Optimizing the Fast Fourier Transform using Mixed-Precision on Tensor Core Hardware

Anumeena Sorna, Xiaohe Cheng, Eduardo D'Azevedo, Kwai Wong, & Stanimire Tomov

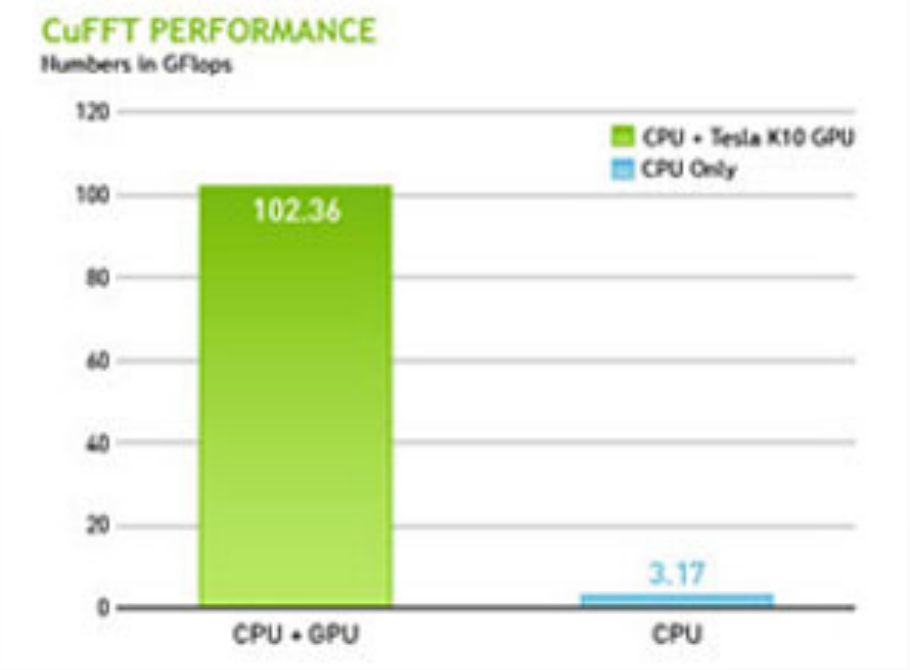
Background

Fast Fourier Transform is a useful tool in many high performance computing applications

FFT computation has high parallelism

Utilize parallel architecture: GPUs, NVIDIA CUDA

State of the art: NVIDIA CUDA Fast Fourier Transform library (cuFFT)

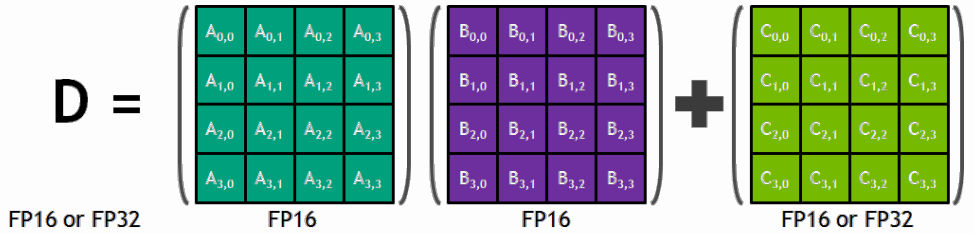


Motivation

Tensor cores in new Volta GPU architecture

Deliver up to 125 Tensor TFLOPS

Speedup FFT calculation if fully utilized



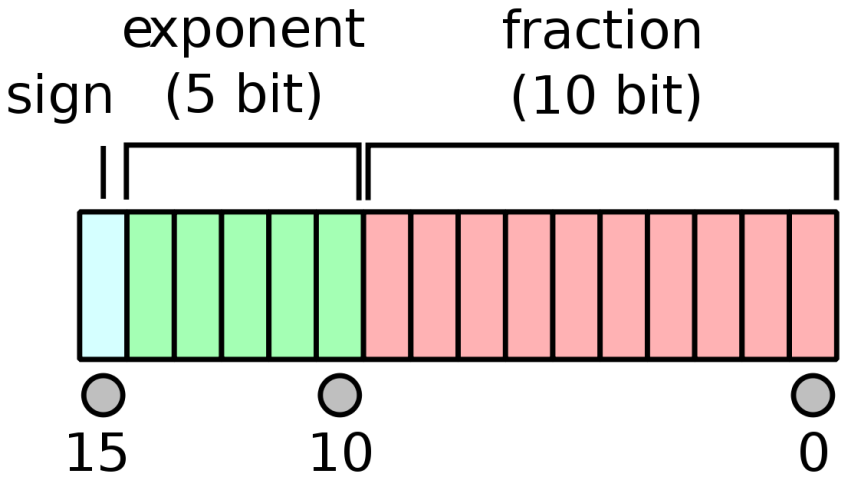
Motivation

cuFFT has yet to utilize tensor cores due to accuracy limitations

Half precision number:

- 65504 (max half precision)
- 6.10352×10^{-5} (minimum positive normal)
- 1.0009765625 (next smallest float after 1)

The narrow dynamic range does not satisfy the requirements of scientific applications



Motivation

Mixed-precision approach in computational physics

- Single precision calculation is faster than double
- Mix 32-bit and 64-bit arithmetic for acceleration
- Design algorithm to maintain 64-bit accuracy

Favorable properties of FFT

- Linearity: Straightforward splitting and combination
- Numerical stability: preserve norm, avoid error propagation

Methodology - FFT

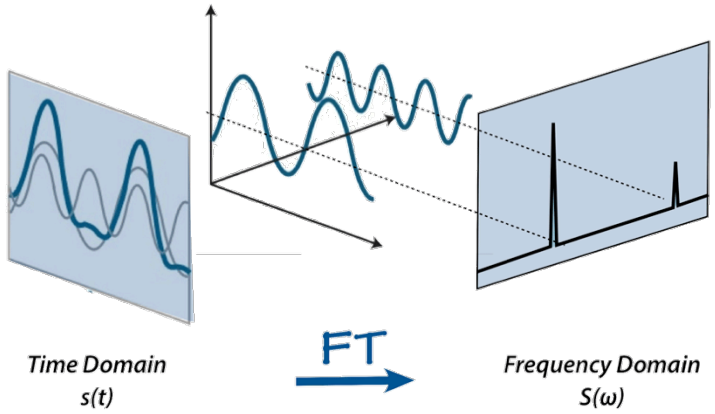
Every signal can be broken down into signals of different frequencies

The Discrete Fourier transform converts a time domain signal to a frequency domain signal:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk}$$

Inverse:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\left(\frac{2\pi}{N}\right)nk}$$



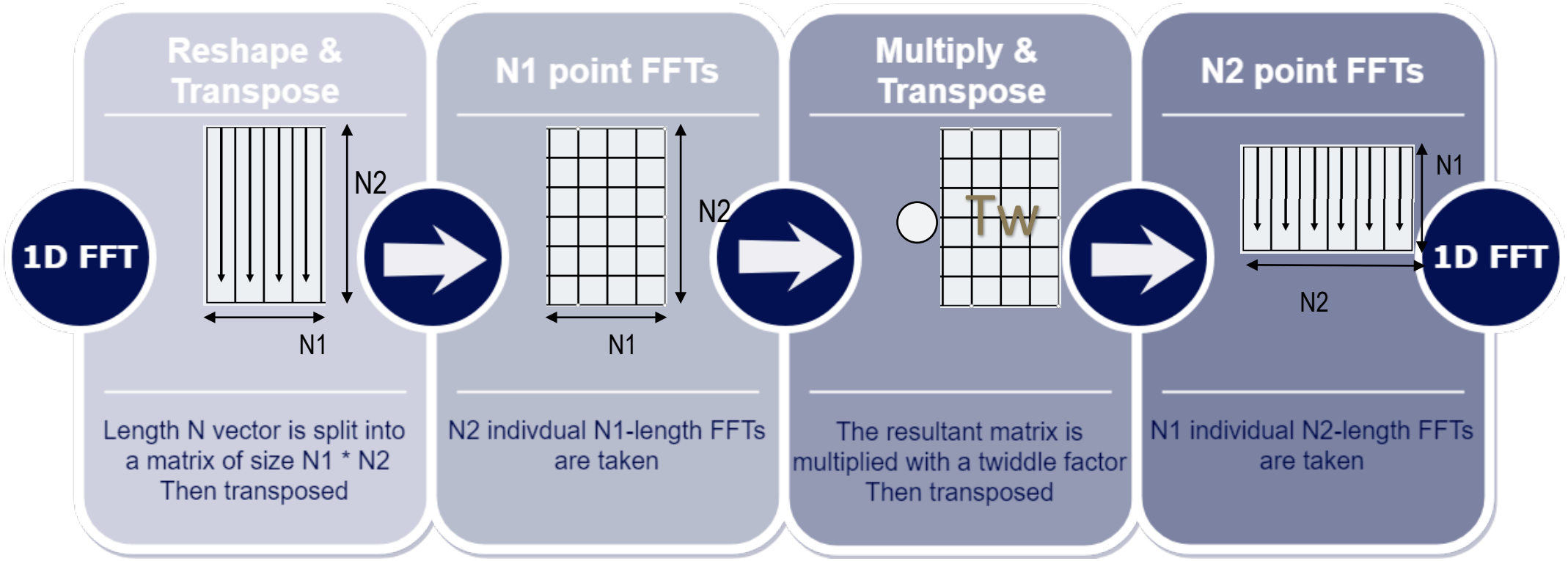
Methodology - FFT

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk}$$

It can be expressed as matrix multiplication:

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

Methodology – FFT ..



Methodology – FFT ..

N1 is chosen to be 4, as 4*4 Fourier matrix is accurately representable in FP16

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$F_{4real} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}$$

$$F_{4imag} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

Methodology - Splitting

Matrix multiplication is needed in:

- The base case of recursion
- The N1-point FFT

We split the FP32 input before multiplication

And carry out matrix multiplication in FP16

$$K(:) = \alpha X{\downarrow}hi\,(:) + \beta X{\downarrow}lo\,(:)$$

Rescale the results and combine them together

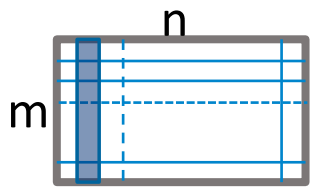
Implementation

We implemented the twiddle multiplication, transpose, splitting, and combine kernels using CUDA to utilize parallel hardware

The splitting factor is dynamically determined

$$\alpha\downarrow j = \max\top i \, |x\downarrow ij \, |$$

$$\beta\downarrow j = \max\top i \, |x \downarrow ij \, |$$



The multiplication is performed by calling cuBLAS API

Batch execution is supported by handling each input independently and in parallel

Implementation

The classical 4-step algorithm requires 3 matrix transpose at every level

We employ the transpose property to avoid 2 of them

$$(F \cdot X \uparrow T) \uparrow T = X \cdot F \uparrow T$$

Note that Fourier matrix is symmetric for N=4

We adapted the transpose and combine kernel, and change the order of operands in matrix multiplication

Experimental Results

The dynamic splitting method preserves high accuracy over a wide range of inputs

Data range	FP16 cuFFT	Dynamic Splitting
$[-10^{-9}, 10^{-9}]$	14.6843805313	0.0000568428
$[-10^{-6}, 10^{-6}]$	0.5265535712	0.0000029240
$[-10^{-3}, 10^{-3}]$	0.0126493834	0.0000029515
[-1.0, 1.0]	0.0126134995	0.0000029261
$[-10^{2}, 10^{2}]$	0.0125578260	0.0000029014
$[-10^{4}, 10^{4}]$	N/A	0.0000028950
$[-10^{10}, 10^{10}]$	N/A	0.0000030171

Table1. Relative error of half-precision cuFFT and our implementation at different input data ranges.

Experimental Results

Compared with matrix multiplication, the time spent on splitting and combine is not significant.

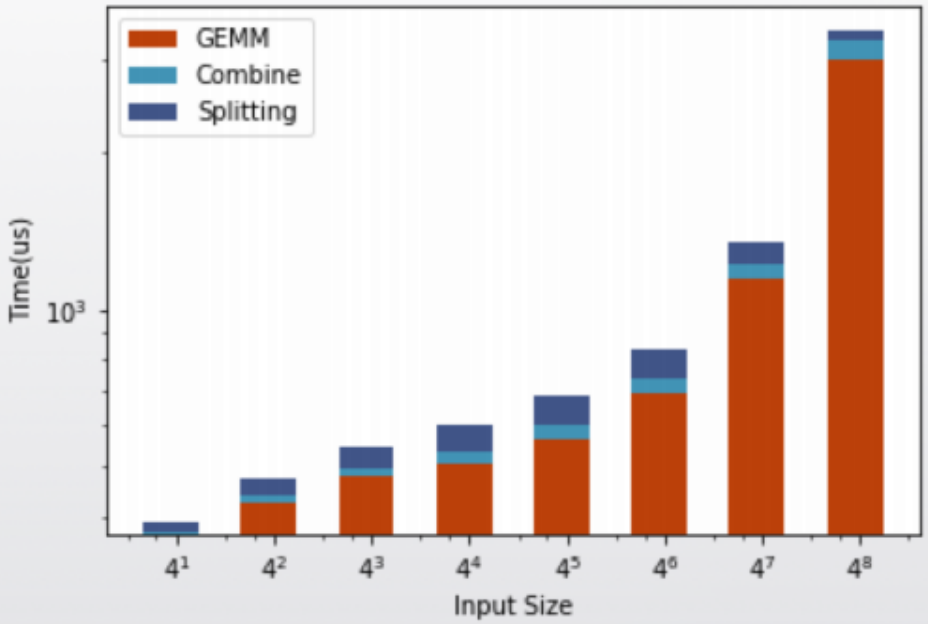


Figure1. Execution time breakdown at different input sizes. The matrix multiplication (by calling cublasGemmStridedBatchedEx) consumes around 90% of total time.

