SDAccel Flows

SDx 2018.2

XILINX.

Objectives

> **After completing this module, you will be able to:**

- >> Distinguish between host application compilation and kernel compilation
- >> List three modes in which you can compile your application
- >> List application development flows

XILINX.

Outline

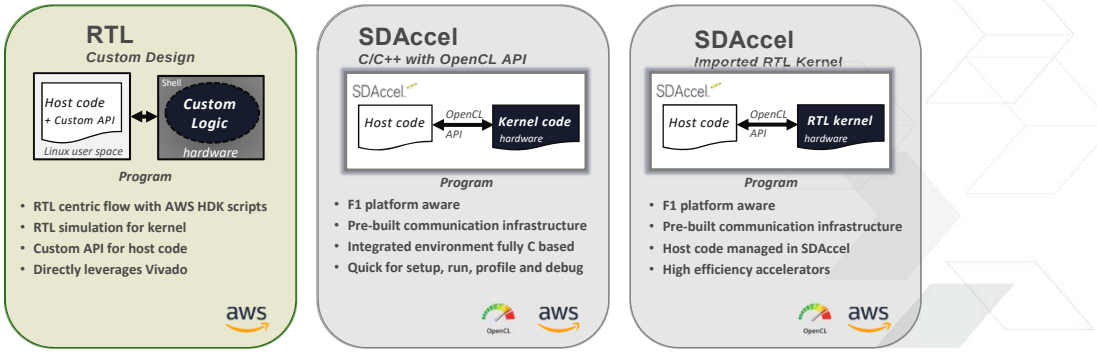
- Development Flows on AWS
- SDAccel Build and Execution Methods
- SDAccel Testing and Execution Modes
- Project Creation and Reports
- Summary
- Lab Intro
- Appendix

AWS Developer Flows

- **AWS provides all necessary tools in the cloud**
  - Run the tools on less expensive general AWS compute instances (e.g. C4/C5)
  - Use costlier F1 instance for hardware verification

AWS Developer Flows (2)

> **F1 programs built with SDAccel (OpenCL) or from RTL/HLS kernels**
>> SDAccel offers a combined host-kernel development flow



AFI Creation Flow Overview

Kernel compilation needed to generate bitstream and AFI

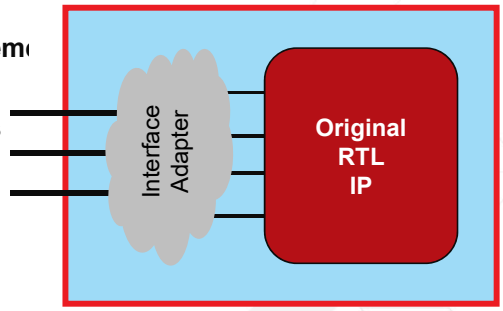
Create SDAccel kernels from C/C++, OpenCL or RTL

Automatically generate the FPGA binary

Create the encrypted Amazon FPGA Image

Creating SDAccel Kernels from RTL IP (1/2)

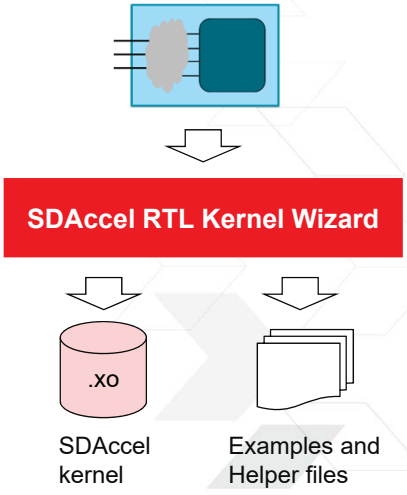
- **Custom RTL IP must be packaged as SDAccel "Kernels"**
- **Kernels must comply with SDAccel interface requirem**
- **Kernels should be designed with performance goals in mind**
  - Interface bandwidth
  - Memory accesses
  - Physical design and timing closure



RTL kernel
with SDAccel compliant interface

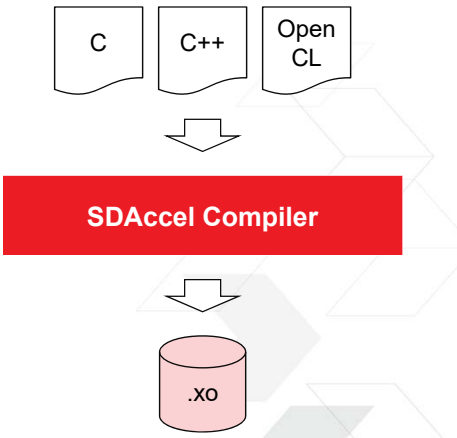
Creating Kernels from RTL IP (2/2)

- **SDAccel RTL Kernel Wizard assists in packaging existing RTL IP as Kernels**
- **Creates Kernel container file (XO file)**
  - Kernel XML meta-data
  - RTL files
  - Vivado IP project
- **XO files are the key 'building blocks' used by SDAccel to assemble the final FPGA design**



