From Model to FPGA: Software-Hardware Co-Design for Efficient Neural Network Acceleration

Kaiyuan Guo¹,², Lingzhi Sui¹, Jiantao Qiu², Song Yao¹, Song Han¹,³, Yu Wang¹,², Huazhong Yang¹

¹ DeePhi Technology ² Tsinghua University, ³ Stanford University

Acknowledgement: Dongliang Xie and DeePhi Engineering Team
DeePhi Tech

- Discovering the philosophy behind deep learning computing
- Founded by Song Yao, Yu Wang, and Song Han in March 2016

- FPGA-based solution provider for deep learning

  ✓ Automatic compilation tool chain + mini board/IP
  ✓ Architecture for CNN and RNN-LSTM
  ✓ Supporting detection, tracking, object/speech recognition, translation, and etc.
• New Platform Expected for Deep Learning
• Trend in Neural Network Design
• Platform Selection
• Overall Flow
• Model Compression: Useful in Real-World Networks
• Activation Quantization: 8 Bits Are Enough
• **Aristotle:** Architecture for CNN Acceleration
• **Descartes:** Architecture for Sparse LSTM Acceleration
• Conclusion
Low-power high-performance platform for deep learning is urgently needed

**Client**

**Requirements**
- Real-time object recognition

**Limitation**
- Battery capacity

**Edge**

**Requirements**
- Real-time video analysis

**Limitation**
- High maintenance cost

**Cloud**

**Requirements**
- Low latency

**Limitation**
- High maintenance/cooling cost
Trend in Neural Network Design

- CNN for Object Recognition
- RNN-LSTM for Speech Recognition

Frameworks for different applications have not been unified
Trend in Neural Network Design

- **CNN: Smaller and Slimmer**

- Smaller: One convolution kernel has fewer computations
- Slimmer: fewer channels, fewer computations, less parallelism

A CNN accelerator should perform better with small Conv kernels and low parallelism
Trend in Neural Network Design

• RNN-LSTM: Larger and Deeper
  - Max dimension: \(128 \rightarrow 256 \rightarrow 512 \rightarrow 1024 \rightarrow 2048 \rightarrow 4096\)
  - Number of LSTM layers: \(1 \rightarrow 3 \rightarrow 5\)

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Source: Lei Jia et al., Baidu

• Larger model size, higher bandwidth requirement
  • An RNN-LSTM accelerator should overcome the bandwidth problem
**FPGA is good for inference applications**

- **CPU**: Not enough energy efficiency
- **GPU**: Extremely efficient in training, not enough efficiency in inference (batch size = 1)
- **DSP**: Not enough performance with high cache miss rate
- **ASIC**: Has high NRE: No clear huge market yet
- **ASIC**: Has long time-to-market but neural networks are in evolution

**FPGA**
- Acceptable power and performance
- Supports customized architecture
- High on-chip memory bandwidth
- Relatively short time to market
- High reliability

**FPGA-based deep learning accelerators meet most products’ requirements**
Software-Hardware Co-Design is Necessary

- Great redundancy in neural networks
  - VGG16 network can be compressed from 550MB to 11.3MB
  - FPGA has limited BRAM and DDR bandwidth
- Different neural network has different computation pattern
  - CNN: Frequent data reuse, dense
  - DNN/RNN/LSTM: No data reuse, sparse
  - Different architectures must adapt to different neural network
- Neural networks are in evolution
  - Architecture must adapts to new algorithms

Limitations of FPGA platform
- Limited BRAM size
- Limited DDR Bandwidth
Algorithm engineers can simply run the compiler tool to implement FPGA acceleration.
Overall Flow

Traditional FPGA-based Acceleration Faced Two Major Problems

- Long development period
  - Hand coded: 2 – 3 months
  - OpenCL and HLS: 1 month
- Insufficient performance and energy efficiency

DeePhi’s workflow solves the two problems in FPGA acceleration

- Compiler + Architecture instead of OpenCL
  - Algorithm designer need to know nothing about hardware
  - Generates instructions instead of RTL code
  - Compilation in 1 minute
- Much higher performance and energy efficiency
  - Hand-coded IP core and efficient architecture design
Model Compression: Useful in Real-World Networks

