Machine learning for embedded deep dive

Presented By

Jingxui Lui
Sr. Product Marketing Manager
Key Machine Learning Applications for Xilinx

Surveillance  ADAS/AD  Robotics  Data Center

Cloud ML

And there are many more …

Edge ML
Xilinx Value Proposition in Edge/Embedded ML

1. Only HW/SW configurable device for fast changing networks

2. High performance / low power with custom internal memory hierarchy

3. Future proof to lower precisions

4. Low latency end-to-end

5. Scalable device family for different applications
Key Challenges for Xilinx in Edge/Embedded ML

1. Deploy ML to Xilinx FPGA easily and quickly

2. Expand ML into non-FPGA customers

3. Delivers excellent performance with power & cost constraints for diverse embedded applications
Deephi Edge ML Solution
Unique, Patented Deep Learning Acceleration Techniques

- Best paper awards for breakthrough DL acceleration
- Deephi’s compression technology can:
  - 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

© Copyright 2018 Xilinx
DeePhi Solution Stack for Edge/Embedded ML

Models
Face detection  Pose estimation  Video analytics  Lane detection  Object detection  Segmentation

Framework
Caffe
Darknet
TensorFlow

Tools & IP
DEEPhi
DNNDK
DPU

HW Platforms
Z7020 Board  Z7020 SOM  ZU2 SOM  ZU2/3 Card  ZU9 Card  ZCU102  ZCU104  Ultra96

Compression
Pruning  Quantization

Compilation
Compiler  Assembler

Runtime
Core API  Loader  Driver  Profiler

DeePhi also has LSTM IP for KU115/VU9P as a part of Cloud ML
DNNDK Overview

- DECENT (DEep ComprEssioN Tool)
- DNNC (Deep Neural Network Compiler)
- DNNAS (Deep Neural Network ASsembler)
- Runtime N²Cube (Cube of Neural Network)
- DPU Simulator – Internal tool
- Profiler DSight
Framework Support

Caffe

- Pruning
- Quantization
- Compilation

- Pruning
- Quantization
- Convertor for caffe

- Quantization & Compilation
  - Eval version
  - Pruning
  - Internal version
DPU IP with High Efficiency

Utilization > 50% for mainstream neural networks

- GoogleNet-V3: 52%
- ResNet-50: 51%
- VGG16: 85%

Source: Published results from Huawei
Supported Operators

- Arbitrary Input Image Size
- Conv
  - Arbitrary Conv Kernel Size
  - Arbitrary Conv Stride/Padding
  - Dilation
- Pooling
  - Max/Avg Pooling
  - Arbitrary Max Pooling Size
    - Avg Pooling kernel size: 2x2~7x7
    - Arbitrary Pooling Stride/Padding
- ReLU / Leaky Relu
- Concat
- Deconv
- Depthwise conv
- Elementwise
- FC(Int8/FP32)
- Mean scale
- Upsampling
- Batch Normalization
- 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></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>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>
<td>●</td>
</tr>
<tr>
<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>
<td>●</td>
</tr>
<tr>
<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>
<td>●</td>
</tr>
<tr>
<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>X</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<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>X</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<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>
<td>○</td>
</tr>
<tr>
<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>
<td>○</td>
</tr>
<tr>
<td>LeakyReLU</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>×</td>
<td>X</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<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>
<td>○</td>
</tr>
<tr>
<td>Concat</td>
<td></td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>×</td>
<td>X</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

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

© Copyright 2018 Xilinx
DPU Typical Options & Interfaces

> B1152
  >> Parallelism: 4 * 12 * 12
  >> target Z7020/ZU2/ZU3

> B4096
  >> Parallelism: 8 * 16 * 16
  >> Target ZU5 and above
# DPU Peak Perf & Power

