1-1-2013

Accelerating Short Read Mapping Using A DSP Based Coprocessor

Shaun I. Gause
University of South Carolina

ACCELERATING SHORT READ MAPPING USING A DSP BASED COPROCESSOR

by

Shaun Gause

Bachelor of Science
University of South Carolina, 2007

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2013

Accepted by:

Jason Bakos, Major Professor

Manton Matthews, Committee Member

John Rose, Committee Member

Lacy Ford, Vice Provost and Dean of Graduate Studies

ABSTRACT

Advances in next generation sequencing technologies have allowed short reads to be generated at an increasing rate, shifting the bottleneck of the sequencing process to the short read mapping computations. High costs and extended processing times drive researchers to pursue more efficient solutions with an overall goal of a short read mapping architecture capable of processing short reads as they are generated. Digital signal processors have shown high performance capabilities while maintaining low power consumption in a wide field of applications. This thesis explores the use of a DSP accelerated exact match short read mapping algorithm, focusing on a performance metric to increase the number of mapped bases per watt-second. The design is implemented and tested for CPU and alternate coprocessor implementation comparisons to analyze the potential benefit of accelerating a memory bound application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

BDP ..... Big Data Problem

BWT ..... Burrows-Wheeler Transform

CPU ..... Central Processing Unit

DARPA ..... Defense Advanced Research Projects Agency

DDR ..... Double Data Rate

DNA ..... Deoxyribonucleic acid

DOD ..... Department of Defense

DOE ..... Department of Energy

DSP ..... Digital Signal Processor

FFT ..... Fast Fourier Transform

FPGA ..... Field Programmable Gate Array

GFLOPs ..... Giga Floating Point Operations Per Second

GMACs ..... Giga Multiply-Accumulate Calculations Per Second

GPU ..... Graphics Processing Unit

HGP ..... Human Genome Project

JTAG ..... Joint Test Action Group

NGS ..... Next Generation Sequencer

SIMD ..... Single Instruction Multiple Data

VLIW ..... Very Large Instruction Word

CHAPTER 1

INTRODUCTION

Deoxyribonucleic acid is the name of the chemical compound necessary for all life. Understanding the interactions of chemical combinations that make up DNA has led to the development of the field of genomics. The ability to view DNA at the most basic level has allowed for advancement in a multitude of applications from Microbiology to Pharmacology. To make viewing individual strands of DNA possible, current technological limitations to read each short sequence results in specialized equipment breaking the genome into small fragments. These short reads are then mapped using a reference genome to determine the original location of the read, allowing the reads to be assembled into a fully sequenced whole genome. Presently the most challenging and computationally intensive portion of the process is short read mapping.

As improvements are made in the next generation of sequencing equipment, faster and more reliable short reads are being generated at an increasing rate. Illumina, one of the most widely used and accepted manufacturer of next generation sequencers, is capable of producing 1.2 billion reads consuming 120 Gb of data in a 27 hour timespan for a whole genome and 6 billion reads consuming 600 Gb of data over an 11 day timespan in a high output mode for multiple genomes [1]. With such a high throughput, advances are needed in the short read mapping methods in order to handle the large volumes of data being produced.

Coprocessors have been widely used to increase performance in CPU based computations for various applications. There has been a mainstream push for FPGAs and now GPUs to be used in heterogeneous computing, but both are difficult to operate, complex to design, and are power hungry. They provide benefits only for specific application's kernels.

This thesis presents an alternative to traditionally-used coprocessors to examine how well a data intensive application of short read mapping maps to a DSP coprocessor technology. The basis for short read mapping is built upon the Burrows-Wheeler Transform (BWT) [2]. Originally designed for data compression, the BWT reorganizes a block of text placing similar characters together. While optimized for data compression, it was additionally determined to be an effective solution for searching strings as well. The Bowtie project [3] developed a short read mapping algorithm based upon the BWT to increase performance in searching for short reads in a large reference genome. Analyzing the repetitive computations performed in Bowtie, an algorithm was developed for a GPU implementation improving the exact match short read mapping by the use of table based searches [4].

