Xilinx Edge AI Solution

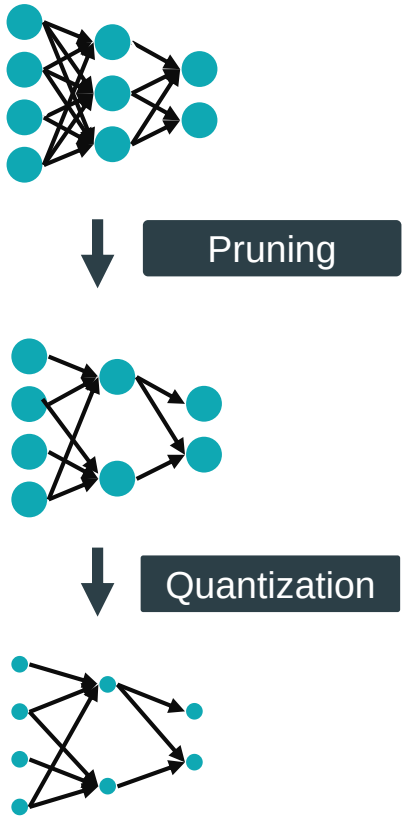
Andy Luo, AI/ML Product Marketing
andy.luo@xilinx.com

Jan 2019

XILINX

EBV Elektronik

Unique, Patented Deep Learning Acceleration Techniques



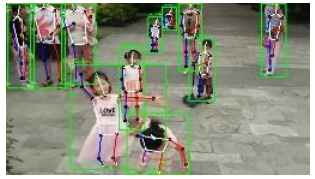
- **Best paper awards for breakthrough DL acceleration**
- **Xilinx's compression technology**
  - Reduce DL accelerator footprint into smaller devices
  - Increase performance per watt (higher performance and/or lower energy)

Unique Pruning Technology Provides a Significant Competitive Advantage

Xilinx Solution Stack for Edge/Embedded ML

Models


Face detection


Pose estimation


Video analytics


Lane detection


Object detection


Segmentation

Framework

Caffe

Darknet

TensorFlow

Tools & IP

DNNDK

DPU

HW Platforms


Z7020 Board


Z7020 SOM


ZU2 SOM


ZU2/3 Card

ZU9 Card


ZCU102


ZCU104

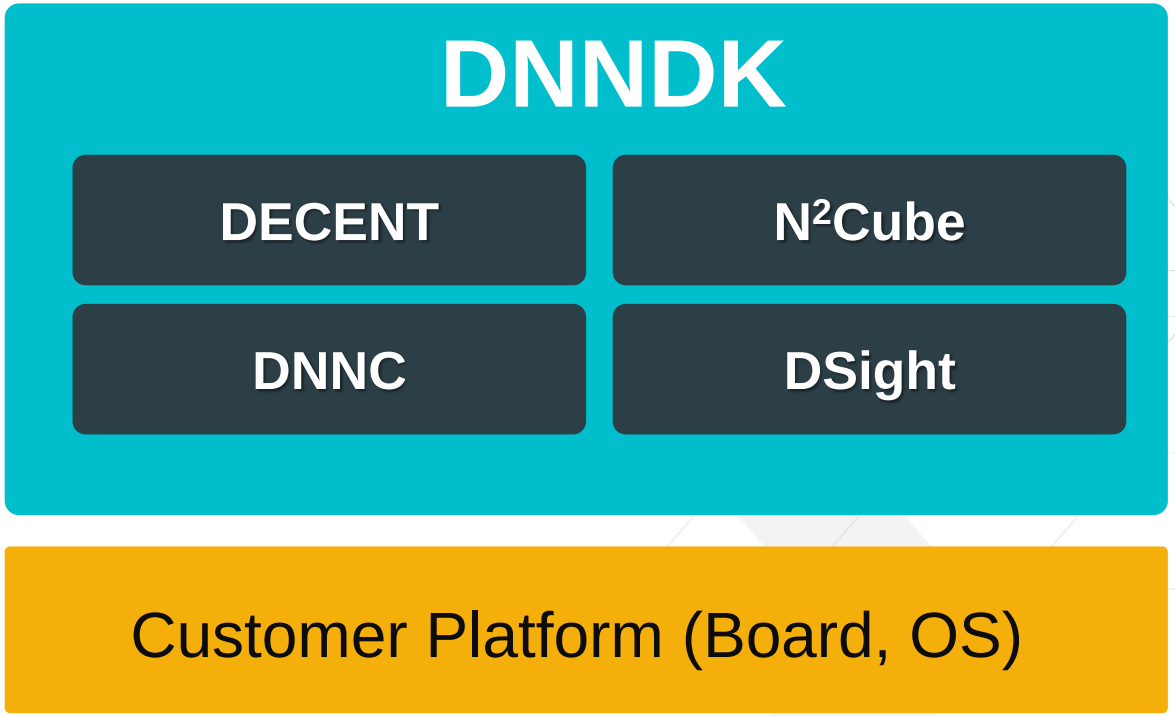

Ultra96

EBV Elektronik

XILINX

DNNDK – Deep Neural Network Development Kit

- DECENT (DEep ComprEssioN Tool)
- DNNC (Deep Neural Network Compiler)
- Runtime N^{2}Cube (Cube of Neural Network)
- Profiler DSight



Framework Support

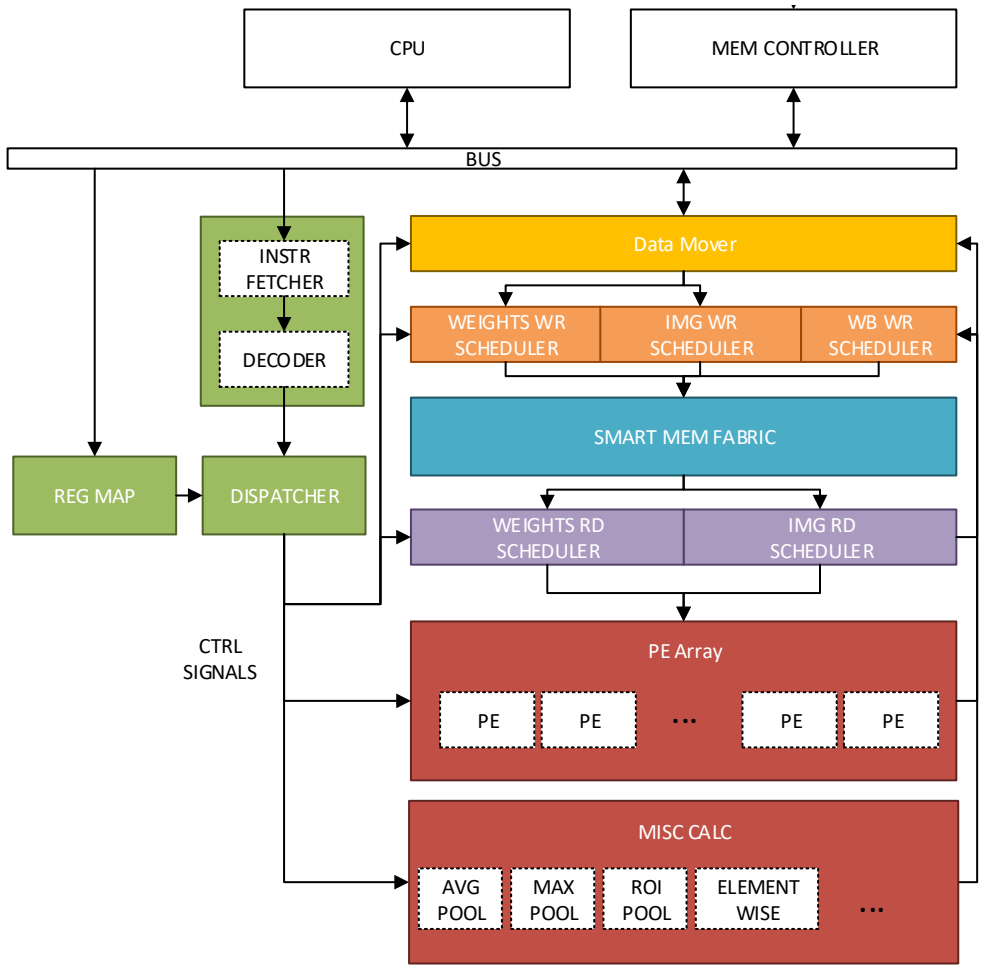
Caffe

- Pruning
- Quantization
- Compilation

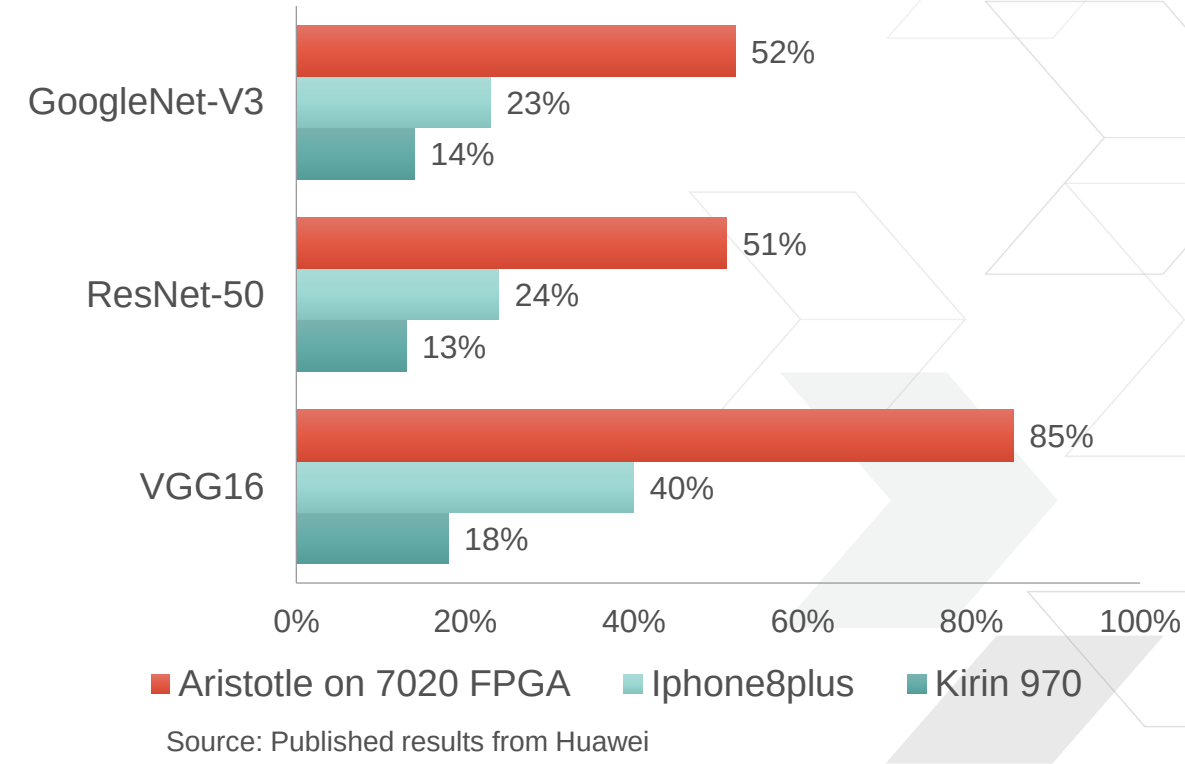
- Pruning
- Quantization
- Convertor to Caffe

- Quantization & Compilation
  - Beta version
- Pruning
  - Beta version

DPU IP with High Efficiency



Utilization > 50% for mainstream neural networks



Source: Published results from Huawei

Supported Operators

- Conv
  - Dilation
- Pooling
  - Max
  - Average
- ReLU / Leaky Relu/ Relu6
- Full Connected (FC)
- Batch Normalization
- Concat
- Elementwise
- Deconv
- Depthwise conv
- Mean scale
- Upsampling
- Split
- Reorg
- Resize (Optional)
- Softmax (Optional)
- Sigmoid (Optional)

