Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

A Proof-of-Concept Performance Portable SYCL-based Fast Fourier Transformation Library

10th International Workshop on OpenCL and SYCL

Vincent R. Pascuzzi
Mehdi Goli (Codeplay)

May 11, 2022

@BrookhavenLab

DOE HPC Systems

2013
~10 PFLOPS

2016
~10 PFLOPS

2018
~100 PFLOPS

2021
~100 PFLOPS

2022
~1000 PFLOPS

Homogeneous

Heterogenous

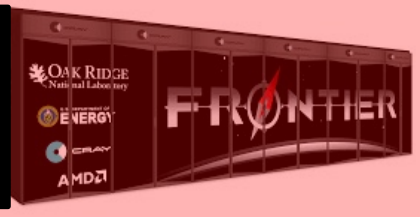
DOE HPC Systems

2013	2016	2018	2021	2022+
~10 PFLOPS	~10 PFLOPS	~100 PFLOPS	~100 PFLOPS	~1000 PFLOPS

"EASY"

Homogeneous

"HARD"



Heterogenous

HEP Center for Computational Excellence (CCE)

A Department of Energy High-Energy Physics program investigating:

- Performance portability
- I/O
- Complex workflows
- Event generators*

Portable Parallelization Strategies (PPS) effort focuses on performance and portability solutions for current and future HEP software

- Select among the participating experiments a number of x86-based ‘testbeds’ and rewrite the codes in various programming models

NVIDIA CUDA

OpenMP

SYCL™

kokkos

* Event generator software is written and maintained by theorists.

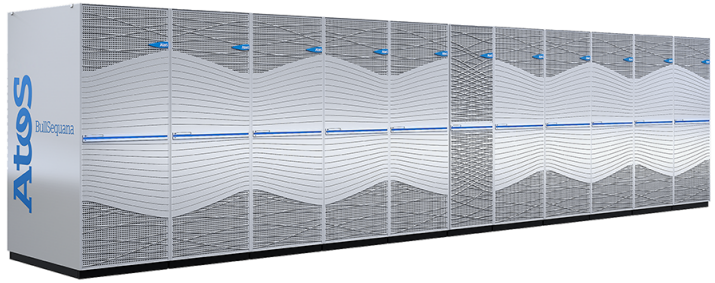
Brookhaven National Laboratory

Limited number of developers (we are physicists) and there are numerous platforms with various architectures

- Large codebases which need to stand the ‘test of time” (~decade)
- We cannot afford to support and maintain multiple codebases
- We need to utilize leadership computing facilities.
- We need portability and attain a fair level of performance.

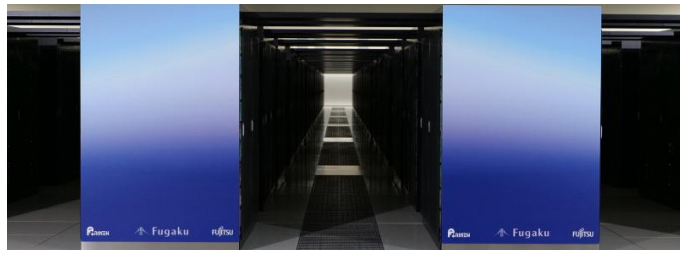
LUMI (Finland)
AMD CPU, AMD GPU

Leonardo (Italy)
Intel CPU, NVIDIA GPU



Swiss National Supercomputing Center (CSCS), 2023
NVIDIA/ARM CPU, NVIDIA GPU

Riken Center for Computational Science, 2021
ARM CPU



-based libraries

Decouple

- from vendors providing single architecture/platform libraries
- boiler-plate code when using interoperability

Readability, maintainability and sustainability

- backwards compatibility (*e.g.*, APIs constantly evolving ⟹ maintain interoperable implementations)
- source polluted with macros – e.g., `#ifdef` – to deal with many backends
- foster inclusivity: community-driven open-source projects

And more...

