FortranX: Harnessing Code Generation, Portability, and Heterogeneity in Fortran

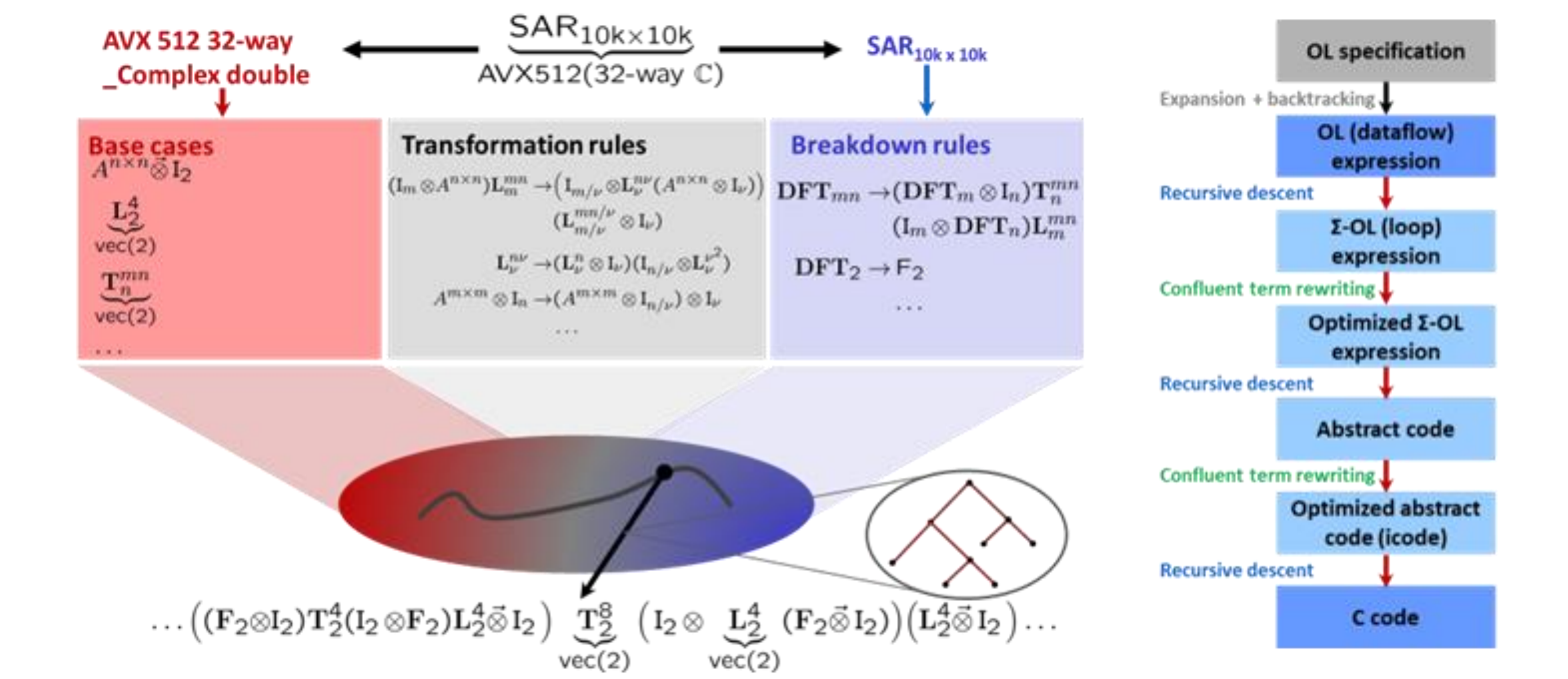
Sanil Rao*, Mike Franusich+, Mohammad Alaul Haque Monil†, Het Mankad†, Jeffery S. Vetter†, Franz Franchetti*

