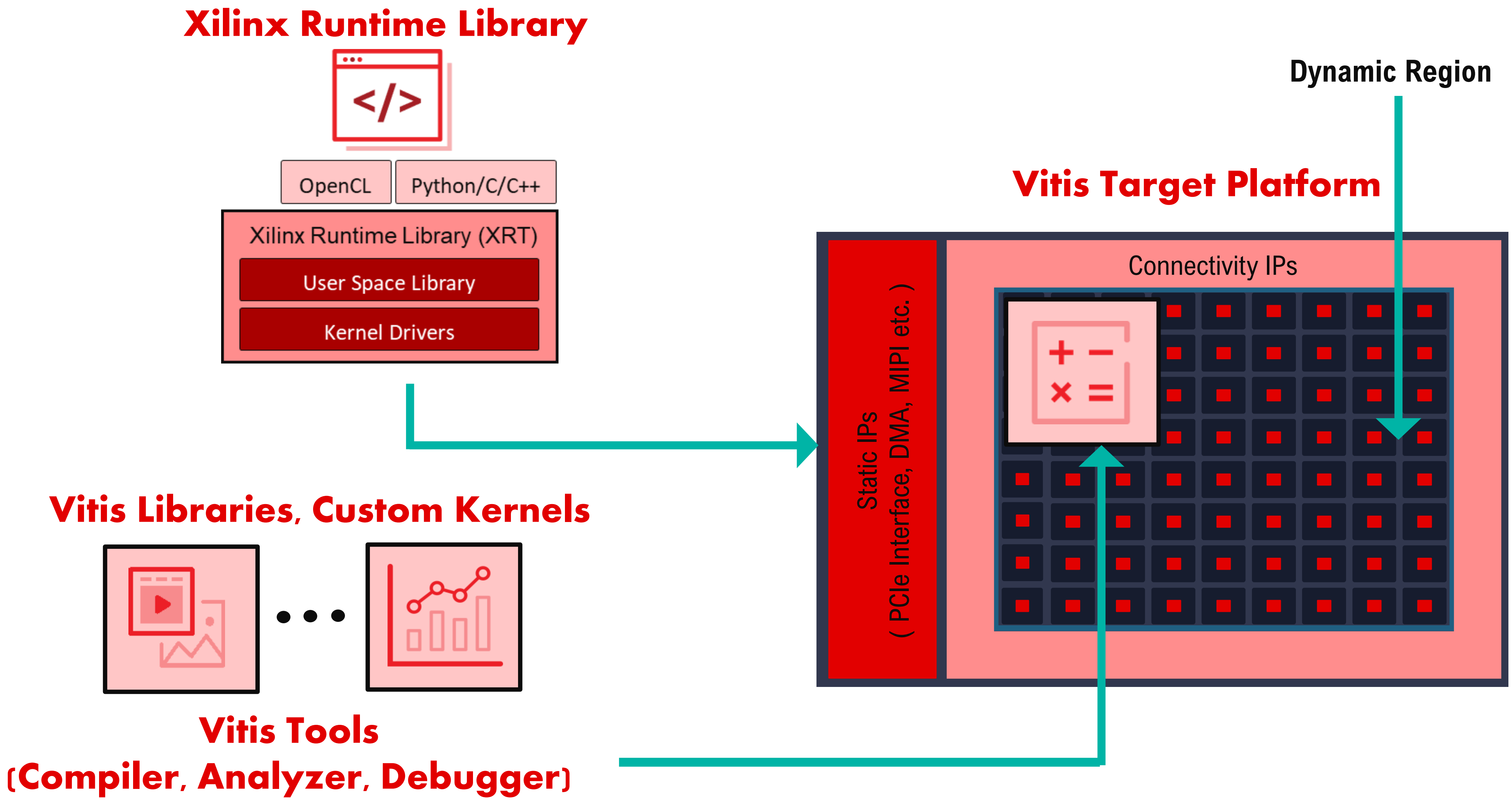
XILINX.

Vitis Accelerated Libraries

Xilinx University Program

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Algorithm to Deployment Using Key Components



Xilinx Runtime Library

OpenCL

Python/C/C++

Xilinx Runtime Library (XRT)

User Space Library

Kernel Drivers

Vitis Libraries, Custom Kernels

Vitis Tools (Compiler, Analyzer, Debugger)

Dynamic Region

Vitis Target Platform

Static IPs (PCIe Interface, DMA, MIPI etc.)

Connectivity IPs

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Steps to Accelerate Applications with Vitis

