GTU Department of Computer Engineering
CSE 344 - Spring 2022
Homework #5 Report

Hasan Mutlu
1801042673

HOW I SOLVED THIS PROBLEM

I have solved this problem by making use of mutexes and condition variables. The main process and the threads worked as described in the PDF. The 2D Discrete Fourier Transform was complicated, I have implemented it as I researched from the internet.

DESGIN EXPLANATIONS

- ➔ In order to synchronize the main process and the threads, I have used 1 mutex and 2 condition variables. All of them are initialized statically, with the macro.
- ➔ The first condition variable is used for main process to wait for threads to finish calculating AxB matrix. The threads increased a counter and signaled to main process. The main process checked if the counter reached to M value.
- ➔ The second condition variable is used for threads to wait to start DFT calculations. After all threads finished the calculation of AxB matrix, the main process raised a flag and broadcasted a signal to the waiting threads.
- ➔ The main process waited for the DFT calculations to finish by using pthread_join.

- ➔ If 2^n is not divisible by M, M is rounded down to the previous divisor of 2^n. For example, if 2^n is 32 and M is 24, it is rounded down to 16. Then, 16 threads are created instead of 24.
- ➔ Simply rounding down the result of 2^n/M was not working because in this way, some columns were left uncalculated. In order to give every thread equal work, I decided to round down the number of M.

ACCELERATION REPORT

My CPU has 8 cores. As it is supposed to be, the performance difference between 2, 4 and 8 threads was very obvious. For thread numbers higher than 8, the performance difference was very minimal.

ACHIEVED AND FAILED REQUIREMENTS

As far as I tested, all requirements are achieved.