Generative AI and AIoT (GenAIoT) Coding Skills Education for Gifted Students (2024)

P4L5.1– Quantized Neural Network (QNN) & Model Compilation for Inference on FPGA

Department of Electrical Engineering

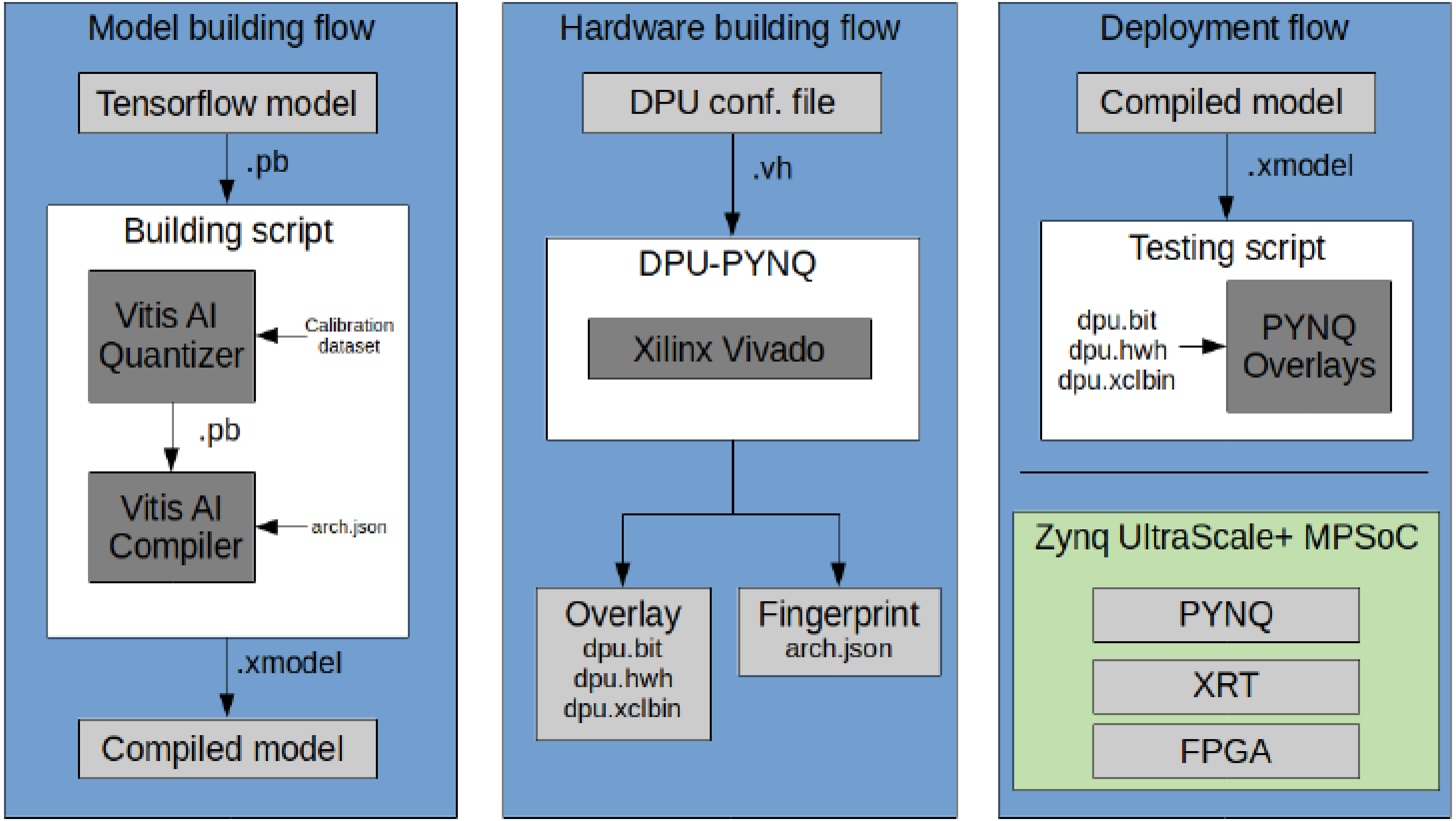
香港城市大學
City University of Hong Kong

Table of Contents

- FPGA inference overview
- Deep learning processing unit (DPU)
- Vitis-AI overview
- Model training, quantization, and compilation for FPGA inference
- PYNQ-DPU
- Lab session:
    - ➢ Inference with PYNQ-DPU on Ultra96-V2 board

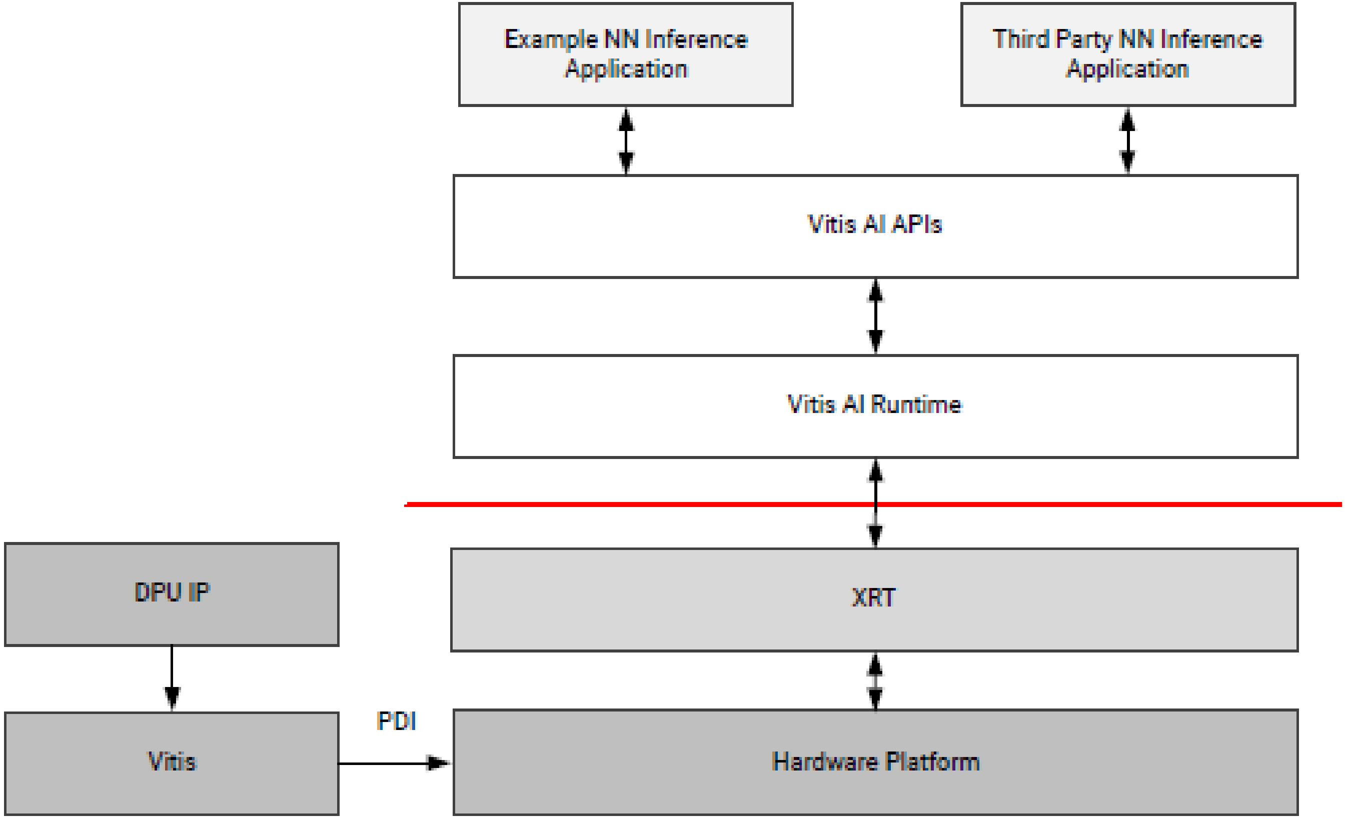
Overview of AI Inference on Xilinx FPGA

- Involves three steps (including hardware and software developments)



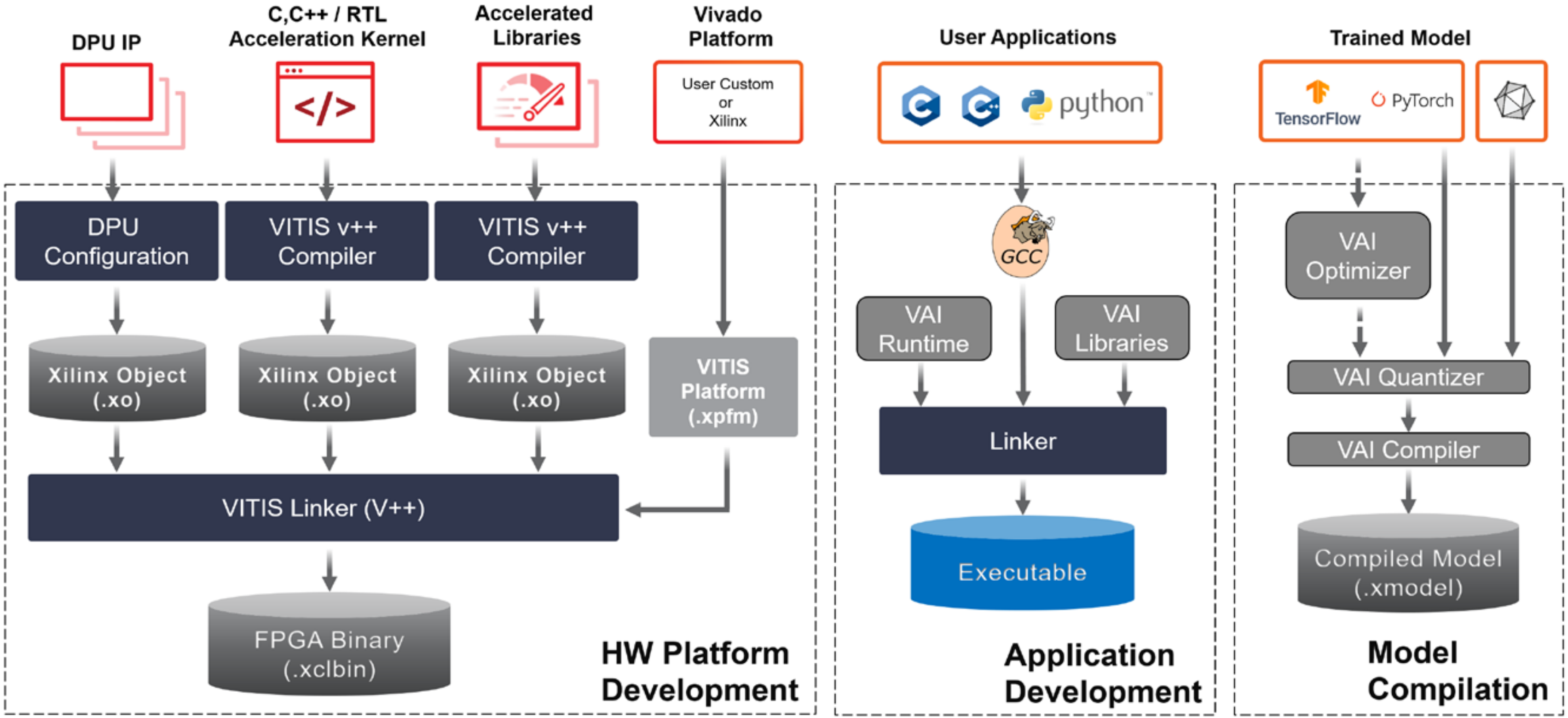
Overview of AI Inference on Xilinx FPGA

- Software is developed on Vitis AI while the hardware on Vitis/Vivado


Example NN Inference Application
Third Party NN Inference Application
Vitis AI APIs
Vitis AI Runtime
DPU IP
XRT
Vitis
PDI
Hardware Platform