The goal of this work is to examine a DSP implementation of the exact match tabling search using the human chromosome 22 as a reference with the focus of improving the mapped bases per watt-second, maximizing both throughput and power consumption. A DSP implementation will be developed and tested with a CPU software implementation focusing on improvements through calculation reduction in the algorithm, parallelism for data distribution, and specialized enhancements for the DSP architecture for increased hardware acceleration performance. A power analyzer will

measure the power consumption for the performance metric calculations. Other coprocessor comparisons will be possible through other studies' listed hardware to extrapolate their power consumption compared to the performances given for an equivalent to the performance metric developed in this thesis.

To introduce this thesis, the next chapter explains the motivation for this work. Next the general background information is presented explaining information on understanding DNA, the big data problem, the Burrows Wheeler Transform, Bowtie, tabling improvements on the exact match search and the use of DSPs as coprocessors. Subsequently this thesis will present related works for coprocessor implementations, the hardware used for testing, the DSP and CPU designs, test cases, the performance metric developed, the results obtained, and a conclusion of the data and performance.

CHAPTER 2

MOTIVATION

The motivation for accelerating short read mapping comes from improvements in the NGS technology allowing for more accurate and a larger number of short reads being produced in less time. These improvements have shifted the bottleneck from sequencing to the mapping algorithm. More adequate solutions are needed for short read mapping to handle the increasing amount of data produced. This chapter discusses the reasoning for choosing this particular area of research and the importance the topic plays in numerous fields of study.

The completion of the Human Genome Project (HGP) in 2003 led to the first informational database created that held the 3 billion chemical base pairs which form the human genome [5]. With the conclusion of the HGP came the possibility to use short read mapping for DNA sequences to map to a human genome reference. Since then, there has been a drive for improvements in the technology used to sequence DNA as well as the computations for processing the enormous amounts of data produced by sequencing. Various methods have been introduced including de novo and reference based sequencing.

De novo assemblies build genomes with no aid from a reference. They take a considerably greater amount of time to execute when compared to reference based assemblies. This method provides a key step in the initial sequencing of a new genome, but is less beneficial for subsequent sequencing of other genomes in the same species.

The most common method of reference based sequencing, short read mapping, has been used due to the availability of human genome references publicly available.

An exact short read mapping algorithm was selected because of the inherent pipeline design that exists in the NGS workflow. Recent studies have shown that short read mapping has better performance by further splitting the mapping process into an exact matching stage followed by an approximate matching algorithm due to a majority (70 to 80%) of reads mapping exactly to the reference genome [6] [7]. The amount of data generated is memory bound, dependent upon cache performance with data dependencies from aborted reads in cases where a read is determined to be a mismatch.

Current methods for processing short reads involve a variety of central processing unit (CPU) and coprocessor based technologies with field programmable gate arrays (FPGAs) and graphical processing units (GPUs). Software implementations with CPUs, such as with Bowtie [3] and WHAM [8] have been the standard but face long runtimes and heavy power consumption. While the use of FPGAs [7] [9] [10] and GPUs [11] [4] [12] as coprocessors for short read mapping has shown speed improvements, these technologies still suffer from individualized limitations that hinder their adoption.

With the increase in popularity of coprocessors for high performance computing, digital signal processors have expanded into high performance applications currently in fields of medical imaging, mission critical applications, testing, and automation [13]. DSPs are, by design, power efficient while maintaining strong computational performance. One of the leading manufacturers of DSPs, Texas Instrument, provides an 8 core DSP package that achieves 1.25 GHz, 320 GMACs, and 160 GFLOPs for each core, while consuming only 10.5 watts [13]. DSP architecture offers more on chip memory for

higher throughput and more localized RAM for improved access times and better performance with memory bound applications, which would offer improved performance per watt over the same time frame when compared to CPU and other coprocessor implementations. Through the same parallel processing techniques used for high performance computing of CPUs, FPGAs and GPUs, examining the use of DSPs may lead to significant performance results at a fraction of the power consumption of other processing technologies.

