



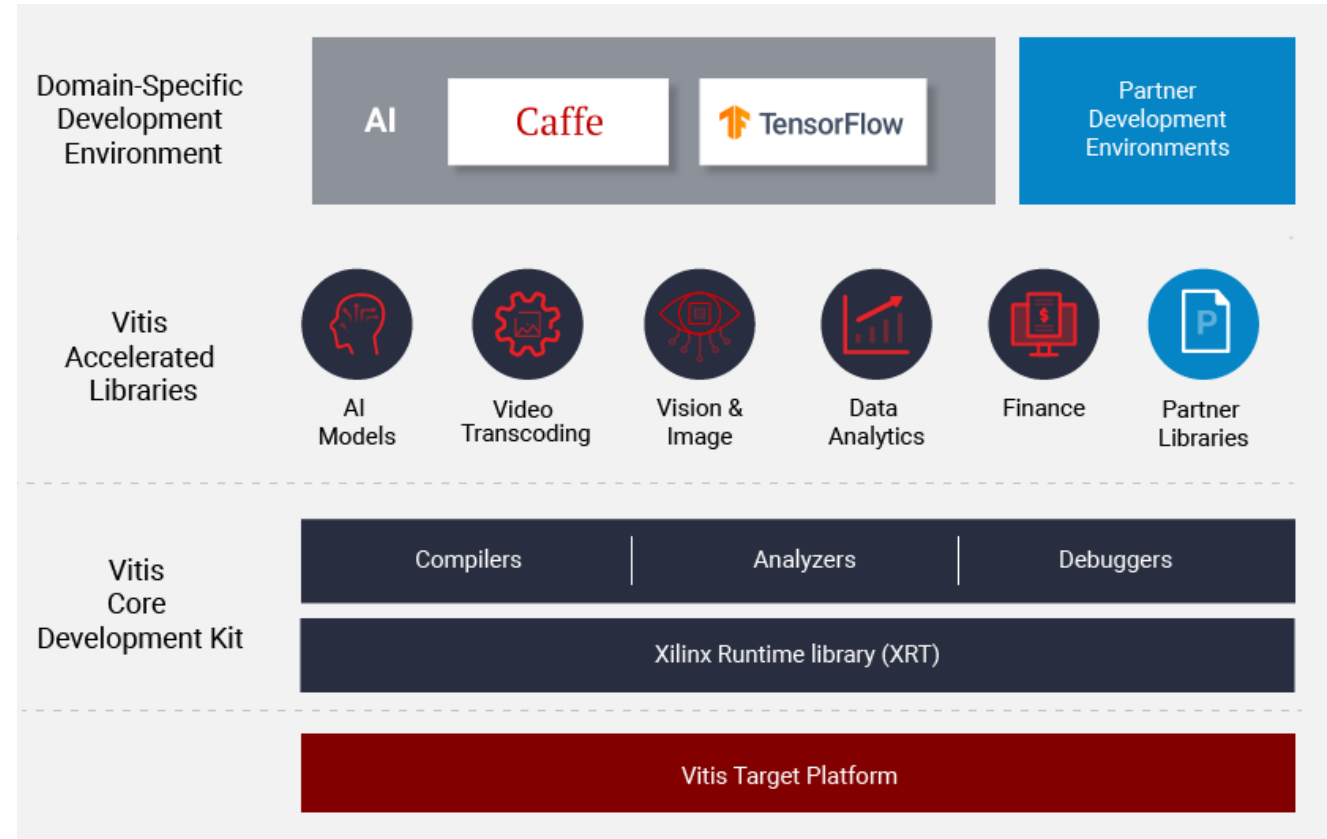
# Building Accelerated Applications with Vitis

Sponsored by



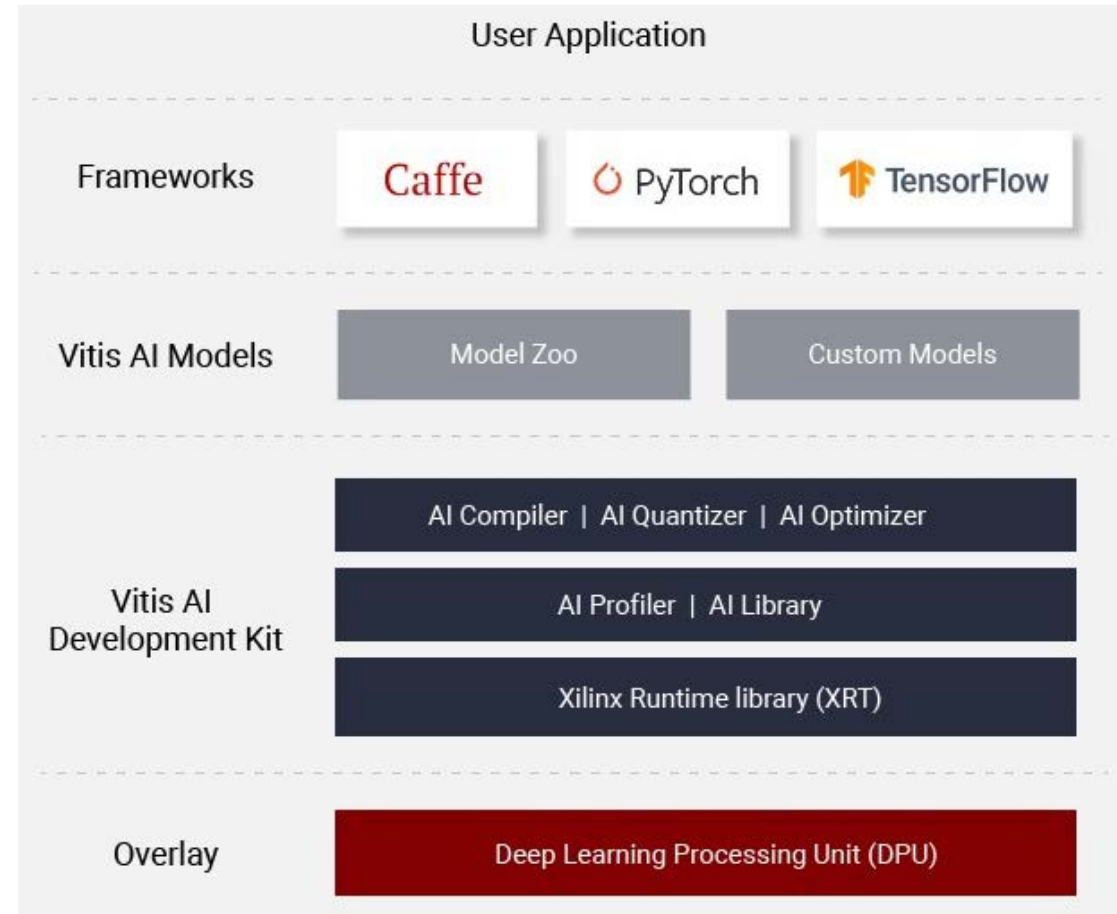
# What is Vitis?

- Vitis is unified software development environment from Xilinx
- Edge and cloud development methodologies
- Support embedded and accelerated flows