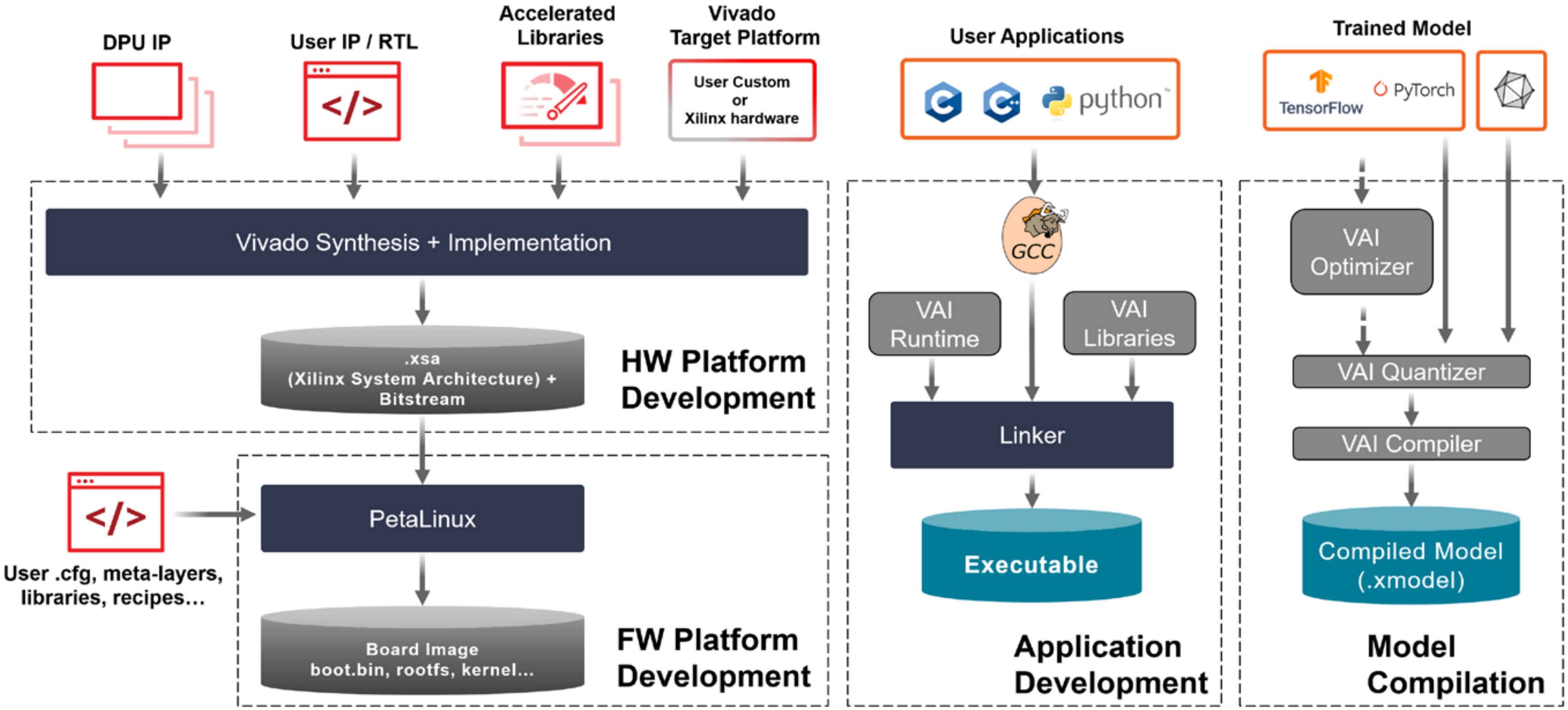
Vitis Flow FPGA/DPU AI Development

- Hardware files generated in Vitis (dpu.xclbin, dpu.hwh and dpu.bit files)



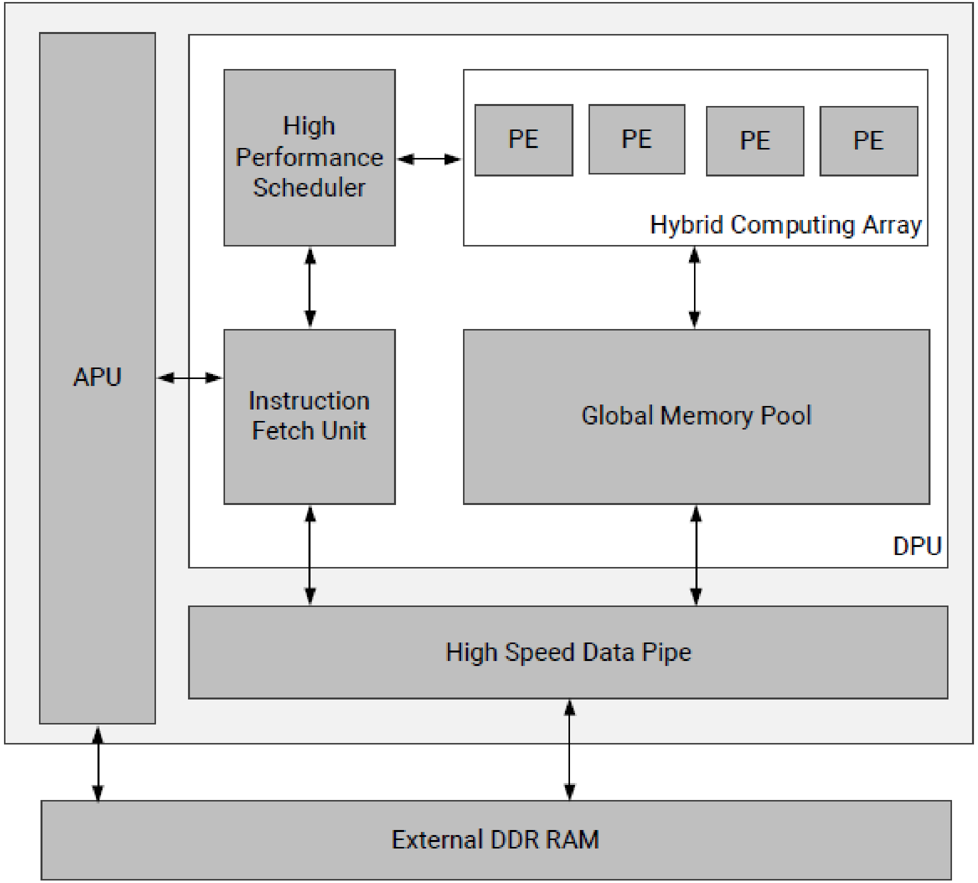
Vivado Flow FPGA/DPU AI Development

- Hardware and OS files generated in Vivado and Petalinux (.xsa, boot.bin, rootfs and kernel files)



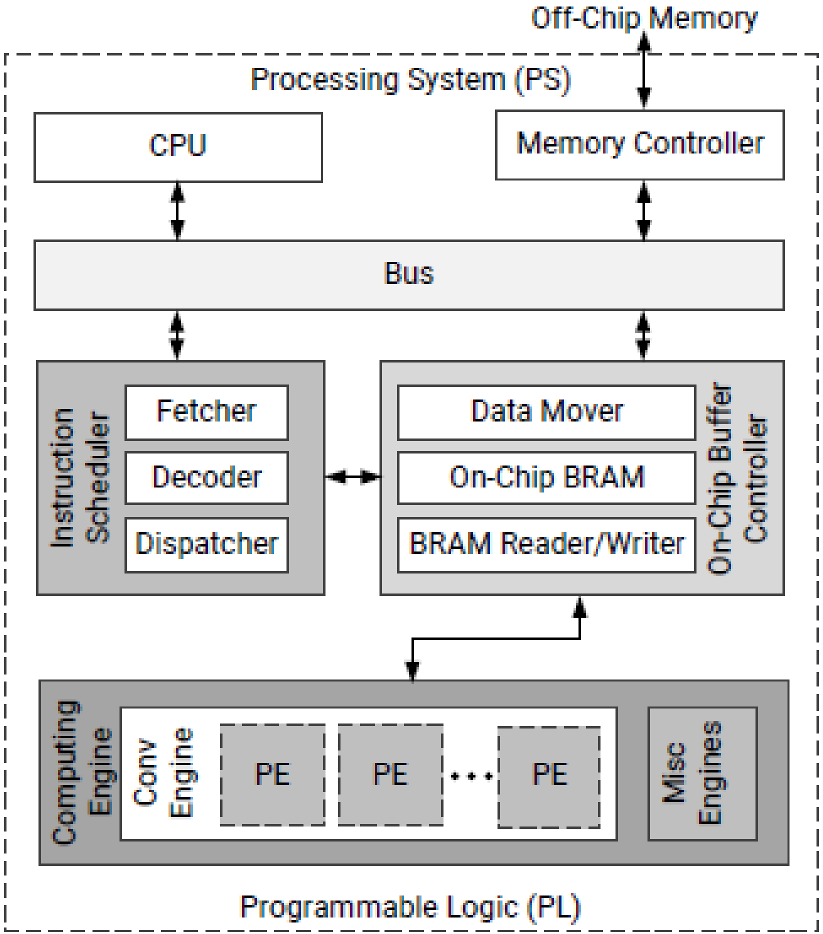
Deep learning processing unit (DPU)

- Accelerates AI applications on FPGA
- Has several Processing Elements (PEs) running in parallel (like GPU cores)
- Basically performs MAC operations
- Comes as a soft-core
- Implemented in Vitis or Vivado Flows

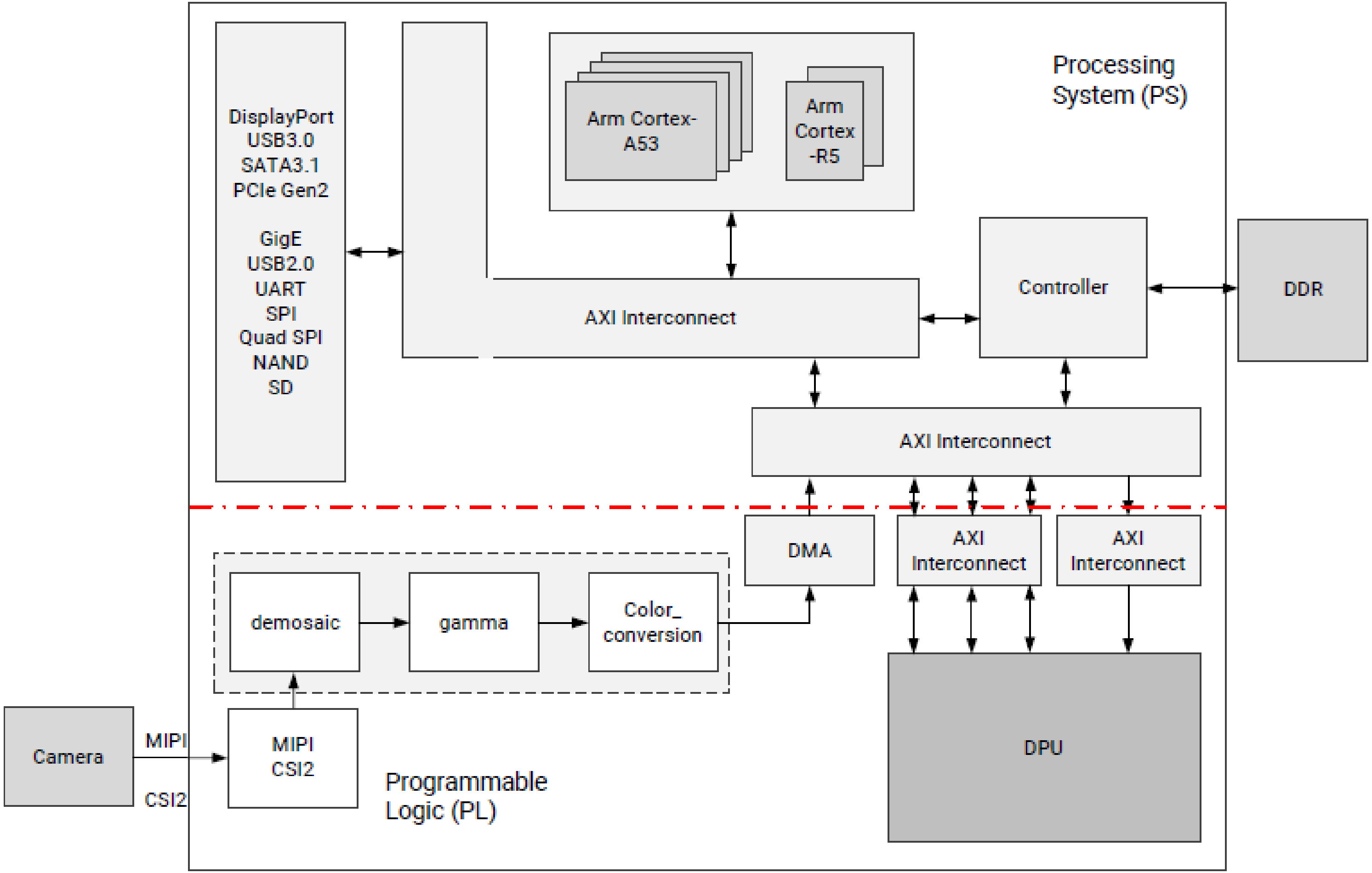


DPUCZDX8G Hardware Architecture

- DPUCZDX8G is the DPU for the Zynq® UltraScale+™ MPSoC.
- It is configurable and optimized for CNN (B512, B800, B1024, B1152, B1600, B2304, B3136)
- A DPU version must be compatible with the Vitis-AI version that supports your chosen model.

