SDAccel Tool Overview

SDX 2018.2

XILINX.

© Copyright 2018 Xilinx

Objectives

- **After completing this module, you will be able to:**
  - List language support SDAccel provides
  - Describe OpenCL execution model in host application
  - List underlying tool technologies SDAccel uses

© Copyright 2018 Xilinx

XILINX.

Outline

- SDAccel Overview
- Host code: OpenCL execution model
- Makefile Flow
- Summary
- Lab Intro

© Copyright 2018 Xilinx

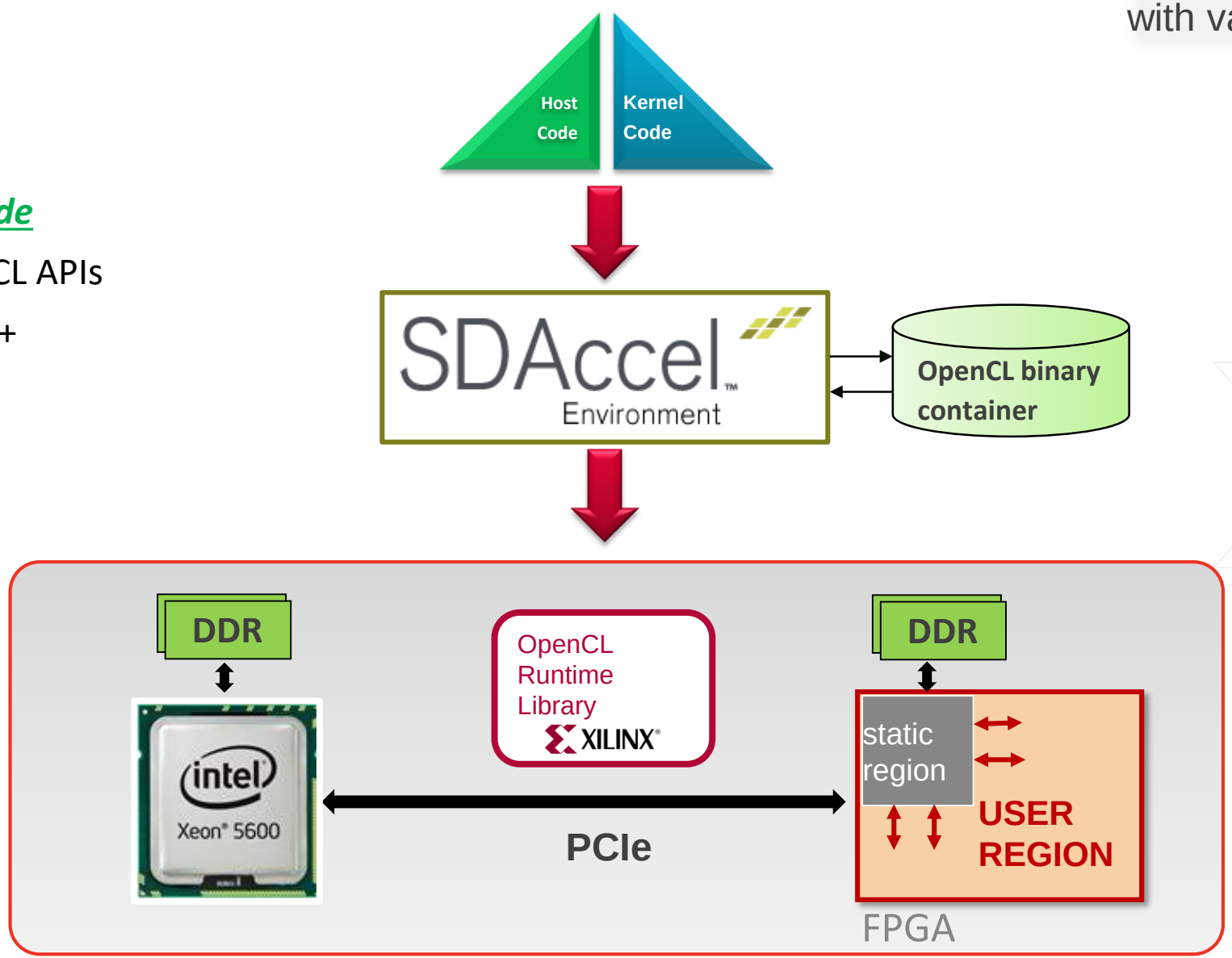
SDAccel Development Environment

SDAccel supports
OpenCL 1.0 Embedded Profile