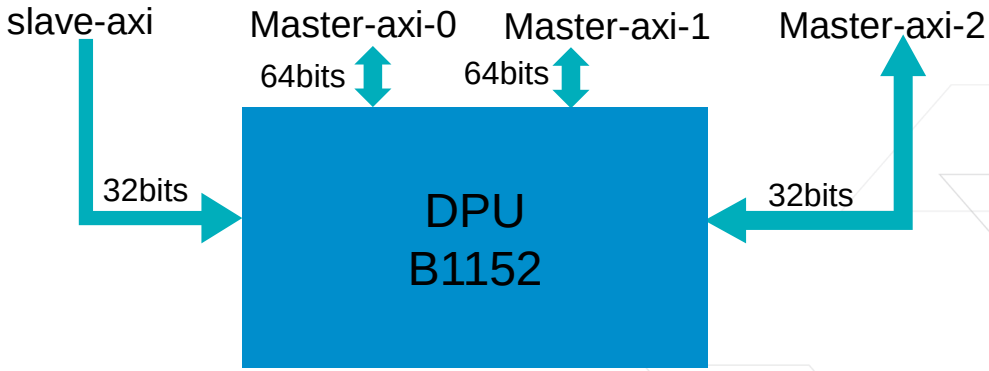
Constraints Between Layers

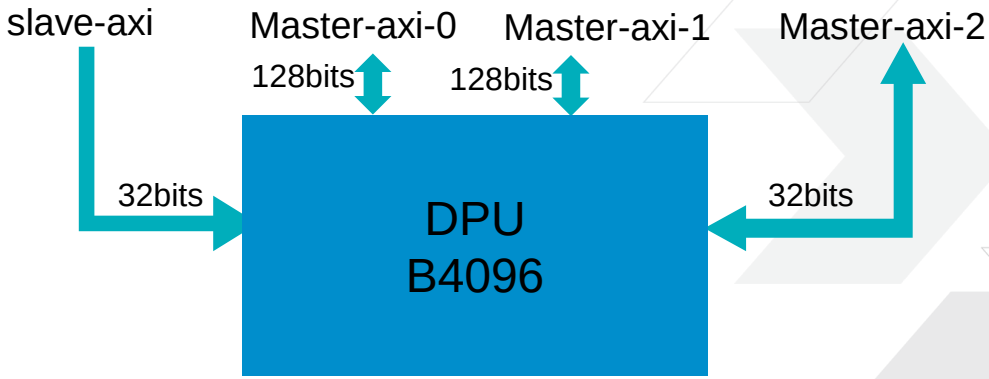
Layer Type \ Next Layer	Conv	Deconv	Depth-wise Conv	Inner Product	Max Pooling	Ave Pooling	BN	ReLU	LeakyReLU	Element-wise	Concat	As Input	As Output
Conv	●	●	○	●	●	○	●	●	○	●	●	●	●
Deconv	●	●	○	●	●	○	●	●	○	●	●	●	●
Depth-wise Conv	●	●	○	●	●	○	●	●	○	●	●	●	●
Inner Product	●	●	○	●	●	○	●	●	○	●	●	●	●
Max Pooling	●	●	○	●	●	○	○	×	×	●	●	●	●
Ave Pooling	○	○	○	○	○	○	○	×	×	○	○	○	○
BN	●	●	○	●	●	○	○	●	×	●	●	○	○
ReLU	●	●	○	●	●	○	○	×	×	●	●	---	●
LeakyReLU	○	○	○	○	○	○	○	×	×	○	○	---	○
Element-wise	●	●	○	●	●	○	○	●	○	●	●	---	●
Concat	●	●	○	●	●	○	○	×	×	●	●	---	●

● : Support　　×: Not support　　○ : Support when selecting additional features

DPU Typical Options & Interfaces

- **3-level parallelism is exploited**
  - Pixel * input channel * output channel
- **Small core - B1152**
  - Parallelism: 4*12*12
  - target Z7020/ZU2/ZU3
- **Big core - B4096**
  - Parallelism: 8*16*16
  - Target ZU5 and above





DPU vs DPU_EU

DPU

- Just include one clock domain
- Instructions: Convolution, Deconvolution, Depthwise Convolution, MaxPool, AveragePool, Elementwise,Softmax,Sigmoid……

DPU_EU/DPU_EU_LP

- Include two clock domains
- Use DSP DDR technique
- Adopt cascade technology to reduce resources
- Use gated clock to reduce power consumption

*DPU_EU_LP in development

DPU_EU Utilization

More DSP

Arch	LUTs	Registers	BRAM*	DSP
B512	17951	28280	69.5	97
B800	20617	35065	87	141
B1024	22327	39000	101.5	193
B1152	22796	40276	117.5	193
B1600	26270	50005	123	281
B2304	29592	57549	161.5	385
B3136	33266	69110	203.5	505
B4096	37495	84157	249.5	641

More LUT

Arch	LUTs	Registers	BRAM*	DSP
B512	20759	33572	69.5	66
B1024	29155	49823	101.5	130
B1152	30043	49588	117.5	146
B1600	33130	60739	123	202
B2304	37055	72850	161.5	290
B3136	41714	86132	203.5	394
B4096	44583	99791	249.5	514

DPU provides flexible option depending on costumer's resources and continues to improve

** URAM also can be used by DPU if device supports, every URAM is roughly used as 3.7 BRAM*

DPU_EU Utilization

LeakyRelu not enabled

Arch	LUTs	Registers	BRAM*	DSP
B512	17951	28280	69.5	97
B800	20617	35065	87	141
B1024	22327	39000	101.5	193
B1152	22796	40276	117.5	193
B1600	26270	50005	123	281
B2304	29592	57549	161.5	385
B3136	33266	69110	203.5	505
B4096	37495	84157	249.5	641

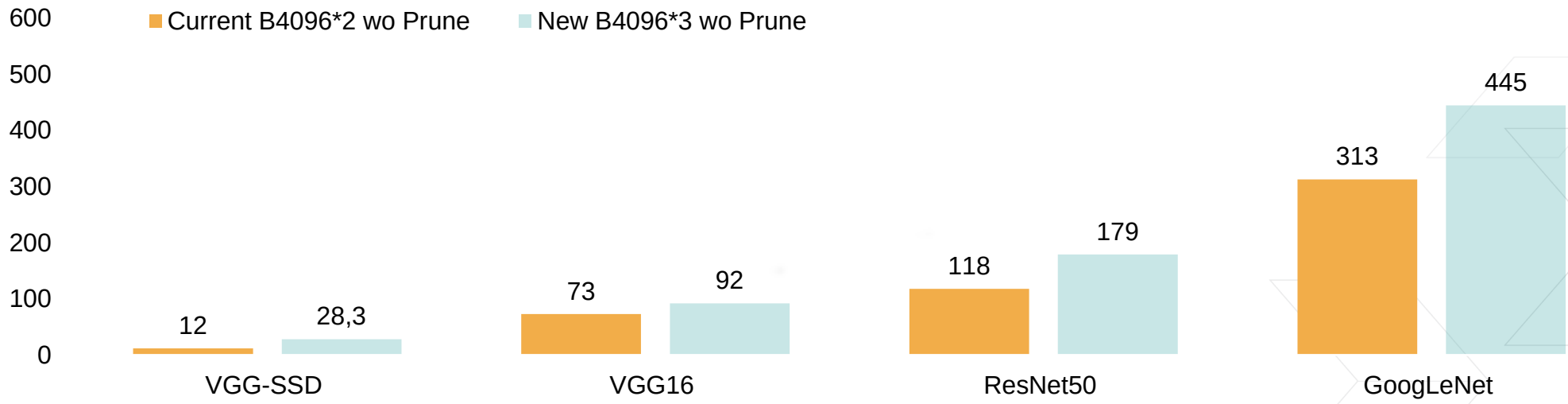
LeakyRelu enabled

Arch	LUTs	Registers	BRAM*	DSP
B512	18371	28292	69.5	97
B800	21162	35079	87	141
B1024	22759	39012	101.5	193
B1152	23453	40292	117.5	193
B1600	26817	50019	123	281
B2304	30268	57565	161.5	385
B3136	34032	69125	203.5	505
B4096	38392	84173	249.5	641

** URAM also can be used by DPU if device supports, every URAM is roughly used as 3.7 BRAM*

Perf Improvement with DPU_EU

Performance Comparison (FPS)

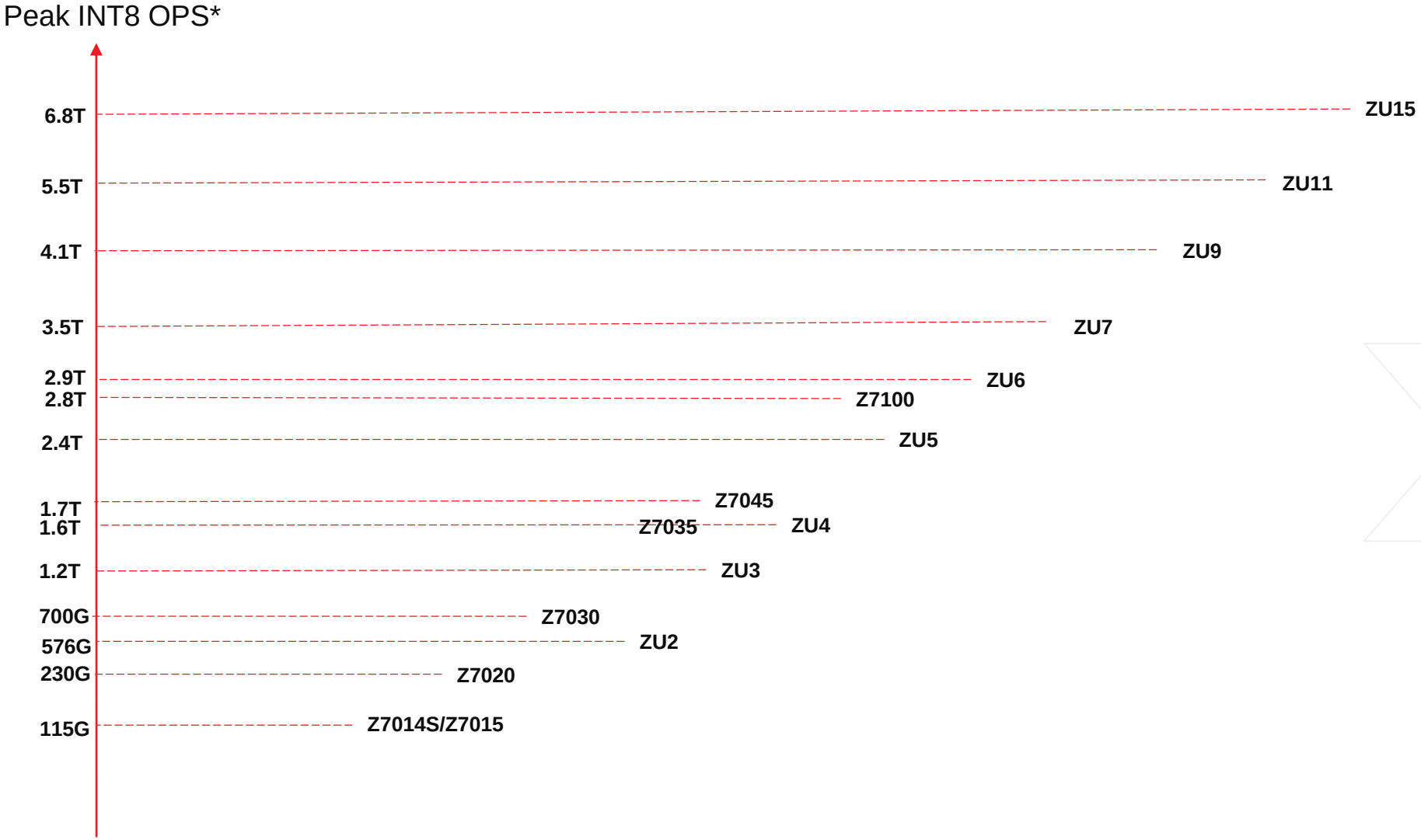


*The FPS of VGG-SSD of end to end performance
*The FPS of VGG16/ResNet50/GoogLeNet is of CONV part (w/o FC layer)

