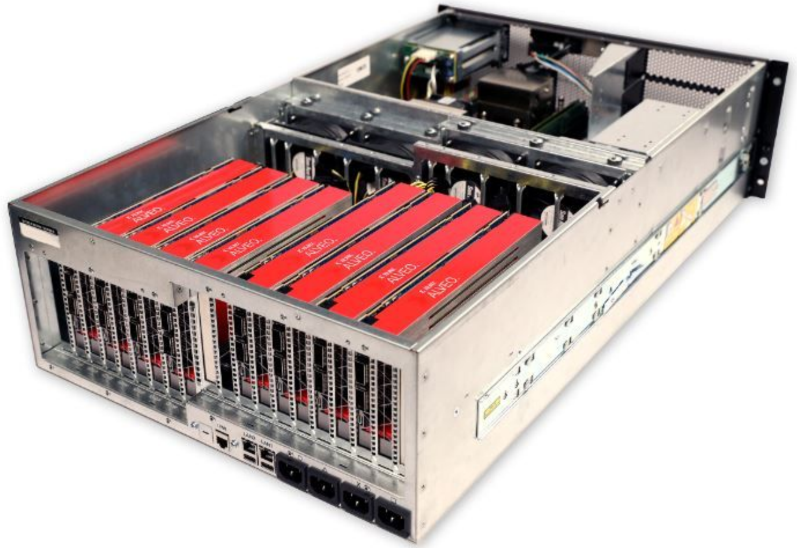
FPGA CNN Accelerator and TVM

Elliott Delaye

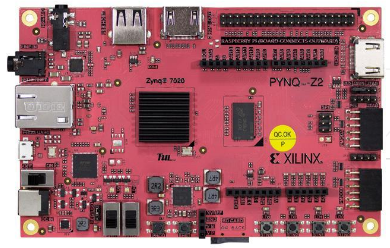
XILINX

TVM Target devices and models

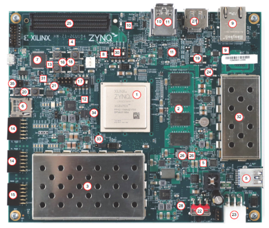
HW Platforms



ZCU102



PYNQ

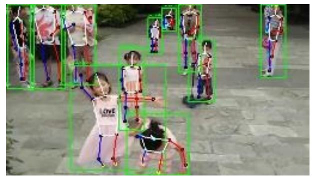


