XILINX®

# Welcome!

# Development with Vitis Accelerated Libraries

Uttara Kumar
Product Marketing Manager

Frédéric Rivoallon
Product Manager

# Agenda

- Quick Recap on Vitis Unified Software Platform
- Introduction to Vitis Accelerated Libraries
- Examples: Linear Algebra, Vision and Quantitative Finance Libraries
- Demo Walkthrough
- Getting Started Resources

# All Developers Can Build and Deploy on All Platforms

Build

Embedded Developers

Enterprise Application Developers

Enterprise Infrastructure Developers

Data & AI Scientists

XILINX VITIS™

Deploy

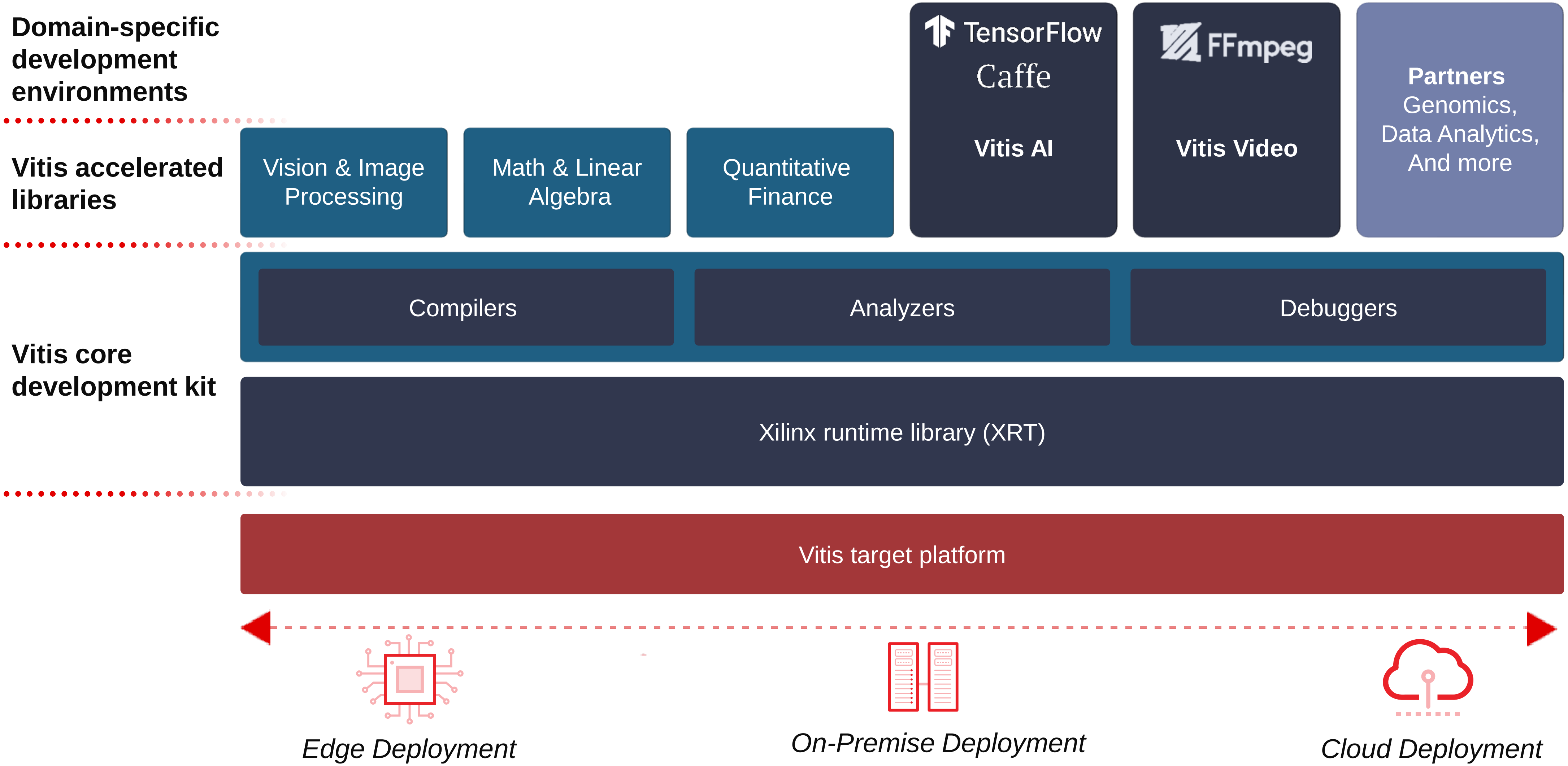
Zynq-7000

Zynq UltraScale+ MPSoC

Alveo

Versal ACAPs

# Vitis Unified Software Platform



**Domain-specific development environments**

TensorFlow
Caffe
**Vitis AI**

FFmpeg
**Vitis Video**

**Partners**
Genomics,
Data Analytics,
And more

**Vitis accelerated libraries**

Vision & Image Processing

Math & Linear Algebra

Quantitative Finance

**Vitis core development kit**

Compilers

Analyzers

Debuggers

Xilinx runtime library (XRT)

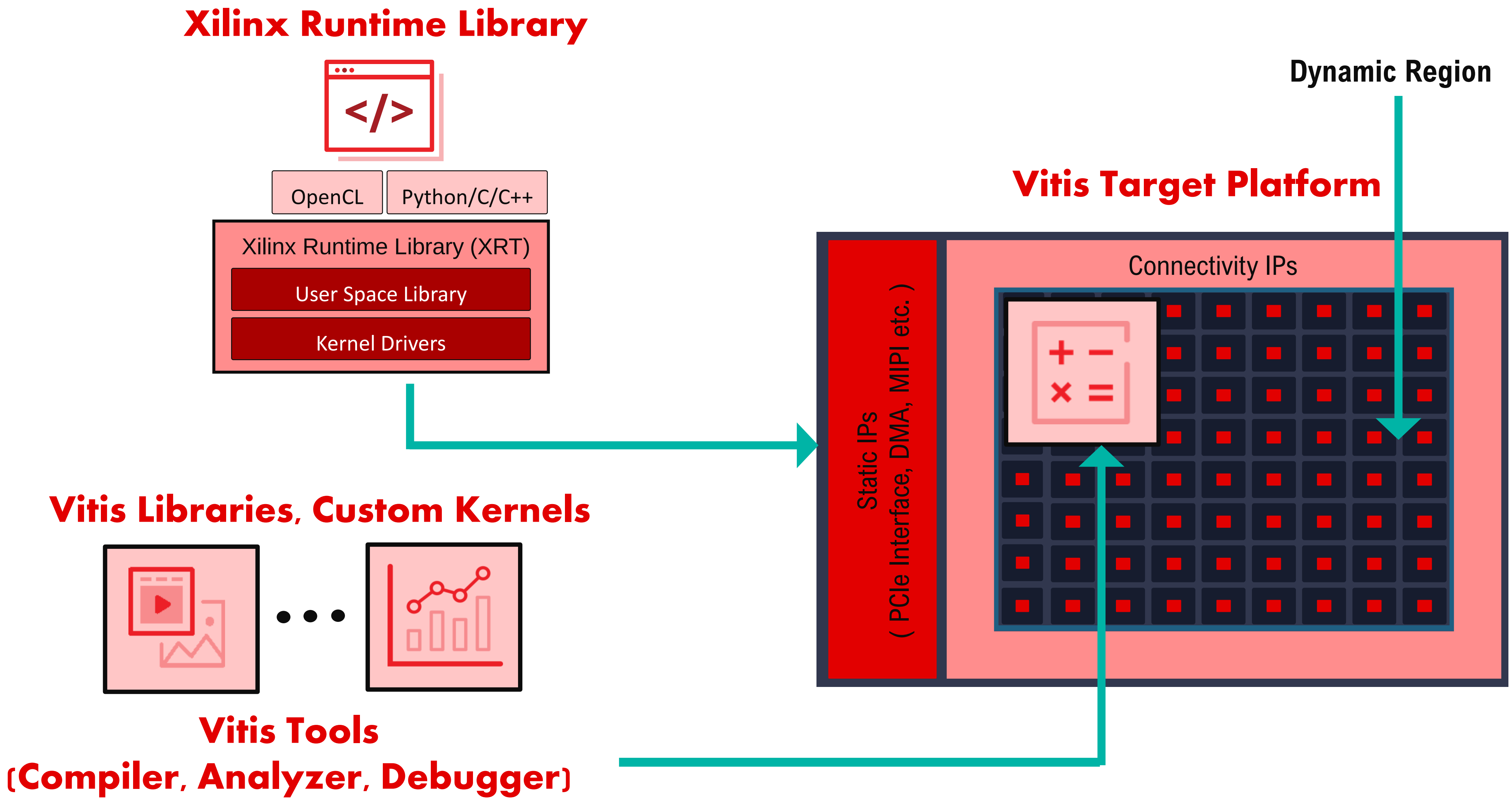
Vitis target platform

*Edge Deployment*

*On-Premise Deployment*

*Cloud Deployment*

# 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

Connectivity IPs

Static IPs

( PCIe Interface, DMA, MIPI etc. )