1 Profile Applications and Identify Performance-critical Functions

2 Design Accelerated Kernels

RTL IPs

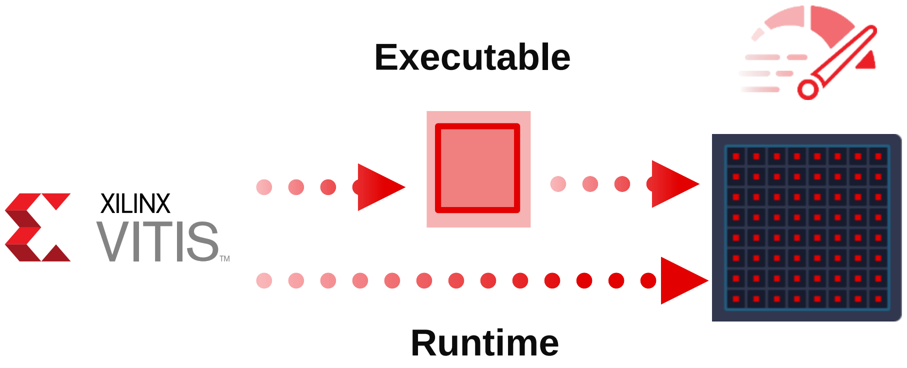
Libraries

C,C++

MATLAB & SIMULINK

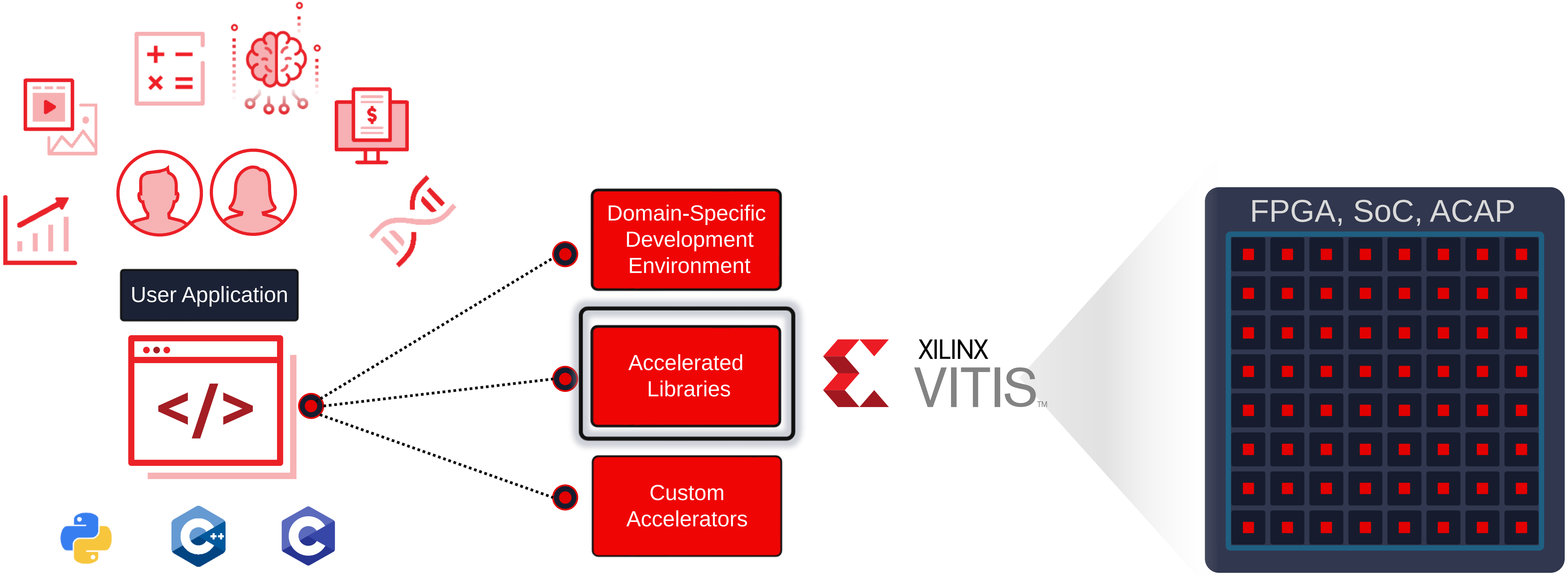
3 Build, Analyze & Debug : Validate Performance Goals Met

4 Deploy Accelerated Application on Xilinx Platforms



© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Software-Defined Application Acceleration



© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Build: Extensive, Open Source Libraries

GitHub

Domain-Specific Libraries

- Vision & Image
- Finance
- Data Analytics & Database
- Data Compression
- Data Security

Common Libraries

- Math
- Linear Algebra
- Statistics
- DSP
- Data Management

https://github.com/Xilinx/Vitis_Libraries

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Develop in Familiar Programming Languages

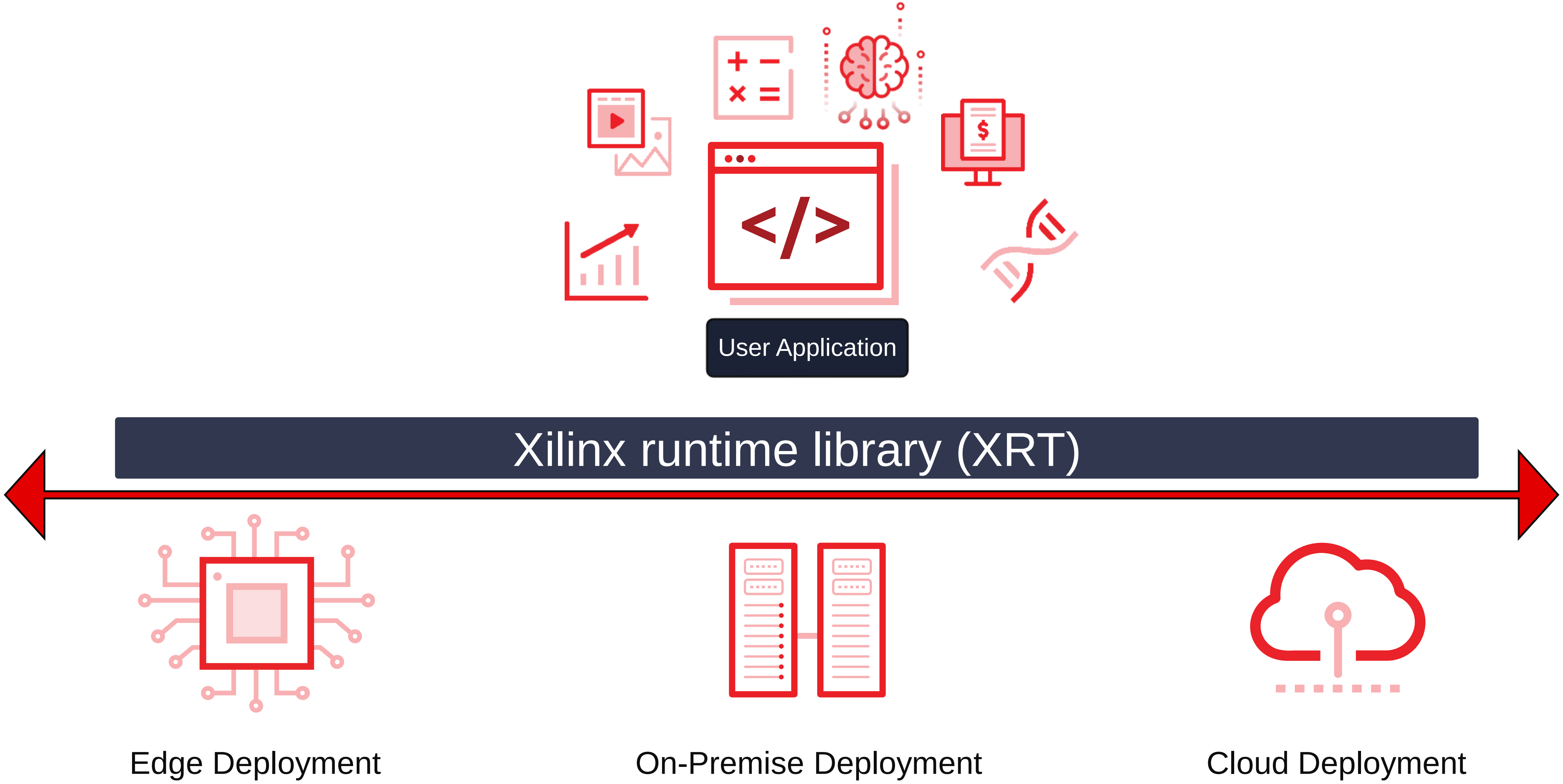
- Source code for library functions written in C/C++
  - Some offer Python APIs directly callable in Python applications
- Automatically synthesized to RTL by Vitis Compilers
  - Using C/C++ to RTL High-level Synthesis (HLS) Technology
- No prior RTL design experience required
  - Libraries enable a familiar CPU/GPU-like design experience

