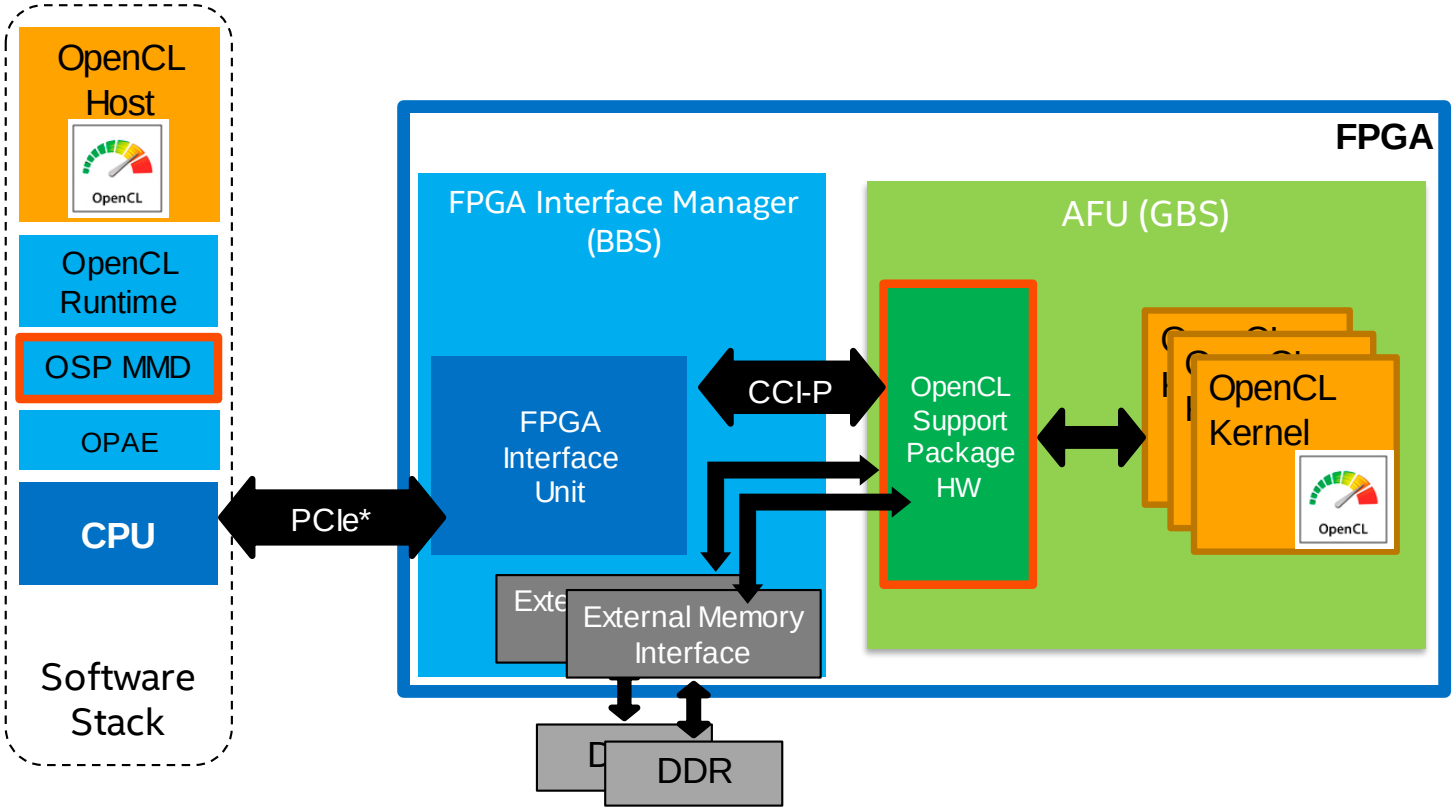
The Acceleration Stack abstracted away from user

- OPAE part of the Host Run-Time
  - Host does not need to interact with OPAE SW directly
- OpenCL™ Support Package(OSP) part of the FPGA Interface Manager
  - Kernel Avalon interface translated to CCI-P by the OSP