# Steps to Accelerate Applications with Vitis

1 Profile Applications and Identify Performance-critical Functions

2 Design Accelerated Kernels

RTL IPs

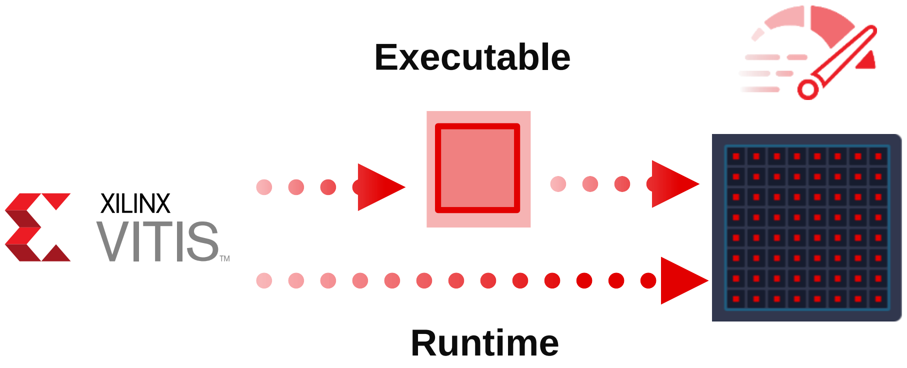
Libraries

C,C++

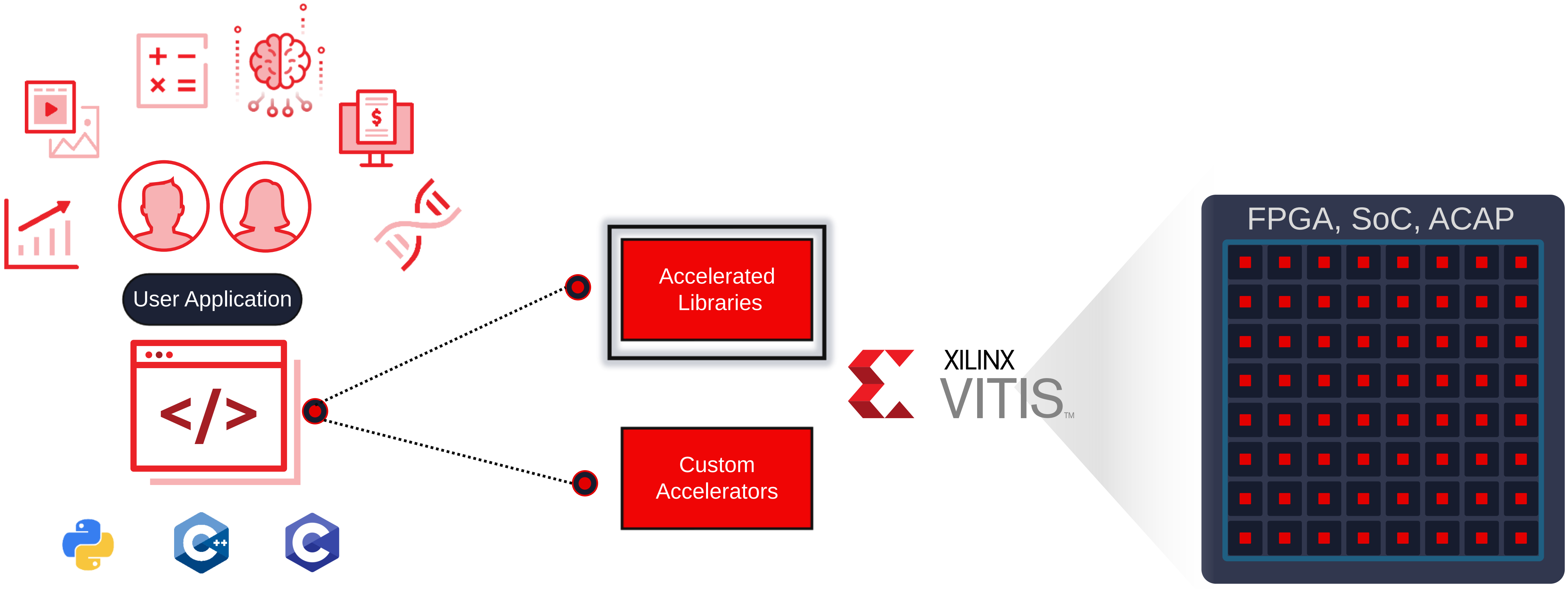
MATLAB & SIMULINK

3 Build, Analyze & Debug : Validate Performance Goals Met

4 Deploy Accelerated Application on Xilinx Platforms



# Designing Accelerated-Kernels



# Use Extensive, Open Source Libraries

## Domain-Specific Libraries

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

Partner Libraries

## Common Libraries

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

**400+ functions across multiple libraries for performance-optimized out-of-the-box acceleration**

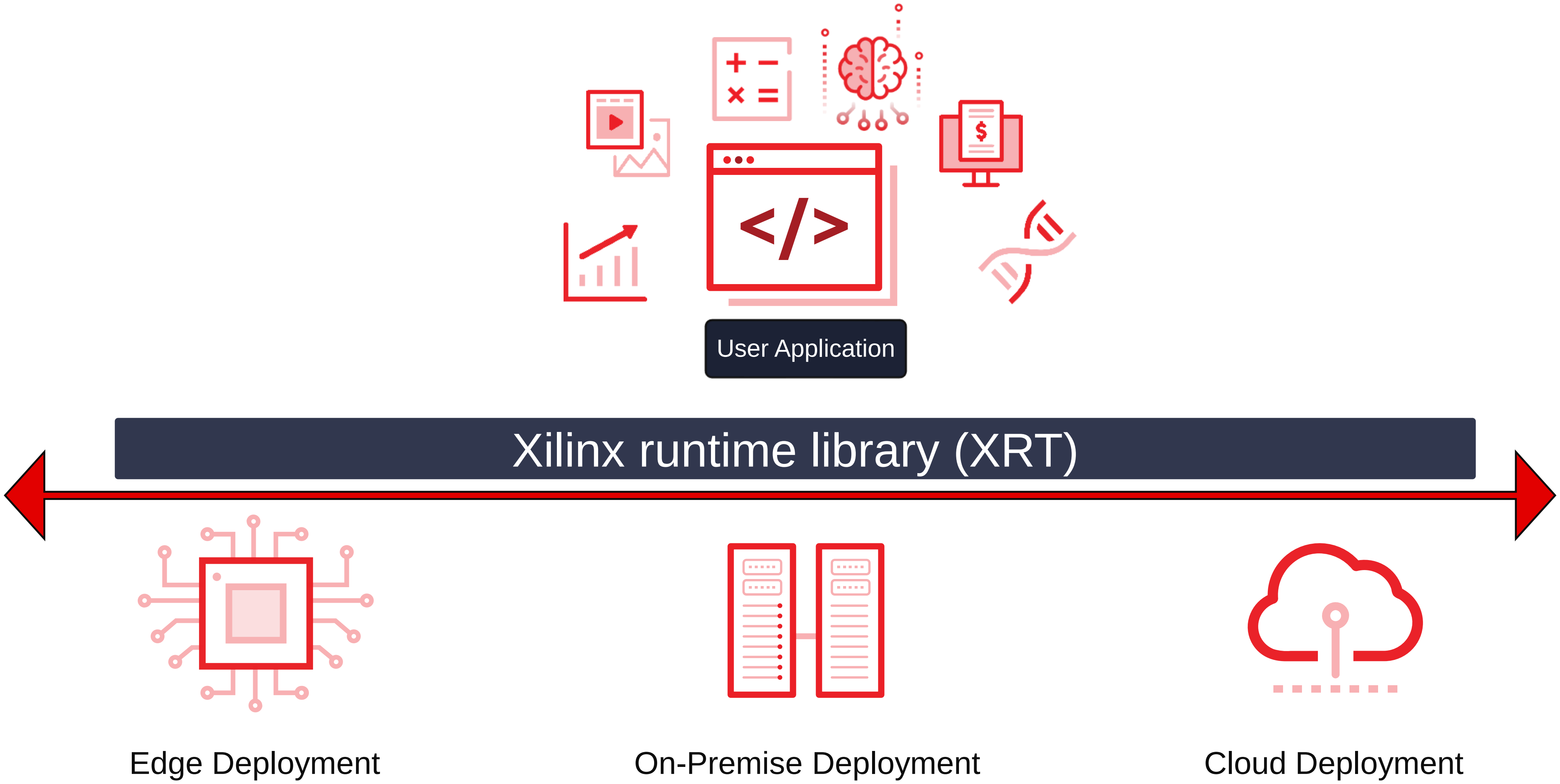
# Develop in Familiar Programming Languages