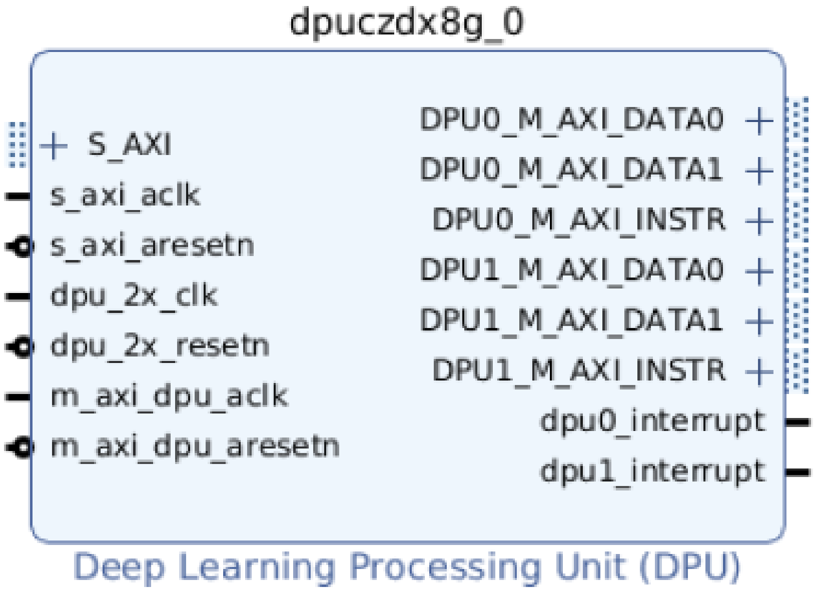


Example System Integration with DPUCZDX8G



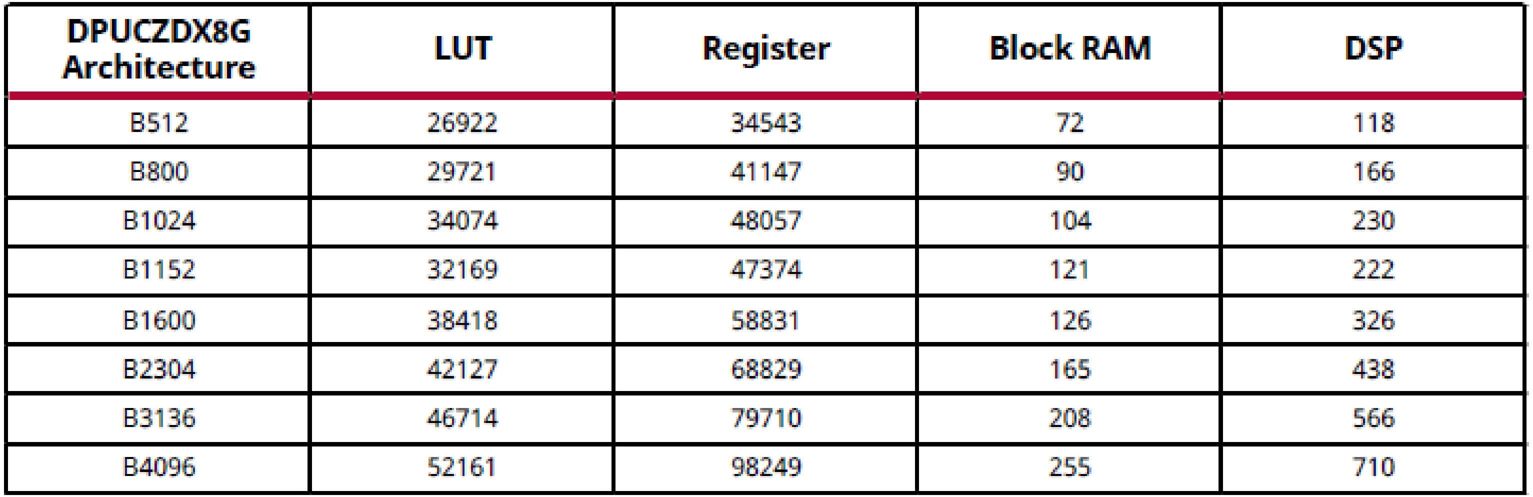
DPUCZDX8G DPU IP

- DPU IP is provided by Xilinx from Vitis-AI GitHub
  - ✓ https://github.com/Xilinx/Vitis-AI/tree/v3.0/dpu
  - ✓ https://www.xilinx.com/bin/public/openDownload?filename=DPUCZDX8G_VAI_v3.0.tar.gz
- Different DPU architectures have different resource consumption and performances
- Can use many cores of the DPU depending on the FPGA capacity.

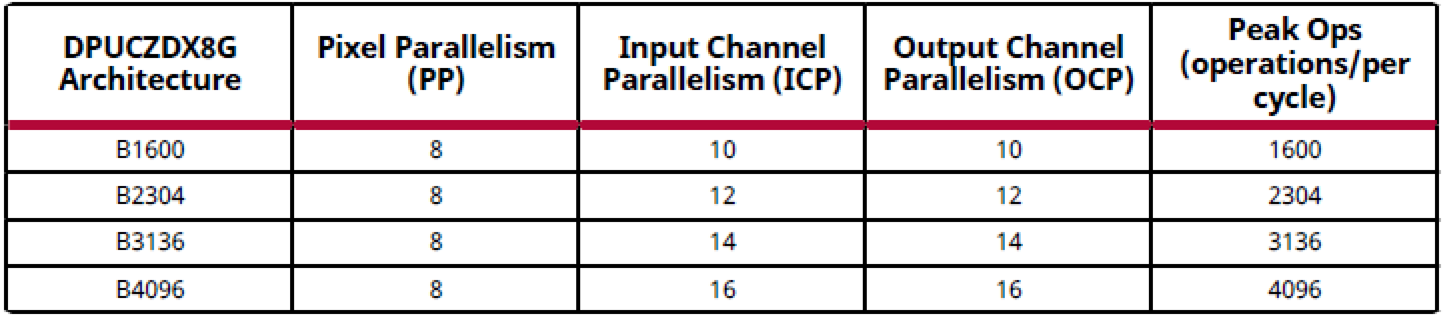


DPUCZDX8G DPU Hardware Resources Consumption

- Different architectures consume different resources and have different performances

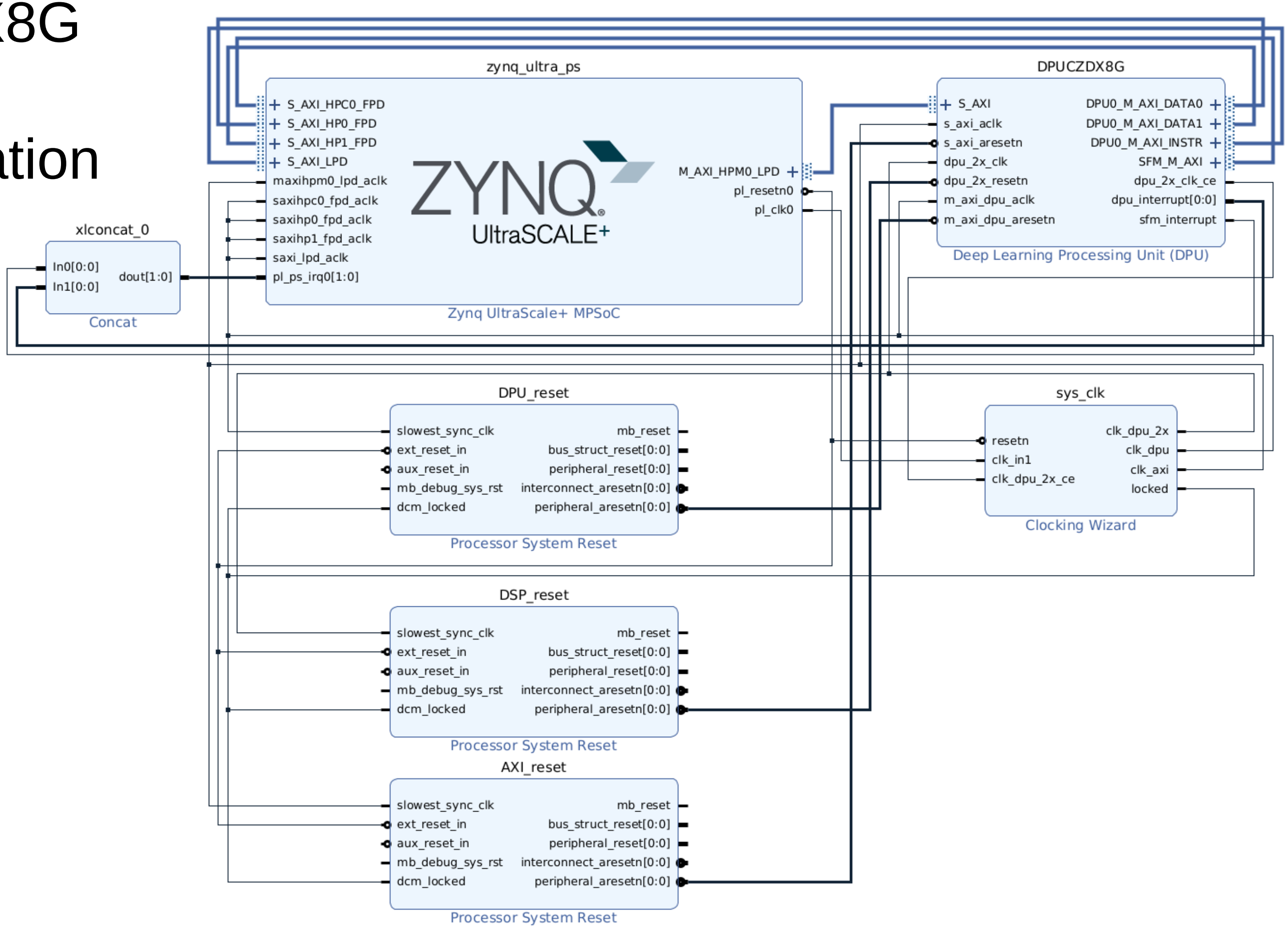


DPUCZDX8G Architecture	LUT	Register	Block RAM	DSP
B512	26922	34543	72	118
B800	29721	41147	90	166
B1024	34074	48057	104	230
B1152	32169	47374	121	222
B1600	38418	58831	126	326
B2304	42127	68829	165	438
B3136	46714	79710	208	566
B4096	52161	98249	255	710