<table>
<thead>
<tr>
<th>Device</th>
<th>LUT</th>
<th>Flip-Flops</th>
<th>Block RAM</th>
<th>DSP (^1)</th>
<th>DPU Config</th>
<th>MACs (^2)</th>
<th>Peak (^3) performance</th>
<th>Frequency</th>
<th>Device Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z7020</td>
<td>53200</td>
<td>106400</td>
<td>4.9Mb</td>
<td>220</td>
<td>1xB1152</td>
<td>576</td>
<td>230GOPS</td>
<td>200MHz</td>
<td>2W</td>
</tr>
<tr>
<td>ZU2</td>
<td>47000</td>
<td>94000</td>
<td>5.3Mb</td>
<td>240</td>
<td>1xB1152</td>
<td>576</td>
<td>576GOPS</td>
<td>500MHz</td>
<td>3.5W</td>
</tr>
<tr>
<td>ZU3</td>
<td>71000</td>
<td>141000</td>
<td>7.6Mb</td>
<td>360</td>
<td>1xB1152</td>
<td>576</td>
<td>576GOPS</td>
<td>500MHz</td>
<td>N/A</td>
</tr>
<tr>
<td>ZU5(^4)</td>
<td>117000</td>
<td>234000</td>
<td>5.1Mb+18Mb</td>
<td>1248</td>
<td>1xB4096</td>
<td>2048</td>
<td>1350GOPS</td>
<td>330MHz</td>
<td>N/A</td>
</tr>
<tr>
<td>ZU7EV</td>
<td>230000</td>
<td>461000</td>
<td>11Mb+27Mb</td>
<td>1728</td>
<td>1xB4096 +2xB1152</td>
<td>2048 +2*576</td>
<td>2240GOPS</td>
<td>350MHz</td>
<td>N/A</td>
</tr>
<tr>
<td>ZU9</td>
<td>274000</td>
<td>548000</td>
<td>32.1Mb</td>
<td>2520</td>
<td>2xB4096</td>
<td>4096</td>
<td>2700GOPS</td>
<td>330MHz</td>
<td>10W</td>
</tr>
</tbody>
</table>

1) One DSP48E is used for two int8 multiplication
2) MACs is constructed by DSP and LUT (if DSP is not enough)
3) Peak performance is calculated by MACs: GOPS = 2*MACs*Frequency
4) Just list our conservative projection in performance
## DPU Utilization

<table>
<thead>
<tr>
<th>Configuration</th>
<th>LUT</th>
<th>Slice_reg</th>
<th>Block Ram</th>
<th>DSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single B1152 on Z7020</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All logic</td>
<td>53200</td>
<td>106400</td>
<td>140</td>
<td>220</td>
</tr>
<tr>
<td>DPU</td>
<td>45535</td>
<td>56961</td>
<td>110.5</td>
<td>220</td>
</tr>
<tr>
<td>Utilization ratio</td>
<td>85.59%</td>
<td>53.53%</td>
<td>78.93%</td>
<td>100.00%</td>
</tr>
<tr>
<td><strong>Single B1152 on ZU2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All logic</td>
<td>47232</td>
<td>94464</td>
<td>150</td>
<td>240</td>
</tr>
<tr>
<td>DPU</td>
<td>40703</td>
<td>55083</td>
<td>112</td>
<td>240</td>
</tr>
<tr>
<td>Utilization ratio</td>
<td>86.18%</td>
<td>58.31%</td>
<td>74.67%</td>
<td>100.00%</td>
</tr>
<tr>
<td><strong>Single B1152 on ZU3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All logic</td>
<td>70560</td>
<td>141120</td>
<td>216</td>
<td>360</td>
</tr>
<tr>
<td>DPU_B1152</td>
<td>36560</td>
<td>68729</td>
<td>115.5</td>
<td>288</td>
</tr>
<tr>
<td>Utilization ratio</td>
<td>51.81%</td>
<td>48.70%</td>
<td>53.47%</td>
<td>66.67%</td>
</tr>
<tr>
<td><strong>Dual B4096 on ZU9</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All logic</td>
<td>274080</td>
<td>548160</td>
<td>912</td>
<td>2520</td>
</tr>
<tr>
<td>DPU</td>
<td>156744</td>
<td>224650</td>
<td>501</td>
<td>2048</td>
</tr>
<tr>
<td>Utilization ratio</td>
<td>57.19%</td>
<td>40.98%</td>
<td>54.93%</td>
<td>81.27%</td>
</tr>
</tbody>
</table>
Perf Improvement with the Next Version DPU