*Carnegie Mellon University, +SpiralGen Inc., †Oak Ridge National Laboratory

Problem

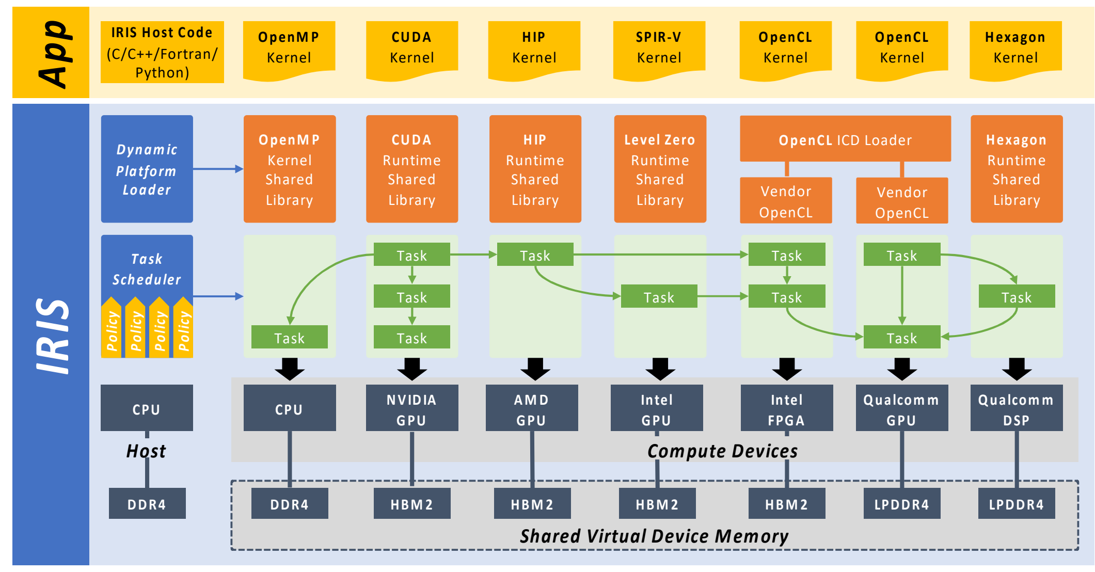
- Many scientific applications are written and maintained in Fortran.
- New systems leverage novel programming models and hardware architectures which are not easily accessible in Fortran.
- FortranX is an automated approach to recognize and optimize Fortran applications, generating optimized kernels that can execute on various hardware platforms without source code modification.

IRISX[3]: SPIRAL + IRIS


AVX 512 32-way _Complex double
$\underbrace{\mathrm{SAR}_{10k\times 10k}}_{\mathrm{AVX512(32\text{-}way}\ \mathbb{C})}$
$\mathrm{SAR}_{10k\times 10k}$
Base cases
$A^{n\times n}\vec{\otimes}\mathrm{I}_2$
$\underbrace{\mathrm{L}_2^4}_{\mathrm{vec}(2)}$
$\underbrace{\mathrm{T}_n^{mn}}_{\mathrm{vec}(2)}$
Transformation rules
$(\mathrm{I}_m\otimes A^{n\times n})\mathrm{L}_m^{mn}\to(\mathrm{I}_{m/\nu}\otimes\mathrm{L}_\nu^{n\nu}(A^{n\times n}\otimes\mathrm{I}_\nu))(\mathrm{L}_{m/\nu}^{mn/\nu}\otimes\mathrm{I}_\nu)$
$\mathrm{L}_\nu^{n\nu}\to(\mathrm{L}_\nu^n\otimes\mathrm{I}_\nu)(\mathrm{I}_{n/\nu}\otimes\mathrm{L}_\nu^{\nu^2})$
$A^{m\times m}\otimes\mathrm{I}_n\to(A^{m\times m}\otimes\mathrm{I}_{n/\nu})\otimes\mathrm{I}_\nu$
Breakdown rules
$\mathrm{DFT}_{mn}\to(\mathrm{DFT}_m\otimes\mathrm{I}_n)\mathrm{T}_n^{mn}(\mathrm{I}_m\otimes\mathrm{DFT}_n)\mathrm{L}_m^{mn}$
$\mathrm{DFT}_2\to\mathrm{F}_2$
$\dots\big((\mathrm{F}_2\otimes\mathrm{I}_2)\mathrm{T}_2^4(\mathrm{I}_2\otimes\mathrm{F}_2)\mathrm{L}_2^4\vec{\otimes}\mathrm{I}_2\big)\underbrace{\mathrm{T}_2^8}_{\mathrm{vec}(2)}\big(\mathrm{I}_2\otimes\underbrace{\mathrm{L}_2^4}_{\mathrm{vec}(2)}(\mathrm{F}_2\vec{\otimes}\mathrm{I}_2)\big)\big(\mathrm{L}_2^4\vec{\otimes}\mathrm{I}_2\big)\dots$
OL specification
Expansion + backtracking
OL (dataflow) expression
Recursive descent
Σ-OL (loop) expression
Confluent term rewriting
Optimized Σ-OL expression
Recursive descent
Abstract code
Confluent term rewriting
Optimized abstract code (icode)
Recursive descent
C code

