Towards Statistical Guarantees in Controlling Quality Tradeoffs for Approximate Acceleration

Divya Mahajan
Amir Yazdanbakhsh
Jongse Park
Bradley Thwaites
Hadi Esmaeilzadeh

Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology
Approximate Computing

Relax the abstraction of “near perfect” accuracy in

Data Processing  
Storage  
Communication

Accept imprecision to improve performance energy efficiency
Approximate Acceleration
Approximate Acceleration
Fixed Error
Lack of Error Control
Lack of Guarantees
Fixed Error
Lack of Error Control
Lack of Guarantees
To tackle these shortcomings we devise MITHRA
Overview

Motivation

Challenges in devising MITHRA

A hardware software solution

Detailing the components of MITHRA
Exploiting Accelerator Characteristics

Approximate Accelerator

Application

Percentage of Output Elements vs. Error

- blackscholes
- fft
- jmeint
- inversek2j
- jpeg
- sobel
Challenges

How to eliminate anomalous invocations?

Approximate Accelerator

Local Error

Final Quality loss

Application
Factors Influencing Error

Accelerator Error = | Output_{accelerator} - Output_{original} |

Output_{accelerator} = f (accelerator inputs, accelerator configuration)
Factors Influencing Error

Accelerator Error = \[ | \text{Output}_{\text{accelerator}} - \text{Output}_{\text{original}} | \]

\[
\text{Output}_{\text{accelerator}} = f(\text{accelerator inputs, accelerator configuration})
\]
Factors Influencing Error

\[ \text{Accelerator Error} = | \text{Output}_{\text{accelerator}} - \text{Output}_{\text{original}} | \]

\[ \text{Output}_{\text{accelerator}} = f (\text{accelerator inputs, accelerator configuration}) \]

\[ \text{Output}_{\text{accelerator}} = f (\text{accelerator inputs}) \]
Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs
Challenges

Approximate Accelerator

Local Error

Final Quality loss

Application

Final Quality loss → Local Error ?
Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error?

Threshold the local error
Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error?

Threshold the local error

What guarantees?
Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error?

Threshold the local error

What guarantees?

Statistical Guarantees
Challenges

- How to eliminate anomalous invocations?
- Look at the accelerator inputs
- Final Quality loss → Local Error?
- Threshold the local error
- What guarantees?
- Statistical Guarantees
- What algorithm at runtime?
Challenges

- How to eliminate anomalous invocations?
- Final Quality loss → Local Error?
- What guarantees?
- What algorithm at runtime?
- Look at the accelerator inputs
- Threshold the local error
- Statistical Guarantees
- Classification in Hardware
MITHRA: A Hardware/Software Solution

Approximate Accelerator

Desired quality requirements

Statistical Optimizer

Generate the threshold for the local error such that the final quality loss meets the requirements

Input Datasets
MITHRA: A Hardware/Software Solution

Approximate Accelerator

Desired quality requirements

Input Datasets

Statistical Optimizer

Classifier Trainer

Generate training data that segregates inputs that give > and < (th)
MITHRA: A Hardware/Software Solution

Approximate Accelerator

Desired quality requirements

Application

Input Datasets

Statistical Optimizer

Classifier Trainer

Hardware Classifier Topology
MITHRA: A Hardware/Software Solution

Approximate Accelerator

Application

Desired quality requirements

Input Datasets

Statistical Optimizer

Classifier Trainer

Hardware Classifier Topology

Application
Statistical Optimizer

Input Datasets

Application

Approximate Accelerator
Statistical Optimizer

Input Datasets

Application

Approximate Accelerator

Precise Result
Statistical Optimizer

Input Datasets

Application

Approximate Accelerator

Local error
Statistical Optimizer

**Input Datasets**

Application

Approximate Accelerator

Local error

Threshold

\( \geq \text{th} \)
Local error > th
Local error < th \rightarrow \text{Approximate Accelerator}
Approximate Accelerator

Desired Quality Loss by the programmer

Final Quality loss

Input Datasets

Application
Binomial Proportion Confidence Interval

Input Datasets

- Less than the desired programmer quality loss → nsuccess
- Greater than the desired programmer quality loss
Binomial Proportion Confidence Interval

\[
\frac{1}{1 + \frac{(n_{\text{trials}} - n_{\text{success}} + 1)}{n_{\text{success}} \times F[1 - \alpha; 2n_{\text{success}}, 2(n_{\text{trials}} - n_{\text{success}} + 1)]}} < \text{SuccessRate}
\]

with a confidence level
Example

\((n_{\text{trials}}, n_{\text{success}}) \Rightarrow r < \text{SuccessRate with a confidence level}\)

