Special Session Paper: Exploiting Quality-Energy Tradeoffs with Arbitrary Quantization

Thierry Moreau
University of Washington

Felipe Augusto
University of Campinas

Patrick Howe
University of Washington

Armin Alaghi
University of Washington

Luis Ceze
University of Washington

ABSTRACT

Approximate computing aims to expose and exploit quality vs. efficiency tradeoffs to enable ever-more demanding applications on energy-constrained devices such as smartphones, or IoT devices. This paper makes the case for arbitrary quantization as a compelling approximation technique that exposes quality vs. energy tradeoffs and provides practical error guarantees.

We present QAPPA (Quality Autotuner for Precision Programmable Accelerators), an autotuning framework for C/C++ programs that automatically minimizes the precision of each arithmetic and memory operation to meet user defined application level quality guarantees. QAPPA integrates energy models of precision scaling mechanisms to produce bandwidth and energy savings estimates for precision scalable accelerator designs. We show that QAPPA can exploit precision scaling mechanisms to meet arbitrary user-provided quality targets on the PERFECT benchmark suite to achieve significant energy savings and memory bandwidth reduction.

ACM Reference format:
Thierry Moreau, Felipe Augusto, Patrick Howe, Armin Alaghi, and Luis Ceze. 2017. Special Session Paper: Exploiting Quality-Energy Tradeoffs with Arbitrary Quantization. In *Proceedings of CODES/ISSS '17 Companion, Seoul, Republic of Korea, October 15–20, 2017,* 2 pages.
https://doi.org/10.1145/3125502.3125544

1 INTRODUCTION

Navigating quality-energy tradeoffs is fundamental to digital systems design, and often starts with data representation, i.e. how to map a set of real valued data to a finite digital representation. This process is called *quantization* and can elegantly trade quality for energy efficiency by tweaking the number of bits needed to represent data. This paper argues towards adopting precision scaling as a general approximation technique for its effectiveness in delivering smooth quality-energy tradeoffs, and practical error guarantees. We assume hardware accelerator architectures that can dynamically scale the precision of its arithmetic and memory operation [3, 7].

CODES/ISSS '17 Companion, October 15–20, 2017, Seoul, Republic of Korea
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5185-0/17/10…$15.00
https://doi.org/10.1145/3125502.3125544

2 QAPPA: A QUALITY AUTOTUNER

QAPPA, shown in Figure 1, is a quality autotuning framework built using ACCEPT [6], the LLVM-based approximate compiler for C and C++ programs. In a nutshell, QAPPA takes an annotated C/C++ program and user-specified, high-level quality requirements and greedily derives quantization settings for each program instruction.

Autotuner. QAPPA attempts to maximize *bit savings* while keeping application accuracy within user-specified margins. We define bit savings as a hardware-agnostic metric that quantifies how much total precision can be trimmed-off in a program over its execution:

$$BitSavings = \sum_{i=1}^{N} \frac{(r_i - q_i)}{r_i} \times \frac{e_i}{\sum_{j=1}^{N} e_j}$$

where r_i and q_i denote the precision in bits of the reference, and quantized instruction i, e_i denotes the number of times instruction i executes, and N denotes the total instructions that can be quantized in the target program. QAPPA uses a greedy search algorithm similar to the one used in [5] in order to maximize bit savings while satisfying user-specified quality requirements. The greedy trial-and-error search relies on program instrumentation to replace expensive arithmetic and memory operations with cheaper operations.

Guarantees. We implement QAPPA to provide two types of quality guarantees: *empirical* and *statistical*. Empirical guarantees provide guarantees as good as the dataset provided by the user: this approach puts pressure on the programmer to provide satisfactory input coverage. Statistical guarantees on the other hand present a robust alternative since they account for input samples that are not covered by the user-provided input-set. Statistical error guarantees are expressed in the form of a Clopper-Pearson [1] confidence interval, requiring the user to model an input distribution.

3 APPLICATION STUDY

We run QAPPA on the PERFECT [2] benchmark suite kernels to quantify approximation opportunities on compute intensive workloads. The PERFECT benchmark suite is composed of kernels covering DSP, radar, vision and machine learning applications domain.

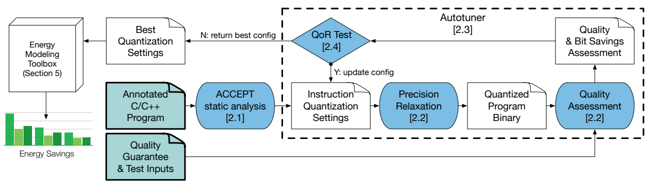
Autotuner
[2.3]
Energy Modeling Toolbox (Section 5)
Best Quantization Settings
N: return best config
QoR Test [2.4]
Quality & Bit Savings Assessment
Y: update config
Annotated C/C++ Program
ACCEPT static analysis [2.1]
Instruction Quantization Settings
Precision Relaxation [2.2]
Quantized Program Binary
Quality Assessment [2.2]
Quality Guarantee & Test Inputs
Energy Savings

Figure 1: QAPPA Autotuner System Architecture.

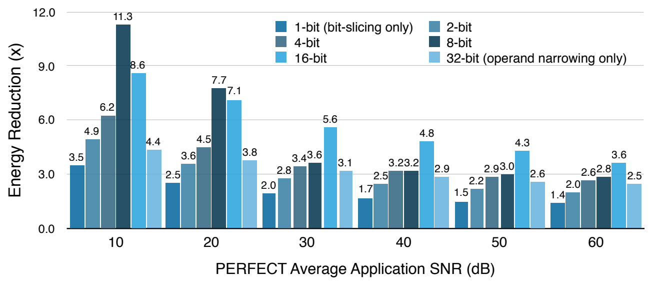

Figure 2: Arithmetic energy reduction on the PERFECT benchmark at different bit slicing granularities and at different SNR targets (higher is better).