Creating Kernels from C/C++, OpenCL (1/2)

- **Parallelizing compiler generates high-performance HW kernels from OpenCL, C, and C++**
- **Advanced optimizations tuned for Xilinx FPGA devices**
  - Memory partitioning
  - DSP block inferencing
  - Loop unrolling, loop pipelining
- **Creates HW kernel with necessary AXI interfaces**
- **Automatically generates SDAccel .xo file**



SDAccel kernel

Creating Kernels from C/C++, OpenCL (2/2)

- **Comprehensive language support**
  - OpenCL 1.3 embedded profile
  - OpenCL 2.0 Pipes
  - OpenCL 2.0 Image Objects
- **N-dimensional kernel ranges**
- **SIMD with vector types**
- **Math library functions**
- **Rich set of examples on Github**

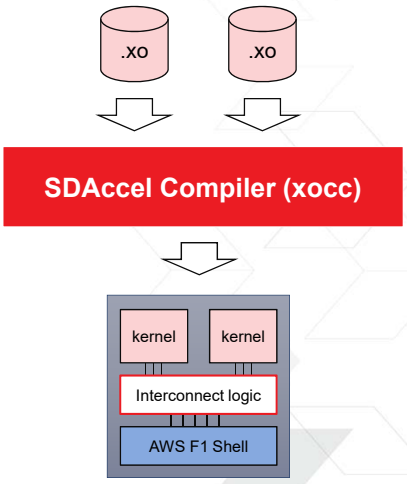
```
__kernel __attribute__ ((reqd_work_group_size(16,16,1)))
void mult(__global int* a,
          __global int* b,
          __global int* output)
{
  int r = get_local_id(0);
  int c = get_local_id(1);
  int rank = get_local_size(0);
  int running = 0;

  for(int index = 0; index < 16; index++){
    int aIndex = r*rank + index;
    int bIndex = index*rank + c;
    running += a[aIndex] * b[bIndex];
  }
  output[r*rank + c] = running;
  return;
}
```

OpenCL matrix multiplication example

Compiling the Platform (1/2)

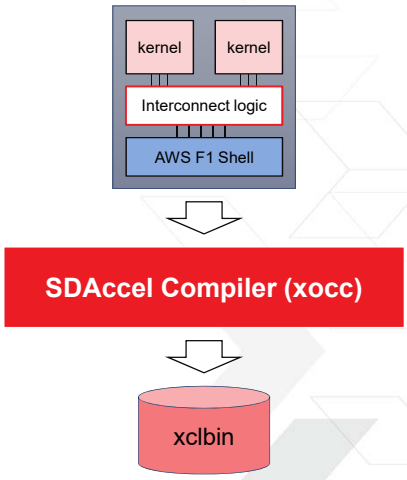
- The SDAccel compiler assembles the FPGA design
- Automatically instantiates the kernels and F1 shell
- Automatically generates DDR interfaces and interconnect logic
- Makes all the necessary connections



Assembled FPGA Design

Compiling the Platform (2/2)

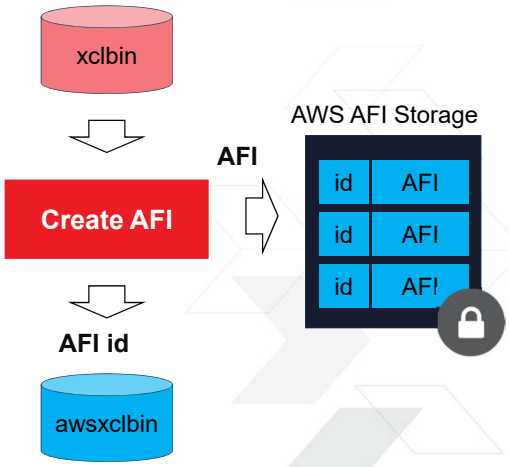
- SDAccel runs synthesis and place & route on assembled FPGA design
- Generates FPGA binary (.xclbin)
- Multiple iterations might be required to meet timing goals
- For best results, Kernels should be designed with recommendations from the *UltraFast Design Methodology Guide for the Vivado Design Suite*



FPGA binary

Creating an Amazon FPGA Image

- Xclbin is converted to an encrypted Amazon FPGA Image (AFI)
- AFIs are created and securely stored by an AWS backend service
- Distributable awsxclbin only contains the AFI id
- AFI id is used at runtime to download the AFI from the Vault into the FPGA
- Application developers have no access to acceleration RTL IP