- 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

# Scalable Across All Xilinx Platforms

*Edge to Cloud Deployments*



User Application

Xilinx runtime library (XRT)

Edge Deployment

On-Premise Deployment

Cloud Deployment

# Different Levels of Abstraction

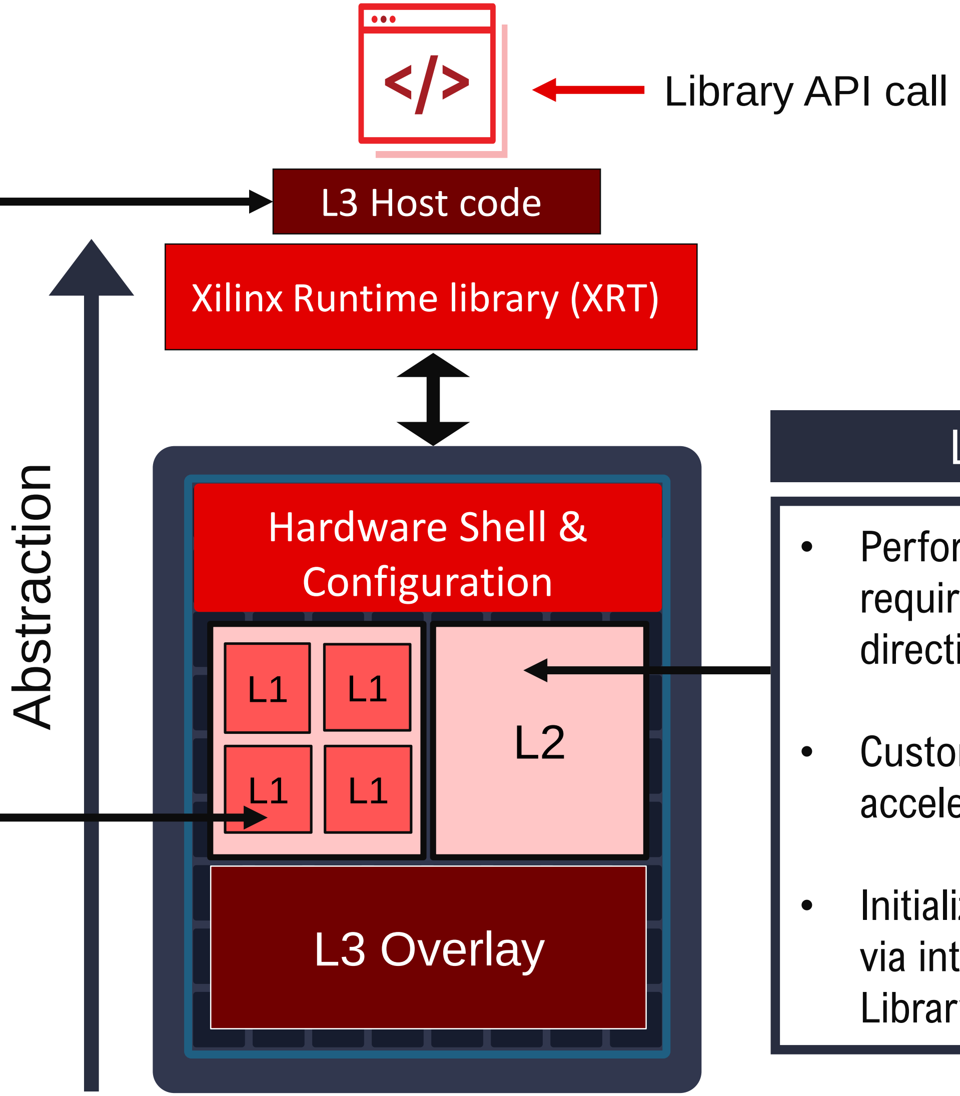
*Leverage as-is, Modify or Combine with Custom Code*

## 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)

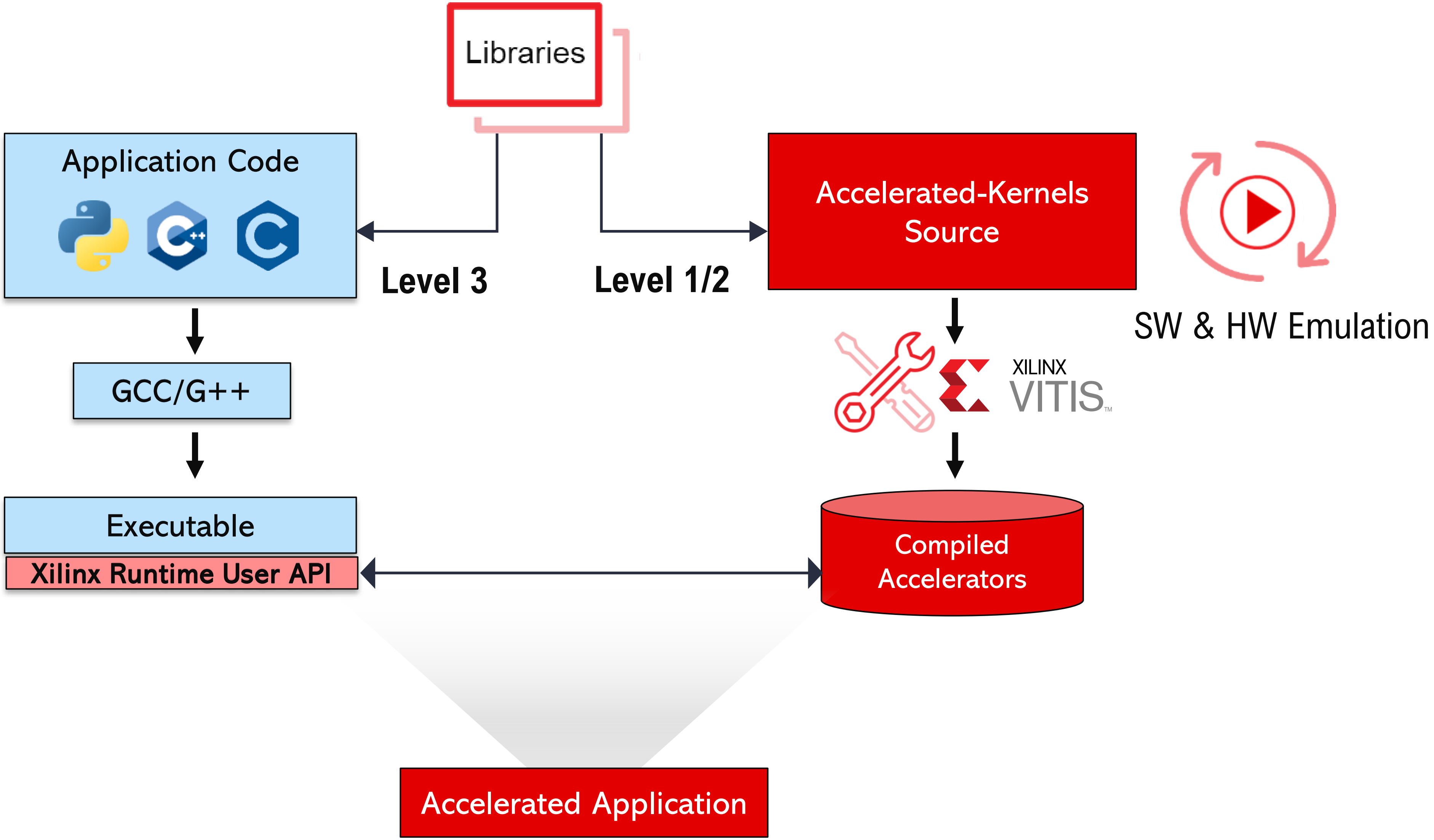
- Basic algorithmic functions (HLS functions) for designing Kernels
- Customize or Combine with other Primitives and Kernels
- Requires compile and built and compiled with Vitis tools