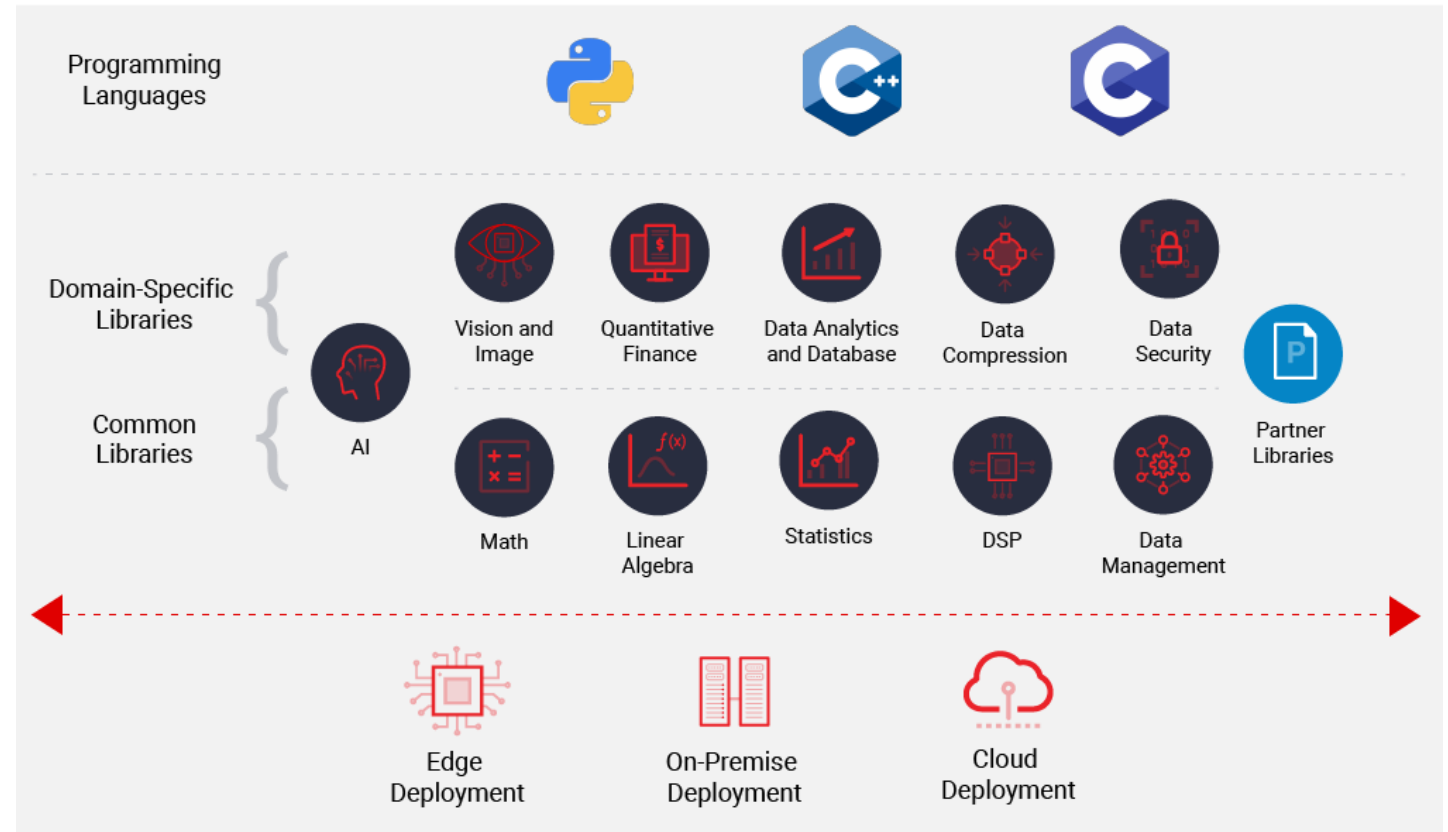
# Vitis AI Development Environment

- Vitis AI enables acceleration of AI inference at edge and cloud
- Supports leading frameworks such as TensorFlow, Caffe and Pytorch
- Works with fixed-point representation and Xilinx Deep Learning Processor Unit (DPU)



# Vitis Accelerated Libraries

- Open Source acceleration ready libraries
- Common libraries offer a set of common functionality
- Domain-specific libraries offer out of the box functions for specific domains (e.g., vision)

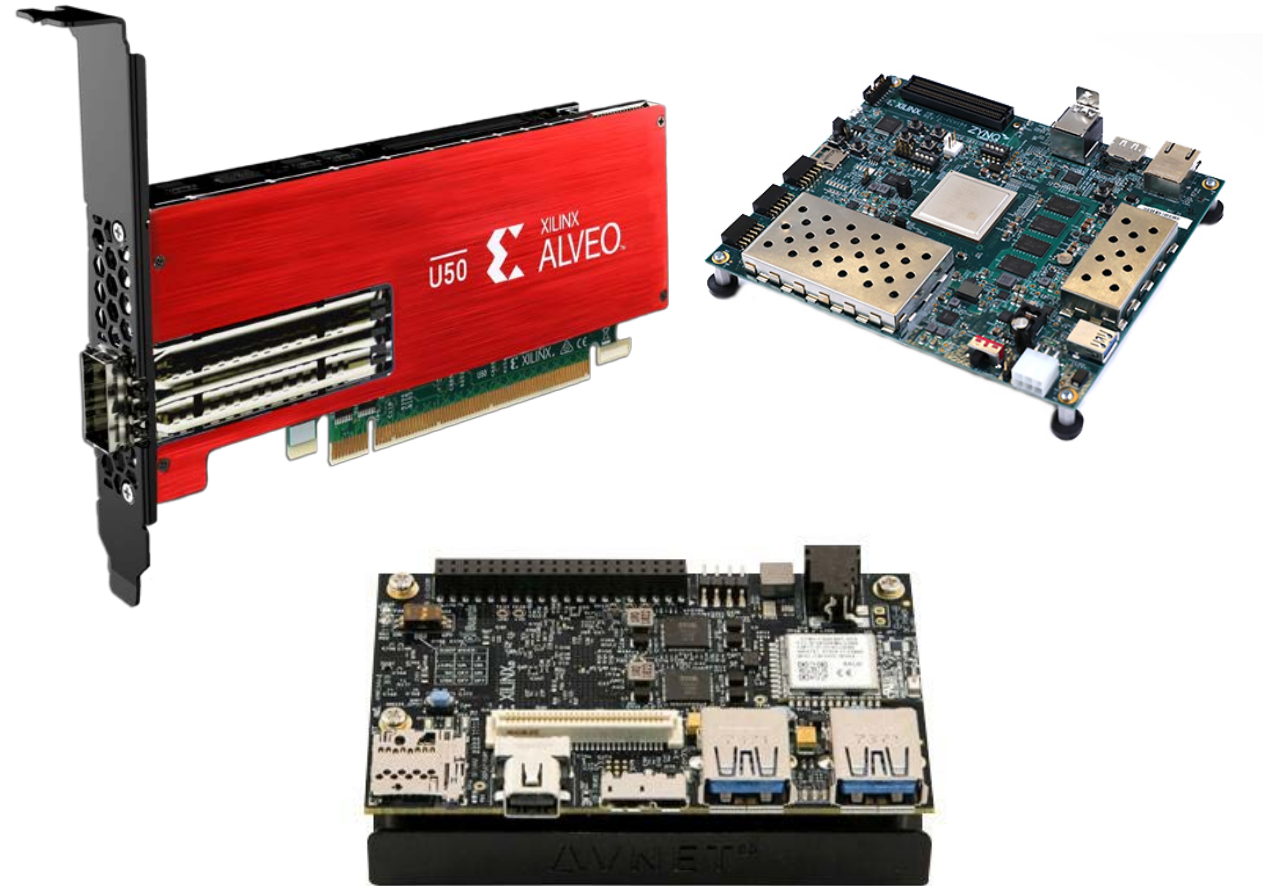


# Vitis Core Development Kit

- GUI and command line tools for compilation, debug and analysis of C, C++ and OpenCL designs
- Use preferred GUI or integrated GUI
- Supports embedded and accelerated flows

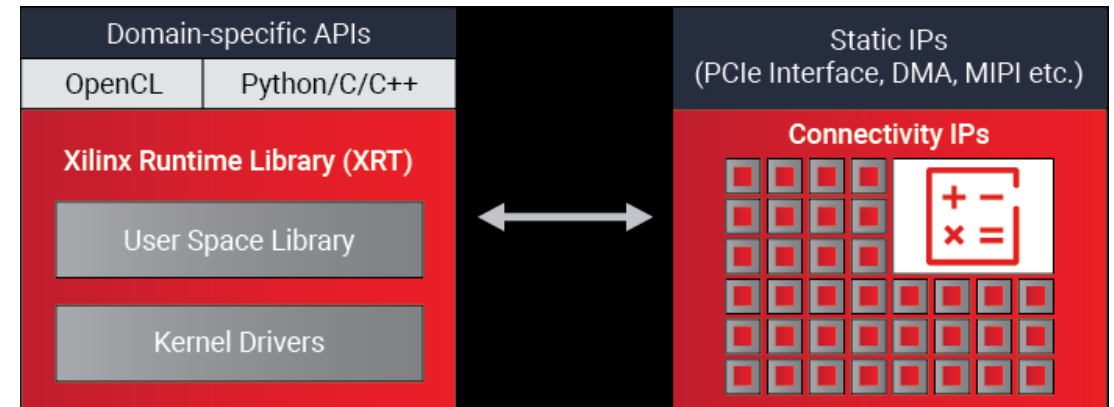
# Vitis Target Platforms

- Embedded
  - SoC: MPSoC, RFSoc, Zynq
  - FPGA: MicroBlaze
- Cloud
  - Alveo
  - AWS F1 Instance
- Embedded SoC and cloud applications can use acceleration flow
- All required files and boot elements are generated



# Xilinx Runtime library

- Xilinx Runtime library (XRT) enables communication between the host and accelerator
- Cloud-based – Host x86
- Embedded – Arm Cortex-A9 or A53
- Provides all libraries, APIs, drivers and utilities.





