Accelerating FFT with half-precision floating point hardware on GPU
Anumeena Sorna (NITT) & Xiaohe Cheng (HKUST)
Mentor: Eduardo D’Azevedo (ORNL) & Kwai Wong (UTK)

Abstract
We present a fast and accurate parallel algorithm for computing the Fast Fourier Transform on the Volta Graphical Processing Unit. We focus on utilizing the speedup due to using half precision multiplications capability of the tensor core hardware without degrading on the precision of the Fourier Transform result. This is done by splitting the input single precision set data into 2 half precision set and recombining at a later step. This Fast Fourier Transform algorithm is widely used in material science applications and we hope to further optimize the algorithm for the domain specific computational needs.

Discrete Fourier Transform (DFT)
The DFT converts time domain signals to frequency domain signals according to the equation:

\[ X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi ink/N} \]

Applications of Fourier transform:
• Speech Processing (MP3)
• Image Processing (JPEG)
• Filtering Algorithms
• Solving Difference Equations
• Fast polynomial Multiplication
• Material Science Domain

The Fast Fourier Transform (FFT)
The FFT would require many computations for a large input sequence of length N. In order to simplify computation the FFT algorithm was developed. The FFT reduces the number of computations needed for N points from O(2N^2) [DFT] to O(2N*\log_2(N)) [FFT].

FTT Algorithm
1. Factor N=n1*n2
2. Take FFT of length n2 (n1 times)
3. Multiply by e^{-2\pi ink/N}
4. Take FFT of length n1 (n2 times)
5. Reorder

Input Sequence Norm Maximum Error
1.0 (Range: [-1,1]) 2.3839121e-01
1000 (Range: [-1000,1000]) 6.1035200e-05

Future Work
• Batch Data Set FFTs
• Efficient Memory allocation to minimize data transmission between host (CPU) and device (GPU)
• Develop function for in place transpose
• Reduce number of transposes required
• Expand for 2 Dimensional FFTs
• Optimize the FFTs for material science applications

Acknowledgements
This 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.