Host Application Development Flow Overview

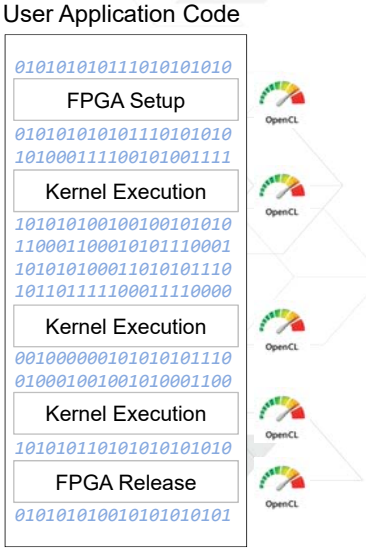
Setup OpenCL in host application

Run host application with FPGA kernels

Use analysis tools to optimize application

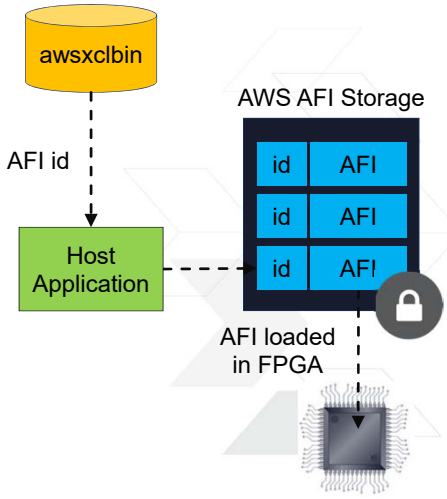
Developing Application Code

- Application written in C/C++, compiled with GCC
- OpenCL API used to communicate with FPGA
- OpenCL runtime and AWS drivers enable the communication with the FPGA hardware
- Host Application can take many forms
  - Standalone executable
  - Plugin, shared lib, etc…
  - Server for client-server system



Executing with the Amazon FPGA Image

- Host application loads the AFI-id from the awsxclbin metadata
- Host application contacts the AWS storage with the AFI-id
- Backend service downloads the AFI into the FPGA
- Host application can dynamically swap and replace AFIs during runtime



SDAccel Build and Execution Methods

XILINX.

Build and Execution Methods and Modes

- **SDAccel supports two methods to build and execute applications**
  - Makefile flow
  - GUI flow
- **Both methods require some environments setup**
- **Both methods go through fundamental steps of compilation**
- **SDAccel supports application compilation, testing, and execution in three modes**
  - SW Emulation
  - HW Emulation
  - System

XILINX.

Environment Setup

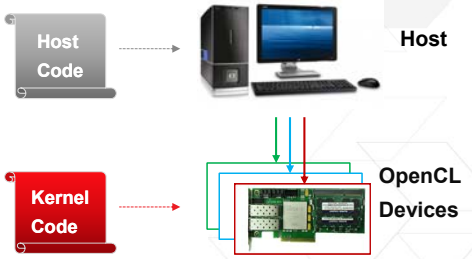
- **AWS**

```
source /opt/xilinx/rte/setup.sh
source /opt/Xilinx/SDx/2018.2.op2258646/settings64.sh
```

- **To get the list of locally supported DSAs**
  - xocc –list xdevices
- **AWS**

  xilinx:aws-vu9p-f1-04261818-dynamic_5_0

Compilation



- **Two parts**
  - **Host**: g++ / gcc compiler
    xcpp (2018.2)
  - **Kernel**: xocc - Xilinx specific
- **g++/gcc compiler: v 4.8.5**
  - Delivered with SDAccel

  `<SDx_install_dir>/Vivado_HLS/lnx64/tools/gcc/bin/g++`
  - To get completer version : `g++ --version`
  - Automatically used by SDAccel setup
- **xocc compiler**
  - Goal: to compile Kernels and store them in XCLBIN binary container

SDAccel Testing and Execution Modes

XILINX

SDAccel Testing and Execution Modes

CPU Emulation	Hardware Emulation	Hardware Execution
Host application runs with a C/C++ or OpenCL model of the Kernels	Host application runs with a simulated RTL model of the Kernels	Host application runs with actual FPGA implementation of the Kernels
Confirm functional correctness of the system	Test the host / kernel integration, get performance estimates	Confirm system runs correctly and with desired performance
Fastest turnaround time	Best debug capabilities	Accurate performance results

XILINX

Project Creation and Reports

XILINX.

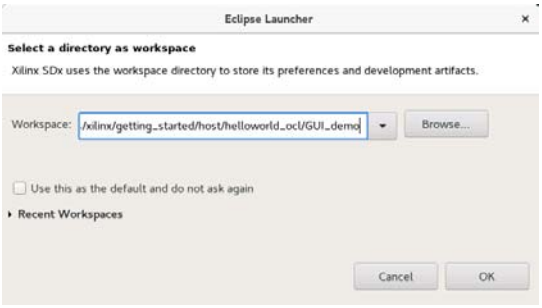
Project Creation Flow

- Identify workspace
- Select platform
- Create a project using predefined template, or empty project and importing source files
- Run software and hardware emulations, and eventually full system build

XILINX.