# Xilinx Runtime library

Key functions of the Runtime include:

- Downloading the FPGA binary
- Memory management between host and accelerator
- Execution management
- Board management

```
adiuvo@Adiuvo: ~  
File Edit View Search Terminal Help  
INFO: == Starting PCIE link check:  
LINK ACTIVE, ATTENTION  
Ensure Card is plugged in to Gen3x16, instead of Gen3x4  
Lower performance may be experienced  
WARN: == PCIE link check PASSED with warning  
INFO: == Starting SC firmware version check:  
SC FIRMWARE MISMATCH, ATTENTION  
SC firmware running on board: 1.8. Expected SC firmware from installed Shell: 4.2.0  
Please use "xbmgmt flash --scan" to check installed Shell.  
WARN: == SC firmware version check PASSED with warning  
INFO: == Starting verify kernel test:  
INFO: == verify kernel test PASSED  
INFO: == Starting DMA test:  
Host -> PCIE -> FPGA write bandwidth = 3335.9 MB/s  
Host <- PCIE <- FPGA read bandwidth = 3238.05 MB/s  
INFO: == DMA test PASSED  
INFO: == Starting device memory bandwidth test:  
.....  
Maximum throughput: 52428 MB/s  
INFO: == device memory bandwidth test PASSED  
INFO: == Starting PCIE peer-to-peer test:  
P2P BAR is not enabled. Skipping validation  
INFO: == PCIE peer-to-peer test SKIPPED  
INFO: == Starting memory-to-memory DMA test:  
bank0 -> bank1 M2M bandwidth: 12100 MB/s  
bank0 -> bank2 M2M bandwidth: 12128.7 MB/s  
bank0 -> bank3 M2M bandwidth: 12114.9 MB/s  
bank1 -> bank2 M2M bandwidth: 12116 MB/s  
bank1 -> bank3 M2M bandwidth: 12118.9 MB/s  
bank2 -> bank3 M2M bandwidth: 12116 MB/s  
INFO: == memory-to-memory DMA test PASSED  
INFO: Card[0] validated with warnings.  
  
INFO: All cards validated successfully but with warnings.  
adiuvo@Adiuvo:~$
```



# Element of Vitis

All projects require a platform

- Hardware element – makes available AXI connections, clocks and Interrupts in the PL to Vitis Compiler
- Software element – provides boot, XRT and QEMU support
- Linux element – FS, Image and SysRoot

**Platform**

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Select a platform from repository | Create a new platform from hardware (XSA)

Find:

+ Add Manage

Name	Board	Flow	Vendor	Path
xilinx_zcu104_base_202010_1	zcu104	Embedded Accel	xilinx	/opt/xilinx/platforms/xilinx_zcu104_base_202010_1/xilinx_zcu104_base_202010_1.xpfm
xilinx_u200_xdma_201830_2	u200	DataCenter Accel	xilinx	/opt/xilinx/platforms/xilinx_u200_xdma_201830_2/xilinx_u200_xdma_201830_2.xpfm

Platform Info

General Info

Name:

Part:

Family:

Description:

A basic static platform targeting the ZCU104 evaluation board, which includes 2GB DDR4, GEM, USB, SDIO interface and UART of the Processing System. It reserves most of the PL resources for user to add acceleration

Acceleration Resources

Clock Frequencies

Clock	Frequency (MHz)
CPU	1200.000000
PL 0	150.000000
PL 1	300.000000
PL 2	75.000000
PL 3	100.000000
PL 4	200.000000
PL 5	400.000000
PL 6	600.000000

Domain Details

Domains

Domain name	Details
xrt	CPU: cortex-a53 OS: linux

< Back | Next > | Cancel | Finish

# Vitis Output

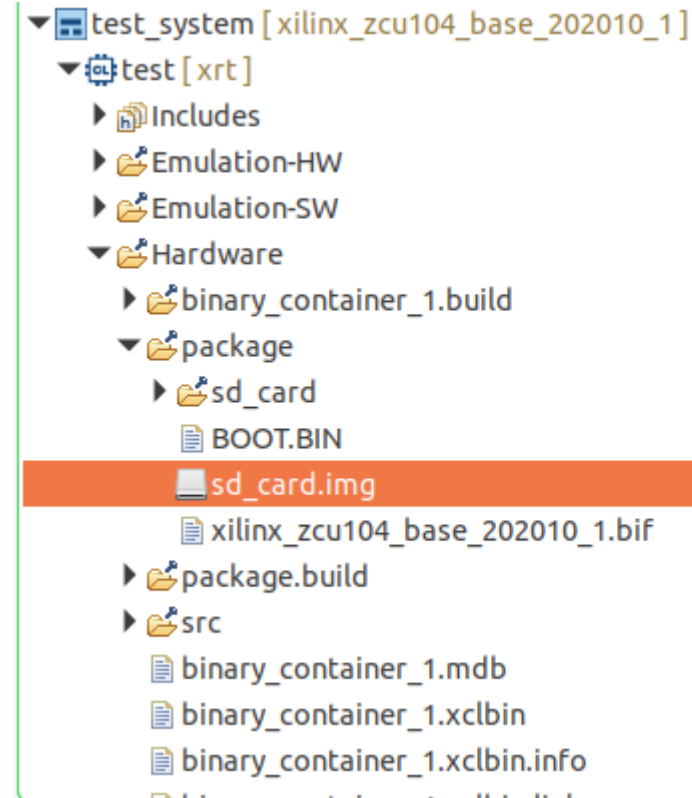
Compiled binary (host) and XCLbin (accelerator)

## Embedded System Output

- SD Card Image
  - Image
  - File System
  - Binary and XCLBin

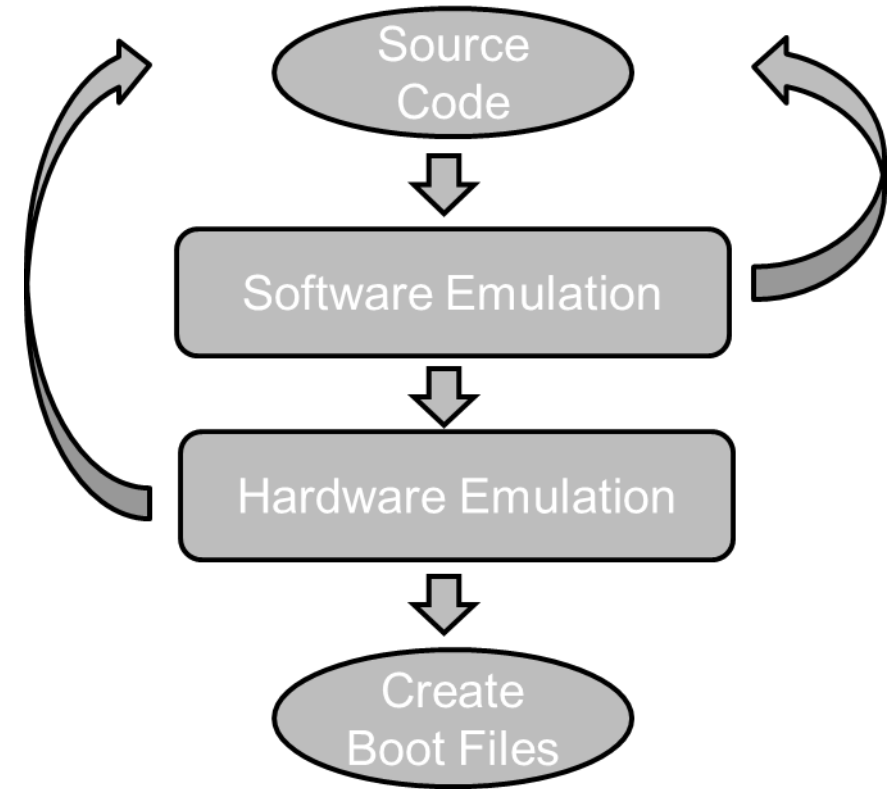
## Cloud output

- Binary and XCLBin



# Vitis Development Flow

- Software Emulation – Syntax errors and algorithm verification
- Hardware Emulation – Optimize performance, interfacing and resources



