Exploiting Quality-Efficiency Tradeoffs with Arbitrary Quantization

Special Session - CODES+ISSS

Thierry Moreau, Felipe Augusto, Patrick Howe
Armin Alaghi, Luis Ceze
Approximate computing: eliminate inefficiencies in systems by producing just-the-right quality
Quantization: going back to basics

noisy, real world sensory input → processing → aggregate analytics, consumed by human etc.

SRAM

ALU

SRAM

ALU
This Talk: A “Limit Study” on Precision Scaling

Assumption: hardware that can dynamically and arbitrarily scale its precision

SW Scope: compute heavy, regular applications

HW Scope: hardware accelerators
Talk Overview

1. How much precision is needed at different stages of a program?

2. How much energy can be saved (upper bound)?

3. How does this inform approximate computing research?
Talk Overview

1. How much precision is needed at different stages of a program?

   QAPPA - Precision Autotuner

2. How much energy can be saved?

3. How does this inform approximate computing research?
QAPPA: Quality Autotuner for Precision-Programmable Accelerators

Goal: Minimize instruction-level precision requirements given a quality target

Built on top of ACCEPT, the approximate C/C++ compiler
http://accept.rocks
QAPPA Autotuner Overview

Default (no savings)

instruction 0
instruction 1
instruction 2
...
instruction n-1
instruction n

savings

bad → OK

application quality
QAPPA Autotuner Overview

Optimized: extraneous precision is shaved off

instruction 0
instruction 1
instruction 2
...
instruction n-1
instruction n

savings

application quality
QAPPA 5-Step Description

Annotated Program

Program Inputs & Quality Metrics

Output Configuration

Quality Autotuner

Accept error injection & instrumentation

Approximate Binary

Execution & Quality Assessment

acceptable static analysis

ILPC*

* Instruction-level Precision Configuration
1. Program Annotation

void conv2d (APPROX pix *in, APPROX pix *out, APPROX flt *filter) {
    for (row) {
        for (col) {
            APPROX flt sum = 0
            int dstPos = ...
            for (row_offset) {
                for (col_offset) {
                    int srcPos = ...
                    int fltPos = ...
                    sum += in[srcPos] * filter[fltPos]
                }
            }
            out[dstPos] = sum / normFactor
        }
    }
}
2. Static Analysis

void conv2d (APPROX pix *in, APPROX pix *out, APPROX flt *filter)
{
    for (row) {
        for (col) {
            APPROX flt sum = 0
            int dstPos = ...
            for (row_offset) {
                for (col_offset) {
                    int srcPos = ...
                    int fltPos = ...
                    sum += in[srcPos] * filter[fltPos]
                }
            }
            out[dstPos] = sum / normFactor
        }
    }
}

ACCEPT identifies safe-to-approximate instructions from data annotations using flow analysis

Instruction-Level Precision Configuration (ILPC)

conv2d:13:7:load:Int32
conv2d:13:10:load:Float
conv2d:13:11:fmul:Float
conv2d:13:12:fadd:Float
conv2d:15:1:fdiv:Float
conv2d:15:7:store:Int32
3. Error Injection

Each instruction in the ILCP acts as a quality knob that the autotuner can use to maximize bit-savings.
4. Quality Assessment

The programmer provides a quality assessment script to evaluate quality on the program output.
5. Autotuning Algorithm

**Greedy iterative algorithm [•]:** reduces precision requirement of the instruction that impacts quality the least

```
config k:
  error = 0.10%

config [k+1, i-1]:
  error = 5.91%

config [k+1, i]:
  error = 0.30%

config [k+1, i+1]:
  error = 0.12%

config [k+2, i-1]:
  error = 5.91%

config [k+2, i]:
  error = 0.33%

config [k+2, i+1]:
  error = 1.6%
```

Finds solution in $O(m^2n)$ worst case where $m$ is the number of static safe-to-approximate instructions and $n$ are the levels of precision for all instructions

[•] Precimonious, Rubio-Gonzalez et al., SC’13
5. Autotuning Algorithm

The autotuner greedily maximizes bit-savings as the quality target is lowered.
## PERFECT Application Study

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Kernels</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERFECT Application 1</td>
<td>Discrete Wavelet Transform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2D Convolution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Histogram Equalization</td>
<td></td>
</tr>
<tr>
<td>Space Time Adaptive Processing</td>
<td>Outer Product</td>
<td></td>
</tr>
<tr>
<td></td>
<td>System Solve</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inner Product</td>
<td></td>
</tr>
<tr>
<td>Synthetic Aperture Radar</td>
<td>Interpolation 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interpolation 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Back Projection</td>
<td></td>
</tr>
<tr>
<td>Wide Area Motion Imaging</td>
<td>Debayer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image Registration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Change Detection</td>
<td></td>
</tr>
<tr>
<td>Required Kernels</td>
<td>FFT 1D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FFT 2D</td>
<td></td>
</tr>
</tbody>
</table>