with various 1.2 / 2.0 features

Host code

– OpenCL APIs

– C / C++



Kernel code

– OpenCL kernel code

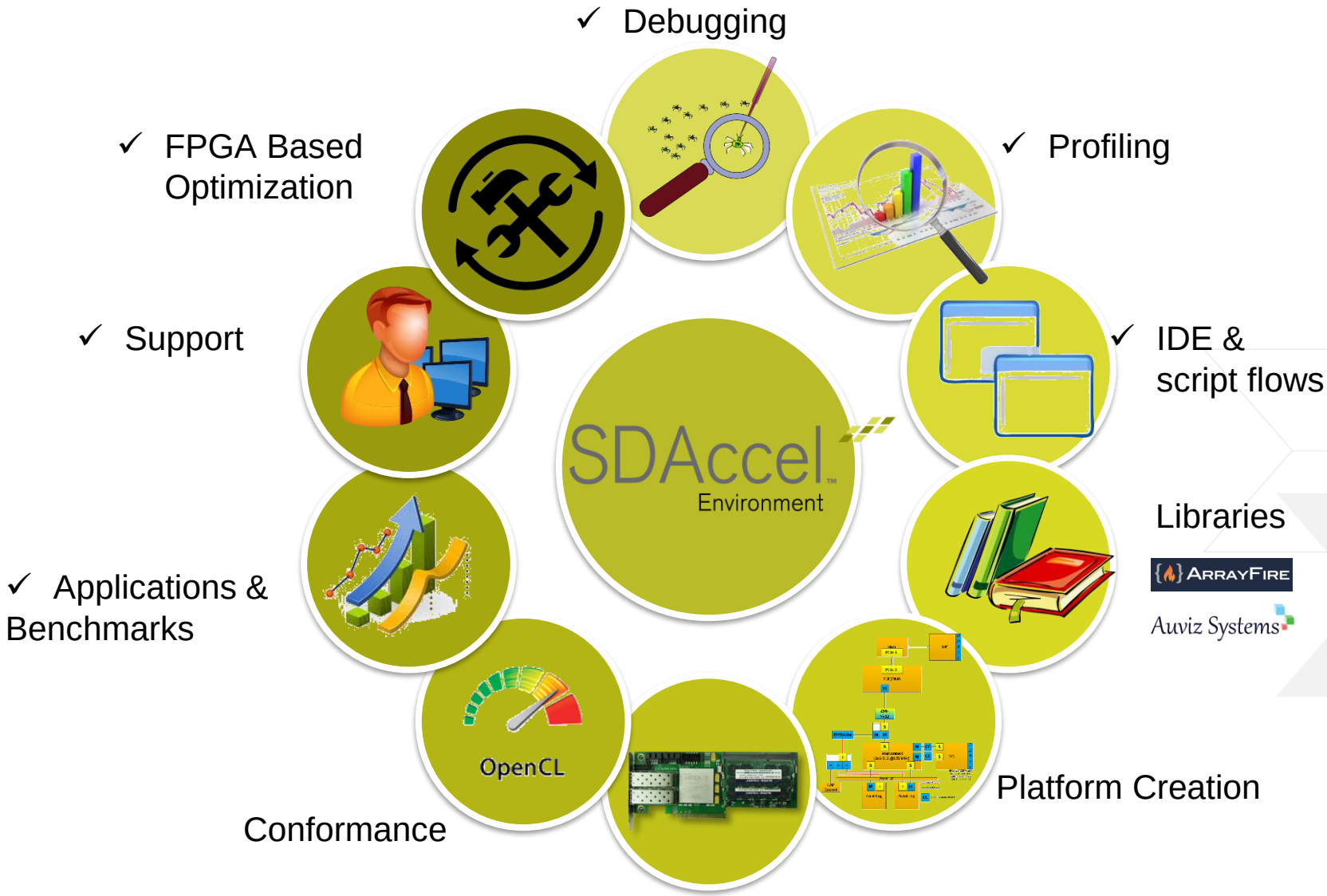
– C/C++

– RTL IP

© Copyright 2018 Xilinx

XILINX

SDAccel – The Complete Development Environment



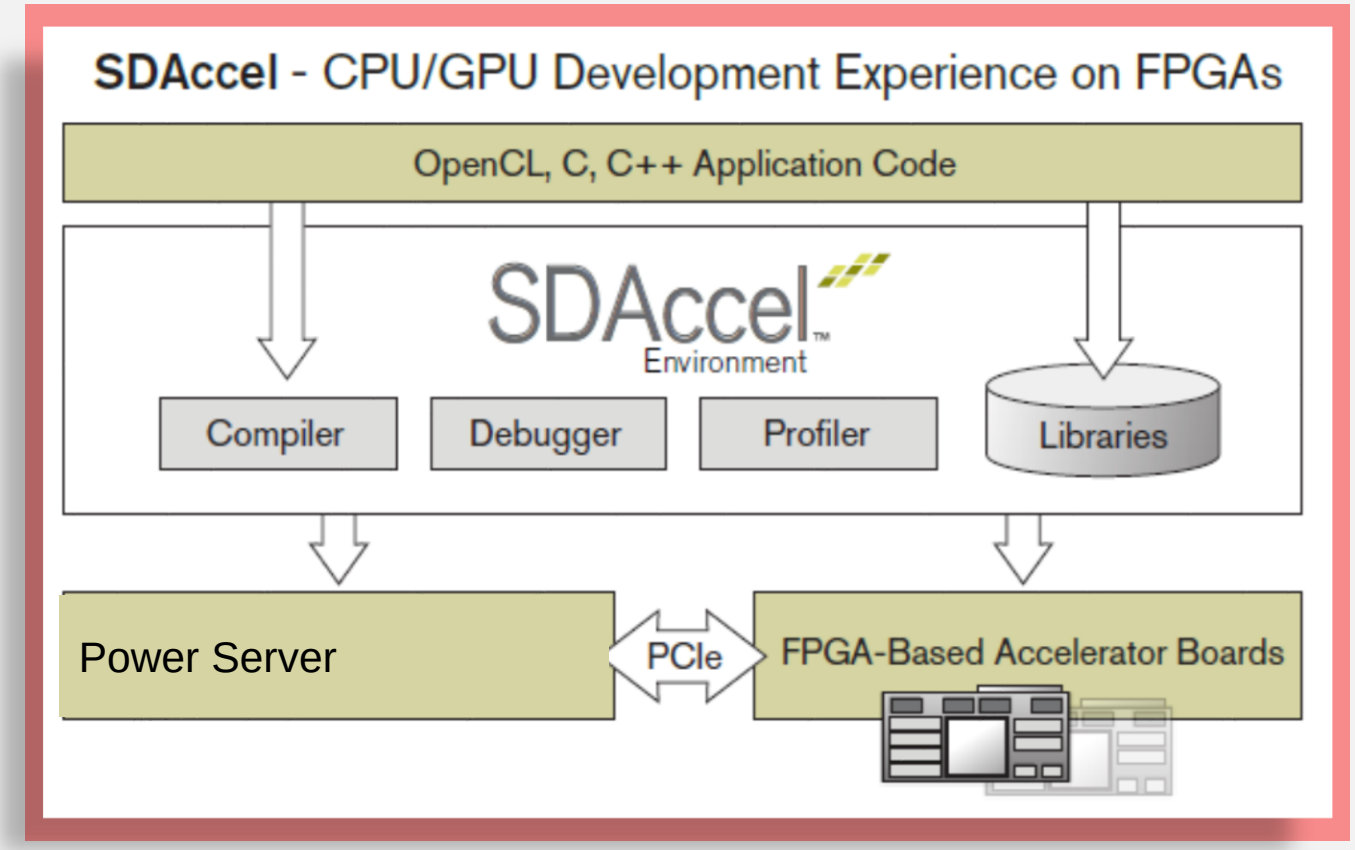
© Copyright 2018 Xilinx

XILINX.

SDAccel Environment and Ecosystem

Application Developers

ArrayFire · Falcon Computing · impulse accelerated technologies · 聯科集團 CLUSTERTECH · ICHEC Irish Centre for High-End Computing · Instigate Design · MULTICOREWARE