- provide template metaprogramming for users to optimize for their target hardware (we don't care about what they want to run on, only that it does run)
- auto-tuning mechanisms: query available devices and configure/tune implementation for automatically optimizing parameters for a given device
- elevate reproducibility as first-class requirement

Performance portability

"An application is performance portable if it achieves a consistent ratio of the actual time to solution to either the best-known or the theoretical best time to solution on each platform with minimal platform specific code required." [1]

Useful metric should[2]:

- Be measured specific to a set of platforms of interest H
- Be independent of the absolute performance across H
- Be zero if a platform in H is unsupported, and approach zero as the performance of platforms in H approach zero
- Increase if performance increases on any platform in H
- Be directly proportional to the sum of scores across H

Performance

- It runs: {Yes, No}
- It runs efficiently with respect to some baseline

Portability

- Can execute on multiple systems
- Adaptable to varying architectures and platforms

$$\mathcal{P}(a,p;H) = \begin{cases} \dfrac{|H|}{\sum_{i \in H} \frac{1}{e_i(a,p)}} & \text{if } i \text{ is supported } \forall_i \in H \\ 0 & \text{otherwise} \end{cases}$$

Productivity

- SLoC, maintainability, sustainability
- Port/migration/translation

Reproducibility

- Crucial for most science
- Results (required precision) cannot depend on hardware

[1] (definition of) Performance Portability, 2016 Department of Energy Center of Excellence Meeting.
[2] Pennycook *et al.* (2019).

SYCL-FFT

Based on Cooley-Tukey (see backup)

- Implements radix-{2,4,8} algorithms

Header-only

- Functor templated class, three template arguments
- `WG_FACTOR` depends on input sequence length, determined *a priori*

Proof-of-concept

- Limited to 1D
- C2C up to 2^{11} length
- Computed out-of-place

```
template <typename T, size_t WG_SIZE,
  int WG_FACTOR, int SYCL_LANGUAGE_VERSION>
class fft_1d {
 public:
  using float2 = sycl::float2;
  if constexpr (SYCL_LANGUAGE_VERSION < 202000) {
    using size_accessor =
      sycl::accessor<size_t, 1,
        sycl::access::mode::read,
        sycl::access::target::global_buffer>;
    using read_accessor =
      sycl::accessor<T, 1,
        sycl::access::mode::read,
        sycl::access::target::global_buffer>;
    using write_accessor =
      sycl::accessor<T, 1,
        sycl::access::mode::write,
        sycl::access::target::global_buffer>;
  } else {
    using stage_accessor =
      sycl::accessor<size_t, 1,
        sycl::access::mode::read,
        sycl::access::target::device>;
    using read_accessor =
      sycl::accessor<T, 1,
        sycl::access::mode::read,
        sycl::access::target::device>;
    using write_accessor =
      sycl::accessor<T, 1,
        sycl::access::mode::write,
        sycl::access::target::device>;
  }
```