E.g., \((100, 80) \Rightarrow 72\% < \text{SuccessRate with 95\% confidence level}\)
Example

\((n_{\text{trials}}, n_{\text{success}}) \rightarrow r < \text{SuccessRate} \quad \text{with a confidence level}\)

E.g., \((100, 80) \rightarrow 72\% < \text{SuccessRate} \quad \text{with 95\% confidence level}\)

**Final Quality Level, Success Rate and Confidence Interval** programmer specified
Statistical Optimization

if desired metrics are not met:

\[ t_{t+1} = t_t - \Delta \]

else if desired metrics are met:

\[ t_{t+1} = t_t + \Delta \]

Reiterate; till the \( t_t \) meets the metrics but \( t_{t+1} \) doesn't
Statistical Optimization

if desired metrics are not met:
\[ th_{t+1} = th_t - \Delta \]
else if desired metrics are met:
\[ th_{t+1} = th_t + \Delta \]

Reiterate; till the \( th_t \) meets the metrics but \( th_{t+1} \) doesn't

Tighter threshold better Final Quality Loss Level, Success Rate and Confidence Interval but lower benefits from approximation
Training the Classifiers

The training data used to generate classifier topology
Hardware Classifiers

Simple algorithm that can be easily implemented in hardware.

We use two techniques for this work:
1. Table Based
2. Neural Network Based
Table-based Classifiers

Classifier

Accelerator Inputs

Multiple Input Signature Register (MISR)

Input Signature

Precise

1
0
1
0
1
1
0
1
A small **ensemble** of table-based classifiers achieve better accuracy and performance.
Neural Network Based Classifiers
# Benchmarks

<table>
<thead>
<tr>
<th>Accelerator Topology</th>
<th>Baseline Error</th>
<th>Neural Classifier Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blackscholes</strong></td>
<td>6 → 8 → 8 → 1</td>
<td>6.02%</td>
</tr>
<tr>
<td><strong>FFT</strong></td>
<td>1 → 4 → 4 → 2</td>
<td>7.22%</td>
</tr>
<tr>
<td><strong>Inversek2j</strong></td>
<td>2 → 8 → 2</td>
<td>7.50%</td>
</tr>
<tr>
<td><strong>JMEINT</strong></td>
<td>18 → 32 → 8 → 2</td>
<td>17.69%</td>
</tr>
<tr>
<td><strong>JPEG Encoding</strong></td>
<td>64 → 16 → 64</td>
<td>7.00%</td>
</tr>
<tr>
<td><strong>Sobel</strong></td>
<td>9 → 8 → 1</td>
<td>9.96%</td>
</tr>
</tbody>
</table>

**Table Classifier Topology** 8 tables each of size 0.5 KB
Energy and Performance Benefits

![Graph showing energy and performance benefits across different application quality loss percentages. The graph compares the performance of Oracle, Table-based, and Neural models. The x-axis represents application quality loss (0.0% to 10.0%), and the y-axis shows speedup and energy reduction.]
Invocation Rate

![Graph showing Invocation Rate vs Application Quality Loss]

- **Oracle**
- **Table-based**
- **Neural**
Varying Success Rate
Conclusion

Hardware software co-design works well

Aims to make statistical guarantees a norm
Thank you
Thank you
False Positive and False Negative Results

- **False Negative—Table-based**
- **False Positive—Table-based**

- **False Negative—Neural**
- **False Positive—Neural**

Application Quality Loss vs. % of False Decisions

- Chart 1: % of False Decisions vs. Application Quality Loss for Table-based methods.
- Chart 2: % of False Decisions vs. Application Quality Loss for Neural methods.
Binomial Proportion Confidence Interval

\((n_{\text{trials}}, n_{\text{success}}) \Rightarrow r < \text{SuccessRate} \) with a confidence level
Evaluation of MITHRA

Comparing geometric mean - speedup, energy reduction and invocation rate across varied set of benchmarks.
False Positive and False Negatives

Graph 1: False Positive and False Negative comparisons using Table-based approach.

Graph 2: False Positive and False Negative comparisons using Neural approach.
Multi-Input Hashing and Input Signature Generation
Multiple Tables for Improved Performance

A small ensemble of table-based Classifiers achieve better accuracy and performance
Neural Network Based Mechanism
Compiler Support for MITHRA

• Programmer provides:
  – Application specific quality requirement
  – Quality Metric
  – Set of representative application inputs

• Algorithm maps final output requirement to a threshold \((\text{th})\) on accelerator error.

Accelerator error < \(\text{th}\) : learn to invoke the accelerator
Accelerator error > \(\text{th}\) : learn to fall back to the original function
Quality vs Benefits Tradeoffs

Approximation vs Benefits vs Output Quality

- Quality
- Approximation
- Benefits