Focus Core Competencies on Algorithm and Application Development

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Scalable Across All Xilinx Platforms

Edge to Cloud Deployments



User Application

Xilinx runtime library (XRT)

Edge Deployment

On-Premise Deployment

Cloud Deployment

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Different Levels of Abstraction

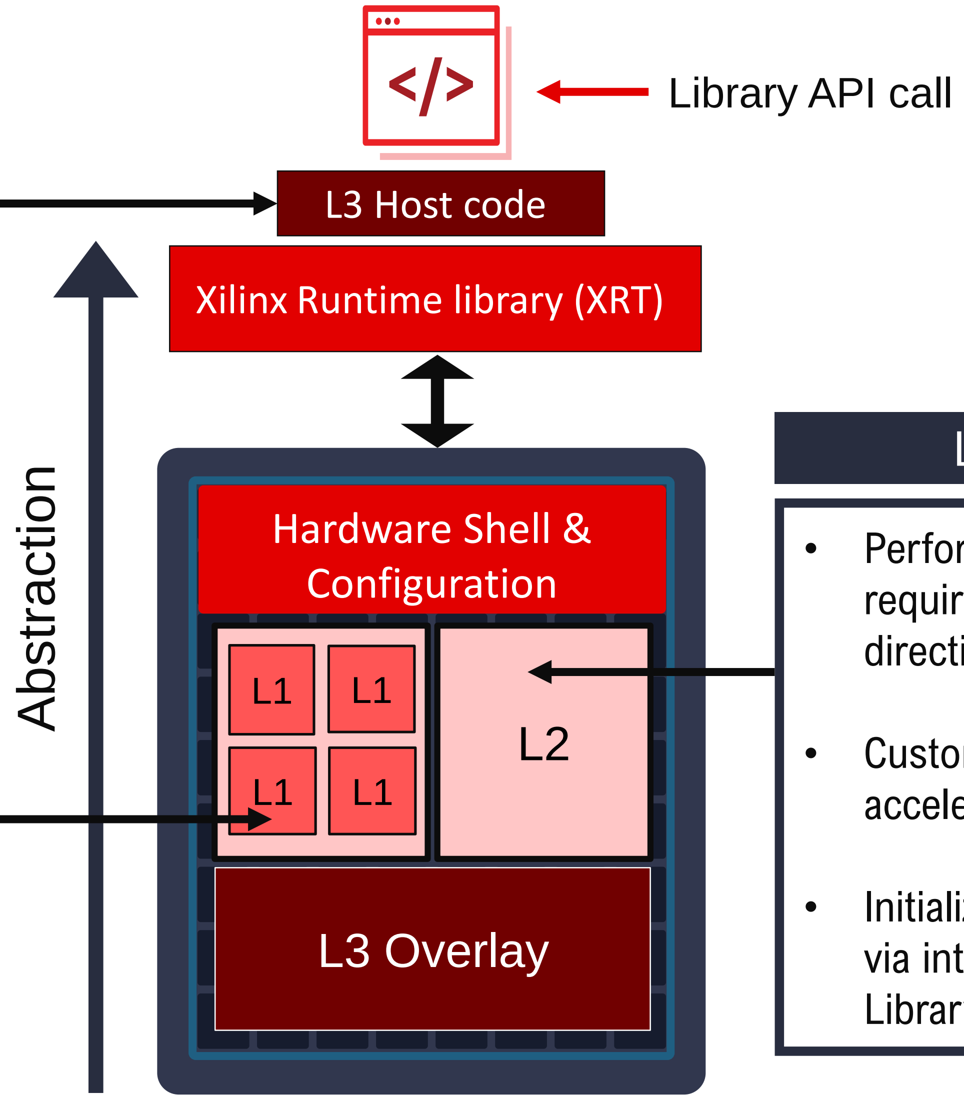
Leverage as-is, Modify or Combine with Custom Code

Library APIs (L3)

- High-level Software APIs directly callable in Host Application
- APIs call pre-built FPGA images, available for specific platforms
- Users can customize and rebuild

Library Primitives (L1)

- Basic algorithmic functions (HLS functions) for designing Kernels
- Customize or Combine with other Primitives and Kernels
- Requires compile and built and compiled with Vitis tools