Cloud Providers

SV · IBM · NIMBIX · aws

Library Providers

Falcon Computing · ArrayFire

Develop with Xilinx Board

Deploy with Board Partners

ALPHA DATA · TUL™ · Semptian 恒扬数据 · COTS Technology

© Copyright 2018 Xilinx

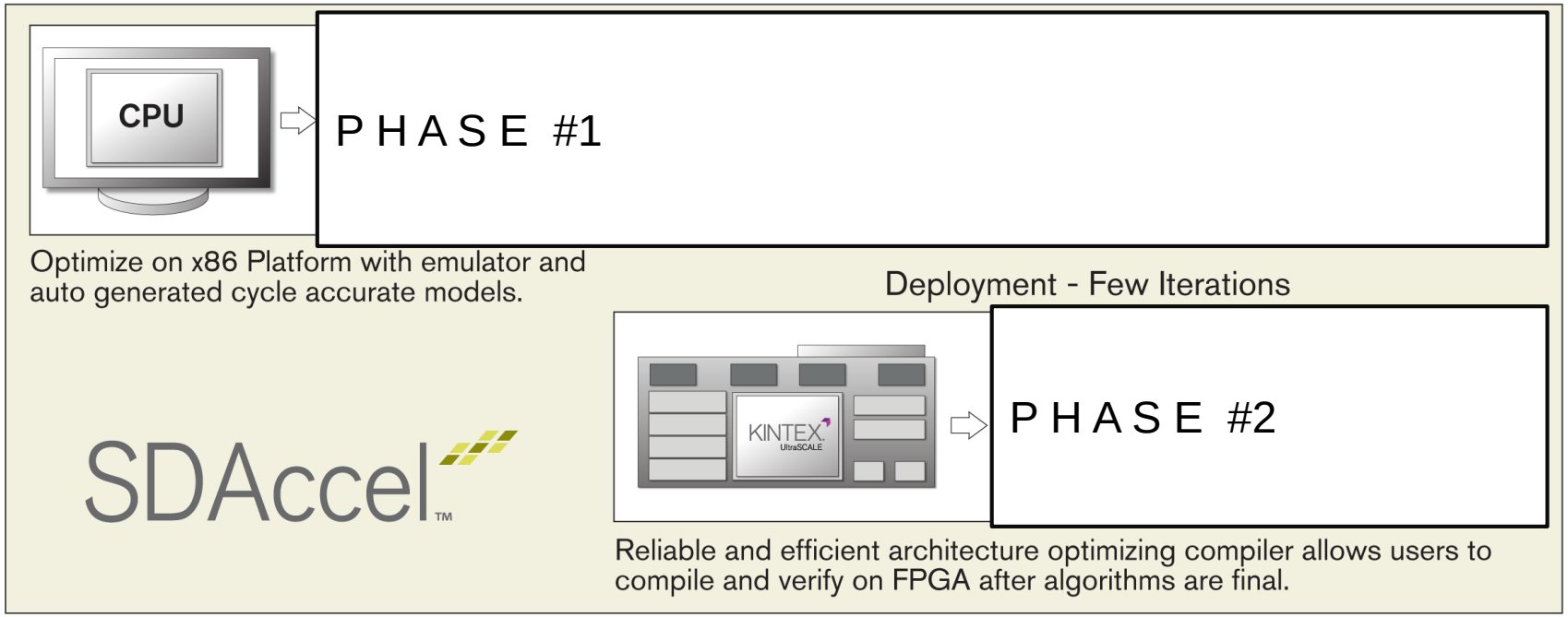
XILINX

SDAccel Flow

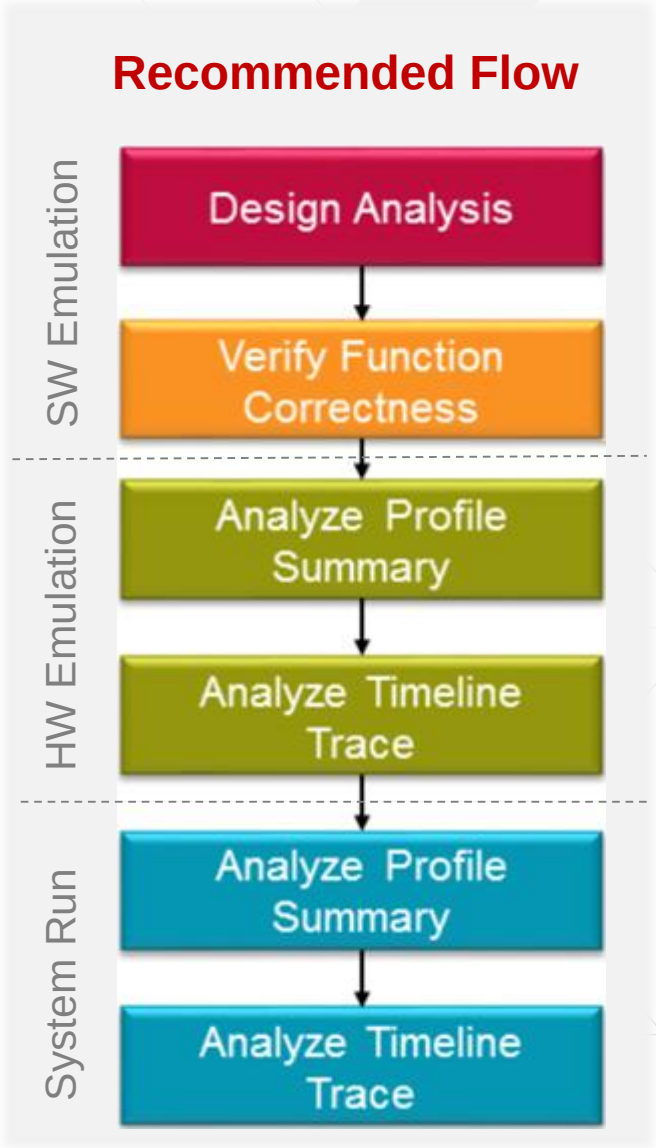
SW Emulation

Hardware Emulation