Resource Utilization Comparison

	DSP	LUT	FF	BRAM
DPU B4096*2	2048	156744	224650	501
DPU_EU B4096*3	1926	110311	255020	748.5

DPU Scalability



** With heterogenous DPUs*

DNNDK Dev Flow

Five Steps with DNNDK

01 Model Compression

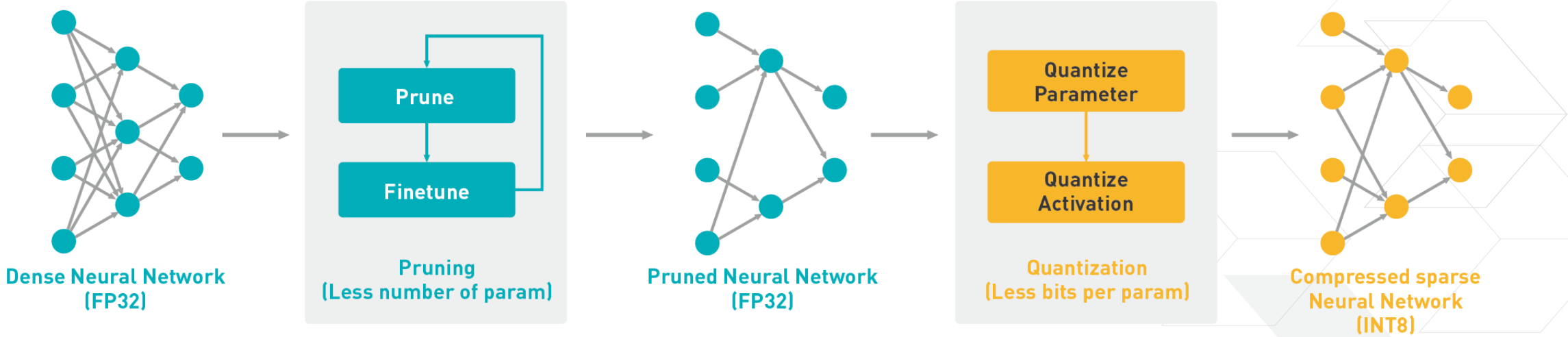
02 Model Compilation

03 Programming

04 Hybrid Compilation

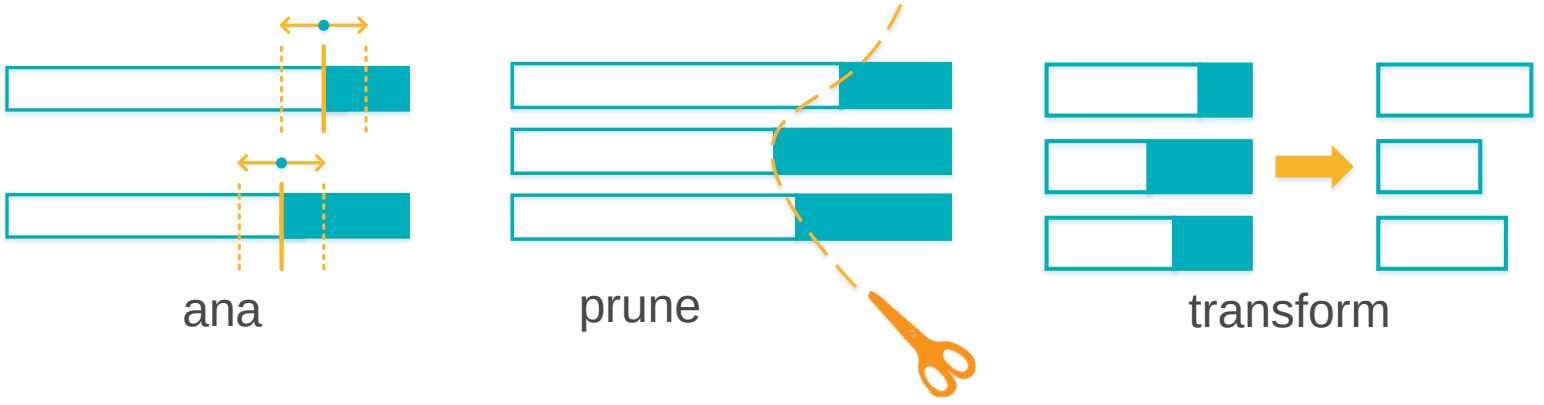
05 Execution

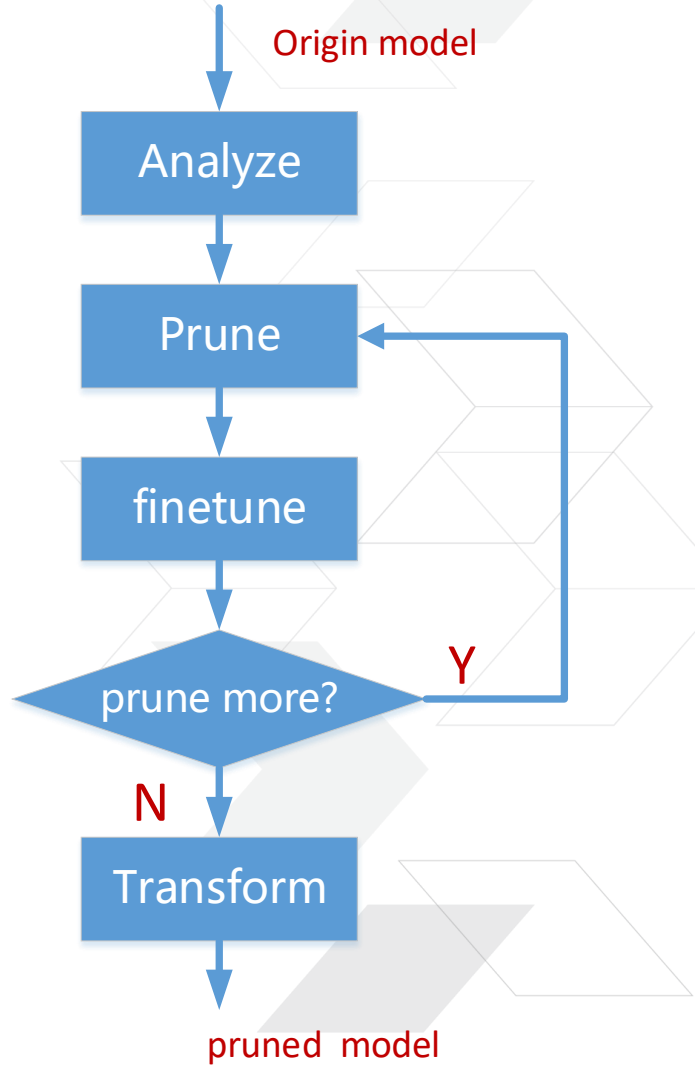
DECENT – Xilinx Deep Compression Tool



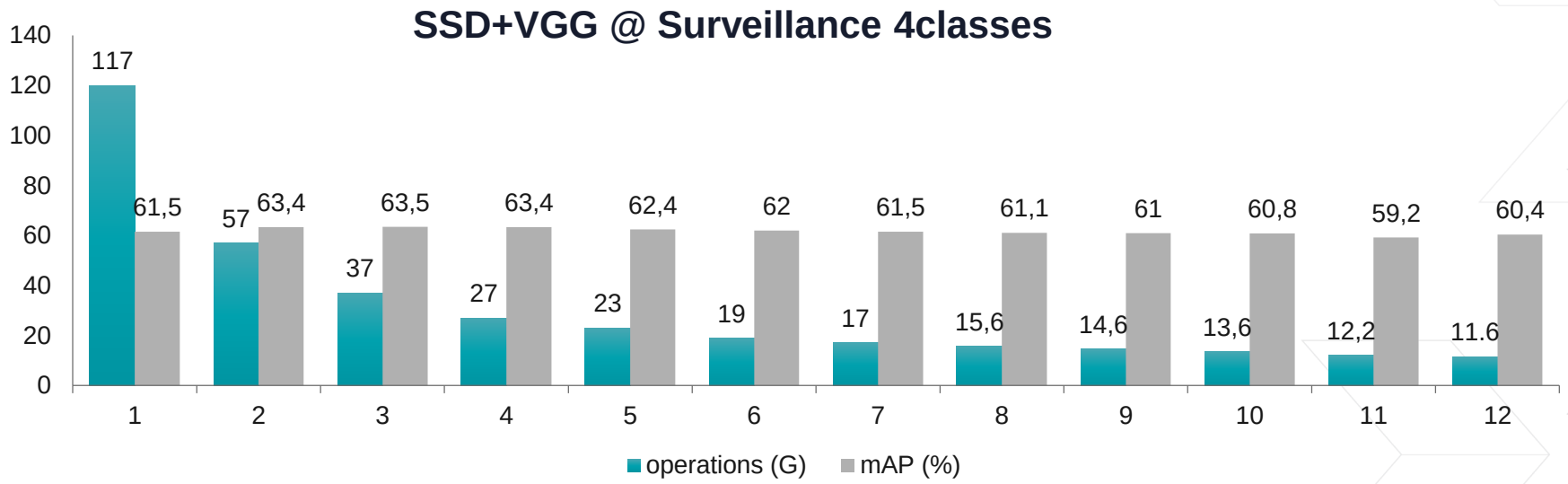
Pruning Tool – decent_p

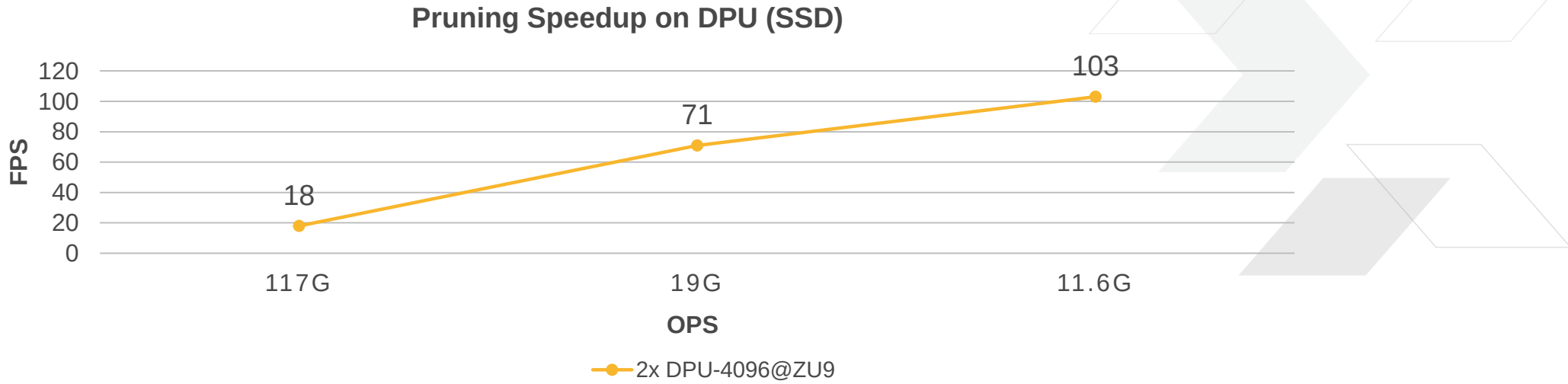
- **4 commands in decent_p**
  - Ana
    - analyze the network
  - Prune
    - prune the network according to config
  - Finetune
    - finetune the network to recover accuracy
  - Transform
    - transform the pruned model to regular model