ZCU104

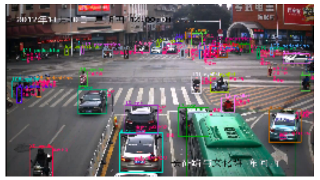
Ultra96

Models

Face detection



Pose estimation



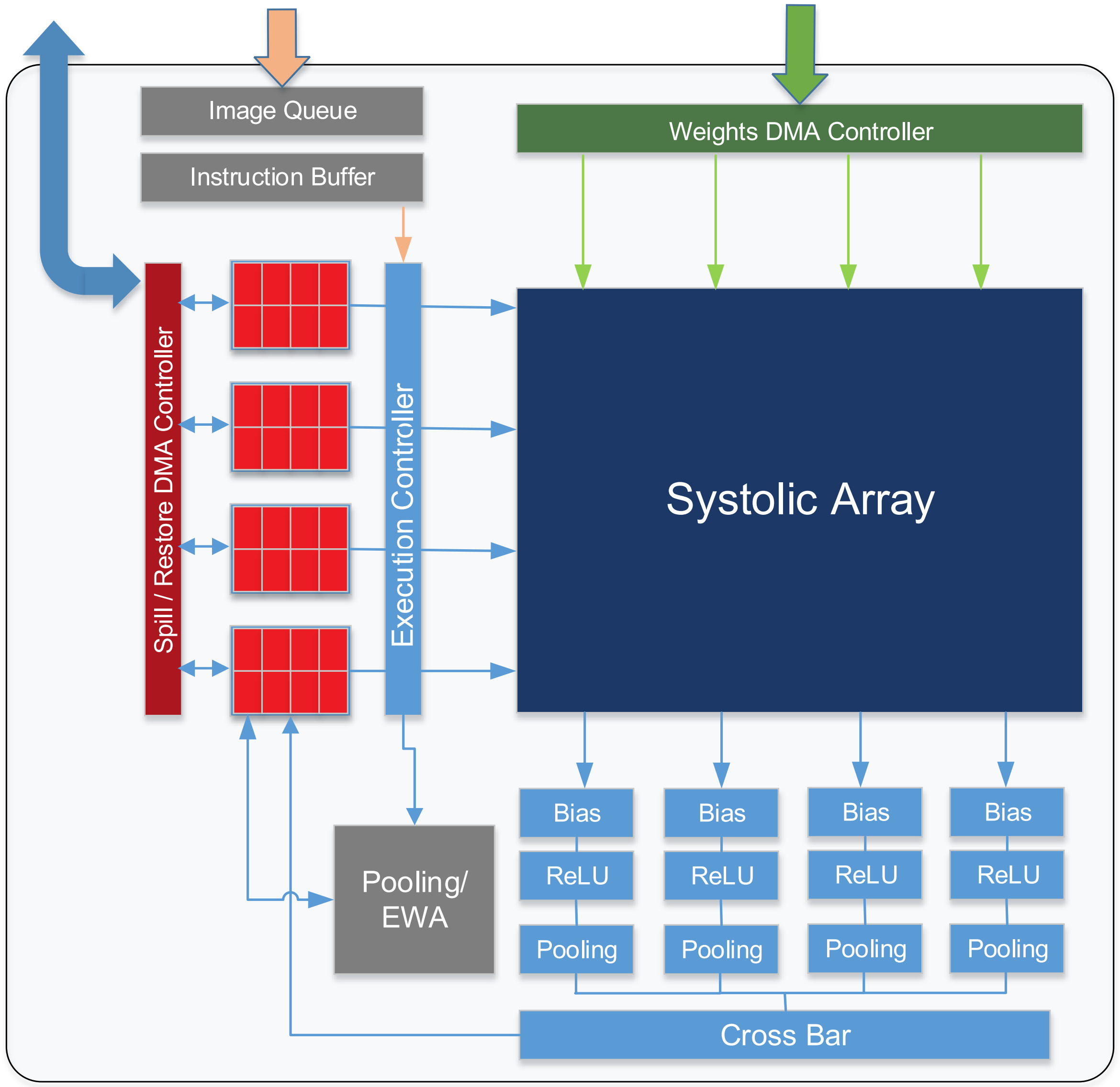
Video analytics

Lane detection

Object detection

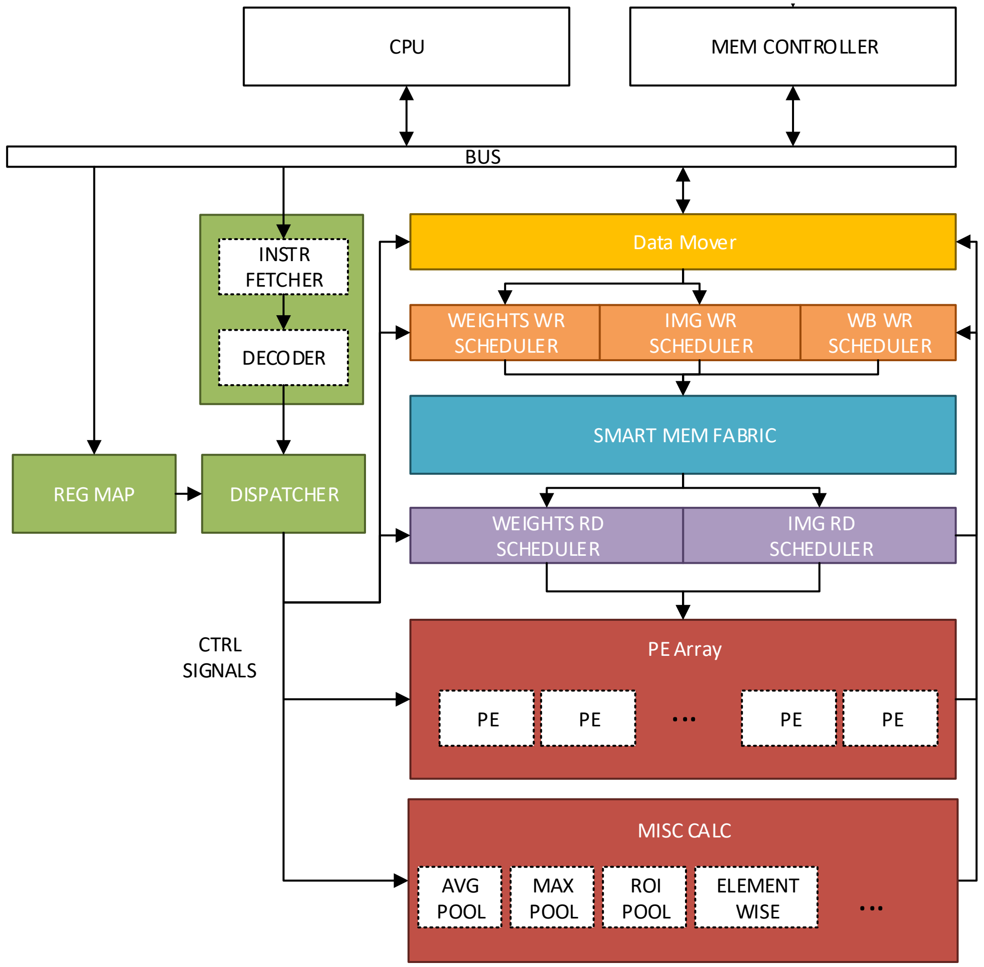
Segmentation

Xilinx Cloud DPU Processor (xDNNv3)

