Compilation and Hardware Support for Approximate Acceleration

Thierry Moreau, Adrian Sampson, Andre Baixo, Mark Wyse, Ben Ransford, Jacob Nelson, Hadi Esmaeilzadeh (Georgia Tech), Luis Ceze and Mark Oskin
University of Washington
moreau@uw.edu

Theme: 2384.004
Approximate Computing

Aims to exploit application resilience to trade-off quality for efficiency
Approximate Computing
Approximate Computing

- ✔ Accurate
- ✗ Expensive
- ✔ Approximate
- ✔ Cheap
Neural Networks as Approximate Accelerators

Esmaeilzadeh et al.
[MICRO 2012]
Neural Acceleration

float foo (float a, float b) {
    ...
    return val;
}

approximation  →  NPU

acceleration
Neural Acceleration

float foo (float a, float b)
{
  ...
  return val;
}
Neural Acceleration

float foo (float a, float b) {
    ...
    return val;
}

*Moreau et. al [HPCA2015]
Neural Acceleration

3.8x speedup and 2.8x efficiency - 10% error

float foo (float a, float b)
{
    ...
    return val;
}
Talk Outline

Introduction

Compiler Support with ACCEPT

SNNAP Accelerator design

Evaluation & Comparison with HLS
Compilation Overview

1. Region detection

code
annotation
Compilation Overview

1. Region detection
   code
   annotation

ACCEPT
region detection & program instrumentation
Compilation Overview

1. Region detection
   code annotation
   region detection & program instrumentation

2. ANN Training
   [training.data] → back prop. & topology search

ACCEPT

Thierry Moreau
Compilation Overview

1. Region detection
   code annotation
   region detection & program instrumentation

2. ANN Training
   [training.data] → back prop. & topology search → ACCEPT

3. Code Generation
   code transformation
   executes SNNAP CPU

Thierry Moreau
Compilation Overview

1. Region detection
   - code annotation

2. ANN Training
   - [training.data] → back prop. & topology search
   - ACCEPT: region detection & program instrumentation

3. Code Generation
   - ACCEPT: code transformation
   - executes

Thierry Moreau
Compilation Overview

1. Region detection
   - code annotation
   - region detection & program instrumentation

2. ANN Training
   - [training.data]
   - back prop. & topology search

3. Code Generation
   - code transformation
   - executes SNNAP on CPU

Thierry Moreau
Programming Model

float sobel (float* p);

float** src;
float** dst;

while (true) {
    src = read_from_camera();
    for (y=0; y < h; ++y) {
        for (x=0; x < w; ++x) {
            dst[y][x] = sobel(& src[y][x]);
        }
    }
    display(dst);
}
Programming Model

```c
APPROX float sobel (APPROX float* p);

... 

APPROX float** src;
APPROX float** dst;

while (true) {
    src = read_from_camera();
    for (y=0; y < h; ++y) {
        for (x=0; x < w; ++x) {
            dst[y][x] = sobel(& src[y][x]);
        }
    }
    display(ENDORSE(dst));
}
```
Programming Model

```c
APPROX float sobel (APPROX float* p);
...

APPROX float** src;
APPROX float** dst;

while (true) {
    src = read_from_camera();
    for (y=0; y < h; ++y) {
        for (x=0; x < w; ++x) {
            dst[y][x] = sobel(& src[y][x]);
        }
    }
    display(ENDORSE(dst));
}
```

✅ no side effects
✅ executes often
Checking for Quality

annotated program

sobel.c
Checking for Quality

annotated program

sobel.c

quality metric

\( d(y, y') \)
Checking for Quality

annotated program: sobel.c

quality metric: $d(y, y')$

input data
Checking for Quality

annotated program

sobel.c

quality metric

\( d(y, y') \)

input data

test

training
Checking for Quality

annotated program

sobel.c

quality metric

d(y, y')

input data

test

training
Talk Outline

Introduction

Compiler Support with ACCEPT

SNNAP Accelerator design

Evaluation & Comparison with HLS
Background: Multi-Layer Perceptrons

neural network

computing a single layer

activation function $f$

$$
\begin{bmatrix}
x_7 \\
x_8 \\
x_9 \\
\end{bmatrix} = f \begin{bmatrix}
w_{67} & w_{57} & w_{47} \\
w_{68} & w_{58} & w_{48} \\
w_{69} & w_{59} & w_{49} \\
\end{bmatrix} \begin{bmatrix}
x_6 \\
x_5 \\
x_4 \\
\end{bmatrix}
$$
Background: Systolic Arrays