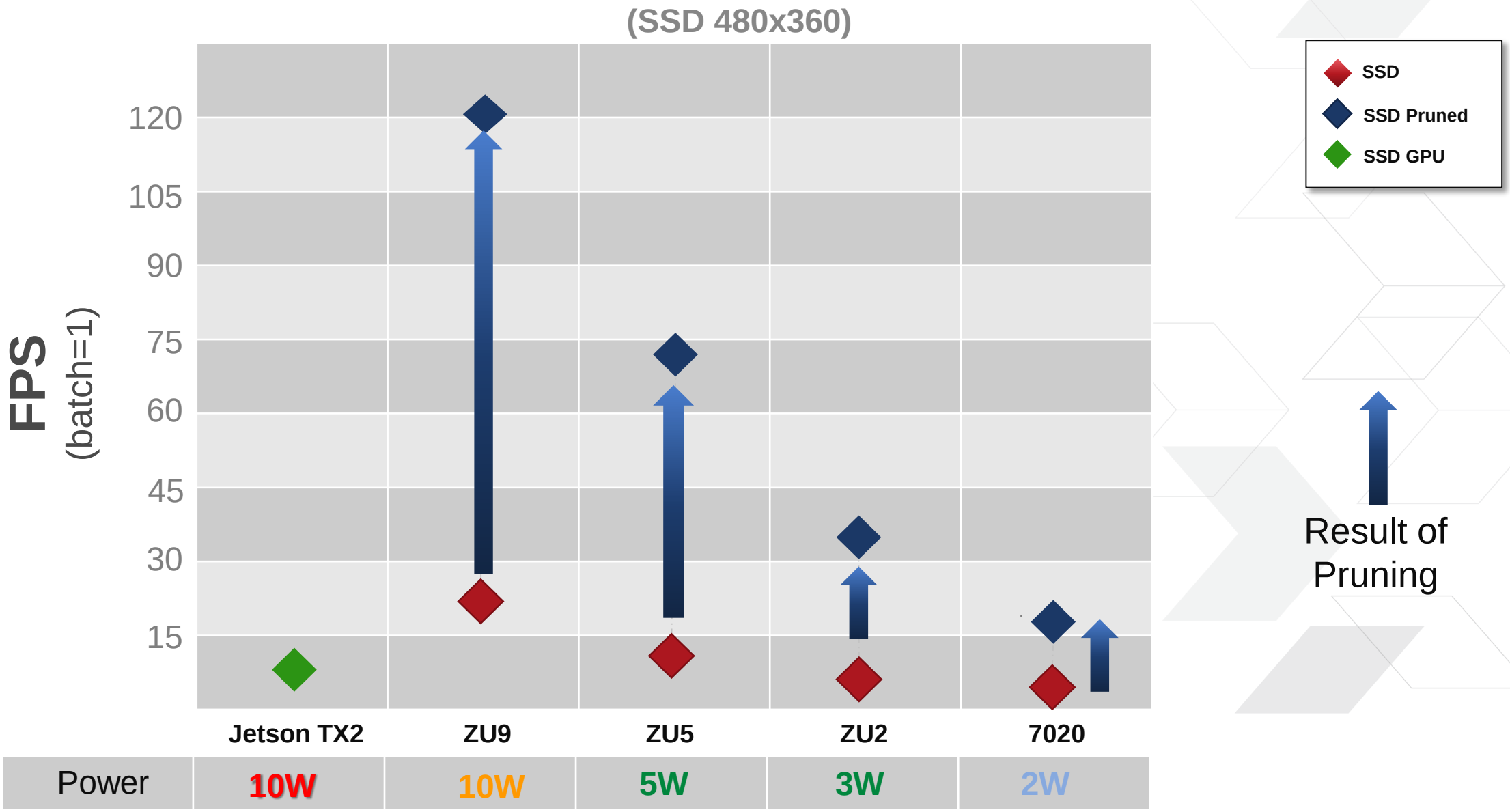


Pruning Example - SSD





Pruning Makes Big Difference



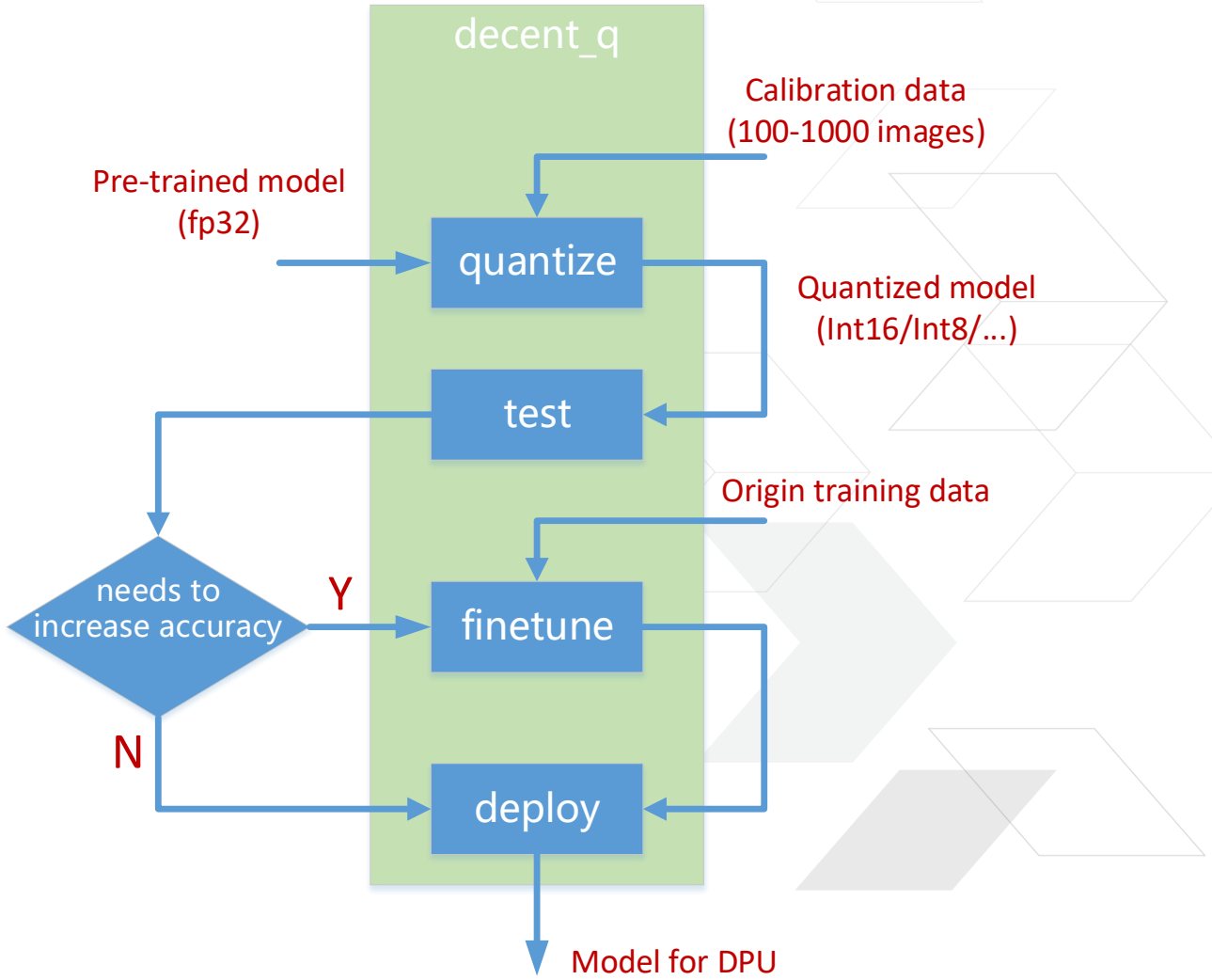
Pruning Results

Classification Networks	Baseline	Pruning Result 1			Pruning Result 2		
	Top-5	Top-5	ΔTop5	ratio	Top-5	ΔTop5	ratio
Resnet50 [7.7G]	91.65%	91.23%	-0.42%	40%	90.79%	-0.86%	32%
Inception_v2 [4.0G]	91.07%	90.37%	-0.70%	60%	90.07%	-1.00%	55%
SqueezeNet [778M]	83.19%	82.46%	-0.73%	89%	81.57%	-1.62%	75%

Detection Networks	Baseline mAP	Pruning Result 1			Pruning Result 2		
		mAP	ΔmAP	ratio	mAP	ΔmAP	ratio
DetectNet [17.5G]	44.46	45.7	+1.24	63%	45.12	+0.66	50%
SSD+VGG [117G]	61.5	62.0	+0.5	16%	60.4	-1.1	10%
[A] SSD+VGG [173G]	57.1	58.7	+1.6	40%	56.6	-0.5	12%
[B] Yolov2 [198G]	80.4	81.9	+1.5	28%	79.2	-1.2	7%

Quantization Tool – decent_q

- **4 commands in decent_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



Quantization Results

- **Uniform Quantization**
  - 8-bit for both weights and activation
  - A small set of images for calibration

Networks	Float32 baseline		8-bit Quantization			
	Top1	Top5	Top1	ΔTop1	Top5	ΔTop5
Inception_v1	66.90%	87.68%	66.62%	-0.28%	87.58%	-0.10%
Inception_v2	72.78%	91.04%	72.40%	-0.38%	90.82%	-0.23%
Inception_v3	77.01%	93.29%	76.56%	-0.45%	93.00%	-0.29%
Inception_v4	79.74%	94.80%	79.42%	-0.32%	94.64%	-0.16%
ResNet-50	74.76%	92.09%	74.59%	-0.17%	91.95%	-0.14%
VGG16	70.97%	89.85%	70.77%	-0.20%	89.76%	-0.09%
Inception-ResNet-v2	79.95%	95.13%	79.45%	-0.51%	94.97%	-0.16%

DNNDK API

dpuOpen()
dpuClose()
dpuLoadKernel()
dpuDestroyKernel()
dpuCreateTask()
dpuRunTask()
dpuDestroyTask()
dpuEnableTaskProfile()
dpuGetTaskProfile()
dpuGetNodeProfile()
dpuGetInputTensor()
dpuGetInputTensorAddress()
dpuGetInputTensorSize()
dpuGetInputTensorScale()
dpuGetInputTensorHeight()
dpuGetInputTensorWidth()
dpuGetInputTensorChannel()
dpuGetOutputTensor()
dpuGetOutputTensorAddress()