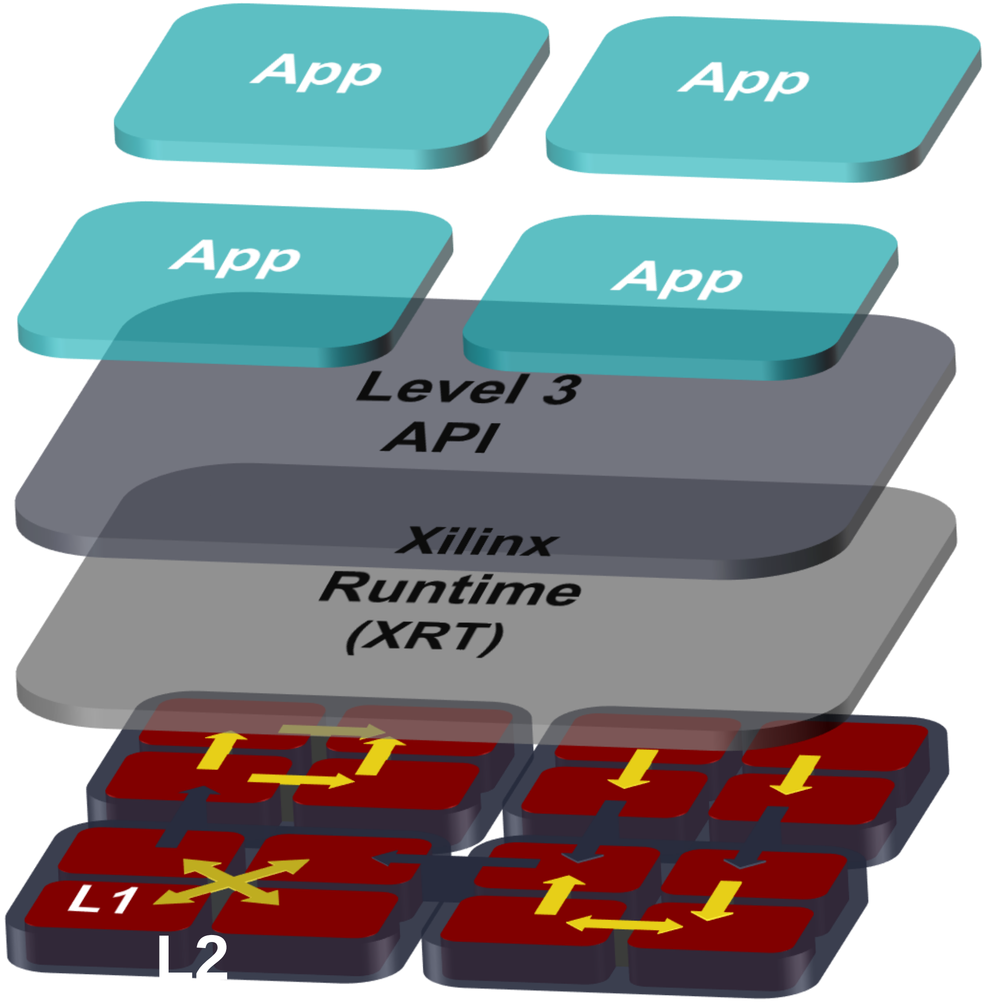
Library Kernels (L2)

- Performance-optimized Kernels, with required interfaces and compiler directives
- Customize, if needed, and build accelerator image with Vitis tools
- Initialization, Data Transfers managed via integration with Xilinx Runtime Library (XRT)

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Accelerated Libraries Structure

- User can interact with the libraries at all three levels
- Modify primitives for a particular application or used them as templates for new ones
- Customize or create new kernels
- Combine existing and custom primitives and kernels to create new libraries
- Modify the library API to support new functions and system configurations



© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

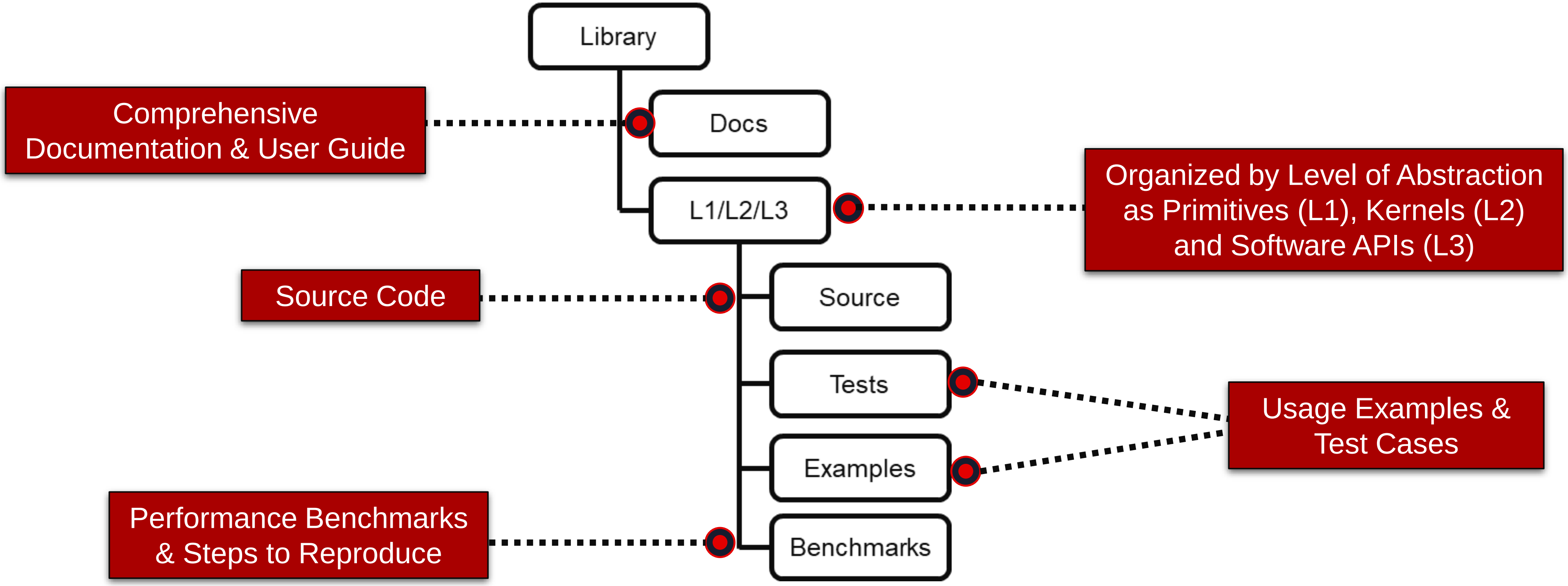
Open-Source & Available to All Developers on GitHub

Access Everything You Need to Get Started

Apache 2.0 License

GitHub

https://github.com/Xilinx/Vitis_Libraries



Comprehensive Documentation & User Guide

Organized by Level of Abstraction as Primitives (L1), Kernels (L2) and Software APIs (L3)

Source Code

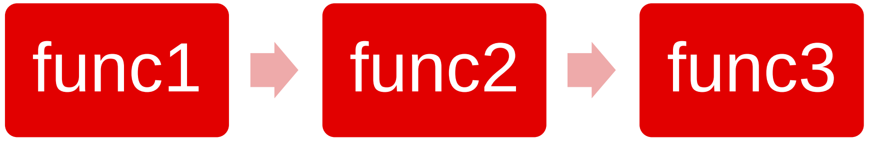
Usage Examples & Test Cases

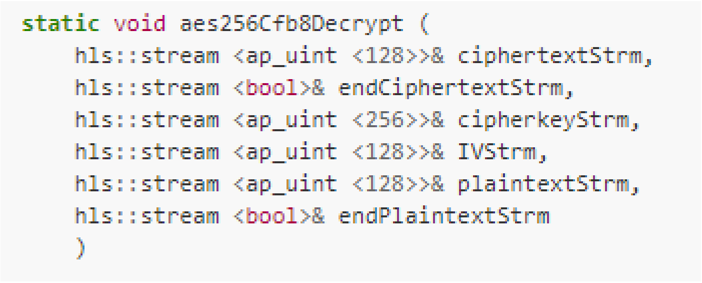
Performance Benchmarks & Steps to Reproduce