SDAccel - Accelerated OpenCL Programming and Deployment



System Run



*Note: SDAccel can be run using **Makefile**, **GUI***

© Copyright 2018 Xilinx

XILINX

Host Code
OpenCL Execution Model

© Copyright 2018 Xilinx

OpenCL Execution Model

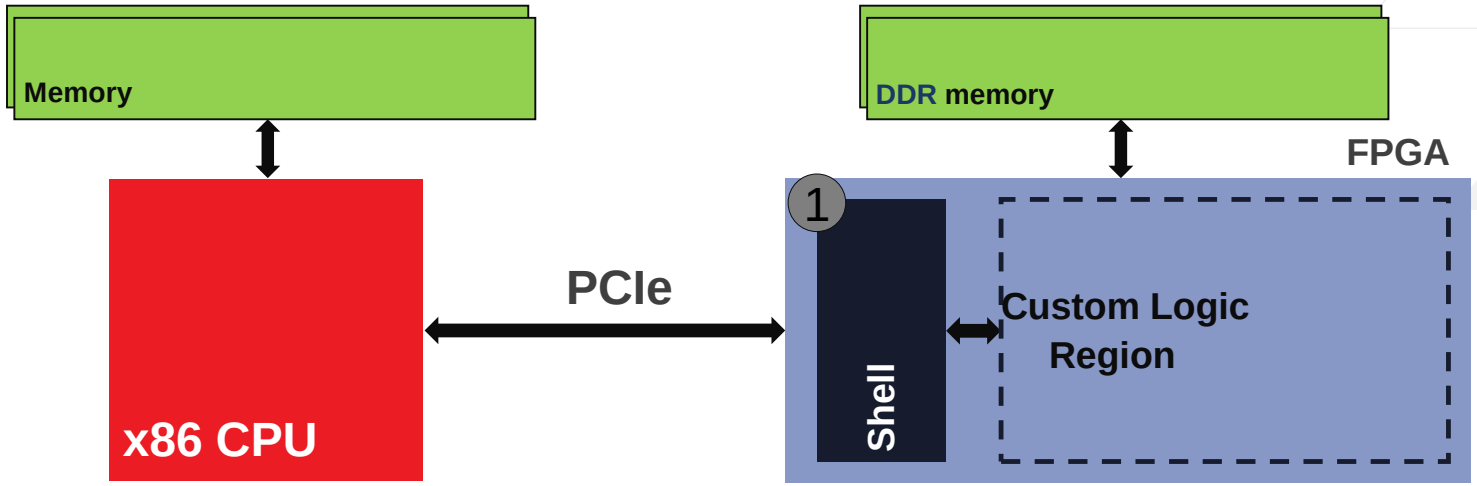
- **OpenCL API provides a high-level description of the key steps of an application executing on F1**