SPIRAL[1]:

- Complete automation of the implementation and optimization task.
- Rewriting systems to perform algorithmic selection and optimization for various hardware backends.



IRIS[2]:

- Portable runtime system supporting various hardware backends.
- Heterogenous execution by utilizing multiple devices for concurrent tasks.

Acknowledgment

This work is funded, in part, by Bluestone, a X-Stack project in the DOE Advanced Scientific Computing Office with program manager Hal Finkel. This research used resources of the Experimental Computing Laboratory (ExCL) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

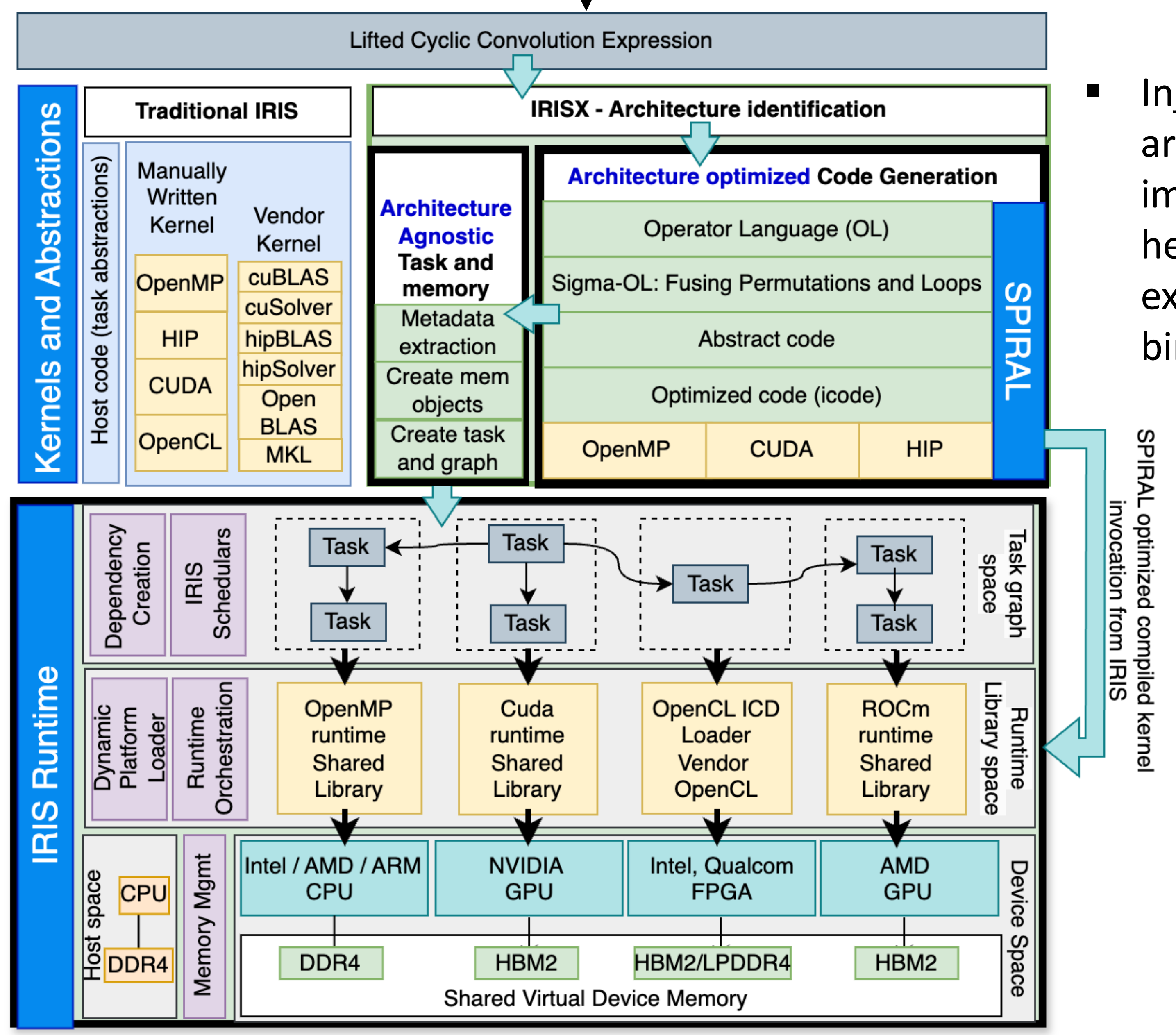
Cyclic Convolution Example

```
program cyclic_convolution
  allocate(out_arr, in_arr, sym_arr, tmp_box)

  planf = fftw_plan_dft_r2c_3d(dims(3),dims(2),
          dims(1),in_arr, tmp_box, FFTW_ESTIMATE)
  planb = fftw_plan_dft_c2r_3d(dims(3),dims(2),
          dims(1),tmp_box, out_arr, FFTW_ESTIMATE)

  ! forward FFT
  call fftw_execute_dft_r2c(planf, in_arr, tmp_box)

  ! pointwise multiply
  tmp_box = tmp_box * sym_arr

  ! inverse FFT
  call fftw_execute_dft_c2r(planb, tmp_box, out_arr)

  call fftw_destroy_plan(planf)
  call fftw_destroy_plan(planb)
end program cyclic_convolution
```