DSPs were chosen because of their inherent low power use coupled with a powerful processing capability. With a specific memory bound algorithm for short read mapping, this thesis examines DSP's ability to enhance a pipeline that is capable of analyzing data from NGS equipment at the same rate the data is being produced. This thesis does not focus on overall completion time as many other studies have done, but instead focuses on a comparison of the number of bases read with respect to the number of watt-seconds consumed.

CHAPTER 3

BACKGROUND

3.1 UNDERSTANDING DNA

DNA is the building block that provides a set of instructions for all living organisms to function. DNA is made up of four chemical bases called nucleotides: Adenine, Cytosine, Guanine and Thymine. Bases combine in a specific order, pairing Adenine with Thymine and Cytosine with Guanine. Organisms within the same species will have a majority of the DNA identical to one another while differing species' DNA can differ drastically. Knowing which nucleotides match to form a base pair allows for only one base of the pair to be considered when dealing with computational genomics. Grouping base pairs into a double helix strand form individual genes. Thousands of these genes are linked in sequence to form a single chromosome. The human genome consists of twenty three chromosomes. Figure 3.1 shows a visual example of the double helix DNA strand and the structures that are formed.

In order to view these strands, sequencing technology was developed which uses certain chemical components to break apart an organism's DNA into small fragments. The smaller fragments can then be closely examined to generate the exact ordering of the bases into short reads. These short reads frequently contain variances which increase the complexity of mapping based on the occurrences of gaps, insertions, deletions and mutations.

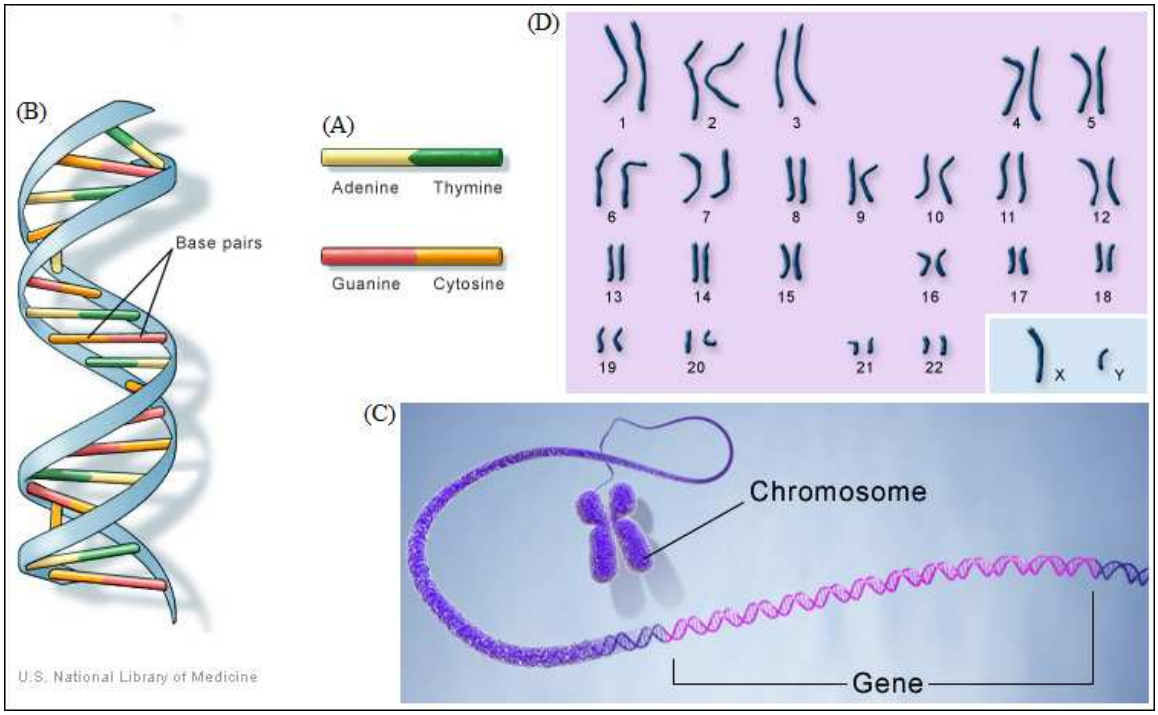

Figure 3.1 Composition of DNA. (A) Four chemical nucleotides combined as base pairs. (B) Base pairs in a double helix strand. (C) DNA strands forming genes which form chromosomes. (D) The human genome consists of 23 chromosomes [14].