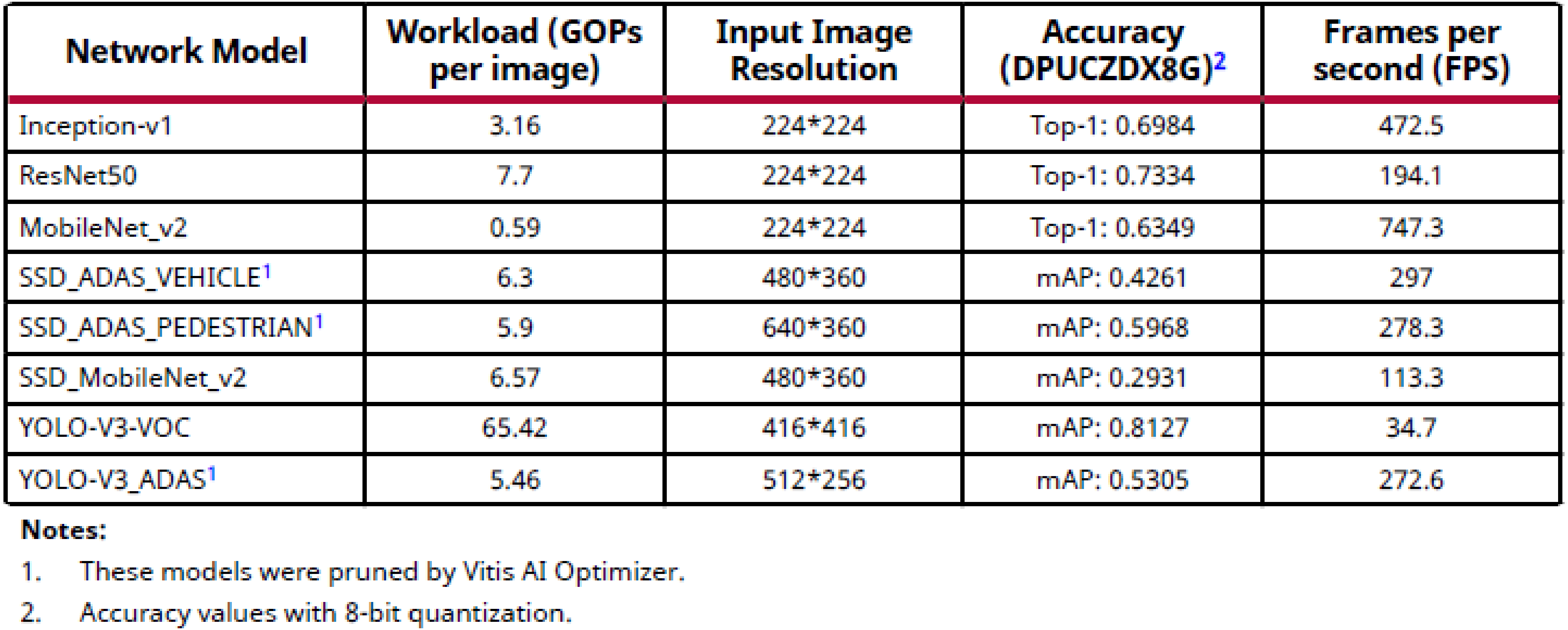


DPUCZDX8G Architecture	Pixel Parallelism (PP)	Input Channel Parallelism (ICP)	Output Channel Parallelism (OCP)	Peak Ops (operations/per cycle)
B1600	8	10	10	1600
B2304	8	12	12	2304
B3136	8	14	14	3136
B4096	8	16	16	4096

DPUCZDX8G Vivado Implementation



Performance of Different Models



Network Model	Workload (GOPs per image)	Input Image Resolution	Accuracy (DPUCZDX8G)[2]	Frames per second (FPS)
Inception-v1	3.16	224*224	Top-1: 0.6984	472.5
ResNet50	7.7	224*224	Top-1: 0.7334	194.1
MobileNet_v2	0.59	224*224	Top-1: 0.6349	747.3
SSD_ADAS_VEHICLE[1]	6.3	480*360	mAP: 0.4261	297
SSD_ADAS_PEDESTRIAN[1]	5.9	640*360	mAP: 0.5968	278.3
SSD_MobileNet_v2	6.57	480*360	mAP: 0.2931	113.3
YOLO-V3-VOC	65.42	416*416	mAP: 0.8127	34.7
YOLO-V3_ADAS[1]	5.46	512*256	mAP: 0.5305	272.6

Notes:

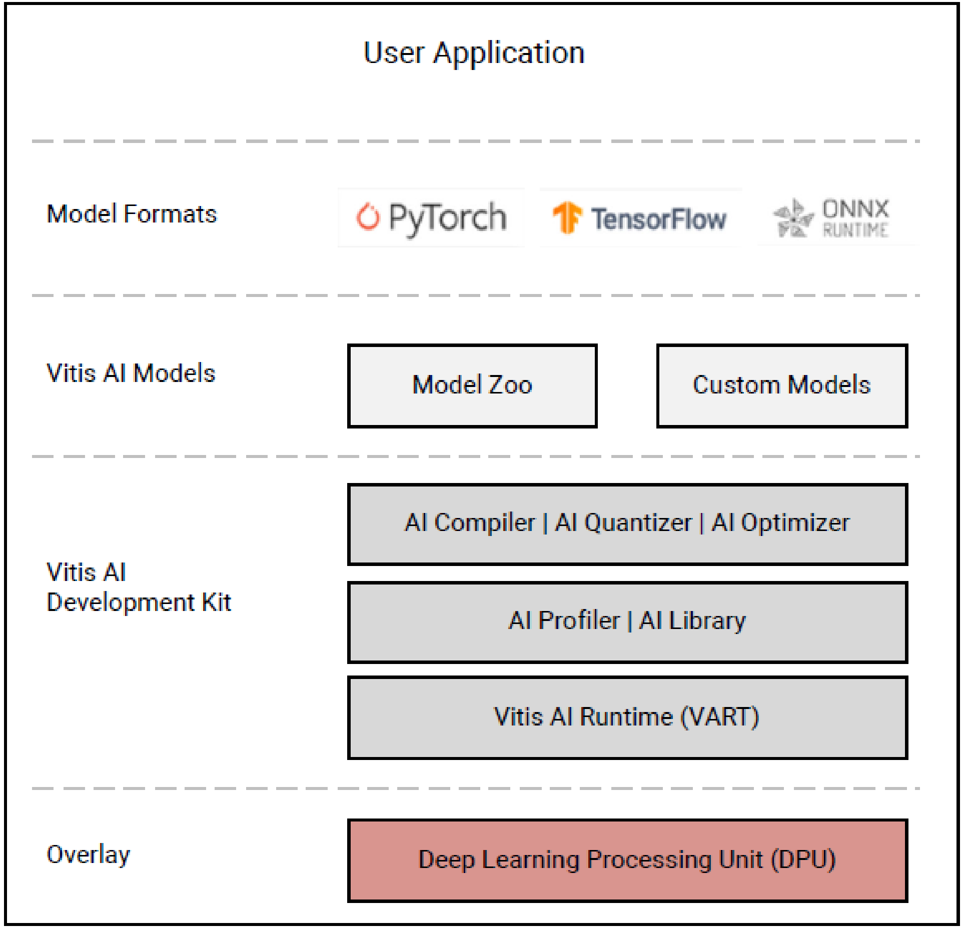
1. These models were pruned by Vitis AI Optimizer.
2. Accuracy values with 8-bit quantization.

Vitis-AI Development Environment

- Set of tools and libraries given by Xilinx for AI deployment on the Xilinx FPGAs.
- Distributed via Docker or Linux sources
- Includes machine learning frameworks, Vitis AI models, development modules and DPU overlays.

https://github.com/Xilinx/Vitis-AI/tree/v3.5/

https://xilinx.github.io/Vitis-AI/3.5/html/docs/install/install.html



User Application

Model Formats: PyTorch, TensorFlow, ONNX RUNTIME

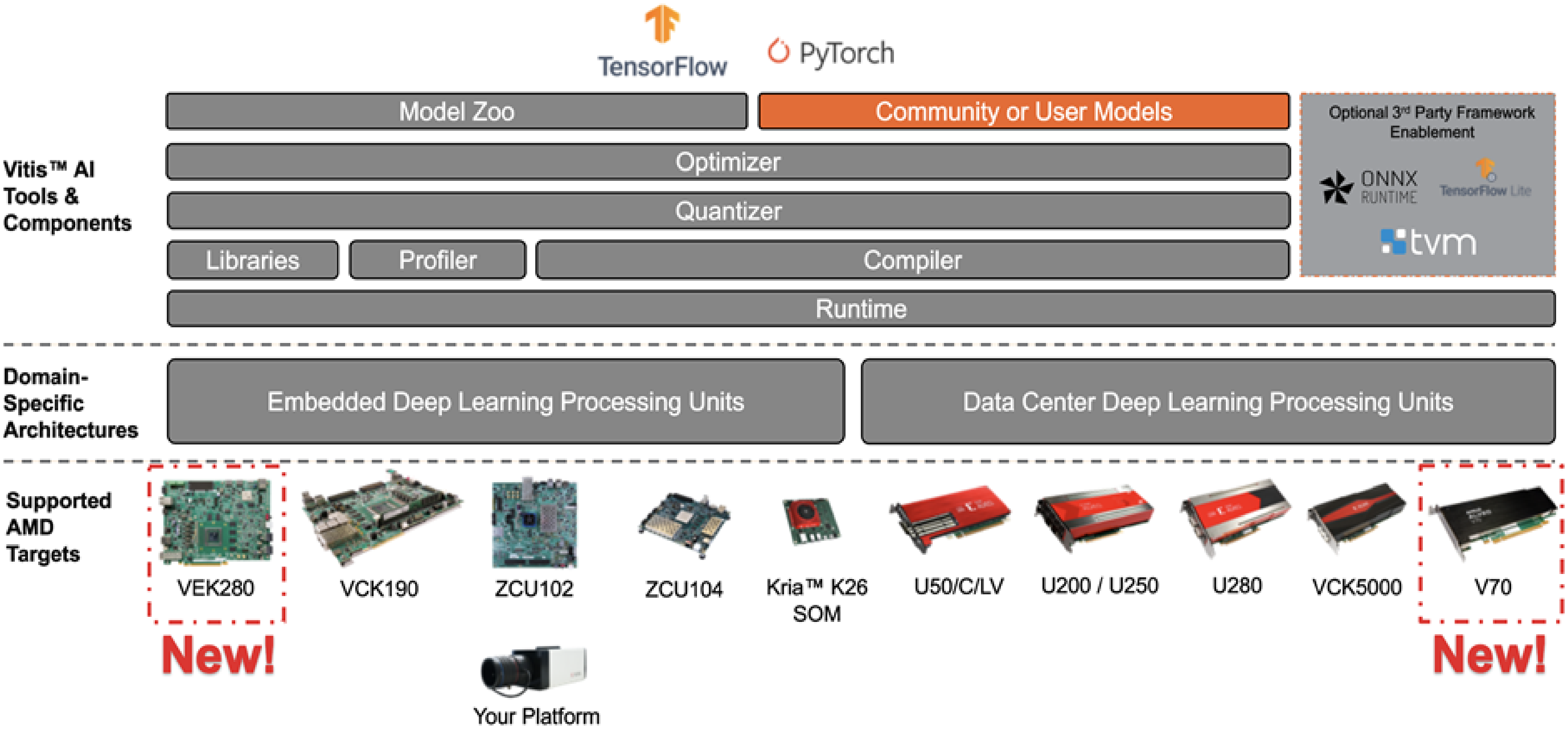
Vitis AI Models: Model Zoo, Custom Models

Vitis AI Development Kit:
- AI Compiler | AI Quantizer | AI Optimizer
- AI Profiler | AI Library
- Vitis AI Runtime (VART)

Overlay: Deep Learning Processing Unit (DPU)

AMD Vitis™ AI Integrated Development Environment

A Complete AI Stack for Adaptable AMD Targets



TensorFlow

PyTorch

Vitis™ AI Tools & Components

Model Zoo

Community or User Models

Optimizer

Quantizer

Libraries

Profiler

Compiler

Runtime

Optional 3rd Party Framework Enablement

ONNX RUNTIME

TensorFlow Lite

tvm

Domain-Specific Architectures

Embedded Deep Learning Processing Units

Data Center Deep Learning Processing Units

Supported AMD Targets

VEK280

VCK190

ZCU102

ZCU104

Kria™ K26 SOM

U50/C/LV

U200 / U250

U280

VCK5000

V70

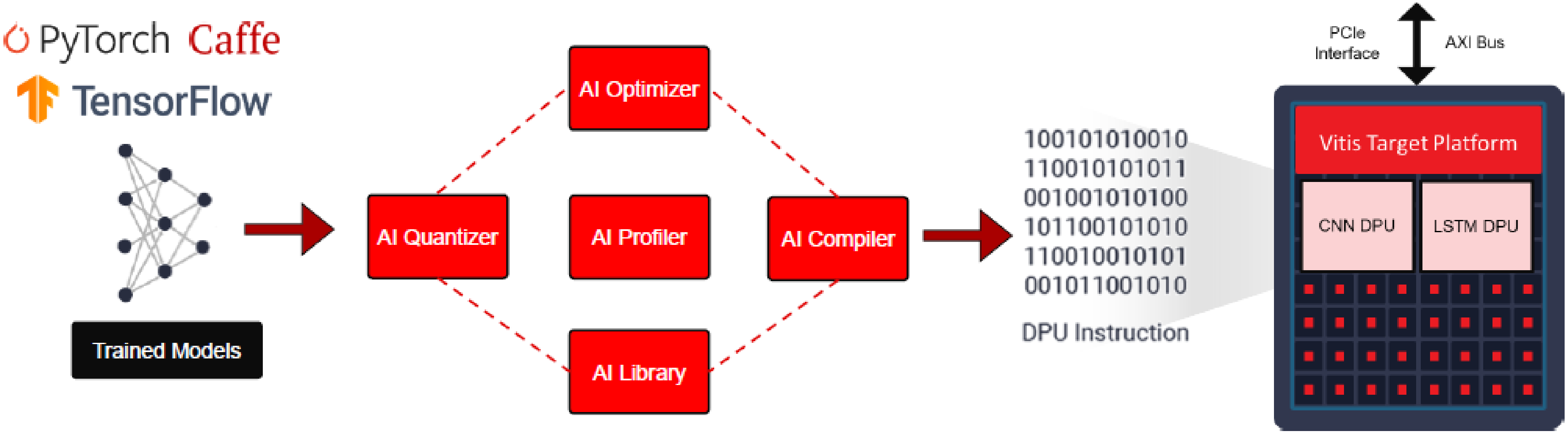
New!