## 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)

# Different Levels of Abstraction : Use Model



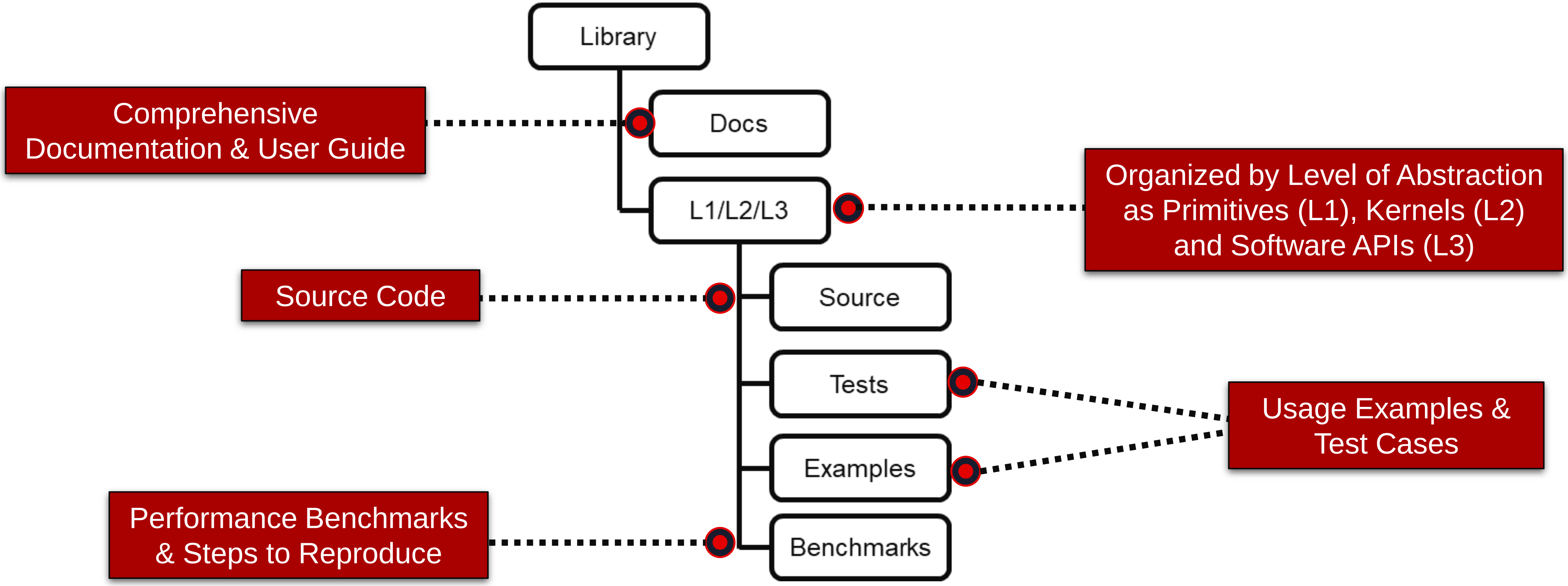
# Open-Source & Available to All Developers on GitHub

*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

# Vitis Vision Library

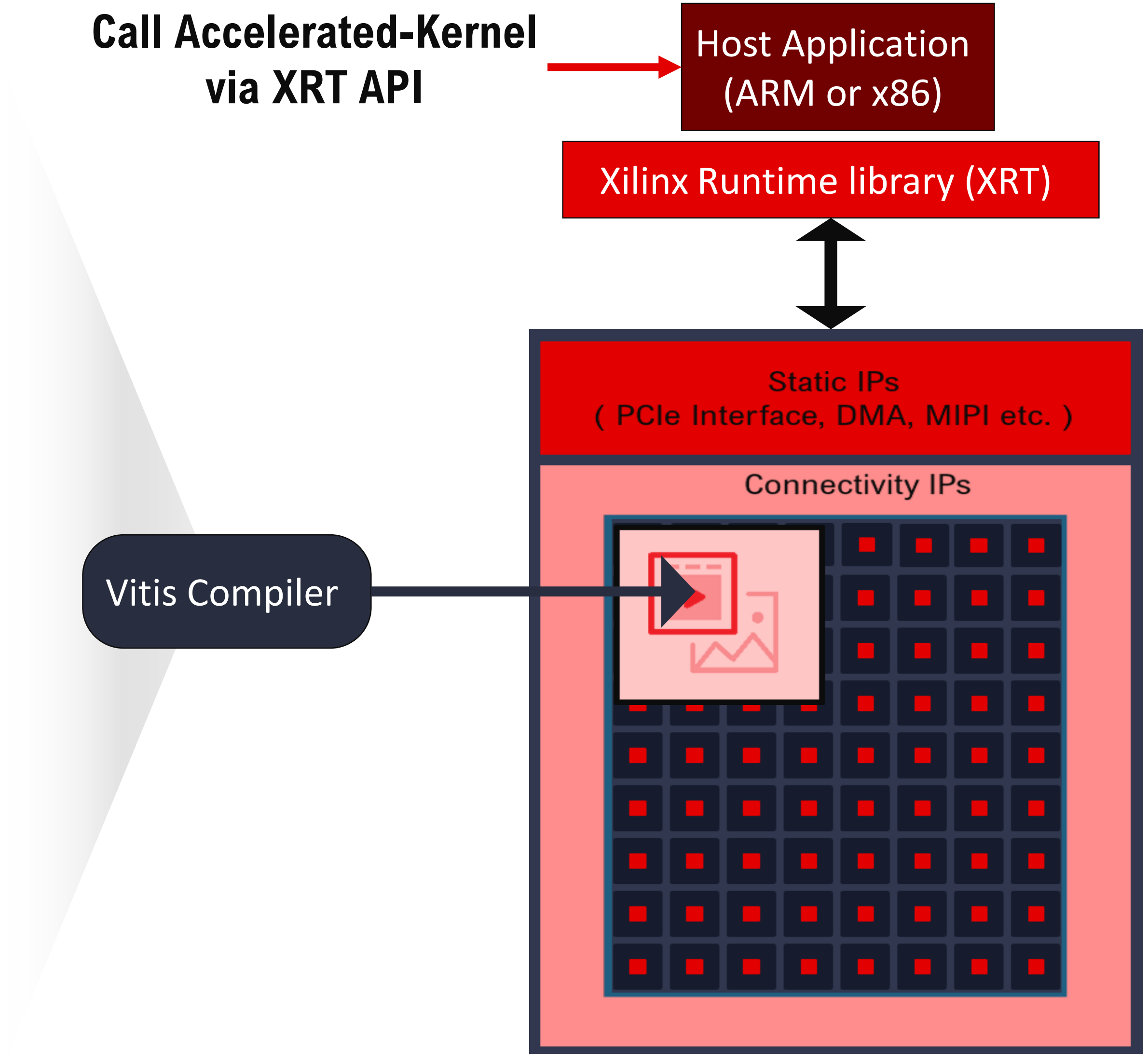
- **Performance-Optimized Library for Vision & Image Processing**
  - FPGA-accelerated versions of familiar OpenCV functions
  - Functions support 4K Frame Sizes
  - Supports both Memory-Mapped and Streaming Applications
  - Multiple pixel/clock processing to meet through requirements

- **Accelerated Functions Include :**
  - Geometric Transforms, Image Statistics, Image Filters
  - Feature Detection and Classifiers
  - 3D Reconstruction, Motion Analysis and Tracking

- **Design Examples**
  - Iterative Pyramidal Optical Flow, Corner Tracking, Color Detection
  - Accelerating Pre-Processing Pipeline for DNN Inferencing

# Vitis Vision Library : General Flow

```
Accelerated-Kernel ( )

{

// Copy data to xf::cv::Mat

xf::cv::Array2xfMat<…>(image_in, in_mat);

// Call accelerated functions

xf::cv::resize<…>(in_mat,resized_mat);

xf::cv::GaussianBlur<…>(resized_mat, out_mat,sigma);

// Copy data from xf::cv::Mat

xf::cv::xfMat2Array<…>(out_mat, image_out);

}
```