To learn more about using OpenCL with FPGAs, visit Intel FPGA Customer Training page

OpenCL™ Adds HW and SW Abstraction

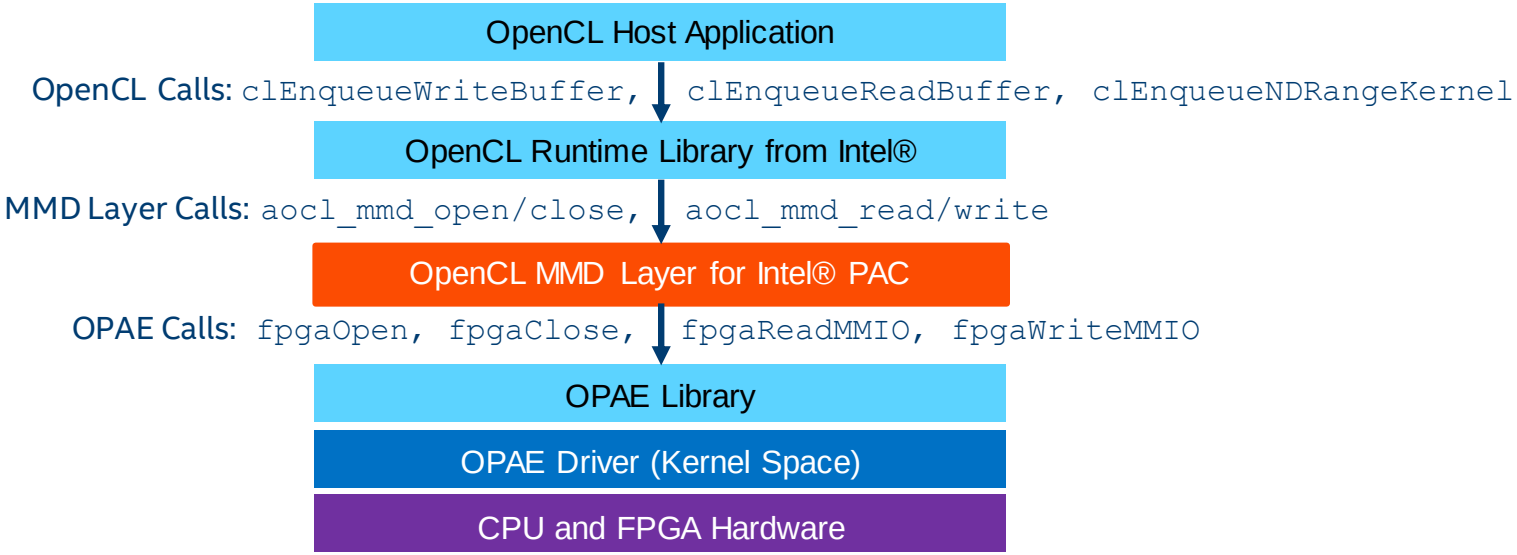


Software Stack and the OpenCL™ MMD Layer

Memory-Mapped Device (MMD) SW connects board driver to OpenCL runtime

- For the Acceleration Stack it links the OpenCL API to OPAE

Uses DMA BBB software driver unmodified and runs it in a separate thread.



OpenCL™ with Acceleration Stack Features

Standard OpenCL FPGA application can be used as is for the Acceleration Stack

- No code changes necessary, just switch the OSP in the compilation environment
  - Acceleration Stack functionality built into the OpenCL Support Package