New!

Your Platform

https://docs.amd.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview

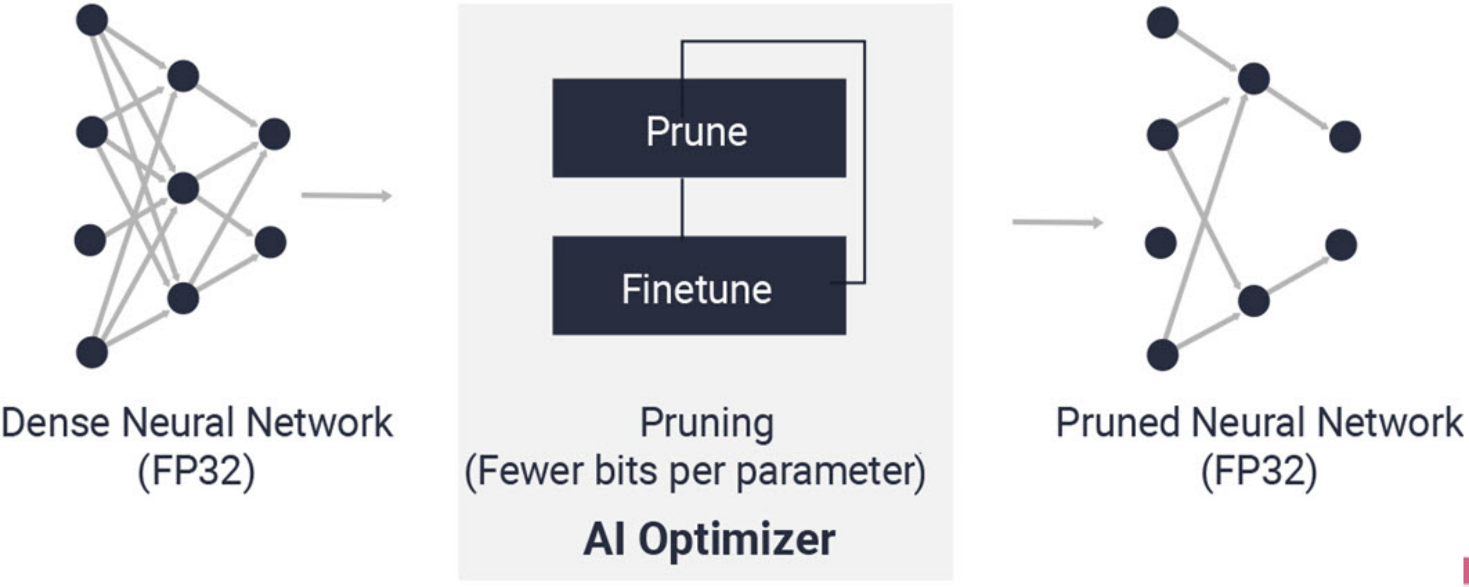
Vitis-AI Software (Model) Development



Vitis AI Development Kit

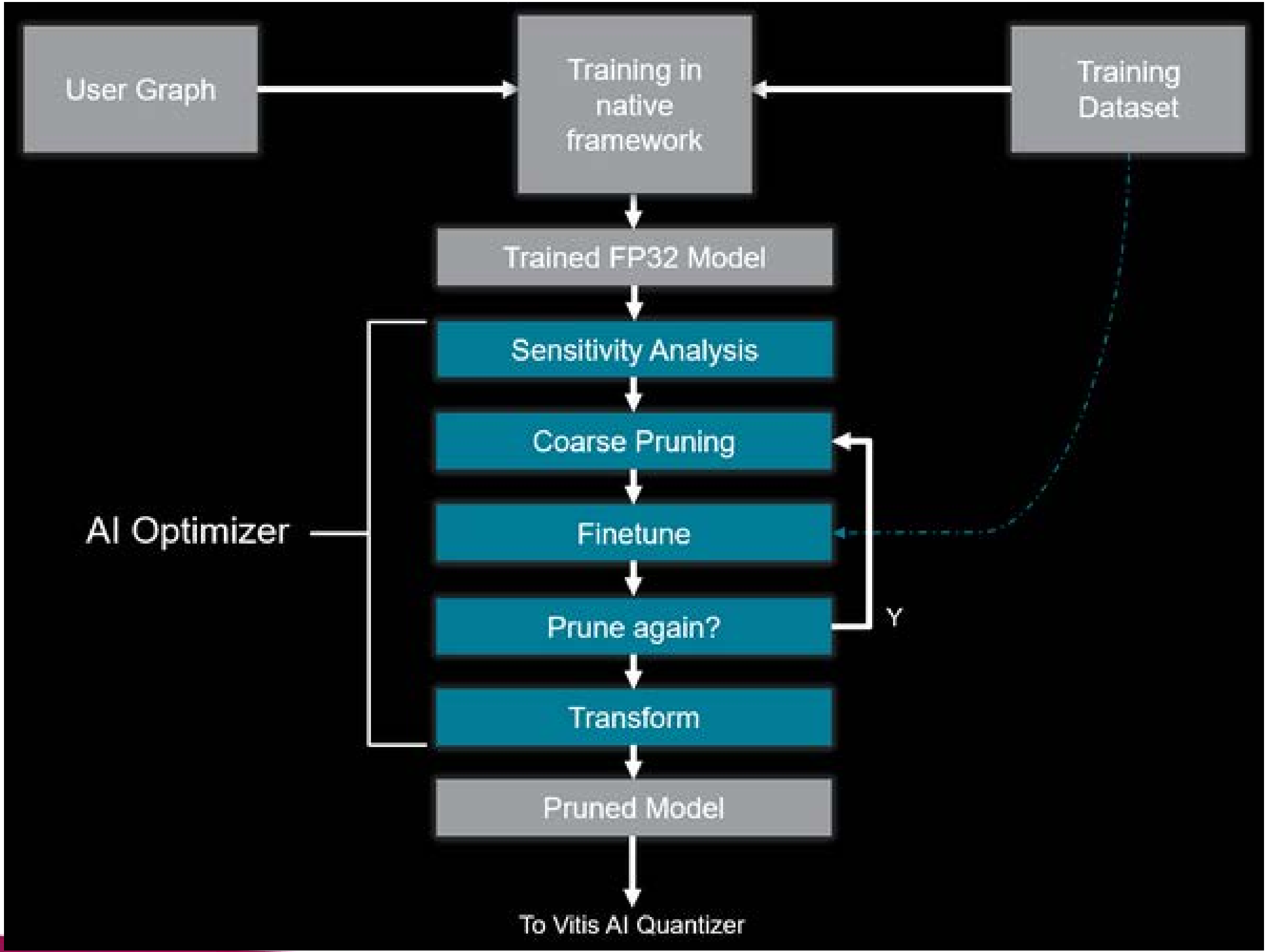
Vitis-AI Optimizer

With world-leading model compression technology, you can achieve an impressive reduction in model complexity, ranging from 5x to 50x, while experiencing minimal accuracy degradation.