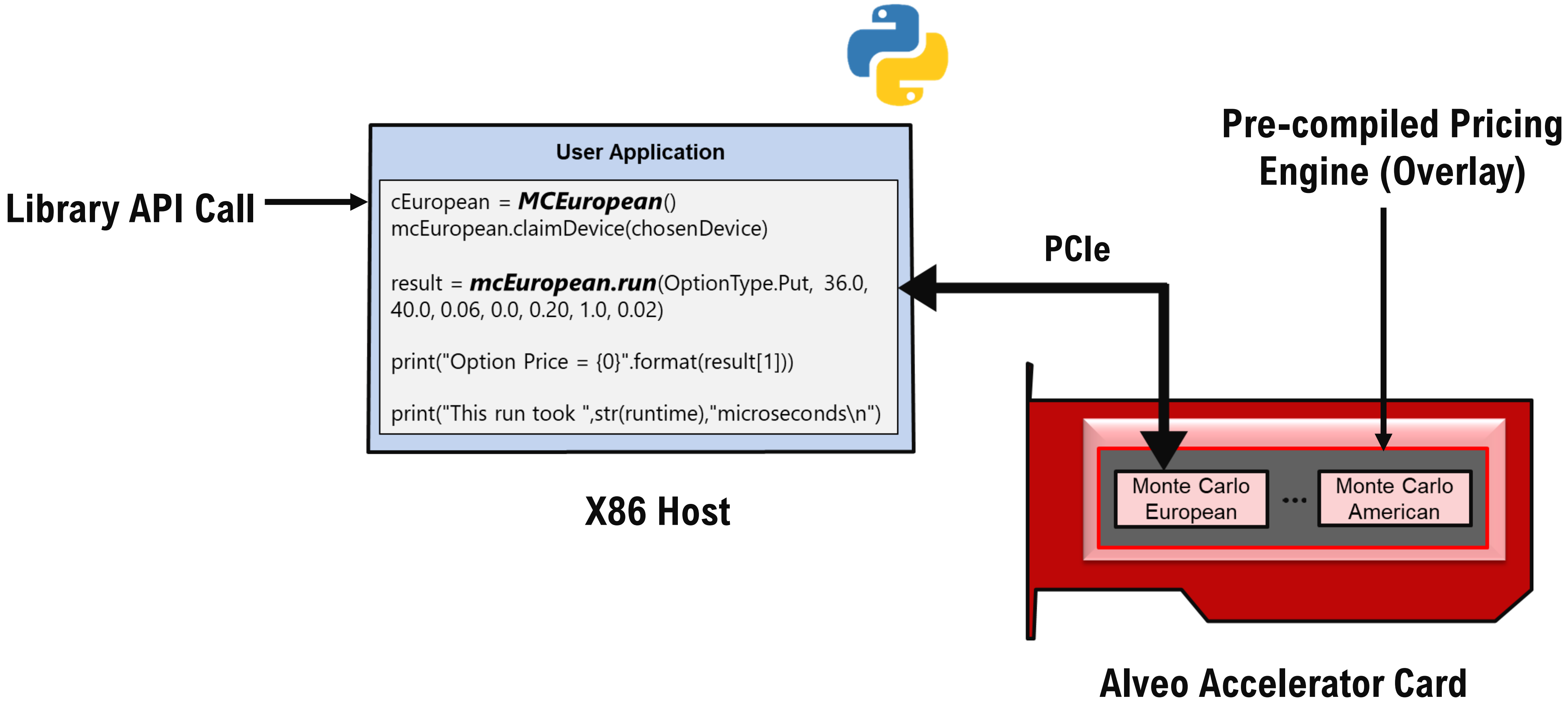
# Vitis Quantitative Finance Library

- **FPGA-Accelerated Library for Financial Workloads**
  - Leverage Xilinx platforms to accelerate time-to-insight and lower TCO
  - Includes plug-and-play library APIs for rapid evaluation of performance
  - Offers performance-optimized primitives to build custom pipelines

- **Applications and Use-Cases**
  - Monte Carlo-based Pricing Models
  - Finite Difference Based Pricing Models
  - Closed-Form Solutions
  - Trade/Portfolio Risk

| Plug-and-Play APIs (L3) | Primitive Function Examples (L1,L2) | |
|---|---|---|
| Monte Carlo American& European Options Pricing | Random Number Generators (Mersenne Twister, Gaussian, LUT-SR) | Inverse Cumulative Normal Distribution |
| Binomial Tree Engine using CRR | Box-Muller Transform | Pentadiagonal Solver |
| Heston FD & Closed Form | Jacobi SVD | Brownian Bridge Transform |
| Merton 76 Jump Diffusion Model | Tridiagonal Solver | Multi-dimensional Sobol |

# Vitis Quantitative Finance Library Level 3 API



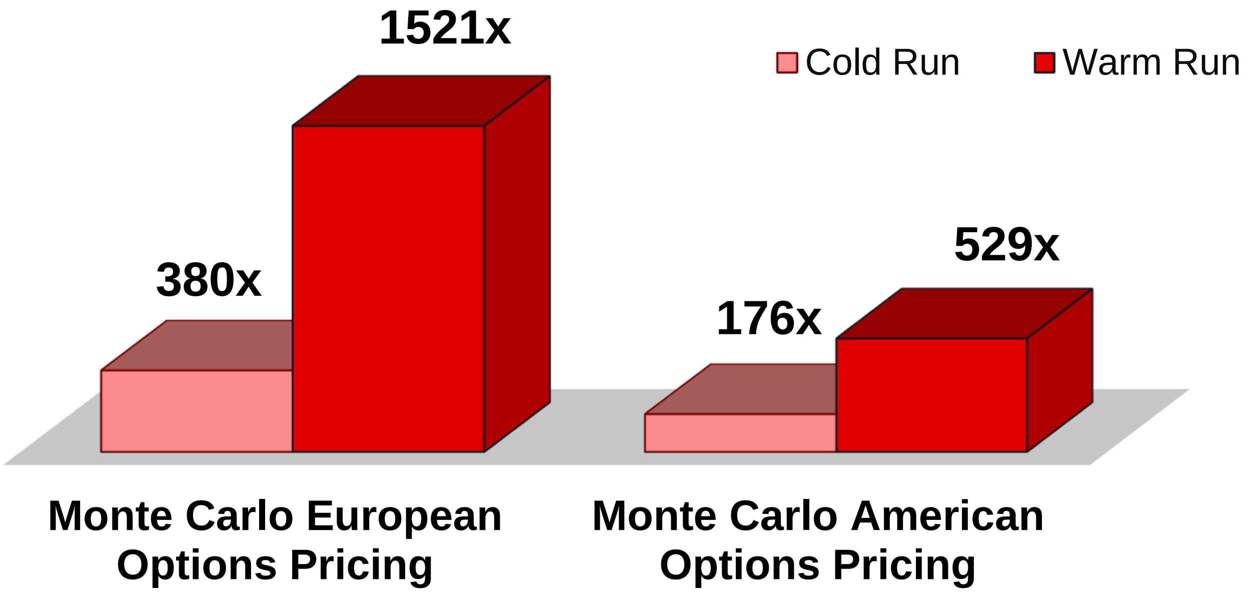
```
cEuropean = MCEuropean()
mcEuropean.claimDevice(chosenDevice)

result = mcEuropean.run(OptionType.Put, 36.0,
40.0, 0.06, 0.0, 0.20, 1.0, 0.02)

print("Option Price = {0}".format(result[1]))

print("This run took ",str(runtime),"microseconds\n")
```

# Quantitative Finance Performance Benchmark

| Monte Carlo European Options Pricing | | |
|---|---|---|
| | Cold Run | Warm Run |
| QuantLib | 20.155 ms | 20.155 ms |
| Vitis Quantitative Finance Library | 0.053 ms | 0.01325 ms |
| Speed-Up | 380X | 1521X |

| Monte Carlo American Options Pricing | | |
|---|---|---|
| | Cold Run | Warm Run |
| QuantLib | 1038.105 ms | 1038.105 ms |
| Vitis Quantitative Finance Library | 5.87 ms | 1.96 ms |
| Speed-Up | 176X | 529X |

## Speed-Up Vs. QuantLib on CPU



**CPU** : 2 Intel(R) Xeon(R) CPU E5-2690 v4 @3.20GHz, 8 cores per processor and 2 threads per core.
**Xilinx :** Vitis Quantitative Finance Library v1.0 running on 1 Alveo U250
**Cold Run**: Pricing Engine starts up in response to a request.
**Warm Run**: Pricing Engine is already running, with sufficient memory allocated to handle the request

# Vitis BLAS Library

*Common Algorithmic Building Blocks*

- **FPGA-accelerated Basic Linear Algebra Subroutines (BLAS)**
  - General Matric Multiply (GEMM) and General Matric-Vector (GEMV) APIs
    - y = a*x + y (all vectors)
    - y = a*A*x + b*y (A is matrix)
    - C = a*A*B + b*C (A, B, C are matrices)

- **GEMM and GEMV APIs can drop-in and replace CPU and GPU-based BLAS**
  - For Rapid Prototyping and Performance Evaluation

- **Use performance-optimized primitives to accelerate custom pipelines**
  - Maximum, Minimum, Euclidean Norm, Dot Products, Scaling and Copying
  - Triangular-matrix & Vector Multiplication, Scalar-Vector Product (axpy)