# OpenCL Framework

- An open industry standard
  - For parallel computing
  - Of heterogeneous systems
- Enables cross-platform functional portability
  - No code changes
  - Portable across CPU, GPU, FPGA, DSP, etc.
  - Important: No performance portability
- Wide market adoption
  - Support implemented by technology leaders
  - Many companies developing applications

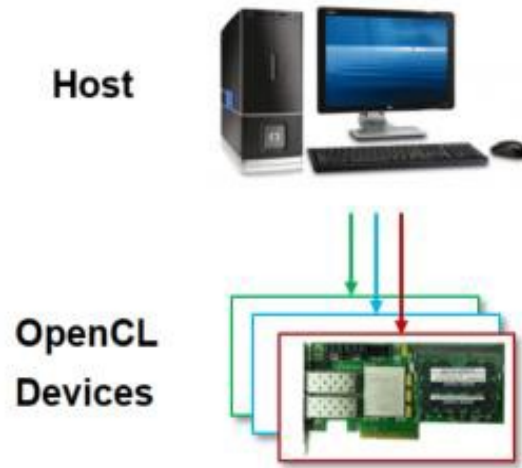


OpenCL

*Khronos Group*

[www.khronos.org](http://www.khronos.org)

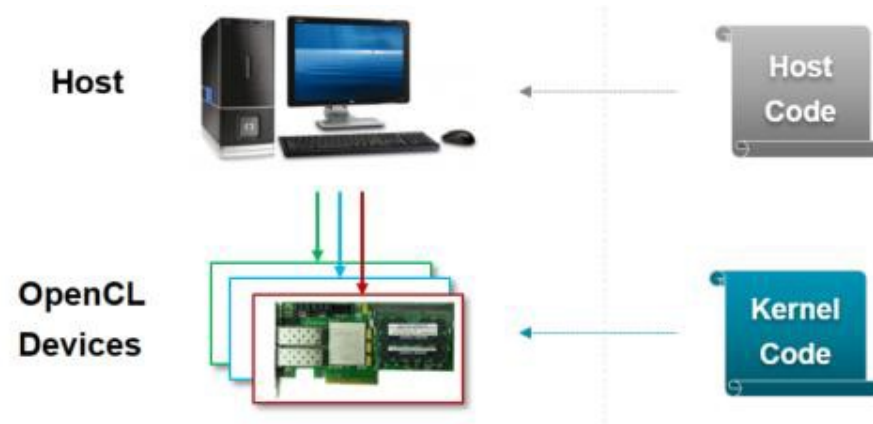
# OpenCL Framework



## Platform model

- Defines representation of ANY platform
- Contains
  - Single host
  - One or more OpenCL devices (compute device)

# OpenCL Framework



## Platform model

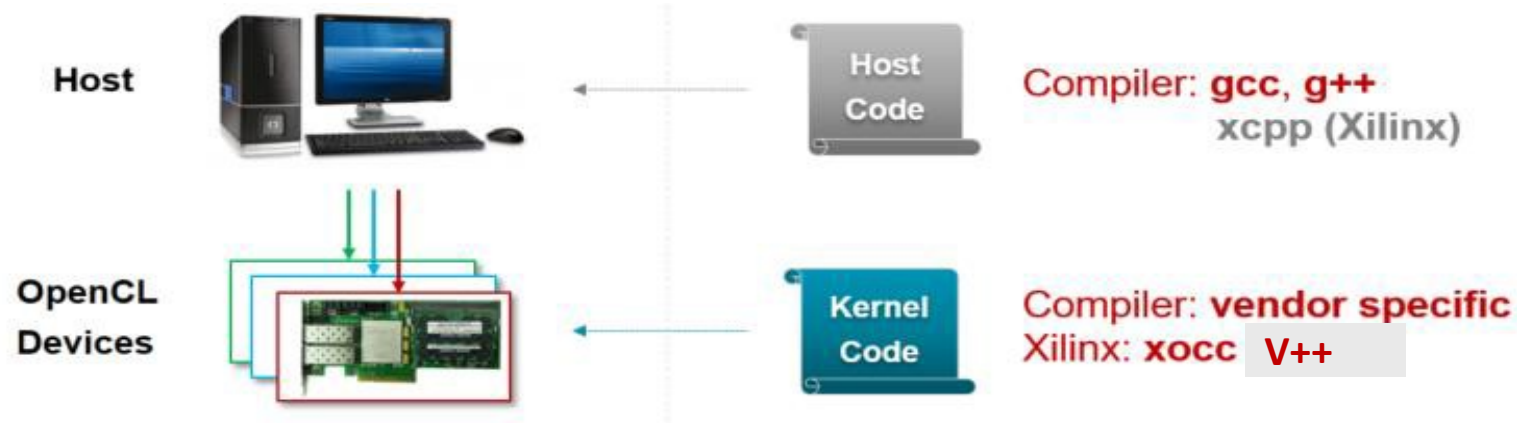
- Defines representation of ANY platform
- Contains
  - Single host
  - One or more OpenCL devices (compute device)

## Execution model

OpenCL application (two parts)

- Host program
  - Manages the entire application: OpenCL APIs
- Kernels (OpenCL C language)
  - Functions to accelerate, run on OpenCL devices

# OpenCL Framework



## Platform model

- Defines representation of ANY platform
- Contains
  - Single host
  - One or more OpenCL devices (compute device)

## Execution model

OpenCL application (two parts)

- Host program
  - Manages the entire application: OpenCL APIs
- Kernels (OpenCL C language)
  - Functions to accelerate: run on OpenCL devices

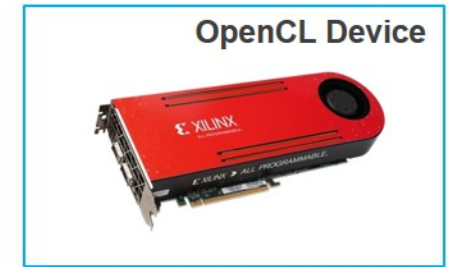


# Execution Model – Command Queues

Interaction between host and device occurs via command queues

- Created by host
- Attached to a single device

**Note:** Multiple command queues can be active within context



Three command types:

- Kernel execution commands
- Memory commands
  - Transfer data between host and different memory objects
- Synchronization commands
  - Put constraints on in the order in which commands are executed

# Memory Model

Three types of memory objects:

- Buffer objects
  - Contiguous block of memory
  - Available to kernels for read/write
  - Programmer can write data to buffers
  - Access to data via pointers
- Image objects (not a part of embedded profile)
  - Hold images only
  - Storage/format can be optimized for specific OpenCL device
  - OpenCL framework provides functions to manipulate images
- Pipes
  - Data organized as FIFO
  - Accessed (read/write) via built in
  - Pipe not accessible from the host



# Five Sub-regions of Memory Objects

## Host memory

- Visible to host only
- OpenCL framework only defines how host memory interacts with OpenCL objects

## Global memory

- Visible to host and device
- All work items in all workgroups can read/write there
- Global on-chip memory – visible to device only