Gaps are missing bases that have not been read. It is the region between the end of one short read and the beginning of another that no other short read covers. Gaps occur because of deficiencies of the NGS technology and are found during alignment after reads have been mapped. The only way to fill these gaps are to re-sequence, generating more short reads that will cover the missing regions and provide a more complete genome sequence. Insertions and deletions occur when a base is inserted or deleted in a sequence. Insertions and deletions can occur naturally in the organism or as a result of the NGS technology. Mutations also occur naturally or as a result of NGS technology. Mutations occur when a base is changed from one nucleotide to another. Naturally occurring variances in DNA can be caused during cell's replication of DNA or variances that exist between two organisms of the same species that define hair color, eye color, or predisposition to a particular ailment.

Knowledge of the organization and function of different genes allows scientists to understand the basic functionality of humans and the reasoning behind health defects and medical problems, among others. This drives the medical community and is a strong support for increased funding and effort spent furthering the field of genomics.

3.2 The Big Data Problem

Big data has been a concept that has been around as long as computational algorithms. Technological improvements are the cause for the rapid data expansion that must be planned and accounted for in future developments. The Big Data Problem (BDP) encompasses the issues associated with the technological bounds to compute large volumes of data, the limits of data storage, and the capability to transport data between systems. These three issues deal with the need to efficiently handle the increasing amounts of data generated as well as the data already available.

Recognizing the importance of big data, funding is readily available for research and development. The US DOD, DOE and DARPA announced in March of 2012 an initiative to invest more than $200 million leading to advances with the big data problem [15]. For genomics, the BDP is primarily that of a computationally limited system. There is far more data being produced than is effectively processed with current algorithms and technology. An example given indicates that for a 1000 petabyte chunk of data, 100 instructions to process one block on a 5 GHz processor would take 635 years to complete [15]. This push is a motivating factor for the amount of research being done to find effective solutions in the field of genomics, which is generating more data at a faster rate with improvements in NGS technology continuously being made.

3.3 BURROWS WHEELER TRANSFORM

The Burrows Wheeler Transform was first developed as an algorithm to be used in conjunction with a compression algorithm. The BWT reorganizes a block of text to make it easier for the compression algorithms to compress the information more effectively by placing similar characters together. The BWT is accomplished with three basic steps. Refer to figure 3.2 for an example of the transformation of string *S="ACACGTATTA"* of size *N=10* for the following steps.

First, an input string *S* of size *N* is selected and copied *N* times forming rows in a matrix. Second, as each row is copied, it is shifted cyclically. Each shift removes the first character of the block of text and places the character at the end. This results in *N* copies of the string *S* generated in a matrix grid of size *N x N* as shown after the first arrow in figure 3.1 (A). The third step of the transform sorts the matrix rows lexicographically where $A < B < \ldots < Y < Z$ as shown after the second arrow in figure 3.2 (A). The transform is taken from the last column of the resulting matrix [2].

The matrix that is generated during the shift and copy steps is known as a last-first mapping. This mapping property allows for the occurrences of a character in the last column to be referenced to the same order of the occurrences of that character in the first column. This principle is the technique that allows the BWT to be used as an efficient searching algorithm, which has been used in multiple bioinformatics applications. Figure 3.2 (B) shows an example of the BWT and a last-first mapping example used to identify the range of rows.

