High-Performance, Deep Neural Networks with Sub-Microsecond Latency on FPGAs for Trigger Applications

- Implementation Strategies and Results -

Noel Nottbeck
Prof. Dr. Volker Büscher, Dr. Christian Schmitt
Johannes Gutenberg-Universität Mainz

ACAT 2019
March 14th
Starting Point

- Applications for ANNs in physics exist
- Inclusion of ANNs into triggers potentially interesting (e.g. Belle II\textsuperscript{1})
- High-performance triggers (e.g. ATLAS detector at CERN):
  - High data rate (LHC: 40 MHz)
  - Extremely low latency (10s to few 100s of ns)
  - often use FPGAs

⇒ Motivates creation of tool for automated network translation to FPGAs

\textsuperscript{1}Track vertex reconstruction with neural networks at the first level trigger of Belle II, doi:10.1051/epjconf/201715000009.
Layer Types

- **Dense Layers**

- **2D Convolutions (Multi-Channel)**

$$
\begin{array}{cccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array} \times 
\begin{array}{cc}
2 & 0 \\
-1 & 1 \\
\end{array} = 
\begin{array}{cc}
1 & 3 \\
7 & 9 \\
\end{array}
$$

- **2D Pooling (Multi-Channel)**

$$
\begin{array}{cccc}
3 & -15 & -3 & 7 \\
11 & 12 & 5 & 0 \\
-1 & 8 & 13 & 11 \\
7 & 5 & 29 & 10 \\
\end{array}
\rightarrow
\begin{array}{cc}
12 & 7 \\
8 & 29 \\
\end{array}
$$

⇒ Requires mostly multiplications, additions, compare/select (... and data flow management)
FPGA Key Design Elements

"Field Programmable Gate Array"

- **look-up tables (LUT)** (combinational logic, \( f : \{0, 1\}^n \rightarrow \{0, 1\} \))
- **registers (FF)** (bit storage)
- **programmable routing** (LUT/register wiring)
- **Specialized units:**
  - DSPs (’simple ALUs’, *MULT w/ subsequent ADD*)
  - block memory
  - ...

⇒ high I/O BW & computation; predictable, ns-scale latencies

image:
Development Aims

- Efficient resource usage
- High freq. → high perf.

⇒ Powerful networks in small FPGAs / spare resources for other algo.
- No in-depth implementation understanding required by user

(’High Level Synthesis for Machine Learning Framework’ with similar approach oriented at high-level language.²)

Arithmetics Implementation

Fixed-point w/ configurable precision (layer-wise) chosen:
- ≤ 16 bits sufficient for ANNs
- easier to implement, less resources required
Inference Performance Limit

- Use DSPs for multiply-accumulate (MAC) ops
  \[\rightarrow 1 \text{ MAC/cycle per DSP}\]
- E.g. ATLAS jFEX trigger sub-system after Phase-I upgrade: Xilinx UltraScale+ XCVU9P FPGA
  - 6840 DSPs
  - max freq: speed grade '1': \(\sim 640 \text{ MHz}\), '3': \(\sim 880 \text{ MHz}\)
  \[\Rightarrow \sim 4 \text{ TMAC/s}, \sim 6 \text{ TMAC/s}\]
- Example LHC data freq.: \(\sim 100k \text{ – } \sim 150k \text{ MAC/data set}\)
  (theoretical peak perf.)
- Ways to gain additional MAC/s exist
  (but currently not used)
General Firmware Structure

- Each layer as own design part
  (no reconfigurable general purpose resources, every layer type specifically tailored for efficient implementation)

- Design changes depending on number of proc. cycles per data set within each layer
  here: $f_{\text{proc}} = C \cdot f_{\text{data}}, C \ll 100$
  ($f_{\text{data}} \sim 10^1 \text{ MHz}, f_{\text{proc}} < 1000 \text{ MHz}, \text{LHC/ATLAS: } C \lesssim 20$, layers can process new data only after $C$ cycles passed)
Dense Layer Idea

- Exploit: Every neuron requires every input
  ⇒ Implement neuron processing in DSP pipelines:
    - Inputs completely reusable
    - Only weight loading/fetching/multiplexing
    - Simple Design
    - Easy parallelization
Dense Layer Results

Utilization:
- \( N_{\text{DSP}} \approx N_{\text{MAC}} \cdot C^{-1} = N_{\text{MAC}} \cdot \frac{f_D}{f_P} \) (close to ideal)
- Negligible LUT, FF util.
- BRAM util. acceptable (highly depends on exact design)

Frequency:
- Many 100 MHz for layers up to \( \sim 10k \) MACs
  \( (f_{\text{max}} \text{ decreases with increasing layer size}) \)
2D Convolution Layer Idea

- Extremely expensive \( (N_{MAC} \approx V_I \cdot V_K) \)
- Reuse inputs as much as possible
- Reuse weights as much as possible
- Fast and efficient implementation challenging

\[
\begin{array}{ccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array} \times 
\begin{array}{cc}
2 & 0 \\
-1 & 1 \\
\end{array} = 
\begin{array}{cc}
1 & 3 \\
7 & 9 \\
\end{array}
\]
2D Convolution Layer Results

Utilization:
- \( N_{\text{DSP}} \approx N_{\text{MAC}} \cdot C^{-1} = N_{\text{MAC}} \cdot \frac{f_{\text{D}}}{f_{\text{P}}} \)
- LUT, FF util. \( \lesssim \frac{1}{6} \) of available per DSP used
- No BRAM required

Frequency:
- 400 MHz – 600 MHz for up to 10k MACs
- 200 MHz – 400 MHz for up to 40k MACs
2D Pooling Layer Idea

