Work in progress: analysis and evaluation of the impact of the code approximation for IoT applications

David Medeiros Cruz[1] and Tiago Almeida[1,2]

[1] *Universidade Federal do Tocantins (UFT), Computer Science Department Palmas/TO, Brasil*
[2] *University of Campinas (UNICAMP), Institute of Computing, Campinas/SP, Brazil*

Reception date of the manuscript: 25/07/2023
Acceptance date of the manuscript: 04/09/2023
Publication date: 16/10/2023

Abstract— Due to the need to improve resource management for computer systems in many levels and applications (mainly for embedded systems and energy consumption), how can we enhance the energy efficiency of computational methods? One approach is through approximate computing, which intentionally introduces controlled errors to save resources such as energy, area, or time. This research aims to empirically measure the impact of introducing approximations in an embedded system by conducting a controlled experiment. To focus on evaluating the impact of the approximations themselves rather than the best methods of implementing them, the approximations will be manually incorporated into the code. The benchmark chosen for evaluation is MiBench due to its widespread usage. All the codes can be recompiled to run on the MIPS architecture of the NodeMCU-ESP8266. A second NodeMCU-ESP8266 will be utilized, connected in series to measure the actual power consumption of the first board. The analysis of results will involve hypothesis tests, where the experiment hypotheses will be statistically evaluated at a specific significance level. By directly comparing variations and experiment data, the proposal's validity will be effectively demonstrated. Since this paper is a work in progress, we will explain the experiment planned to be run.

Keywords—Approximate Computing, Energy Efficiency, Internet of Things, Software Approximations.

I. Introduction

Computing has changed over the last decades. The design of computing systems has shifted from single-core systems to multi-core and heterogeneous systems. The main reason for this transformation is the physical limitation of miniaturization of transistors to improve the performance of computational systems [1].

Another factor to consider in the current context is the decentralization of the processing. Where exactly is the application executed? In many cases, applications are exclusively run on a cloud system, while in others, embedded systems are used, or a combination of both solutions.

If we also consider the need for better resource management, each scenario will have different energy demands, making it more complex to find suitable designs for each case.

How could we improve the energy efficiency of computational systems? One paradigm that aims to address this question is approximate computing [2, 3]. In applications where there is a sensory limitation, meaning that humans would not be able to perceive errors in computation, where an exact output does not exist, such as in machine learning, or where computation is based on probabilities, there is resilience to errors. These applications can exhibit outputs with a certain threshold of error, and there would be no issue. In such cases, approximate computing can introduce controlled errors to save physical resources of the computational system, such as energy, area, or time.

The scenario with good error resilience and significant energy and performance constraints is the embedded system scenario (or, depending on the context, the Internet of Things). There is a range of research addressing the use of approximations for high-performance systems, with only a few studies focusing on embedded systems. In this regard,

> *What is the impact on energy consumption of a set of approximation techniques in software on a controlled system?*

Based on this research question, the remaining part of the research project will be based on, experimented with, and discussed in the following sections.

II. Problem definition

To illustrate the problem, consider a simple matrix multiplication application. Considering i) square matrices with sizes of 1000, 2000, 3000, 4000, 5000, and 7000; ii)

Contact data: David Medeiros Cruz, david.medeiros@uft.edu.br

randomly generated values from a uniform distribution; iii) an Intel Core i5-9300H 2.40GHz CPU (Kaby Lake processor with 8 cores in one encapsulation) and; iv) ten repetitions for each input size. Thus, Fig. 1 shows how energy consumption scales in this scenario. Measurements were taken using the `perf` tool available for the Linux operating system and using embedded power counters on Intel processors.

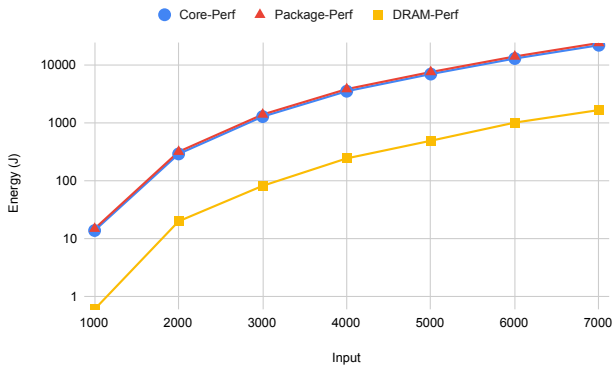

Fig. 1: Variation of energy consumption according to input size for a matrix multiplication application.

In Fig. 1, energy consumption is measured in joules, which represents $J = \frac{W}{s}$, where W is power measured in watts and s is time measured in seconds. "Core" represents the energy consumption measured in the processing core, "Package" represents the energy consumption measured across the entire chip encapsulation, and "DRAM" represents the energy consumption measured by the RAM. The energy consumption of "Core" and "Package" is quite similar since it is a serial application, meaning the processing is performed by a single core.