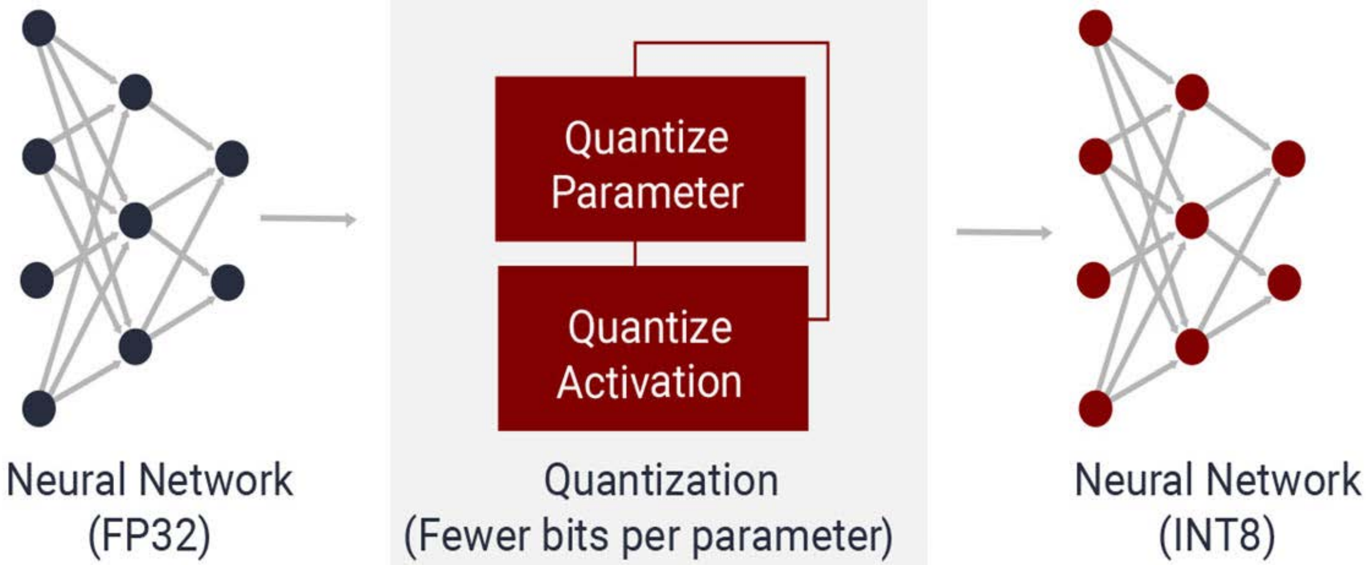
Vitis-AI Optimizer

- Optimization is optional

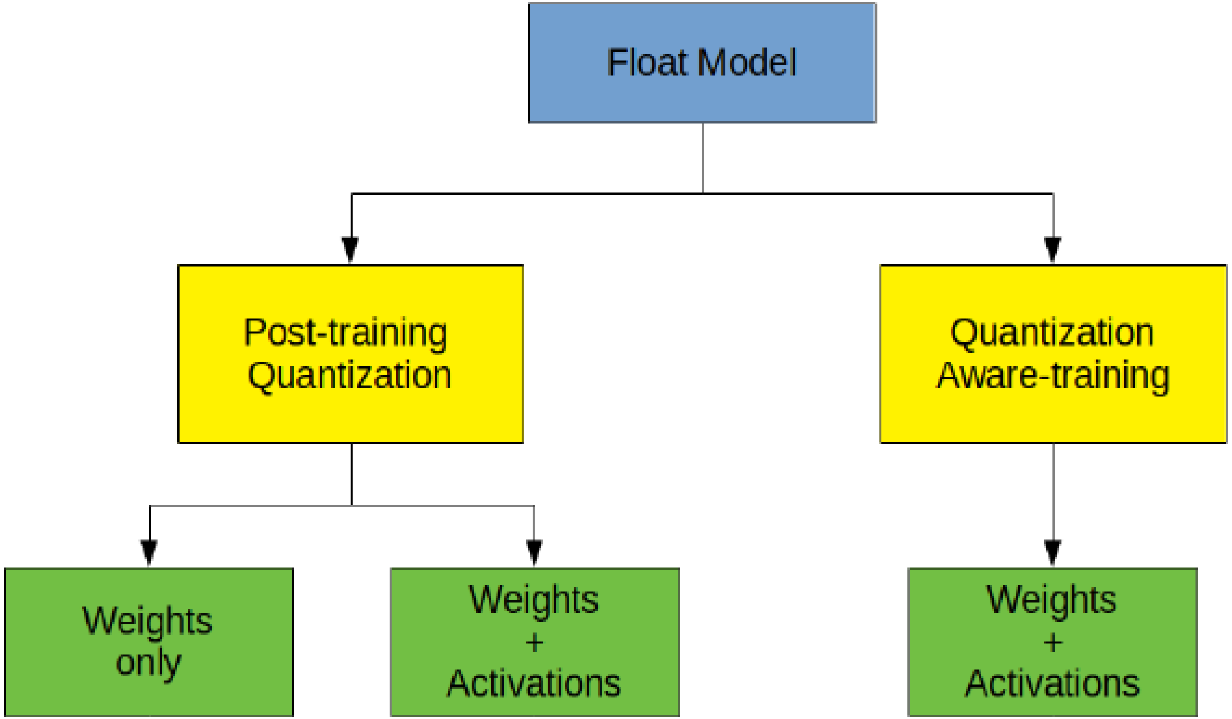


Vitis-AI Quantizer

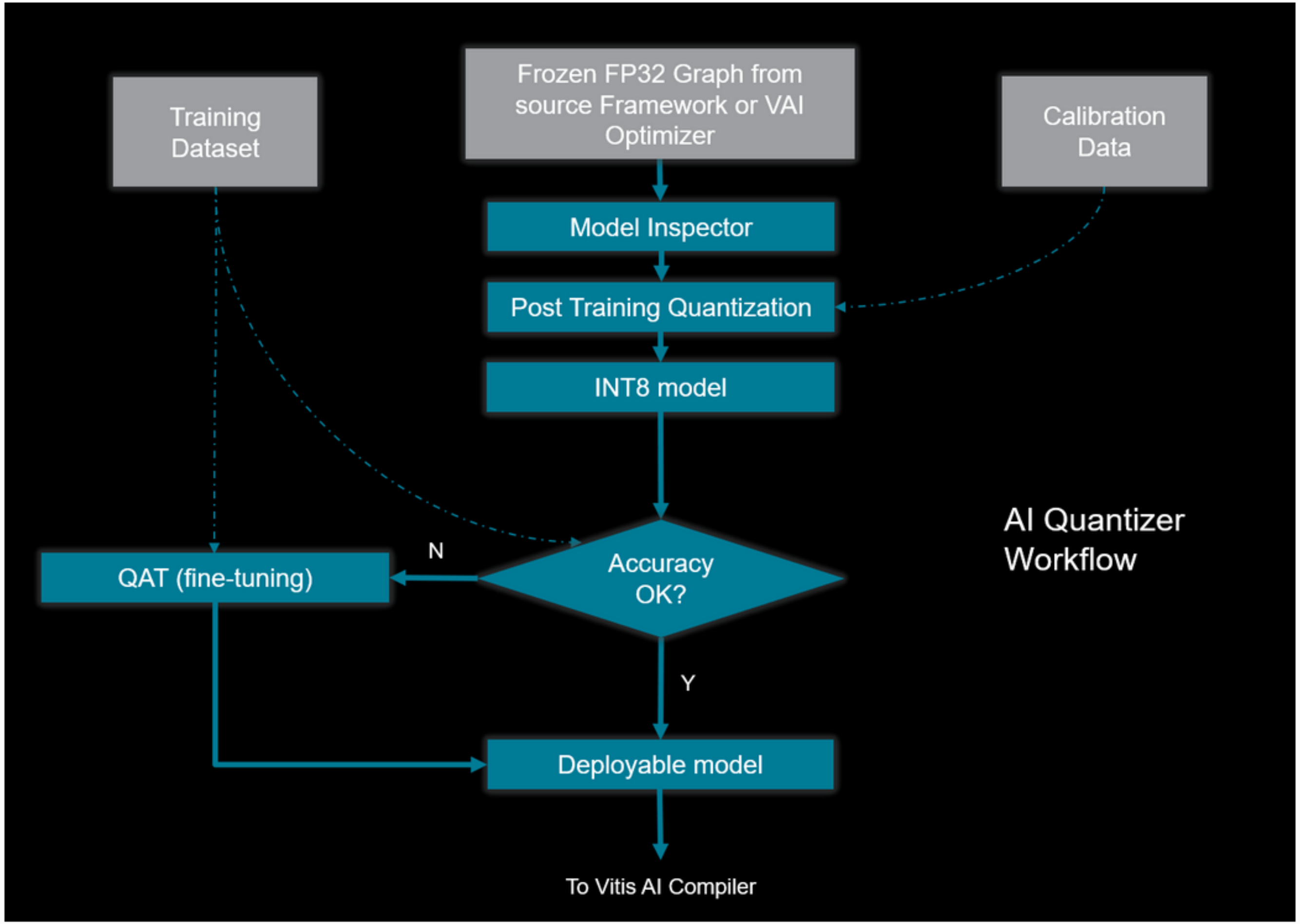
- ✓ Converts 32-bit floating-point weights and activations to fixed-point formats like INT8
- ✓ Significantly reduces computational complexity while preserving prediction accuracy. Reduces memory bandwidth demand, faster processing speed and improved power efficiency compared to the floating-point model.



Vitis-AI Quantizer



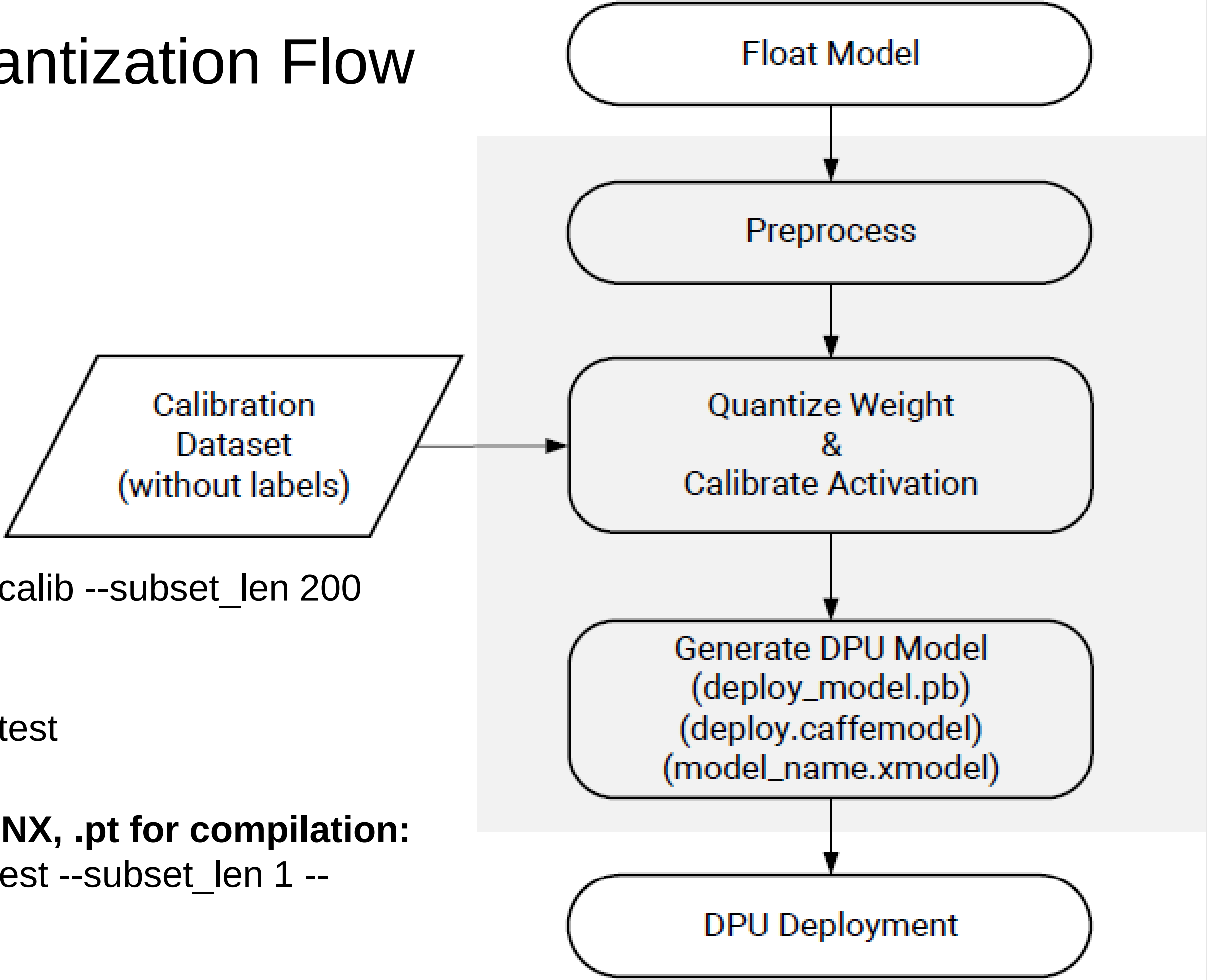
Quantization



Vitis AI Quantizer Workflow

Vitis-AI Quantizer: Quantization Flow

Install using docker:
[docker] $ conda activate vitis-ai-pytorch


Float Model
Preprocess
Calibration Dataset (without labels)
Quantize Weight & Calibrate Activation
Generate DPU Model (deploy_model.pb) (deploy.caffemodel) (model_name.xmodel)
DPU Deployment

Running quantization:

```
python resnet18_quant.py --quant_mode calib --subset_len 200
```

Evaluate quantized model:

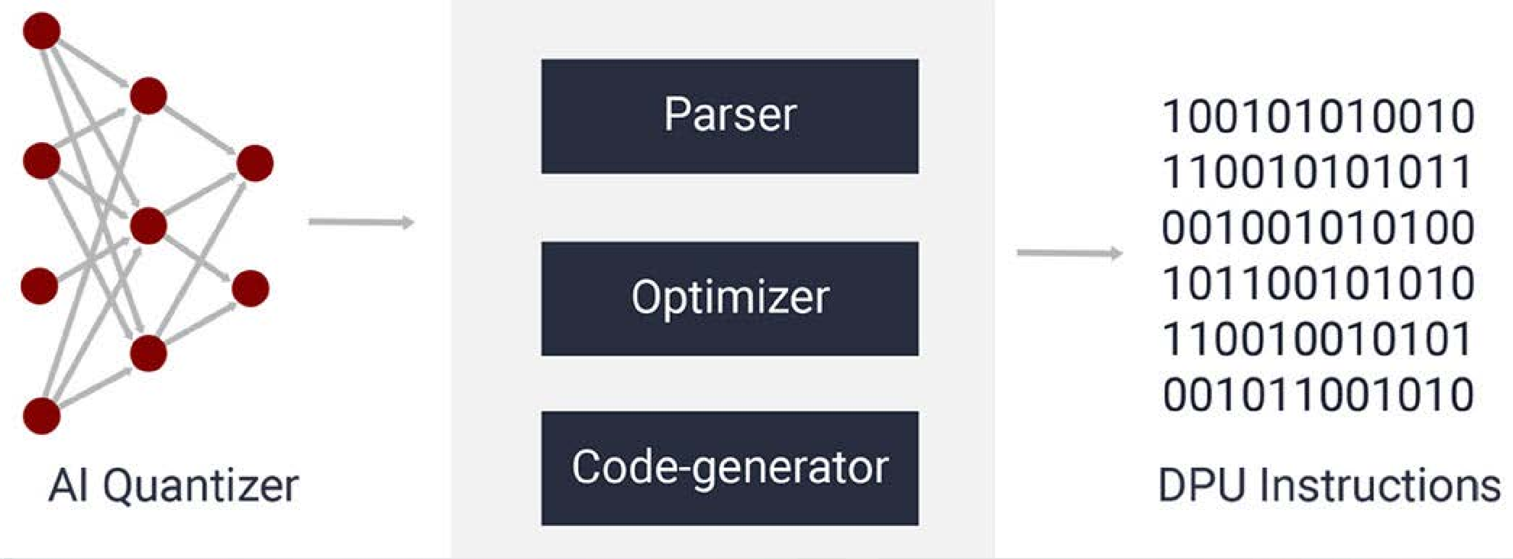
```
python resnet18_quant.py --quant_mode test
```