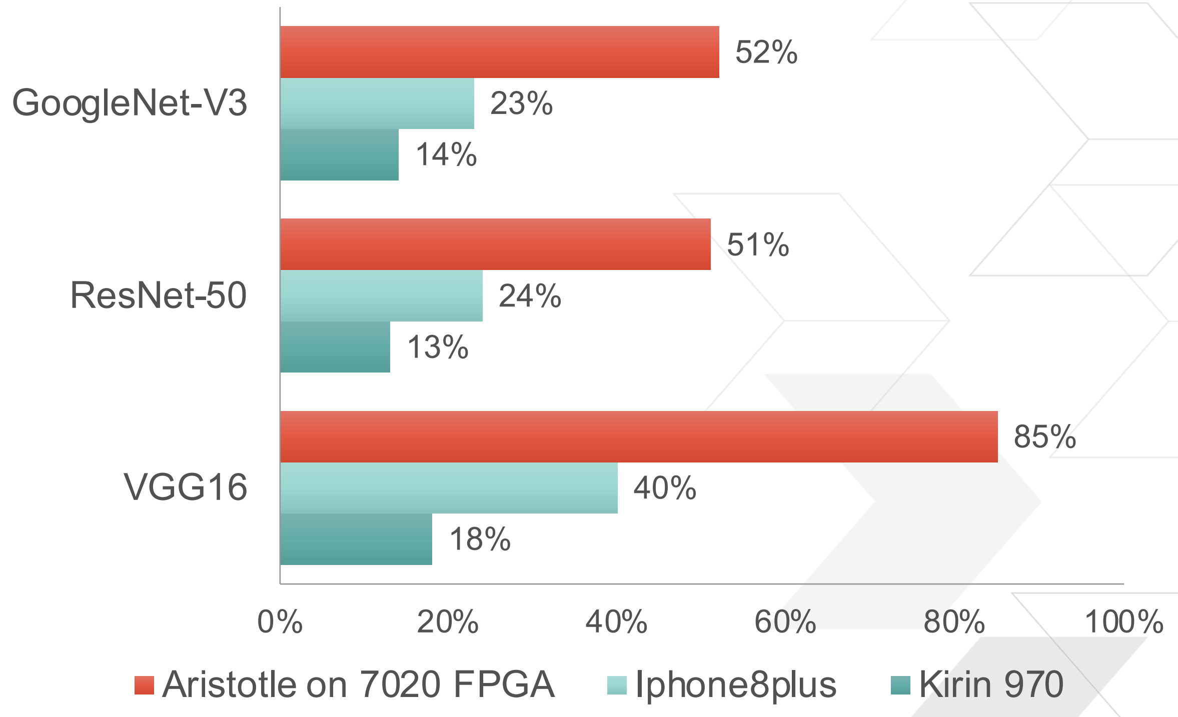


- Configurable Overlay Processor
- DNN Specific Instruction Set
  - Convolution, Max Pool etc.
- Any Network, Any Image Size
- High Frequency & High Compute Efficiency
- Supported on
  - U200 – 3 Instances
  - U250 – 4 Instances
  - Amazon F1
- ~1536 DSPs @ 700MHz

Xilinx Edge DPU IP (DPUv2)