1. **Powering-Up**
2. **Runtime Initialization**
3. **Device Configuration**
4. **Buffer Allocation**
5. **Writing Buffers to FPGA Memory**
6. **Running the Accelerators**
7. **Reading Buffers from FPGA Mem**

© Copyright 2018 Xilinx

1. Powering-Up

- On power-up, the FPGA is initialized
- At this stage, the only logic in the FPGA is the Shell
- The Shell will be managing the communications with the host



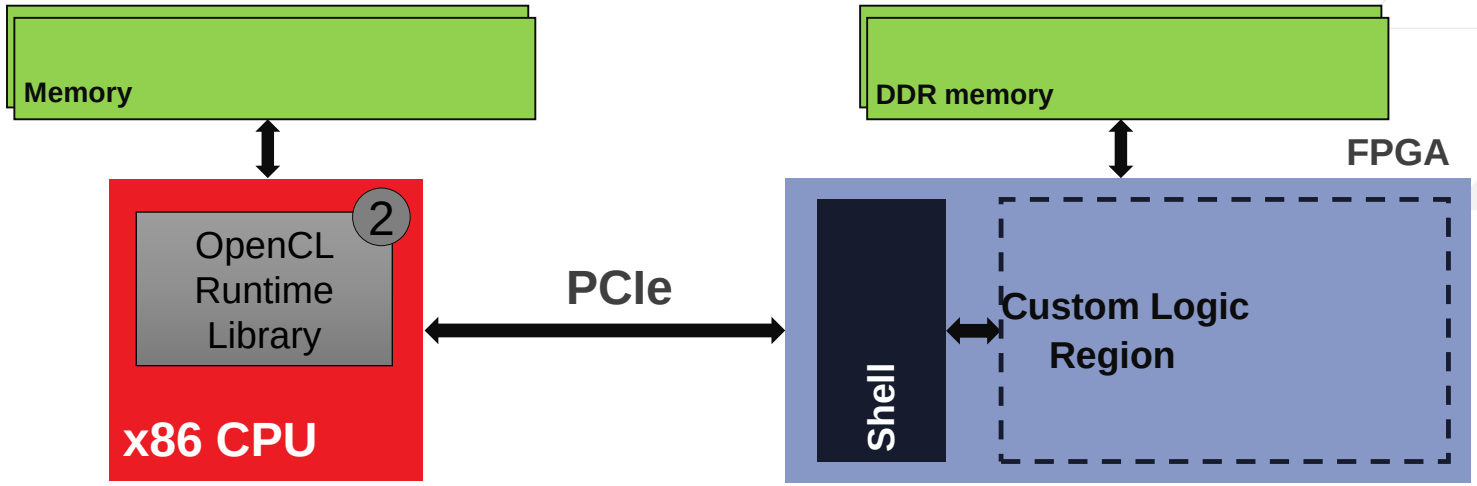
© Copyright 2018 Xilinx

2. Runtime Initialization

- **Creation of OpenCL context and device**
  - Context → Platform
  - Device → FPGA
- **Creation of OpenCL command queues used to send commands to the FPGA**

```
context = clCreateContextFromType(…);
clGetDeviceIDs(…, &device_id, …);

queue = clCreateCommandQueue(context, device_id, …);
```

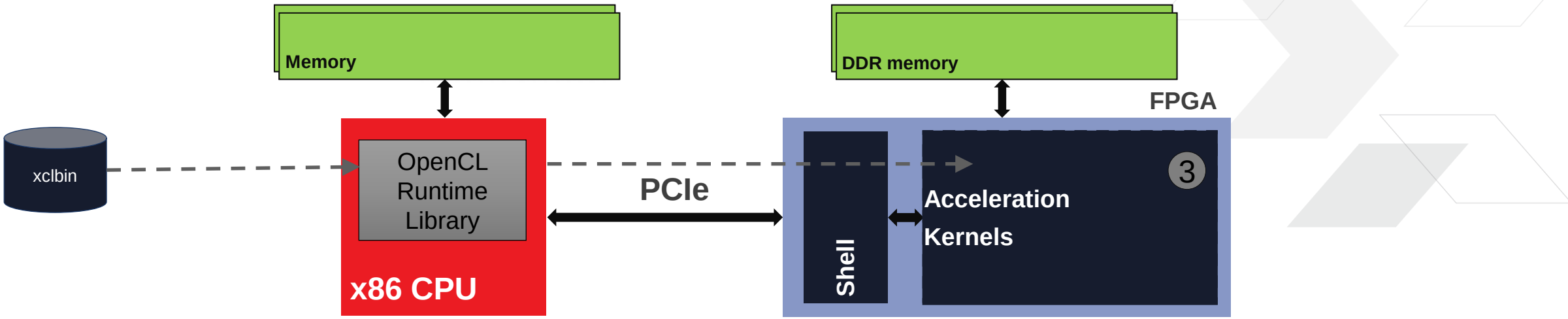


© Copyright 2018 Xilinx

3. Device Configuration

- **Host programs the FPGA by calling clCreateProgramWithBinary**
  - Loads the .xclbin file

```
program = clCreateProgramWithBinary(…);
```