- Deep Compression: Useful for RNN-LSTM and FC layers in CNN

With re-training, we can achieve:
- < 10% sparsity for real-world FC layers in CNN
- ~ 15% sparsity for real-world LSTMs
- 4 bit weight quantization with no accuracy loss

Deep Compression is useful in real-world neural networks and can save a great deal of computations and bandwidth demands

Source: Song Han et al., Stanford University

Different gate in LSTM has different sensitivity
### Activation Quantization: 8 Bits Are Enough

- **Image classification on ILSVRC 2012**

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>FP32</td>
<td>FIXED-16</td>
<td>FIXED-8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORIGINAL</td>
<td>RAW</td>
<td>RE-TRAIN</td>
<td>RAW</td>
</tr>
<tr>
<td></td>
<td>65.77%</td>
<td>65.78%</td>
<td>67.84%</td>
<td>65.58%</td>
</tr>
<tr>
<td></td>
<td>86.64%</td>
<td>86.65%</td>
<td>88.19%</td>
<td>86.38%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>Top-1</td>
<td>Top-5</td>
<td>Top-1</td>
<td>Top-5</td>
</tr>
<tr>
<td></td>
<td>68.60%</td>
<td>88.65%</td>
<td>68.70%</td>
<td>88.45%</td>
</tr>
<tr>
<td></td>
<td>88.70%</td>
<td>88.45%</td>
<td>88.70%</td>
<td>88.45%</td>
</tr>
<tr>
<td></td>
<td>62.75%</td>
<td>85.70%</td>
<td>62.75%</td>
<td>85.70%</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>Top-1</td>
<td>Top-5</td>
<td>Top-1</td>
<td>Top-5</td>
</tr>
<tr>
<td></td>
<td>58.69%</td>
<td>81.37%</td>
<td>58.69%</td>
<td>81.37%</td>
</tr>
<tr>
<td></td>
<td>58.69%</td>
<td>81.35%</td>
<td>81.36%</td>
<td>80.32%</td>
</tr>
<tr>
<td></td>
<td>57.27%</td>
<td>80.35%</td>
<td>57.27%</td>
<td>80.35%</td>
</tr>
</tbody>
</table>

- **Object detection on PASCAL VOC 2007**
  - **R-FCN:** < 2% mAP loss without re-training using 8-bit quantization
  - **YOLO:** < 1% mAP loss without re-training using 8-bit quantization
Activation Quantization: 8 Bits Are Enough

- Image classification: Results comparison
  - GoogLeNet
  - SqueezeNet
  - VGG16
<table>
<thead>
<tr>
<th>GoogLeNet</th>
<th>SqueezeNet</th>
<th>VGG16</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP32</td>
<td>FIXED-8</td>
<td>FP32</td>
</tr>
<tr>
<td>Shetland Sheepdog</td>
<td>Shetland Sheepdog</td>
<td>Shetland Sheepdog</td>
</tr>
<tr>
<td>Collie</td>
<td>Collie</td>
<td>Collie</td>
</tr>
<tr>
<td>Borzoi</td>
<td>Borzoi</td>
<td>Border collie</td>
</tr>
<tr>
<td>Afghan hound</td>
<td>Pomeranian</td>
<td>Afghan hound</td>
</tr>
<tr>
<td>Pomeranian</td>
<td>Afghan hound</td>
<td>Pomeranian</td>
</tr>
</tbody>
</table>

- Object detection: Results comparison
  - SqueezeNet + R-FCN
  - Similar proposal results with lower confidence

- Most differences are in low-priority guesses
• Based on Zynq 7000 Series FPGA
• Optimized for 3x3 Conv kernels
• Supports different Conv stride sizes
• Scalable design (1PE, 2PE, 4PE, 12PE) on Zynq 7010/7020/7030/7045
• Supports mainstream deep learning object framework: R-FCN, YOLO, and etc
Aristotle: Processing Element Architecture

- Integrate convolvers, adder tree, non-linearity, and pooling units into one PE
- Fully pipeline without intermediate data load/store
- Supports dynamic-precision quantization
From Model to Instructions

