Deep Neural Networks Benchmark Suite for FPGAs Utilizing a TensorFlow to Routing High-Level Synthesis

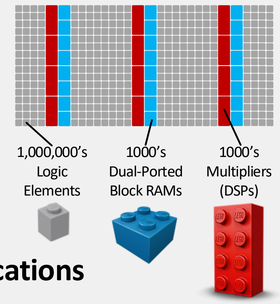
Ameer M.S. Abdeladi[*], Daniel Holanda Noronha[+], Steven J.E. Wilton[+], and Lesley Shannon[*]

[*] School of Engineering Science; Simon Fraser University; Burnaby BC, Canada, V5A 1S6
[+] Dept. of Electrical and Computer Engineering, The University of British Columbia, Vancouver BC, Canada V6T 1Z4

Motivation: FPGAs vs. GPUs for DNNs Inference

- More computation for quantized networks
- More efficient for irregular structures
- Higher performance per watts
- Higher acceleration (parallelism, pipelining, etc.)

→ **FPGAs are used as inference platforms for deep learning applications**



Need a metric to evaluate FPGA platforms CAD and architecture for DNNs inference

Deep Learning Benchmark Suite for FPGAs

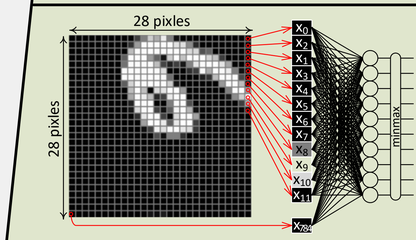
- Enables evaluating the compatibility of different FPGA architectures and CAD algorithms for deep learning application
- Comprises a diverse selection of deep learning applications, e.g., image classification, video classification, speech recognition, and language modeling
- Based on a complete design flow, starting from deep learning application described in python, and ending with an FPGA design

A Representative Collection of Deep Learning Applications

Benchmark	Application	Affiliation	Input	#parameters (10^6)	Layers												
					ConvNet			Feed-Forward		Pooling		Recurrent					
					1D	2D	3D	Fully-connected	Dropouts	Average	Max	Vanilla	LSTM	GRU	Softmax	total
AlexNet	Image classification	UofT	224×224 RGB image	60		✓		✓			✓				✓	8
Inception-V1	Image classification	Google	224×224 RGB image	5		✓		✓	✓	✓	✓				✓	22
VGG-16-C	Image classification	Oxford	224×224 RGB image	134		✓		✓			✓				✓	16
ResNet-34	Image classification	Microsoft	224×224 RGB image	30		✓		✓		✓	✓				✓	34
C3D	Video classification	Facebook AI	16-Frame 128×171 RGB	17.5		✓	✓	✓			✓				✓	15
DeepSpeech2	Speech recognition	Baidu Research	power normalized audio / 20ms windows	38	✓	✓		✓				✓		✓	✓	11
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Benchmark Design Flow

ML application described in TensorFlow's Python API




```
import tensorflow as tf
import numpy as np
x = tf.placeholder(tf.float32, shape=[None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  with tf.device("device:XLA_CPU:0"):
    y = tf.nn.softmax(tf.add(tf.matmul(x, W)[0], b))
  sess.run(y,{x: mnist_images,w: weights, b: bias})
```

Python

XLA JIT Compiler

Accelerated Linear Algebra (XLA) is a domain-specific just-in-time (JIT) compiler for TensorFlow Graphs [2]

TensorFlow XLA LLVM IR

TensorFlow to LegUp LLVM Compatibility

- Downgrades LLVM IR to match LegUp LLVM format
  - UBC flow, known as LeFlow [3], finds and downgrades nonmatching LLVM IR instructions
  - SFU flow compiles LLVM into an intermediate assembly, them decompiles assembly back to the correct LLVM format
- Maps LLVM inputs and output to memory blocks

LegUp-compatible LLVM IR

FPGA architecture and user constraints

C to Verilog high-level synthesis [4] utilizing LLVM compiler front end

Verilog HDL

VTR
Verilog-to-Routing
an open-source framework for conducting FPGA architecture and CAD research and development [5]

FPGA area and speed reports

References

[1] Nurvitadhi et al., "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?" In Proc. of *the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17)*, Feb. 2017.
[2] Chris Leary and Todd Wang, "XLA: TensorFlow, compiled!" *TensorFlow Dev. Summit*, Feb. 2017.
[3] Daniel Holanda Noronha and Steven J.E. Wilton, "LeFlow: Enabling Flexible FPGA High-Level Synthesis of Tensorflow Deep Neural Networks," 2018. [Online]. Available: https://github.com/danielholanda/LeFlow
[4] A. Canis et al., "LegUp High-Level Synthesis," chapter in FPGAs for Software Engineers, Springer, 2016.
[5] Luu et al., "VTR 7.0", ACM Transactions on Reconfigurable Technology and Systems, vol. 7, no. 2, pp. 1-30, 2014.

This research has been funded in part by the Computing Hardware for Emerging Intelligent Sensory Applications (COHESA) project.
COHESA is financed under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP485577-15.