# Vitis Solver Library

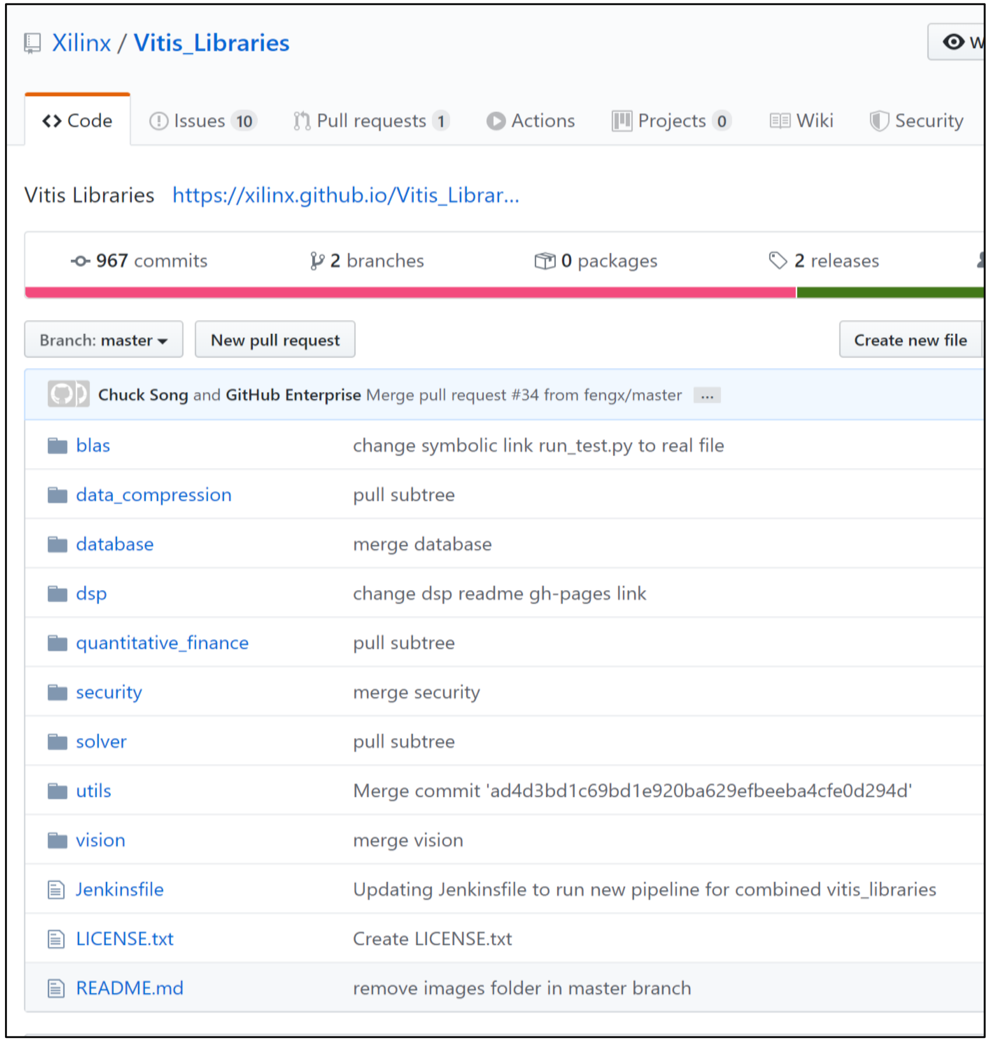
*Common Algorithmic Building Blocks*

- **FPGA-accelerated Matrix Decomposition, Linear and Eigenvalue Solvers**
  - Optimized based on Matrix Algorithms in LAPACK
- **Combine with BLAS library functions to accelerate complex algorithms.**
  - Example: Finite Differences uses tri-diagonal solver and vector-matrix multiplication
- **Performance-optimized Primitives Include**
  - Cholesky Decomposition, LU Decomposition, QR Decomposition & SVD Decomposition
  - Tridiagonal Linear Solvers, Solvers for Triangular, Symmetric and Non-symmetric matrices
  - Matrix Inverse, Jacobian Eigenvalue Solvers

# Explore All Vitis Libraries

https://github.com/Xilinx/Vitis_Libraries

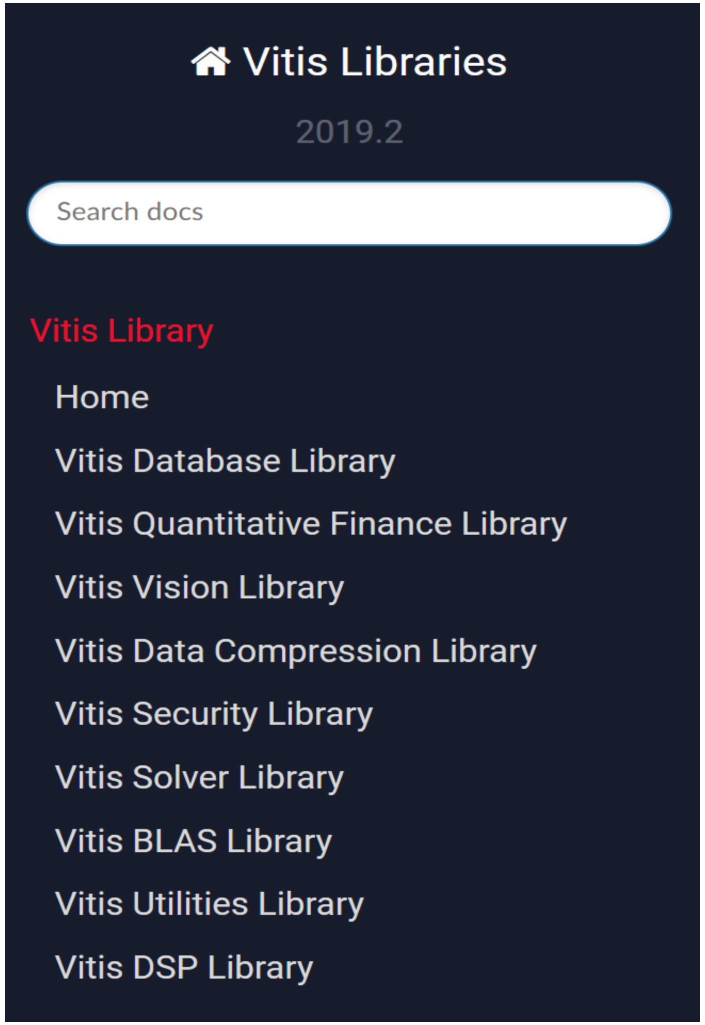
https://xilinx.github.io/Vitis_Libraries/



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

Chuck Song and GitHub Enterprise Merge pull request #34 from fengx/master ...

| | |
|---|---|
| blas | change symbolic link run_test.py to real file |
| data_compression | pull subtree |
| database | merge database |
| dsp | change dsp readme gh-pages link |
| quantitative_finance | pull subtree |
| security | merge security |
| solver | pull subtree |
| utils | Merge commit 'ad4d3bd1c69bd1e920ba629efbeeba4cfe0d294d' |
| vision | merge vision |
| Jenkinsfile | Updating Jenkinsfile to run new pipeline for combined vitis_libraries |
| LICENSE.txt | Create LICENSE.txt |
| README.md | remove images folder in master branch |