$$X_k = E_k + \omega_N^k O_k + \omega_N^{3k} O'_k$$

$$X_{k+N/2} = E_k - (\omega_N^k O_k + \omega_N^{3k} O'_k)$$

$$X_{k+N/4} = E_{k+N/4} - i(\omega_N^k O_k - \omega_N^{3k} O'_k)$$

$$X_{k+3N/4} = E_{k+N/4} + i(\omega_N^k O_k - \omega_N^{3k} O'_k)$$

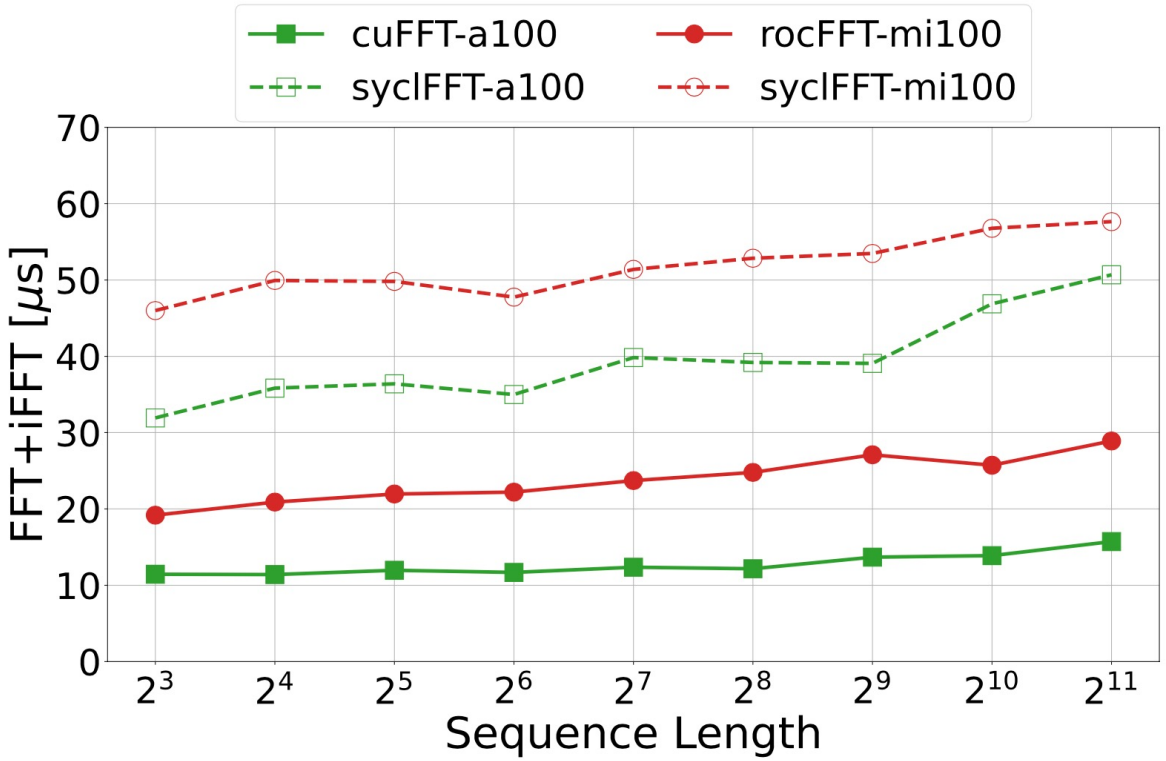
```
  using local_rw_accessor =
    sycl::accessor<T, 1,
      sycl::access::mode::read_write,
      sycl::access::target::local>;
  if constexpr (SYCL_LANGUAGE_VERSION >= 202000) {
    [[sycl::reqd_work_group_size(WG_SIZE)]]
  }
  fft1d(size_accessor stage_sizes,
    read_accessor inputs,
    write_accessor outputs,
    local_rw_accessor local_shared,
    size_t work_group_size,
    int direction);
  void operator()(sycl::nd_item<1> item) const {...}
  inline void radix_2(sycl::nd_item<1> item,
    size_t stage_mod,
    float2* temp) const {...}
  inline void radix_4(sycl::nd_item<1> item,
    size_t stage_mod,
    float2* temp) const {...}
  inline void radix_8(sycl::nd_item<1> item,
    size_t stage_mod,
    float2* temp) const {...}
};
```

Experimental setup

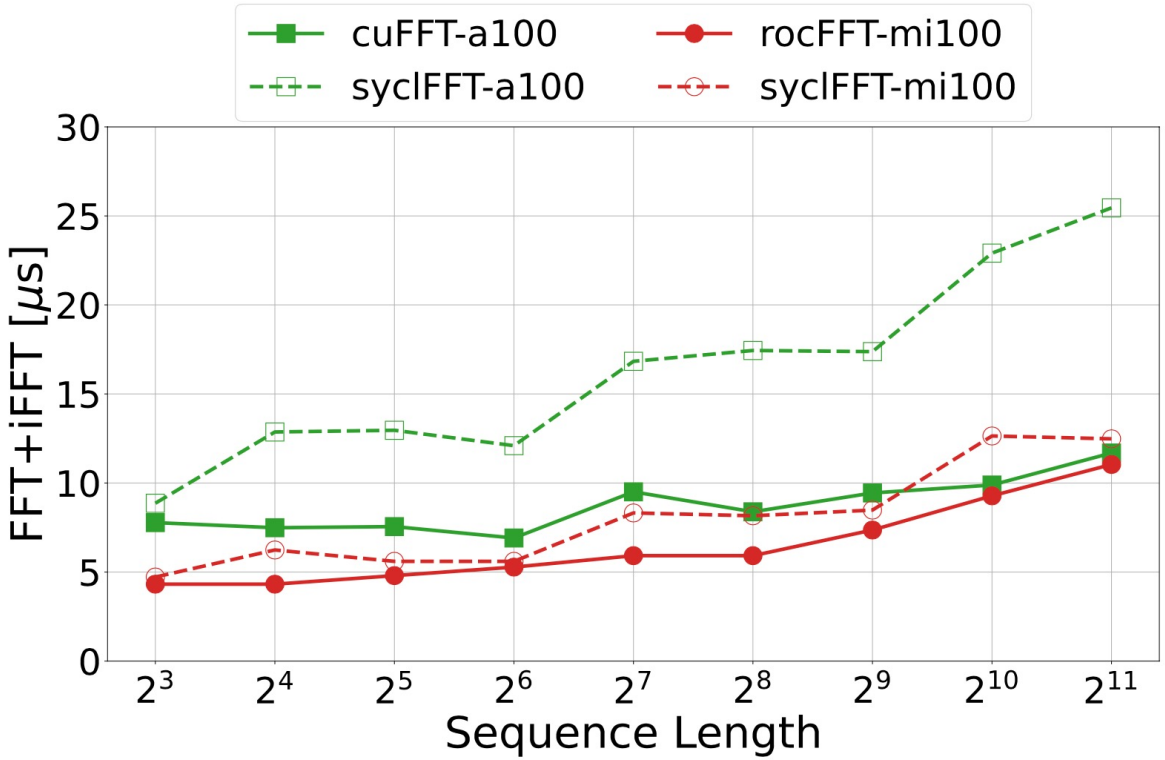
Device (Architecture)	Maximum Work-Group Size	Backend	Compiler(s)	Native Library
ARM Neoverse-N1 (ARMv8-A)	4096	POCL 1.9 (pre-gde9b966b)	ComputeCpp 2.8.0 `armclang 21.1`	ARM PL 21.1
Intel Xeon E3-1585 v5 (x86_64)	8192	OpenCL 3.0 oneTBB 2022.0.2	ComputeCpp 2.8.0 `sycl-nightly/20220223`	oneMKL 2022.0.2
AMD MI-100 (CDNA)	256	HIP 4.2.0	`sycl-nightly/20220223` `hipcc 4.2.21155`	rocfft 4.2.0
NVIDIA A100 (Ampere)	1024	PTX64	`sycl-nightly/20220223` `nvcc 11.5.0`	cufft 11.5.0
Intel Iris P580 (Gen9)*	256	OpenCL 3.0	ComputeCpp 2.8.0	—