## Constant memory

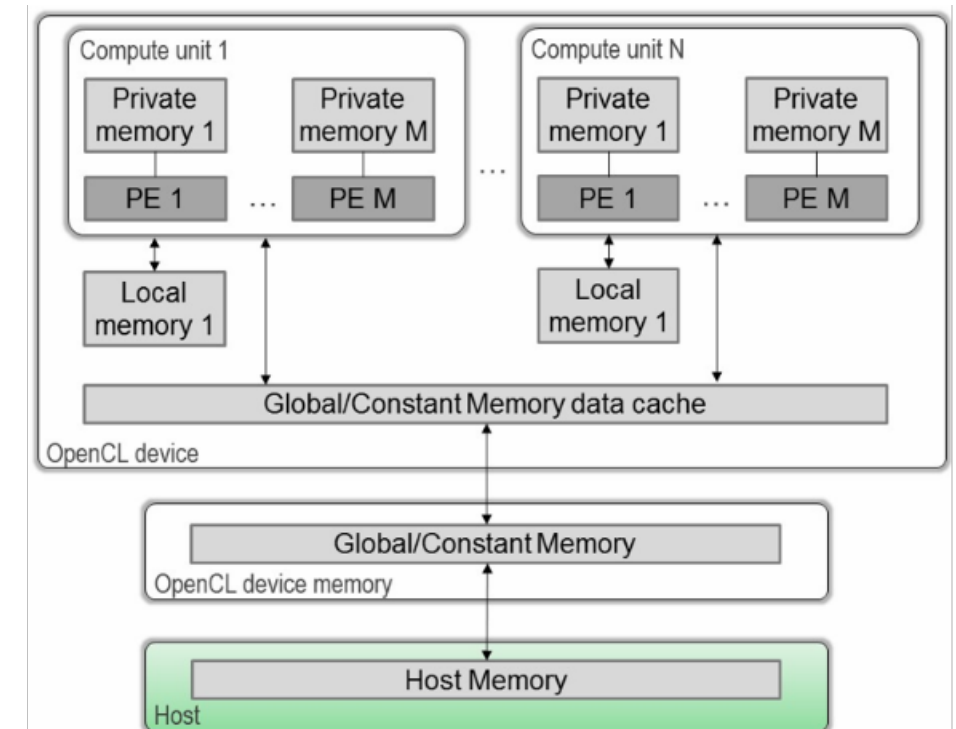
- Region of global memory
- Work items – read access only

## Local memory

- Local to workgroup (shared by all work-items in a group)

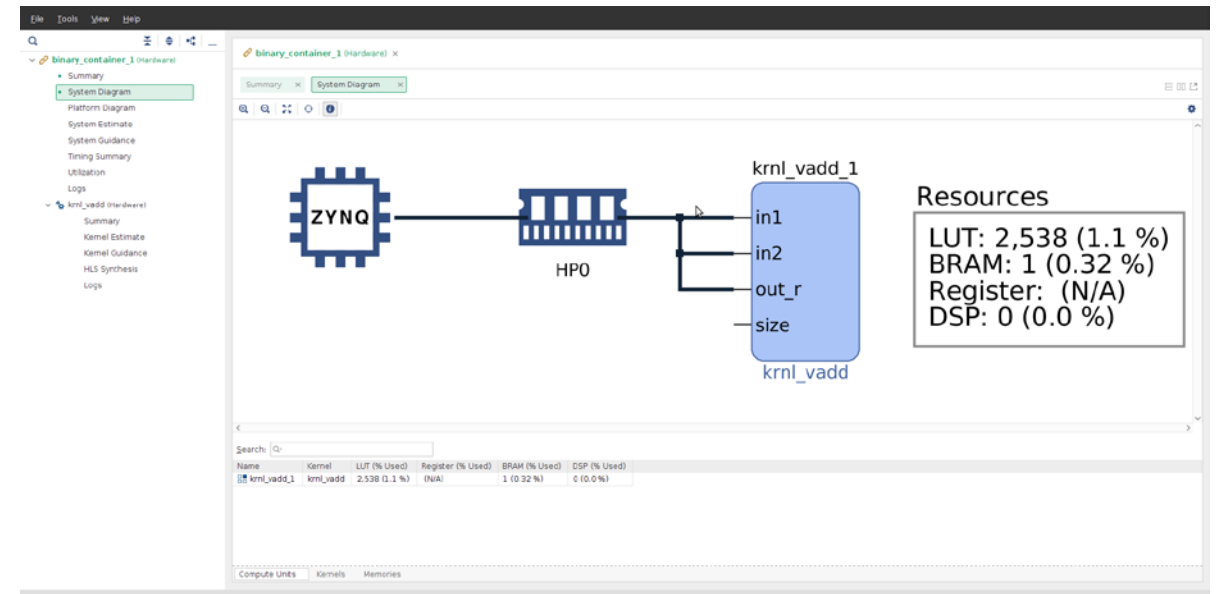
## Private memory

- Accessible by a work-item



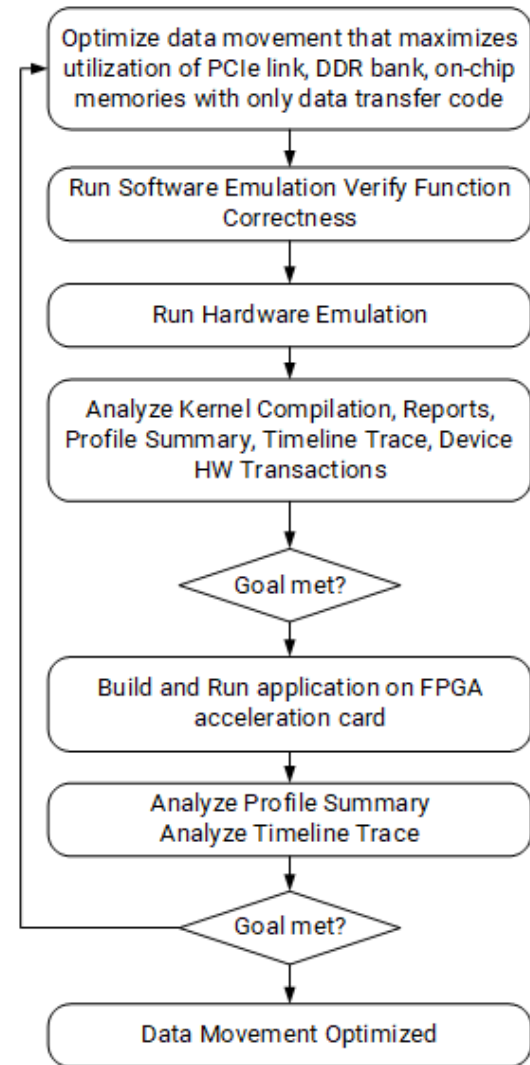
# Optimization

- Optimization possible at both host and kernel
- Enables most responsive solution
- Host optimization
- Kernel optimization possible in OpenCL and C/C++
  - Optimization syntax differs



# Host Optimization

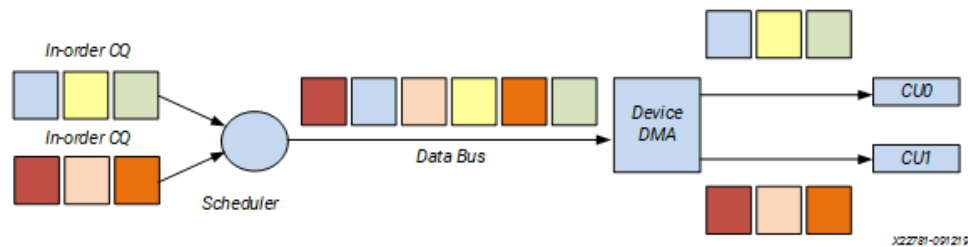
- Optimize the data movement in the application before optimizing computation
- Compute Unit Scheduling
  - Multiple In-Order Command Queues
  - Single Out-of-Order Command Queue



