Neural Network-Based Accelerators for Transcendental Function Approximation

Schuyler Eldridge*  Florian Raudies†  David Zou*
Ajay Joshi*

*Department of Electrical and Computer Engineering, Boston University
†Center for Computational Neuroscience and Neural Technology, Boston University
schuye@bu.edu

May 22, 2014
This work was supported by a NASA Office of the Chief Technologist’s Space Technology Research Fellowship.
Figure 1: Trends in CMOS technology [Moore et al., 2011 Salishan]
Accelerators to the Rescue?

Energy Efficient Accelerators...
- Lessen the utilization crunch of Dark Silicon
- Are cheap due to plentiful transistor counts
- Are typically special-purpose

Approaches to General Purpose Acceleration
- QsCores – Dedicated hardware for frequent code patterns [Venkatesh et al., 2011 MICRO]
- NPU – Neural network-based approximation of code regions [Esmaeilzadeh et al., 2012 MICRO]
Neural Networks (NNs) as General-Purpose Accelerators

The good and the bad...

- NNs are general-purpose approximators [Cybenko, 1989 *Math. Control Signal*, Hornik, 1991 *Neural Networks*

- But... NNs are still approximate

Approximation may be acceptable

- Modern recognition, mining, and synthesis (RMS) benchmarks are robust [Chippa et al., 2013 *DAC*]
Library-Level Approximation with NN-Based Accelerators

Big Idea

- Use NNs to approximate library-level functions
  - cos, exp, log, pow, and sin
- Explore the design space of NN topologies
  - Define and use an energy–delay–error product (EDEP) metric
- Evaluate energy–performance improvements
  - Use an energy–delay product (EDP) metric
- Evaluate accuracy of...
  - NN-based accelerators vs. a traditional approach
  - Applications using NN-based accelerators
Multilayer Perceptron (MLP) NN Primer

Figure 2: NN with $i \times h \times o$ nodes.

Equations

$$y = \phi \left( \sum_{k=1}^{n} x_k w_k \right)$$

$$\phi_{\text{sigmoid}} = \frac{1}{1 + e^{-2x}}$$

$$\phi_{\text{linear}} = x$$
NN-Based Approximation Requires Input–Output Scaling

Approximating Unbounded Functions on Bounded Domains

- NNs cannot handle unbounded inputs
- Input–output scaling can extend the effective domain and range of the approximated function
- This approach is suitable when...
  - A small region is representative of the whole function
  - There exist easy\(^a\) operations to scale inputs and outputs
- Specifically, we use the CORDIC [Volder, 1959 *IRE Tran. Comput.*] scalings identified by Walther [Walther, 1971 *AFIPS*]

\(^a\)By “easy”, I mean multiplication with a constant, addition, bitshifts, and rounding.
Walther’s Scaling Approach [Walther, 1971 AFIPS] for \( \exp x \)

**Scaling Steps**

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

**Similar Scalings Exist**
- \( \cos x \) and \( \sin x \)
- \( \log x \)

**Equations**

\[
\exp(q \log 2 - d) = 2^q \exp(-d) \\
q = \left\lfloor \frac{x}{\log 2} + 1 \right\rfloor \\
d = x - q \log 2
\]

**Figure 4:** Graphical scaling for \( \exp x \)
Walther’s Scaling Approach [Walther, 1971 AFIPS] for $\exp x$

Scaling Steps
1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist
- $\cos x$ and $\sin x$
- $\log x$

Figure 4: Graphical scaling for $\exp x$
Walther’s Scaling Approach [Walther, 1971 AFIPS] for $\exp x$

**Scaling Steps**
1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

**Similar Scalings Exist**
- $\cos x$ and $\sin x$
- $\log x$

**Formula**
\[
\exp(q \log 2 - d) = 2^q \exp(-d)
\]
\[
q = \left\lfloor \frac{x}{\log 2} + 1 \right\rfloor
\]
\[
d = x - q \log 2
\]
\[
\hat{x} = q \log 2 - d
\]

**Figure 4:** Graphical scaling for $\exp x$
Walther’s Scaling Approach [Walther, 1971 AFIPS] for $\exp x$

$\exp(q \log 2 - d) = 2^q \exp(-d)$

$q = \left\lfloor \frac{x}{\log 2} + 1 \right\rfloor$

$d = x - q \log 2$

$\hat{x} = q \log 2 - d$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Figure 4: Graphical scaling for $\exp x$
Walther’s Scaling Approach [Walther, 1971 AFIPS] for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Figure 4: Graphical scaling for $\exp x$