Generate To generate the XMODEL, ONNX, .pt for compilation:

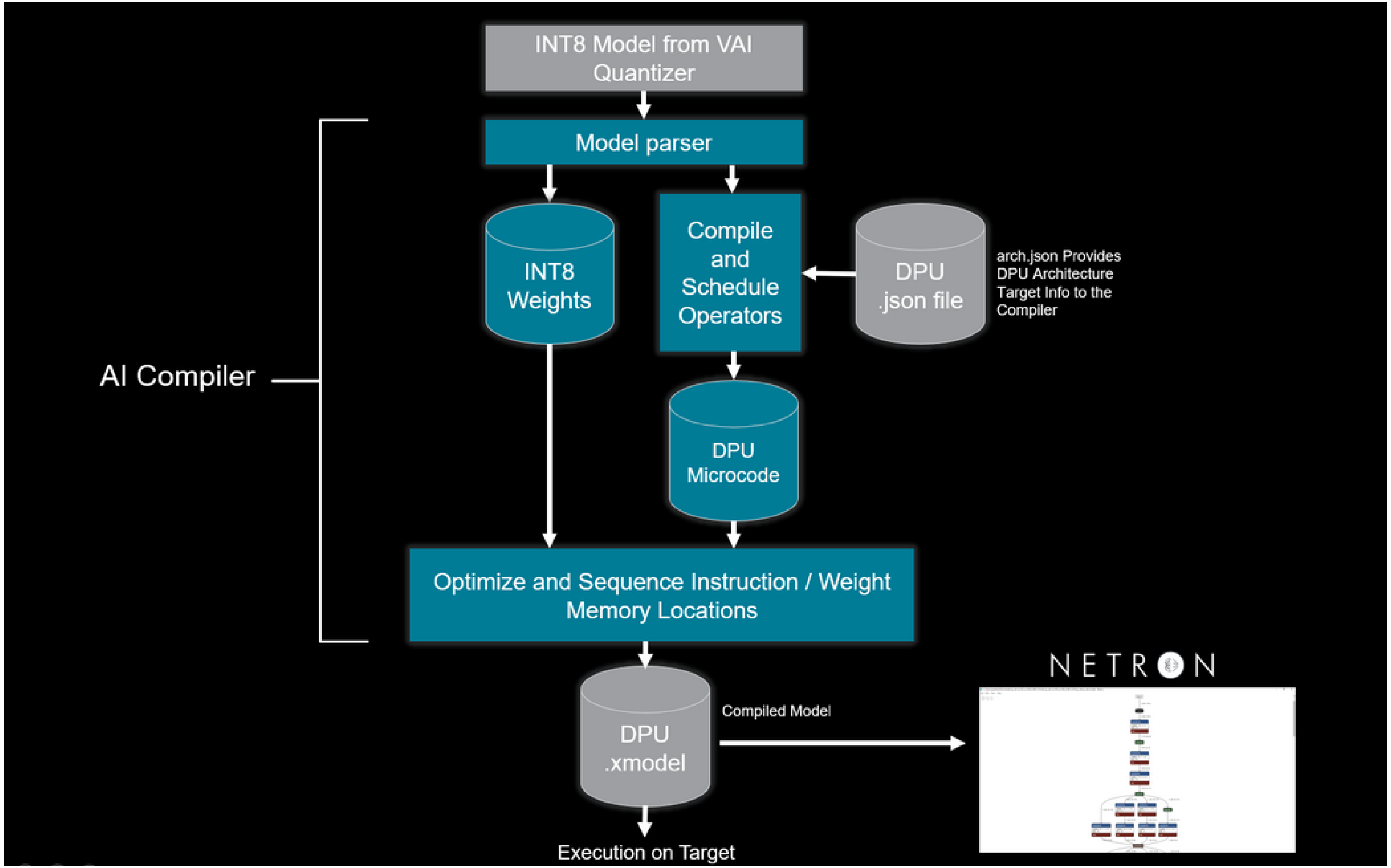
```
python resnet18_quant.py --quant_mode test --subset_len 1 --
batch_size=1   --deploy
```

Vitis-AI Compiler

- ✓ The Vitis AI compiler maps the AI model to a highly efficient instruction set and dataflow model (x.model).
- ✓ Performs sophisticated optimizations such as layer fusion, instruction scheduling, and reuses on-chip memory as much as possible.



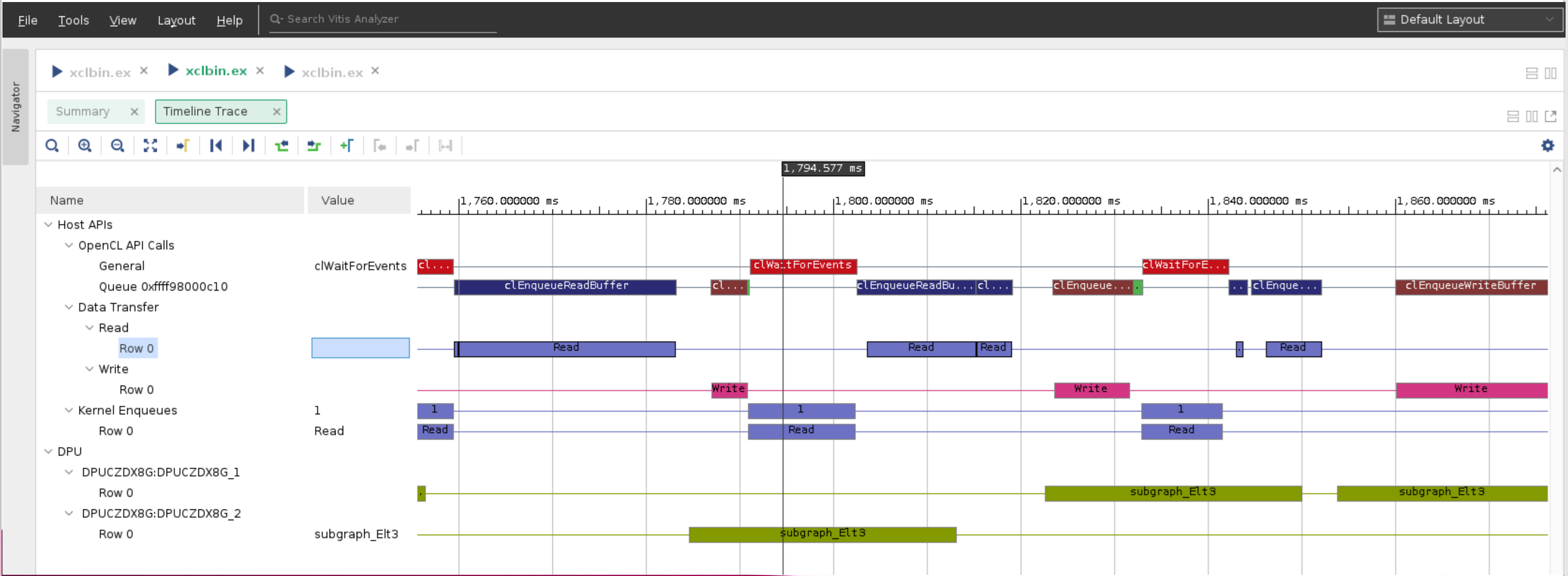
Compiler


INT8 Model from VAI Quantizer
Model parser
INT8 Weights
Compile and Schedule Operators
DPU .json file
arch.json Provides DPU Architecture Target Info to the Compiler
AI Compiler
DPU Microcode
Optimize and Sequence Instruction / Weight Memory Locations
NETRON
DPU .xmodel
Compiled Model
Execution on Target

Vitis AI Compiler Workflow

Vitis-AI Compiler

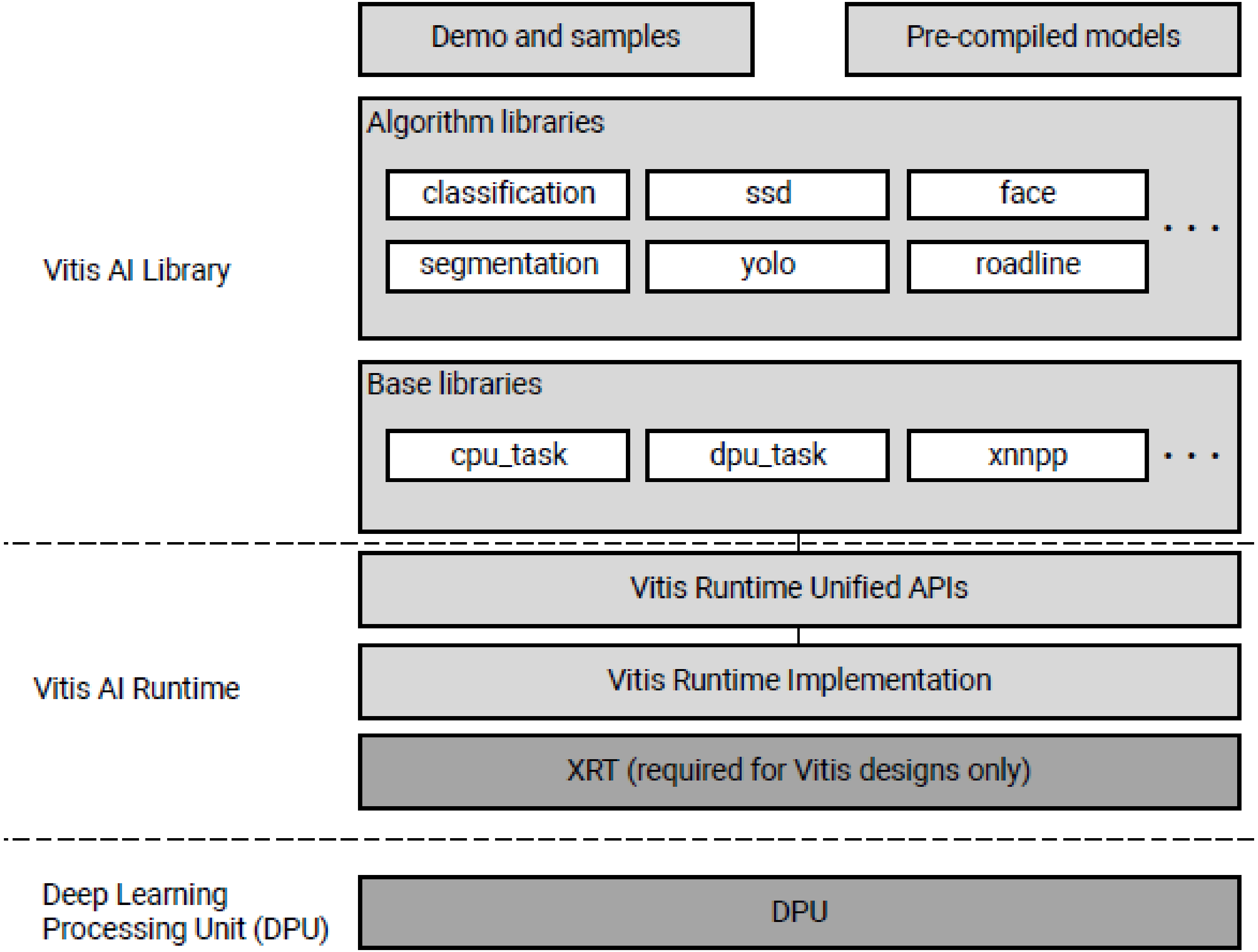
- ✓ Meticulously analyzes and visualizes AI applications to identify bottlenecks
- ✓ Optimally allocate computing resources across different devices.
- ✓ Empowers developers with comprehensive insights for efficient performance tuning.



Vitis-AI Library

- ✓ Collection of high-level libraries and APIs designed for efficient AI inference with DPUs.
- ✓ Seamlessly integrating with the Xilinx Runtime (XRT) and built upon the Vitis AI runtime with Vitis runtime unified APIs.
- ✓ Ensures a smooth and unified experience..