Easily leverage the capabilities of the Acceleration Stack

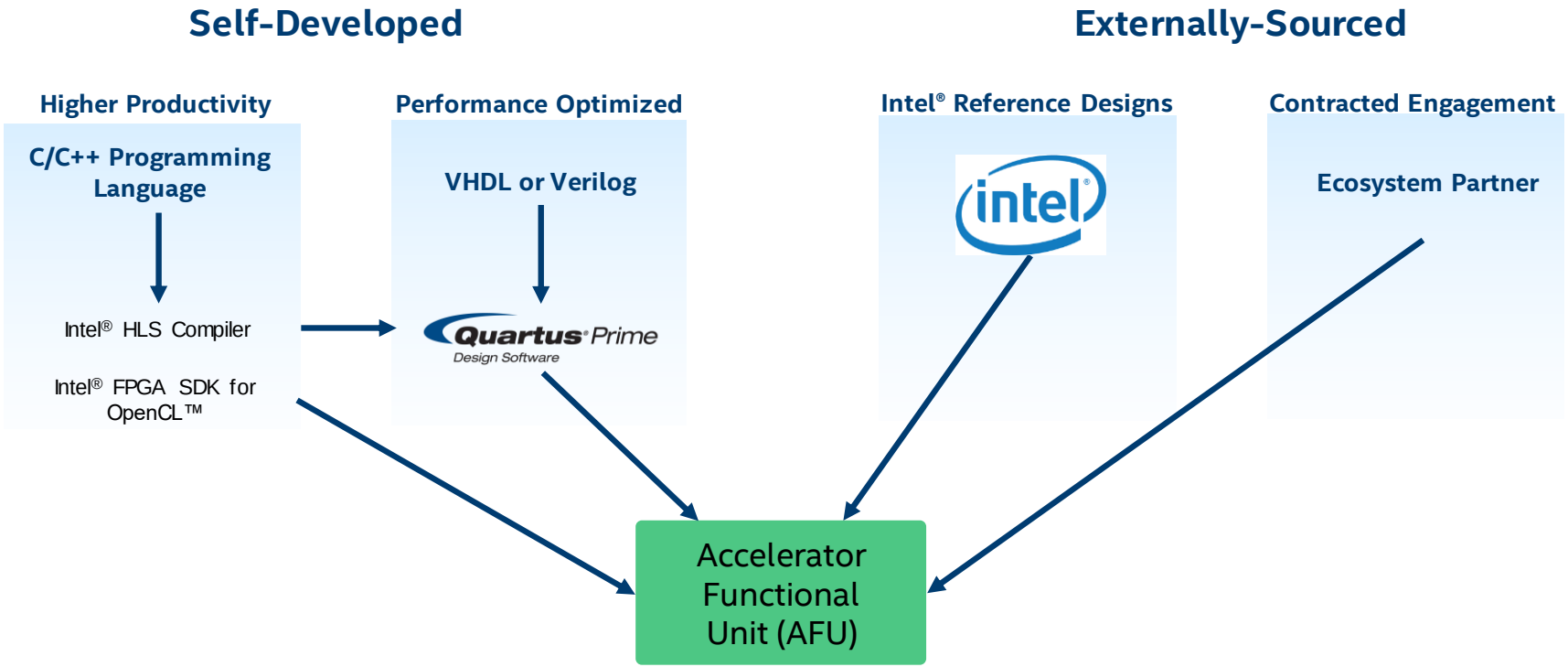
- Allow OpenCL to be used with virtualization
- No need develop RTL conforming to CCI-P interface
- No need to code at the lower OPAE level

Switch between OpenCL and Non-OpenCL Acceleration Stack applications without rebooting

- FPGA Interface Manager for both are the same


ACCELERATOR FUNCTIONAL UNITS
(AFU)

How Can FPGA Accelerators Be Created?

