Xilinx Edge AI Solution

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

Jan 2019
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
DNNDK – Deep Neural Network Development Kit

> DECENT (DEep ComprEssioN Tool)
> DNNC (Deep Neural Network Compiler)
> Runtime N²Cube (Cube of Neural Network)
> Profiler DSight

Customer Platform (Board, OS)
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

- **GoogleNet-V3**: Utilization 52%
  - 23% of total capacity
  - Source: Published results from Huawei

- **ResNet-50**: Utilization 51%
  - 24% of total capacity

- **VGG16**: Utilization 85%
  - 40% of total capacity

**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

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Next Layer Type</th>
<th>Conv</th>
<th>Deconv</th>
<th>Depth-wise Conv</th>
<th>Inner Product</th>
<th>Max Pooling</th>
<th>Ave Pooling</th>
<th>BN</th>
<th>ReLU</th>
<th>LeakyReLU</th>
<th>Element-wise</th>
<th>Concat</th>
<th>As Input</th>
<th>As Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv</td>
<td>Conv</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Deconv</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Depth-wise Conv</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Inner Product</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Max Pooling</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Ave Pooling</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>BN</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>ReLU</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>LeakyReLU</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Element-wise</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>Concat</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

- ●: Support
- ○: Support when selecting additional features
- X: Not support
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

© Copyright 2019 Xilinx
**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

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

#### More LUT

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

**DPU provides flexible option depending on customer’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

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

### LeakyRelu enabled

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

* 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)

- **Current B4096*2 wo Prune**
- **New B4096*3 wo Prune**

<table>
<thead>
<tr>
<th>Model</th>
<th>DSP</th>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-SSD</td>
<td>12</td>
<td>28.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG16</td>
<td>73</td>
<td>92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>118</td>
<td>179</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>313</td>
<td>445</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Peak INT8 OPS*

- Z7014S/Z7015 115G
- Z7020 230G
- Z7030 700G
- ZU2 576G
- ZU3 1.2T
- ZU4 1.6T
- ZU5 2.4T
- ZU6 2.9T
- ZU7 3.5T
- ZU9 4.1T
- ZU11 5.5T
- ZU15 6.8T

* 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

Dense Neural Network (FP32) → Prune → Finetune → Pruned Neural Network (FP32) → Quantize Parameter → Quantize Activation → Compressed sparse Neural Network (INT8)
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

SSD+VGG @ Surveillance 4classes

Pruning Speedup on DPU (SSD)

FPS

OPS

117G
19G
11.6G

18
71
103

2x DPU-4096@ZU9

© Copyright 2019 Xilinx
Pruning Makes Big Difference

(SSD 480x360)

<table>
<thead>
<tr>
<th>FPS (batch=1)</th>
<th>ZU9</th>
<th>ZU5</th>
<th>ZU2</th>
<th>7020</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>10W</td>
<td>10W</td>
<td>5W</td>
<td>3W</td>
</tr>
</tbody>
</table>

Result of Pruning

© Copyright 2019 Xilinx
## Classification Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Baseline</th>
<th>Pruning Result 1</th>
<th>Pruning Result 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top-5</td>
<td>ΔTop5</td>
<td>ratio</td>
</tr>
<tr>
<td>Resnet50 [7.7G]</td>
<td>91.65%</td>
<td>91.23%</td>
<td>-0.42%</td>
</tr>
<tr>
<td>Inception_v2 [4.0G]</td>
<td>91.07%</td>
<td>90.37%</td>
<td>-0.70%</td>
</tr>
<tr>
<td>SqueezeNet [778M]</td>
<td>83.19%</td>
<td>82.46%</td>
<td>-0.73%</td>
</tr>
</tbody>
</table>

## Detection Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Baseline mAP</th>
<th>Pruning Result 1</th>
<th>Pruning Result 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mAP</td>
<td>ΔmAP</td>
<td>ratio</td>
</tr>
<tr>
<td>DetectNet [17.5G]</td>
<td>44.46</td>
<td>45.7</td>
<td>+1.24</td>
</tr>
<tr>
<td>SSD+VGG [117G]</td>
<td>61.5</td>
<td>62.0</td>
<td>+0.5</td>
</tr>
<tr>
<td>[A] SSD+VGG [173G]</td>
<td>57.1</td>
<td>58.7</td>
<td>+1.6</td>
</tr>
<tr>
<td>[B] Yolov2 [198G]</td>
<td>80.4</td>
<td>81.9</td>
<td>+1.5</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Networks</th>
<th>Float32 baseline</th>
<th>8-bit Quantization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top1</td>
<td>Top5</td>
</tr>
<tr>
<td>Inception_v1</td>
<td>66.90%</td>
<td>87.68%</td>
</tr>
<tr>
<td>Inception_v2</td>
<td>72.78%</td>
<td>91.04%</td>
</tr>
<tr>
<td>Inception_v3</td>
<td>77.01%</td>
<td>93.29%</td>
</tr>
<tr>
<td>Inception_v4</td>
<td>79.74%</td>
<td>94.80%</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>74.76%</td>
<td>92.09%</td>
</tr>
<tr>
<td>VGG16</td>
<td>70.97%</td>
<td>89.85%</td>
</tr>
<tr>
<td>Inception-ResNet-v2</td>
<td>79.95%</td>
<td>95.13%</td>
</tr>
</tbody>
</table>
### DNNDK API