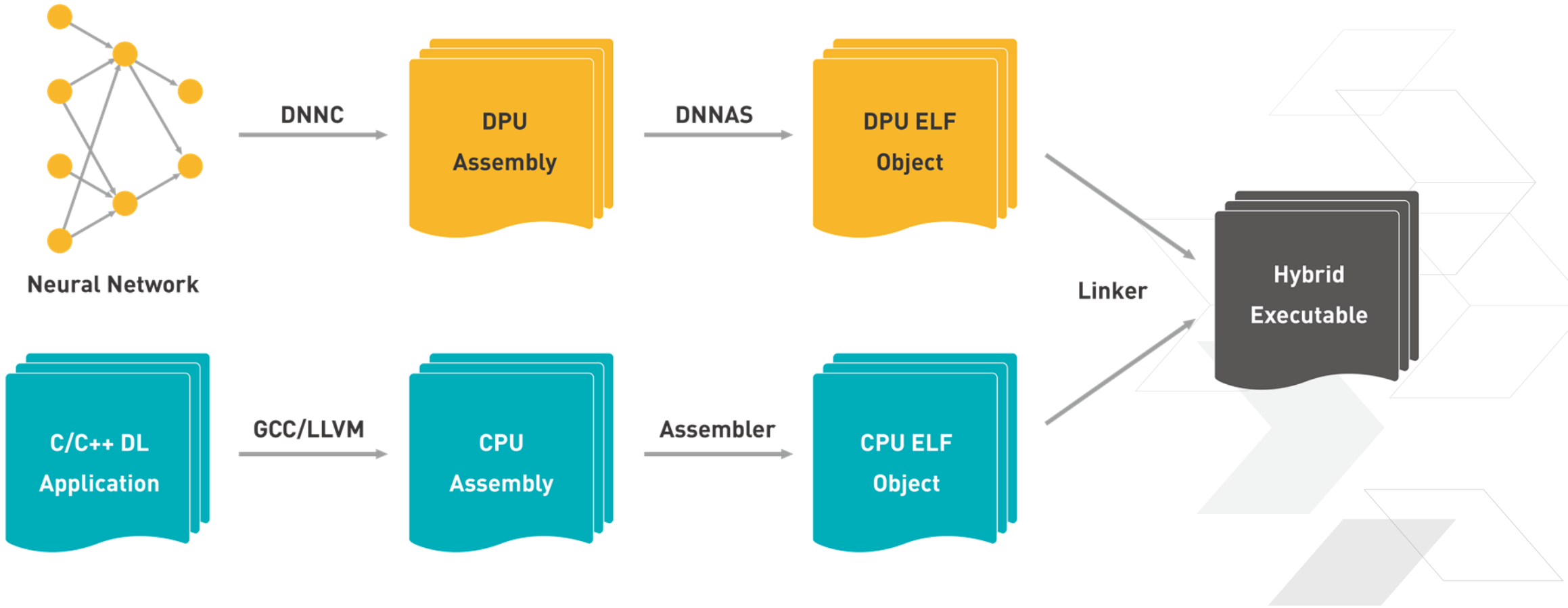
dpuGetOutputTensorSize()
dpuGetOutputTensorScale()
dpuGetOutputTensorHeight()
dpuGetOutputTensorWidth()
dpuGetOutputTensorChannel()
dpuGetTensorSize()
dpuGetTensorAddress()
dpuGetTensorScale()
dpuGetTensorHeight()
dpuGetTensorWidth()
dpuGetTensorChannel()
dpuSetIntputTensorInCHWInt8()
dpuSetIntputTensorInCHWFP32()
dpuSetIntputTensorInHWCInt8()
dpuSetIntputTensorInHWCFP32()
dpuGetOutputTensorInCHWInt8()
dpuGetOutputTensorInCHWFP32()
dpuGetOutputTensorInHWCInt8()
dpuGetOutputTensorInHWCFP32()

- **High-level Tensor-based APIs**
- **Please refer to DNNDK User Guide**

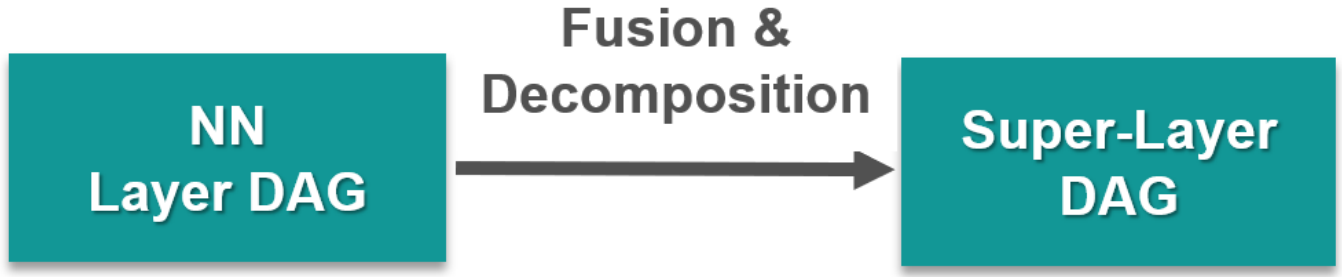
Programming with DNNDK API

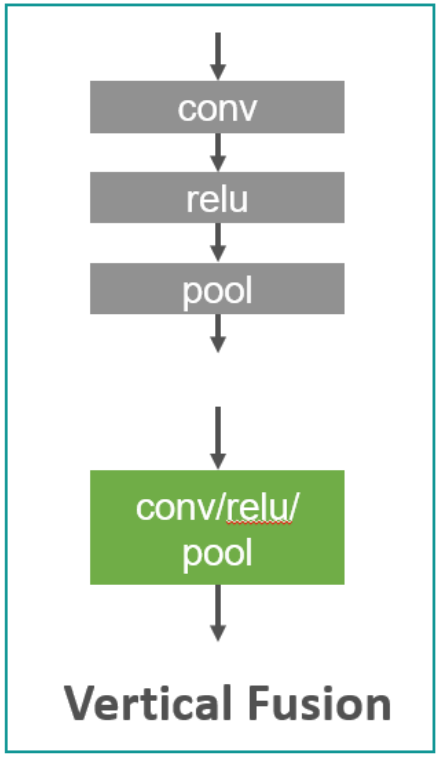
```c
int main(int argc, char *argv[])
{
    DPUKernel *kernel_conv;
    DPUKernel *kernel_fc;
    DPUTask   *task_conv;
    DPUTask   *task_fc;
    char      *input_addr;
    char      *output_addr;

    /* DNNDK API to attach to DPU driver */
    dpuInit();

    /* DNNDK API to create DPU kernels for CONV & FC networks */
    kernel_conv = dpuLoadKernel("resnet50_conv", 224, 224);
    kernel_fc = dpuLoadKernel("resnet50_fc", 1, 1);

    /* Create tasks from CONV & FC kernels */
    task_conv = dpuCreateTask(kernel_conv);
    task_fc = dpuCreateTask(kernel_fc);

    /* Set input tensor for CONV task and run */
    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_conv));
    setInputImage(Mat &image, input_addr);
    dpuRunTask(task_conv);
    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_conv));

    /* Run average pooling layer on CPU */
    run_average_pooling(output_addr);

    /* Set input tensor for FC task and run */
    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_fc));
    setFCInputData(task_fc, input_addr);
    dpuRunTask(task_fc);
    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_fc));

    /* Diaplay the Classification result from FC task */
    displayClassificationResult(output_addr);

    /* DNNDK API to destroy DPU tasks/kernels */
    dpuDestroyTask(task_conv);
    dpuDestroyTask(task_fc);

    dpuDestroyKernel(kernel_conv);
    dpuDestroyKernel(kernel_fc);

    /* DNNDK API to dettach from DPU driver and free DPU resources */
    dpuFini();

    return 0;
}
```

