HLS4ML Progress Update

Parallel Computing System Lab
National Yang Ming Chiao Tung University
### Results (Reuse Factor = 1) (Green < 10us, All Can Fit KU115)

<table>
<thead>
<tr>
<th>Latency (clk period=5ns)</th>
<th>Orginal</th>
<th>Design 1</th>
<th>Design 2</th>
<th>Design 3</th>
<th>Design 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 / 37</td>
<td>185 (0.92us)</td>
<td>348 (1.74us)</td>
<td>290 (1.45us)</td>
<td>276 (1.58us)</td>
<td>216 (1.08us)</td>
</tr>
<tr>
<td>112 / 148</td>
<td>NA</td>
<td>1047 (5.24us)</td>
<td>821 (4.11us)</td>
<td>751 (3.76us)</td>
<td>523 (2.62us)</td>
</tr>
<tr>
<td>448 / 592</td>
<td>NA</td>
<td>3831 (19.16us)</td>
<td>2933 (14.67us)</td>
<td>2647 (13.24us)</td>
<td>1747 (8.74us)</td>
</tr>
<tr>
<td>1792 / 2368</td>
<td>NA</td>
<td>14967 (74.83us)</td>
<td>11381 (56.91us)</td>
<td>10231 (51.16us)</td>
<td>6643 (33.22us)</td>
</tr>
<tr>
<td>7168 / 9472</td>
<td>NA</td>
<td>59511 (297.56us)</td>
<td>45173 (225.87us)</td>
<td>long synthesis</td>
<td>long synthesis</td>
</tr>
</tbody>
</table>

### DSP / FF / LUT (%)

<table>
<thead>
<tr>
<th>Latency (clk period=5ns)</th>
<th>Orginal</th>
<th>Design 1</th>
<th>Design 2</th>
<th>Design 3</th>
<th>Design 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 / 37</td>
<td>305%/64%/378%</td>
<td>6% / 1% / 0%</td>
<td>6% / 1% / 0%</td>
<td>8% / 1% / 1%</td>
<td>8% / 1% / 1%</td>
</tr>
<tr>
<td>112 / 148</td>
<td>NA</td>
<td>6% / 1% / 0%</td>
<td>6% / 1% / 0%</td>
<td>8% / 1% / 1%</td>
<td>8% / 2% / 3%</td>
</tr>
<tr>
<td>448 / 592</td>
<td>NA</td>
<td>6% / 1% / 0%</td>
<td>6% / 1% / 1%</td>
<td>8% / 2% / 4%</td>
<td>8% / 5% / 12%</td>
</tr>
<tr>
<td>1792 / 2368</td>
<td>NA</td>
<td>6% / 1% / 0%</td>
<td>6% / 1% / 2%</td>
<td>8% / 6% / 9%</td>
<td>8% / 11% / 34%</td>
</tr>
<tr>
<td>7168 / 9472</td>
<td>NA</td>
<td>6% / 1% / 0%</td>
<td>6% / 2% / 2%</td>
<td>long synthesis</td>
<td>long synthesis</td>
</tr>
</tbody>
</table>
Design 1 (Adjust Pipeline/Partition to Attain Efficient Design)

- Adjust Loop Pipeline to Avoid Resource Explosion
  a. Change pipeline pragma from top function to inner loop.
  b. Pipeline in top function will unroll all loops in the hierarchy below[1].
  c. Unroll cause excessive resource usage and makes the pipeline directives in inner loop functionless.

- Array Partition and Reshape to Attain Efficient Resource Usage
  a. Don’t need all instances in arrays available at the same time.
     - Too many sub-arrays would cost a lot of LUTs.
  b. We only partition array in dimension = 2 (feature dimension).
     - Partition in dim=2 can enable concurrent accesses to all features of one node or edge

Design 2 (Reduce Latency of BRAM Reset)

Remove BRAM Reset for Lower Latency

- **Original Reset (with BRAM reset)**
  a. Two relational model, and each takes \((n_{node} \times n_{out})\) cycles to reset aggregation BRAM.
  b. It takes total \((2 \times n_{node} \times n_{out})\) cycles to reset.

- **Improved Reset (without BRAM reset, using additional flag)**
  a. Reset operation can be replaced by initial element assignment.
  b. So, combine reset with accumulation.
  c. Using additional flag to mark which elements are initialized.
     - not initialized \(\rightarrow\) direct assignment (replaces reset)
     - initialized \(\rightarrow\) accumulate
Design 3 (Remove Latency of Carried Dependency)

Remove Latency of Carried Dependency

- **Original Relational_loop**
  - a. Read/Write operate on same memory port which induces carried dependency.
  - b. All layers in relational_loop are forced to operate at Interval Cycle=2.

- **Improved Relational_loop**
  - a. Use local registers to store results of simultaneously read/write.
  - b. Reduce relational_loop Interval Cycle from 2 to 1.
Design 4 ("Dataflow" to Pipeline Functions)

- Add Dataflow Directive to Reduce Latency
  a. "Dataflow" changes two sequentially executed functions into a pipeline flow

- Design3 Relational_model
  a. Write back to global memory can’t start until relational_loop complete.
  b. Waste the advantage of pipeline.

- Design4 Relational_model
  a. Vivado_HLS uses shift registers to overlap relational_loop and writeback_loop.
  b. Aggregation and writeBack work concurrently.
  c. Reduce roughly 35% total latency.
Summary and Questions

• **Summary of Our Progress**
  a. Try to take advantages of Vivado HLS flow (Design 1 ~ Design 4)
     ■ Same data flow and accelerator architecture
  b. Working on performance profiling and possible architecture refinement

• **Questions**
  a. What is the current system architecture?
     ■ Input sources, output target, etc.
      • Sensors → FPGA/DRAM? → minotr/control?
     ■ Data storage of FPGA? (DRAM, SSD, etc)
      • Do we need to consider extra data access time, such as DRAM access time, according to the system architecture?
  b. More testing data
     ■ For larger graphs
     ■ Quality test for optimizations (networks, quantization, pruning, etc)