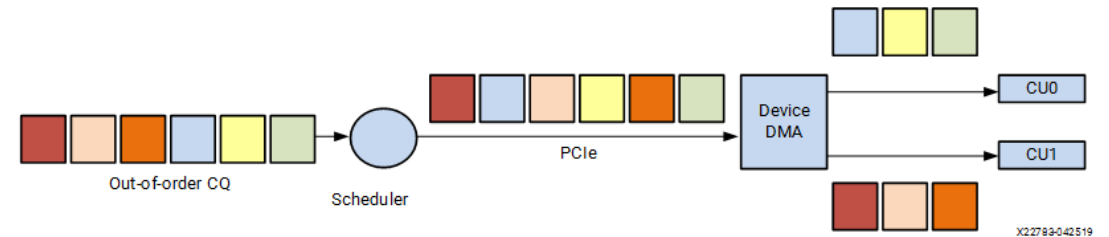
X22239-082719

# Host Optimization

Multiple In-Order Command Queues

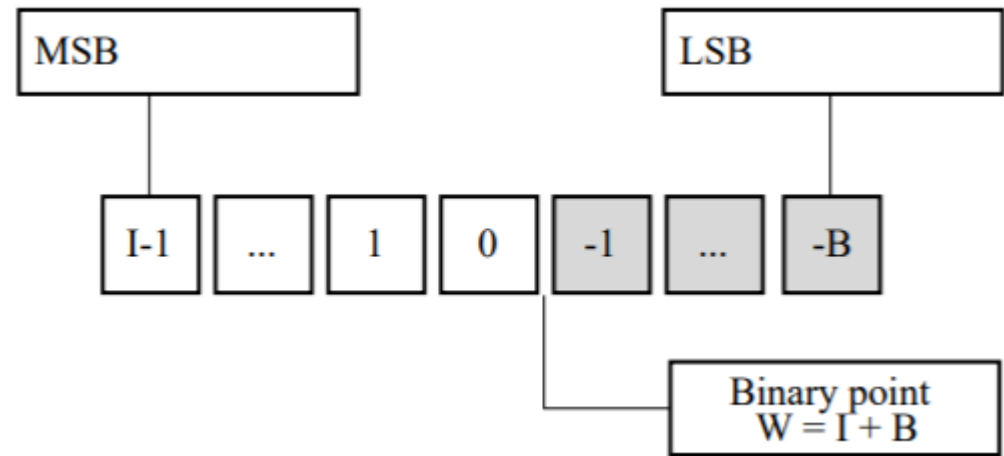


Single Out-of-Order Command Queue



# Kernel Optimization – Data Types

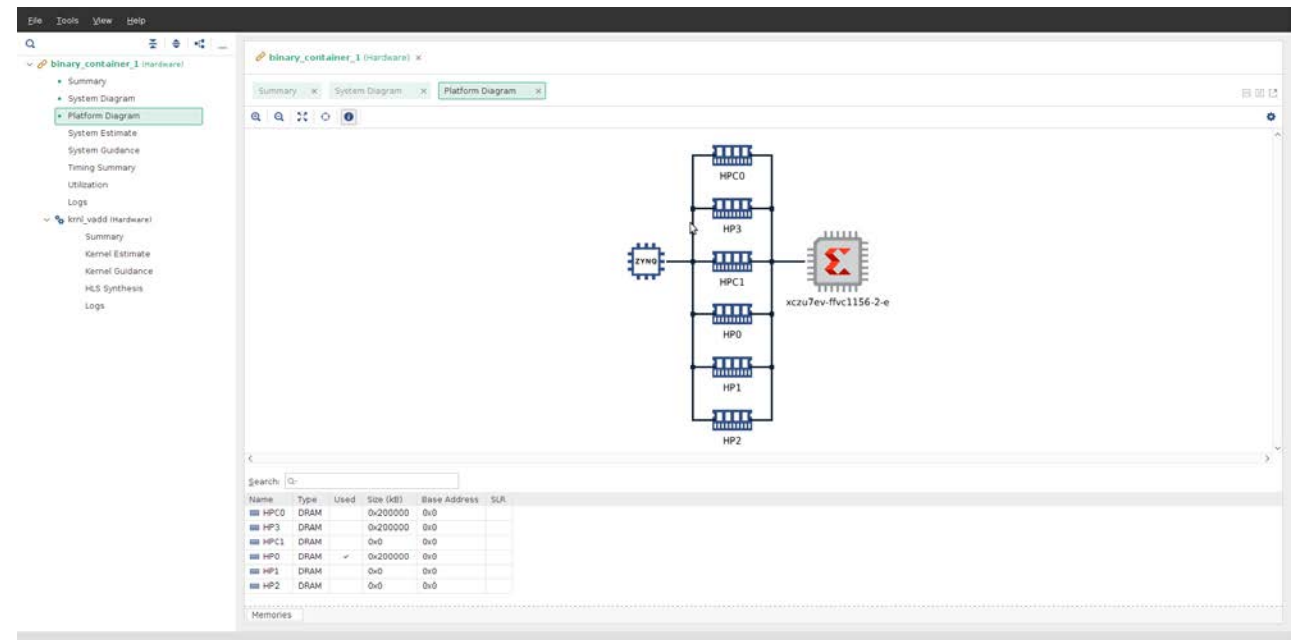
- Avoid native C data types (e.g., int, float, double)
- Best performance is using bit accurate types (C/C++ Kernels)
  - Arbitrary Precision Integer
  - Arbitrary Precision fixed point
- Enables smaller and faster logic implementations





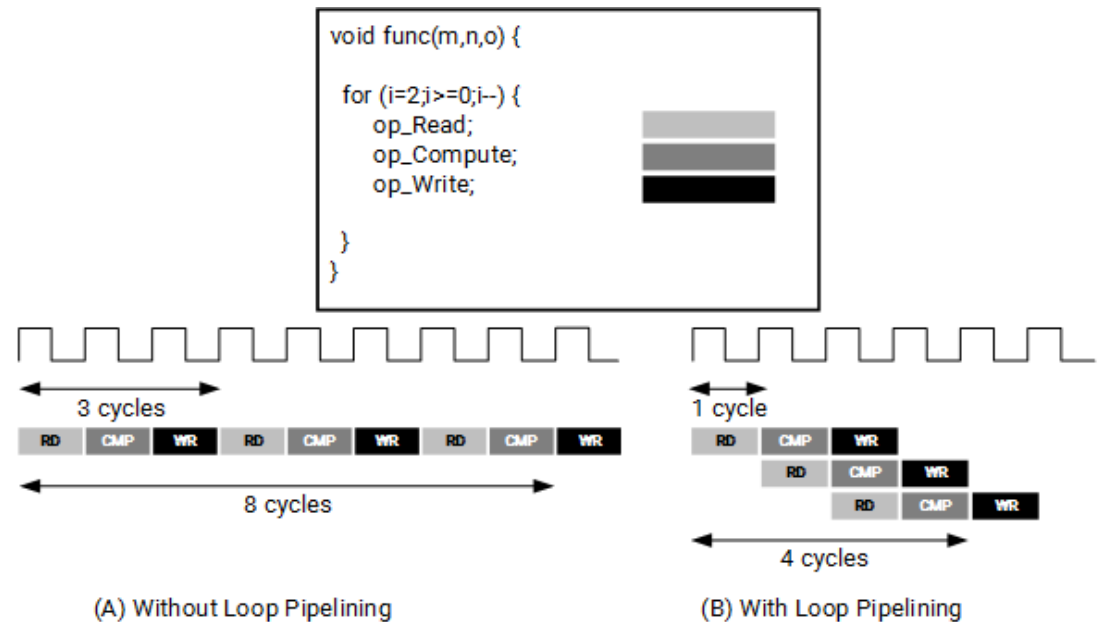
# Kernel Optimization – Interfacing

- Two types of data transfer
  - Data Pointers via global memory (M\_AXI)
  - Scalar direct to kernel (AXI\_LITE)
- Vitis automatically selects interface type
- Max data width is 512 bits – maximum performance leverages this



# Kernel Optimization – Pipelining

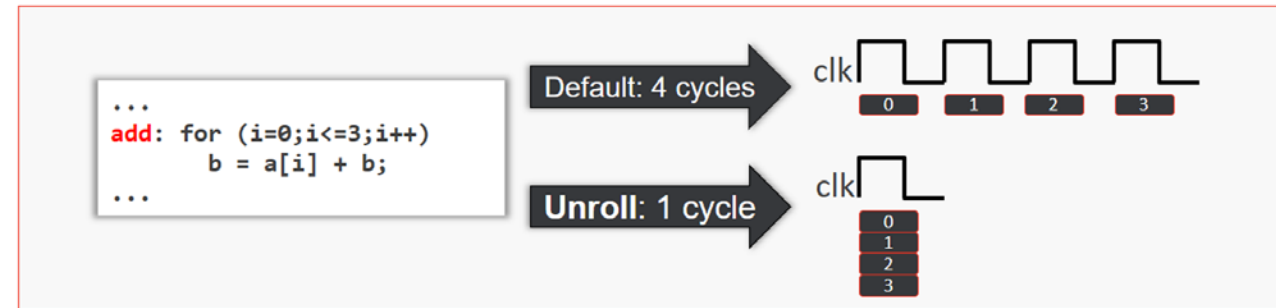
- By default, every iteration of a loop only starts when the previous iteration has finished
- Pipelining the loop executes subsequent iterations in a pipelined manner