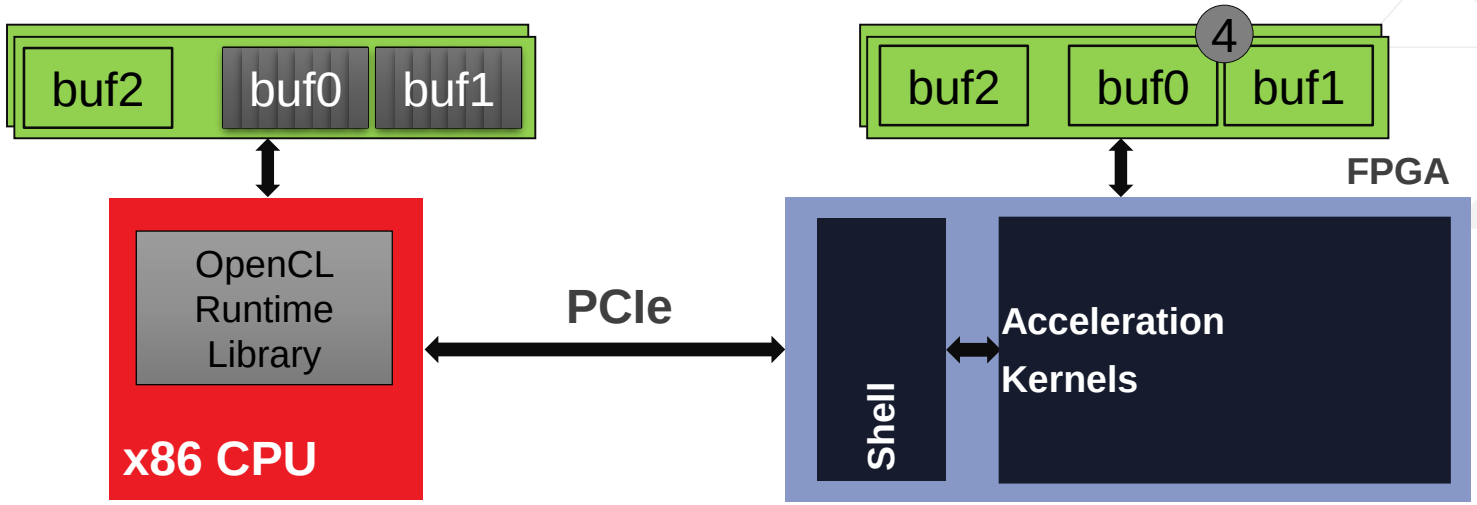


© Copyright 2018 Xilinx

4. Buffer Allocation

- Host allocates buffers in the device
- Buffers are used to transfer data from the CPU to the FPGA and back

```
buf0 = clCreateBuffer(context, CL_MEM_READ_ONLY, …);
buf1 = clCreateBuffer(context, CL_MEM_READ_ONLY, …);
buf2 = clCreateBuffer(context, CL_MEM_WRITE_ONLY, …);
```