Fig. 2 presents the breakdown of the processing for each input size. It shows how much each component, "Core," "Package," and "DRAM," contributes to the total application energy consumption. For all input scenarios, the behavior is the same: "Core" and "Package" account for the majority of energy consumption, as this is a CPU-bound application, meaning it is more dependent on processing rather than data traffic with memory.

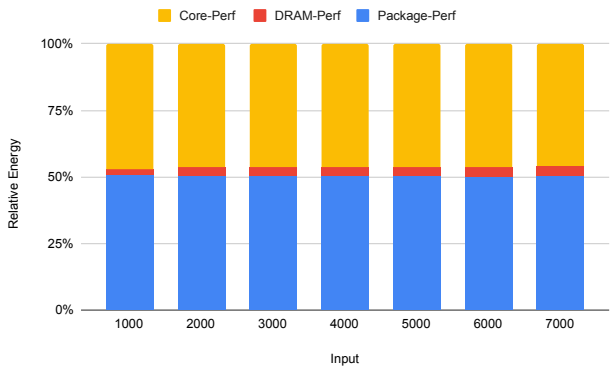

Fig. 2: Breakdown by each part and its respective energy consumption according to the input size for a matrix multiplication application.

Therefore, the objective of this study is to conduct a controlled experiment to empirically measure the impact of introducing approximations in an embedded system. In what context will these approximations be inserted? Although the focus is on software approximation, there are different insertion contexts, such as compile-time or real-time for adjustment of approximations. However, the objective of this study is to evaluate the impact of the approximations themselves and not the best ways to insert them, so the approximations will be manually inserted into the code.

In line with this, the experiment will be conducted on a NodeMCU-ESP8266 microcontroller. This is a widely used microcontroller model with relatively low cost and is also used for educational purposes.

The dependent variables in this experiment, that is, the variables that are measured to assess the cause-and-effect relationship, are power (measured in watts), energy (measured in joules), time (measured in seconds), and the application error (all error metrics used are described in Table 1 [4]).

TABLE 1: DESCRIPTION OF THE ERROR METRICS USED IN THIS PROJECT.

Metrics	Description	Equation	
MAE*	Mean Absolute Error	$\frac{1}{n}\sum_{i=0}^{n}\lvert O_i^{ac} - O_i^{ax}\rvert$	(1)
WCE*	Worst-Case Error	$\max \forall i \lvert O_i^{ac} - O_i^{ax}\rvert$	(2)
MRED*	Mean Relative Error Distance	$\frac{1}{n}\sum_{i=0}^{n}\frac{\lvert O_i^{ac} - O_i^{ax}\rvert}{O_i^{ac}}$	(3)
MSE*	Mean Squared Error	$\frac{1}{n}\sum_{i=0}^{n}(O_i^{ac} - O_i^{ax})^2$	(4)

* O_i^{ac} is a i-th precise output, and O_i^{ax} é a i-th approximate output.

Finally, the following research hypothesis is considered, based on the research question stated in the previous section [5]:

> *Null hypothesis (H_0): There is no difference in energy consumption and performance in an embedded application when software approximations are introduced.*
>
> *Alternative hypothesis (H_a): There is a difference in energy consumption and performance in an embedded application when software approximations are introduced for at least one pair i and j, where $i \neq j$, and i represents the precise application, and j represents the same application with an approximation from a set of approximations.*

In this experiment, the objective is to empirically evaluate the stated hypothesis, and there is no need to perform a statistical hypothesis testing method, as it is beyond the scope of this project to analyze the significance of any existing differences. The presentation of line, bar, and scatter plots is sufficient to interpret the results.

ISSN: 2675-3588

a. Experimental setup

Regarding the types of software approximation, the following types will be used [6, 7]: i) Code Perforation: This involves identifying parts of the code that are resilient to errors and can be "skipped" during execution, such as functions and code segments. ii) Loop Perforation: This involves identifying loops that are also resilient to errors and partially executing these loops. iii) Bitwidth transformation: This involves identifying variables in the code that can be transformed into representations with fewer bits, or transforming floating-point variables into integers, or integers into unique characters.

In terms of the benchmark to be evaluated, the MiBench [8] will be considered due to its widespread use. This benchmark consists of applications divided into six categories: Industrial applications; Consumer applications; Office applications; Network applications; Security applications; Telecommunication applications.

Although MiBench was designed to evaluate the ARM architecture, all codes can be recompiled to run on the MIPS architecture of the NodeMCU-ESP8266.

Another important point is how the measurement data will be collected. In this regard, a second NodeMCU-ESP8266 will be used, connected in series to measure the real power consumption of the first board. Since the NodeMCU-ESP8266 system already has an embedded WiFi antenna, it is a viable platform for collecting data and sending it to a traditional computer with measurement information. The first step of this research project will be the construction and testing of the energy consumption measurement platform.