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

HLS C++ Kernel Methodology

- Deep pipeline of tasks is a key to gain performance advantage to CPU/GPU

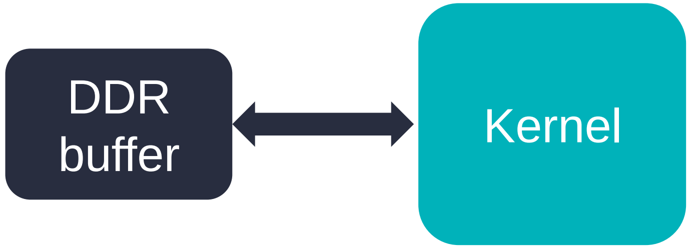


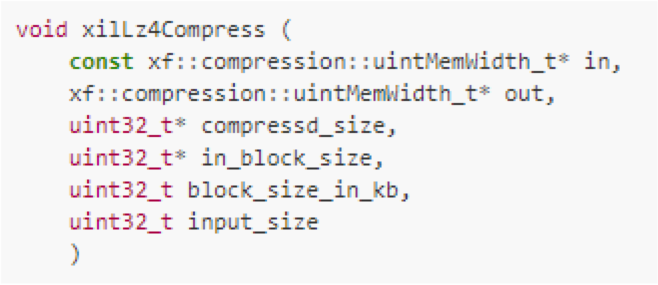


```
static void aes256Cfb8Decrypt (
    hls::stream <ap_uint <128>>& ciphertextStrm,
    hls::stream <bool>& endCiphertextStrm,
    hls::stream <ap_uint <256>>& cipherkeyStrm,
    hls::stream <ap_uint <128>>& IVStrm,
    hls::stream <ap_uint <128>>& plaintextStrm,
    hls::stream <bool>& endPlaintextStrm
    )
```

  - So many of the L1 primitives have streaming(FIFO) interface.

- L2 kernels typical work with DDR input, so they often have pointer interface.

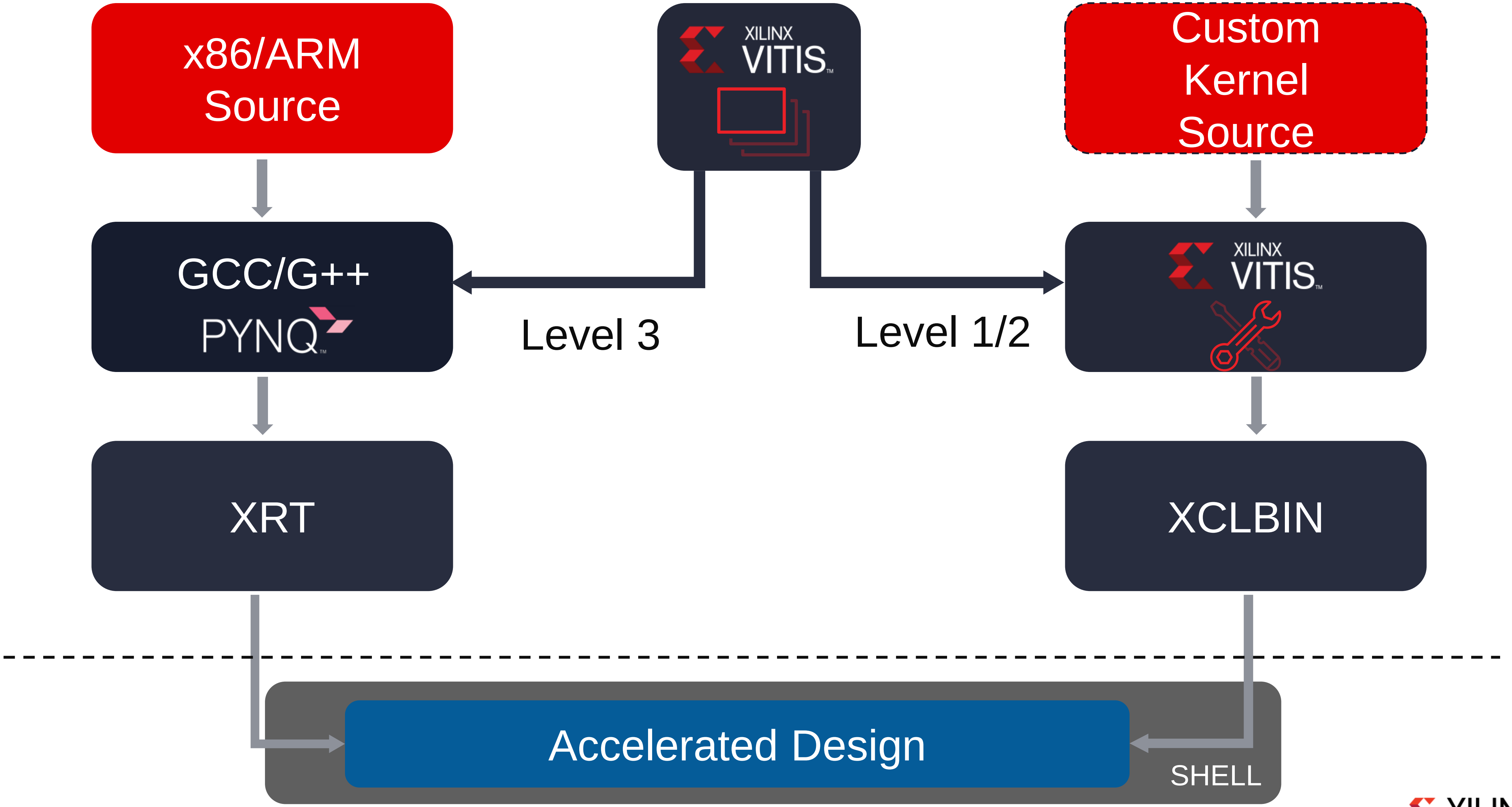




```
void xilLz4Compress (
    const xf::compression::uintMemWidth_t* in,
    xf::compression::uintMemWidth_t* out,
    uint32_t* compressd_size,
    uint32_t* in_block_size,
    uint32_t block_size_in_kb,
    uint32_t input_size
    )
```

