Towards Designing and Implementing Approximate Accelerators

Jorge Castro-Godínez∗†, 1, Muhammad Shafique‡, Jörg Henkel∗

∗ Kalrsruhe Institute of Technology, 76131 Karlsruhe, Germany
† Instituto Tecnológico de Costa Rica, 30101 Cartago, Costa Rica
‡ Vienna University of Technology, 1040 Vienna, Austria

ABSTRACT
The design and implementation of approximate accelerators impose several challenges, particularly when using approximate arithmetic circuits. During the design, error estimation is required to assess the accuracy of different approximate versions and to help in the selection of approximate components. On the other hand, the implementation of approximate accelerators requires to produce results with errors within a given threshold, and for this an error correction mechanism, in many cases, is required. In this work, we present: a) a compiler-driven methodology for error estimation of approximate designs and, b) an approach to reduce the required error correction performed at software when using approximate accelerators, in order to improve performance gains.

KEYWORDS: Approximate computing; error modeling; error correction; design automation

1 Introduction
In the last decade, the need for energy-efficient computing has motivated the coming forth of Approximate Computing (AxC), a novel design paradigm relevant to applications with inherent resilience to errors, such as image processing, multimedia, database processing, and machine learning [SHR+16, CHSH20]. For such applications, AxC proposes to reduce the accuracy of the results (computational quality) produced to lower the required resources (computational effort), for instance, delay, power, and energy.

One way to harness the AxC paradigm is through approximate accelerators. The idea behind an accelerator is to offload a highly-frequent and compute-intensive section of an application to a dedicated hardware design, while the rest of the application is executed by a host processor. From the AxC perspective, approximate accelerators exploit error resilience as frequently-executed but error-tolerant sections of a program are performed by dedicated approximate hardware designs.

1E-mail: jorge.castro-godinez@kit.edu, jocastro@itcr.ac.cr
Designing and implementing approximate accelerators with approximate arithmetic units, however, impose several challenges. In both cases, an accuracy threshold is given to guide the design and execution of approximate accelerators. In the design phase, error estimation mechanisms are required to assess the accuracy of different approximate versions and to help in the selection of approximate components that would meet the error constraint. Once an approximate accelerator is integrated to a processor, runtime error correction mechanisms are required to provide statistical guarantees that the overall accuracy is met, which normally implies the re-computation at software level of those accelerator invocations generating errors beyond the defined threshold.

2 Error Estimation

In the process of designing approximate accelerators, evaluation of the accuracy is required to determine if a given threshold, for a specific error metric, is satisfied. However, when designing approximate accelerators using approximate components such as approximate adders and multipliers, their individual error characteristics can be used to estimate the accuracy of the design in which they are used. In [CGES*18], we present compiler-driven methodology for estimating the error propagation to the outputs of an approximate accelerator. To enable this, we define a set of models to estimate the propagation of error distributions represented as a probability distribution. This models are based on a set of heuristics to model error propagation for individual calculations, for which the individual error characterization of approximate components [HCSH20], particularly approximate adders and multipliers, is required.

For the error estimation, we consider a software model of the accelerator design. This code can be annotated with custom pragma directives to indicate which accurate operations are replaced by approximate ones. Then, a modified version of Clang is to handle these annotations and to append metadata to the intermediate representation (IR) of the code. From this IR, a data-flow graph representation of the accelerator is built and it is statically analyzed using the proposed error propagation models to obtain an accuracy estimation of the approximate accelerator.

3 Error Correction

In the literature, runtime mechanisms to monitor and control the accuracy of the results produced when using approximate accelerators have been proposed. The common idea consists in using light-weight pre-trained error predictors to estimate if the invocation of an approximate accelerator would produce an error beyond a given threshold for a particular input data set. If the predictor forecasts an unacceptable error, a recovery signal indicates the host processor to recover the last invocation by re-computing it accurately per software.

To mitigate the required re-computations, we present a selective error correction approach to balance the cost of performing correction in [CGSH19]. Our methodology explore low-overhead error correction in approximate accelerators by selectively correcting most significant errors, in terms of their magnitude, without losing the gains of approximations. We particularly consider the case of approximate accelerators designed with approximate functional units such as approximate adders. To identify and evaluate the correction scenar-
ios, we use our error estimation approach [CGES+18] to evaluate a software model of the approximate accelerator design.

Our novel methodology significantly reduces the required exact re-computations on the host processor, as it can be seen in Figure [1] for an approximate Gaussian filter. As it can be noticed, this occurs as the error threshold is reduced. This reduction on the re-computations causes an increment in performance, as up to 20% speedup improvement on top of using error predictors proposed by the related work is achieved, despite the delay reduction experienced by the accelerator due to the selective error correction introduced in the design.

References