Table 1: Device hardware and software versions for each platform considered in these studies. The `sycl-nightly/x` compilers refer to the specific branch of the Intel LLVM compiler project. All systems run openSUSE 15.3, kernel version 5.3.18. No native libraries exist for Intel Iris P580 iGPU (marked by asterisk); performance plots from this device are presented in App. A.

Computational Performance Ampere and CDNA

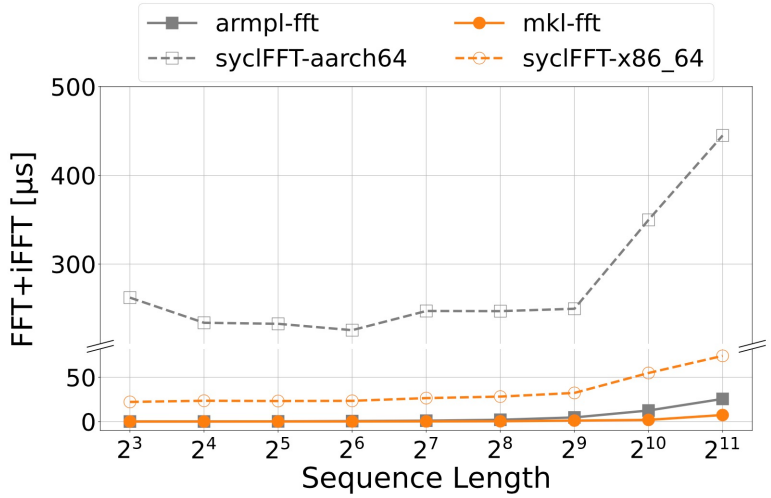


(a) Smallest total (kernel dispatch + execution) execution times.

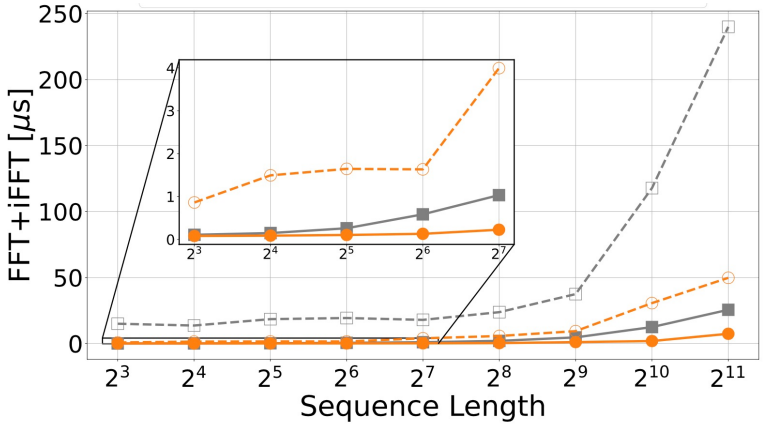


(b) Smallest kernel execution times.