computing a single layer

\[
\begin{bmatrix}
    x_7 \\
    x_8 \\
    x_9
\end{bmatrix}
= f
\begin{bmatrix}
    W_{67} & W_{57} & W_{47} & x_6 \\
    W_{68} & W_{58} & W_{48} & x_5 \\
    W_{69} & W_{59} & W_{49} & x_4
\end{bmatrix}
\]
PU Micro-Architecture

systolic array

processing unit

PU
control

Storage

PE
PE
PE
f

x_6
x_5
x_4

W_{49} \ W_{48} \ W_{47}

W_{59} \ W_{58} \ W_{57}

W_{69} \ W_{68} \ W_{67}

f

x_7
x_8
x_9

31
PU Micro-Architecture

systolic array

processing unit

1 - processing elements in DSP logic

PU

control

Storage

PE

PE

PE

PE

f

X6

X5

X4

W49  W48  W47

W59  W58  W57

W69  W68  W67

X7

X8

X9
PU Micro-Architecture

systolic array  

processing unit  

X6  
X5  
X4  
W49 W48 W47  
W59 W58 W57  
W69 W68 W67  

1 - processing elements in DSP logic  
2 - local storage for synaptic weights  

Thierry Moreau
PU Micro-Architecture

systolic array

processing unit

1 - processing elements in DSP logic

2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions
PU Micro-Architecture

systolic array

1 - processing elements in DSP logic

processing unit

2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions

4 - vertically micro-coded sequencer

Thierry Moreau
Multi-Processing Units
CPU-SNNAP Integration

- Coherent reads & writes with accelerator coherency port
- Custom mastering interface
- Low-latency event signaling, sleep & wakeup
- $L2$
- $L1$
- CPU
- DMA master
- Scheduler
- Bus
- PU

Thierry Moreau
Talk Outline

Introduction

Programming model

SNNAP design:

- Efficient neural network evaluation
- Low-latency communication

Evaluation & Comparison with HLS
Evaluation

Neural acceleration on SNNAP (8x8 configuration, clocked at 1/4 of $f_{\text{CPU}}$) vs. precise CPU execution

<table>
<thead>
<tr>
<th>application</th>
<th>domain</th>
<th>error metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>option pricing</td>
<td>MSE</td>
</tr>
<tr>
<td>fft</td>
<td>DSP</td>
<td>MSE</td>
</tr>
<tr>
<td>inversek2j</td>
<td>robotics</td>
<td>MSE</td>
</tr>
<tr>
<td>jmeint</td>
<td>3D-modeling</td>
<td>miss rate</td>
</tr>
<tr>
<td>jpeg</td>
<td>compression</td>
<td>image diff</td>
</tr>
<tr>
<td>kmeans</td>
<td>ML</td>
<td>image diff</td>
</tr>
<tr>
<td>sobel</td>
<td>vision</td>
<td>image diff</td>
</tr>
</tbody>
</table>
Whole-Application Speedup

<table>
<thead>
<tr>
<th>Application</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>bscholes</td>
<td>10.8</td>
</tr>
<tr>
<td>fft</td>
<td>38.1</td>
</tr>
<tr>
<td>inversek2j</td>
<td>2.7</td>
</tr>
<tr>
<td>jmeint</td>
<td>1.5</td>
</tr>
<tr>
<td>jpeg</td>
<td>2.3</td>
</tr>
<tr>
<td>kmeans</td>
<td>1.3</td>
</tr>
<tr>
<td>sobel</td>
<td>2.4</td>
</tr>
<tr>
<td>GEOMEAN</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Energy Savings

Energy = Power on (DRAM + SoC) * Runtime +36%

bscholes: 7.8
fft: 2.2
inversek2j: 28.0
jmeint: 1.1
jpeg: 1.7
kmeans: 0.9
sobel: 1.8
GEOMEAN: 2.8
float foo (float a, float b) {
    ...
    return val;
}
Conclusion

float foo (float a, float b) {
    ... return val;
}

compiler-support

approximation

acceleration

ACCEPT

HW-support

NPU

Thierry Moreau
Conclusion

3.8x speedup & 2.8x energy savings

float foo (float a, float b)
{
    ...
    return val;
}
Compilation and Hardware Support for Approximate Acceleration

Thierry Moreau, Adrian Sampson, Andre Baixo, Mark Wyse, Ben Ransford, Jacob Nelson, Luis Ceze and Mark Oskin
University of Washington
moreau@uw.edu

ACCEPT: http://accept.rocks    SNNAP: upon request