Performance Comparison (FPS)

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

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

Resource Utilization Comparison

<table>
<thead>
<tr>
<th></th>
<th>DSP</th>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current B4096*2</td>
<td>2048</td>
<td>156744</td>
<td>224650</td>
<td>501</td>
</tr>
<tr>
<td>Next Version B4096*3</td>
<td>1926</td>
<td>110311</td>
<td>255020</td>
<td>748.5</td>
</tr>
</tbody>
</table>
DPU Scalability

Peak Perf
INT8 (OPS)

6.8T
5.5T
4.1T
3.5T
2.9T
2.8T
2.4T
1.7T
1.6T
1.2T
700G
576G
230G
115G
102G
56G

ZU15
ZU11
ZU9
ZU7
ZU6
ZU5
ZU4
ZU3
ZU2
ZU10
Z7045
Z7035
Z7030
Z7020
Z7014S/Z7015
Z7012S
Z7014S/Z7015
Z7030
Z7020
Z7010

DPU Configuration

<table>
<thead>
<tr>
<th>DPU Configuration</th>
<th>LUTs</th>
<th>Registers</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>B256 (8x4x4)</td>
<td>16132</td>
<td>25064</td>
<td>43</td>
<td>66</td>
</tr>
<tr>
<td>B256 (2x8x8)</td>
<td>15286</td>
<td>22624</td>
<td>53.5</td>
<td>50</td>
</tr>
<tr>
<td>B288 (4x6x6)</td>
<td>15812</td>
<td>23689</td>
<td>46</td>
<td>62</td>
</tr>
<tr>
<td>B512 (4x8x8)</td>
<td>20177</td>
<td>31782</td>
<td>69.5</td>
<td>98</td>
</tr>
<tr>
<td>B1024 (8x8x8)</td>
<td>27377</td>
<td>46241</td>
<td>101.5</td>
<td>194</td>
</tr>
<tr>
<td>B1152 (4x12x12)</td>
<td>28698</td>
<td>46906</td>
<td>117.5</td>
<td>194</td>
</tr>
<tr>
<td>B1600 (8x10x10)</td>
<td>30877</td>
<td>56267</td>
<td>123</td>
<td>282</td>
</tr>
<tr>
<td>B2304 (8x12x12)</td>
<td>34379</td>
<td>67481</td>
<td>161.5</td>
<td>396</td>
</tr>
<tr>
<td>B3136 (8x14x14)</td>
<td>38555</td>
<td>79867</td>
<td>203.5</td>
<td>506</td>
</tr>
<tr>
<td>B4096 (8x16x16)</td>
<td>40865</td>
<td>92630</td>
<td>249.5</td>
<td>642</td>
</tr>
</tbody>
</table>

* B256/288/512/3136 work in progress
DECENT – Deephi Deep Compression Tool

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

Deep compression makes algorithm smaller and lighter.

Compression efficiency

Deep Compression Tool can achieve significant compression on CNN and RNN.

Accuracy

Algorithm can be compressed 7 times without losing accuracy under SSD object detection framework.
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 Results

### Classification Networks

<table>
<thead>
<tr>
<th>Networks</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>Top-5</td>
<td>ΔTop5</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>Networks</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>
Pruning Example - SSD

SSD+VGG @Deephi Surveillance 4classes

Pruning Speedup on DPU (SSD)
Makes Big Difference with Pruning

(SSD 480x360)

<table>
<thead>
<tr>
<th>FPS (batch=1)</th>
<th>Jetson TX2</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>
<td>2W</td>
</tr>
</tbody>
</table>