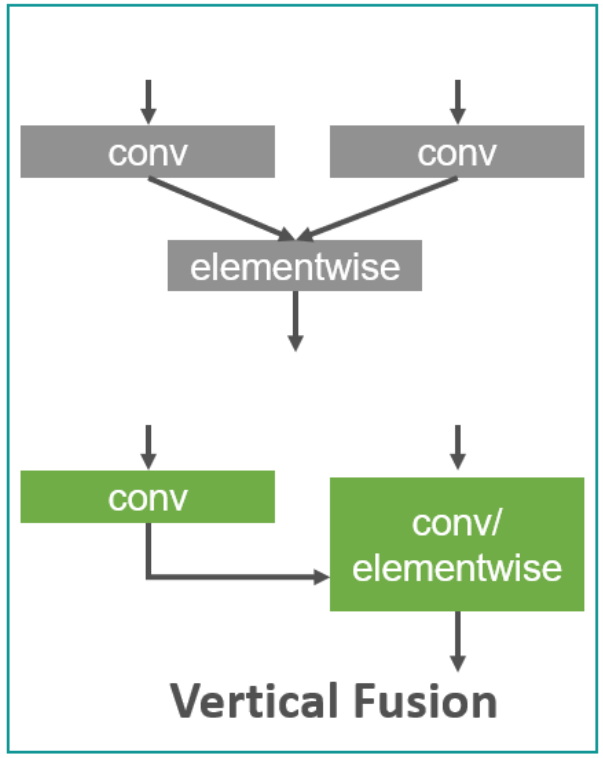
DNNDK Hybrid Compilation Model

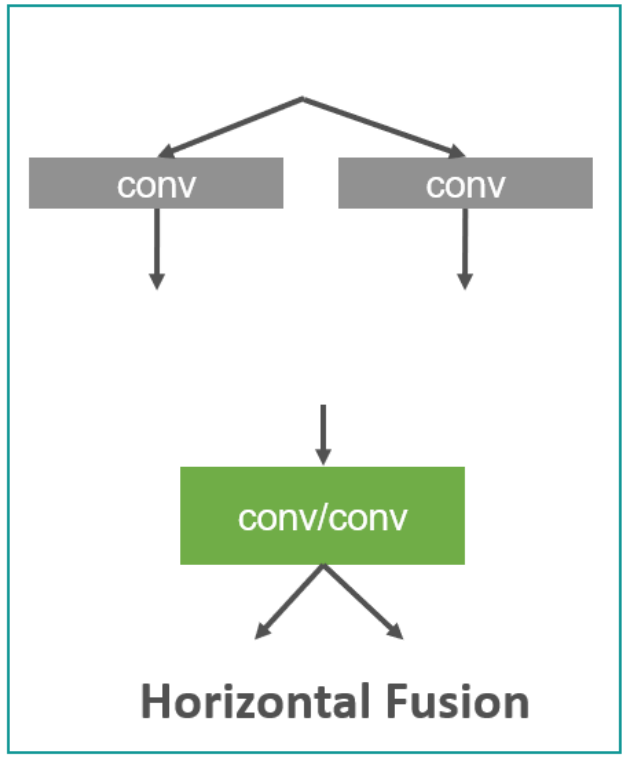


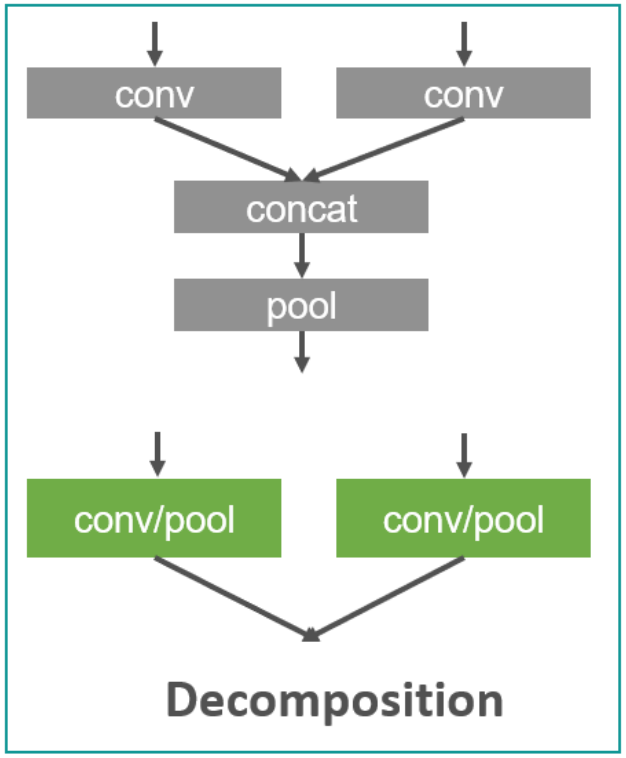
Optimization in DNNC



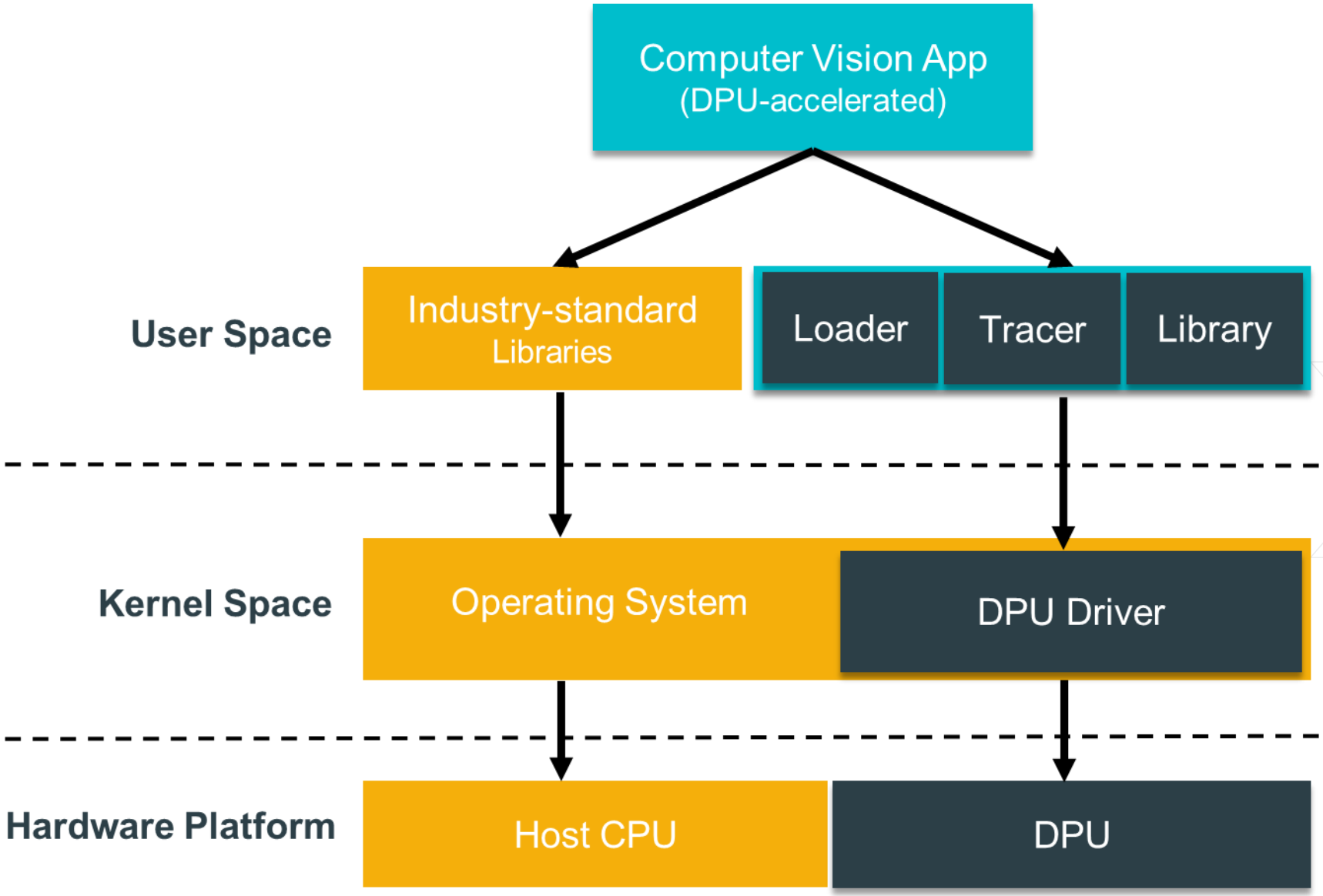








DNNDK Runtime Engine



Runtime N²Cube

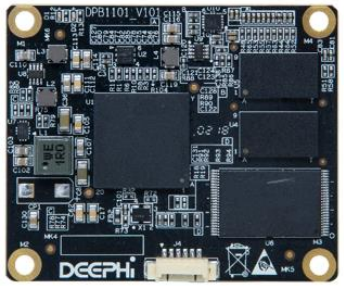
- Library
- Loader
- Tracer
- Driver

Supported Networks

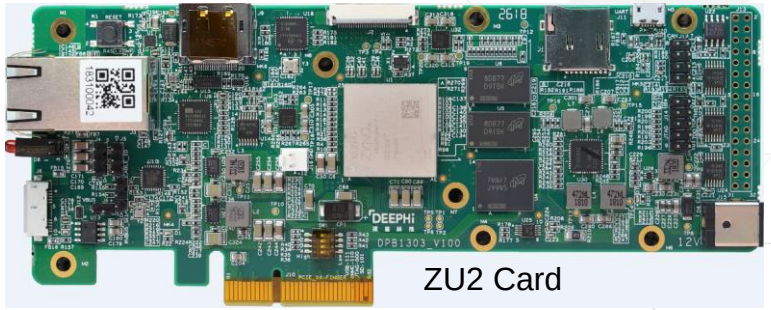
Application	Module	Algorithm	Model Development	Compression	Deployment
Face	Face detection	SSD, Densebox	✓	✓	✓
	Landmark Localization	Coordinates Regression	✓	N / A	✓
	Face recognition	ResNet + Triplet / A-softmax Loss	✓	✓	✓
	Face attributes recognition	Classification and regression	✓	N / A	✓
Pedestrian	Pedestrian Detection	SSD	✓	✓	✓
	Pose Estimation	Coordinates Regression	✓	✓	✓
	Person Re-identification	ResNet + Loss Fusion	✓		
Video Analytics	Object detection	SSD, RefineDet	✓	✓	✓
	Pedestrian Attributes Recognition	GoogleNet	✓	✓	✓
	Car Attributes Recognition	GoogleNet	✓	✓	✓
	Car Logo Detection	DenseBox	✓	✓	
	Car Logo Recognition	GoogleNet + Loss Fusion	✓	✓	
	License Plate Detection	Modified DenseBox	✓	✓	✓
	License Plate Recognition	GoogleNet + Multi-task Learning	✓	✓	✓
ADAS/AD	Object Detection	SSD, YOLOv2, YOLOv3	✓	✓	✓
	3D Car Detection	F-PointNet, AVOD-FPN	✓		
	Lane Detection	VPGNet	✓	✓	✓
	Traffic Sign Detection	Modified SSD	✓		
	Semantic Segmentation	FPN	✓	✓	✓
	Drivable Space Detection	MobilenetV2-FPN	✓		
	Multi-task (Detection+Segmentation)	Xilinx	✓		

Out-of-box Supported Boards

- ZCU102
- ZCU104
- Avnet Ultra96
- Z7020 SOM
- ZU2 PCIe board
- ZU2 SOM
- ZU9 PCIe Card



Z7020 SOM

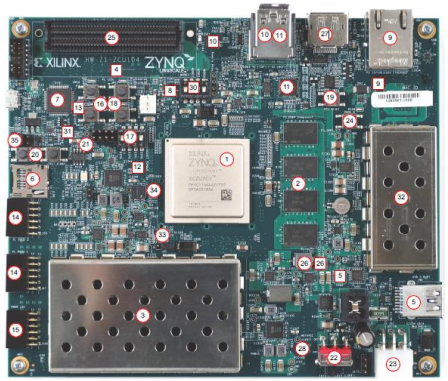


ZU2 Card

ZU2 SOM

ZU9 Card

Ultra96

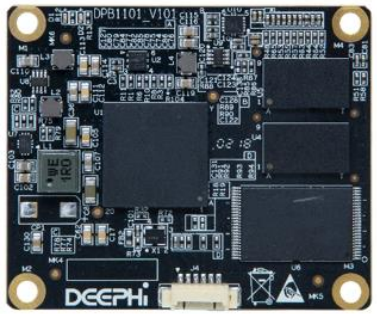


ZCU104


ZCU102

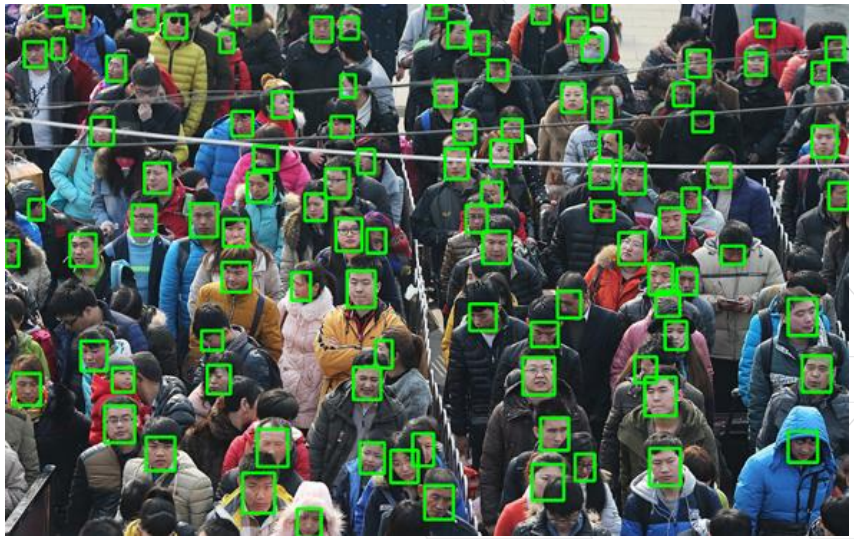
XILINX

Video Surveillance ML Solutions

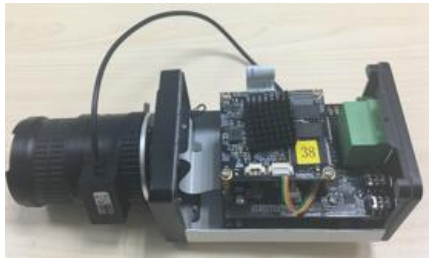


Intelligent
IP Camera Solution

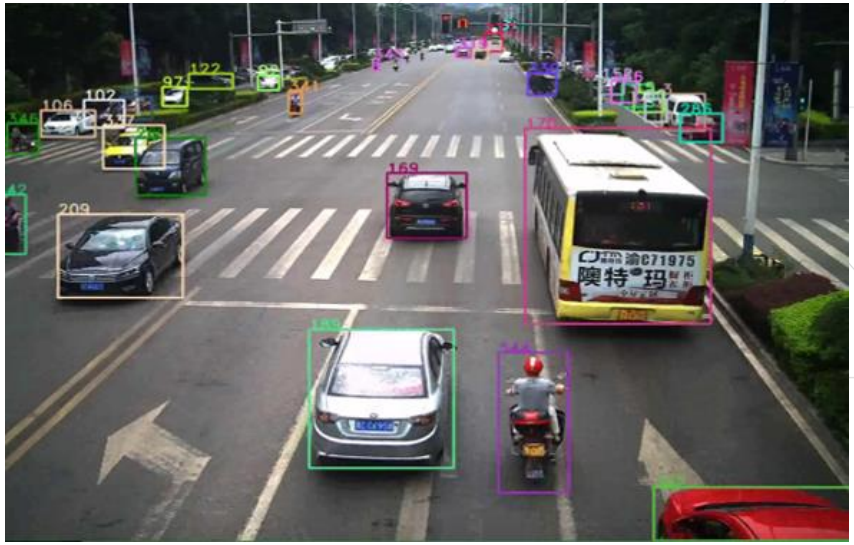
Video Analytics
Acceleration Solution



Face recognition camera
with Zynq7020



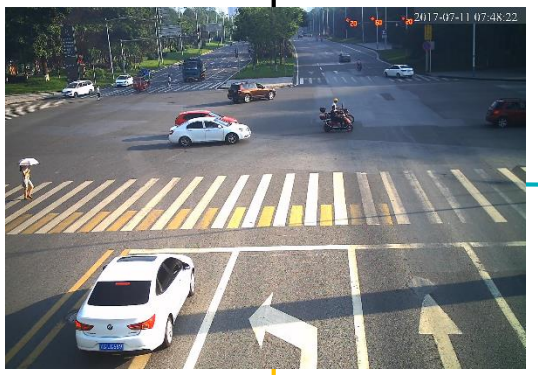
12-channel 1080P Video Analytics
with ZU9EG



EBV Elektronik

XILINX

Video Surveillance ML Ref Design



Detection & Tracking

Person Attributes

Gender : Female
Upper color : Yellow
Lower color : White
Hat : No
Backpack : No
Handbag : No
Other bag : No

Detection & Tracking

Person Attributes

Gender : Male
Upper color : Black
Lower color : Black
Hat : No
Backpack : No
Handbag : No
Other bag : No