The example searches for the query “CGTA” in the original reference string “ACACGTATTA” using the first and last columns of the transformation matrix taken

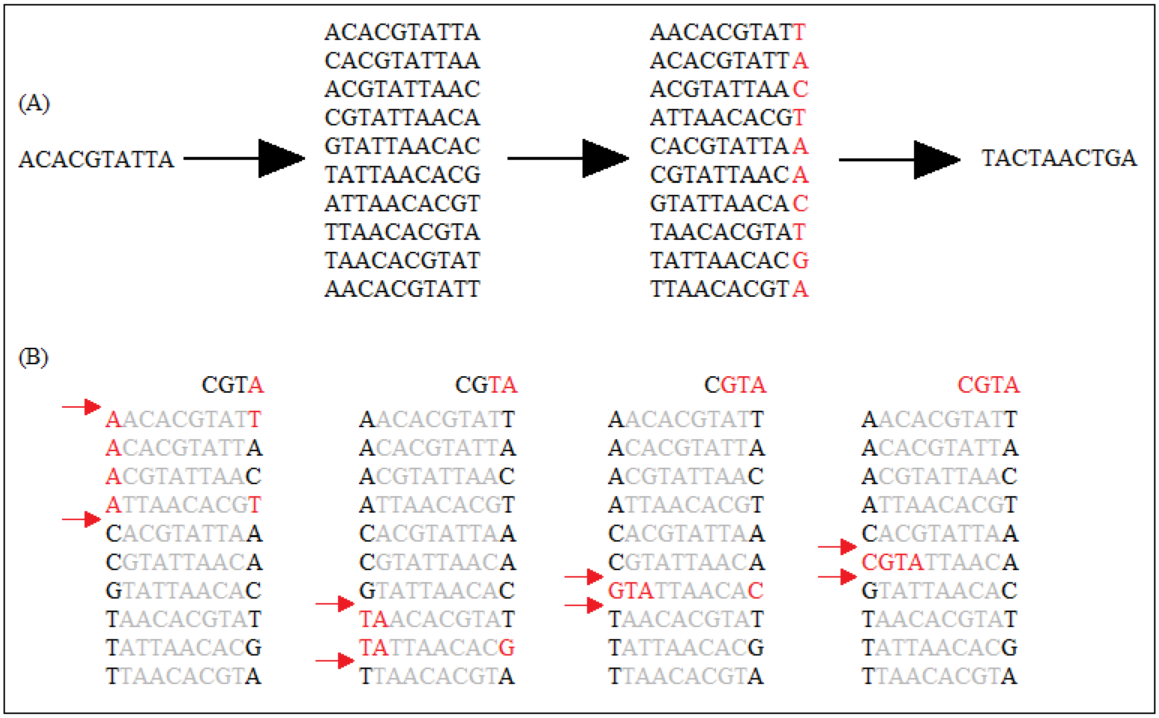

Figure 3.2 Burrows-Wheeler Transform. (A) The Burrows-Wheeler matrix and transformation of ‘ACACGTATTA’. (B) Steps taken to narrow the range of rows for an exact match of the query ‘CGTA’ in the reference string ‘ACACGTATTA’.

from figure 3.2 (A). The first column contains the character currently being searched and the last column contains the next character for comparison. In a last-first mapping, the first character is the end character of the query, iterating from right to left. The search begins with the last character, ‘A’, of the query “CGTA”. The initial range of rows is selected from the first column containing the character ‘A’, which are rows 0 to 3 in the example, giving a range of 4 possible matches. The last column within the given range is compared to the next character from the query, ‘T’. Within the same row range, the last column contains the first and second occurrences of ‘T’ from the transform out of three possibilities, reducing the possible matches to 2. The next step moves the range in the first column to the first and second ‘T’ in the list as shown in the second step of figure 3.2 (B). These steps are repeated for the remaining characters from the query ‘G’ and ‘C’ as

indicated in the remaining steps of figure 3.2 (B). Once all of the characters of the query have been iterated through, an exact match exists if the range is greater than 0. For the example listed only 1 match exists. The range indicates the total number of exact matches if the range is greater than 1. As BWT was primarily designed for data compression, the location of the query to the original reference is not considered in the BWT example and will be described later in this thesis.

