“FFConv: An FPGA-based Accelerator for Fast Convolution Layers in Convolutional Neural Network”

Muhammad Adeel Pasha - Lahore University of Management Sciences

January 20, 2022
tinyML Talks Strategic Partners

Additional Sponsorships available – contact Olga@tinyML.org for info
Executive Strategic Partners
Arm: The Software and Hardware Foundation for tinyML

1. Connect to high-level frameworks
   - Profiling and debugging tooling such as Arm Keil MDK

2. Supported by end-to-end tooling
   - Optimized models for embedded
   - Runtime (e.g. TensorFlow Lite Micro)

3. Connect to Runtime
   - Optimized low-level NN libraries (i.e. CMSIS-NN)
   - RTOS such as Mbed OS
   - Arm Cortex-M CPUs and microNPUs

AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers
@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

© 2020 Arm Limited (or its affiliates)
Advancing AI research to make efficient AI ubiquitous

Power efficiency
- Model design, compression, quantization, algorithms, efficient hardware, software tools

Personalization
- Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning
- Robust learning through minimal data, unsupervised learning, on-device learning

Perception
- Object detection, speech recognition, contextual fusion

Reasoning
- Scene understanding, language understanding, behavior prediction

Action
- Reinforcement learning for decision making

A platform to scale AI across the industry

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
Neural Decision Processors
- At-Memory Compute
- Sustained High MAC Utilization
- Native Neural Network Processing

ML Training Pipeline
- Enables Production Quality Deep Learning Deployments

Data Platform
- Reduces Data Collection Time and Cost
- Increases Model Performance

End-to-End Deep Learning Solutions for TinyML & Edge AI

partners@syntiant.com
www.syntiant.com
Platinum Strategic Partners
WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with little to no drop in accuracy for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite's design space exploration

Deploy more models to your device without sacrificing performance or battery life with our easy-to-use software

BECOME BETA USER bit.ly/testdeeplite
Add Advanced Sensing to your Product with Edge AI / TinyML

Reality AI solutions
- Pre-built sound recognition models for indoor and outdoor use cases
- Solution for industrial anomaly detection
- Pre-built automotive solution that lets cars “see with sound”

Reality AI Tools® software
- Build prototypes, then turn them into real products
- Explain ML models and relate the function to the physics
- Optimize the hardware, including sensor selection and placement

Pre-built Edge AI sensing modules, plus tools to build your own

https://reality.ai  info@reality.ai  @SensorAI  Reality AI
BROAD AND SCALABLE EDGE COMPUTING PORTFOLIO

Microcontrollers & Microprocessors

Arm® Core
- Arm® Cortex®-M 32-bit MCUs
  - Arm ecosystem, Advanced security, Intelligent IoT
- Arm®-based High-end 32 & 64-bit MPUs
  - High-resolution HMI, Industrial network & real-time control
- Arm® Cortex®-M0+ Ultra-low Power 32-bit MCUs
  - Innovative process tech (SOTB), Energy harvesting

Renesas Core
- Ultra-low Energy 8 & 16-bit MCUs
  - Bluetooth® Low Energy, SubGHz, LoRa®-based Solutions
- High Power Efficiently 32-bit MCUs
  - Motor control, Capacitive touch, Functional safety, GUI
- 40nm/28nm process Automotive 32-bit MCUs
  - Rich functional safety and embedded security features

Renesas Synergy™
- Arm®-based 32-bit MCUs for Qualified Platform
  - Qualified software and tools

Core technologies

AI
- A broad set of high-power and energy-efficient embedded processors

Security & Safety
- Comprehensive technology and support that meet the industry’s stringent standards

Digital & Analog & Power Solution
- Winning Combinations that combine our complementary product portfolios

Cloud Native
- Cross-platforms working with partners in different verticals and organizations
Gold Strategic Partners
Maxim Integrated: Enabling Edge Intelligence

**Advanced AI Acceleration IC**

The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

[www.maximintegrated.com/MAX78000](http://www.maximintegrated.com/MAX78000)

---

**Low Power Cortex M4 Micros**

Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

[www.maximintegrated.com/microcontrollers](http://www.maximintegrated.com/microcontrollers)

---

**Sensors and Signal Conditioning**

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

[www.maximintegrated.com/sensors](http://www.maximintegrated.com/sensors)
LatentAI

Adaptive AI for the Intelligent Edge

Latentai.com
Build Smart IoT Sensor Devices From Data

SensiML pioneered TinyML software tools that auto generate AI code for the intelligent edge.