Detection & Tracking

Car Attributes

Color : White
Type : BUICK

Plate Detection

License Recognition

Color : Blue
Number :渝C LC689

EBV Elektronik

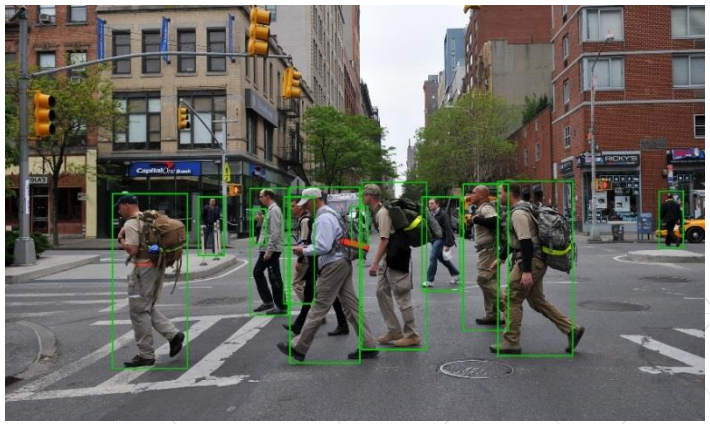
XILINX

ADAS/AD ML Reference Design

2D/3D Object Detection

Lane Detection

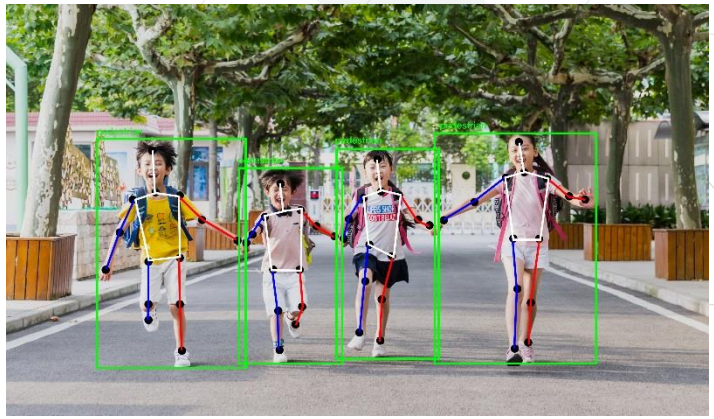
Pedestrian Detection



Segmentation + Detection

Segmentation

Pose Estimation



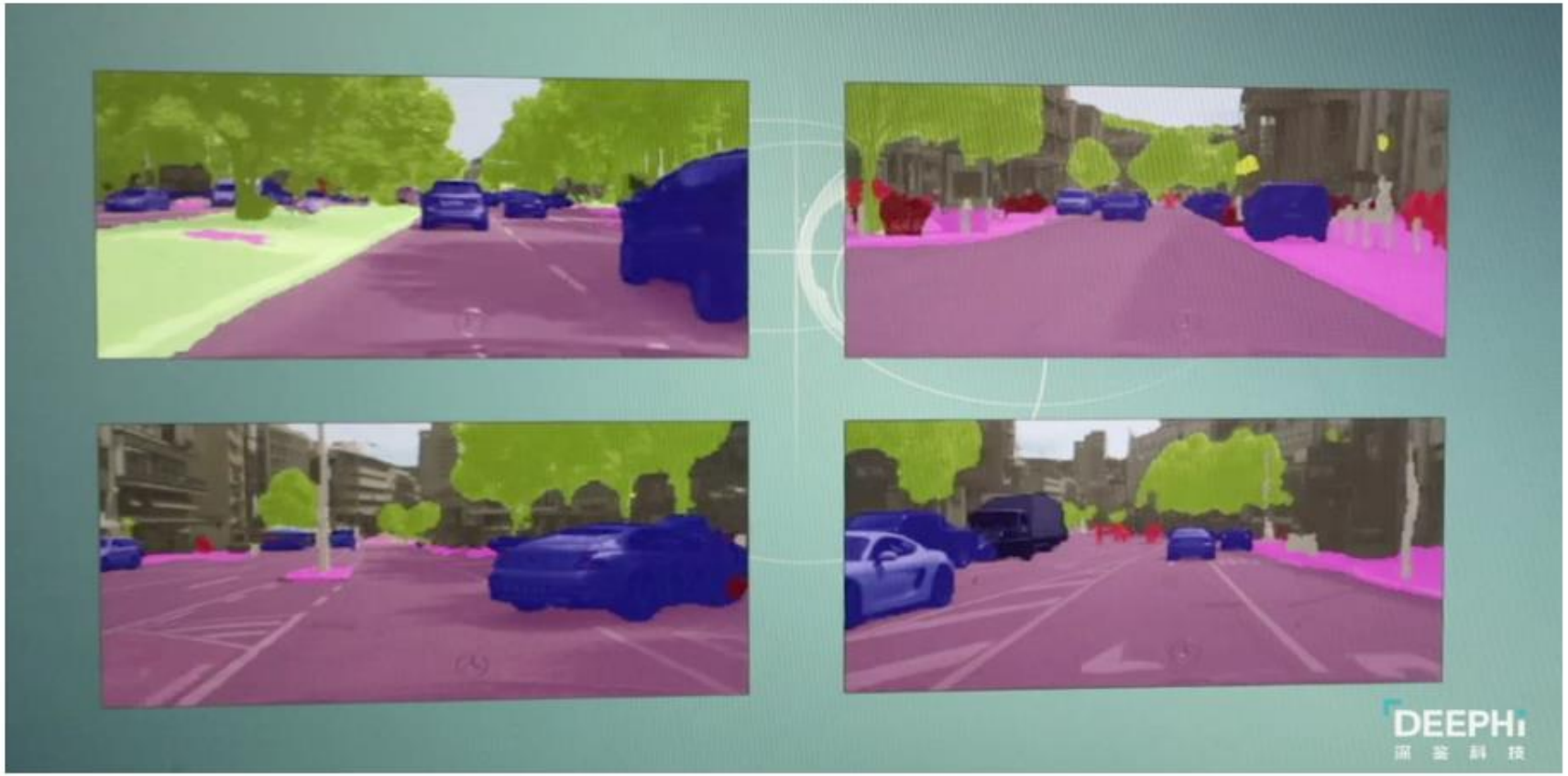
8CH Detection Demo

- Xilinx device
  - ZU9EG
- Network
  - SSD compact version
- Input image size to DPU
  - 480 * 360
- Operations per frame
  - 4.9G
- Performance
  - 30fps per channel

*Removed Video

4-ch Segmentation + Detection Demo

- Xilinx device
  - ZU9EG
- Network
  - FPN compact version
  - SSD compact version
- Input image size to DPU
  - FPN – 512 * 256
  - SSD – 480 * 360
- Operations per frame
  - FPN – 9G
  - SSD – 4.9G
- Performance
  - 15fps per channel