The BWT takes advantage of the repetitive nature of the English language by grouping similar characters. The same principle can be used toward genomic applications due to the repetitive nature of DNA and the simplistic selection of only four possibilities of nucleotides. The use of BWT has become the basis for numerous short read mapping and alignment algorithms, most notably the Bowtie project [3].

3.4 BOWTIE

The Bowtie project was one of the first short read mapping and alignment algorithms capable of a notable speedup over previous hash table algorithms. Bowtie introduces the BWT to index the reference genome and searches the index to map the short reads achieving speedups of 60x for Maq [16] and 350x for SOAP [17].

Several improvements were made over the previous leading algorithms. The use of the BWT greatly reduced search time and memory footprint to allow for the process to run on a standard desktop computer platform. To aid in calculating the original positions of the transform matrix, a ‘$’ character is appended to the string prior to the shift and copy steps. While this addition slightly increases the size of the transform, the index is able to be calculated after the transformation allowing for faster index building. The size difference for the transform was only an additional character while the matrix increased

trivially to *2N+1* for a reference string *S* of size *N*. Additionally Bowtie includes approximate matching with backtracking to attempt to increase the number of reads mapped by iterating 2 levels back when no exact match is found. Parallelism is introduced as the increase in processing threads allows for a distribution of concurrent searches. The steps employed are a two-phase process. The first phase builds the reference index while the second phase matches short reads to the reference. Once built, the reference index does not have to be rebuilt unless a new reference sequence is needed. Figure 3.3 demonstrates the modifications to the original BWT.

In the first phase, the reference index is built by first appending a $ to the reference string as shown in the first step of figure 3.3 (A). The string is then shifted and copied before sorting lexicographically where *$ < A < C < G < T*, as is done on the BWT. The shift index is built based on the location of the '$' in each row of the transformation matrix. The shift index indicates the number of positions each row was shifted to the left and references the location in the original reference string. In the example shown in the last step of figure 3.3 (A), the original sequence row will have an index value of 0, while the row beginning with '$' will have an index value of 10.

In the second phase, only the first and last column of the transformation is needed. The exact match search reductions are the same as those performed in the BWT, but only using "A", "C", "G", "T", and "$" instead of a full alphabet. Once the search is completed in figure 3.3 (B) as was done in the BWT examples, a resulting match was located. The shift index position associated with the row from the query match in the last step in figure 3.3 (B) shows an index of 3, which indicates that the query can be located at the 3rd character in the original reference string, beginning with 0. The query is located