- End-to-end AI workflow
- Multi-user auto-labeling of time-series data
- Code transparency and customization at each step in the pipeline

We enable the creation of production-grade smart sensor devices.

sensiml.com
SynSense builds sensing and inference hardware for ultra-low-power (sub-mW) embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
miniature dreams can come true...

 TinyML Summit 2022
 March 28-30, 2022
 Hyatt Regency San Francisco Airport
 https://www.tinyml.org/event/summit-2022/

 The Best Product of the Year and the Best Innovation of the Year awards are open for nominations between November 15 and February 28.

 TinyML Research Symposium 2022
 March 28, 2022
 https://www.tinyml.org/event/research-symposium-2022

 More sponsorships are available: sponsorships@tinyML.org
Join Growing tinyML Communities:

tinyML - Enabling ultra-low Power ML at the Edge

The tinyML Community
https://www.linkedin.com/groups/13694488/

7.7k members in 43 Groups in 33 Countries

2.5k members & 4.3k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
## Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, February 1</td>
<td>Muhammad Shafique, New York University Abu Dhabi (NYUAD), UAE</td>
<td>Energy-Efficiency and Security for TinyML and EdgeAI: A Cross-Layer Approach</td>
</tr>
</tbody>
</table>

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting
Reminders

Slides & Videos will be posted tomorrow

Please use the Q&A window for your questions

tinyml.org/forums  youtube.com/tinyml
Muhammad Adeel Pasha

Muhammad Adeel Pasha (Senior Member, IEEE) received the B.Sc. degree in Electrical Engineering from the University of Engineering and Technology (UET), Lahore, Pakistan, in 2004 and the M.S. and Ph.D. degrees in Electrical and Computer Engineering (ECE) with specialization in Embedded Systems from University of Nice Sophia-Antipolis, Nice and University of Rennes-I, Rennes, France in 2007 and 2010, respectively. He is currently working as an associate professor with the Department of Electrical Engineering, Lahore University of Management Sciences (LUMS), Pakistan. He is also the director of the Electronics and Embedded Systems Lab at LUMS since 2014. He has several years of research and development experience and has published numerous refereed papers in major international journals and conferences. His research interests include energy-efficient hardware design for compute-intensive applications, real-time scheduling for multicore systems, and future platforms for green computing.
FFConv: An FPGA-based Accelerator for Fast Convolution Layers in Convolutional Neural Network

Muhammad Adeel Pasha

Afzal Ahmad and Muhammad Adeel Pasha

FFConv: An FPGA-based Accelerator for Fast Convolution Layers in Convolutional Neural Networks
DOI: https://doi.org/10.1145/3380548
Outline

• Motivation
• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution
• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization
• Proposed Hardware Design
• Results
• Motivation
• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution
• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization
• Proposed Hardware Design
• Results
Convolutional Neural Networks (CNNs)

- Object Detection
- Object Classification
- Semantic Segmentation
- Autonomous Driving
- Machine Vision
Motivation

Due to high computational and power consumption requirements of CNNs, FPGA-based HW Accelerators are gaining interest