- Optimization by fusing operations (merging multiplication into FFT stages) and removing unneeded temporaries.

Idiomatic Translation Compiler Pass

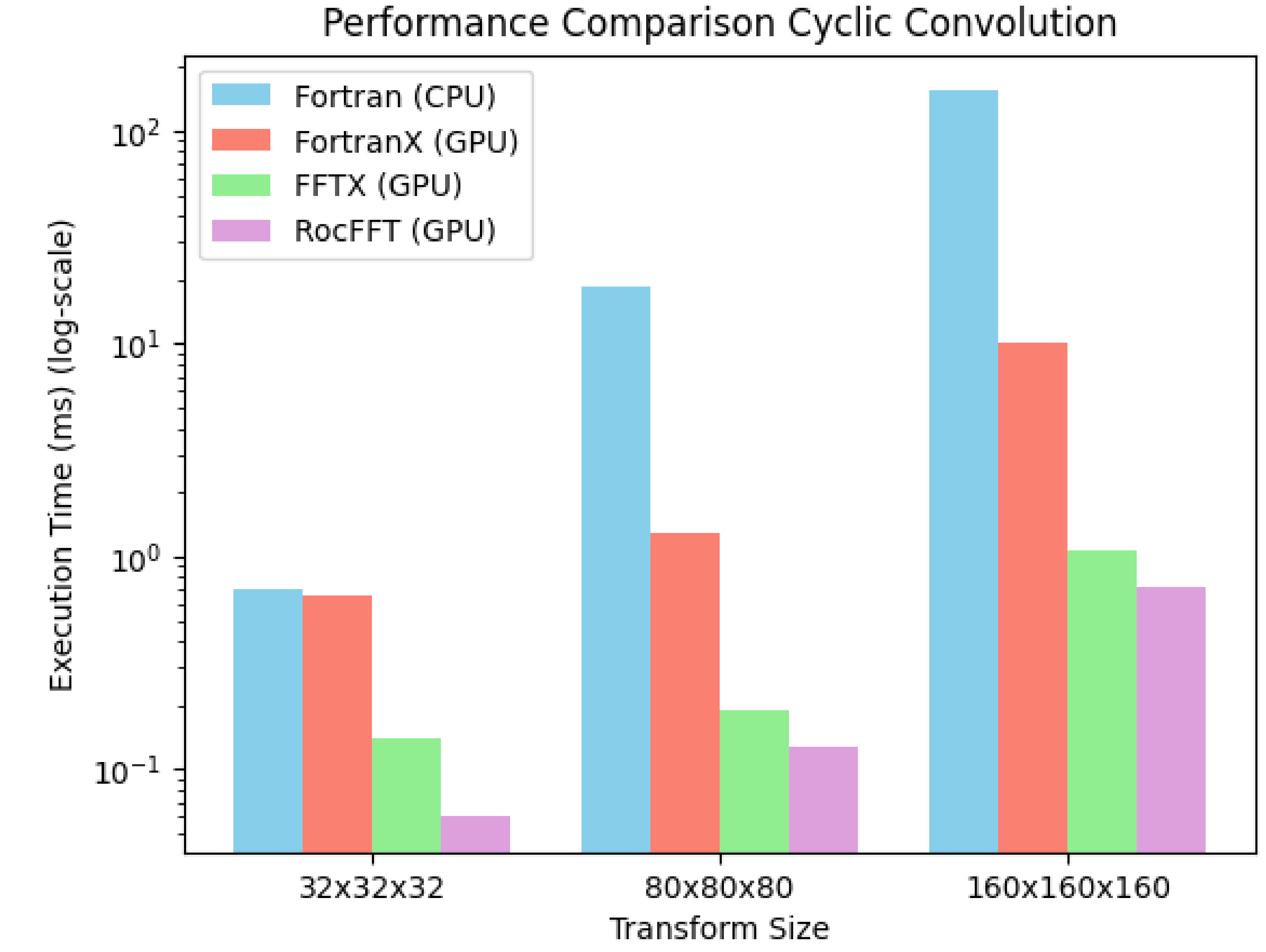
- Compiler pass that recognizes the idiom of cyclic convolution.



- Injection of IRISX for architecture-specific implementations and heterogenous task execution in a portable binary.

a.out

Performance



- Without source code modification, FortranX achieves up to **15x** performance improvement over the sequential CPU baseline and can be further optimized for specific hardware and overhead reduction.

Conclusion & Future Work

- FortranX enables the compiler to recognize key algorithms and leverages IRISX for optimized kernel generation and task execution.
- We are continuing development of FortranX for large cross-motif applications including sparse linear algebra and structured grids.

References

[1] F. Franchetti, T.-M. Low, T. Popovici, R. Veras, D. G. Spampinato, J. Johnson, M. Puschel, J. C. Hoe, and J. M. F. Moura, “SPIRAL: Extreme performance portability,” Proceedings of the IEEE, special issue on “From High Level Specification to High Performance Code”, vol. 106, no. 11, 2018.

[2] J. Kim, S. Lee, B. Johnston, and J. S. Vetter, “IRIS: A portable runtime system exploiting multiple heterogeneous programming systems,” in 2021 IEEE High Performance Extreme Computing Conference, HPEC 2021, Waltham, MA, USA, September 20-24, 2021, pp. 1–8, IEEE, 2021.

[3] Sanil Rao, Mohammad Alaul Haque Monil, Het Mankad, Jeffrey Vetter, and Franz Franchetti. 2023. FFTX-IRIS: Towards Performance Portability and Heterogeneity for SPIRAL Generated Code. In Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23). Association for Computing Machinery, New York, NY, USA, 1635–1641. https://doi.org/10.1145/3624062.3624242