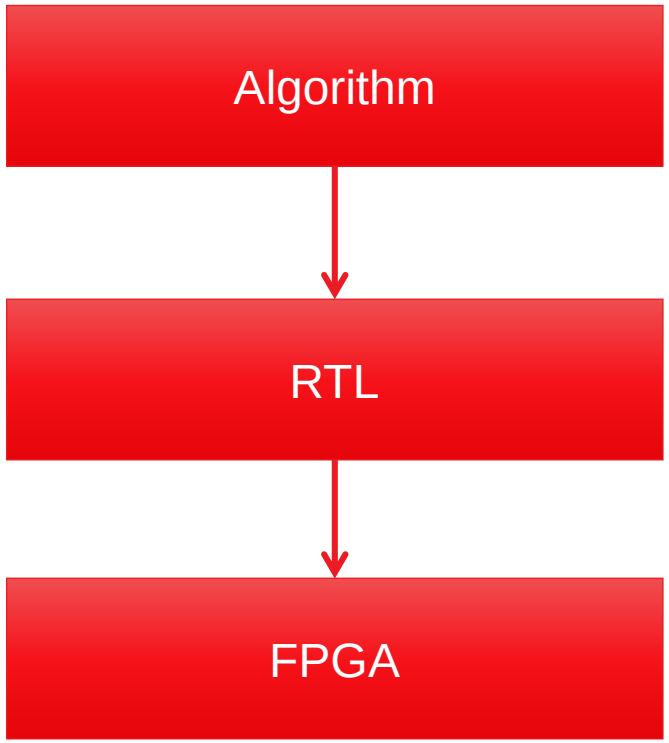
*Removed Video

ML Development with DPU/DNNDK

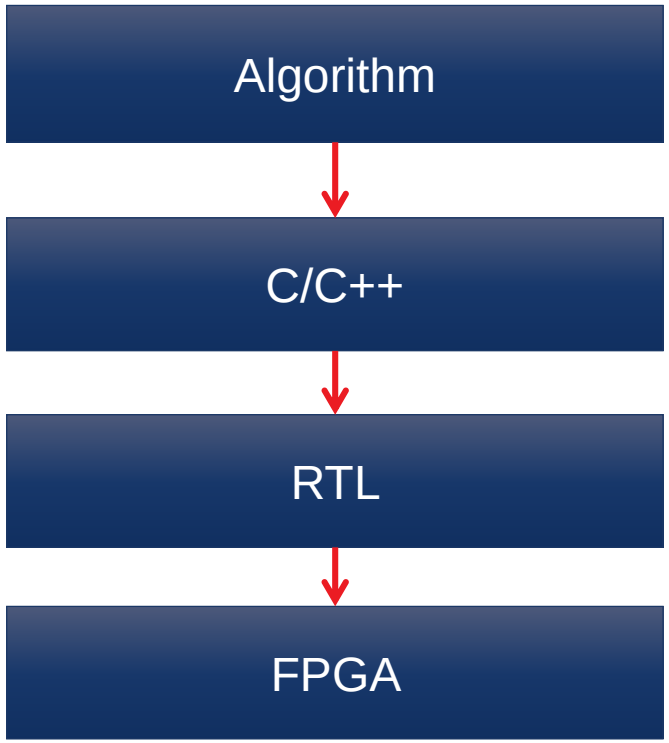
XILINX

EBV Elektronik

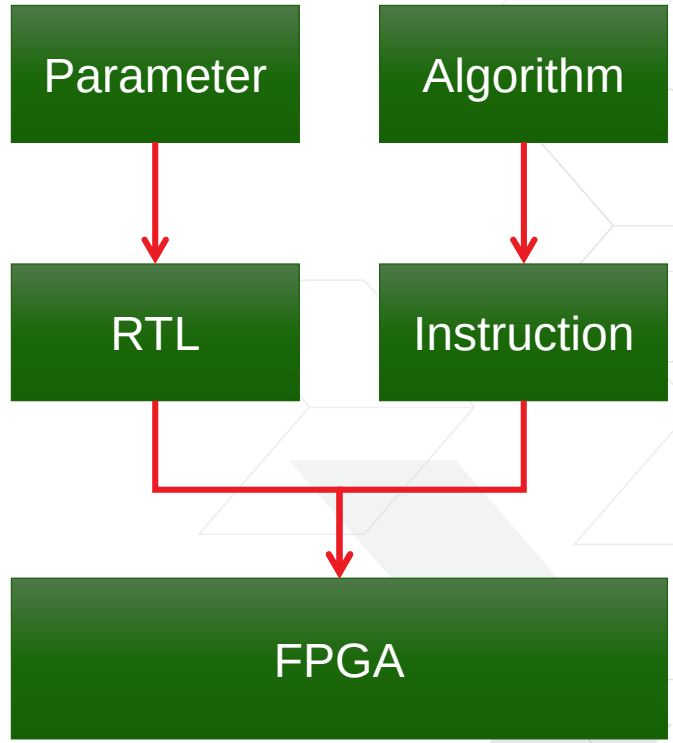
Development Method



Traditional



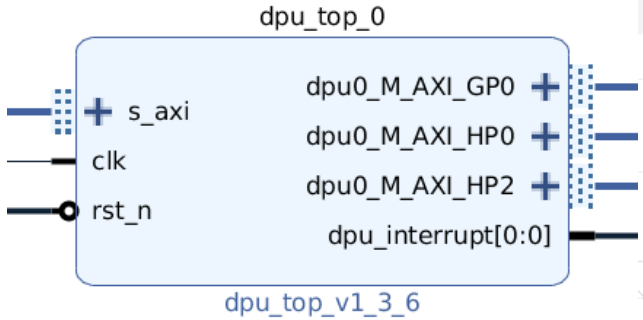
OpenCL/HLS

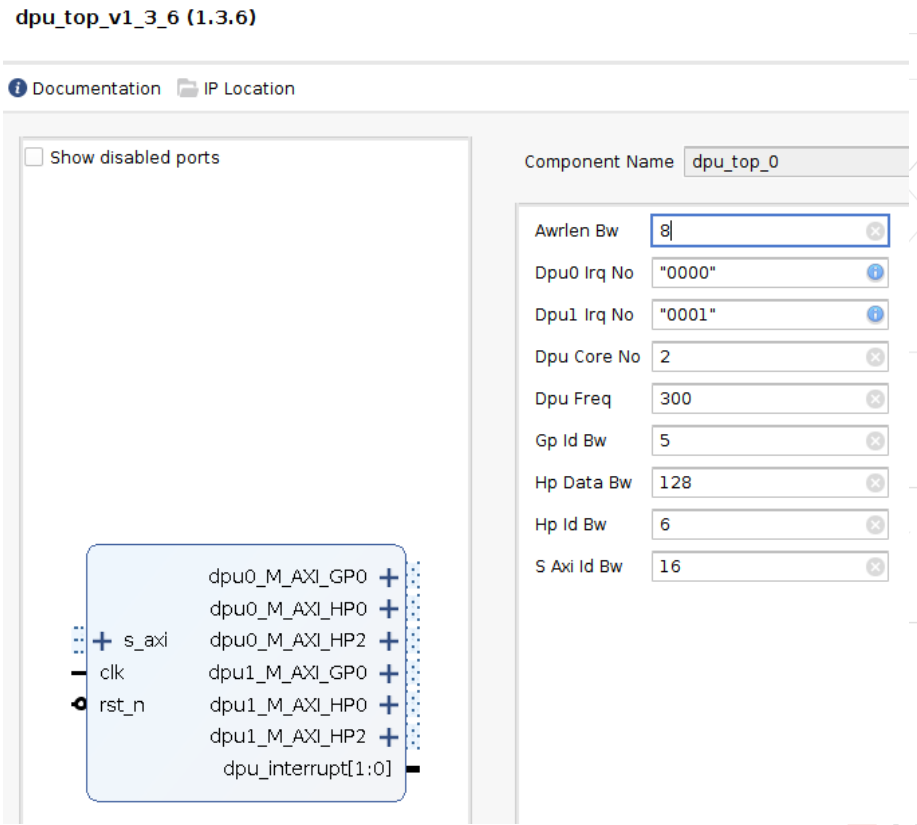


Xilinx

HW Integration with Vivado IPI

- **Add DPU IP into repository**
- **Add DPU into block design**
- **Configure DPU parameters**
- **Connect DPU with MPSoC(for reference)**
  - M_AXI_HP0 <-> S_AXI_HP0_FPD (ZYNQ)
  - M_AXI_HP2 <-> S_AXI_HP1_FPD (ZYNQ)
  - M_Axi_GP0 <-> S_AXI_LPD(ZYNQ)
  - s_axi <-> M_AXI_HPM0_LPD (ZYNQ)
- **Assign Reg address for DPU in address editor**
  - e.g. 0x80000000, 4K space for one DPU
- **Create top wrapper**
- **Generate bitstream**
- **Generate BOOT.BIN using Petalinux etc.**

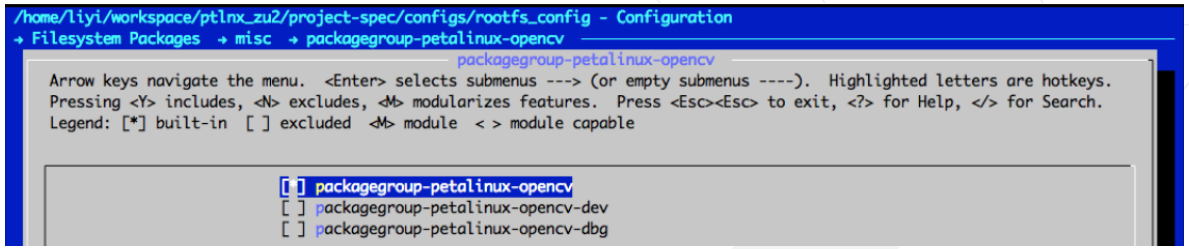


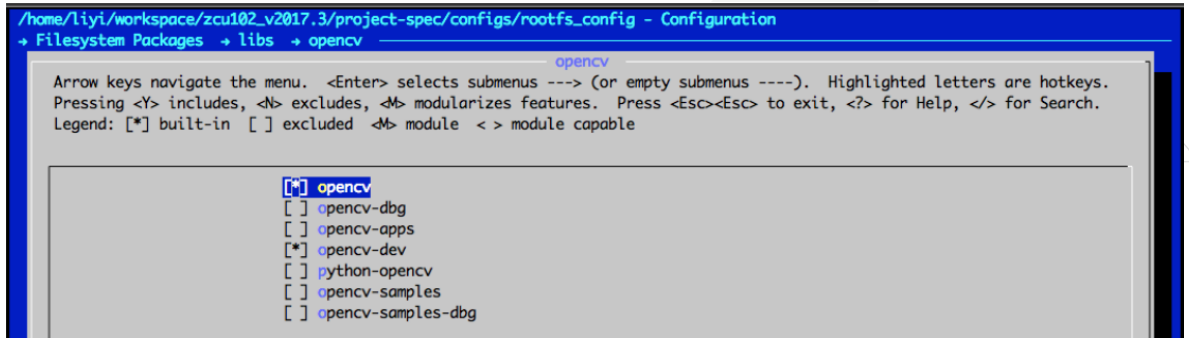


SW Integration with SDK

- **Device tree configuration**
  - set interrupt number according to block design
  - set core-num
- **OpenCV configuration**
  - Enable in Filesystem Packages -> misc or libs
- **Driver and DNNDK lib**
  - Provide kernel information & OpenCV version to Xilinx
  - Xilinx will provide driver and DNNDK package with install script
  - Install driver and DNNDK lib

```
amba {
    ...
    dpu@80000000 {

        compatible = "deephi, dpu";
        interrupt-parent = <&intc>;
        interrupts = <0x0 106 0x1 0x0 107 0x1>;
        reg = <0x0 0x80000000 0x0 0x700>;
        memory = <0x60000000 0x20000000>;
        core-num = <0x2>;
    };
    ....

}
```

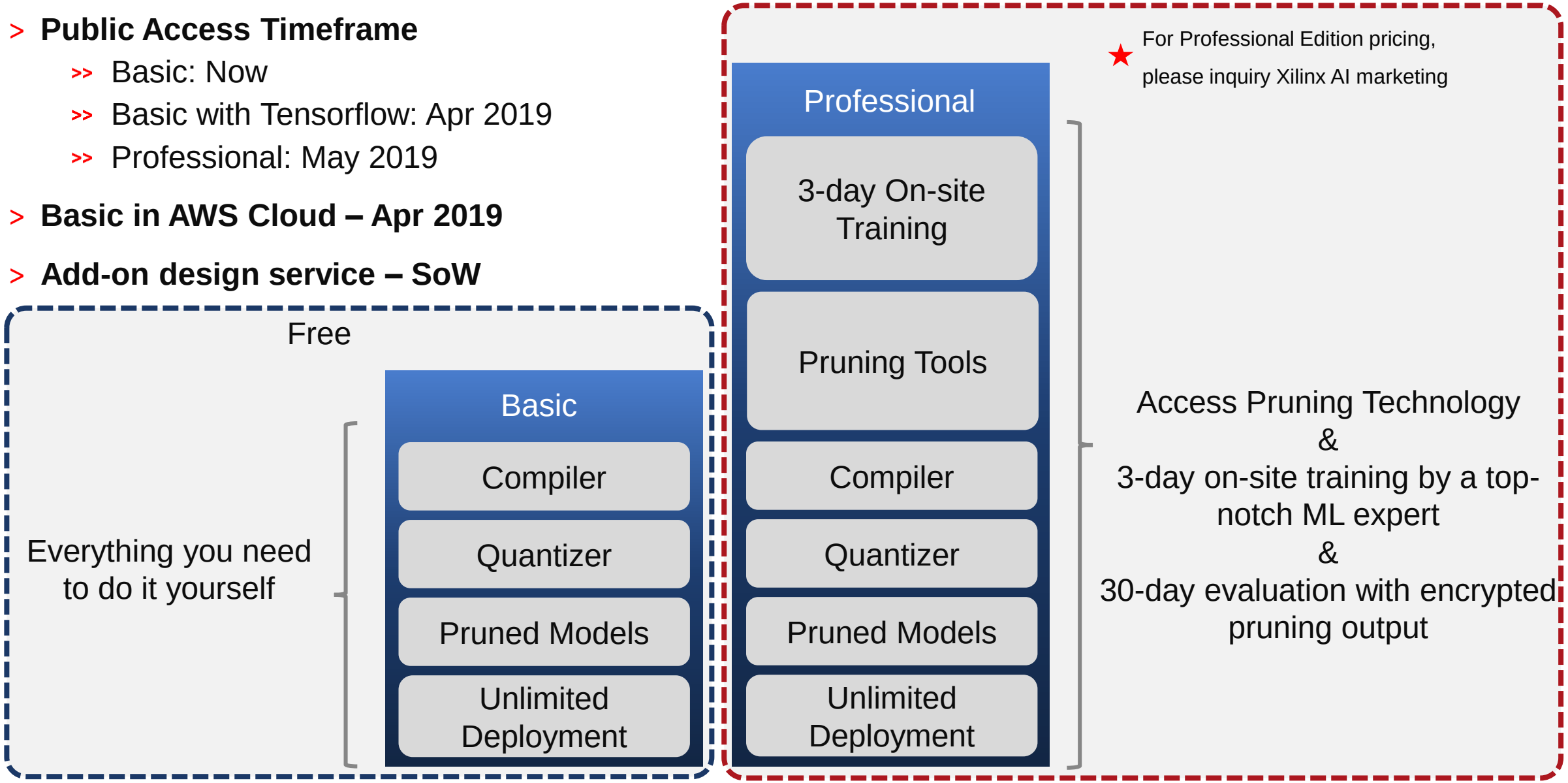




Availability

Basic and Professional Editions

- **Public Access Timeframe**
  - Basic: Now
  - Basic with Tensorflow: Apr 2019
  - Professional: May 2019
- **Basic in AWS Cloud – Apr 2019**
- **Add-on design service – SoW**



Availability

- **DNNDK & DPU**
  - DNNDK basic edition - Download from Xilinx.com
  - Pruning tool, separate upon request
  - DPU available for evaluation & system integration upon request
- **Demos & Ref Designs**
  - General: Resnet50, Googlenet, VGG16, SSD, Yolo v2/v3, Tiny Yolo v2/v3, Mobilenet v1/v2 etc..
  - Video surveillance: face detection & traffic structure
  - ADAS/AD: multi-channel detection & segmentation
  - DPU TRD (Work in progress)
- **Documentation**
  - DNNDK user guide – UG1327
  - DNNDK for SDSoC user guide – UG1331
  - Edge AI tutorials - https://github.com/Xilinx/Edge-AI-Platform-Tutorials
  - DPU product guide & tutorial (Work in progress)
- **Request or Inquiry**
  - Please contact Andy Luo, andy.luo@xilinx.com

DEEPHi 深鉴科技 Now Part of XILINX

Adaptable.
Intelligent.

XILINX

EBV Elektronik