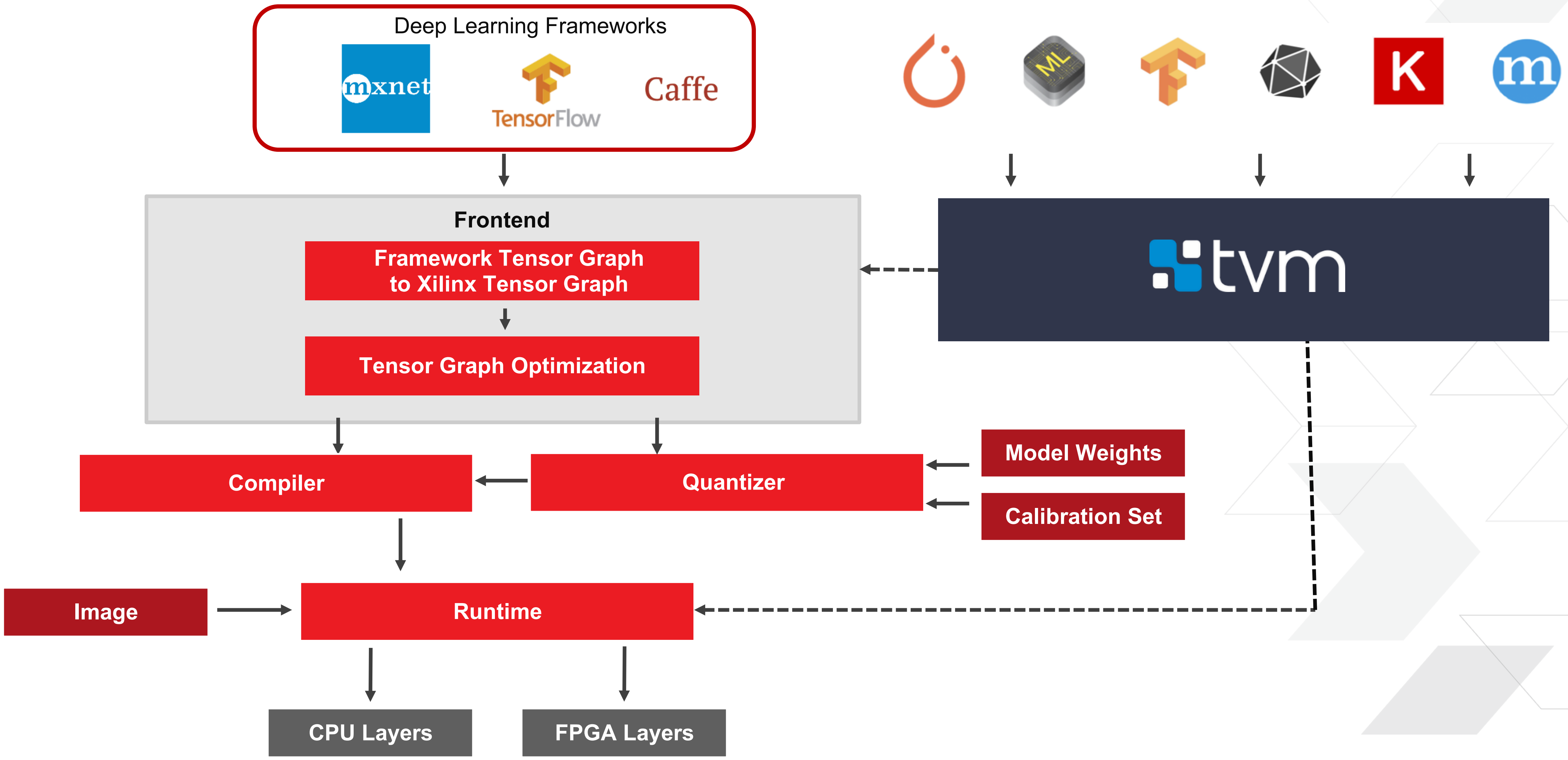


Efficiency > 50% for mainstream neural networks



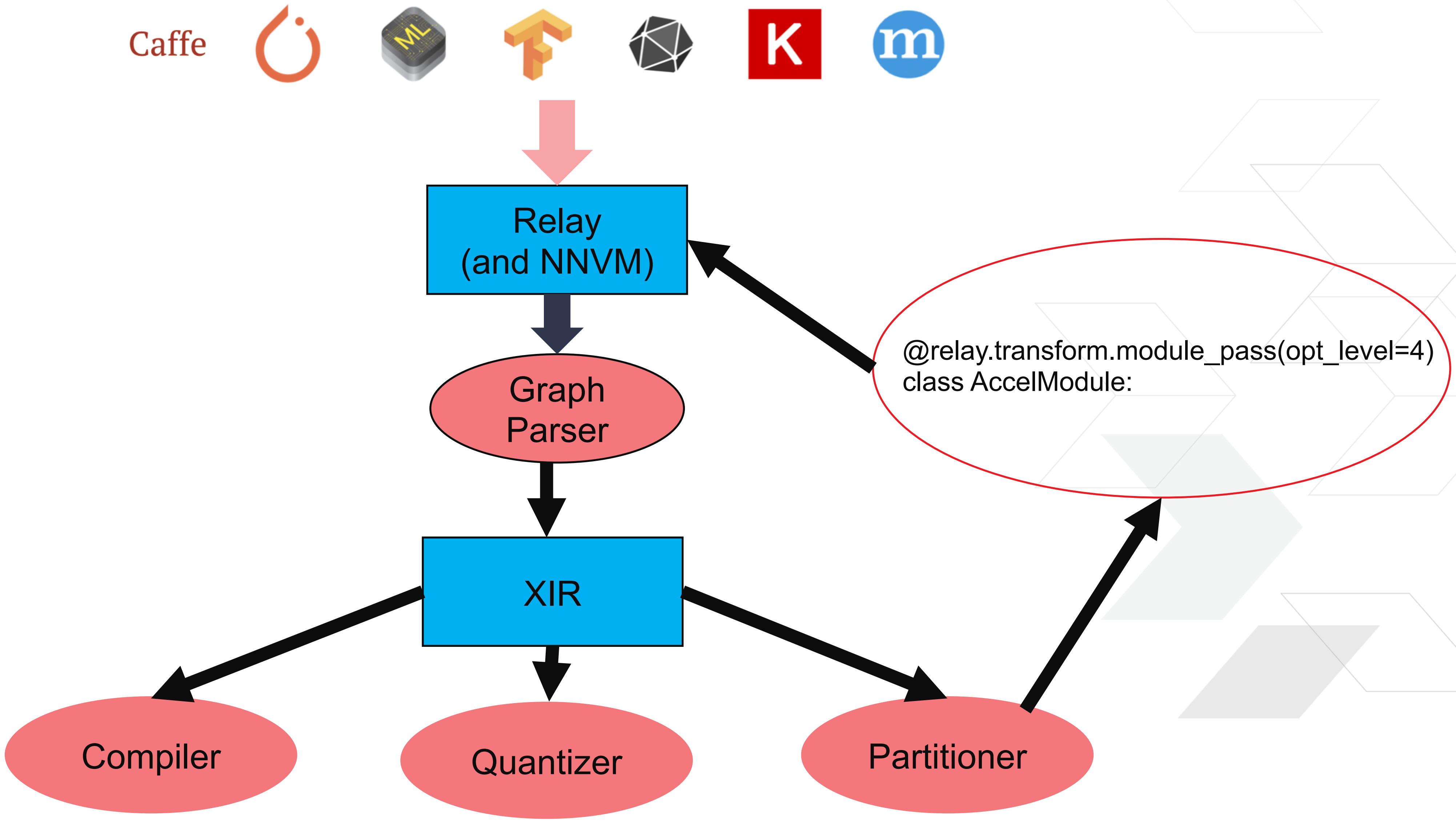
Source: Published results from Huawei

Inference Flow



https://github.com/xilinx

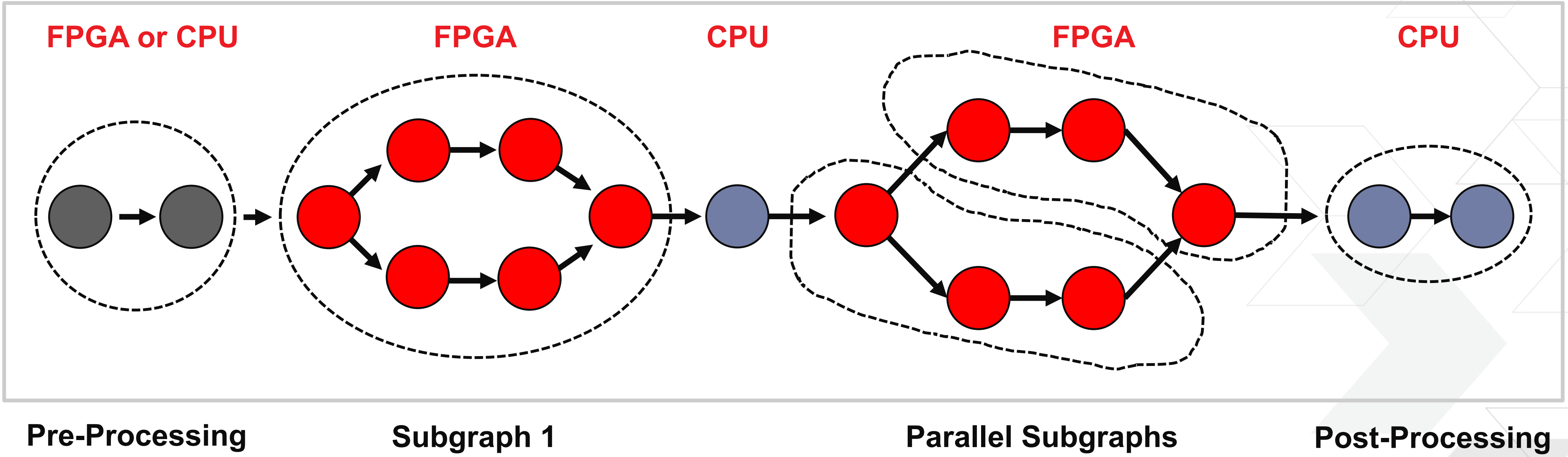
TVM as Unified ML Front End