Invoke SDAccel and Identify Workspace

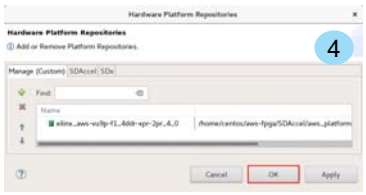
- **Invoke SDAccel in GUI mode by typing *sdx* at the terminal prompt**
  - The workspace selection dialog box will appear
  - Browse to desired directory and click Select
  - Clicking OK will create a directory



XILINX.

Select Custom Platform

- **Click on the Add Custom Platform link**
  1. The *Hardware Platform Repositories* form will be displayed
  2. Click on the **Plus** sign
  3. Browse to `SDAccel/aws-platform/Xilinx-aws-vu9p-f1-04261818_dynamic_5_0` and click **OK**
  4. Click **Apply** and **OK** again



XILINX.

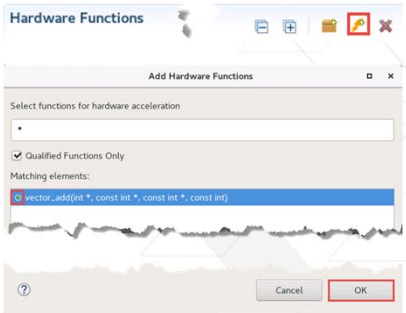
Create a Project and Import Source Files

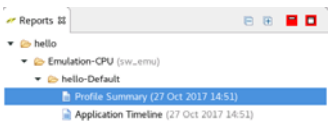
- **Click on the Create SDx Project link**
  - Enter a project name in the *Project name* field and click **Next**
  - Observe the selected hardware platform, click **Next**
  - Observe System configuration and Runtime language, click **Next**
  - Select *Empty Application* and click **Finish**
    - You can add the source file after the project is created
- **Right-click on *src* in the project explorer and select import**
  - Select **General** > **Filesystem** and then click on **Next**
  - Browse to the source file directory and click **OK**
  - Select the necessary files
  - Click **Finish**
    - The files will be imported under the src folder

XILINX.

Software Emulation

- **Select function(s) to be accelerated**
  - Click on the button to see functions defined in the design
    - Notice in this example *vector_add* is the only function selected
    - Click **OK**
    - Notice that the function is added for acceleration
- **Notice that by default Emulation-SW (software) is selected**
- **Click on the *Run* button**
  - This will build (compile) the project and run it
- **A new tab, Reports, will be created**



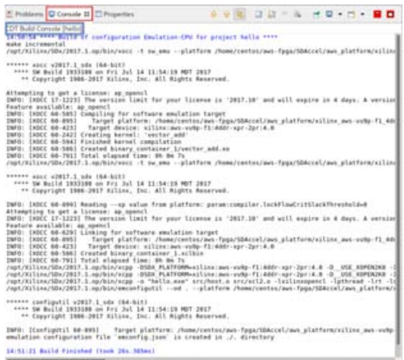


XILINX.

Software Emulation Run Output

> **See the compilation progress in the CDT Build Console view of the Console tab**

> **See the loading and execution of the application in the Console tab**



```
****** 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]$
```


XILINX

Hardware Emulation and System Runs

> **Select Emulation-HW or System as an active configuration**

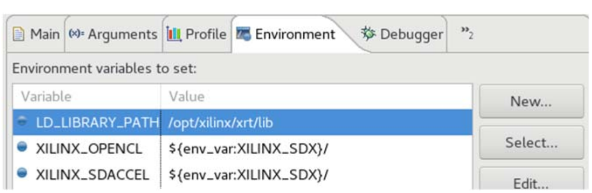
> **On AWS, Xilinx Runtime (xrt) library is used to perform hardware emulation and in hardware system run**

>> It requires super user elevation for execution

> **Elevate to su and source the runtime environment by executing the following commands from the current build directory and then start SDx**

```
$ sudo sh
$ source /opt/xilinx/xrt/setup.sh
```

> **In Run Configuration set $LD_LIBRARY_PATH**



XILINX

Hardware Emulation Output

```
INFO: [ConfigUtil 60-895]   Target platform: /home/centos/aws-fpga/SDAccel/aws_platform/xilinx_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=hw_emu ./helloworld
xclProbe found 1 FPGA slots with xocl driver running
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.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xclbin
Loading: 'xclbin/vector_addition.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xclbin'
INFO: [SDx-EM 01] Hardware emulation runs simulation underneath. Using a large data set will result in
long simulation times. It is recommended that a small dataset is used for faster execution. This flow d
oes not use cycle accurate models and hence the performance data generated is approximate.
Result =
19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
TEST PASSED
INFO: [SDx-EM 22] [Wall clock time: 18:36, Emulation time: 0.00375683 ms] Data transfer between kernel(
s) and global memory(s)
BANK0           RD = 1.000 KB               WR = 0.500 KB
BANK1           RD = 0.000 KB               WR = 0.000 KB
BANK2           RD = 0.000 KB               WR = 0.000 KB
BANK3           RD = 0.000 KB               WR = 0.000 KB
```