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Design Flows



x86/ARM Source

GCC/G++ PYNQ

XRT

Custom Kernel Source

XCLBIN

Level 3

Level 1/2

Accelerated Design

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Libraries Description

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

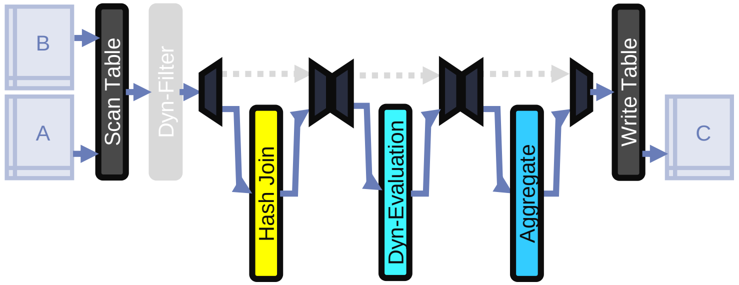
Vitis Vision Library

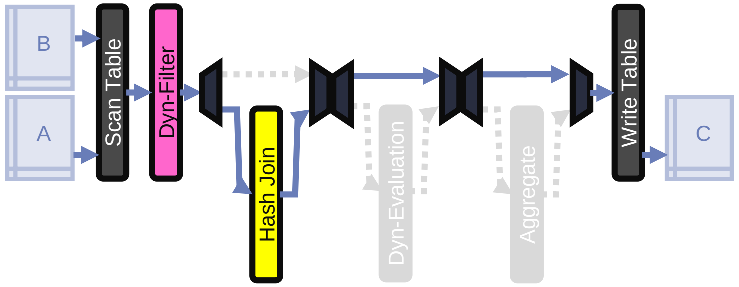
- Performance-optimized kernel and primitive functions for
  - Color and bit-depth conversion, channel extractions, pixel-wise arithmetic ops.
  - Geometric transforms, image statistics, image filters
  - Feature detection and classifiers
  - 3D reconstructions
  - Motion Analysis and Tracking
- Support for color image processing and multi-channel support
- Multiple pixel/clock processing to meet through requirements
- Familiar OpenCV API interface

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Database Library

- Accelerate both data-intensive and compute-intensive applications common in Relation Database Management
- Optimized implementation of execution plan steps, like hash-join and aggregation
- The kernels can be used to map a sequence of execution plan steps, without having to compile different binaries for each query.

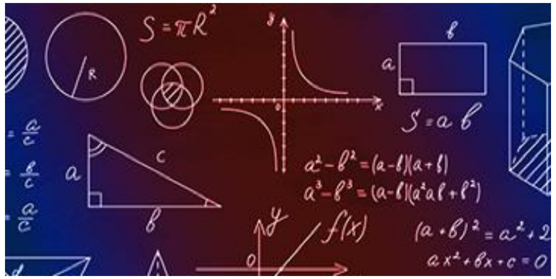




© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis BLAS Library

- Performance-optimized implementation of Basic Linear Algebra Subroutines (BLAS)
- General Matrix Multiply (GEMM) and General Matrix-Vector (GEMV) APIs available as pre-compiled accelerators with C, C++, and Python interfaces
- Drop-in and replace CPU and GPU-based BLAS operations for rapid prototyping and evaluation
- Leverage library primitives and kernels to design unique accelerated algorithms



© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Data Compression Library

- Performance optimized library to accelerate the Lempel-Ziv (LZ) data compression and decompression algorithms.
- Scalable compression engine can be instantiated multiple times and run concurrently to meet high-throughput demands.
- Off-the-Shelf LZA and Snappy compression/decompression available.
- Use the low-level primitives as components to design your own.

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Data Security Library

- Brings real-time performance to security applications
- Block ciphers like Advanced Encryption Standard (AES), and Data Encryption Standards (DES)
- Streaming ciphers like ChaCha20 and Rivest Cipher 4(RC4)
- Hashing methods like Message-Digest (MD) algorithms
- Secure Hash Algorithms (SHA-1, SHA-2, SHA-3) BLAKE2, and SHAKE

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Quantitative Finance Library

- Optimized functions allows user to build accelerated computational solutions for financial workloads.
  - Options-pricing
  - Modeling
  - Trading
  - Evaluation and risk management
- Library APIs can be called directly in your C, C++, and Python host applications.
- Multiple examples available
  - Heston Finite Difference
  - Monte Carlo Black Scholes American and European models

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Solver Library

- Performance-optimized standard matrix decomposition, linear solvers, and eigen value solvers
- Accelerate applications across multiple domains
  - Computational Finance
  - RADAR, LiDAR
  - Computer Vision
  - DSP, Controls
- Combine the library kernels to accelerate end-to-end processing pipelines

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

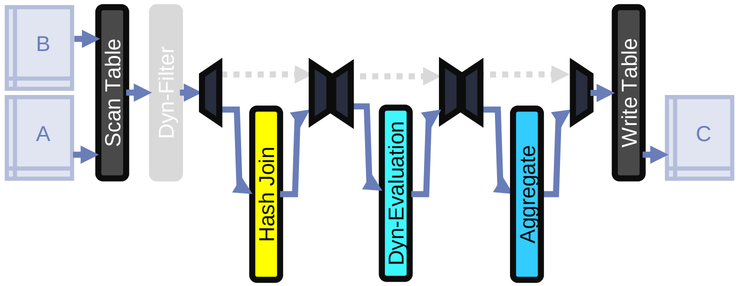
Vitis Data Analytics

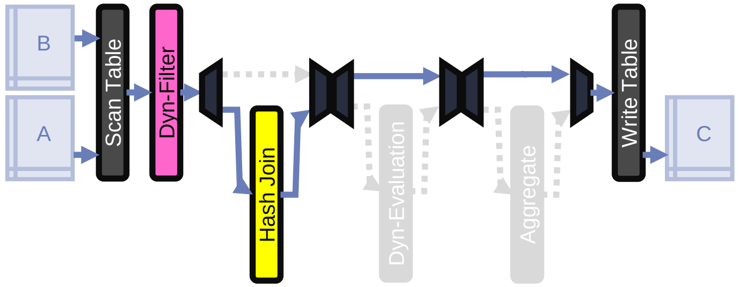
- Performance-optimized for data analytics applications
- Data Mining
  - *Classification; Clustering and Regression*
- Text Processing (2020.2)
  - *Regular expression; Geo-IP*
- DataFrame (2020.2)
  - Dtore and load multiple types of fixed and variable data length
- Multiple example available