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpuOpen()</td>
</tr>
<tr>
<td>dpuClose()</td>
</tr>
<tr>
<td>dpuLoadKernel()</td>
</tr>
<tr>
<td>dpuDestroyKernel()</td>
</tr>
<tr>
<td>dpuCreateTask()</td>
</tr>
<tr>
<td>dpuRunTask()</td>
</tr>
<tr>
<td>dpuDestroyTask()</td>
</tr>
<tr>
<td>dpuEnableTaskProfile()</td>
</tr>
<tr>
<td>dpuGetTaskProfile()</td>
</tr>
<tr>
<td>dpuGetNodeProfile()</td>
</tr>
<tr>
<td>dpuGetInputTensor()</td>
</tr>
<tr>
<td>dpuGetInputTensorAddress()</td>
</tr>
<tr>
<td>dpuGetInputTensorSize()</td>
</tr>
<tr>
<td>dpuGetInputTensorScale()</td>
</tr>
<tr>
<td>dpuGetInputTensorHeight()</td>
</tr>
<tr>
<td>dpuGetInputTensorWidth()</td>
</tr>
<tr>
<td>dpuGetOutputTensor()</td>
</tr>
<tr>
<td>dpuGetOutputTensorAddress()</td>
</tr>
<tr>
<td>dpuGetOutputTensorSize()</td>
</tr>
<tr>
<td>dpuGetOutputTensorScale()</td>
</tr>
<tr>
<td>dpuGetOutputTensorHeight()</td>
</tr>
<tr>
<td>dpuGetOutputTensorWidth()</td>
</tr>
<tr>
<td>dpuGetInputTensorInCHWInt8()</td>
</tr>
<tr>
<td>dpuGetInputTensorInCHWFP32()</td>
</tr>
<tr>
<td>dpuGetInputTensorInHWCInt8()</td>
</tr>
<tr>
<td>dpuGetInputTensorInHWCFP32()</td>
</tr>
<tr>
<td>dpuGetOutputTensorInCHWInt8()</td>
</tr>
<tr>
<td>dpuGetOutputTensorInCHWFP32()</td>
</tr>
<tr>
<td>dpuGetOutputTensorInHWCInt8()</td>
</tr>
<tr>
<td>dpuGetOutputTensorInHWCFP32()</td>
</tr>
</tbody>
</table>

### 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;
    DPUPort *task_conv;
    DPUPort *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));

    /* Display 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 detach from DPU driver and free DPU resources */
    dpuFini();

    return 0;
}
```
DNNDK Hybrid Compilation Model

- Neural Network
  - DNCC
  - DPU Assembly
    - DNNAS
    - DPU ELF Object
      - Linker
      - Hybrid Executable

- C/C++ DL Application
  - GCC/LLVM
  - CPU Assembly
    - Assembler
    - CPU ELF Object
      - Linker
      - Hybrid Executable
Optimization in DNNC

Fusion & Decomposition

NN Layer DAG → Super-Layer DAG

Vertical Fusion

Vertical Fusion

Horizontal Fusion

Decomposition
DNNDK Runtime Engine

- Computer Vision App (DPU-accelerated)
  - Industry-standard Libraries
  - Loader
  - Tracer
  - Library

- Runtime N²Cube
  - Library
  - Loader
  - Tracer
  - Driver

- User Space
  - Kernel Space
  - Hardware Platform
    - Operating System
    - DPU Driver
    - Host CPU
    - DPU
## Supported Networks

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

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

© Copyright 2019 Xilinx
Video Surveillance ML Solutions

Intelligent IP Camera Solution

Face recognition camera with Zynq7020

Video Analytics Acceleration Solution

12-channel 1080P Video Analytics with ZU9EG
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

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

Color: White
Type: BUICK

Color: Blue
Number: 渝C LC689
ADAS/AD ML Reference Design

2D/3D Object Detection

Lane Detection

Segmentation

Pedestrian Detection

Segmentation + Detection

Pose Estimation

© Copyright 2019 Xilinx

>> 32
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
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
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**

---

Free

Everything you need to do it yourself

**Basic**

- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

**Professional**

- 3-day On-site Training
- Pruning Tools
- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

---

Access Pruning Technology & 3-day on-site training by a top-notch ML expert & 30-day evaluation with encrypted pruning output

For Professional Edition pricing, please inquiry Xilinx AI marketing

© Copyright 2019 Xilinx
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
Adaptable.
Intelligent.