HW Emulation

Application Output

Data transfer rate

Emulation Console

```
INFO: [SDx-EM 01] Hardware emulation runs detailed simulation undernea
INFO: [SDx-EM 22] [Wall clock time: 15:51, Emulation time: 0.007938 ms]
BANK0       RD = 2.000 KB       WR = 1.000 KB
BANK1       RD = 0.000 KB       WR = 0.000 KB
BANK2       RD = 0.000 KB       WR = 0.000 KB
BANK3       RD = 0.000 KB       WR = 0.000 KB
```

```
Attempting to get a license: ap_opencl
Feature available: ap_opencl
INFO: [XOCC 60-585] Compiling for hardware emulation target
Running SDx Rule Check Server on port:33547
INFO: [XOCC 60-895]   Target platform: /home/centos/aws-fpga/SDAccel/aws_platform/xilinx_aws-vu9p-f1-
04261818_dynamic_5_0/xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xpfm
INFO: [XOCC 60-423]   Target device: xilinx_aws-vu9p-f1-04261818_dynamic_5_0
INFO: [XOCC 60-242] Creating kernel: 'vector_add'

===>The following messages were generated while  performing high-level synthesis for kernel: vector_a
dd Log file:/home/centos/aws-fpga/Makefile_flow/helloworld_ocl/_x/vector_addition.hw_emu.xilinx_aws-v
u9p-f1-04261818_dynamic_5_0/vector_add/vivado_hls.log :
INFO: [XOCC 204-61] Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequ
ency constraints.
INFO: [XOCC 204-61] Pipelining loop 'Loop 1.1'.
INFO: [XOCC 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 3.
INFO: [XOCC 204-61] Pipelining loop 'Loop 1.2'.
INFO: [XOCC 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 3.
INFO: [XOCC 204-61] Pipelining loop 'vadd_writeC'.
INFO: [XOCC 204-61] Pipelining result : Target II = 1, Final II = 1, Depth = 3.
INFO: [XOCC 60-594] Finished kernel compilation
INFO: [XOCC 60-244] Generating system estimate report...
INFO: [XOCC 60-1092] Generated system estimate report: /home/centos/aws-fpga/Makefile_flow/helloworld
_ocl/_x/reports/vector_addition.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0/system_estimate_vector
_addition.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xtxt
INFO: [XOCC 60-586] Created xclbin/vector_addition.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0.xo
```


XILINX.

Hardware Emulation HLS Report

Synthesis Report for 'vector_add'

General Information

Date:	Fri Oct 27 15:49:32 2017
Version:	2017.1 (Build 1846317 on Fri Jul 14 12:21:14 MDT 2017)
Project:	vector_add
Solution:	solution_OCL_REGION_0
Product family:	virtexuplus
Target device:	xcvu9p-flgb2104-2-i

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	2.92	1.08

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	2.92	1.08

Latency (clock cycles)

Summary

Latency		Interval	
min	max	min	max	Type
?	?	?	?	none

Detail

Instance

N/A

Loop

	Latency			Initiation Interval				
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	?	?	?	-	-	?	no
+ readA	?	?	3	1	1	?	yes
+ readB	?	?	3	1	1	?	yes
+ vadd_writeC	?	?	3	1	1	?	yes

Utilization Estimates

Summary

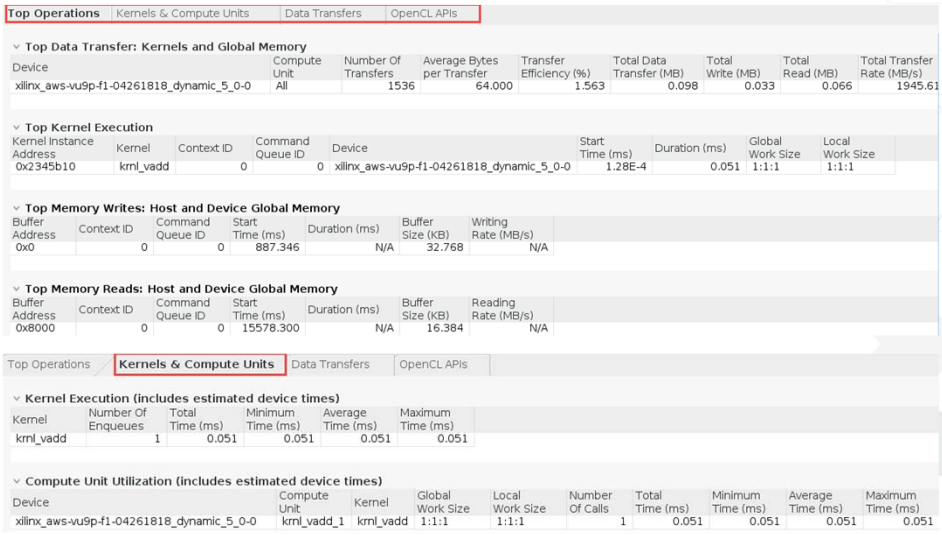
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	853	-
FIFO	-	-	-	-	-
Instance	2	-	796	1068	-
Memory	2	-	0	0	-
Multiplexer	-	-	-	2010	-
Register	-	-	1522	-	-
Total	4	0	2318	3931	0
Available	4320	6840	2364480	1182240	960
Utilization (%)	~0	0	~0	~0	0