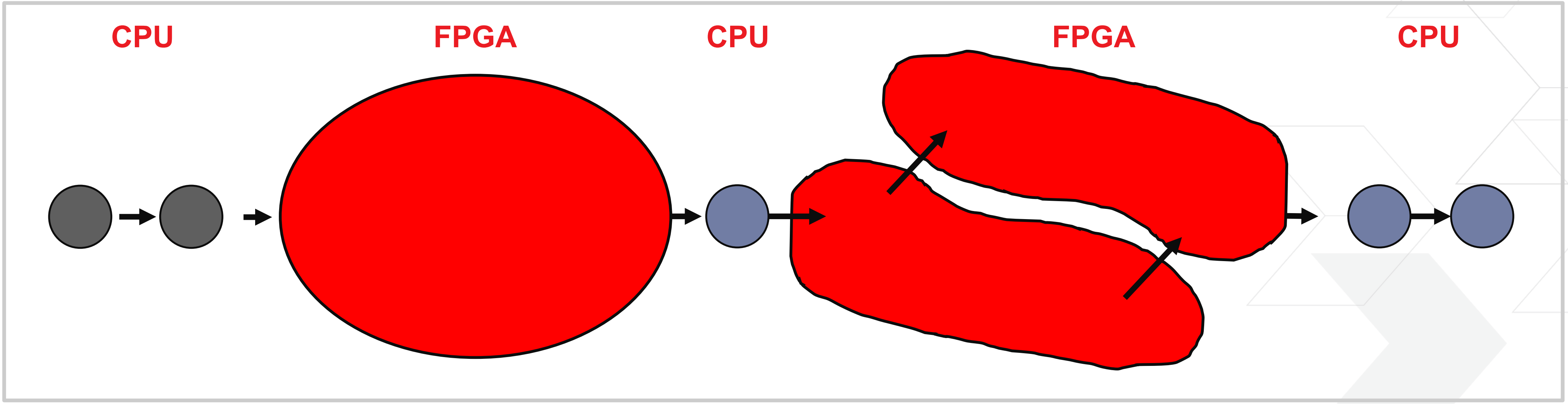
```
@relay.transform.module_pass(opt_level=4)
class AccelModule:
```

TVM Partitioning

- More than supported/not supported, pattern matching graph colorization
- Choices how to partition especially for multi-branch networks (i.e. YOLOv3, SSD)



