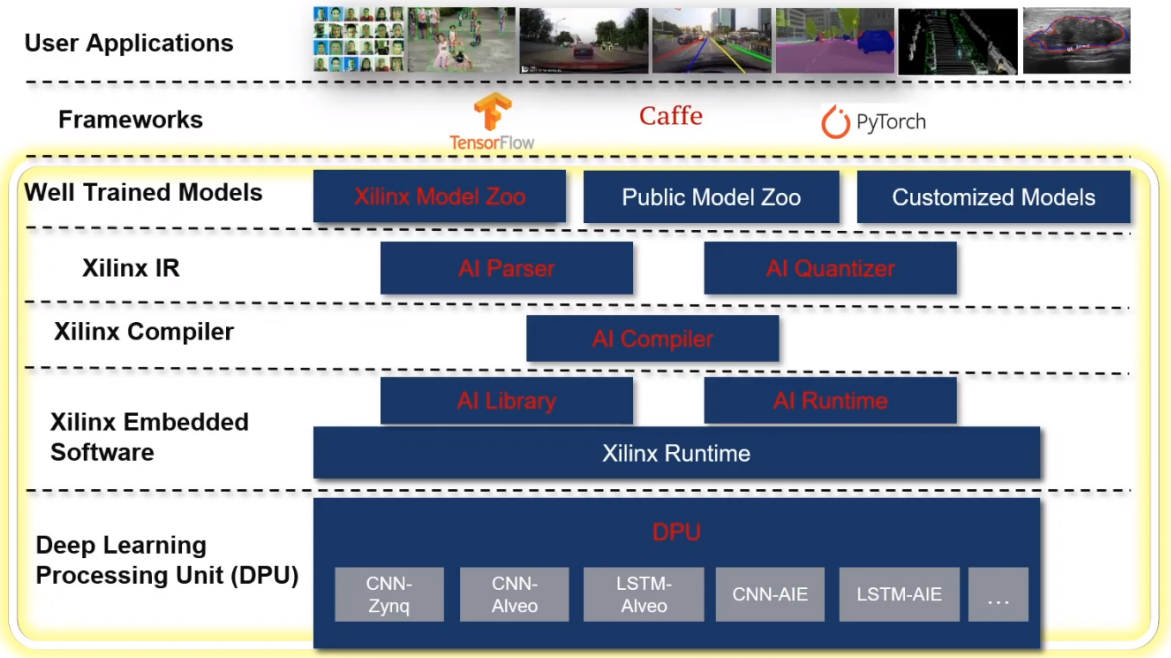
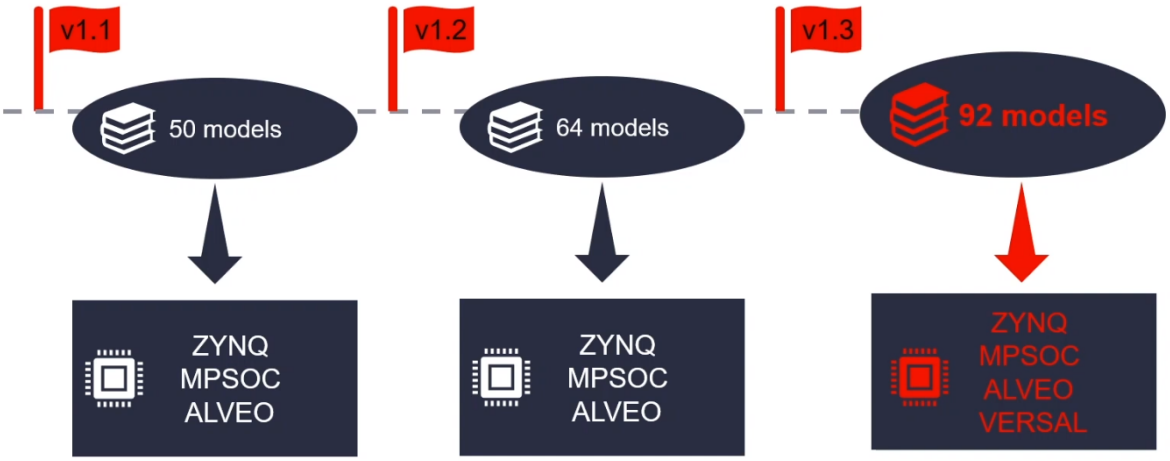
Vitis AI: Unified AI Inference Solution Stack



Model Zoo

- Provides state-of-art models.
- Yaml file for each model
- Link for different overlays
- Readable from AI Library



AI Parser & Quantizer: Workflow

In Vitis AI, the AI parser and quantizer are key components of the workflow for deploying and optimizing deep learning models on Xilinx FPGA devices. Here is an overview of the workflow:

1. Model Conversion: The workflow typically begins with converting a trained deep learning model from popular frameworks such as TensorFlow, PyTorch, or Caffe into an intermediate representation format supported by Vitis AI, such as ONNX (Open Neural Network Exchange) or Caffe prototxt.