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_control_AWVALID	in	1	s_axi	control	pointer
s_axi_control_AWREADY	out	1	s_axi	control	pointer
s_axi_control_AWADDR	in	6	s_axi	control	pointer
s_axi_control_WVALID	in	1	s_axi	control	pointer
s_axi_control_WREADY	out	1	s_axi	control	pointer
s_axi_control_WDATA	in	32	s_axi	control	pointer
s_axi_control_WSTRB	in	4	s_axi	control	pointer
s_axi_control_ARVALID	in	1	s_axi	control	pointer
s_axi_control_ARREADY	out	1	s_axi	control	pointer
s_axi_control_ARADDR	in	6	s_axi	control	pointer
s_axi_control_RVALID	out	1	s_axi	control	pointer
s_axi_control_RREADY	in	1	s_axi	control	pointer
s_axi_control_RDATA	out	32	s_axi	control	pointer
s_axi_control_RRESP	out	2	s_axi	control	pointer
s_axi_control_BVALID	out	1	s_axi	control	pointer
s_axi_control_BREADY	in	1	s_axi	control	pointer
s_axi_control_BRESP	out	2	s_axi	control	pointer
ap_clk	in	1	ap_ctrl_hs	vector_add	return value
ap_rst_n	in	1	ap_ctrl_hs	vector_add	return value
interrupt	out	1	ap_ctrl_hs	vector_add	return value
stall_start_ext	out	1	ap_ctrl_hs	vector_add	return value
stall_done_ext	out	1	ap_ctrl_hs	vector_add	return value
stall_start_str	out	1	ap_ctrl_hs	vector_add	return value
stall_done_str	out	1	ap_ctrl_hs	vector_add	return value
stall_start_int	out	1	ap_ctrl_hs	vector_add	return value
stall_done_int	out	1	ap_ctrl_hs	vector_add	return value
m_axi_gmem_AWVALID	out	1	m_axi	gmem	pointer
m_axi_gmem_AWREADY	in	1	m_axi	gmem	pointer
m_axi_gmem_AWADDR	out	64	m_axi	gmem	pointer

XILINX.

Hardware Emulation Run Reports – Profile Summary



Top Operations | Kernels & Compute Units | Data Transfers | OpenCL APIs

Top Data Transfer: Kernels and Global Memory

Device	Compute Unit	Number Of Transfers	Average Bytes per Transfer	Transfer Efficiency (%)	Total Data Transfer (MB)	Total Write (MB)	Total Read (MB)	Total Transfer Rate (MB/s)
xilinx_aws-vu9p-f1-04261818_dynamic_5_0-0	All	1536	64.000	1.563	0.098	0.033	0.066	1945.61

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x2345b10	krnl_vadd	0	0	xilinx_aws-vu9p-f1-04261818_dynamic_5_0-0	1.28E-4	0.051	1:1:1	1:1:1

Top Memory Writes: Host and Device Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x0	0	0	887.346	N/A	32.768	N/A

Top Memory Reads: Host and Device Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Reading Rate (MB/s)
0x8000	0	0	15578.300	N/A	16.384	N/A

Top Operations | Kernels & Compute Units | Data Transfers | OpenCL APIs

Kernel Execution (includes estimated device times)

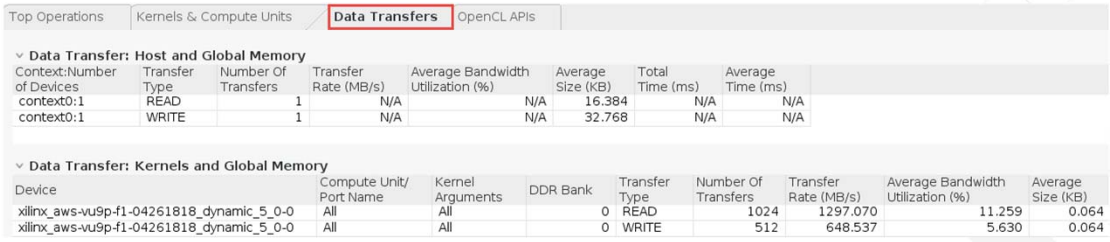
Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
krnl_vadd	1	0.051	0.051	0.051	0.051

Compute Unit Utilization (includes estimated device times)

Device	Compute Unit	Kernel	Global Work Size	Local Work Size	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
xilinx_aws-vu9p-f1-04261818_dynamic_5_0-0	krnl_vadd_1	krnl_vadd	1:1:1	1:1:1	1	0.051	0.051	0.051	0.051

XILINX.