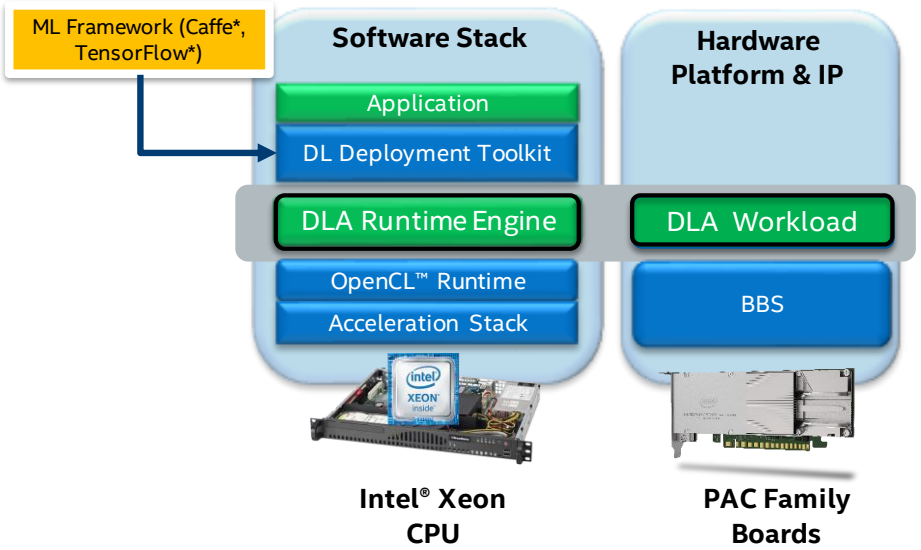


Growing List of Accelerator Solution Partners

Easing Development and Data Center Deployment of Intel FPGAs For Workload Optimization

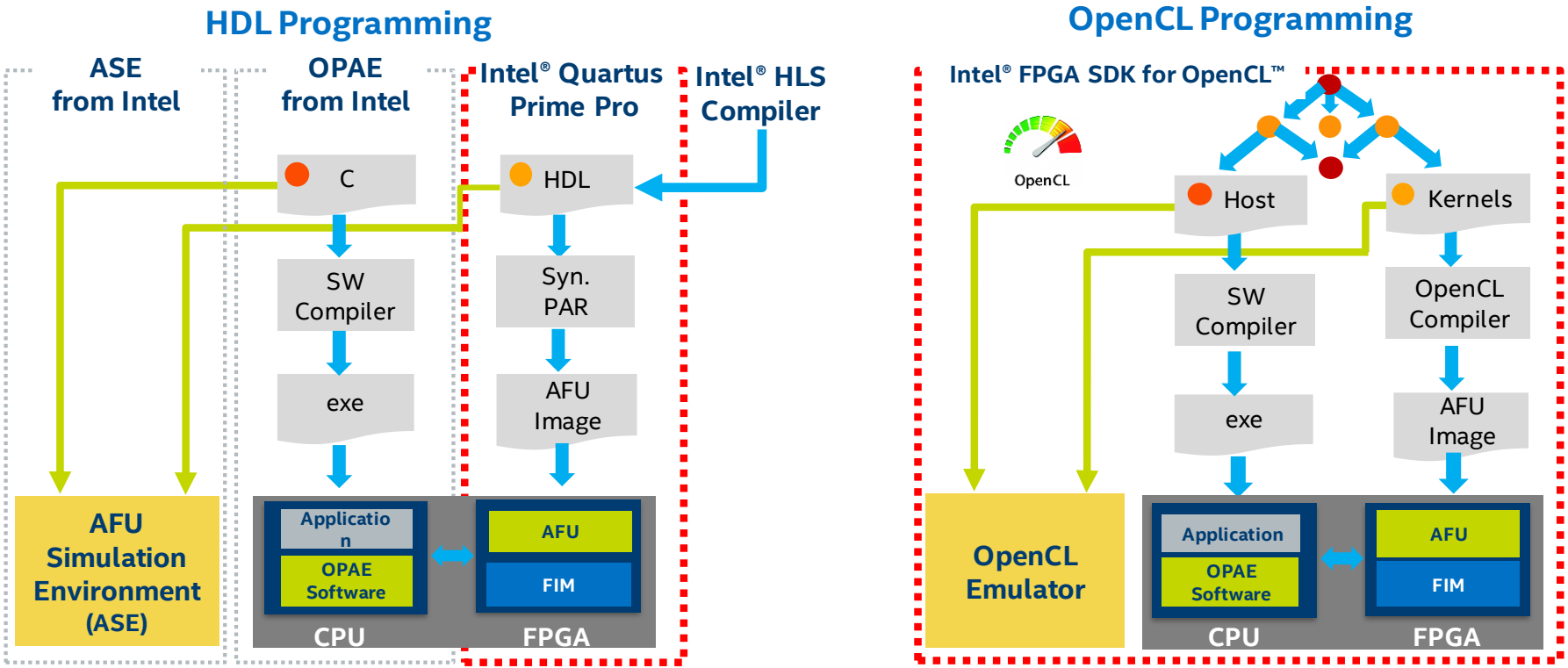
Machine Learning on Intel® FPGA Platform

