# John Peterson & Yashwardhan Singh

# Parmesh Ramanathan

# DATE \@ "MMMM d, y" May 14, 2014

BLASTn Word Matching

Developing a Solution for Pico FPGA Hardware

The Problem

The BLAST software provides heuristic searching for proteins and nucleotides. In its time, this concept was a breakthrough for genetic research. Scientists could use the program to find similar strings of DNA which play an important role in the identifying of genetic patterns between organisms. The BLAST algorithm is split into three portions: word matching, ungapped extension, and gapped extension. The different portions of the algorithm and the consecutive nature of processing a query using BLAST can lead to a large response time for large queries/databases. This team explored documentation for various other projects, including Mercury BLAST, TreeBLAST, and the Mitrion-C Project to determine the best way to adapt portions of BLAST to run on the Pico Computing FPGA PCI interface. The initial goal of this team’s project was to obtain a 100x speedup on the BLASTn architecture using the FPGA hardware. A few roadblocks were encountered along the way, partially due to the team’s lack of familiarity with the BLAST software and partially due to the struggles of designing an effective FPGA architecture for the desired functionality.

The Architecture

After running profiling on the BLASTn software using gprof, the team split into two pairs; one focused on the dynamic programming portion, and this half of the team focused on word matching. Specifically, this team focused on implementing the s\_BlastNaScanSubject\_8\_4 module to process word matching of word length 8 with a stride of 4 bases. With a thorough testbench, this module took 34.66% of the processing time of the entire BLAST application. In addition, the ScanSubject module is similar in functionality to other modules within the word matching portion of the BLAST software, such as s\_BlastSmallNaScanSubject\_8\_4. Implementing this function in hardware, speeding it up serially, and parallelizing it will likely be able to provide a significant optimization to the BLAST software.

The architecture chosen is based upon the Pico stream api. The software module takes pointers to the query and the database, both in compressed format (two bits per base pair), and other parameters such as range to scan and maximum number of hits to accept. It then writes the query to memory of the FPGA using the PicoBus. Then, the software opens a stream to send the database, in compressed format, to the hardware. Simultaneously, the software uses the ReadStream api to wait for results to be sent back. Sending and receiving data at the same time increases throughput of the module, and is achieved through creating separate software threads and Pico streams. When one of the desired parameters such as scan range or max hits has been reached, the module returns. This modified module is intended to mimic the original module, returning the number of hits found and an array filled with the results: offset of each hit with regards to both the query and database.

Meanwhile, the hardware increments through the received database stream using buffers to prevent data loss, and compares each the database to the query at each index of each. Every index of the query is compared simultaneously, and the database is streamed through one base at a time. When a hit is found, the hardware increments a counter it keeps of the hits, and pushes the index of the query and database to be written out via stream. The software periodically reads this counter via the PicoBus, compares the number of hits to its own value, and if more hits were generated, reads the appropriate number of results from the FPGA stream.

Currently, the hardware module has one comparator instance and streams in the database from start to finish. However, it would be possible and likely beneficial to implement multiple comparators inside the FPGA and stream separate portions of the database through simultaneously. On the hardware side, this would involve replicating the comparator logic. And on the software side, it would involve determining an efficient way to split the database and read back multiple result streams concurrently. Additionally, the possibility of extending to different word sizes would take some adjustments in both hardware and software. If the software were able to modify the comparator logic, such as writing a word length variable to the PicoBus and setting the comparators to match that length, this module could replace more word matching functions inside BLAST.

Challenges Faced

As mentioned above, the team struggled to fully understand the implementation of BLAST and how to integrate other successful projects’ approaches into the design of a Pico Hardware BLASTn system. Until the suggestion was given by Corey Olson to run detailed profiling on the software and to approach the problem from a numerical runtime standpoint, the team lacked an effective approach. Another challenge this team faced involved integration with the BLAST software suite. As it stands, the word matching functionality is not currently running inside of the BLAST source code. The complexity and obfuscation of the BLAST Makefile system, combined with the challenges of integrating Pico Hardware functionality posed significant problems for integrating and testing with the original (but slightly modified) software. As such, the word matching has only been run and tested manually. If the team had more time, it would be beneficial to take the C APIs created by Corey Olson and attempt to integrate the software more in depth. This pair also faced challenges implementing the architecture chosen and the challenges of combining software and hardware. The architecture chosen for word matching was modified at least twice due to concerns of both synthesis errors and ineffective hardware processing.

Three main priorities remain for the future of this project:

* 1. Integrate and run inside the BLAST C/C++ software
  2. Parallelize the database processing inside the FPGA and increase throughput
  3. Expand to cover additional word matching functions in BLAST

Additional future initiatives would likely include weeding out the process that generates a lookout table of the database, as that structure is only used by the software word matching and not needed for this implementation on the FPGA.

The challenges this team faced are not intended to take away from the potential that the BLAST suite has for optimization, however. The accomplishments of this group may lay the groundwork for a successful implementation and speedup on Pico Computing FPGA hardware in the near future.