2. AI Quantizer: The AI quantizer is used to quantize the model to lower precision, typically from floating-point (FP32) to fixed-point (INT8) or reduced-precision floating-point (FP16). Quantization reduces the memory footprint and computational requirements of the model, making it more suitable for deployment on FPGA devices. The AI quantizer performs calibration and quantization-aware training techniques to minimize the loss of accuracy caused by quantization.

3. AI Compiler: Once the model is quantized, the AI compiler is used to compile the model for the target FPGA device. The AI compiler performs various optimizations, such as layer fusion, memory optimization, and kernel customization, to improve the performance and efficiency of the model on the FPGA.

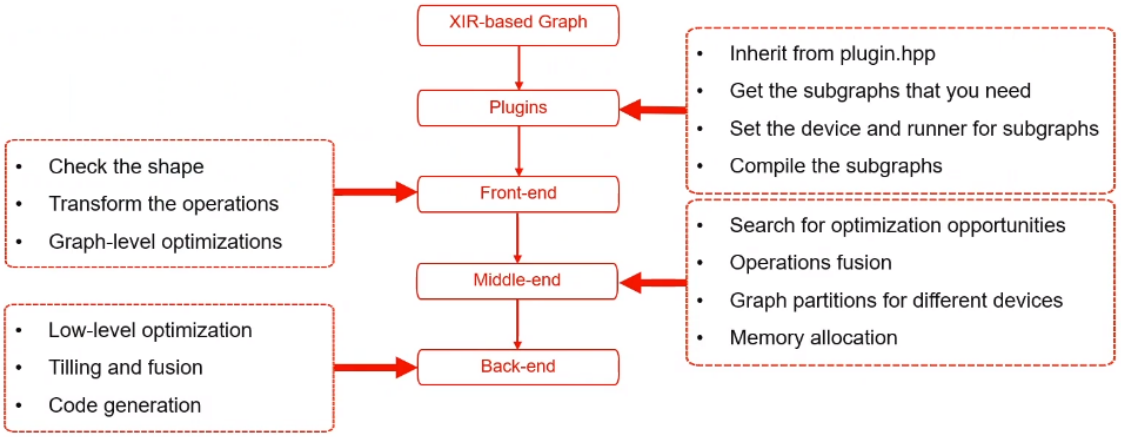
4. AI Library: The AI library provides a collection of pre-optimized FPGA-accelerated functions, known as AI kernels, for common deep learning operations. These kernels are specifically designed and optimized for the target FPGA architecture and can be used to accelerate the execution of the quantized model.

5. FPGA Deployment: After compilation, the generated bitstream along with the quantized model and AI kernels are deployed onto the target FPGA device. The FPGA device, with its parallel processing capabilities, allows for efficient execution of the quantized model, providing low-latency and high-throughput performance.

The Vitis AI workflow leverages the Xilinx Vitis unified software platform, which integrates tools, libraries, and runtime environments for FPGA development. It enables developers to optimize deep learning models for FPGA acceleration and deploy them in real-world applications that require low latency and high performance.

Note that the workflow may vary depending on the specific requirements of the application and the target FPGA device.