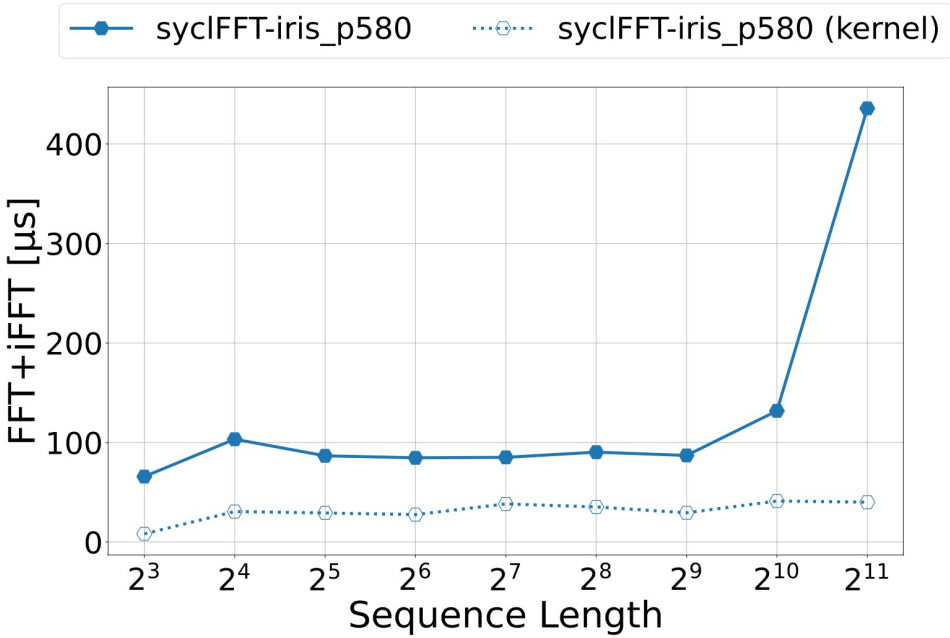
Computational Performance RISC, x86 and Gen9



(a) Smallest total (kernel dispatch + execution) execution times.

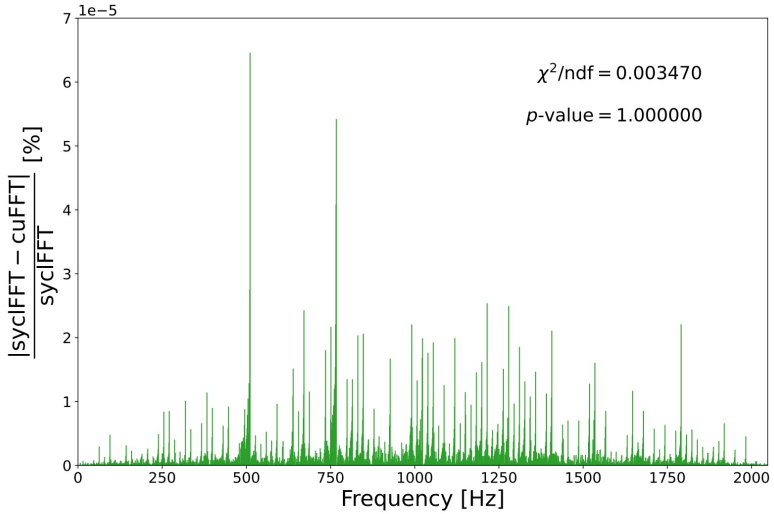


(b) Smallest kernel execution times.