Hardware Emulation Run Reports – Profile Summary (2)



Top Operations | Kernels & Compute Units | Data Transfers | OpenCL APIs

Data Transfer: Host and Global Memory

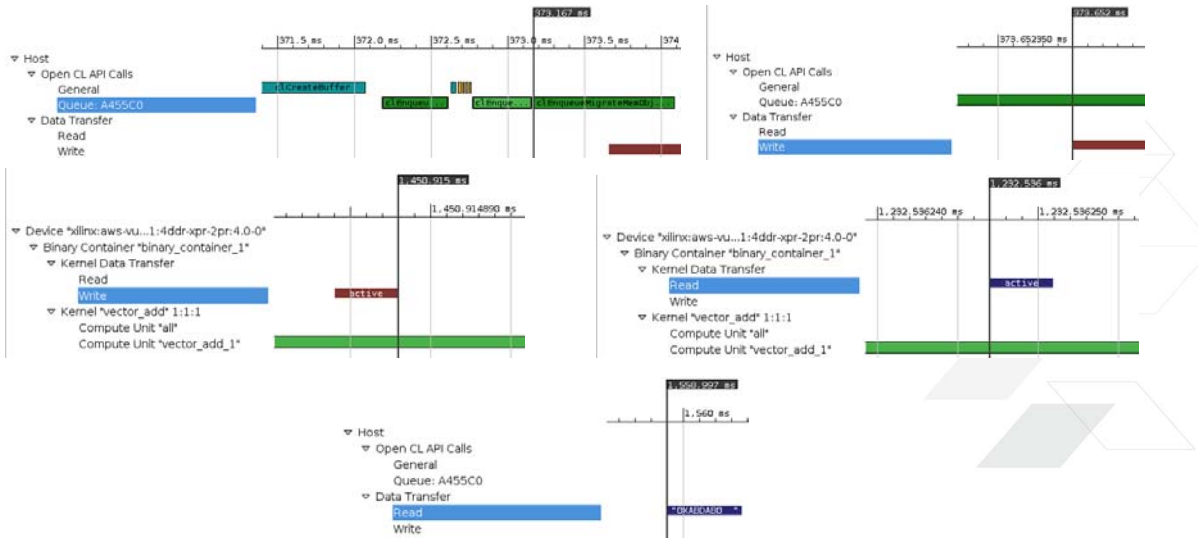
Context:Number of Devices	Transfer Type	Number Of Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	N/A	N/A	16.384	N/A	N/A
context0:1	WRITE	1	N/A	N/A	32.768	N/A	N/A

Data Transfer: Kernels and Global Memory

Device	Compute Unit/ Port Name	Kernel Arguments	DDR Bank	Transfer Type	Number Of Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Size (KB)
xilinx_aws-vu9p-f1-04261818_dynamic_5_0-0	All	All	0	READ	1024	1297.070	11.259	0.064
xilinx_aws-vu9p-f1-04261818_dynamic_5_0-0	All	All	0	WRITE	512	648.537	5.630	0.064

XILINX.

Hardware Emulation Run Reports – Application Timeline



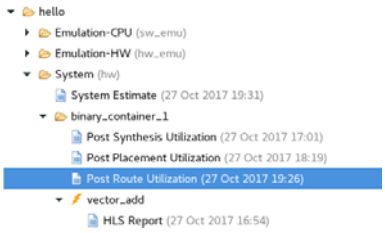
XILINX.

Hardware (System) Build

- Select the System mode by clicking on the drop-down button
- Right-click on the project folder and select Clean Project
- Click on the *Run* button
- When run is completed, click through various reports (next slide)
- Select File > Exit to close the GUI
- Next create an AFI image
- Run the application

XILINX.

Hardware (System) Build Reports



hello | Utilization

Report name: Utilization
Project name: hello
Created: 27 Oct 2017 19:26

Name	LUT	LUTMem	REG	BRAM	URAM	DSP
Platform	290598	39795	378071	545	0	12
User Budget	891170	552045	1985465	1615	960	6828
Used Resources	3529	1538	3484	2	0	0
Unused Resources	887641	550507	1981981	1613	960	6828
vector_add (1)	3529	1538	3484	2	0	0
vector_add_1	3529	1538	3484	2	0	0

