Etna: MLIR-Based System-Level Design and Optimization for Transparent Application Execution on CPU-FPGA Nodes

Stephanie Soldavini*§, Felix Suchert†§, Serena Curzel*, Michele Fiorito*, Karl Friebel†,
Fabrizio Ferrandi*, Radim Cmar‡, Jeronimo Castrillon† and Christian Pilato*
*Politecnico di Milano, Milano, Italy – Email: firstname.lastname@polimi.it
†TU Dresden, Dresden, Germany – Email: firstname.lastname@tu-dresden.de
‡Sygic, Bratislava, Slovakia – Email: rcmar@sygic.com
§Both authors contributed equally to this research.

I. The *Etna* Design Methodology

Specialized hardware is often key to accelerate big data applications [2, 3]. However, while High-Level Synthesis (HLS) has advanced considerably in the past decades, offloading to FPGAs still requires significant manual effort from platform experts [4]. This is especially the case for industrial applications and when kernels may execute, interchangeably, on CPU or FPGA. To reduce this effort, we present *Etna*, an integrated MLIR-based development approach for applications with retargetable kernels. As shown in Figure 1 (bottom), *Etna* takes as inputs a set of kernels for both CPU (C/C++) and FPGA execution (C/C++/MLIR for HLS), the FPGA description, and the MLIR representation of the application's dataflow graph (DFG). *Etna* supports **Application Composition**, **System Generation**, and integration with **HLS** tools for kernel synthesis. This is enabled by two novel MLIR dialects: `dfg` to describe the interactions among the kernels and `olympus` to describe the system-level architecture. `dfg` represents a generic graph model that can be extracted, e.g., from implicit dataflow languages [5]. In **Application Composition**, kernels marked as offloaded in the `dfg` dialect are lowered to `olympus` for hardware generation. The remaining kernels are lowered to LLVM-IR for code generation. *Olympus* takes the `olympus` representation of the offloaded portion of the DFG and performs **System Generation** to create an optimized system architecture and host drivers. For kernel **HLS** we use Bambu [1] for its unique support for data containers. The resulting HDL is instantiated within the system architecture. Finally, all CPU-side sources (application LLVM-IR, CPU kernel sources, FPGA kernel drivers) are linked to produce an executable.

II. Experimental Results

We evaluate *Etna* on the industry grade application of traffic analysis map matching, which processes sequences of noisy GPS trajectories of moving vehicles and generates speed annotations on road segments. Our experiments ran on an AMD EPYC 7282 CPU with 16 cores (i.e., 32 threads), 64 GB RAM and CentOS 7.9.2009. Attached to the system was a Xilinx Alveo u280 FPGA.

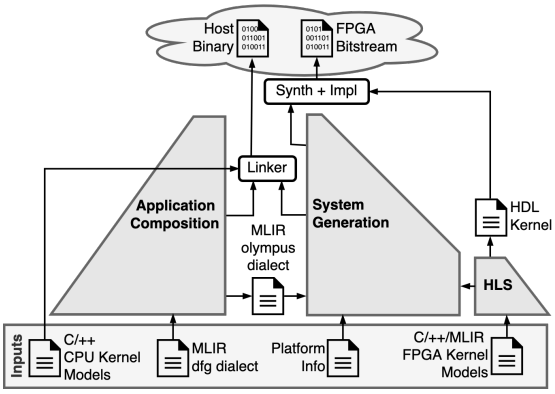

Fig. 1. The general architecture of the *Etna* flow.

Offloading the projection kernel achieved $1.9\times$ speedup over the pure CPU execution, as it is the most time consuming portion of the application. This kernel used 11.05% of the LUTs, and thus could be replicated 8 times for a utilization of $\sim$ 88%. The Xilinx Vitis Analyzer tool shows that the kernel executes 59% of the total runtime. This leads to a total achievable speedup of $\frac{1}{(1-0.59)+\frac{0.59}{8}} = 2.06$ (additional to the obtained $1.9\times$), for a total speedup over CPU of $3.9\times$.

References

[1] F. Ferrandi et al. "Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications". In: *Proc. of DAC*. 2021, pp. 1327–1330.
[2] C. Pilato et al. "A System Development Kit for Big Data Applications on FPGA-based Clusters: The EVEREST Approach". In: *Proc. of DATE*. Mar. 2024, 6 pp.
[3] C. Pilato et al. "EVEREST: A design environment for extreme-scale big data analytics on heterogeneous platforms". In: *Proc. of DATE*. 2021, pp. 1320–1325.
[4] S. Soldavini et al. "Automatic creation of high-bandwidth memory architectures from domain-specific languages: The case of computational fluid dynamics". In: *ACM TRETS* 16.2 (2023), pp. 1–34.
[5] F. Suchert et al. "ConDRust: Scalable Deterministic Concurrency from Verifiable Rust Programs". In: *ECOOP 2023*. LIPIcs. 2023, 33:1–33:39.

This work was partially funded by the EU Horizon 2020 Programme under grant agreement No 957269 (EVEREST).