Graph Neural Networks on FPGAs with hls4ml

Vladimir Loncar, Jan Schulte, Mia Liu, Phil Harris

Fast Machine Learning for Science workshop
September 25-28 2023
https://indico.cern.ch/event/1283970/

NSF OAC-2117997

https://a3d3.ai/
Introduction

- **Graphs** are often a natural representation of data (**nodes**) and their relation to each other (**edges**)
  - In particle or nuclear physics collider experiments, hits in **charged particle tracking** detectors can be represented as the nodes in a graph
  - Allows for single particle (e.g. particle trajectory) or event level (e.g. rare $\tau \rightarrow 3\mu$ decays) inference using **Graph Neural Networks (GNNs)**
- Use of GNNs in systems with strict latency constraints (e.g trigger systems of sPhenix or CMS) requires **FPGA** implementations
- **Support for GNNs** in tools like **hls4ml** is therefore desirable
- Missing pieces so far:
  - GNNs usually implemented in **pytorch/pytorch geometric (PyG)**, only limited pytorch support in hls4ml
  - **Missing support** for several typical operations in GNNs, such as scatter_*
- Presented today is a **prototype** for conversion and HLS code generation of a PyG GNN model in hls4ml
Improved pytorch support

- First step to support PyG model: **Improve general pytorch support in hls4ml**
- Pytorch models are **defined as classes** inheriting from a “**Module” class.”

  Operations in the model are defined in the **“forward” function**
  - Can be pytorch classes and function, but also **general python operations** such as “getitem” and “getattr”

- **torch.FX** package allows for **symbolic tracing** to capture all model operations
- Rewrote pytorch converter in hls4ml; supports a large set of NN layers
  - Included in master branch, **will enter v0.8**
  - RNN support exists in limited form in a separate branch, to be included
  - Started to work on supporting brevitas models
GNN support in hls4ml

- Parsing of GNN models in PyG can use largely the same converter.
- Extended operations supported in hls4ml based on what was needed to implement a GNN developed for track reconstruction in the sPhenix trigger
  - scatter_* operations, such as scatter_add
  - Python operations such as “getitem”
  - “gather” operations and operations such as “ones()” and “zeros()”
- For more general GNN support, need to also add support for PyG MessagePassing layers.
- Successfully converted and synthesized the sPhenix tracking GNN for the first time last week
  - Large model, had to be broken up into pieces
  - scatter_* implementation not optimized, large resources usage
sPHENIX tracking GNN hls4ml synthesis results

- Network inputs: nodes=80, edges=100
  
  - Input network
    - Can be parallelized to be “nodes” times faster (i.e., 15ns)

<table>
<thead>
<tr>
<th>Latency</th>
<th>BRAMs</th>
<th>DSPs</th>
<th>FFs</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 us</td>
<td>6.5%</td>
<td>0.3%</td>
<td>5%</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

- Edge network

<table>
<thead>
<tr>
<th>Latency</th>
<th>BRAMs</th>
<th>DSPs</th>
<th>FFs</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 us</td>
<td>15%</td>
<td>2%</td>
<td>20%</td>
<td>65%</td>
</tr>
</tbody>
</table>

- Node network (results from HLS synthesis, vivado synthesis OOM’d)
  - Need to optimize the scatter_add function (expecting ~2us for the net)

<table>
<thead>
<tr>
<th>Latency</th>
<th>BRAMs</th>
<th>DSPs</th>
<th>FFs</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 us</td>
<td>42%</td>
<td>7%</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Extremely preliminary - DO NOT TRUST NUMBERS
Outlook

- GNN implementation in hls4ml in prototype stage
  - Currently we know we support one specific model, sort of
  - Significant optimization still necessary
- Need to study how much this can be generalized to other use cases
- Different GNN models using different PyG classes will likely need some adaptation of the converter, possibly also additional HLS code
- Can not all be provided centrally by us, but we are happy to assist users in implementing their needed functionality