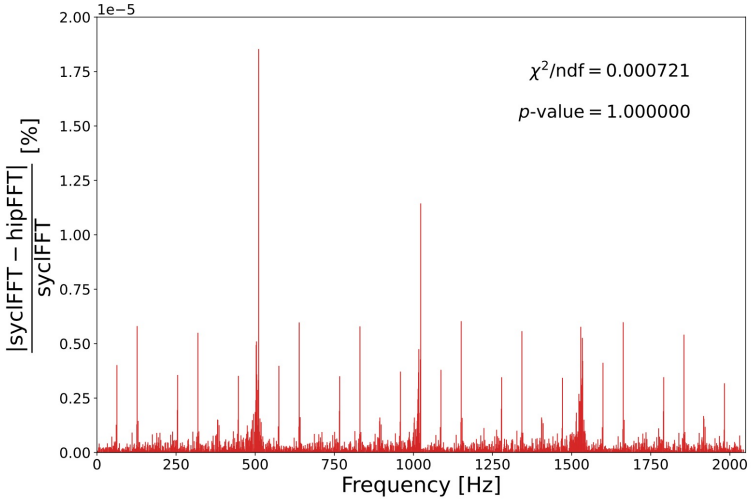


Reproducibility

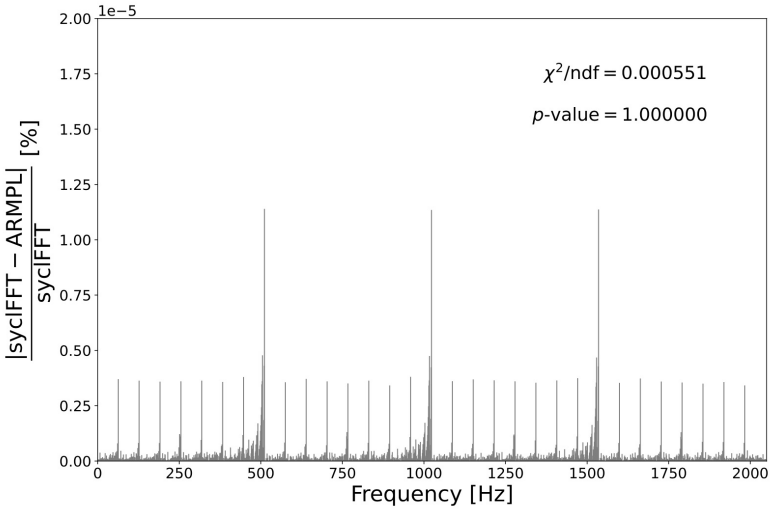
$$\chi^2_{\text{reduced}} = \sum_i^N \frac{(s_i - n_i)^2}{n_i} \frac{1}{\text{ndf}}$$



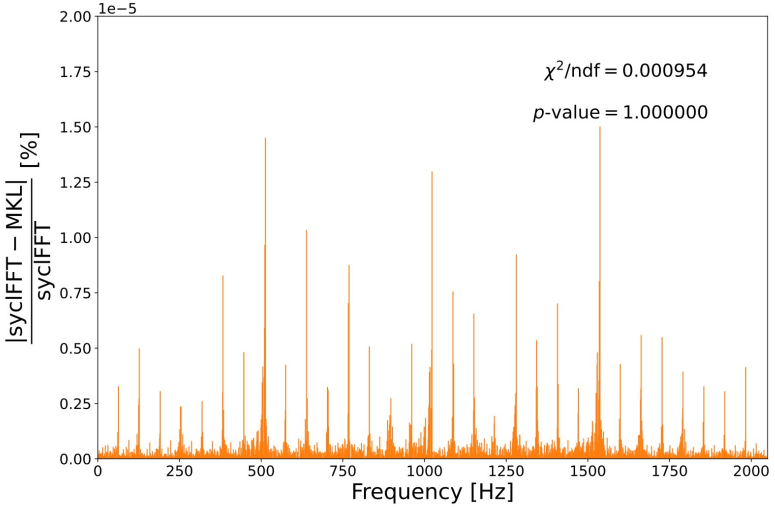
(a) NVIDIA A100



(b) AMD MI-100



(a) ARM Neoverse



(b) Intel x86_64

Conclusions

Help grow SYCL ecosystem with purely SYCL-based libraries

- Decouple from vendors; improve readability, maintainability and focus on reproducibility
- Foster inclusive community-driven open-source software projects

PoC SYCL-FFT

- Limited to 1D, C2C, lengths up to 2^{11}
- To our knowledge, first demonstration of intel/llvm HIP backend
- Kernel runtime competitive with vendor-optimized libraries
- Small kernels suffer from launch latencies, sporadic and highly fluctuate on some hardware

Ongoing and future work

- Continued improvements to SYCL backend implementations and offloading mechanisms potential close gaps
- SYCL-FFT support for {2,3}D FFT, accommodate arbitrary input sizes
- Further analyses including hipSYCL (highly optimized HIP backend)

Conclusions

Special ‘thanks’ to our ANL friends---**Benjamin Allen**, **Kalyan Kumaran** and **Kevin Harms**---for all their support and access to test machines!

Help grow SYCL ecosystem with purely SYCL-based libraries

- Decouple from vendors; improve readability, maintainability and focus on reproducibility
- Foster inclusive community-driven open-source software projects

PoC SYCL-FFT

- Limited to 1D, C2C, lengths up to 2^{11}
- To our knowledge, first demonstration of intel/llvm HIP backend
- Kernel runtime competitive with vendor-optimized libraries
- Small kernels suffer from launch latencies, sporadic and highly fluctuate on some hardware

Ongoing and future work

- Continued improvements to SYCL backend implementations and offloading mechanisms potential close gaps
- SYCL-FFT support for {2,3}D FFT, accommodate arbitrary input sizes
- Further analyses including hipSYCL (highly optimized HIP backend)