Parser -> Scheduling -> Code Generator

Prototxt -> Caffemodel -> Quantized Model

Re-train/No re-train

Compiler

Host CPU

Neural Network Accelerator

DDR

<table>
<thead>
<tr>
<th>Instruction 0</th>
<th>Data 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction 1</td>
<td>Data 1</td>
</tr>
<tr>
<td>Instruction 2</td>
<td>Data 2</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
Descartes: Architecture for Sparse LSTM Acceleration

- **EIE (Efficient Inference Engine):** Extremely efficient, but not for FPGA
  - Designed by Song Han et al. from Stanford University and published on ISCA 2016
  - 102 GOPS@600 mW, 800MHz

- **EIE chip (64PE):**
  - 10.13 MB SRAM
  - 64 Multiplier
  - 800MHz

- **Xilinx KU060:**
  - 4.75 MB BRAM
  - 2760 DSP
  - 250-300MHz

- **Xilinx KU115:**
  - 9.49MB BRAM
  - 5520 DSP
  - 250-300MHz

- FPGA has significantly more computing units but strictly limited on-chip memory
- LSTM cannot utilize activation sparsity
Descartes: Architecture for Sparse LSTM Acceleration

- Designed for LSTM: Supports any matrix size and layer number
- Supports any sparsity
- Considers scheduling and non-linear functions in LSTM
- Scalable design (16/32/64 PEs for each thread)
- Two modes: Batch (high throughput) / No Batch (low latency)
Evaluation: Platform and Benchmark for CNN

- **Platform Comparison**
  - **Nvidia Tegra K1 SoC**
    - 28 nm
    - ARM Cortex-A15 CPU
    - Kepler GPU 192 Cores
    - Caffe with CuDNN
  - **Xilinx Zynq 7000 Series**
    - 28nm
    - 85k/125k/350k logic cells
    - 220/400/900 DSP
    - 4.9/9.3/19.1Mb BRAM

- **Benchmark**
  - **VGG16**
    - Image classification
    - 30.68 Gop, 13 Conv layers
  - **YOLO Tiny**
    - General object detection
    - 5.54 Gop, 9 Conv layers
  - **Customized Network**
    - Face alignment
    - 104.6 Mop, 9 Conv layers

Copyright © DeePhi Tech 2016
Evaluation: Resource Utilization with Aristotle Architecture

- **Zynq 7020**

<table>
<thead>
<tr>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>53200</td>
<td>106400</td>
<td>140</td>
</tr>
<tr>
<td>Used</td>
<td>27761</td>
<td>26600</td>
<td>75</td>
</tr>
<tr>
<td>Ratio</td>
<td>52%</td>
<td>22%</td>
<td>54%</td>
</tr>
<tr>
<td>Ratio: 64%</td>
<td>22%</td>
<td>72%</td>
<td>100%</td>
</tr>
</tbody>
</table>

2 Processing elements
Peak performance: 86.4GOPS@150MHz

- **Zynq 7030**

<table>
<thead>
<tr>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>78600</td>
<td>157200</td>
<td>265</td>
</tr>
<tr>
<td>Used</td>
<td>43118</td>
<td>34097</td>
<td>203</td>
</tr>
<tr>
<td>Ratio</td>
<td>55%</td>
<td>22%</td>
<td>77%</td>
</tr>
<tr>
<td>Ratio: 64%</td>
<td>22%</td>
<td>72%</td>
<td>100%</td>
</tr>
</tbody>
</table>

4 Processing elements
Peak performance: 172.8GOPS@150MHz

- **Zynq 7045**

<table>
<thead>
<tr>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>218600</td>
<td>437200</td>
<td>545</td>
</tr>
<tr>
<td>Used</td>
<td>139385</td>
<td>85172</td>
<td>390.5</td>
</tr>
<tr>
<td>Ratio</td>
<td>64%</td>
<td>19%</td>
<td>72%</td>
</tr>
<tr>
<td>Ratio: 64%</td>
<td>19%</td>
<td>72%</td>
<td>100%</td>
</tr>
</tbody>
</table>