**Signal to Noise Ratio (SNR)**

- [120dB to 10dB]
- (0.0001% to 31.6% MSE)
Opportunity of Approximations

QAPPA Analyzes PERFECT Dynamic Instruction Mix

- **load/store**: 27%
- **int arith**: 25%
- **fp arith**: 31%
- **control**: 11%
- **math**: 1%

Safe to approximate

Precise
Average Precision Reduction Achieved Across PERFECT Kernels

Dynamic precision reduction on safe-to-approximate instructions

Target Application SNR (dB)

Approximate | High Quality

More savings
Average Precision Reduction Achieved Across PERFECT Kernels

Dynamic precision reduction on safe-to-approximate instructions

PERFECT Manual
0.001% MSE
Average Precision Reduction Achieved Across PERFECT Kernels

Approximate Computing 10% MSE
Talk Overview

1. How much precision is needed at different stages of a program?

   QAPPA - Precision Autotuner

2. How much energy can be saved (upper bound)?

   Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?
Translating Precision Reduction into Energy Savings (Compute)

Baseline ALU

No savings
Translating Precision Reduction into Energy Savings (Compute)

Baseline ALU

Value Truncation

No savings

Less Power

QUORA [MICRO‘13]
Translating Precision Reduction into Energy Savings (Compute)

Baseline ALU

Value Truncation

Bit-Sliced

QUORA [MICRO'13]

Stripes [MICRO'16]

No savings

Less Power

Higher Throughput
Case Study: Precision Scaled Adder

**Goal:** Design an precision scalable adder that can elegantly trade lower precision for energy savings

**Exploration:** Combine value truncation and bit slicing techniques, and vary the slice width in increments of powers of 2

**Methodology:** Post-place-and-route prime-time power analysis on 65nm TSMC library
Precision Scaled Adder Study

Input Bit-Width

Energy Cost (pJ)

- technique 1: value truncation

offset due to static power
Precision Scaled Adder Study

Input Bit-Width

Energy Cost (pJ)

technique 2: bit slicing
Case Study: Precision-Scaled Adder

we look at different slice widths in powers of 2 increments

a 2-bit slice seems to be the energy-optimal design point
PERFECT Study: Compute Energy Savings

Average Compute Energy Savings vs. Application SNR

Energy Savings (x) - Higher is Better

Application SNR (dB) - Higher is Better

PERFECT Study: Compute Energy Savings

PERFECT Study: Compute Energy Savings

PERFECT Study: Compute Energy Savings

PERFECT Study: Compute Energy Savings
PERFECT Study: Compute Energy Savings

Average Compute Energy Savings vs. Application SNR

At 40dB a 16b sliced ALU can achieve 4.8 energy reduction!
PERFECT Study: Compute Energy Savings

Average Compute Energy Savings vs. Application SNR

At 20dB the optimal design point shifts to 8-bit slice width.
Talk Overview

1. How much precision is needed at different stages of a program?

   QAPPA - Precision Autotuner

2. How much energy can be saved (upper bound)?

   Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?

   Comparative Study of Approximation Techniques
Comparative Study

Many papers on approximate computing state:
“Our technique provided $n$ times speedup at $x\%$ error”

**Problem**: This give us a data point but doesn’t quite say much about the merits of the technique at trading accuracy for efficiency

**Solution**: Use QAPPA to produce quick comparison results to assess effectiveness of technique
Methodology (1/2): Spice simulation of ALU/FPU design under different voltage overscaling factors.
Comparative Study - Voltage Overscaling

**Methodology (2/2):** Then we feed the error model into QAPPA’s error injection framework to assess application error.

**Results:** Precision scaling always produces better quality/efficiency.
Future Directions in Architecture/CAD

Precision Scaling Architectures: Need to see more precision-scaled accelerators for more applications of the likes of Quora[MICRO’13], Stripes[MICRO’16]

CAD tools with Quality Awareness: Need to see more tools that can leverage quantization, especially in the FPGA community, of the likes of AHLS[DATE’17]
Conclusion

1. How much precision is needed at different stages of a program?

   QAPPA - Precision Autotuner

2. How much energy can be saved (upper bound)?

   Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?

   Comparative Study of Approximation Techniques
Exploiting Quality-Efficiency Tradeoffs with Arbitrary Quantization

*Special Session - CODES+ISSS*

**Thierry Moreau**, Felipe Augusto, Patrick Howe
Armin Alaghi, Luis Ceze

Thank you!