$\exp(q \log 2 - d) = 2^q \exp(-d)$

$\hat{x} = q \log 2 - d$

$\hat{x} = \lfloor x \log 2 + 1 \rfloor$

$d = x - q \log 2$

$\hat{x} = q \log 2 - d$

$2^q \exp_{NN}(-d)$
Walther’s Scaling Approach [Walther, 1971 AFIPS] for \( \exp x \)

**Scaling Steps**

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

**Similar Scalings Exist**

- \( \cos x \) and \( \sin x \)
- \( \log x \)

**Figure 4:** Graphical scaling for \( \exp x \)
Fixed Point Accelerator Architecture for $1 \times 3 \times 1$ NN

Figure 5: Block diagram of an NN-based accelerator architecture
### Candidate NN Topologies
- Fixed point
- 1–15 hidden nodes
- 6–10 fractional bits

### NN Evaluation Criteria
- Energy
- Performance
- Accuracy

### Energy–Delay–Error Product (EDEP)
- Optimal NN topology minimizes EDEP

\[
EDEP = \text{energy} \times \frac{\text{latency in cycles}}{\text{frequency}} \times \text{mean squared error}
\]
### NN Topology Evaluation – Results

<table>
<thead>
<tr>
<th>Func.</th>
<th>NN</th>
<th>MSE ($\times 10^{-4}$)</th>
<th>Energy (pJ)</th>
<th>Area (um$^2$)</th>
<th>Freq. (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos</td>
<td>h1_b6</td>
<td>9</td>
<td>8</td>
<td>1300</td>
<td>340</td>
</tr>
<tr>
<td>sin</td>
<td>h1_b6</td>
<td>7</td>
<td>8</td>
<td>1300</td>
<td>340</td>
</tr>
<tr>
<td>exp</td>
<td>h3_b7</td>
<td>2</td>
<td>25</td>
<td>3600</td>
<td>340</td>
</tr>
<tr>
<td>log</td>
<td>h3_b7</td>
<td>1</td>
<td>25</td>
<td>3600</td>
<td>340</td>
</tr>
<tr>
<td>pow</td>
<td>h3_b7</td>
<td>432</td>
<td>102</td>
<td>3600</td>
<td>340</td>
</tr>
</tbody>
</table>

**Evaluation Notes**

- Evaluated with a 45nm predictive technology model (PTM)
- $a^b = \exp (b \log a)$
NN Topology Evaluation Results

Figure 6: NN-based functions and their errors. Note: Error is plotted on a log scale using the right y axis.

Evaluation Notes
- Functions well approximated by their NNs
- Due to input–output scaling, error is proportional to output value
Evaluation Approach

**Approach – Energy**
- Determine traditional glibc instruction breakdown
- Determine energy/instruction in 45nm PTM
- Determine glibc energy/function
- Compare traditional and NN-based execution using EDP

**Approach – Accuracy**
- Replace all transcendental function calls with NNs
- Evaluate application output accuracy
## Traditional glibc Instruction Breakdown

**Table 2:** Mean floating point instruction counts.

<table>
<thead>
<tr>
<th>Func.</th>
<th>addsd</th>
<th>addss</th>
<th>mulsd</th>
<th>mulss</th>
<th>subsd</th>
<th>subss</th>
<th>Total Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos</td>
<td>7</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>115</td>
</tr>
<tr>
<td>cosf</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>7</td>
<td>103</td>
</tr>
<tr>
<td>exp</td>
<td>11</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>expf</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>218</td>
</tr>
<tr>
<td>log</td>
<td>18</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>227</td>
</tr>
<tr>
<td>logf</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>4</td>
<td>143</td>
</tr>
<tr>
<td>pow</td>
<td>32</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>338</td>
</tr>
<tr>
<td>powf</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>26</td>
<td>355</td>
</tr>
<tr>
<td>sin</td>
<td>8</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>109</td>
</tr>
<tr>
<td>sinf</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>5</td>
<td>97</td>
</tr>
</tbody>
</table>

### Abbreviations Used

- **ss** or, e.g., `cosf` ≡ single precision
- **sd** or, e.g., `cos` ≡ double precision
### Traditional glibc $\textit{energy/instruction}$