Table 1. State of GPU vs FPGA-based CNN Implementations

<table>
<thead>
<tr>
<th></th>
<th>[7]</th>
<th>[23]</th>
<th>[2]</th>
<th>[42]</th>
<th>[24]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device</strong></td>
<td>TitanX</td>
<td>TitanX</td>
<td>ZCU102</td>
<td>Arria 10</td>
<td>Arria 10</td>
</tr>
<tr>
<td><strong>Network Arch.</strong></td>
<td>AlexNet</td>
<td>VGG16</td>
<td>VGG16</td>
<td>AlexNet</td>
<td>VGG16</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>float32</td>
<td>float32</td>
<td>fixed16</td>
<td>float16</td>
<td>fixed16</td>
</tr>
<tr>
<td><strong>Throughput (GOPS/s)</strong></td>
<td>6,937</td>
<td>5,600</td>
<td>2,941</td>
<td>1,382</td>
<td>1,790</td>
</tr>
<tr>
<td><strong>Frame rate (fps)</strong></td>
<td>5,120</td>
<td>182.5</td>
<td>95.8</td>
<td>1,020</td>
<td>58.3</td>
</tr>
<tr>
<td><strong>Power (W)</strong></td>
<td>227</td>
<td>134</td>
<td>23.6</td>
<td>45</td>
<td>37.5</td>
</tr>
<tr>
<td><strong>Power Efficiency (GOPS/s/W)</strong></td>
<td>30.56</td>
<td>41.80</td>
<td>124.60</td>
<td>30.71</td>
<td>47.78</td>
</tr>
</tbody>
</table>

\(^a\)NVIDIA Jetson TX2 Pascal GPU implementation.
\(^b\)Lightweight, custom implementation.
Outline

• Motivation

• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution

• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization

• Proposed Hardware Design

• Results
Convolutional Neural Networks (CNNs)

• Conv. layers utilize over 90% of the total computational budget of a typical CNN [1] and are used for feature extraction in images.

• Winograd Minimal Filtering [2] based Fast Convolution algorithms [3] can be used to reduce the arithmetic complexity of conv. layers of CNNs.

• Fast Conv. algorithms alter the algebraic framework of conv layers, reducing the no. of multiplications at the cost of an increase in arithmetic operations due to added transform stages.


Spatial Convolution

- Spatial Convolution is multiplication and addition (MulAdd) of corresponding elements of a learned kernel and an input feature map using

\[ Y_{i,k,x,y} = \sum_{c=1}^{C} \sum_{v=1}^{r} \sum_{u=1}^{r} D_{i,c,x+u,y+v} G_{k,c,u,v} \]

To compute \( F(m \times m, r \times r = 4 \times 4, 3 \times 3) \) in spatial conv, we need \( r^2 \times m^2 = 144 \) multiplications.
Fast Convolution

- Fast conv. algorithms work on the principle of **algorithmic strength reduction** whereby complex operations (multiplications) are replaced by cheaper operations (additions).

\[ Y = A^T [(B^T dB) \odot (GgG^T)] A \]
Convolutional Neural Networks (CNNs)

To compute $F(m \times m, r \times r = 4 \times 4, 3 \times 3)$ in fast conv., we need $(m+r-1)^2 = 36$ multiplications. Saving $4\times$ in multiplication complexity!
Convolutional Neural Networks (CNNs)

• Data Transform Stage $[B^T dB]$
  \[B^T = \begin{pmatrix}
  4 & 0 & -5 & 0 & 1 & 0 \\
  0 & -4 & -4 & 1 & 1 & 0 \\
  0 & -4 & -4 & -1 & 1 & 0 \\
  0 & -2 & -1 & 2 & 1 & 0 \\
  0 & 2 & -1 & -2 & 1 & 0 \\
  0 & 4 & 0 & -5 & 0 & 1 \\
\end{pmatrix}\]

• Filter Transform Stage: $[G g G^T]$  
  \[G = \begin{pmatrix}
  1/4 & 0 & 0 \\
  -1/6 & -1/6 & -1/6 \\
  -1/6 & 1/6 & -1/6 \\
  1/24 & 1/12 & 1/6 \\
  1/24 & -1/12 & 1/6 \\
  0 & 0 & 1 \\
\end{pmatrix}\]
Convolutional Neural Networks (CNNs)

• Inverse Transform Stage $[A^T MA]$
Outline

- Motivation
- Convolutional Neural Networks (CNNs)
  - Spatial Convolution
  - Fast Convolution
- Design Space Exploration
  - Filter Size Optimization
  - Data Quantization
- Proposed Hardware Design
- Results
Design Space Exploration: Filter Size Optimization

- Multiplication complexity of fast conv decreases quadratically with output tile size, $m$
- Transform complexity due to the added transform stages sees a quadratic increase with output tile size, $m$
Design Space Exploration: Filter Size Optimization

• Tradeoff between Decreased Multiplication Complexity and Increased Arithmetic Complexity.