Performance Analysis

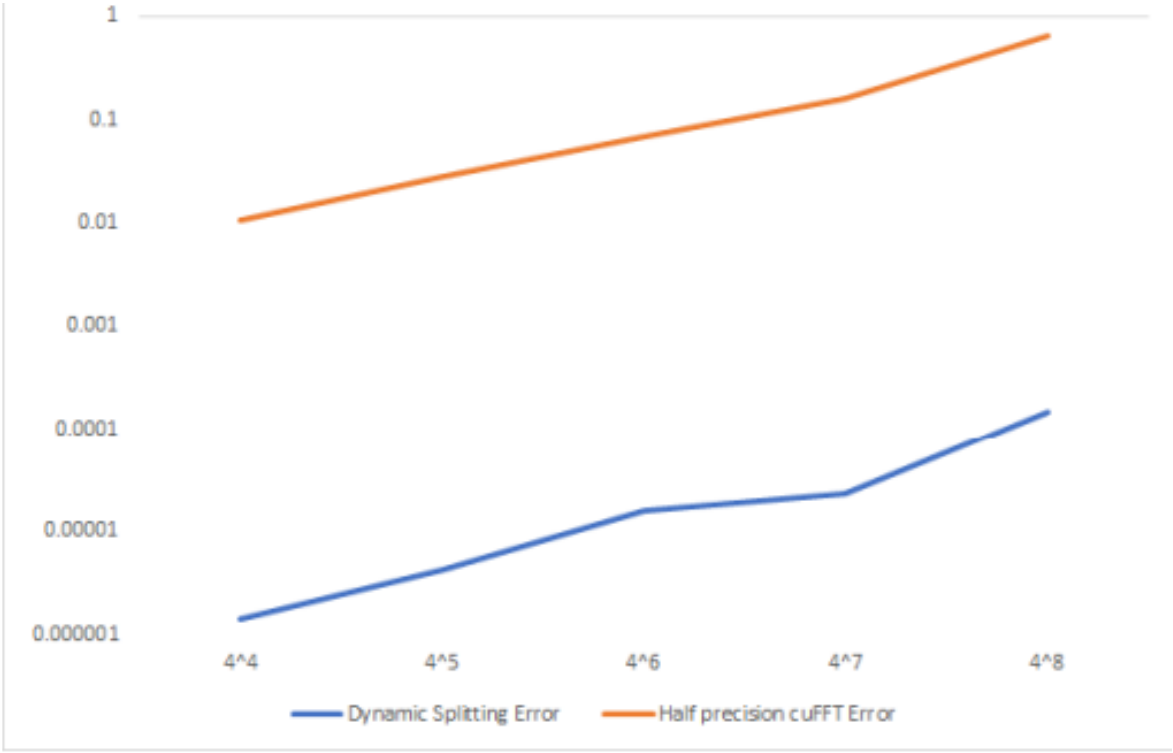
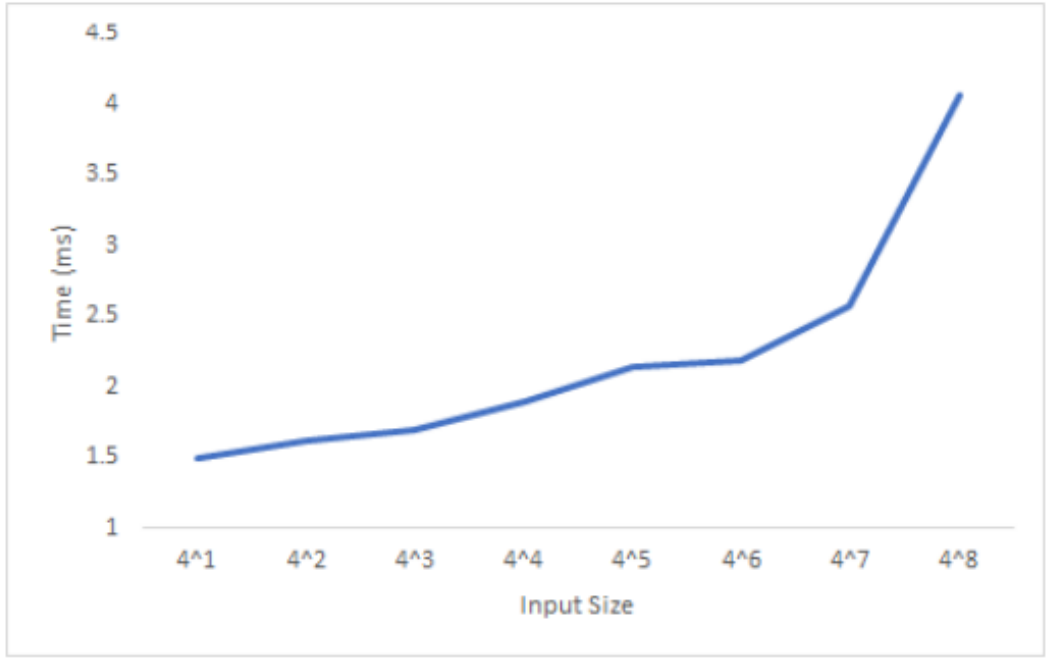


