Xilinx DNN Processor
An Inference Engine, Network Compiler + Runtime for Xilinx FPGAs

Rahul Nimaiyar, Brian Sun, Victor Wu, Thomas Branca, Yi Wang, Justin Oo, Elliott Delaye, Aaron Ng, Paolo D'Alberto, Sean Settle, Bill Teng, Manasa Bollavaram, Chaithanya Dudha, Hanh Hoang, Swati Gupta, Alina Huang, Ephrem Wu, Samuel Bayliss, Phil James-Roxby, Ralph Wittig, Ashish Sirasao, Ravi Sunkavalli

21st August 2018
Xilinx Inference Engine – DNN Processor + Compiler

<table>
<thead>
<tr>
<th>Trained Model</th>
<th>Compiler + Runtime</th>
<th>Xilinx DNN Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Trained Model" /></td>
<td><img src="image2" alt="Compiler + Runtime" /></td>
<td><img src="image3" alt="Xilinx DNN Processor" /></td>
</tr>
</tbody>
</table>

- **Trained Model**: A trained model is required as input.
- **Compiler + Runtime**: A compiler and runtime are used to deploy the trained model on the Xilinx DNN Processor.
- **Xilinx DNN Processor**: The processor is depicted with various components such as Image Queue, Instruction Buffer, and Cross Bar.

**Key Features**:

- **Low Latency, High Throughput – Batch 1**: Enables efficient processing with minimal delay.
- **60-80% Efficiency**: High efficiency in terms of computational performance.
- **No FPGA Expertise Required**: Simplifies deployment and integration.

© Copyright 2018 Xilinx
Virtex® UltraScale+™ VU9P FPGA

- 16nm TSMC FF+ FPGA
- 2.5M System Logic Cells
- 6840 DSP Blocks (18x27 MACs)
- 382 Mbit On-Die SRAM
- 4 DDR4-2400 x72 Channels
- VU9P Virtex UltraScale+ FPGA
- 21 TOPS (INT8)
- 382 Mbit on-chip SRAM
- 64 GByte on-board DRAM
- 75W
Motivations for Deep Learning on FPGA

- **Data Parallel**
  - 2D Array of MACs
  - Flexible on-chip memory access
    - High Bandwidth, Multiple Access Ports

- **Data Reuse**
  - Near Memory Compute
  - Programmable routing for data & filter reuse

- **Compression & Sparsity**
  - Flexible Data Types
    - FP32/16, INT16/8/4/2, Binary/Ternary
  - Sparsity friendly compute
Virtex® UltraScale+™ Full Spectrum of Memory

5 Tiers of Memory -> Build custom memory hierarchy.
500 Mb of On-chip Memory and Tb/s of On-chip Memory Bandwidth
Xilinx DNN Processor (xDNN)

- Configurable Overlay Processor
- DNN Specific Instruction Set
  - Convolution, Max Pool etc.
- Any Network, Any Image Size
- High Frequency & High Compute Efficiency
- Compile and run new networks
xDNN – Channel Parallel Systolic Array

2D Channel Parallel Datapath
Systolic Array
Distributed Weight Buffers

Micro-Architecture Optimized for underlying Ultrascale+ FPGA Fabric
xDNN Processor – Tensor Memory

Channel Parallel Concurrent Access

From Systolic Array
From DDR DMA
To Systolic Array
To DDR DMA
To Parallel Processing
Efficient Memory Utilization

- Previous Layer Output
- 1x1 Conv
- 3x3 Conv Reduce
- 5x5 Conv Reduce
- 3x3 Conv
- 5x5 Conv
- Concatenated Output

Tensor Memory

- Time: $T_n$
- $T_{n+1}$
- $T_{n+2}$
- $T_{n+3}$

Previous Layer Output

- Input for Next Layer

© Copyright 2018 Xilinx
xDNN Compiler + Runtime

Deep Learning Frameworks

TensorFlow
Caffe
mxnet

Frontend

Framework Tensor Graph to Xilinx Tensor Graph

Tensor Graph Optimization

Compiler

Quantizer

Image

Runtime

CPU Layers
FPGA Layers

Model Weights
Calibration Set

https://github.com/Xilinx/ml-suite

© Copyright 2018 Xilinx
Graph Partitioning

- One time loading of sub-graph instructions
- Data Flow Execution

Pre-Processing | Subgraph 1 | Parallel Subgraphs | Post-Processing

FPGA or CPU | FPGA | CPU | FPGA | CPU

1 Core -> Multi-Core -> Multi-Chip
Fusing of Operations

- Fuse operations as pipe-lined stages
- Access activations only once
Instruction Level Parallelism