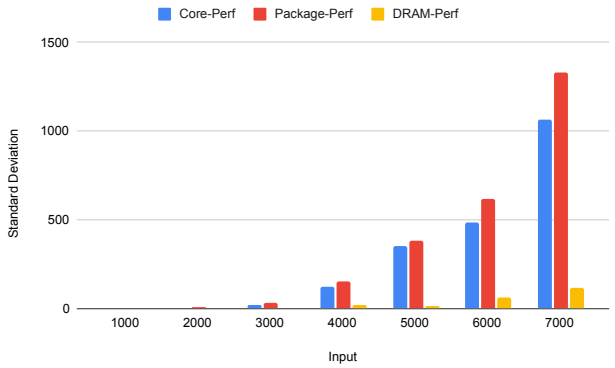

Fig. 3: Standard deviation of energy consumption according to input size for a matrix multiplication application over ten replicates of the experiment.

Returning to the motivational example of the matrix multiplication application explained earlier, Fig. 3 demonstrates the standard deviation for ten different repetitions. As the size of the input increases, the standard deviation also increases, growing almost exponentially. This highlights the need for an external system, such as the proposed one, to measure consumption and obtain more reliable results.

Fig. 4 shows the general idea of this project. An original code in C/C++, for a matrix multiplication, gives a design with no error and the maximum energy consumption, represented in the Pareto front plot. Each gray dot represents a possible approximation inserted to save energy. When applying a loop perforation by a specific `rate`, the run flow will skip some iteration to save energy, introducing acceptable error in the application and also improving the efficiency, represented by the green dot in Fig. 4. Since other NodeMCU-ESP8266 will be used, for real energy consumption, we can publish on web the measurements.

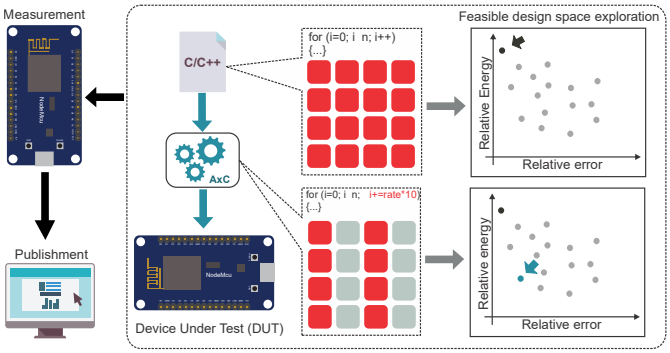

Fig. 4: General diagram and data flow of the whole project exemplifying the usage of loop perforation to improve energy efficiency.

III. Expected results and final considerations

After collecting experimental data, descriptive statistics can be used to describe and graphically present interesting aspects of the dataset. These aspects include measures that indicate, for example, where the data are placed on a scale and how concentrated or dispersed the dataset is [5]. In this sense, by the next months, we expect to have a measurement system and analyze the impact of approximations in an embedded system. As far as we know, many researchers have explored energy-efficient with approximate computing in IoT nodes based on FPGAs and ARM architecture, and our proposal has not been explored yet.

References

[1] M. Alioto, V. De, and A. Marongiu, "Energy-quality scalable integrated circuits and systems: Continuing energy scaling in the twilight of moore's law," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, pp. 653–678, 2018.

[2] N. E. Jerger and J. S. Miguel, "Approximate computing," *IEEE Micro*, vol. 38, pp. 8–10, 2018.

[3] A. Aponte-Moreno, A. Moncada, F. Restrepo-Calle, and C. Pedraza, "A review of approximate computing techniques towards fault mitigation in hw/sw systems," in *2018 IEEE 19th Latin-American Test Symposium, LATS 2018*, vol. 2018-Janua, 2018, pp. 1–6.

[4] J. Castro-Godínez, M. Shafique, and J. Henkel, "Ecax: Balancing error correction costs in approximate accelerators," *ACM Transactions on Embedded Computing Systems*, vol. 18, 10 2019.

[5] P. R. C. Wohlin, M. H. Ohlsson, M. C. Bjorn Regnell ost, and A. Wesslén, *Experimentation in Software Engineering*. Elsevier, 2013, vol. 1.

[6] L. M. Miranda, "Llvm-act: uma ferramenta baseada em profiling para seleção de técnica de computação aproximada," Master's thesis, Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, RN, 2022.

[7] L. O. P. dos Reis, "Targeting broad software approximations with the accept framework: Ampliando aproximações em software com o framework accept," Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, Campinas, SP, 2021.

[8] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded

benchmark suite," in *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop*, ser. WWC '01. USA: IEEE Computer Society, 2001, p. 3–14.

 ISSN: 2675-3588