TVM Graph Partitioning/Fusion



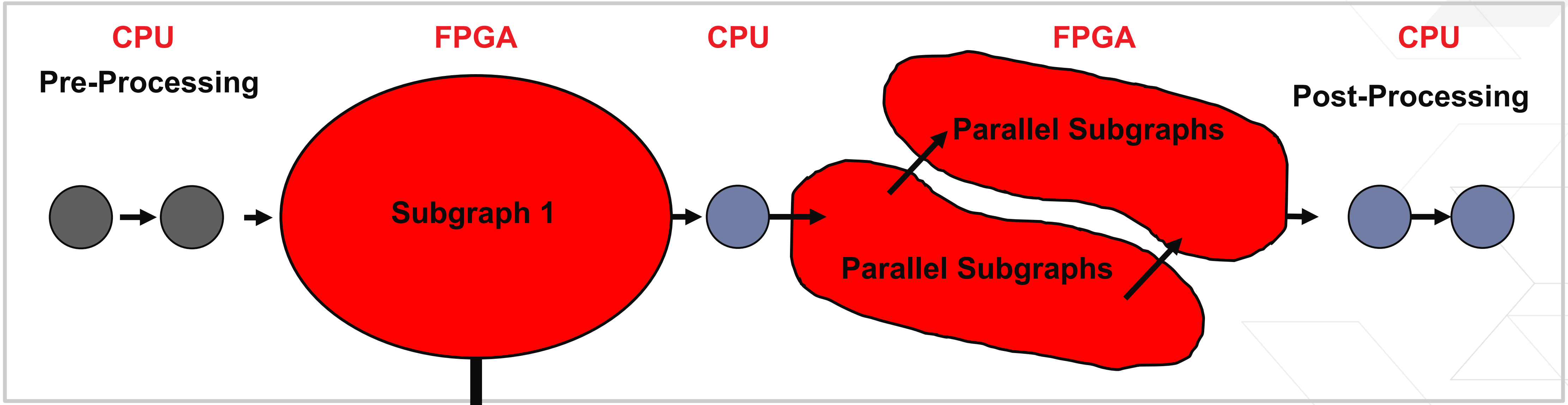
Pre-Processing

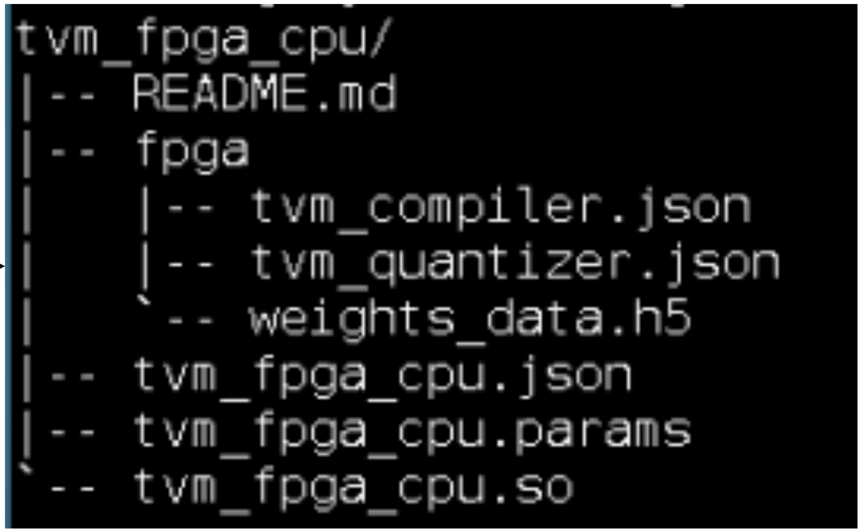
Subgraph 1

Parallel Subgraphs

Post-Processing

TVM Code Generation





```
tvm_fpga_cpu/
|-- README.md
|-- fpga
|   |-- tvm_compiler.json
|   |-- tvm_quantizer.json
|   `-- weights_data.h5
|-- tvm_fpga_cpu.json
|-- tvm_fpga_cpu.params
`-- tvm_fpga_cpu.so
```

Registering external accelerator function

```
@reg.register_compute("accel", level=15)
def compute_accel(attrs,inputs,outputs):
    op = 'accel'
    name = 'accel0'
    attrs_dict = { k: attrs[k] for k in attrs.keys() }
    input_names = [inpt.op.name for inpt in inputs]
    in_shapes = [[int(i) for i in inpt.shape] for inpt in inputs]
    out_shapes = [[int(i) for i in outputs[0].shape]]
    # EXTERNAL FUNCTION TO RUN THE FUSED OPERATION
    out = tvm.extern(outputs[0].shape, inputs, lambda ins, outs: tvm.call_packed('tvm.accel.accel_fused', attrs['path'],
attrs['output_layout'], attrs['model_name'], outs[0], *ins ), name=name)
return out
```

Example of FPGA node in TVM graph

```
{
  "nodes": [
    {
      "op": "null",
      "name": "data",
      "inputs": []
    }
    {
      "op": "tvm_op",
      "name": "xdnn0",
      "attrs": {
        "flatten_data": "0",
        "func_name": “accel_fused",
        "num_inputs": "1",
        "num_outputs": "1"
      },
      "inputs": [[0, 0, 0]]
    },
    {
      "op": "tvm_op",
      "name": "flatten0",
      "attrs": {
        "flatten_data": "0",
        "func_name": "fuse_flatten",
        "num_inputs": "1",
        "num_outputs": "1"
      },
      "inputs": [[1, 0, 0]]
    },
```

Calls XDNN’s TVM registered function to access the FPGA runtime APIs

XILINX.