X14277-110217

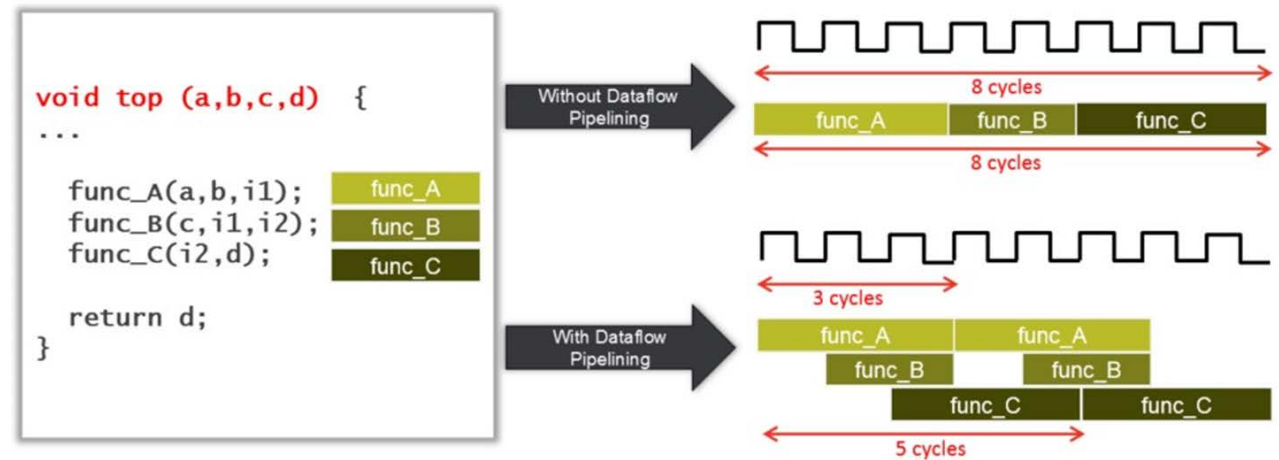
# Kernel Optimization – Unrolling

- Unrolling a loop enables the full parallelism
- Full or partial unroll
- Data dependencies in loops can impact the results of loop pipelining or unrolling



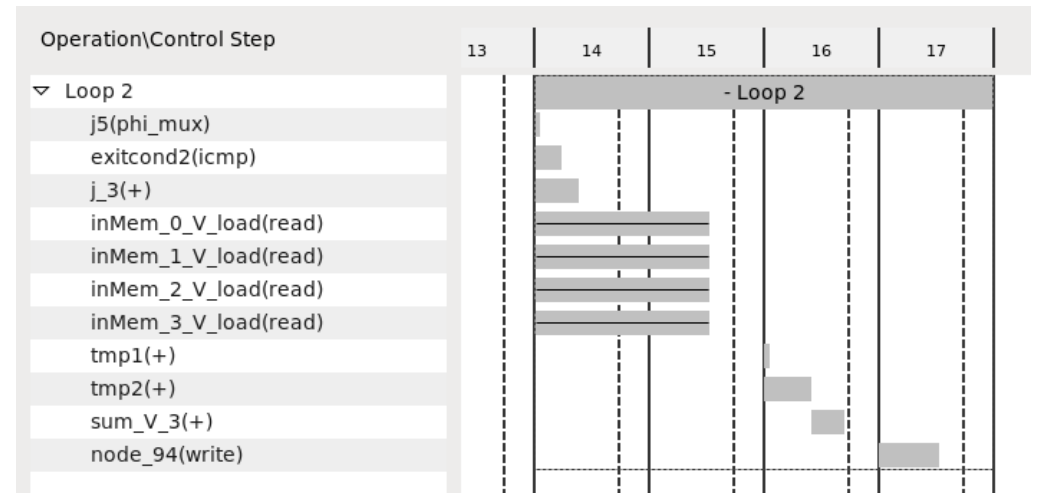
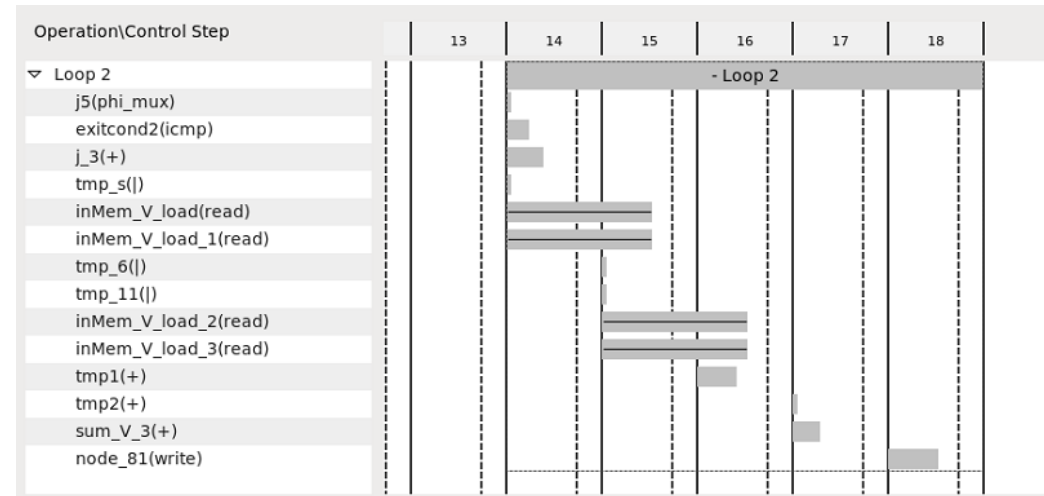
# Kernel Optimization – DataFlow

- Improve kernel performance by enabling task-level pipelining
- Be careful of:
  - Single producer-consumer violations.
  - Bypassing tasks.
  - Feedback between tasks.
  - Conditional execution of tasks.
  - Loops with multiple exit conditions or conditions defined within the loop



# Kernel Optimization – Memory

- Limited BRAM access bandwidth, can heavily impact the overall performance
- Ability to partition and reshape arrays can increase bandwidth
- Partition – Separates into different BRAMS
- Reshape – Allows combination of words