#### Table 3: Parameters of traditional glibc implementations of floating point instructions.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Area ($\text{um}^2$)</th>
<th>Freq. (MHz)</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>addss</td>
<td>640</td>
<td>390</td>
<td>1</td>
</tr>
<tr>
<td>addsd</td>
<td>1500</td>
<td>390</td>
<td>2</td>
</tr>
<tr>
<td>mulss</td>
<td>6500</td>
<td>280</td>
<td>36</td>
</tr>
<tr>
<td>mulsd</td>
<td>16200</td>
<td>140</td>
<td>80</td>
</tr>
</tbody>
</table>

#### Evaluation Notes
- Evaluated in the NCSU 45nm predictive technology model
- For scale, one NN-based exp function uses 25 pJ
- Latency of one cycle
### Table 4: Mean floating point energy

<table>
<thead>
<tr>
<th>Function</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos</td>
<td>967</td>
</tr>
<tr>
<td>cosf</td>
<td>365</td>
</tr>
<tr>
<td>exp</td>
<td>1158</td>
</tr>
<tr>
<td>expf</td>
<td>453</td>
</tr>
<tr>
<td>log</td>
<td>995</td>
</tr>
<tr>
<td>logf</td>
<td>415</td>
</tr>
<tr>
<td>pow</td>
<td>2561</td>
</tr>
<tr>
<td>powf</td>
<td>1292</td>
</tr>
<tr>
<td>sin</td>
<td>909</td>
</tr>
<tr>
<td>sinf</td>
<td>311</td>
</tr>
</tbody>
</table>

**Observation**

- Energy consumption is 2 orders of magnitude higher than NN-based implementation.
Table 5: NN-based EDP is significantly lower than glibc. Data is normalized to sin EDP, $3 \times 10^{-19}$.

<table>
<thead>
<tr>
<th>Func.</th>
<th>EDP-NN</th>
<th>EDP-Single</th>
<th>EDP-Double</th>
</tr>
</thead>
<tbody>
<tr>
<td>cos</td>
<td>1</td>
<td>55</td>
<td>161</td>
</tr>
<tr>
<td>exp</td>
<td>4</td>
<td>1052</td>
<td>269</td>
</tr>
<tr>
<td>log</td>
<td>4</td>
<td>86</td>
<td>328</td>
</tr>
<tr>
<td>pow</td>
<td>31</td>
<td>666</td>
<td>1256</td>
</tr>
<tr>
<td>sin</td>
<td>1</td>
<td>44</td>
<td>144</td>
</tr>
</tbody>
</table>

Table 6: Applications that spend most of their cycles computing transcendental functions see large EDP improvements.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Transcendental Cycles</th>
<th>Single</th>
<th>Double</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>46%</td>
<td>56%</td>
<td>55%</td>
</tr>
<tr>
<td>swaptions</td>
<td>39%</td>
<td>62%</td>
<td>61%</td>
</tr>
<tr>
<td>bodytrack</td>
<td>2%</td>
<td>98%</td>
<td>98%</td>
</tr>
<tr>
<td>canneal</td>
<td>1%</td>
<td>99%</td>
<td>99%</td>
</tr>
</tbody>
</table>

Approximating Transcendental Functions

- Energy-delay product is $68 \times$ lower vs. glibc
- Mean squared error is $9 \times 10^{-3}$
- Application improvements follow Amdahl’s law
NN-Based Accelerators in Applications – Accuracy

Table 7: Application output MSE and percent error using NN-based accelerators.

| Benchmark     | MSE ($\times 10^{-1}$) | $E[|\% error|]$ |
|---------------|------------------------|-----------------|
| blackscholes  | 4.00                   | 25%             |
| bodytrack     | 2.00                   | 30%             |
| ferret        | 0.01                   | 2%              |
| swaptions     | 60.00                  | 37%             |
| canneal       | $2.89 \times 10^8$     | 0.0025%         |

MSE and Percent Error

- Qualitatively low error
- canneal has 1 large output, hence high MSE and low percent error
Accelerators demonstrate EDP reductions...
- 68x lower EDP than glibc
- 78% of the EDP of traditional applications

Library-level approximation is a suitable target for NN-based acceleration

Work in this area can be improved by enabling NN-based accelerators to approximate additional functions and applications through...
- Extensions to additional libraries
- Capabilities to automatically identify and approximate functions
Appendix Contents

1 References

Venkatesh, G. et al. (2011). Qscores: trading dark silicon for scalable energy efficiency with quasi-specific cores. In MICRO.

Esmaeilzadeh, H. et al. (2012). Neural acceleration for general-purpose approximate programs. In MICRO.