![Graph showing the tradeoff between decreased multiplication complexity and increased arithmetic complexity for different filtering algorithms.](image)
Outline

• Motivation
• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution
• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization
• Proposed Hardware Design
• Results
• Low-precision training and inference in conventional CNNs has been widely studied
  • But quantization in fast conv. algorithms affect the classification accuracy differently due to the added transform stages
• Since the kernel weights are centered around zero with a low standard deviation, only one bit for integer portion and all remaining bits for the fractional portion would achieve maximum precision of the network based on fixed-point data types (Q1. ψ).

ψ = 15 costs only 0.27%, 0.29%, and 0.41% loss in classification accuracy for VGG16, AlexNet, and Shufflenet, respectively, compared to a float32 implementation.
Design Space Exploration: Feature Map Quantization

- Feature map inputs depend on the input image or outputs of previous layer and have high absolute values.
- We found that if the number of bits assigned to the integer portion is enough to hold the integer without overflowing, there is minimal loss in accuracy. Hence fixed-point data types (12.\(\phi\)) would suffice with \(\phi \geq 0\).

Q12.0 gives only 0.024%, 0.017%, and 0.118% drop in accuracy for VGG16, AlexNet, and Shufflenet, respectively, compared to float32.

For contrast, a float32 data type uses 2.67\(\times\) more bits than Q12.0 \(\rightarrow\) more proportional resources in hardware.
Outline

• Motivation
• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution
• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization
• Proposed Hardware Design
• Results
Base Design for a Convolution Engine (CE)

- Data Transform Stage (3-Stage Pipeline)
- Filter Transform Stage (3-Stage Pipeline)
- Element-wise Multiplication (EWM) Stage (implemented using DSP Blocks)
- Inverse Transform Stage (2-Stage Pipeline)
How to Optimize Memory Reuse?

We used Roofline Model to Compute the Theoretical Limits on Memory and Compute Intensities.

Roofline model for VGG16 depicting compute and bandwidth ceilings for both fast-conv-based and spatial-conv based CNNs.
Data Reuse to reduce External Memory Access

• Kernel Map Reuse
  • Every loaded kernel tile is fully reused until the current channel of the full batch of feature maps is convolved with the kernel tile

• Feature Map Overlap Reuse
Final FFConv Architecture
Outline

• Motivation
• Convolutional Neural Networks (CNNs)
  • Spatial Convolution
  • Fast Convolution
• Design Space Exploration
  • Filter Size Optimization
  • Data Quantization
• Proposed Hardware Design
• Results
Qualitative Comparison

Table 2. Qualitative Feature Comparison of FPGA-based Fast-conv Implementations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$m, r$</td>
<td>4,3</td>
<td>4,3</td>
<td>2,3</td>
<td>4,3$^a$</td>
<td>4,3</td>
</tr>
<tr>
<td>Quantization</td>
<td>No</td>
<td>Limited</td>
<td>No</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Multipliers Per DSP</td>
<td>0.25</td>
<td>1</td>
<td>0.25</td>
<td>2$^b$</td>
<td>1</td>
</tr>
<tr>
<td>Accuracy Measurement</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Stage Level Optimizations</td>
<td>No</td>
<td>Limited</td>
<td>No</td>
<td>Limited</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$^a$1D fast-conv, $F(m, r)$.

$^b$DSP blocks in most Intel FPGAs allow such a configuration.


Quantitative Comparison (VGG16-D)