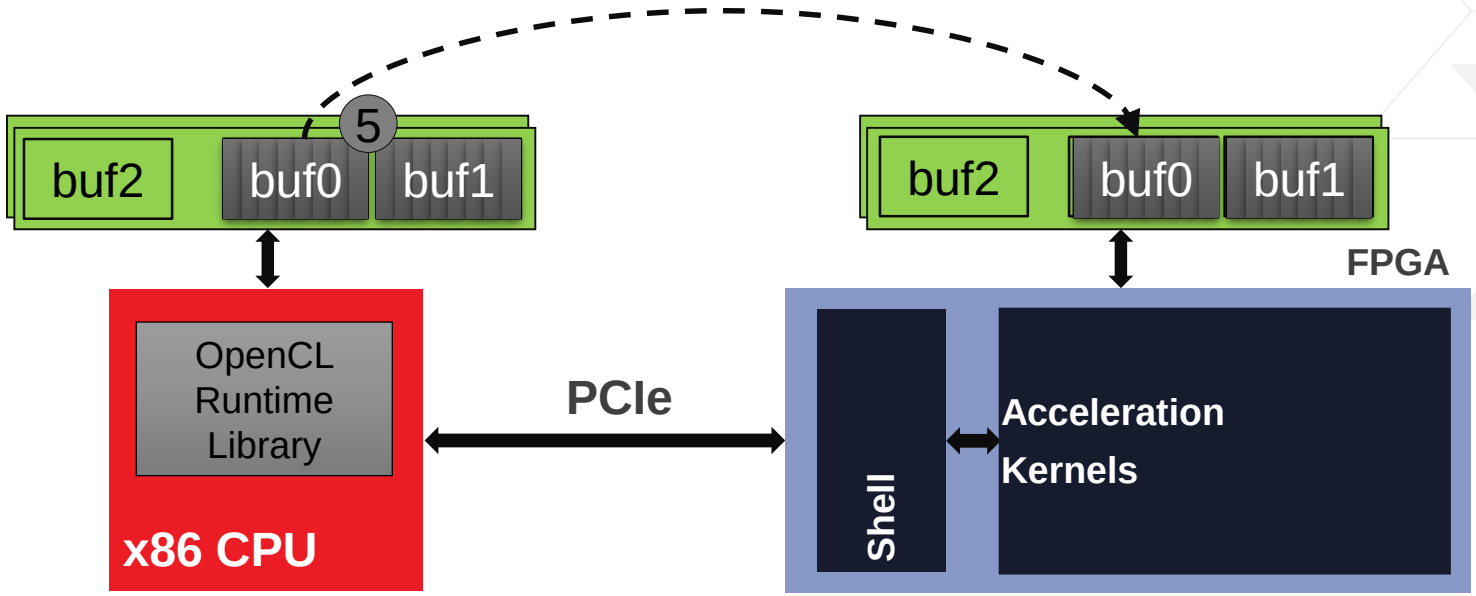


© Copyright 2018 Xilinx

5. Writing Buffers to FPGA Memory

- **Host copies data to be processed from local memory to the buffer in the FPGA DDR memory**

```
clEnqueueWriteBuffer(queue, buf0, …);
clEnqueueWriteBuffer(queue, buf1, …);
```

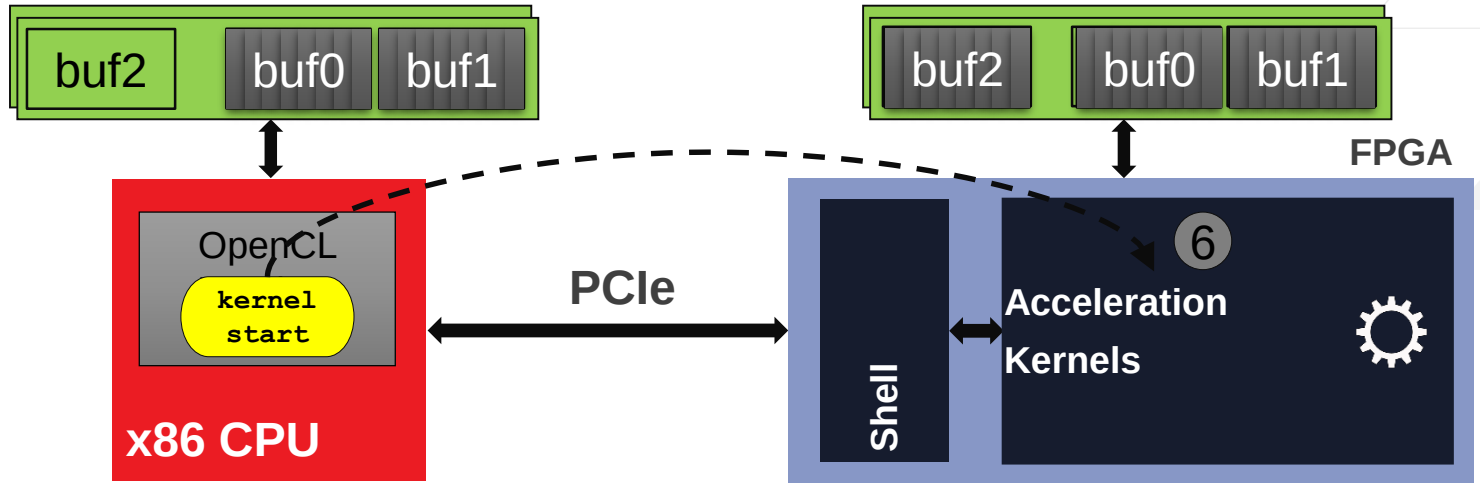


© Copyright 2018 Xilinx

6. Running the Accelerators

- Host schedules execution of the desired kernel with clEnqueueTask
- Runtime starts the Kernel
- Kernel processes data previously copied to from host buffer to DDR

```
kernel = clCreateKernel(program, "mykernel", …);
clSetKernelArg(kernel, 0, sizeof(cl_mem), &buf0);
clSetKernelArg(kernel, 1, sizeof(cl_mem), &buf1);
clSetKernelArg(kernel, 2, sizeof(cl_mem), &buf2);
err = clEnqueueTask(queue, kernel, 0, NULL, NULL);
clFinish(queue);
```



