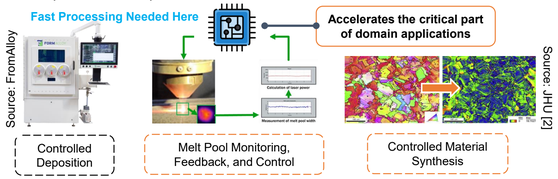
Accelerating Scientific Workflows with LLM-Driven Compiler Optimizations for Generated High-Performance Hardware

Max Ramstad, Nicolas Bohm Agostini, Antonino Tumeo

Motivations

- High-level synthesis (HLS) tools enable the rapid development of specialized hardware for FPGAs and ASICs but require expert domain scientists to optimize designs effectively.
- Target hardware (HW) platforms (FPGA or ASIC) exhibit diverse hardware characteristics that can be leveraged to accelerate the application.
- Large language models (LLMs) demonstrate proficiency in addressing domain-specific programming queries.
- Designing optimizations by hand proves to be complex and cumbersome, even for experienced developers.
- Domain scientists require optimized workloads but often lack the resources to implement the optimizations themselves.



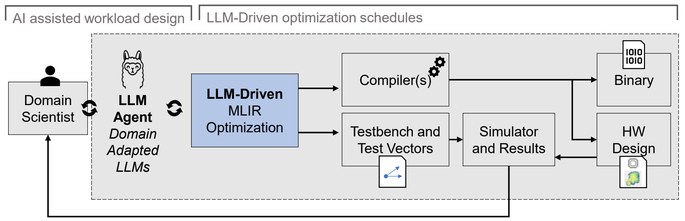
Challenges

- **Code correctness:** Ensuring generated MLIR transformation schedules are syntactically valid and free of bugs remains critical for code correctness.
- **Optimization Quality:** Addressing potential suboptimal transformations produced by LLMs due to their limited understanding of nuanced hardware behaviors is essential for maintaining optimization quality.
- **Optimal prompting:** Designing prompts that accurately capture the optimization requirements of complex kernels proves to be a tedious and involved process.

Approach

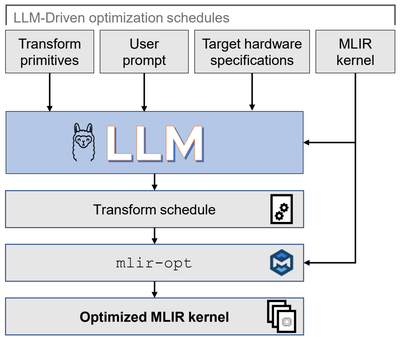
- Providing the LLM with additional target and optimization context to enhance accuracy.
- Leveraging the LLM agent to automate the creation of MLIR transform schedules using information about the target architecture.
- Creating a library of transform primitives and examples to improve LLM accuracy.
- Building solutions with the MLIR transform dialect, eliminating the need to recompile the compiler.
- Integrating seamlessly into state-of-the-art HLS flows (SODA) [3] or converting optimized kernels into host code.

End-to-End Flow



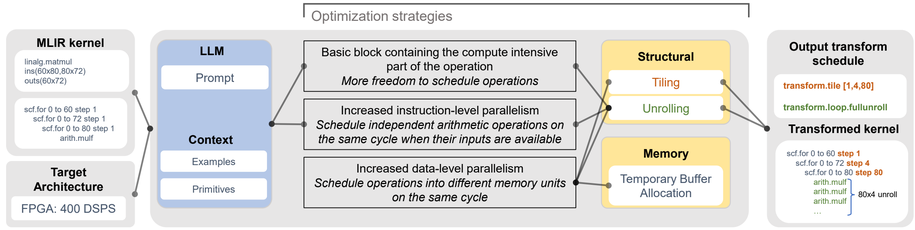
- Consuming optimized kernels through HLS to generate high-quality hardware designs.
- Utilizing an LLM agent with MLIR [1] compilers to transform an MLIR kernel into an optimized kernel for the HLS flow.
- Targeting FPGAs and ASICs currently, but a similar system can be enabled to optimize computational kernels for other hardware targets such as CPUs and GPUs.

Methods



- Providing the LLM with transform dialect primitives and examples to ensure the generation of syntactically correct transformation schedules.
- Exposing target hardware specifications to highlight important architectural characteristics for kernel implementation.
- Using the MLIR kernel as the initial unoptimized code targeted by the generated transform schedule.
- Guiding the LLM with a user prompt to reason and generate an applicable transform schedule, factoring in user requests or modifications to optimize the kernel for the target architecture.

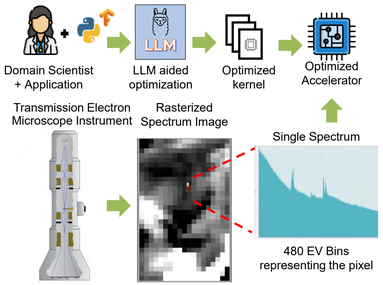
MLIR Transform Example

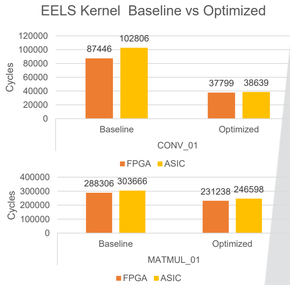


- LLM reasons that the target architecture can be optimized by tiling to match the number of computational units and fully unrolling the tile.
- LLM creates transform schedule that tiles and unrolls by [1, 4, 80], enabling 360 concurrent operations to be executed in parallel by the accelerator.

EELS Encoder Model

- Target: Data processing pipeline of scientific instruments of DOE labs, such as **electron energy loss spectroscopy (EELS)**
- EELS data is often noisy and high-dimensional and demands computational models (encoders) that can perform robust denoising and representation of data prior to classification [2].
- Measurable speedups observed during inference thanks to generated accelerators for critical kernels.
- Latency gains enable real-time controlled material synthesis.





References

[1] Lattner, Chris, et al. "MLIR: Scaling compiler infrastructure for domain specific computation." *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2021.
[2] Hollenbach, Jonathan D., et al. "Embedding theory in ML toward real-time tracking of structural dynamics through hyperspectral datasets." *arXiv preprint arXiv:2312.05201* (2023).
[3] Bohm Agostini, Nicolas, et al. "Bridging python to silicon: The SODA toolchain." *IEEE Micro* 42.5 (2022):

U.S. DEPARTMENT OF ENERGY BATTELLE
PNNL is operated by Battelle for the U.S. Department of Energy

SODA Benchmarks

Panda-Bambu HLS (v 0.9.7)

Logical EELS Encoder

SODA Benchmarks: https://github.com/pnnl/soda-benchmarks
Panda-Bambu HLS: https://panda.dei.polimi.it
Logical EELS Encoder: https://github.com/hollejd1/logicalEELS

Pacific Northwest NATIONAL LABORATORY
www.pnnl.gov