Acceleration Stack Platform Solution

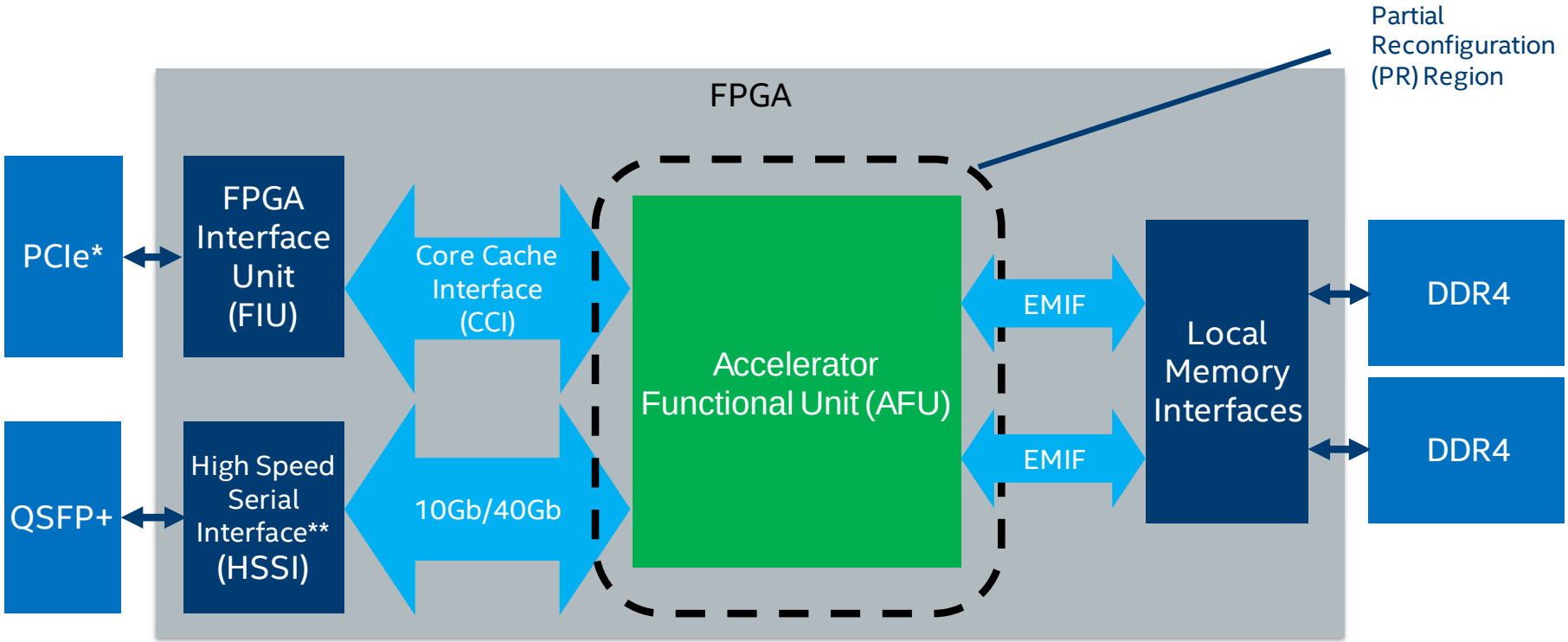


Ships with bitstreams for FPGA as part of OpenVINO™ toolkit

Accelerator Functional Unit Development

