UNIFY: Compilation and Hardware Adaptation for Unified Computing

I. Introduction

The need for a continued regarding compute performance has driven the rise of heterogeneous architectures and non-conventional computing paradigms. To support such Domain or Application Specific Architectures, which may rely on mechanisms like data streaming and re-configurable technologies, new programming and compilation flows are required. This may take the form of dedicated Domain-Specific Languages (DSLs), designed for architecture specific features of hardware. However, this specialization and unconventional mechanisms adopted in DSAs often lead to a rather limited (or non-existent) support for general programming languages, such as C/C++. Consequently, while some architectural enhancements may be introduced, supported by DSLs and modified compilers, the over-arching issue of performance of general-purpose processors (GPPs) is not directly addressed.

Instead, simultaneously addressing the adoption of new compiler technologies and new architecture innovations has been recognized as the holistic approach to avoid this foreseeable fragmentation between specialized and general-purpose computing. The UNIFY project aims to develop a compilation flow where general purpose programming languages can target non-conventional processors, augmented with accelerated computing capabilities, and other heterogeneous architectures like Coarse Grained Reconfigurable Arrays.

II. UNIFY Project

The main objective of UNIFY is to extend compilation tools and flows to abstract the application structural information from the characteristics of the underlying hardware, realizing a integrated compilation flow from general purpose language to different hardware back-ends, as shown in Figure 1.

A new Structural Representation Language (SRL) specification shall be proposed to allow the automatic detection and representation of deterministic and non-deterministic computing structures and memory access schemes. The compilation flow will infer SRL constructs from the compiler's internal representation language, and then translate them back either to general-purpose machine-code, accelerator code, or for mapping to DSL constructs (targeting DSAs).

The approach will exploit the information gathered during compilation to support the definition and adaptation of the underlying re-configurable processing architectures supported by data streaming schemes, thus promoting runtime domain-specific specialization, resource management, and performance and energy consumption balancing.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT), under project 2022.06780.PTDC (DOI:10.54499/2022.06780.PTDC).

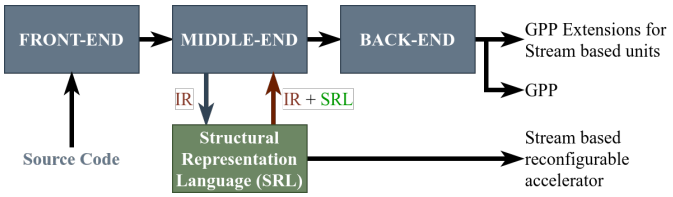

Fig. 1. UNIFY Compilation Flow

A. Research Plan and Methodology

We propose the combination of several new and unconventional computing paradigms and techniques, applied both at the compilation and hardware levels, in a single unified flow. Although achieving a true unification is challenging and possibly still several years away, this project aims at taking significant steps in that direction. To achieve this, the project proposes the following objectives:

- Develop a new compiler internal Structural Representation Language (SRL) to represent high-level computing structures, describe memory access patterns, expose spatial and temporal computation characteristics.
- Develop compiler extensions to infer the SRL specification and translate it to target-specific code.
- Investigate new stream-based reconfigurable architectures and execution models which can be targeted via the SRL and compilation chain.
- Deploy a fully functional and open-source computing ecosystem, with an unprecedented integration between re-configurable hardware and innovative compiler extensions, providing performance beyond current day GPPs.

B. On-Going Work

The project has developed the envisoned formats for the SRL at several levels, namely at DSL level, and at MLIR level. Relying on MLIR technology will allow for the composability of SRL with the remaining compilation flow [1]. Also developed are a streaming engine based on a custom RISC-V extension [2], supported by simulation tools [3], and a custom compiler [4]. Further steps include generating the respective backend code through the high-level expressiveness of the SRL.

REFERENCES

[1] M. C. da Silva, L. Sousa, N. Paulino, and J. Bispo, "A dsl and mlir dialect for streaming and vectorisation," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, I. Skliarova, P. Brox Jiménez, M. Véstias, and P. C. Diniz, Eds., 2024, pp. 181–190.

[2] J. M. Domingos, N. Neves, N. Roma, and P. Tomás, "Unlimited Vector Extension with Data Streaming Support," in *Proc. of the 48th Annual ACM/IEEE Intl. Symp. on Computer Architecture*, 2021, pp. 209–222.

[3] A. Fernandes, L. Crespo, N. Neves, P. Tomás, N. Roma, and G. Falcao, "Functional validation of the risc-v unlimited vector extension," *IEEE Embedded Systems Letters*, June 2024, accepted.

[4] N. Neves, J. M. Domingos, N. Roma, P. Tomás, and G. Falcao, "Compiling for Vector Extensions With Stream-Based Specialization," *IEEE Micro*, vol. 42, no. 5, pp. 49–58, 2022.