Result of DeePhi Pruning

© Copyright 2018 Xilinx
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

© Copyright 2018 Xilinx
## 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>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpuOpen()</td>
<td>dpuGetOutputTensorSize()</td>
</tr>
<tr>
<td>dpuClose()</td>
<td>dpuGetOutputTensorScale()</td>
</tr>
<tr>
<td>dpuLoadKernel()</td>
<td>dpuGetOutputTensorHeight()</td>
</tr>
<tr>
<td>dpuDestroyKernel()</td>
<td>dpuGetOutputTensorWidth()</td>
</tr>
<tr>
<td>dpuCreateTask()</td>
<td>dpuGetOutputTensorChannel()</td>
</tr>
<tr>
<td>dpuRunTask()</td>
<td>dpuGetTensorSize()</td>
</tr>
<tr>
<td>dpuDestroyTask()</td>
<td>dpuGetTensorAddress()</td>
</tr>
<tr>
<td>dpuEnableTaskProfile()</td>
<td>dpuGetTensorScale()</td>
</tr>
<tr>
<td>dpuGetTaskProfile()</td>
<td>dpuGetTensorHeight()</td>
</tr>
<tr>
<td>dpuGetNodeProfile()</td>
<td>dpuGetTensorWidth()</td>
</tr>
<tr>
<td>dpuGetInputTensor()</td>
<td>dpuGetTensorChannel()</td>
</tr>
<tr>
<td>dpuGetInputTensorAddress()</td>
<td>dpuSetInputTensorInCHWInt8()</td>
</tr>
<tr>
<td>dpuGetInputTensorSize()</td>
<td>dpuSetInputTensorInCHWF32()</td>
</tr>
<tr>
<td>dpuGetInputTensorScale()</td>
<td>dpuSetInputTensorInHWCInt8()</td>
</tr>
<tr>
<td>dpuGetInputTensorHeight()</td>
<td>dpuSetInputTensorInHWCFP32()</td>
</tr>
<tr>
<td>dpuGetInputTensorWidth()</td>
<td>dpuGetOutputTensorInCHWInt8()</td>
</tr>
<tr>
<td>dpuGetInputTensorChannel()</td>
<td>dpuGetOutputTensorInCHWF32()</td>
</tr>
<tr>
<td>dpuGetOutputTensor()</td>
<td>dpuGetOutputTensorInHWCInt8()</td>
</tr>
<tr>
<td>dpuGetOutputTensorAddress()</td>
<td>dpuGetOutputTensorInHWCFP32()</td>
</tr>
</tbody>
</table>

> For more details, refer to DNNDK User Guide

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));

    /* 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 */
    dpuFinil();

    return 0;
}
DNNDK Hybrid Compilation Model
Optimization in DNNC

- **Fusion & Decomposition**
  - **NN Layer DAG**
    - **Super-Layer DAG**
  - **Vertical Fusion**
    - `conv`
    - `relu`
    - `pool`
    - `conv/relu/pool`
  - **Vertical Fusion**
    - `conv`
    - `conv`
    - `elementwise`
  - **Horizontal Fusion**
    - `conv/conv`
  - **Decomposition**
    - `conv/pool`
    - `conv/pool`
    - `concat`
    - `pool`
DNNDK Runtime Engine

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

User Space

Kernel Space

Operating System

DPU Driver

Hardware Platform

Host CPU

DPU

Runtime N²Cube
- Library
- Loader
- Tracer
- Driver

© Copyright 2018 Xilinx
## 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>Deephi</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Measured Performance (Cont.)