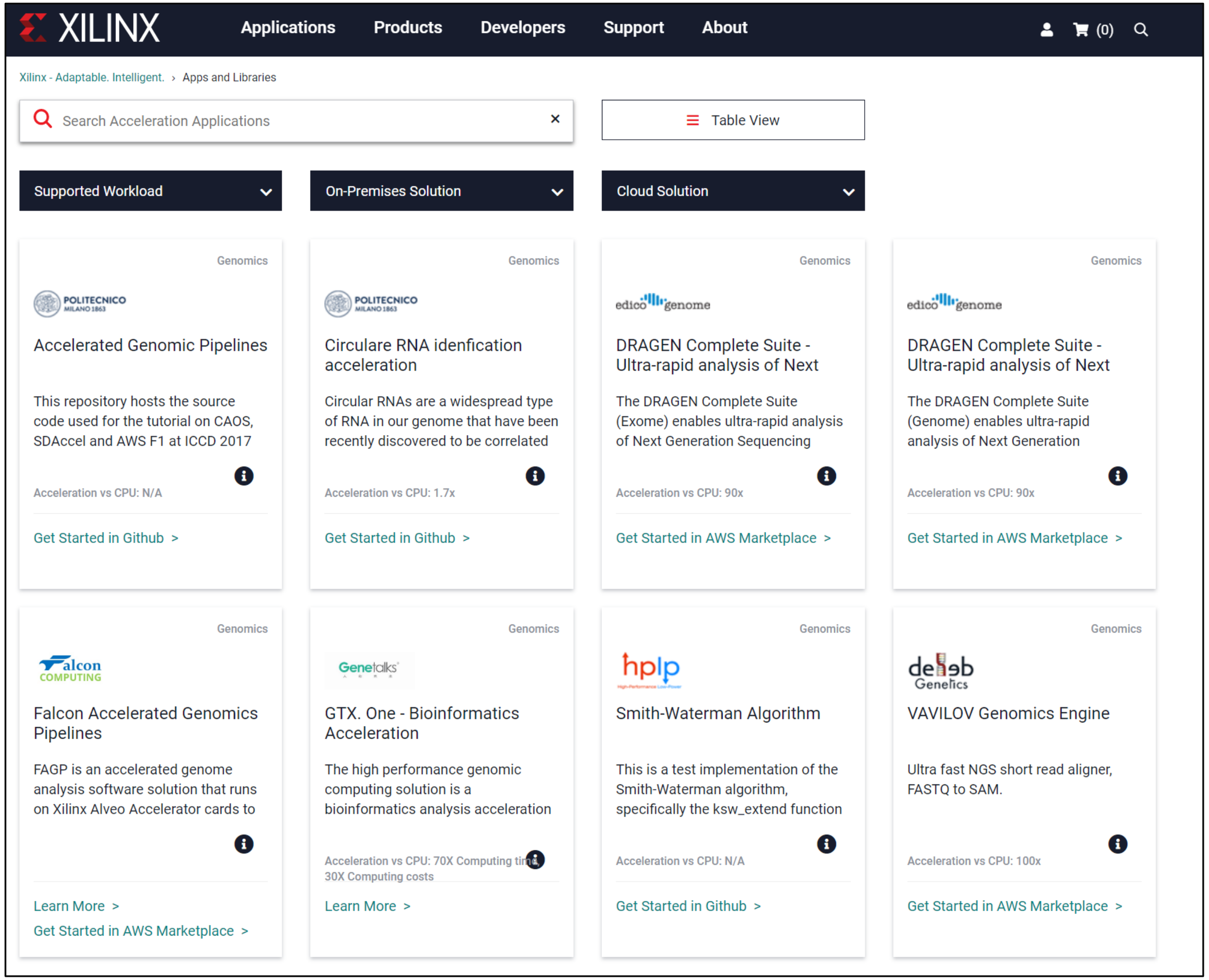
Vitis Library

- Home
- Vitis Database Library
- Vitis Quantitative Finance Library
- Vitis Vision Library
- Vitis Data Compression Library
- Vitis Security Library
- Vitis Solver Library
- Vitis BLAS Library
- Vitis Utilities Library
- Vitis DSP Library

# Let’s Dive into a Demo Walkthrough

# Explore Partner Applications and Libraries

https://www.xilinx.com/products/apps-and-libraries.html



Genomics
POLITECNICO MILANO 1863
Accelerated Genomic Pipelines
This repository hosts the source code used for the tutorial on CAOS, SDAccel and AWS F1 at ICCD 2017
Acceleration vs CPU: N/A
Get Started in Github >

Genomics
POLITECNICO MILANO 1863
Circulare RNA idenfication acceleration
Circular RNAs are a widespread type of RNA in our genome that have been recently discovered to be correlated
Acceleration vs CPU: 1.7x
Get Started in Github >

Genomics
edico genome
DRAGEN Complete Suite - Ultra-rapid analysis of Next
The DRAGEN Complete Suite (Exome) enables ultra-rapid analysis of Next Generation Sequencing
Acceleration vs CPU: 90x
Get Started in AWS Marketplace >

Genomics
edico genome
DRAGEN Complete Suite - Ultra-rapid analysis of Next
The DRAGEN Complete Suite (Genome) enables ultra-rapid analysis of Next Generation
Acceleration vs CPU: 90x
Get Started in AWS Marketplace >

Genomics
Falcon COMPUTING
Falcon Accelerated Genomics Pipelines
FAGP is an accelerated genome analysis software solution that runs on Xilinx Alveo Accelerator cards to
Learn More >
Get Started in AWS Marketplace >

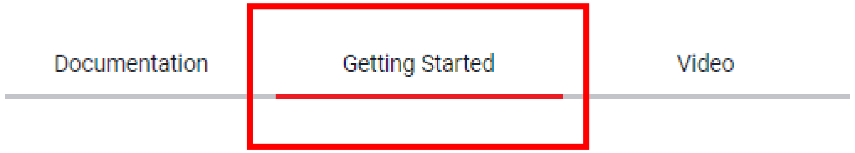
Genomics
Genetalks
GTX. One - Bioinformatics Acceleration
The high performance genomic computing solution is a bioinformatics analysis acceleration
Acceleration vs CPU: 70X Computing ti
30X Computing costs
Learn More >

Genomics
hplp
Smith-Waterman Algorithm
This is a test implementation of the Smith-Waterman algorithm, specifically the ksw_extend function
Acceleration vs CPU: N/A
Get Started in Github >

Genomics
delieb Genetics
VAVILOV Genomics Engine
Ultra fast NGS short read aligner, FASTQ to SAM.
Acceleration vs CPU: 100x
Get Started in AWS Marketplace >

# Start Developing With Vitis Accelerated Libraries

- Download Vitis Unified Software Platform – Free !



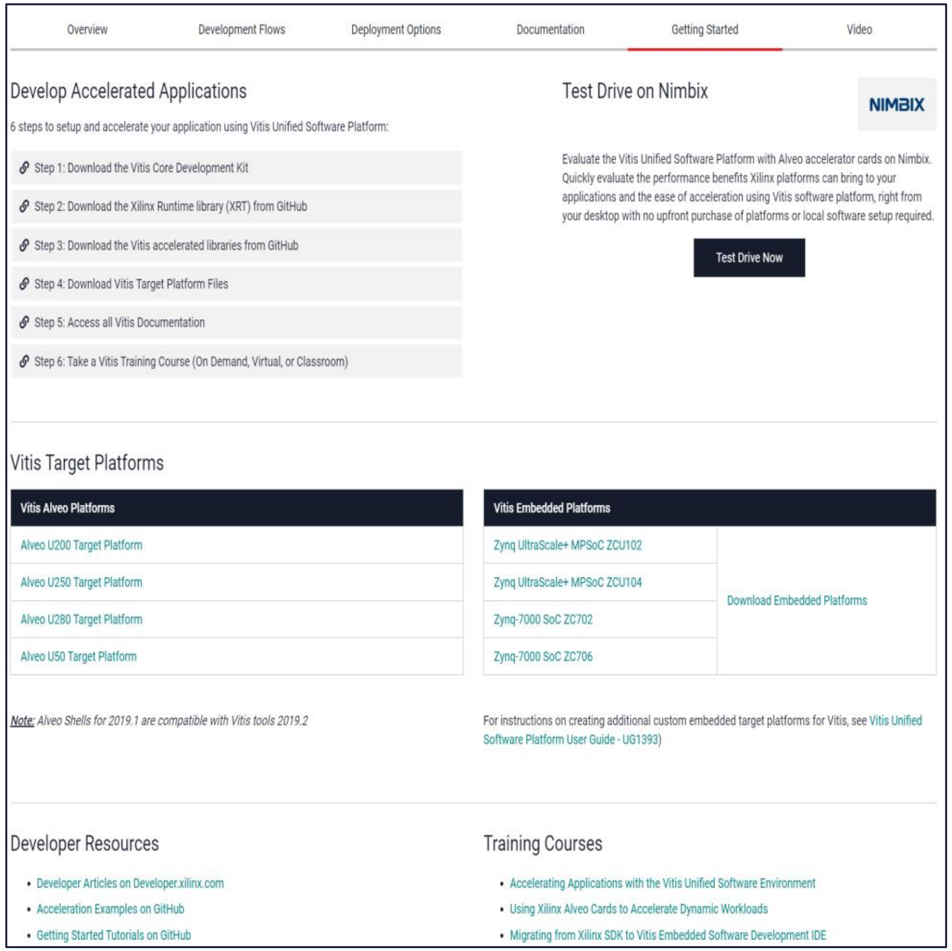
XILINX VITIS™

[www.xilinx.com/vitis](www.xilinx.com/vitis)

- Download Vitis Libraries from GitHub

GitHub

[https://github.com/Xilinx/Vitis_Libraries](https://github.com/Xilinx/Vitis_Libraries)



Develop Accelerated Applications

6 steps to setup and accelerate your application using Vitis Unified Software Platform:

- Step 1: Download the Vitis Core Development Kit
- Step 2: Download the Xilinx Runtime library (XRT) from GitHub
- Step 3: Download the Vitis accelerated libraries from GitHub
- Step 4: Download Vitis Target Platform Files
- Step 5: Access all Vitis Documentation
- Step 6: Take a Vitis Training Course (On Demand, Virtual, or Classroom)

Test Drive on Nimbix

Evaluate the Vitis Unified Software Platform with Alveo accelerator cards on Nimbix. Quickly evaluate the performance benefits Xilinx platforms can bring to your applications and the ease of acceleration using Vitis software platform, right from your desktop with no upfront purchase of platforms or local software setup required.