Fig. 1. Accuracy of half-precision cuFFT and our implementation.

Performance Analysis



Average Time for Mixed Precision FFT	Average Time for 32-bit FFT
2.788934 ms	6.334454 ms

Fig 3. Average execution time of FFT with growing input sizes

Ongoing work

Currently we are trying to overcome the glitch that only allows input sizes less than 64 kilobits. We expect to see an increased speed of execution with larger input sizes

Using the 3M Method and other techniques for further optimization

Input-aware auto-tuning splitting algorithm for ill-conditioned inputs may further improve execution speed and accuracy

Conclusions

Dynamic splitting method performs matrix multiplication in half precision.

Utilizes the tensor cores and efficiently computes fast Fourier transform.

Effectively emulates single precision calculation, and produces highly accurate results from a variety of inputs.

Acknowledgements

This research project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville.

This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.

This material is based upon work supported by the U.S. DOE, Office of Science, BES, ASCR, SciDAC program. This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05- 00OR22725.

References

[1] B.-Y. T. Shing-Tai Pan, Chih-Chin Lai, “The implementation of speech recognition systems on FPGA-based embedded systems with SOC architecture,” IJICIC, vol. 7, no. 10, 2011.

[2] M. F. H. Buijs, A. Pomerleau, “Implementation of a fast Fourier transform (FFT) for image processing applications,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 22, pp. 420–424, 1974.

[3] J. Kong and S. Yu, “Fourier transform infrared spectroscopic analysis of protein secondary structures,” Acta biochimica et biophysica Sinica, vol. 39, no. 8, pp. 549–559, 2007.

[4] X. L. Shuo Chen, “A hybrid GPU/CPU FFT library for large FFT problems,” IEEE 32nd International Performance Computing and Communications Conference, 2014.

[5] S. C. Stefano Markidis, I. P. Erwin Laure, and J. Vetter, “NVIDIA tensor core programmability, performance and precision,” Eighth International Workshop on Accelerators and Hybrid Exascale Systems, 2018.

References

[6] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek, and J. Kurzak, “Mixed precision iterative refinement techniques for the solution of dense linear systems,” The International Journal of High Performance Computing Applications, vol. 21, no. 4, pp. 457–466, 2007.

[7] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov, “Accelerating scientific computations with mixed precision algorithms,” Computer Physics Communications, vol. 180, no. 12, pp. 2526–2533, 2009.

[8] S. Le Grand, A. W. Gotz, and R. C. Walker, “SPFP: Speed without ¨ compromisea mixed precision model for GPU accelerated molecular dynamics simulations,” Computer Physics Communications, vol. 184, no. 2, pp. 374–380, 2013.

[9] W. M. Gentleman and G. Sande, “Fast fourier transforms: for fun and profit,” in Proceedings of the November 7-10, 1966, fall joint computer conference. ACM, 1966, pp. 563–578.

References

[10] D. H. Bailey, “FFTs in external or hierarchical memory,” The Journal Of Supercomputing, vol. 4, p. 2335, 1989.

[11] (2018, Jul.) cublas batched gemm throw not supported error with large batch size. [Online]. Available: https://stackoverflow.com/questions/51500189/cublas-batchedgemm-throw-not-supported-error-with-large-batch-size