Previous Layer Output
- 1x1 Conv
- 3x3 Conv Reduce
- 5x5 Conv Reduce
- 3x3 Max Pool

Concatenated Output
- 3x3 Conv
- 5x5 Conv
- 1x1 Conv

Parallel Execution
- 3x3 Conv Reduce
- 3x3 Conv
- 3x3 Max Pool
- 5x5 Conv Reduce
- 5x5 Conv
- 1x1 Conv

Systolic Array

Image Queue
Instruction Buffer
Weights DMA Controller
Execution Controller
Pooling/ EWA
Bias
ReLU
Pooling
Cross Bar

Time
Automatic Intra Layer Tiling

- Tile when Feature Map size exceeds on-chip memory
- Work on full feature map depth
# xDNN Key Functions

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>(N \times M, \text{Stride} \ 1,2,4,8) (N,M=1-15)</td>
</tr>
<tr>
<td>Max Pool</td>
<td>(N \times M, \text{Stride} \ 1,2,4,8) (N,M=1-15)</td>
</tr>
<tr>
<td>Avg Pool</td>
<td>(N \times M, \text{Stride} \ 1,2,4,8) (N,M=1-15)</td>
</tr>
<tr>
<td>Dilated Convolution</td>
<td>Factor 1,2,4</td>
</tr>
<tr>
<td>De-Convolution</td>
<td>(N \times M, \text{Stride} \ 1,2,4,8) (N,M=1-15)</td>
</tr>
<tr>
<td>Up-sampling</td>
<td>Factor 2,4,8,16</td>
</tr>
<tr>
<td>Activation</td>
<td>ReLU, pReLU</td>
</tr>
<tr>
<td>Elementwise Addition</td>
<td>Any – Square, Rectangular</td>
</tr>
<tr>
<td>Precision</td>
<td>Int8, Int16</td>
</tr>
<tr>
<td>Network</td>
<td>Classification: e.g. ResNet</td>
</tr>
<tr>
<td></td>
<td>Object Detection: e.g. YOLO v2,</td>
</tr>
<tr>
<td></td>
<td>Segmentation: e.g. MaskRCNN</td>
</tr>
</tbody>
</table>
xDNN Implementation on VU9P

<table>
<thead>
<tr>
<th>Resource</th>
<th>Count</th>
<th>VU9P Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>612k</td>
<td>52%</td>
</tr>
<tr>
<td>DSPs</td>
<td>5493</td>
<td>80%</td>
</tr>
<tr>
<td>BRAM</td>
<td>228</td>
<td>38%</td>
</tr>
<tr>
<td>URAM</td>
<td>864</td>
<td>92%</td>
</tr>
</tbody>
</table>

800MHz – 90% of device Fmax

- 3 Large 96x16 PEs– 1 in each SLR
- Systolic Array at 800 MHz; Rest of logic at 400 MHz
xDNN Compute Efficiency

GoogLeNet v1 Efficiency

xDNN 74%

Other Architecture

60-80% Efficiency Across Networks

Full Benchmark Data at Xilinx Developers Forum – Oct’ 1, 2018
Custom Deep Learning Pipeline

Integrate Custom Applications with xDNN. Lower end-to-end latency

Video + ML
Genomics + ML
Risk Modelling + ML
Database + ML
Network IPS + ML
Storage + ML
Xilinx DNN Processor - Summary

XObject Performance

- 60-80%
- 90%

Efficiency

Frequency

- No FPGA Expertise Needed
- Compile & Run Trained Networks
- Deploy on AWS F1, other Cloud Platforms
- Deploy in PCIe Cards
XDF connects software developers and system designers to the deep expertise of Xilinx engineers, partners, and industry leaders.

Silicon Valley October 1-2  
Beijing October 6th  
Frankfurt December 10

Learn More www.xilinx.com/xdf