- **Baseline Network**
- **Pruned Network**
- **Deephi Designed Network**

<table>
<thead>
<tr>
<th>Network</th>
<th>Performance (FPS)</th>
<th>Computation (GOPS per image)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>117, 19.7</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>65, 25</td>
<td></td>
</tr>
<tr>
<td>Yolov2</td>
<td>36, 42</td>
<td></td>
</tr>
<tr>
<td>VGG16</td>
<td>30, 73</td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>3.8, 150</td>
<td></td>
</tr>
<tr>
<td>Yolov2</td>
<td>16, 95</td>
<td></td>
</tr>
<tr>
<td>VGG16</td>
<td>20, 100</td>
<td></td>
</tr>
<tr>
<td>SSD</td>
<td>11.6, 129</td>
<td></td>
</tr>
<tr>
<td>FPN</td>
<td>8.9, 120</td>
<td></td>
</tr>
<tr>
<td>Tiny Yolov2</td>
<td>7, 168</td>
<td></td>
</tr>
<tr>
<td>Tiny Yolov3</td>
<td>5.6, 170</td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>3.2, 313</td>
<td></td>
</tr>
<tr>
<td>Inception v1</td>
<td>1.6, 481</td>
<td></td>
</tr>
<tr>
<td>Inception v1(3.2, 313)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPGNet</td>
<td>10, 30</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>36.42</td>
<td></td>
</tr>
<tr>
<td>Yolov2</td>
<td>17, 54</td>
<td></td>
</tr>
<tr>
<td>VGG16</td>
<td>30, 73</td>
<td></td>
</tr>
<tr>
<td>FPN</td>
<td>8.9, 120</td>
<td></td>
</tr>
<tr>
<td>SSD</td>
<td>117, 19.7</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>65.25</td>
<td></td>
</tr>
</tbody>
</table>
Out-of-box Supported Boards

- DP8000
  - Z7020 SOM
- DP2400
  - ZU9 PCIe card
- Deephi ZU2/3 board
- Xilinx ZCU102
- Xilinx ZCU104
- Avnet Ultra96
Video Surveillance ML Solutions

Intelligent IP Camera Solution

Face recognition camera with Zynq7020

Video Analytics Acceleration Solution

8-channel 1080P Video Analytics with ZU9EG

© Copyright 2018 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

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

Car Attributes

Color: White
Type: BUICK

License Recognition

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

2D/3D Object Detection

Lane Detection

Pedestrian Detection

Segmentation + Detection

Segmentation

Pose Estimation
8-ch 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
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
ML Development with Deephi Solution
Development Method

Traditional

OpenCL/HLS

DeePhi

Algorithm

RTL

FPGA

Algorithm

C/C++

RTL

FPGA

Parameter

RTL

Instruction

FPGA
Two Development Flows of Using Deephi DPU IP

- **Vivado & SDK**
  - Traditional flow
  - Bottom up approach
  - Suitable for FPGA designer
  - Fine-grained customization

- **SDSoC**
  - New high-level abstraction flow
  - Top down approach
  - Suitable for algorithm & software developer
  - Higher Productivity
HW Integration with Vivado IPI

> Steps

>> 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
HW Integration with Vivado IPI (Cont.)

Steps (Cont.)

- Create top wrapper
- Generate bitstream
- Generate BOOT.BIN using Petalinux etc.

Note
- The port data width is consistent with DPU data width
- For frequency > 333MHz, clock wizard is needed between MPSoC and DPU
- Interrupt configuration was shown in binary.
  [3]: 0 - pl_ps_irq0 ; 1 - pl_ps_irq1
  [2:0]: interrupt number 0~7
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 Deephi
  >> Deephi will provide driver and DNNDK package with install script
  >> Install driver and DNNDK lib
HW Integration with C-callable IP

> **Steps**

- Create header file
- Package IP in Vivado
- Create Makefile to generate *.a
- Configure DPU parameters
- Build application software

```c
#include <dpu.hpp>

void main()
{
  uint32_t start = 0x1;
  dpu_set_start(start);
}
```

**Makefile**

```
# Create a Library
sdx_pack -header dpu.hpp -lib libdpu.a \
-func dpu_set_start -map starts_axi:in:0x10 -func-end \
-ip ../iprepo/dpu/component.xml -control none \
-add-ip-repo ../iprepo/src/ \
-target-family zynqplus \
-target-cpu cortex-a53 -target-os linux -verbose \
-header <header.h> -func <function_name> -map <swName>=><hwName>:direction:offset -func-end \
-ip <component.xml>: IP packed by the Vivado IP integrator, only one top IP allowed.
```
Deephi DPU IP Integration with SDSoC
How to Use DNNK in SDSoC

Only 3 steps!