Backup

FFT 101

- Discretize input function (time/space), map to frequency domain

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} = \sum_{n=0}^{N-1} x_n \omega_N^{kn} \qquad x_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n \omega_N^{-kn}$$

,

- Cooley-Tukey: exploit periodicity, chunk length-*N* FFT into smaller ones

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} \omega_{N/2}^{(2n)k/N} + \sum_{n=0}^{N/2-1} x_{2n+1} \omega_{N/2}^{(2n+1)k/N}$$

$$X_k = \sum_{n_2=0}^{N/2-1} x_{2n_2} \omega_{N/2}^{n_2 k} + \sum_{n_4=0}^{N/4-1} \left(\omega_N^k x_{4n_4+1} \omega_{N/4}^{n_4 k} + \omega_N^{3k} x_{4n_4+3} \omega_{N/4}^{n_4 k} \right)$$

$T(N) = N \log n$
with $\log n$ splits

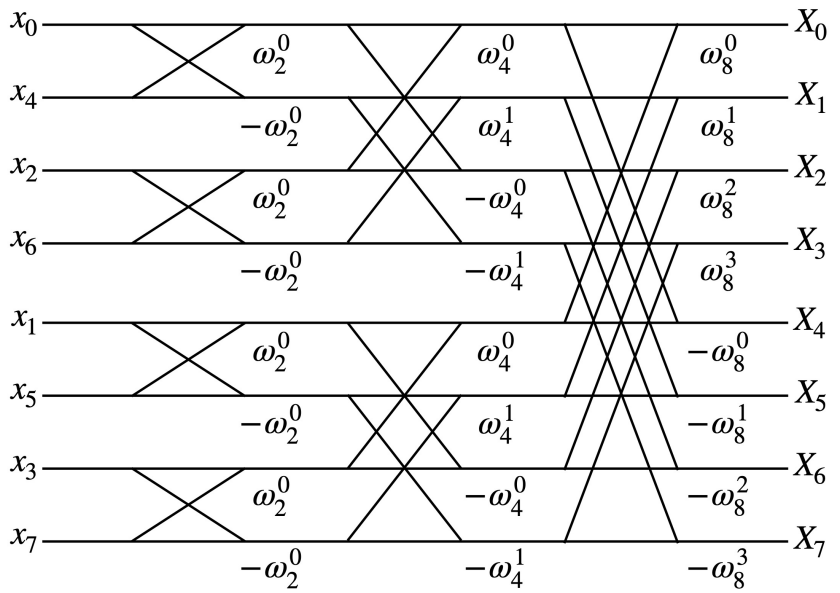


Figure 1: Illustration of a radix-2 DIT on an DFT with input size $N = 8$. Intersecting vertical lines—displaying butterfly-like patterns—among the input, $\{x_i\}$, correspond to combinations of additions and subtractions as per the twiddle factors, ω_N^k; see main text.

Auxiliary Figures

Device	Compiler + Backend	Launch Latency [μs]
ARM Neoverse-N1	ComputeCpp 2.8.0 + POCL 1.9	200-250
Intel Xeon E3-1585 v5	ComputeCpp 2.8.0 + OpenCL 3.0	~ 50
Intel Iris P580	ComputeCpp + OpenCL 3.0	650-800
AMD MI-100	Intel LLVM + HIP 4.2.0	~ 80
NVIDIA A100	Intel LLVM + CUDA 11.5.0	~ 40 (13)