Registering TVM op in Python at runtime

File contrib_xlnx.py:

```
…
@tvm.register_func("tvm.accel.accel_fused")
def accel_fused(graph_path, output_layout,  out, *ins ):
    path   = c_char_p(graph_path.value).value
    layout = c_char_p(output_layout.value).value
…
```

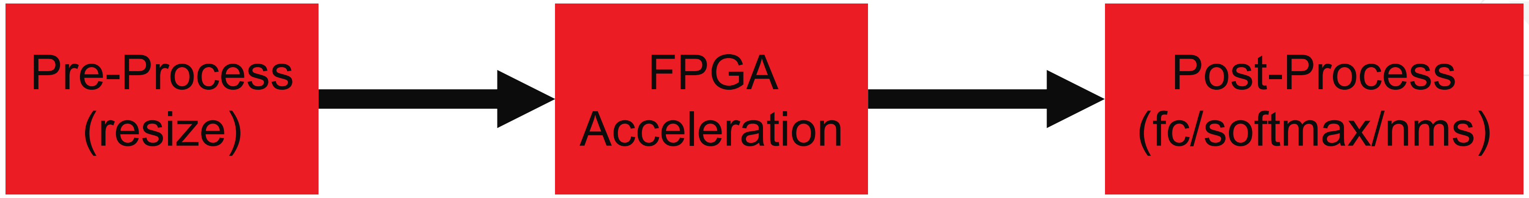
XILINX

Performance Pipelines

> **References to our latest results:**
  - https://github.com/Xilinx/AI-Model-Zoo (embedded i.e. ZC104/Ultra96)
  - https://github.com/Xilinx/ml-suite/blob/master/examples/caffe/Benchmark_README.md
  - Two measurements we track: Latency & Throughput

> **ML pipeline contains multiple stages, performance limited by slowest one**

> **Performance results based on Xilinx own runtime pipeline available in github**
  - (https://github.com/Xilinx/ml-suite/blob/master/examples/deployment_modes/mp_classify.py)

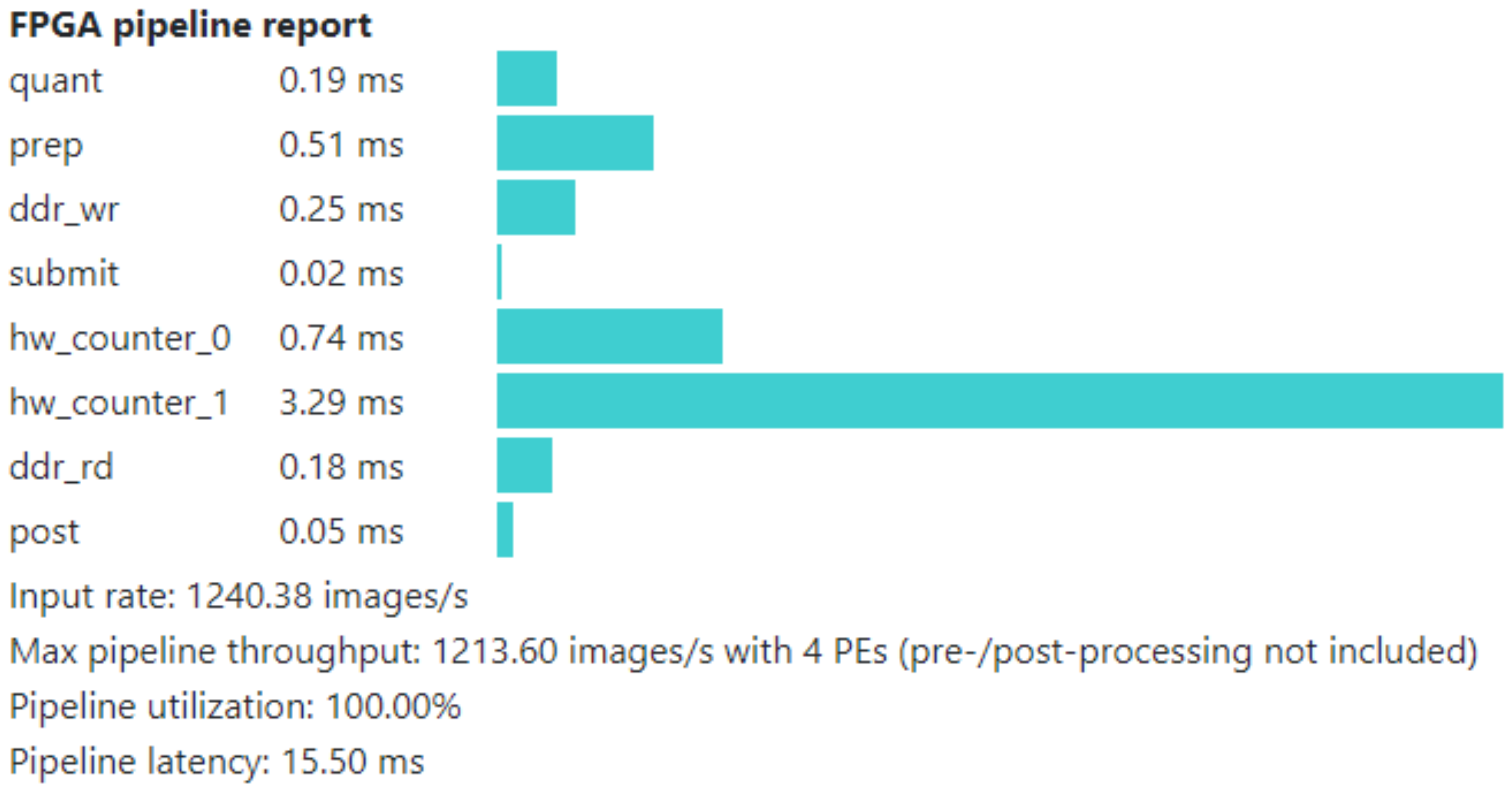


  - Streamlined multi-process pipeline using shared memory
  - Usually need >4 Pre-Process cores running to keep up with FPGA