Vitis Target Platforms

| Vitis Alveo Platforms |
|---|
| Alveo U200 Target Platform |
| Alveo U250 Target Platform |
| Alveo U280 Target Platform |
| Alveo U50 Target Platform |

| Vitis Embedded Platforms | |
|---|---|
| Zynq UltraScale+ MPSoC ZCU102 | Download Embedded Platforms |
| Zynq UltraScale+ MPSoC ZCU104 | |
| Zynq-7000 SoC ZC702 | |
| Zynq-7000 SoC ZC706 | |

*Note: Alveo Shells for 2019.1 are compatible with Vitis tools 2019.2*

For instructions on creating additional custom embedded target platforms for Vitis, see Vitis Unified Software Platform User Guide - UG1393)

Developer Resources

- Developer Articles on Developer.xilinx.com
- Acceleration Examples on GitHub
- Getting Started Tutorials on GitHub

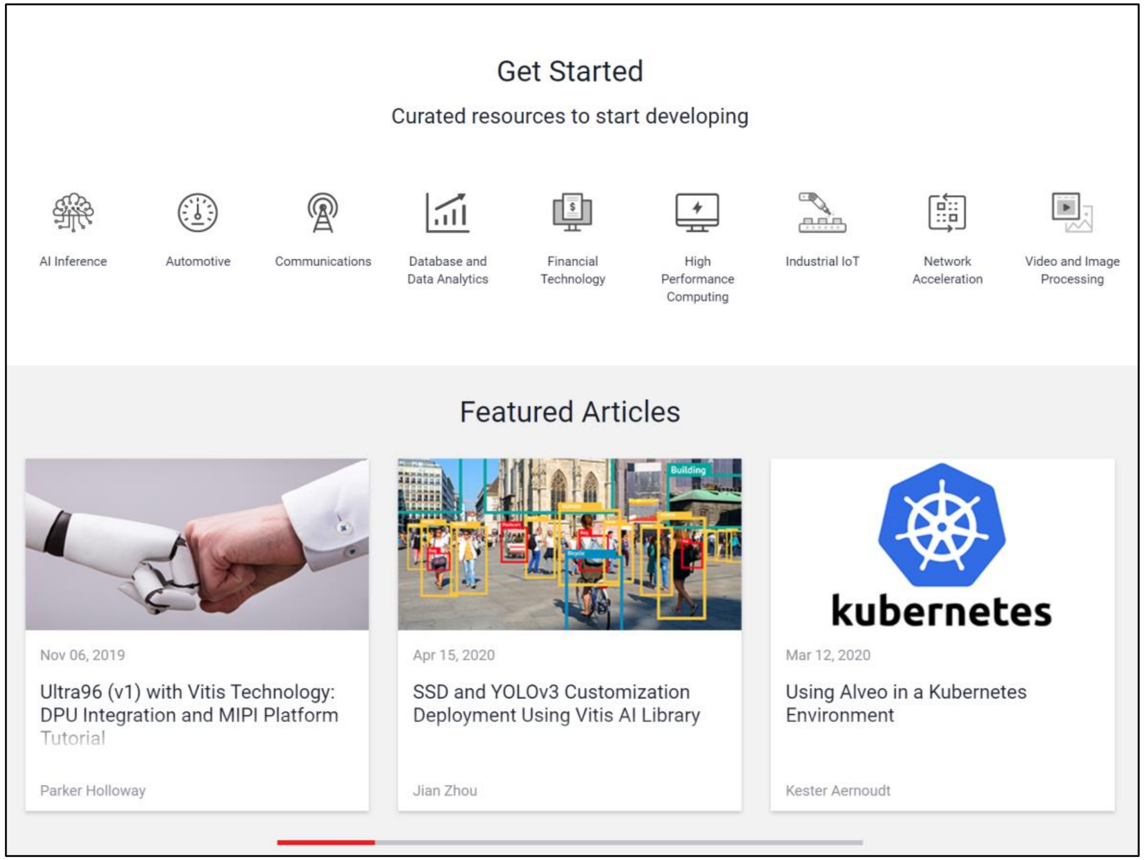
Training Courses

- Accelerating Applications with the Vitis Unified Software Environment
- Using Xilinx Alveo Cards to Accelerate Dynamic Workloads
- Migrating from Xilinx SDK to Vitis Embedded Software Development IDE

# Developer Articles & Resources

- Get Started with “How-To” Developer Articles & Tutorials

[https://developer.xilinx.com/](https://developer.xilinx.com/)



Get Started

Curated resources to start developing

Featured Articles

Nov 06, 2019
Ultra96 (v1) with Vitis Technology: DPU Integration and MIPI Platform Tutorial
Parker Holloway

Apr 15, 2020
SSD and YOLOv3 Customization Deployment Using Vitis AI Library
Jian Zhou

Mar 12, 2020
Using Alveo in a Kubernetes Environment
Kester Aernoudt

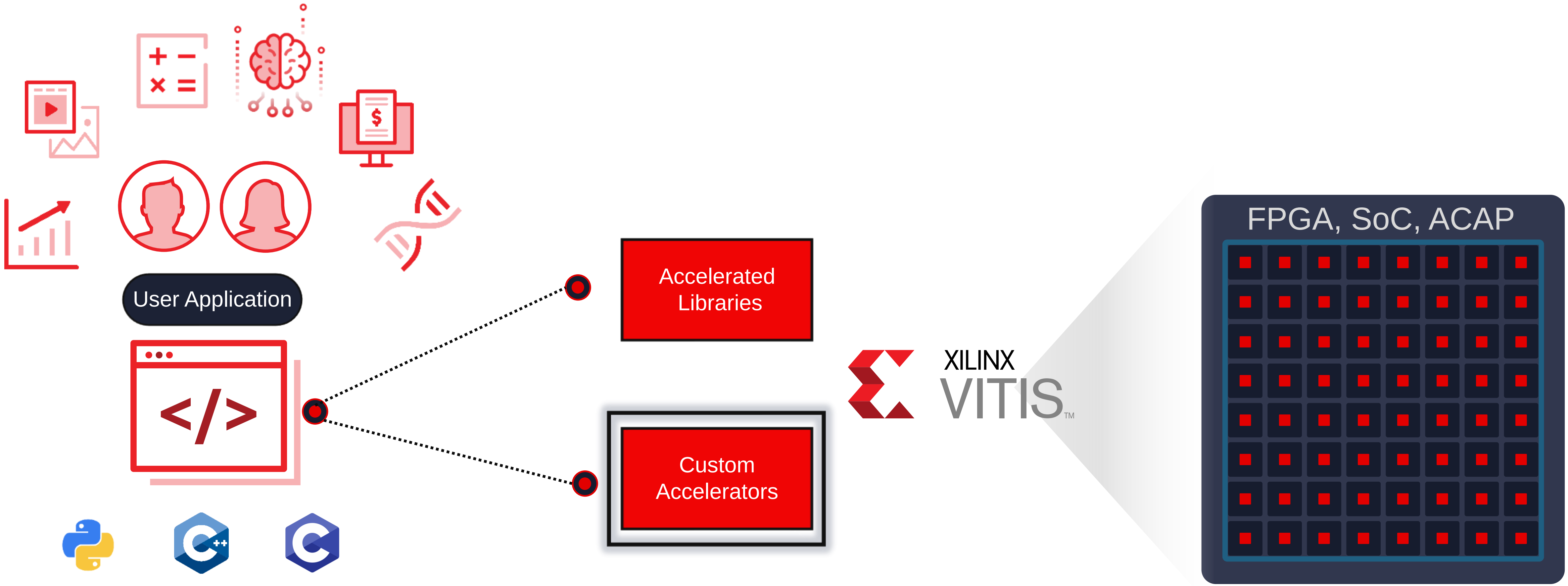
- Getting Started Tutorials

GitHub

[https://github.com/Xilinx/Vitis-Tutorials](https://github.com/Xilinx/Vitis-Tutorials)

# Design Methodology for Developing Custom Accelerators

*Tune in Next Thursday, April 30*



# Q&A : Type In Your Questions

XILINX®

# Thank You