- Fewer and simpler operations
- Not much optimization required

![Maximum Pooling](image)
2D Pooling Layer Results

- **Utilization:**
  - LUT, FF util. negligible w.r.t 2D Conv limitations
  - No DSP, BRAM

- **Frequency:**
  - Many 100 MHz for multiple thousand inputs
Example Networks

- Networks up to $\sim 50k$ MACs implemented via toolkit
- MNIST digit classification
  (scalable example image recognition task)
- Architectures include: Single/Multiple Convs, Pooling, Flattening, Single/Multiple Dense
Example Network Results: Utilization

- Utilization dominated by 2D Convs (apart from BRAM from dense layers)
- \[ N_{DSP} \approx N_{MAC} \cdot \frac{f_D}{f_P} \]
- LUT, FF util. \( \lesssim \frac{1}{6} \) of available per DSP used
- Acceptable BRAM util. by dense layers

(example network on device:
\(~ 13k\) MACs, 400 MHz
21k LUTs (\(< 2\%\)),
35k FFs (\(< 2\%\)),
1310 DSPs (\(~ 19\%\)),
166 BRAMs (\(~ 8\%)\))
Example Network Results: Frequency

- Timing closure for networks $\leq 15$ kMAC at LHC data freq.
- $\leq 30$ kMAC: $f_P > 300$ MHz
- $\leq 60$ kMAC: $f_P > 200$ MHz
**Relative Timing Closure Depending on Network Multiplication Count**

- **Target Frequency (MHz)**
  - $f_{\text{max}} / f_{\text{target}}$:
    - $\geq 1$ (timing met)
    - $\approx 0.9$
    - $\approx 0.8$
    - $\approx 0.7$
    - $\approx 0.6$
    - $\approx 0.5$
    - $\approx 0.4$

- **Multiplications**

---

Noel Nottbeck
Johannes Gutenberg-Universität Mainz
Example Network Results: Latency

- Depends on layer amount, network size
- $\leq 100 \text{ ns (4 layers, few 1k MACs)}$ –
  $\lesssim 300 \text{ ns (6 layers, multiple 10k MACs)}$
Frequency Closure Details

Actual Latency

Design Latency

Latency (ns)

Multiplications

Actual Latency

Design Latency

Multiplications

Noel Nottbeck  Johannes Gutenberg-Universität Mainz 20 / 22
Conclusion

- Networks consisting of Dense, 2D Conv/Pooling Layers implementable
- Keras network $\rightarrow$ VHDL files for inclusion into FPGA design (via Python script)
- Up to multiple 10 kMAC possible at multiple 100 MHz even at 40 MHz LHC data rate
- Improvement plans indicate possibility for even larger networks, better timing
- Paper in preparation
Thanks for your attention!

noelnottbeck@gmail.com
Motivation Example: Jet Clustering Network

- NN close to Anti-$k_t$ online (all jets)
- Outperforms even Anti-$k_t$ offline (full granularity) for close-by jets
  $\Rightarrow$ learns underlying truth information
- 117 k weights, $\sim$ 17 M MACCs
- Smaller net with $\sim$ 3 M MACCs still outperforms for close-by jets

![Graph 1](image1.png)  
![Graph 2](image2.png)

(all jets)  
(close-by jets, $\Delta R_{truth} < 0.5$)
Dense Layer Results

- **Utilization:**
  - \( N_{DSP} \approx N_I \cdot N_N \cdot \frac{f_D}{f_P} \)
  - \( N_{LUT} \ll 10/DSP \)
  - \( N_{FF} \ll 20/DSP \)
  - BRAM util. acceptable (highly depends on exact design)

- **Frequency:**
  - \(< 2k\) MACs: \( f_P \geq 640 \text{ MHz} \)
  - \( > 10k\) MACs: \( f_P > 400 \text{ MHz} \)
2D Convolution Layer Results

- **Utilization:**
  - \( N_{\text{DSP}} \approx V_I \cdot V_O \cdot \frac{f_D}{f_P} \)
  - \( N_{\text{LUT}} \lesssim 30/\text{DSP} \)
  - \( N_{\text{FF}} \lesssim 50/\text{DSP} \)
  - No BRAM util

- **Frequency:**
  - \(< 5k\) MACs: \( f_P > 400 \text{ MHz} \)
  - \(\sim 30k\) MACs: \( f_P > 200 \text{ MHz} \)

![Graph showing the relationship between Multiplications and Maximum Frequency (MHz)](image)
### 2D Pooling Layer Results

- **Utilization:**
  - \( N_{\text{LUT}} \leq 16/\text{input} \)
  - \( N_{\text{FF}} \leq 30/\text{input} \)
  - No DSP, BRAM

- **Frequency:**
  - < 2k inputs: \( f_P \geq 640 \text{ MHz} \) (mostly)
  - > 2k inputs: \( f_P > 500 \text{ MHz} \) (mostly)

---

\( f_{\text{tgt}} = 640 \text{ MHz} \)
Dense Layer Pipelining Concept

- Each cycle: Next part of inputs available, partial results propagate down the pipeline
- Inputs updated only once per data set, no multiplexing etc.
- Weights cycle during each data set
- Parallel weighting $\rightarrow$ Multiple shorter pipelines
- Neuron parallelization $\rightarrow$ Replicate structure
Dense Layer Firmware Structure

+ Control structure
Complex ... if interested in details, please ask me offline or send an e-mail
2D Pooling Layer Firmware Structure

- Input Values
- Buffer Memory
- Working Memory 0
- Row Unit 0
- Working Memory 1
- Row Unit 1
- Result Multicast
- Row Write Enable