Neural Networks are “Hot Hot Hot!!!”

- Neural networks are layered structures consisting of neurons useful for:
  - Classification
  - Regression
  - Problems with difficult to discern solutions
- They have broad use in high tech and big data, e.g., Google [1]
- They also enable alternative computing methodologies
  - Automatic parallelization, e.g., ASC [2]
  - Approximate computing, e.g., NPU [3]

Our Vision of General-Purpose Neural Network Computing

1. Treat neural networks as a functional primitive backed by hardware acceleration [1]
2. Follow a transaction model for neural network computation
3. Exploit anticipated sharing of neural networks across applications
   - We think about neural acceleration like floating point acceleration
   - We break this into two distinct contributions:
     - X-FILES – Software/Hardware extensions for transaction management
     - DANA – One possible backend accelerator that interfaces with the X-FILES

Safe Multi-Transaction Management using ASIDs

- Processes grouped into address spaces, identified with ASIDs (Figure 2 left)
- An ASID defines a set of processes that share neural network configurations, identified with NNIDs
- An OS-managed ASID–NNID Table enables NNID dereferencing (Figure 2 right)
- All transparently handled by the OS

X-FILES: Software/Hardware Extensions

- A defined user and supervisor interface for neural networks (Table 1)
- Uses ASIDs and NNIDs to allow access to shared neural networks
- Relies on a Hardware arbiter to maintain architectural state (Figure 3)
- Currently supports a subset of FANN [1]

DANA: A Multi-Transaction Accelerator

- A backend accelerator aligning with our multi transaction vision (Figure 3)
- Understands a FANN derivative binary description of a neural network
- Implemented in Chisel [1]
- Supporting fixed point feedforward and gradient descent learning computations

Usage with a RISC-V Microprocessor

1. Grab a Rocket Chip RISC-V Microprocessor [1]
2. Build a RISC-V toolchain
3. Grab a copy of our X-FILES/DANA accelerator [2]
4. Build an FPGA configuration for Rocket + X-FILES/DANA
5. User processes can safely throw transactions at X-FILES
   - With support for feedforward and learning computation
   - Figure 4 shows three separate FPGAs all running Rocket + X-FILES/DANA

Open Source Plans and Acknowledgments

- Remaining Items
  - Linux kernel integration
  - Support for asynchronous data transfer
- This work was supported by the following:
  - A NASA Space Technology Research Fellowship
  - An NSF Graduate Research Fellowship
  - NSF CAREER awards
  - A Google Faculty Research Award

Open Source Availability

- Should be ready by the end of February
- On GitHub (soon!): github.com/bu-icsg/xfiles-dana

Table 1: X-FILES software library functions for communication with the X-FILES arbiter and managing the ASID–NNID Table

<table>
<thead>
<tr>
<th>Function</th>
<th>User/Supervisor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tid = newWriteRequest(asid, learningType, numOutputs)</td>
<td>user</td>
<td>Initiate a new transaction returning a TID</td>
</tr>
<tr>
<td>writeData(tid, *inputs, num_input)</td>
<td>user</td>
<td>For register mode, write input data and, optionally, training data</td>
</tr>
<tr>
<td>readDataSpinlock(tid, *output, num_output)</td>
<td>user</td>
<td>For register mode, try to read output data until successful</td>
</tr>
<tr>
<td>tidKill(tid)</td>
<td>user</td>
<td>Kills an executing transaction</td>
</tr>
<tr>
<td>oldTid = setAsid(asid)</td>
<td>supervisor</td>
<td>Change the ASID returning the old TID to the OS for storage</td>
</tr>
<tr>
<td>setAsid(*table)</td>
<td>supervisor</td>
<td>Set the ASID–NNID Table Pointer and the number of ASIDs</td>
</tr>
<tr>
<td>asidNNIDTableCreate(**table, numAsid, numNNid)</td>
<td>supervisor</td>
<td>ASID–NNID Table constructor</td>
</tr>
<tr>
<td>asidNNIDTableDestroy(**table)</td>
<td>supervisor</td>
<td>ASID–NNID Table destructor</td>
</tr>
<tr>
<td>nnid = addNnid(**table, asid, *nnConfiguration)</td>
<td>supervisor</td>
<td>Adds an NN Configuration to an existing ASID–NNID Table</td>
</tr>
<tr>
<td>removeNnid(**table, nnid)</td>
<td>supervisor</td>
<td>Remove a specific NNID from an existing ASID–NNID Table</td>
</tr>
</tbody>
</table>

Boston University

Schuyler Eldridge1 (schuye@bu.edu)  Tommy Unger2  Marcia Sahaya Louis1
Amos Waterland3  Margo Seltzer3  Jonathan Appavoo2  Ajay Joshi1
1 Boston University Department of Electrical and Computer Engineering
2 Boston University Department of Computer Science
3 Harvard University School of Engineering and Applied Sciences


