RFNoC Neural-Network Library using Vivado HLS (rfnoc-hls-neuralnet)

EJ Kreinar
Team “E to the J Omega”
Overview

An RFNoC out-of-tree module that can be used to simulate, synthesize, and run a neural network on an FPGA.
Why?

RF + FPGA + Machine Learning

“venn diagram”

FPGA

Machine Learning

RF Processing

1

2

??

3
Why?

- Conventional approach
- RFNoC

Nothing New
Why?

- Becoming very popular...
- AWS “F1”
  - Zynq UltraScale
- Intel / Microsoft
  - Altera FPGA SoC
- Lots of research
  - Grad schools
  - Commercial
- Not much open source?

**Designed for Image Processing**
Why?

- “Cognitive radios” (not really machine learning)
- Neural network success in other fields
- GRCon 2016
  - Demonstrated interest & results
- DARPA BAA 2017

Early adopter interest
Why?

- No current solution that combines all 3 technologies
- Lots of computing required
- Potentially requires running on embedded devices with low SWAP and high sample rate

rfnoc-hls-neuralnet
**Fundamental Problems**

- Neural network performance is HIGHLY dependent on architecture.
- FPGA development requires many iteration cycles of resource vs throughput vs latency tradeoffs.
- ... plus glue code.
- ... plus interaction with a processor.

A good solution will allow developers to *efficiently* iterate & change neural network architectures on the FPGA.
HLS + RFNoC

1. **C++** neural network implementation
2. **Verilog** RFNoC compute engine
3. **XML** gnuradio-companion block interface

Designed as a tool to efficiently implement FPGA neural networks
How To Use: C++

Neural net library provides pre-optimized “building blocks” that can instantiate a network similar to TensorFlow

- Fully connected layer
- IQ convolution layer
- One-dimensional convolution layer
- Activation functions:
  - relu
  - sigmoid
  - tanh
- Maxpool operation (size 2, stride 2)
How To Use: C++

Simulate the neural network using both floating point and fixed point data types

Synthesize into HDL code using Vivado HLS

Easy prototyping & simulation of neural network in C++
### Selected Resource Estimates

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Size</th>
<th>BRAM18</th>
<th>DSP48</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Connected Layer</td>
<td>40 x 40</td>
<td>8</td>
<td>8</td>
<td>592</td>
<td>1128</td>
</tr>
<tr>
<td>Fully Connected Layer</td>
<td>784 x 256</td>
<td>193</td>
<td>8</td>
<td>647</td>
<td>1222</td>
</tr>
<tr>
<td>IQ Convolution</td>
<td>2 x 8 x 128</td>
<td>0</td>
<td>64</td>
<td>13440</td>
<td>37542</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>56</td>
<td>182</td>
</tr>
<tr>
<td>Maxpool (size 2, stride 2)</td>
<td>128 x 64</td>
<td>2</td>
<td>0</td>
<td>77</td>
<td>252</td>
</tr>
</tbody>
</table>

**Key Notes:**

1. Dense fully connected layers require too much BRAM for E310/E312
2. Large IQ convolution runs a high on FFT/LUT. Could be optimized more
3. Sigmoid/Maxpool essentially free
How To Use: HDL

Insert synthesized verilog module into an RFNoC Compute Engine (CE)

Convenient verilog wrapper interacts with synthesized HDL to format RFNoC packet sizes correctly:

“nnet_vector_wrapper.v”

Run RFNoC testbench to simulate HDL CE
How To Use: Gnuradio Companion

RFNoC CEs designed to use the “default” gnuradio ettus block definition. No custom C++ driver code required. Some data type wrangling needed.

Gnuradio + UHD manages software interface for X300 and E310 FPGAs.
Examples

Four examples provided as inspiration:

1. One-layer image classification based on Udacity’s machine learning course
   a. HLS implementation
   b. RFNoC HDL and GRC
2. Two layer image classification based on Udacity’s machine learning course
   a. HLS implementation ONLY
3. RF modulation recognition using a set of expert features
   a. HLS implementation
   b. RFNoC HDL and GRC
4. Convolutional network for modulation recognition of streaming IQ data
   a. HLS implementation ONLY
Future (Potential) Improvements

- Additional neural network types
  - 2D convolution, recurrent networks
- Programmable weights
  - Currently weights may only be hardcoded into HLS C++ code
- Improve weight storage
  - Use larger FPGA BRAMs
  - External/off chip RAM?
- Optimize existing neural network blocks for wider applications
- Alternate neural network architectures (binarized, etc)

Will prioritize based on feedback...
Live Demo

Modulation recognition on expert features

Thank You!