We follow the PERFECT manual guidelines for quality assessment and use the Signal-to-Noise Ratio (SNR) quality metric across all benchmarks to measure quality degradation.

Approximation Opportunities. QAPPA relies on ACCEPT to identify which instructions in a program can be quantized based on user annotations: on average, 64% of all dynamic instructions can be approximated. Those instructions are for the most part composed of *expensive* operations such as floating-point arithmetic, memory loads and stores. The bulk of the instructions that cannot be approximated are composed of control instructions and integer arithmetic used for address computation, neither of which can be approximated without compromising the safety of the program.

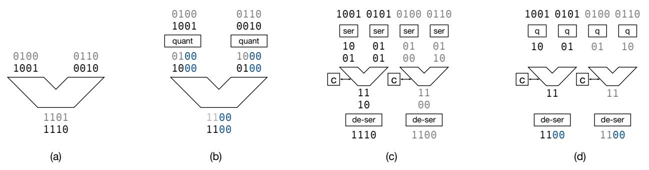

Figure 3: Precision scaling mechanisms overview. (a) Default wide addition on wide adder. (b) Narrow addition on wide adder. (c) Wide addition on narrow adder (d) Narrow addition on narrow adder (combination of (b) and (c)).

Hardware Mechanisms. We survey two hardware *dynamic precision scaling* mechanisms shown in Figure 3 and discuss the savings in arithmetic energy and memory bandwidth that these mechanisms achieve on accelerator designs executing the PERFECT kernels. The first technique, operand narrowing (3.b), aims to minimize power by reducing transistor switching [7] on wide compute units. The second technique, bit slicing (3.c) (or operator narrowing), utilizes narrow compute unit in parallel to time-multiplex the computation of wider operations, effectively scaling throughput with precision on data-parallel workloads [3]. We synthesize adder and multiplier designs of varying widths using the Synopsys Design Compiler with the TSMC-65nm library and collect power data with PrimeTime PX on post place-and-route designs to build an energy modeling library which we incorporate into QAPPA.

Evaluation. We run QAPPA on the PERFECT benchmark suite to guide our choice of an energy-optimal precision scaling mechanism at different quality targets ranging from 60dB down to 10dB. Figure 2 shows energy savings across all PERFECT benchmarks over a standard arithmetic unit executing 32 bit arithmetic operations.

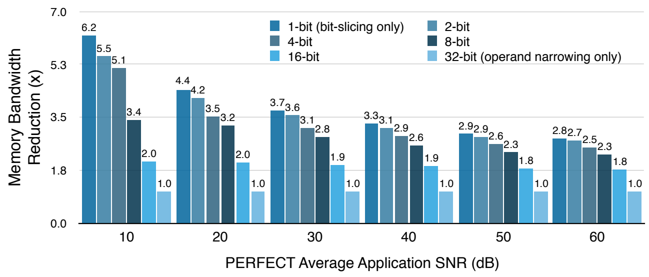

Figure 4: Ideal bandwidth reduction on PERFECT benchmark suite at different data packing granularities and at different SNR targets (higher is better).

Combining bit slicing and operand narrowing can greatly maximize energy efficiency: a slice width of 16 bits yields optimal energy reductions by 3.6× and 4.8× at 40dB and 60dB over the baseline arithmetic unit. Additionally, applying aggressive quantization to data can significantly minimize memory bandwidth. Figure 4 shows data movement reduction for a fixed-function accelerator. We vary the data packing granularity from 1 to 32 bits and derive the resulting memory bandwidth reduction. A data packing granularity of 1 bit can achieve 4.4×, 3.3×, and 2.8× average memory bandwidth reduction on the PERFECT kernels at 20dB, 40dB and 60dB.

Discussion. Data packing at fine granularities can increase both software and hardware overheads for packing and unpacking. A hardware designer might therefore want to align the data packing granularity with the bit slicing width of the precision scalable compute units to minimize control overheads. The optimal data granularity will differ based on the target system energy breakdown between memory, computation, and control.

4 CONCLUSION.

We present QAPPA, a framework that fine-tunes quantization requirements of C/C++ programs, while meeting user defined, application level quality guarantees. QAPPA incorporates hardware models of precision scaling mechanisms to inform the user of potential energy savings and memory bandwidth reduction. The insights of this work are discussed in more detail in the following technical report [4], which also compares the technique of precision scaling with other mainstream approximation techniques.

REFERENCES

[1] E. S. Pearson C. J. Clopper. 1934. The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika* 26, 4 (1934), 404–413.
[2] Kevin Barker et al. 2013. *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*. PNNL and Georgia Tech.
[3] P. Judd, J. Albericio, and A. Moshovos. 2016. Stripes: Bit-Serial Deep Neural Network Computing. *IEEE Computer Architecture Letters* (2016).
[4] T. Moreau, F. Augusto, P. Howe, Alaghi A., and Ceze L. 2017. *QAPPA: A Framework for Navigating Quality-Energy Tradeoffs with Arbitrary Quantization*. Technical Report UW-CSE-17-03-02. U. Washington.
[5] C. Rubio-Gonzalez, C. Nguyen, H. D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey, C. Iancu, and D. Hough. 2013. Precimonious: Tuning Assistant for Floating-Point Precision. In *Int. Conf. High Performance Computing, Networking, Storage and Analysis*.
[6] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze, and M. Oskin. 2015. *ACCEPT: A Programmer-Guided Compiler Framework for Practical Approximate Computing*. Technical Report UW-CSE-15-01-01. U. Washington.
[7] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. 2013. Quality Programmable Vector Processors for Approximate Computing. In *IEEE/ACM Int. Symp. Microarchitecture (MICRO)*.