AI Compiler

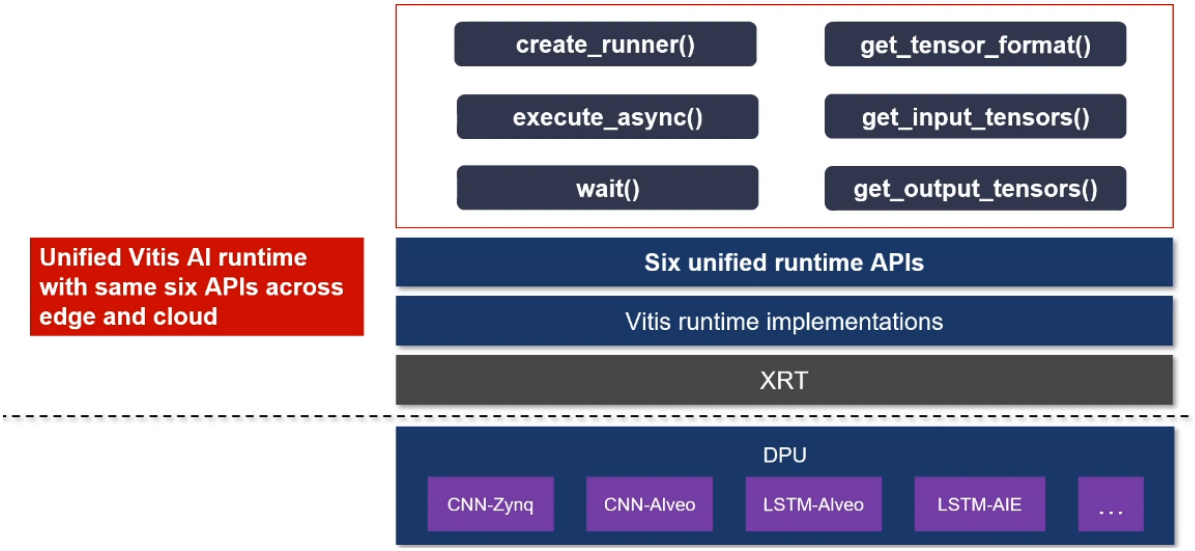


The first step the compiler does is convert the tensorflow or pytorch model into a XIR based graph, it is a intermediate graph representation similar to ONNX. It is a computation graph with various subgraphs consisting various nodes of operations.To understand it a little bit of graph theory is required but for now we can assume it is a common representation used by vitis ai.

Plugin is a feature that determines which subgraphs should be run on other hardware(other than DPU) like CPU, GPU or other FPGA based accelerators. This is helpful because not all operations are supported by DPU, so these operations can be separated from the graph and offloaded or assigned to be run on other hardware.

Regarding the front-end, middle-end and back-end, it is the subdivision of operations performed by compiler. The compiler in Vitis AI is closed source so not much is known about the whole internal workings of the compiler and it is also not required for us to know in depth of how it works fully. We need to know is DPU like a processor designed to specifically run CNN related operations and Compiler compiles the XIR graph into machine level code that is accepted by DPU. There are certain operations supported by DPU and we modify the NN architecture based around that while the unsupported operations we either replace it by supported operations or offload those to other hardware like CPU, GPU or FPGA implemented functions using Plugins. This is the general abstraction that we need to know.

VART: Unified runtime APIs



VART (Vitis AI Runtime) is a software component of the Vitis AI framework provided by Xilinx. It provides a set of unified runtime APIs (Application Programming Interfaces) that enable developers to interact with and control AI applications running on FPGA-based platforms.

The VART APIs serve as an interface between the host CPU and the FPGA accelerator, allowing seamless communication and coordination between the two components. These APIs abstract the underlying hardware details and provide a high-level programming model for developers to leverage the computational power of the FPGA accelerator in their AI applications.

Some key features and functionalities of VART's unified runtime APIs include:

1. Model Loading: VART APIs allow developers to load pre-trained AI models onto the FPGA accelerator. These models can be in popular formats such as TensorFlow or Caffe.

2. Model Execution: Once the model is loaded, VART provides APIs to execute the model on the FPGA accelerator. Developers can pass input data to the model and retrieve the output results.

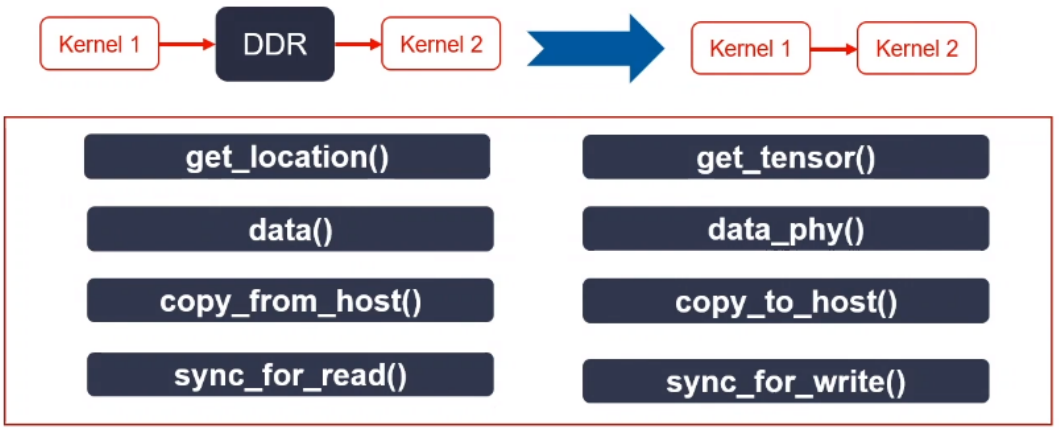
3. Data Movement: VART supports efficient data movement between the host CPU and the FPGA accelerator. It provides APIs for data transfer, including zero-copy techniques to minimize data copying overhead.

4. Resource Management: VART APIs help manage the resources of the FPGA accelerator, such as memory allocation and deallocation. This ensures efficient utilization of the FPGA resources and avoids memory conflicts.

5. Performance Optimization: VART provides APIs for performance profiling and tuning. Developers can analyze the execution time and resource utilization of their AI applications and optimize them for better performance.