Table 3. Performance Comparison for VGG16-D

<table>
<thead>
<tr>
<th></th>
<th>[8]</th>
<th>[4]</th>
<th>[5]</th>
<th>FFC_{conv}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of DSPs</td>
<td>780</td>
<td>2736</td>
<td>2520</td>
<td>2304</td>
</tr>
<tr>
<td>Number of LUTs</td>
<td>182.62K</td>
<td>107.84K</td>
<td>600K</td>
<td>337.3K</td>
</tr>
<tr>
<td>Data precision (bits)</td>
<td>16</td>
<td>32</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>150</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Inference Time (ms)</td>
<td>163.4</td>
<td>28.05</td>
<td>10.08</td>
<td>8.47</td>
</tr>
<tr>
<td>Throughput (GOPS/s)</td>
<td>187.8</td>
<td>1094.3</td>
<td>3044.7</td>
<td>3623.9</td>
</tr>
<tr>
<td>DSP efficiency (GOPS/DSP)</td>
<td>0.241</td>
<td>0.400</td>
<td>1.208</td>
<td>1.573</td>
</tr>
<tr>
<td>Slice LUT efficiency (GOPS/s/1000LUTs)</td>
<td>1.03</td>
<td>10.15</td>
<td>5.08</td>
<td>10.74</td>
</tr>
<tr>
<td>Power (W)</td>
<td>9.63</td>
<td>36.32</td>
<td>23.60</td>
<td>26.91</td>
</tr>
<tr>
<td>Power efficiency (GOPS/s/W)</td>
<td>19.50</td>
<td>30.13</td>
<td>124.60</td>
<td>134.67</td>
</tr>
</tbody>
</table>

Quantitative Comparison (AlexNet)

Table 4. Performance Comparison for AlexNet

<table>
<thead>
<tr>
<th></th>
<th>[9]</th>
<th>[5]</th>
<th>[7]</th>
<th>FFConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m, r$</td>
<td>-</td>
<td>4.3</td>
<td>4.3</td>
<td>4.3</td>
</tr>
<tr>
<td>Number of DSPs</td>
<td>2,240</td>
<td>2,520</td>
<td>1,476</td>
<td>2,304</td>
</tr>
<tr>
<td>Number of LUTs</td>
<td>303.56K</td>
<td>600K</td>
<td>153.75K</td>
<td>337.3K</td>
</tr>
<tr>
<td>Data precision (bits)</td>
<td>32</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>100</td>
<td>200</td>
<td>303</td>
<td>200</td>
</tr>
<tr>
<td>Inference Time (ms)</td>
<td>8.600</td>
<td>0.4757</td>
<td>0.3526</td>
<td>0.2029</td>
</tr>
<tr>
<td>Throughput (GOPS/s)</td>
<td>78.23</td>
<td>1,414.3</td>
<td>1,908.1</td>
<td>3,331.6</td>
</tr>
<tr>
<td>DSP efficiency (GOPS/s/DSP)</td>
<td>0.035</td>
<td>0.561</td>
<td>1.294</td>
<td>1.446</td>
</tr>
<tr>
<td>Slice LUT efficiency (GOPS/s/1000LUTs)</td>
<td>0.258</td>
<td>2.36</td>
<td>12.41</td>
<td>9.88</td>
</tr>
<tr>
<td>Power (W)</td>
<td>16.81</td>
<td>23.6</td>
<td>45.0</td>
<td>26.91</td>
</tr>
<tr>
<td>Power efficiency (GOPS/s/W)</td>
<td>4.654</td>
<td>59.93</td>
<td>42.40</td>
<td>123.81</td>
</tr>
</tbody>
</table>

$^a$ID Fast-conv, $F(m, r)$.

$^b$Two 18 × 18 multipliers per DSP.

Conclusions

- We designed and implemented FFConv, an FPGA-based conv. layer accelerator for CNNs
  - It uses Winograd Minimal Filtering based Fast Conv. Algorithms

- We performed extensive Design Space Exploration for
  - Finding optimized Winograd Filter size
  - Optimal fixed-point implementation

- We explored memory bandwidth-compute tradeoffs to mitigate the impact of memory bottlenecks in our design

- We optimized and constrained our design to the FPGA platform of our choice (Xilinx Virtex 7)

- Efficient data reuse schemes were utilized for both feature and kernel maps, resulting in high computation to communication (CTC) ratio

- We benchmarked our design using three popular CNN architectures, VGG16-D, AlexNet, and Shufflenet, achieving up to 42.59× the throughput, up to 41.31× DSP efficiency, and up to 26.6× power efficiency for different networks compared to the previous state-of-the-art designs.
Thank You 😊

Q n A
Copyright Notice

This multimedia file is copyright © 2022 by tinyML Foundation. All rights reserved. It may not be duplicated or distributed in any form without prior written approval.

tinyML® is a registered trademark of the tinyML Foundation.

www.tinyml.org