Machine learning for embedded deep dive

Presented By

Andy Luo
Sr. Product Marketing Manager
2018-10-01
Key Machine Learning Applications for Xilinx

- Surveillance
- ADAS/AD
- Robotics
- Data Center

And there are many more …

Edge ML

Cloud 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

© Copyright 2018 Xilinx
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
Xilinx Announces the Acquisition of DeePhi Tech

Deal to Accelerate Data Center and Intelligent Edge Applications

BEIJING and SAN JOSE, Calif., July 17, 2018 – Xilinx, Inc. (NASDAQ: XLNX), the leader in adaptive and intelligent computing, announced today that it has acquired DeePhi Tech, a Beijing-based privately held start-up with industry-leading capabilities in machine learning, specializing in deep compression, pruning, and system-level optimization for neural networks.
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)
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

Aristotle on 7020 FPGA
GoogleNet-V3: 52%
ResNet-50: 51%
VGG16: 85%

Source: Published results from Huawei

© Copyright 2018 Xilinx
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</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>×</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>×</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>×</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>×</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

- ●: Support
- ○: Support when selecting additional features
- ×: Not support
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></th>
<th>LUT</th>
<th>Flip-Flops</th>
<th>Block RAM</th>
<th>DSP&lt;sup&gt;1)&lt;/sup&gt;</th>
<th>DPU Config</th>
<th>MACs&lt;sup&gt;3)&lt;/sup&gt;</th>
<th>Peak&lt;sup&gt;3)&lt;/sup&gt; 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&lt;sup&gt;4)&lt;/sup&gt;</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>53200</td>
<td>106400</td>
<td>140</td>
<td>220</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>47232</td>
<td>94464</td>
<td>150</td>
<td>240</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>70560</td>
<td>141120</td>
<td>216</td>
<td>360</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>274080</td>
<td>548160</td>
<td>912</td>
<td>2520</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)

- **VGG-SSD**
  - Current B4096*2: 12 FPS
  - New B4096*3: 28.3 FPS

- **VGG16**
  - Current B4096*2: 73 FPS
  - New B4096*3: 92 FPS

- **ResNet50**
  - Current B4096*2: 118 FPS
  - New B4096*3: 179 FPS

- **GoogLeNet**
  - Current B4096*2: 313 FPS
  - New B4096*3: 445 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

<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

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
DNNDK Dev Flow

Five Steps with DNNDK

01 Model Compression
02 Model Compilation
03 Programming
04 Hybrid Compilation
05 Execution
DECENT – Deepphi Deep Compression Tool

Dense Neural Network (FP32) → Prune → Pruning (Less number of param) → Pruned Neural Network (FP32) → Quantize Parameter → Quantize Activation → Compressed sparse Neural Network (INT8)
Deep Compression Overview

Deep compression
Makes algorithm smaller and lighter

Highlight

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

- Analyze
  - 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>Model</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>-0.42%</td>
<td>40%</td>
</tr>
<tr>
<td>Inception_v2 [4.0G]</td>
<td>91.07%</td>
<td>-0.70%</td>
<td>60%</td>
</tr>
<tr>
<td>SqueezeNet [778M]</td>
<td>83.19%</td>
<td>-0.73%</td>
<td>89%</td>
</tr>
</tbody>
</table>

### Detection Networks

<table>
<thead>
<tr>
<th>Model</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>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
</tr>
<tr>
<td>105</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>35</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

- **Jetson TX2**: 10W
- **ZU9**: 10W
- **ZU5**: 5W
- **ZU2**: 3W
- **7020**: 2W

Result of DeePhi Pruning
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

- 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()
- dpuSetInputTensorInCHWInt8()
- dpuSetInputTensorInCHWFP32()
- dpuSetInputTensorInHWCIInt8()
- dpuSetInputTensorInHWCFP32()
- dpuGetOutputTensorInCHWInt8()
- dpuGetOutputTensorInCHWFP32()
- dpuGetOutputTensorInHWCIInt8()
- dpuGetOutputTensorInHWCFP32()