By providing unified runtime APIs, VART simplifies the development process for AI applications on FPGA-based platforms. It abstracts the complexities of the underlying hardware and enables developers to focus on implementing and optimizing their AI algorithms, while leveraging the computational power of the FPGA accelerator.

VART: Zero Copy



New added APIs to achieve zero copy

In the context of Vitis AI, "zero copy" refers to a technique that allows data to be shared between the host CPU and the FPGA accelerator without unnecessary data copying. It enables efficient data movement between the CPU and the accelerator, minimizing latency and reducing memory bandwidth requirements.

Typically, when data needs to be processed by an FPGA accelerator, it is first transferred from the CPU's memory to the FPGA's memory. This transfer involves copying the data, which can introduce overhead in terms of time and memory bandwidth.

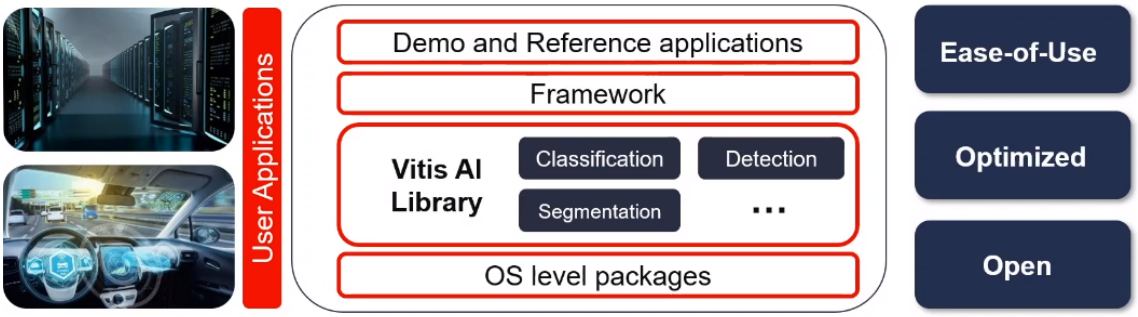
Zero copy eliminates the need for this intermediate data copy by allowing the accelerator to directly access the host CPU's memory. This means that the data resides in the same memory space accessible by both the CPU and the accelerator. As a result, the accelerator can read or write data directly from/to the CPU's memory, avoiding the need for data duplication.

By using zero copy, the data transfer between the CPU and the accelerator becomes more efficient, as it eliminates unnecessary data copying overhead. This can be particularly beneficial when dealing with large datasets or real-time applications that require low latency.

Vitis AI provides APIs and libraries that enable developers to implement zero copy data movement between the CPU and the FPGA accelerator, optimizing the overall performance of AI applications running on FPGA-based platforms.

Vitis AI Library: the What?

- **Vitis AI Library provides high-level API based libraries across different vision tasks: classification, detection, segmentation and etc.**
  - **Reference applications to help customers' fast prototyping**
  - **Optimized codes used in AI applications and products**

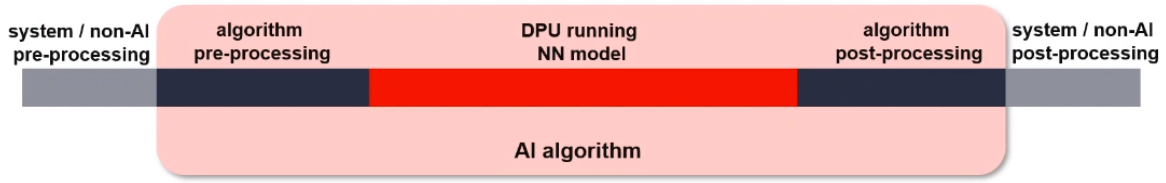


AI Application General Processing Flow

- **A typical abstraction of processing flow:**

- **Algorithm-level processing**
  - » Data normalization before sending to DPU
  - » Post processing (e.g. bounding boxes decoding in detection)
- **Additional system-level workloads for AI inference**
  - » Color conversion / resizing
  - » Path planning / control / status update

What Vitis AI Library Provides



- **AI Library offers libraries for**
  - Algorithm-level optimization
  - Open and easy to extend
  - Directly support models in AI Model Zoo

AI Library Samples

- The Vitis AI Library provides image test samples ,video test samples, performance test samples for all the above networks. Each sample has the following four kinds of test sample.
  - test_jpeg_[model type]
  - test_video_[model type]
  - test_performance_[model type]
  - test_accuracy_[model type]
- In addition, the kit provides the corresponding performance test program. For video based testing, we recommend to use raw video for evaluation. Because decoding by software libraries on Arm® CPU may have inconsistent decoding time, which may affect the accuracy of evaluation.

Easy-to-Use APIs to Deploy Full Algorithm