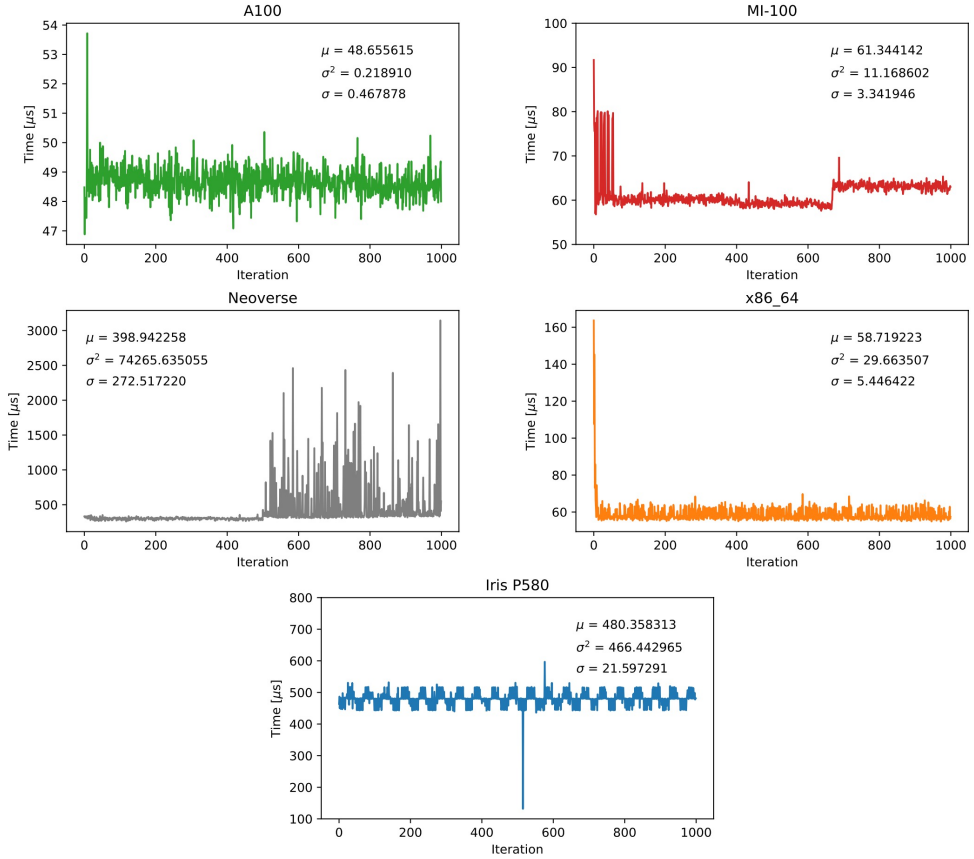


Figure 7: Distributions of 1000 combined kernel launch and execution times of SYCL-FFT across all hardware for an input sequence length of 1024.

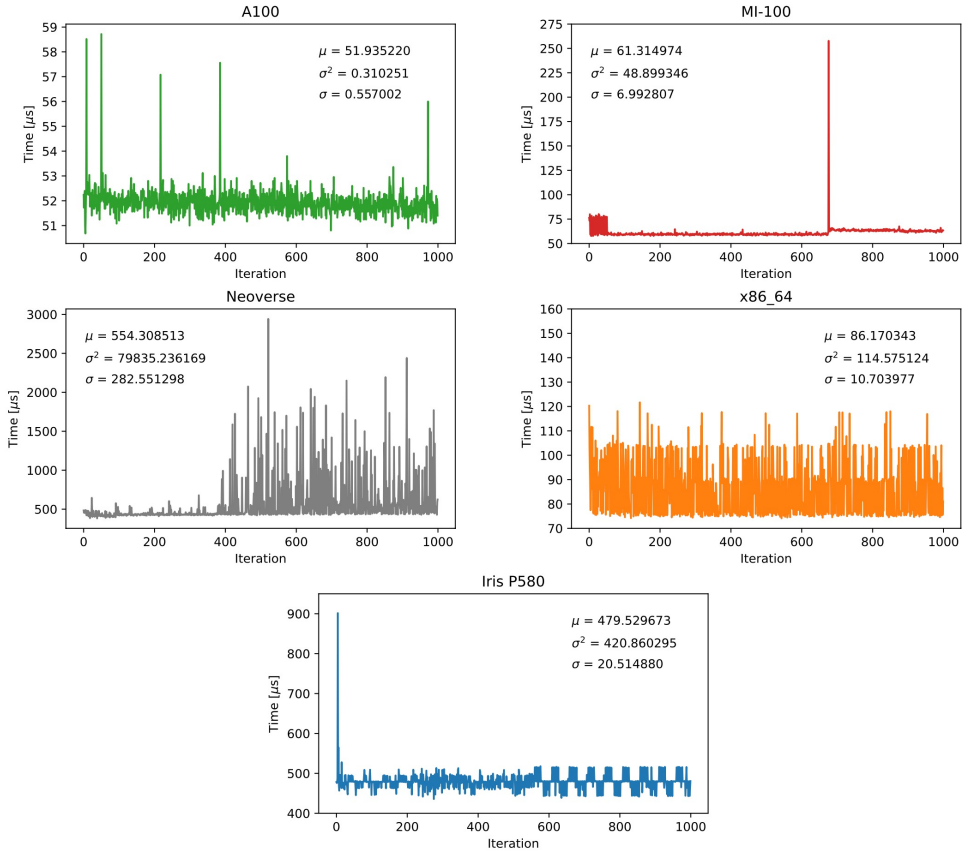


Figure 8: Distributions of 1000 combined kernel launch and execution times of SYCL-FFT across all hardware for an input sequence length of 2048.