hello | Utilization | system_estimate.xtxt

Report name: System Estimate
Project name: hello
Created: 27 Oct 2017 19:31

```
Version:    xocc v2017.1_sdx (64-bit)
Build:      SW Build 1933108 on Fri Jul 14 11:54:19 MDT 2017
Copyright:  Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
Created:    Fri Oct 27 19:31:31 2017

Design Name:              binary_container_1
Target Device:            xilinx:aws-vu9p-f1:4ddr-xpr-2pr:4.0
Target Clock:             250MHz
Total number of kernels: 1

Kernel Summary
Kernel Name  Type  Target              OpenCL Library      Compute Units
vector_add   clc   fpga0:OCL_REGION_0  binary_container_1  1

OpenCL Binary:     binary_container_1
Kernels mapped to: clc_region

Timing Information (MHz)
Compute Unit  Kernel Name  Module Name  Target Frequency  Estimated Frequency
vector_add_1  vector_add   vector_add   250               342.465759

Latency Information (clock cycles)
Compute Unit  Kernel Name  Module Name  Start Interval  Best Case  Avg Case  Worst Case
vector_add_1  vector_add   vector_add   undef           undef      undef     undef

Area Information
Compute Unit  Kernel Name  Module Name  FF    LUT   DSP  BRAM
vector_add_1  vector_add   vector_add   2317  3895  0    4
```


XILINX.

Summary

XILINX.

Summary

- Host application compilation is done through Gcc compiler generating .exe file that runs on a host (x86) CPU
- Kernels are compiled using xocc compiler, and Vivado place and route is used to generate xclbin file
- Emulation-SW, Emulation-HW, and System are the three modes in which you can compile your application
- SDAccel supports both Makefile and GUI development methods

XILINX.

Lab Intro

XILINX.

Lab Intro

- In this lab you will use one of the application templates available in SDAccel to create a project using the GUI flow. You will use one function as a target kernel and build the design. You will go through all three build modes to test the functionality
- You will review HLS report for the generated kernel and the system estimate report for the entire system
- You will perform profile and timing analysis for both Emulation-HW and System builds

XILINX.

Appendix

XILINX.

Host Compilation

> **Host compilation is required to generate executable file**

```
CXX  = xcpp
# HOST Sources and Host Executable files
HOST_EXE      = host.exe
HOST_SRC_CPP  = ../src/host_1.cpp        ../src/common_help_functions.cpp
HOST_SRC_H    = ../src/host_kernel_def.h ../src/common_help_functions.h
# Runtime Libraries
OPENCL_INC = $(XILINX_SDX)/runtime/include/1_2
OPENCL_LIB = $(XILINX_SDX)/runtime/lib/x86_64
# Compilation
.PHONY: all
all: compile

compile: $(HOST_EXE)

$(HOST_EXE): $(HOST_SRC_CPP) $(HOST_SRC_H)
        $(CXX) -lOpenCl -I$(OPENCL_INC) -L$(OPENCL_LIB) -o $@ $(HOST_SRC_CPP)
```

Compilation against ICD* file

```
$ make -f ../make_files/make_file_compile.mk all
```

➡ **host.exe**

ICD = Installable Client Driver*

ICD File

> **ICD – Installable Client Driver**
  - >> Enables Host code to work with platforms from Multiple Vendors
  - >> xilinx.icd file – should be located in **/etc/OpenCL/vendors**
    - Should contain this line
      `libxilinxopencl.so`
  - >> Automatically created by running **install.sh** generated by **xbinst**
  - >> You may create it manually using **sudo** mode

Note: if xilinx.icd file does not exist and you do not have permission to create it (ex: Xilinx machines) replace **-lOpenCL** by **-lxilinxopencl** in Makefile

> ***Note: SDx GUI does not use ICD yet.***

Kernel Compilation

- **Two Modes:**
  - **Build** mode
    - Single xocc command generates XCLBIN file
    - Convenient to use when all kernels are located in a single file
  - **Compilation / Link** mode
    - Two Commands used to generate XCLBIN
      - `xocc --compile`
      - `xocc --link`
    - Convenient to use when working
      - With multiple Kernel files
      - Need to compile only a subset of kernels
    - SDx GUI uses this mode
- **Kernel compilation needed to generate bitstream and AFI**

Kernel Compilation – SW Emulation

```
XOCC = xocc
DEVICE = xilinx:xil-accel-rd-ku115:4ddr-xpr:4.0
TARGET = sw_emu

# Kernel Source and XCLBIN files
XCLBIN        = kernels.$(TARGET).xclbin
KERNEL_SRC_CL = ../src/K_ALL.cl


# Compilation
.PHONY: all
all: compile

compile: $(XCLBIN)

$(XCLBIN): $(KERNEL_SRC_CL)
        $(XOCC) -t $(TARGET) --platform $(DEVICE) -o $@ $(KERNEL_SRC_CL)
```

Tip: naming convention helps to store XCLBINs for all targets

-t : defines compile target
- sw_emu: SW emulation
- hw_emu: Hardware emulation
- hw: Hardware

Build mode

```
$ make -f ../make_files/make_file_compile.mk all
```

➡ **kernels.sw_emu.xclbin**


Adaptable.
Intelligent.
XILINX.
© Copyright 2018 Xilinx