Write it

Compile it

Run it

Software define development
Resnet50 Example with C-callable DPU IP in SDSoC
A Long Time for Every Build?

> SDSoC compiler compares the new data-motion network with the last one

> If the same, vpl will not be called to rerun syn & impl

> It only takes a few minutes if –
  >> Use the same C-callable IP library
  >> Use the same platform
  >> Use the same project setting

Generating data motion network
INFO: [DMAnalysis 83-4494] Analyzing hardware accelerators...
INFO: [DMAnalysis 83-4497] Analyzing callers to hardware accelerators...
INFO: [DMAnalysis 83-4444] Scheduling data transfer graph for partition 0
INFO: [DMAnalysis 83-4446] Creating data motion network hardware for partition 0
INFO: [DMAnalysis 83-4448] Creating software stub functions for partition 0
INFO: [DMAnalysis 83-4450] Generating data motion network report for partition 0
INFO: [DMAnalysis 83-4454] Rewriting caller code
Skipping block diagram (BD), address map, port information and device registration for partition 0
Rewrite caller functions
Multiple Sensors & Networks with C-callable DPU IP

ZCU102 (ZU9)

ARM Cortex-A53

SDSoC Application

<table>
<thead>
<tr>
<th>Video Lib</th>
<th>App Stub</th>
<th>Video Lib</th>
</tr>
</thead>
<tbody>
<tr>
<td>V4L2</td>
<td>DM* Driver</td>
<td>DRM</td>
</tr>
</tbody>
</table>

Linux

- SDSoC 2018.2 Linux
- 4 CNN models
  - Face detect, Joint detect, Traffic SSD, Ped SSD
  - 30, 12, 15, 13 FPS respectively
- 3 Live inputs + file / HDMI output
- Under 10 Watts

File

USB3

HDMI

MIPI

ISP/VPSS*

1x Deephi DPU

Face detect

Traffic SSD

Ped SSD

Joint detect

DDR

DDR

HDMI

© Copyright 2018 Xilinx
Availability
Basic and Professional Editions

- **Timeframe**
  - Early Access: Now
  - Public Access: Jan 2019
- **To be available on AWS in Cloud Editions**
- **Add-on design service**

### DeePhi Basic
- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

### DeePhi Professional
- 3-day On-site Training
- Pruning Tools

Accessibility:
- Free
- Everything you need to do it yourself

**Pricing TBD**
- Early Access: Now
- Public Access: Jan 2019
- To be available on AWS in Cloud Editions
- Add-on design service

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

> **DNNDK**
  > For DP8000(Z7020)/DP8020(ZU2) board, download from Deephi website
  > For other boards, separate package upon request
  > For pruning tool, separate upon request

> **Demos & Ref Designs**
  > General: Resnet50, Googlenet, VGG16, SSD, Yolo v2-v3, Tiny Yolo v2-v3, Mobilenet v2 etc..
  > Video surveillance: face detection & traffic structure
  > ADAS/AD: multi-channel detection & segmentation
  > C-callable DPU IP with SDSoC: Resnet50, Quad networks(Pedstrian, Pose, Face, Traffic)

> **Documentation**
  > DNNDK user guide
  > C-callable DPU IP w SDSoC user guide
  > DPU IP system integration user guide (Work in progress)
  > Pruning user guide (Work in progress)

> **Request or Inquiry**
  > Please contact Andy Luo, andy.luo@xilinx.com
Key Takeaway

1. Edge/Embedded ML bring great opportunities and challenges for Xilinx

2. Xilinx offers cutting-edge end-to-end Edge/Embedded ML solution

3. Tool/IP/Demo/Ref design available now for evaluation & development