12 Processing elements
Peak performance: 518.4GOPS@150MHz

- **Tegra K1 GPU** - Peak performance: 326 GFOPS
Evaluation: Performance of Aristotle Architecture

- Runtime and performance*1 on TK1 and Zynq 7020

- Aristotle architecture performs better when network is small but has limited peak performance
- Zynq 7020 consumes 20% - 30% power of TK1 and costs less of TK1
- 1.78x higher performance on Zynq 7030 compared with Zynq 7020
- 4.94x higher performance on Zynq 7045 compared with Zynq 7020

*1 All results are measured with batch_size = 1
Evaluation: Platform and Benchmark for LSTM

• Platform Comparison

Nvidia K40 GPU
- 28nm
- 2880 CUDA Cores
- 810MHz / 875MHz
- 12GB GDDR5

Kintex Ultrascale Series
- 20nm
- 4.75/9.49MB BRAM (KU060/115)
- 2760/5520 DSP (KU060/115)
- 300MHz

• Benchmark: Real-world LSTM for Speech Recognition

- Max matrix size: 4096*1536
- Consider scheduling of multiple matrixes
- Consider non-linear functions
- 100 frames per second
Evaluation: Performance and Resource Utilization of Descartes Architecture

### Performance Comparison

<table>
<thead>
<tr>
<th>Platform</th>
<th>GPU K40*1</th>
<th>FPGA KU060</th>
<th>FPGA KU115</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense or Sparse</td>
<td>Dense</td>
<td>Sparse (10% sparsity)</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>810/875 MHz</td>
<td>300 MHz</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>FP32</td>
<td>FIXED-4 to FIXED-16</td>
<td></td>
</tr>
<tr>
<td>Threads to be Supported</td>
<td>Not limited</td>
<td>2 (Separate) / 32 (Batch)</td>
<td></td>
</tr>
<tr>
<td>Peak Performance</td>
<td>4.29 TFOPS</td>
<td>4.8 TOPS^3</td>
<td>9.6 TOPS^4</td>
</tr>
<tr>
<td>Real Power</td>
<td>235W</td>
<td>30 – 35W</td>
<td>45 – 50W</td>
</tr>
</tbody>
</table>

### Resource Utilization

#### KU060

<table>
<thead>
<tr>
<th></th>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>331680</td>
<td>663360</td>
<td>1080</td>
<td>2760</td>
</tr>
<tr>
<td>Used</td>
<td>298875</td>
<td>446655</td>
<td>1011</td>
<td>1505</td>
</tr>
<tr>
<td>Ratio</td>
<td>90%</td>
<td>67%</td>
<td>94%</td>
<td>55%</td>
</tr>
</tbody>
</table>

#### KU115

<table>
<thead>
<tr>
<th></th>
<th>LUT</th>
<th>FF</th>
<th>BRAM</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>663360</td>
<td>1326720</td>
<td>2160</td>
<td>5520</td>
</tr>
<tr>
<td>Used</td>
<td>563403</td>
<td>848990</td>
<td>1155</td>
<td>2529</td>
</tr>
<tr>
<td>Ratio</td>
<td>85%</td>
<td>64%</td>
<td>54%</td>
<td>46%</td>
</tr>
</tbody>
</table>

*1 Results on K40 GPU were provided by DeePhi’s partners

*2 Generally, real performance is 85%-90% of peak performance with Descartes architecture

*3 480GOPS for dense LSTM

*4 960 GOPS for dense LSTM

Copyright @ DeePhi Tech 2016
• DeePhi: Making deployment of deep learning algorithms simple and efficient
  – Automatic compilation tool
    • Deep compression
    • Activation quantization
    • Compiler
  – Aristotle: Architecture for CNN acceleration
  – Descartes: Architecture for sparse LSTM acceleration

Evaluation boards will be shipped in Oct 2016
Apply for test at partner@deephi.tech

New architecture for CNN revealed in Q4 2016
Thank You!

Song Yao
Founder & CEO
songyao@deephi.tech

About us
– www.deephi.com
Collaborate with us
– partner@deephi.tech
Join us
– dream@deephi.tech