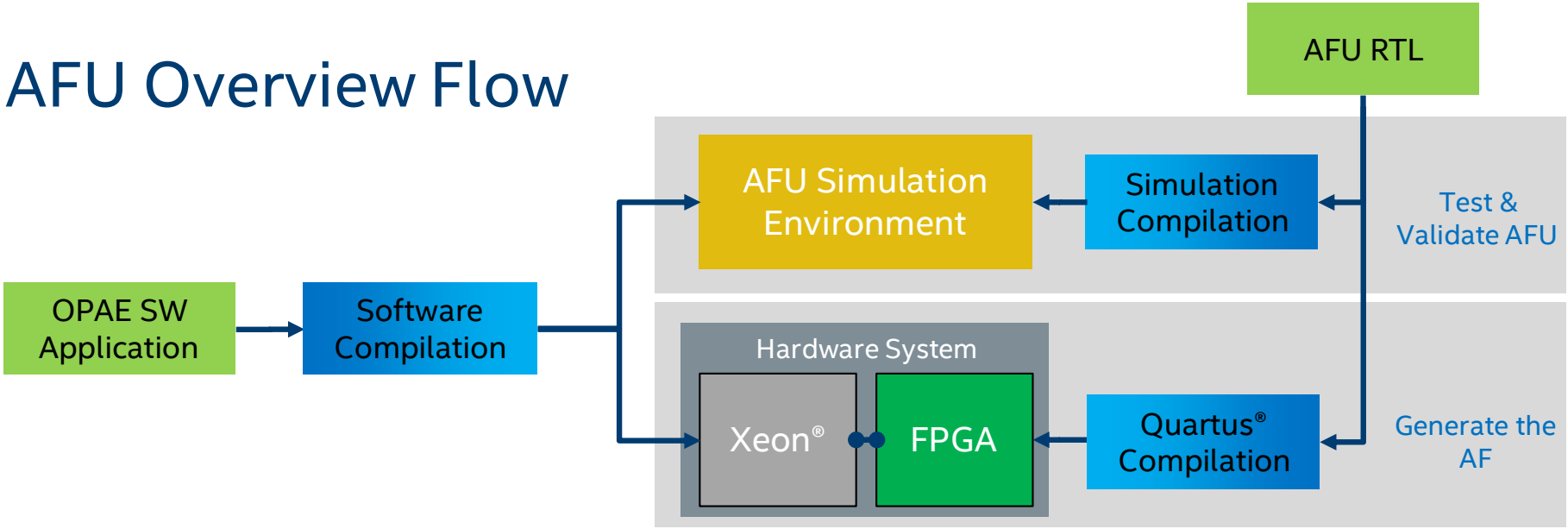


FPGA Components (Acceleration Stack v1.1)



- Could be other interfaces in the future (e.g. UPI)
- ** Available in v1.1 of Acceleration Stack