> **TVM pipeline needed. CPU/FPGA partitions ideally run in parallel**

FPGA Pipeline report in MLSuite 1.5

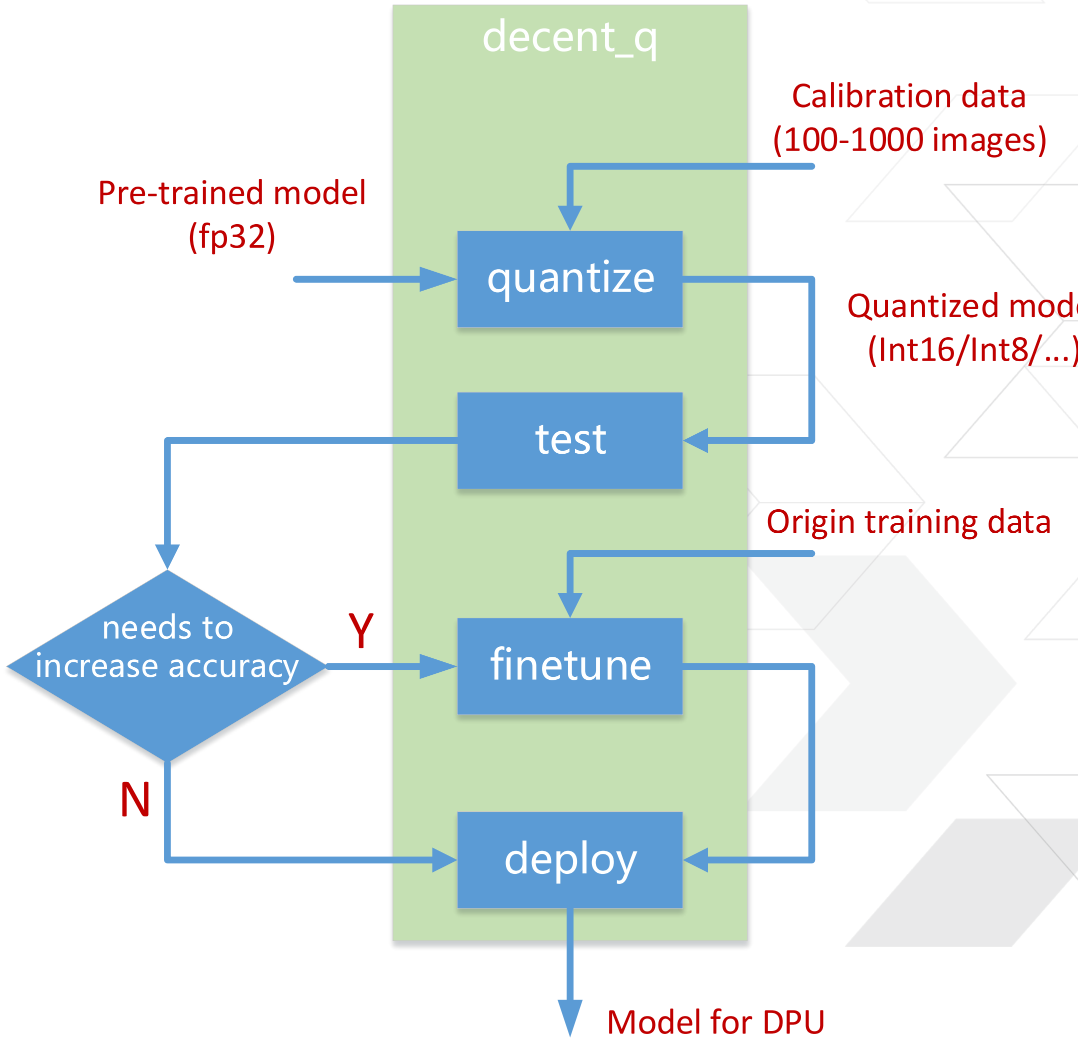
(animated gif of ResNet-50, view in slideshow mode)



```
FPGA pipeline report
quant          0.19 ms
prep           0.51 ms
ddr_wr         0.25 ms
submit         0.02 ms
hw_counter_0   0.74 ms
hw_counter_1   3.29 ms
ddr_rd         0.18 ms
post           0.05 ms
Input rate: 1240.38 images/s
Max pipeline throughput: 1213.60 images/s with 4 PEs (pre-/post-processing not included)
Pipeline utilization: 100.00%
Pipeline latency: 15.50 ms
```

Quantization Tool – vai_q

- **4 commands in vai_q**
  - quantize
    - Quantize network
  - test
    - Test network accuracy
  - finetune
    - Finetune quantized network
  - deploy
    - Generate model for DPU
- **Data**
  - Calibration data
    - Quantize activation
  - Training data
    - Further increase accuracy



Adaptable.
Intelligent.

XILINX®