> For more details, refer to DNNDK User Guide

http://www.deephi.com/technology/dnndk
# 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));

    /* 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

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

C/C++ DL Application → GCC/LLVM → CPU Assembly → Assembler → CPU ELF Object
Optimization in DNNC
DNNDK Runtime Engine

Runtime N^2Cube
- Library
- Loader
- Tracer
- Driver
# 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

- Inception v1 (3.2, 313)
- Tiny Yolov2 (7, 168)
- Tiny Yolov3 (5.6, 170)
- Yolov2 (36, 42)
- VGG16 (30, 73)
- Yolov3 (65, 25)
- ResNet50 (7.7, 118)
- SSD (117, 19.7)
Measured Performance (Cont.)

<table>
<thead>
<tr>
<th>Network</th>
<th>Performance (FPS)</th>
<th>Computation (GOP per image)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception v1</td>
<td>(3.2, 313)</td>
<td></td>
</tr>
<tr>
<td>Inception v1</td>
<td>(1.6, 481)</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>(5.6, 170)</td>
<td></td>
</tr>
<tr>
<td>Yolov2</td>
<td>(3.6, 95)</td>
<td></td>
</tr>
<tr>
<td>VGG16</td>
<td>(20, 100)</td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>(3.8, 150)</td>
<td></td>
</tr>
<tr>
<td>SSD</td>
<td>(11.6, 129)</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>(17, 54)</td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>(7.7, 118)</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>VGG16</td>
<td>(30, 73)</td>
<td></td>
</tr>
<tr>
<td>Yolov2</td>
<td>(16, 95)</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>(5.6, 170)</td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>(7.7, 118)</td>
<td></td>
</tr>
<tr>
<td>SSD</td>
<td>(117, 19.7)</td>
<td></td>
</tr>
<tr>
<td>Tiny Yolov2</td>
<td>(7, 168)</td>
<td></td>
</tr>
<tr>
<td>Yolov3</td>
<td>(65, 25)</td>
<td></td>
</tr>
</tbody>
</table>
Measured Performance (Cont.)

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

The diagram illustrates the measured performance of various networks in terms of **Performance (FPS)** versus **Computation (GOPS per image)**.

For instance:
- **SSD (117, 19.7)**
- **Yolov3 (65, 25)**
- **VGG16 (30, 73)**
- **Inception v1 (1.6, 481)**
- **Deephi Designed Network**

The network performances are marked with different symbols and colors in the graph.
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
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

Detection & Tracking → Person Attributes

Detection & Tracking → Car Attributes

Color: White
Type: BUICK

Color: Blue
Number: 渝C LC689

Plate Detection → License Recognition
ADAS/AD ML Reference Design

2D/3D Object Detection

Lane Detection

Segmentation

Pedestrian Detection

Segmentation + Detection

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
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
Deephi DPU IP Integration with SDSoC
How to Use DNNK in SDSoC

Only 3 steps!

1. Write it
2. Compile it
3. 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: [DMAAnalysis 83-4494] Analyzing hardware accelerators...
INFO: [DMAAnalysis 83-4497] Analyzing callers to hardware accelerators...
INFO: [DMAAnalysis 83-4444] Scheduling data transfer graph for partition 0
INFO: [DMAAnalysis 83-4446] Creating data motion network hardware for partition 0
INFO: [DMAAnalysis 83-4448] Creating software stub functions for partition 0
INFO: [DMAAnalysis 83-4450] Generating data motion network report for partition 0
INFO: [DMAAnalysis 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)

<table>
<thead>
<tr>
<th>ARM Cortex-A53</th>
<th>SDSoc Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Lib</td>
<td>App Stub</td>
</tr>
<tr>
<td>V4L2</td>
<td>Video Lib</td>
</tr>
<tr>
<td></td>
<td>DM* Driver</td>
</tr>
<tr>
<td></td>
<td>DRM</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
</tr>
</tbody>
</table>

- 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
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
- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

---

Free

Everything you need to do it yourself

---

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

---

Pricing TBD

---

© Copyright 2018 Xilinx
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