AFU Overview Flow

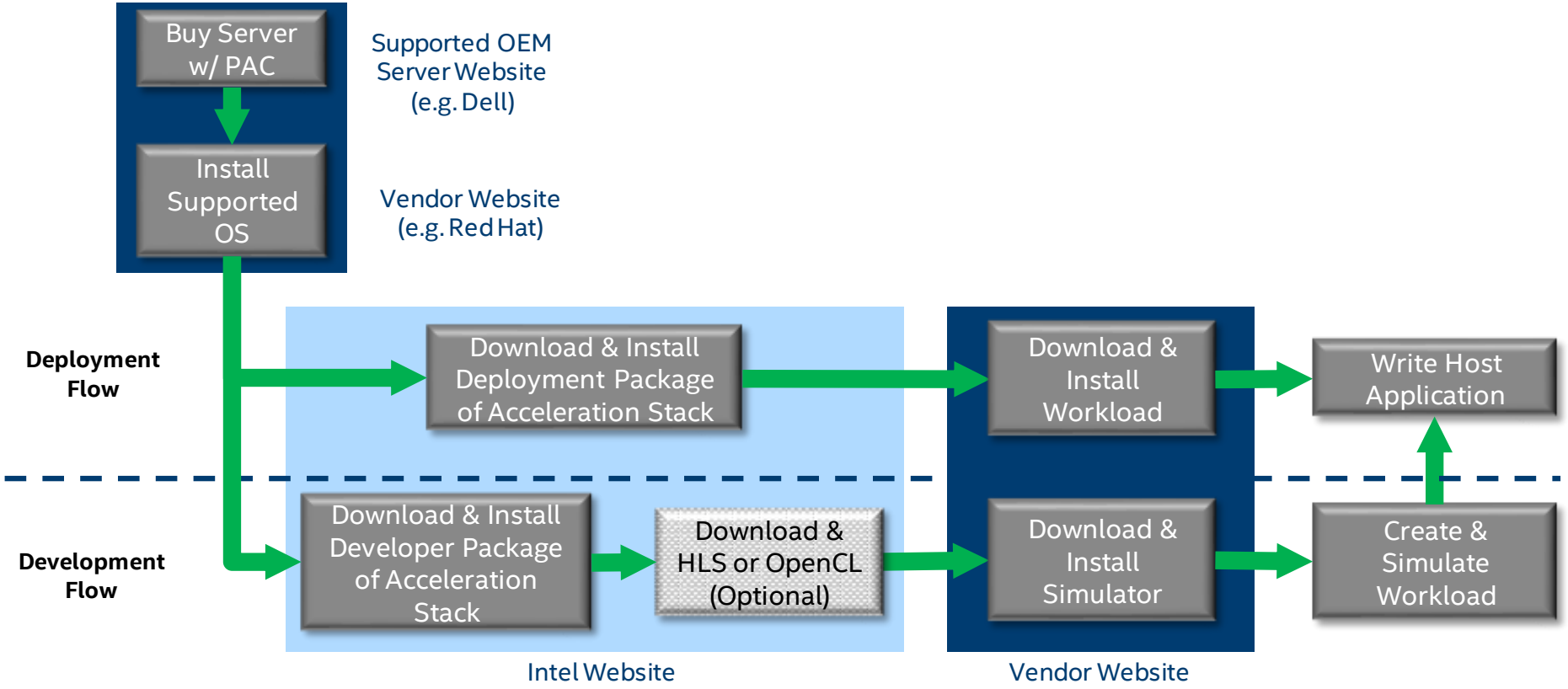


AF Simulation Environment (ASE) enables seamless portability to real HW

- Allows fast verification of OPAE software together with AF RTL without HW
  - SW Application loads ASE library and connects to RTL simulation
- For execution on HW, application loads Runtime library and RTL is compiled by Intel® Quartus into FPGA bitstream


GETTING STARTED

Out-of-Box Flow for Acceleration Stack



Download Acceleration Stack on Acceleration Hub

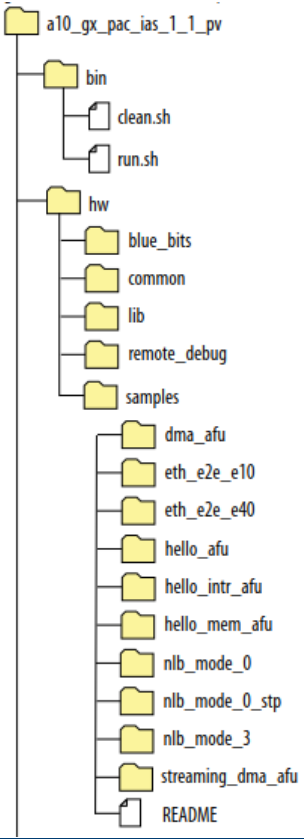
Overview | Platforms | Acceleration Stack | Downloads | Solutions | Partners | Support | Academia

In today's world, the number of connected devices and the amount of data continues to increase every day. The rate at which data arrives from these devices into data centers also continues to increase. By leveraging Intel® FPGAs as accelerators, a wide range of workloads can be enhanced to accommodate this increased data, and new demands for analyzing it.