Vitis-AI Library



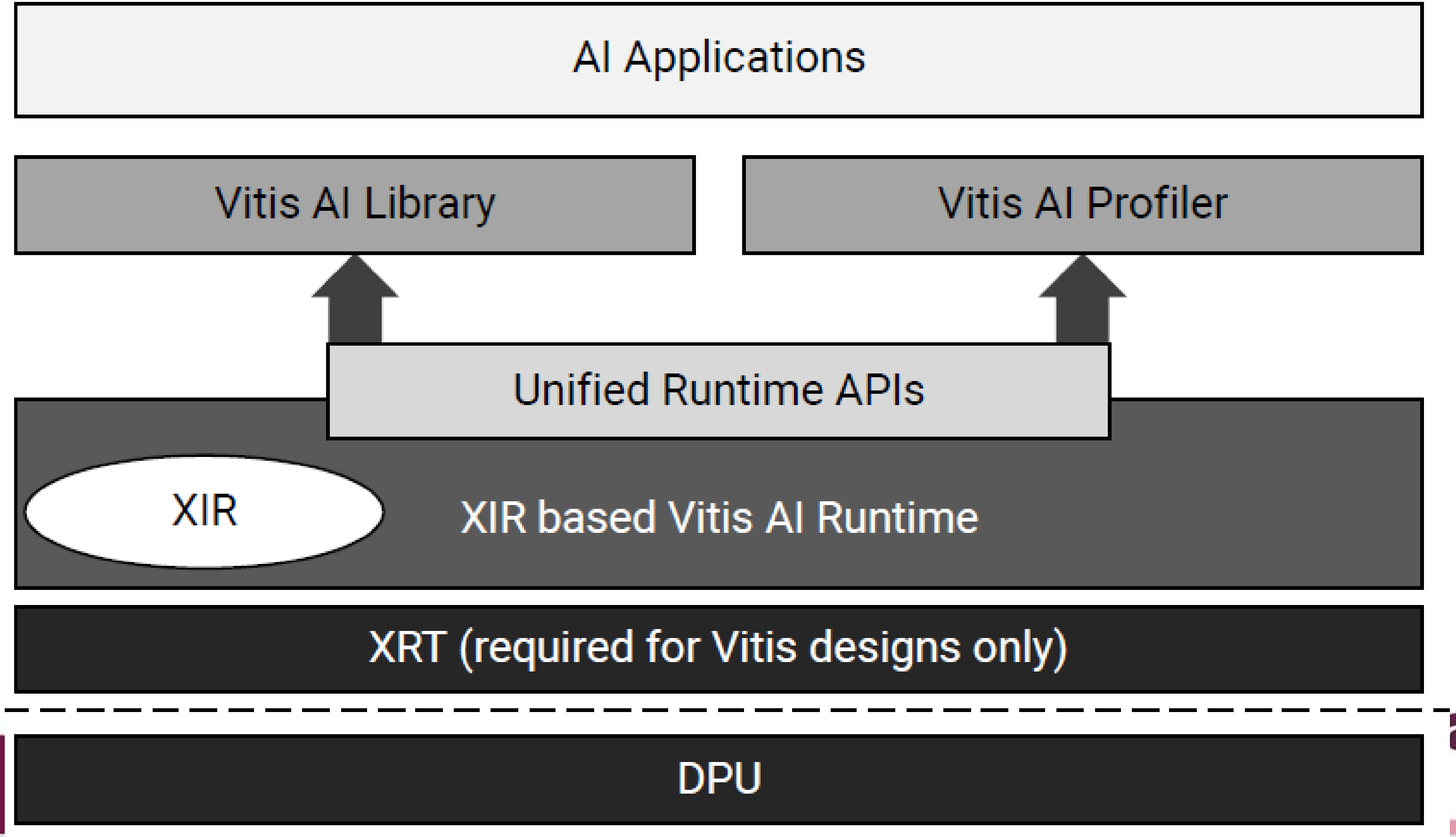
Vitis-AI Runtime

✓ An interface between Vitis AI libraries and the accelerator

The following are the features of the AI runtime API:

- Asynchronous submission of jobs to the accelerator
- Asynchronous collection of jobs from the accelerator
- C++ and Python implementations
- Support for multi-threading and multi-process execution

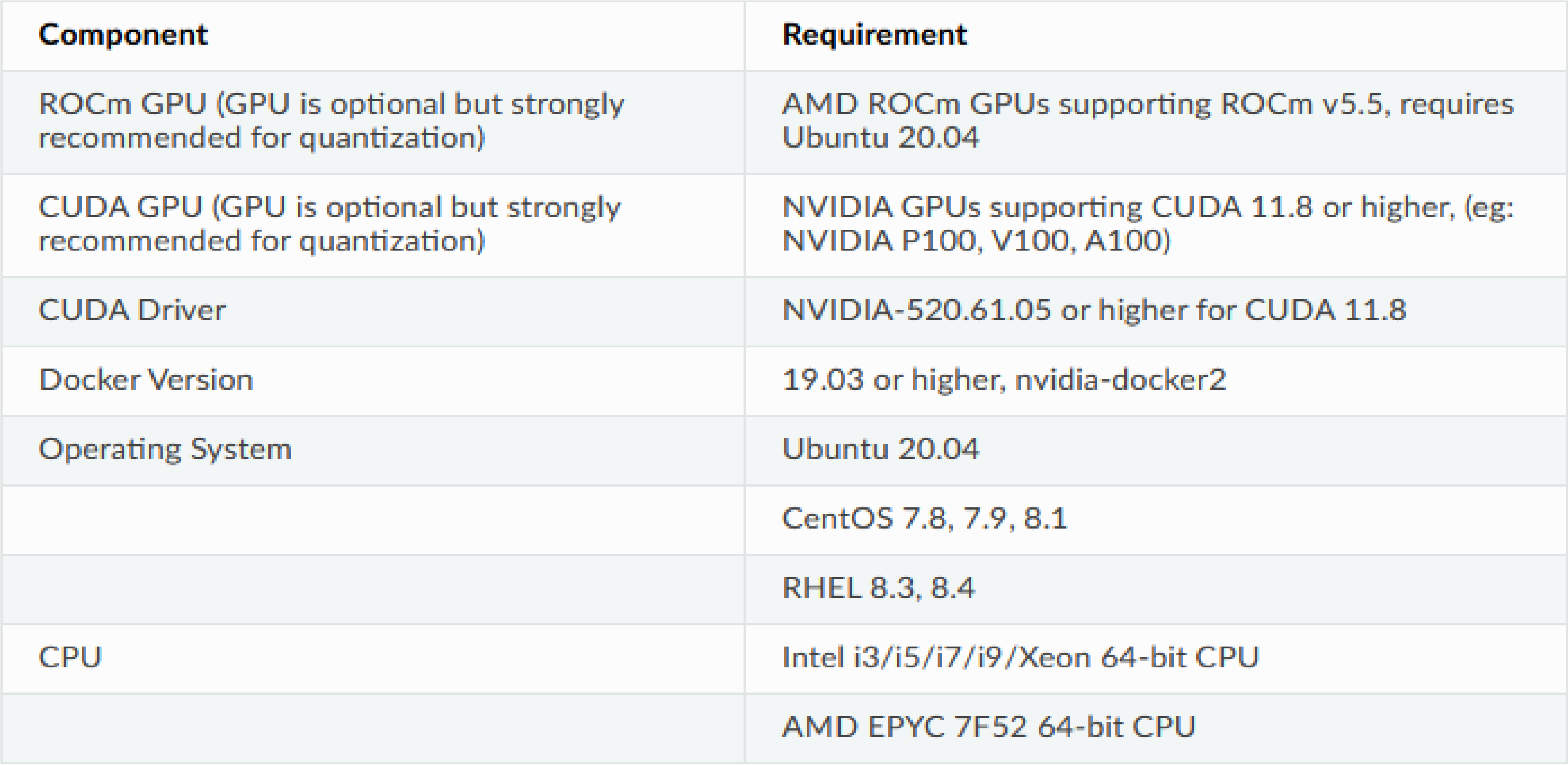
Vitis-AI Runtime Stack (VART Stack)



Vitis AI Containers

- ✓ Vitis AI 3.5 release uses containers to distribute the AI software.
- ✓ The release consists of the following components:
  - Tools container
    - ✓ Containers distributed through Docker Hub (https://hub.docker.com/u/Xilinx)
    - ✓ Unified compiler flow
    - ✓ Pre-built conda environment to run frameworks
    - ✓ Versal Runtime tools
  - Examples on the public GitHub (https://github.com/Xilinx/Vitis-AI)
  - Vitis AI Model Zoo (https://github.com/Xilinx/Vitis-AI/tree/v3.5/model_zoo)

Vitis AI Host (Developer) Machine Requirements

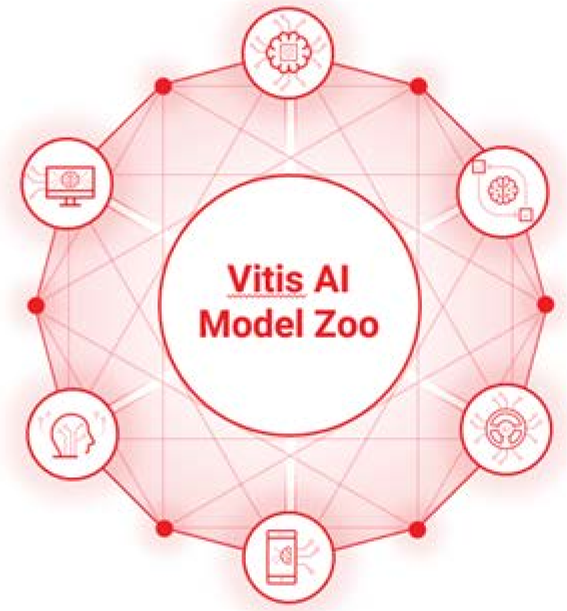


Component	Requirement
ROCm GPU (GPU is optional but strongly recommended for quantization)	AMD ROCm GPUs supporting ROCm v5.5, requires Ubuntu 20.04
CUDA GPU (GPU is optional but strongly recommended for quantization)	NVIDIA GPUs supporting CUDA 11.8 or higher, (eg: NVIDIA P100, V100, A100)
CUDA Driver	NVIDIA-520.61.05 or higher for CUDA 11.8
Docker Version	19.03 or higher, nvidia-docker2
Operating System	Ubuntu 20.04
	CentOS 7.8, 7.9, 8.1
	RHEL 8.3, 8.4
CPU	Intel i3/i5/i7/i9/Xeon 64-bit CPU
	AMD EPYC 7F52 64-bit CPU

Vitis-AI Model Zoo

- Curated collection of finely tuned deep learning models designed to
- accelerate the deployment of AI inference on AMD platforms.
- Encompasses diverse applications, such as ADAS/AD, video surveillance, robotics, and data centers.
- Equips developers with powerful tools and optimized models to unlock the advantages of deep learning acceleration.

Vitis-AI Model Zoo



- Rich Models in PyTorch, TensorFlow and ONNX
- Open and Free on GitHub for All Developers
- Advanced Optimization, Pruning Included
- Retrainable with Custom Dataset

Vitis-AI Model Zoo: Links

Model Zoo Details & Performance table online:
https://xilinx.github.io/Vitis-AI/3.5/html/docs/reference/ModelZoo_Github_web.htm

Model Zoo Github:
https://github.com/Xilinx/Vitis-AI/tree/v3.0/model_zoo

Vitis AI Copyleft Model Zoo.
https://github.com/Xilinx/Vitis-AI-Copyleft-Model-Zoo

Documentation:
https://xilinx.github.io/Vitis-AI/3.5/html/docs/workflow-model-zoo.html

Spreadsheet:
https://xilinx.github.io/Vitis-AI/3.5/html/_downloads/ff9554ff9ff6240811c20ede15113dbd/ModelZoo_Github.xlsx

PYNQ-DPU

- Is a platform facilitating DPU deployment on FPGAs using PYNQ
  - ✓ https://github.com/Xilinx/DPU-PYNQ/tree/design_contest_3.5
- Provides pre-built DPU hardware and compiled models
- Provides tools for Re-building your own DPU and hardware
- Supports different Xilinx FPGA boards

Quick Install

In the Jupiter Lab Terminal run:

```
>> pip3 install pynq-dpu --no-build-isolation
```

***Then go to your Jupyter Notebook home folder and fetch the notebooks**:*

```
>> cd $PYNQ_JUPYTER_NOTEBOOKS
>> pynq get-notebooks pynq-dpu -p .
```

專業 創新 胸懷全球
Professional · Creative
For The World

Department of
Electrical Engineering

香港城市大學
City University of Hong Kong