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis Utilities Library

- The glue logic and common shims
- Streaming interfaces are used extensively, as a methodology. This library provides a list of APIs to manipulate streams, e.g.
  - Distribute data received through a FIFO in round-robin way to N FIFOs.
  - Blocks for reading DDR into FIFO or writing FIFO into DDR.
    - It’s not always easy to make it right first time, so we provide “known-to-work” modules for basic patterns.
- Example: Utilities being used in Vitis Database Library





© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

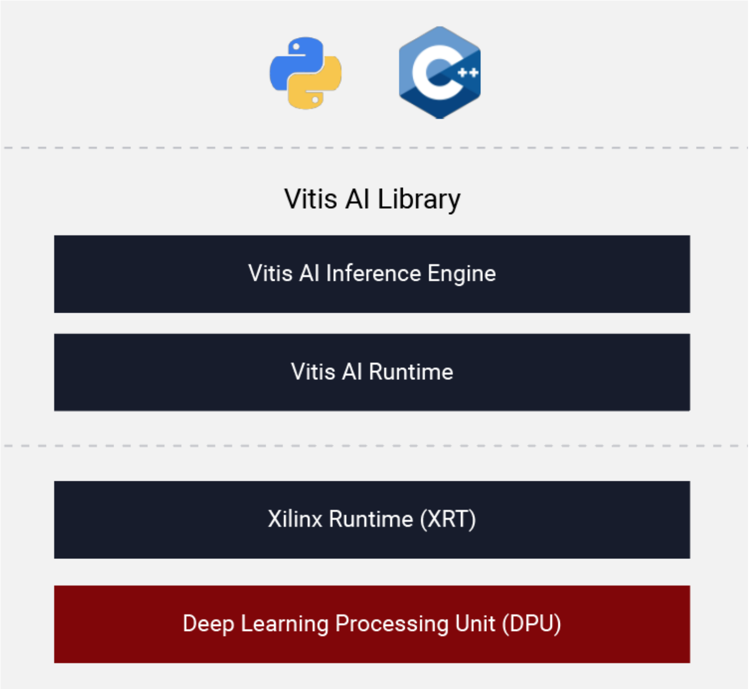
Other Vitis Libraries

- Sparse
- DSP
- Graph
- Codec
- HPC

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Vitis AI Library

- Lightweight set of C++ and Python APIs enabling easy application development.
- Handles all task scheduling, memory management, and interrupt handing.



Vitis AI Library

Vitis AI Inference Engine

Vitis AI Runtime

Xilinx Runtime (XRT)

Deep Learning Processing Unit (DPU)

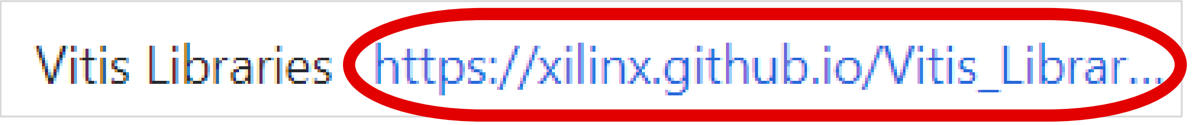
© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Get Started

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

How to learn a library

- Start from the document:



Vitis Libraries https://xilinx.github.io/Vitis_Librar...

  - The landing page of document and sub-repo top README usually answers “what is in here”
  - The HTML doc provide for each level
    - API list: the brief doc on parameters and arguments
    - Design info: usually contains block diagram, resource/performance info
- Each API has a test
  - The quickest way to known “how to use this API”
  - L1/tests are HLS tests, rest are all Vitis tests.