FPGAs are silicon devices that can be dynamically reprogrammed with a datapath that exactly matches your workloads, such as data analytics, image inference, encryption, and compression. This versatility enables the provisioning of a faster processing, more power efficient, and lower latency service – lowering your total cost of ownership, and maximizing compute capacity within the power, space, and cooling constraints of your data centers.

Traditionally, FPGAs require deep domain expertise to program for, but the Intel® Acceleration Stack for Intel Xeon® CPU with FPGAs simplifies the development flow and enables rapid deployment across the data center. Intel is partnering with FPGA intellectual property (IP) developers, server original equipment manufacturers (OEMs), virtualization platform providers, operating system (OS) vendors, and system integrators to enable customers to efficiently develop and operationalize their infrastructure.

Example Designs to Get Started



Example	Description
Hello AFU	Simple AFU with direct CCI connection for MMIO access
Hello Intr AFU	Example use of user interrupts
Hello Mem AFU	Example showing using USR Clock to auto close timing in the AFU
DMA AFU	Example DMA AFU to move data between host memory and local FPGA memory. Uses BBB and bridges Avalon to CCI
Streaming DMA AFU	Example DMA AFU to move data between host memory and the AFU directly as a streaming packet
Eth e2e e10	10Gb Ethernet loopback design
Eth e2e e40	40Gb Ethernet loopback design
NLB mode 0	Native LoopBack adaptor (rd/wr) with more features
NLB mode 0 stp	Native LoopBack adaptor with SignalTap remote debug
NLB mode 3	Native LoopBack adaptor (rd/wr)

Recommended Getting Started Process

Install VM with Linux and install Acceleration Stack Development Flow

Run through quick start guide to validate environment

- Run ASE on Hello AFU
- Regenerate hello_afu .gbs file and host software application

If applicable, run the OpenCL example

Examine the DMA_AFU and/or Streaming_DMA_AFU example to understand how to move data using DMA with AFU

Create your own by modifying one of the example designs (e.g. DLA through OpenVINO™ toolkit)

Follow-On Training:

Online Training Course

- Introduction to the Acceleration Stack for Intel® Xeon w/ FPGA
- OpenCL™ Development with the Acceleration Stack
- RTL development and acceleration with the Acceleration Stack
- Application Development on the Acceleration Stack for Intel® Xeon® CPU with FPGAs
- Introduction to High-Level Synthesis (7 courses)
- Introduction to Parallel Computing w/ OpenCL on FPGAs
- Deploying Intel FPGAs for Inferencing with OpenVINO Toolkit
- Programmers' Introduction to the Intel® FPGA Deep Learning Acceleration Suite

Instructor Led Training Courses

- Introduction to High-Level Synthesis with Intel® FPGAs
- High-Level Synthesis Advanced Optimization Techniques
- Introduction to OpenCL
- Optimizing OpenCL™ for Intel® FPGAs (16 Hours Course)

https://www.altera.com/support/training/overview.html

Summary

Acceleration Stack for Intel Xeon CPU with FPGAs enables seamless integration of the FPGA accelerators into clusters

DLA Suite comes with OpenVINO as bitstreams for the PAC cards that runs on the acceleration stack

The OPAE software layer makes adoption of the FPGA into the host application code seamless and

The acceleration stack makes building accelerator functions much faster and board agnostic through the use of a standardized FIM

Legal Disclaimers/Acknowledgements

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at www.intel.com.

Intel, the Intel logo, Intel Inside, the Intel Inside logo, MAX, Stratix, Cyclone, Arria, Quartus, HyperFlex, Intel Atom, Intel Xeon and Enpirion are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

OpenCL is the trademark of Apple Inc. used by permission by Khronos

*Other names and brands may be claimed as the property of others


intel®
experience
what's inside™