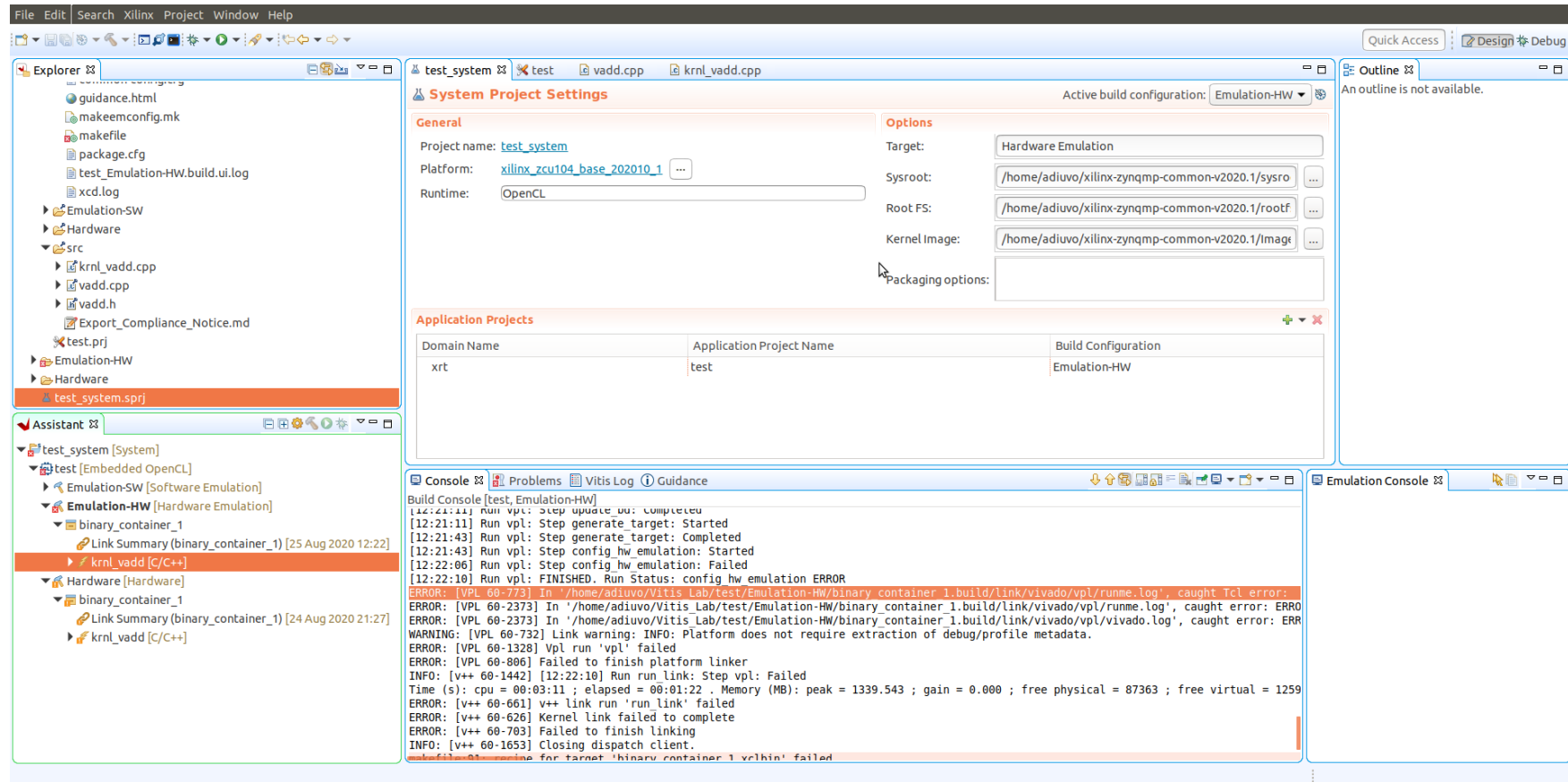
# Kernel Optimization - Pragmas

Optimization	C/C++	OpenCL
Pipeline	#pragma HLS PIPELINE	__attribute__((xcl_pipeline_loop))
Unroll	#pragma HLS UNROLL	__attribute__((opencl_unroll_hint))
DataFlow	#pragma HLS DATAFLOW	__attribute__((xcl_dataflow))
Memory	#pragma HLS ARRAY_PARTITION	

Further information can be found at

[https://www.xilinx.com/html\\_docs/xilinx2020\\_1/vitis\\_doc/optimizingperformance.html#fhe1553474153030](https://www.xilinx.com/html_docs/xilinx2020_1/vitis_doc/optimizingperformance.html#fhe1553474153030)

# Vitis GUI – Project Settings





# Vitis GUI – Project Setting

The screenshot displays the Vitis GUI interface with the 'Application Project Settings' window open. The 'General' tab is active, showing the following configuration:

- Project name: test
- Platform: xilinx\_zcu104\_base\_202010\_1
- Runtime: OpenCL
- Number of devices: 1

The 'Options' tab shows the following settings:

- Target: Hardware Emulation
- Host debug: ☒
- Kernel debug: ☒
- Kernel debug mode: Waveform
- Report level: Default
- Hardware optimization: Default optimization (-O0)

The 'Hardware Functions' table lists the components in the project:

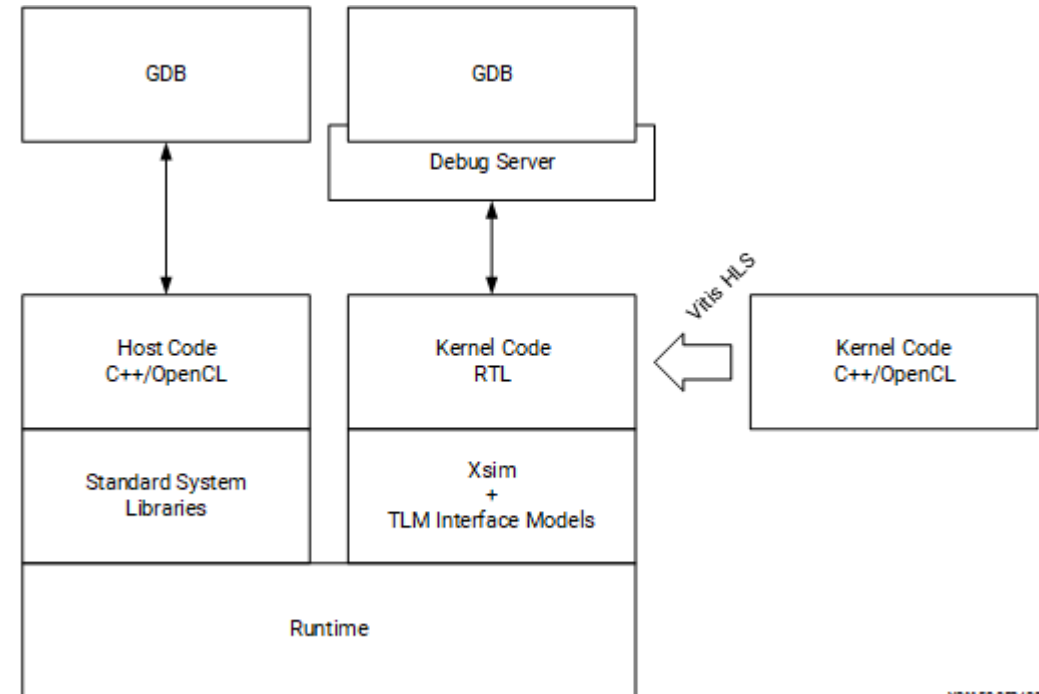
Name	Compute Units	Port Data Width	Max Memory Ports
binary_container_1			
krnl_vadd	1	Auto	

The 'Assistant' pane on the left shows the project hierarchy, with 'test' selected under 'Emulation-HW'. The 'Console' pane at the bottom displays the build log, which includes the following error messages:

```
ERROR: [VPL 60-732] In '/home/adiuvo/Vitis_Lab/test/Emulation-HW/binary_container_1.build/link/vivado/vpl/runme.log', caught Tcl error: ERROR: [VPL 60-2373] In '/home/adiuvo/Vitis_Lab/test/Emulation-HW/binary_container_1.build/link/vivado/vpl/runme.log', caught error: ERROR: [VPL 60-2373] In '/home/adiuvo/Vitis_Lab/test/Emulation-HW/binary_container_1.build/link/vivado/vpl/vivado.log', caught error: ERROR: [VPL 60-732] Link warning: INFO: Platform does not require extraction of debug/profile metadata. ERROR: [VPL 60-1328] Vpl run 'vpl' failed ERROR: [VPL 60-806] Failed to finish platform linker INFO: [v++ 60-1442] [12:22:10] Run run link: Step vpl: Failed Time (s): cpu = 00:03:11; elapsed = 00:01:22 . Memory (MB): peak = 1339.543; gain = 0.000; free physical = 87363; free virtual = 1259 ERROR: [v++ 60-661] v++ link run 'run link' failed ERROR: [v++ 60-626] Kernel link failed to complete ERROR: [v++ 60-703] Failed to finish linking INFO: [v++ 60-1653] Closing dispatch client. makefile-91: recipe for target 'binary_container_1_vclbin' failed
```

# Vitis-Debug

- Can Debug
  - Software Emulation
  - Hardware Emulation
- Hardware flow insert ILA
- Debugging will use QEMU and logic simulator



X21159052420

# Hands On Labs



# ADIUVO

ENGINEERING AND TRAINING, LTD.

[www.adiuvoengineering.com](http://www.adiuvoengineering.com)