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

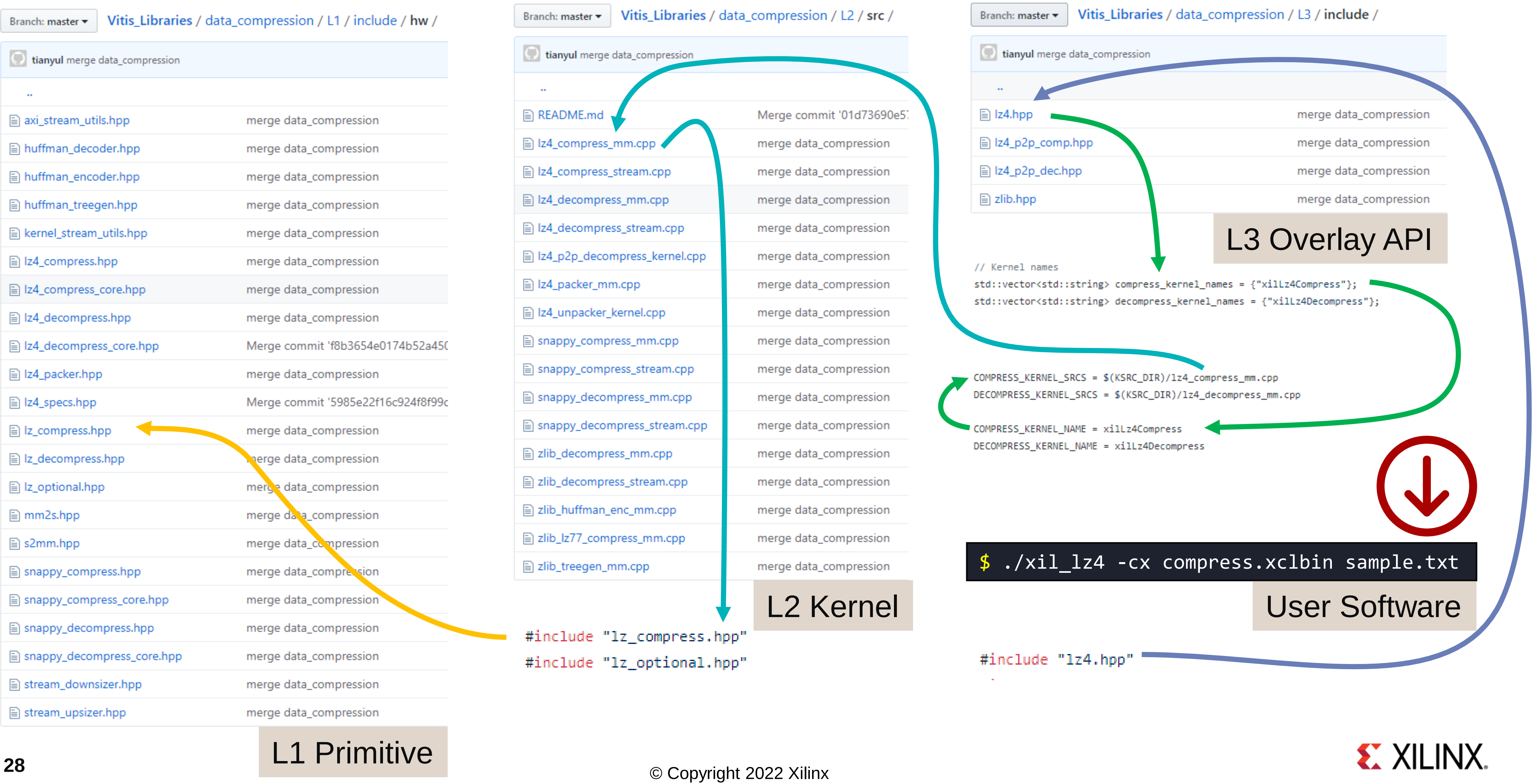
Run Tests/Benchmarks/Demos by `make`

- All the projects in all libraries are driven by makefiles.
- `make help` for general information.
- Typical use

```
$ source <install path>/Vitis/2021.1/settings64.sh
$ source /opt/xilinx/xrt/setup.sh
$ export PLATFORM_REPO_PATHS=/opt/xilinx/platforms
# Vitis case
$ make run DEVICE=/opt/xilinx/platforms/xilinx_u250_xdma_201830_2/
xilinx_u250_xdma_201830_2.xpfm TARGET=hw_emu
# HLS case
$ make run DEVICE=/opt/xilinx/platforms/xilinx_u250_xdma_201830_2/
xilinx_u250_xdma_201830_2.xpfm COSIM=1
```

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Real Example: Data Compression Lib



Branch: master ▾ Vitis_Libraries / data_compression / L1 / include / hw /

tianyul merge data_compression

..	
axi_stream_utils.hpp	merge data_compression
huffman_decoder.hpp	merge data_compression
huffman_encoder.hpp	merge data_compression
huffman_treegen.hpp	merge data_compression
kernel_stream_utils.hpp	merge data_compression
lz4_compress.hpp	merge data_compression
lz4_compress_core.hpp	merge data_compression
lz4_decompress.hpp	merge data_compression
lz4_decompress_core.hpp	Merge commit 'f8b3654e0174b52a45(
lz4_packer.hpp	merge data_compression
lz4_specs.hpp	Merge commit '5985e22f16c924f8f99c
lz_compress.hpp	merge data_compression
lz_decompress.hpp	merge data_compression
lz_optional.hpp	merge data_compression
mm2s.hpp	merge data_compression
s2mm.hpp	merge data_compression
snappy_compress.hpp	merge data_compression
snappy_compress_core.hpp	merge data_compression
snappy_decompress.hpp	merge data_compression
snappy_decompress_core.hpp	merge data_compression
stream_downsizer.hpp	merge data_compression
stream_upsizer.hpp	merge data_compression

L1 Primitive

Branch: master ▾ Vitis_Libraries / data_compression / L2 / src /

tianyul merge data_compression

..	
README.md	Merge commit '01d73690e57
lz4_compress_mm.cpp	merge data_compression
lz4_compress_stream.cpp	merge data_compression
lz4_decompress_mm.cpp	merge data_compression
lz4_decompress_stream.cpp	merge data_compression
lz4_p2p_decompress_kernel.cpp	merge data_compression
lz4_packer_mm.cpp	merge data_compression
lz4_unpacker_kernel.cpp	merge data_compression
snappy_compress_mm.cpp	merge data_compression
snappy_compress_stream.cpp	merge data_compression
snappy_decompress_mm.cpp	merge data_compression
snappy_decompress_stream.cpp	merge data_compression
zlib_decompress_mm.cpp	merge data_compression
zlib_decompress_stream.cpp	merge data_compression
zlib_huffman_enc_mm.cpp	merge data_compression
zlib_lz77_compress_mm.cpp	merge data_compression
zlib_treegen_mm.cpp	merge data_compression

L2 Kernel

```
#include "lz_compress.hpp"
#include "lz_optional.hpp"
```

Branch: master ▾ Vitis_Libraries / data_compression / L3 / include /

tianyul merge data_compression

..	
lz4.hpp	merge data_compression
lz4_p2p_comp.hpp	merge data_compression
lz4_p2p_dec.hpp	merge data_compression
zlib.hpp	merge data_compression

L3 Overlay API

```
// Kernel names
std::vector<std::string> compress_kernel_names = {"xilLz4Compress"};
std::vector<std::string> decompress_kernel_names = {"xilLz4Decompress"};
```

```
COMPRESS_KERNEL_SRCS = $(KSRC_DIR)/lz4_compress_mm.cpp
DECOMPRESS_KERNEL_SRCS = $(KSRC_DIR)/lz4_decompress_mm.cpp

COMPRESS_KERNEL_NAME = xilLz4Compress
DECOMPRESS_KERNEL_NAME = xilLz4Decompress
```

```
$ ./xil_lz4 -cx compress.xclbin sample.txt
```

User Software

```
#include "lz4.hpp"
```

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

License

- Licensed under Apache 2.0 license, which is quite permissive.
  - Users do not need to pay Xilinx for the code.
  - Users can charge their customers for products built with our libraries.
  - Users can modify the code, or give it to anyone without telling Xilinx.
  - Commercial use permitted

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

Remarks

- Check if functionality is supported
  - Adapt
  - Improve
  - Reuse
- New releases brings improvements/enhancements and new functions/domains
  - Vitis 2020.2 HPC Library
    - Optimized for FP32

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.

XILINX®

Thank You

© Copyright 2022 Xilinx
For XUP teaching use only. Not for republication or redistribution.