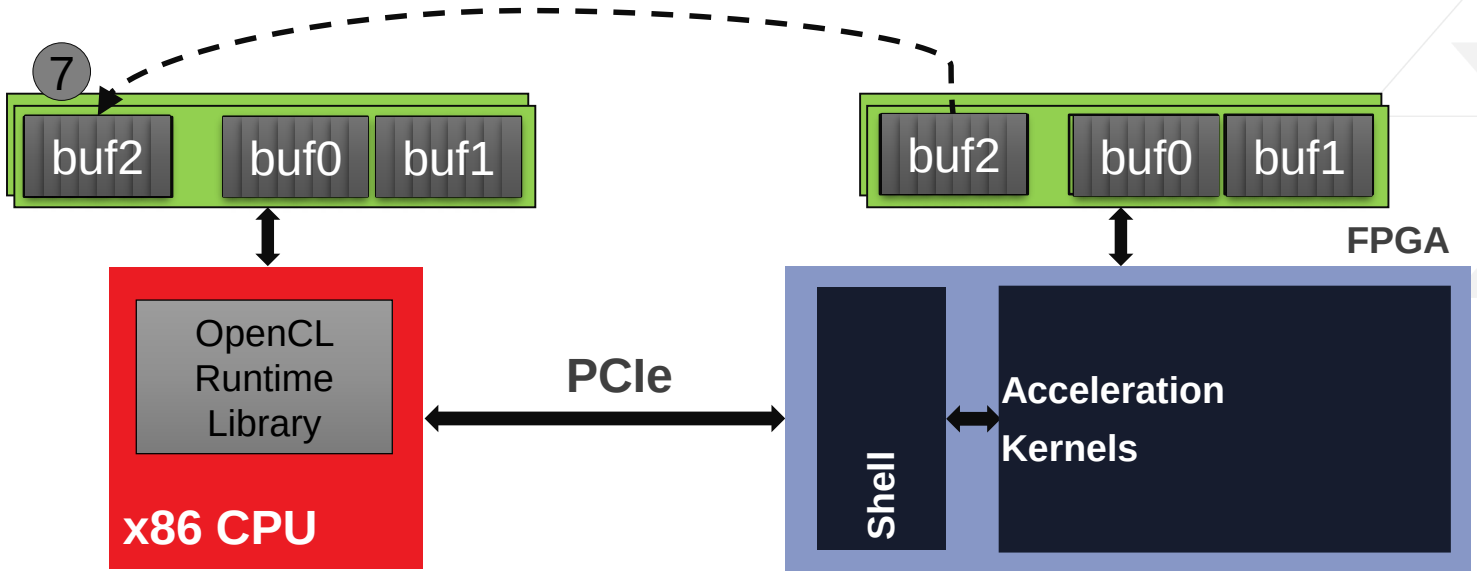
© Copyright 2018 Xilinx

7. Reading Buffers from FPGA Mem

- **The host retrieves the results by scheduling a copy of DDR content back to host memory**

```
clEnqueueReadBuffer(queue, buf2, ..., &readevent );

clWaitForEvents(1, &readevent);
```



© Copyright 2018 Xilinx

Makefile Flow

© Copyright 2018 Xilinx

Environment Setup for SDAccel

- **Before you run the SDAccel tools, either in Makefile or GUI flow, you need to setup the PATH variable and other environment variables**
- **Execute the following commands to source the SDAccel and SDx setup scripts**

```
$ cd ~/aws-fpga
$ source sdaccel_setup.sh
$ source $XILINX_SDX/settings64.sh
```


© Copyright 2018 Xilinx

SW Emulation of the *hello_world* Application

- **Validate and refine the application using software emulation**
  - Validate correctness of the application
  - Refine algorithm, if necessary
    - Fast compilation and run on a CPU
- **Execute the following commands to compile and run the application using the Makefile flow**

```
$ cd $SDACCEL_DIR/examples/xilinx/getting_started/host/helloworld_ocl/
$ make clean
$ make check TARGETS=sw_emu DEVICES=$AWS_PLATFORM all
```

  - The application will be compiled, if it is not up to date, and executed on the x86 CPU (host machine) displaying the result (see next slide)

Note: In the above command if either TARGETS is misspelled or value is not sw_emu, hw_emu, or hw then the full hardware bitstream generation process will be done

© Copyright 2018 Xilinx

Software Emulation Output

```
****** configutil v2018.2.op (64-bit)
  **** SW Build 2258646 on Thu Jun 14 20:02:38 MDT 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: [ConfigUtil 60-895]   Target platform: /home/centos/aws-fpga/SDAccel/aws_platform/xilin
x_aws-vu9p-f1-04261818_dynamic_5_0/xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xpfm
emulation configuration file `emconfig.json` is created in current working directory
XCL_EMULATION_MODE=sw_emu ./helloworld
xclProbe found 1 FPGA slots with xocl driver
Found Platform
Platform Name: Xilinx
Found Device=xilinx_aws-vu9p-f1-04261818_dynamic_5_0
XCLBIN File Name: vector_addition
INFO: Importing xclbin/vector_addition.sw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xclbin
Loading: 'xclbin/vector_addition.sw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xclbin'
Result =
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42
TEST PASSED
make: Nothing to be done for `all'.
[centos@ip-172-31-48-105 helloworld_ocl]$
```

SW Emulation

Target Device

Application Output

© Copyright 2018 Xilinx

XILINX

Summary

© Copyright 2018 Xilinx

Summary

- SDAccel supports C, C++, and OpenCL languages
- SDAccel uses Vivado, Vivado HLS, OpenCL compilers
- OpenCL API are provided for application execution on F1
- SDAccel uses Eclipse environment and rich ecosystem

© Copyright 2018 Xilinx

Lab Intro

© Copyright 2018 Xilinx

Lab Intro

- In this lab you will go through a Makefile flow to build an “hello world” example, which performs vector addition of 256 elements design for software emulation
- You then will modify the host code, using *gedit* editor, to change the values it adds and number of elements on which addition is performed. You will recompile the application and see the result
- You will use pre-compiled binary image and run it on the AWS F1 instance

© Copyright 2018 Xilinx

Adaptable.
Intelligent.

XILINX

© Copyright 2018 Xilinx