Basic Local Alignment Search Tool (BLASTn) for Hardware Acceleration

Alden Param, Alex Chan, Hmayak Apetyan, Jacob London, Simon Tutak, Siva Prabakar, Dr. Aly, Dr. Ellabaan
California State Polytechnic University, Pomona Department of Electrical and Computer Engineering

Introduction

Since the completion of the first draft of the human genome in 2001, genomic data has been far outpacing Moore's law and is expected to reach the Exabyte scale with the next year and surpass even YouTube and Twitter by 2025 [1]. However, even with the advent of third generation sequencing technologies, it is difficult to process the overwhelming amount of genomic data. These technologies have prohibitively high computational costs; over 1,300 CPU hours are required to analyze the data using a reference, and over 15,600 CPU hours are required to assemble the reads de novo (or without reference) [1]. One of the popular methodologies created to speed up this process is BLAST, or Basic Local Alignment Search Tool. For DNA databases and DNA queries, BLASTn is used. Even with processes such as BLASTn, it can take a significant amount of time to process all of the database. Another issue with processes like BLASTn is that it can be inaccurate, as it is a heuristic algorithm and thus takes shortcuts to lower the processing time—which can result in a slightly less accurate analysis. This cost in accuracy is repaid through the speed at which BLASTn runs compared to a pure Smith-Waterman analysis of an entire database.

Method

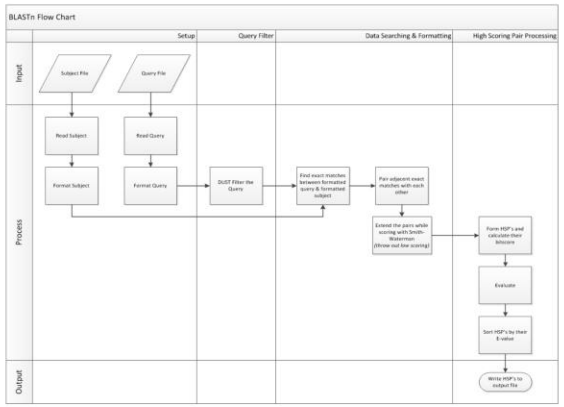

Fig. 1. The BLASTn flowchart process.

Evaluation

With all of our test data, the Python took around 35 minutes to run. With this prototype, we moved onto the C++ implementation. The first version of our C++ implementation took approximately 31 minutes to fully execute. After resolving additional inefficiencies in our code, running the program with a smaller dataset took approximately 1 minute and 5 seconds (total duration 1:05). Using C++ compiler optimization settings, the runtime was reduced to 8.6 seconds with the small dataset and 8 minutes and 30 seconds (total duration 8:30) with the large dataset.

Conclusion

Setting out at the beginning of the project we planned on executing it through three stages. First we would implement the algorithm in Python to help us understand the inner workings of the algorithm. The second stage was to implement the algorithm in a lower level programming language, we did this in C++. The third stage consisted of implementing the slowest to run portions of the algorithm into hardware, this turned out to be the smith waterman portion of BLASTn. We implemented the Smith-Waterman algorithm onto a Nexys 4 DDR FPGA.

Acknowledgements

1. Dr. Mohamed Aly for the assistance in both consultancy and reference for the development of this project.
2. Dr. Ellabaan for assistance through provided data and feedback regarding algorithm development and progress.
3. The ECE department at California Polytechnic State University, Pomona.