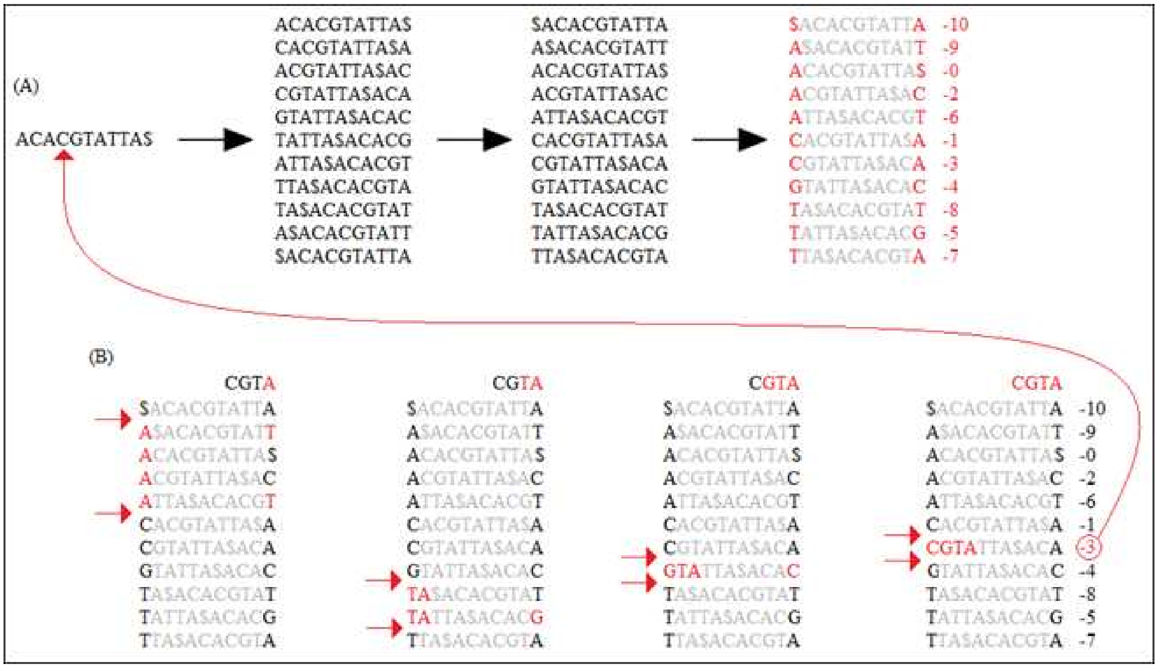

Figure 3.3 Bowtie algorithm of the Burrows-Wheeler Transform. (A) Indexing phase generates a first column, last column and a reference index. (B) Searching phase iteration reductions for the search query ‘CGTA’ of reference sequence ‘ACACGTATTA’.

in the original reference by the arrow from the last step (B) to the reference string (A) in figure 3.3.

To increase the percentage of short reads mapped, Bowtie uses an approximate matching backtracking strategy when a read is determined to be a mismatch. A mismatch occurs in the exact matching stage as each base of a short read is compared to the reference. Once a base does not match the base expected from the reference comparison, the read is identified as a mismatch. The mismatched reads are passed to the next stage for approximate matching. Bases found to be mismatched are replaced with matching bases to allow the algorithm to continue searching. If replacing one base does not find a match, then additional bases are replaced. This process of backtracking drastically increases the runtime for each number of mismatches allowed. If the number of mismatches in a short read is larger than what is allowed by the algorithm, the short read

is identified by the approximate matching stage as a mismatch. Short reads not mapped in either the exact match or approximate match are considered mismatches and lead to the overall percentage of reads not mapped by the algorithm. The standard backtracking level of Bowtie is 2 or 3 levels, which allows for 2 or 3 mismatches per short read. Bowtie is capable of running higher levels of backtracking, but causes exponential increases in runtime.

The algorithm developed led to performance mapping of 29.8 million 35 base reads per CPU hour run on a standard desktop workstation with 2.4 GHz Intel core 2 processor with 2GB ram and 33.8 million 35 base reads per CPU hour on a server with a 4 core 2.4 Ghz AMD Opteron processor with 32GB ram [3]. These performance results led to Bowtie being used as a mainstream benchmark for short read mapping for comparisons with later CPU based technologies as well as heterogeneous processing alternatives. The limitations of Bowtie are the computationally intensive reoccurring calculations necessary to search the reference index. Finding alternative methods to reduce the computational load are necessary for future improvements in processing time.

3.5 Exact Matching High Throughput Sequencing

As Bowtie grew in popularity for short read mapping, there has been an increase in the amount of research done to further increase the performance of the algorithm and find alternative architectures more suitable to the high throughput needs of the memory bound sort read mapping. Su Chen and Hai Jiang of Arkansas State University developed improvements based on Bowtie and the BWT with the focus of increasing performance to handle the increase demand and generation of short reads from NGS technology by the use of tables to reduce computational complexity, a more efficient exact matching block

sort improvement of Bowtie's algorithm and investigation in the use of a GPU coprocessor to further improvements over the standardized CPU technology [4].

An analysis of the Bowtie algorithm revealed three improvements to reduce the complexity of the computation to an *O(lg(n))* table based search. The first step proposed locates the top and bottom positions for the first column of the Bowtie-BWT matrix for each base. Second is to count the occurrences of each 'A', 'C', 'G' and 'T' above the current row for the last column of the matrix. Finally a sum of the occurrences of each 'A', 'C', 'G' and 'T' is calculated from the last column of the matrix. Figure 3.4 demonstrates the optimization tables generated from the steps listed above.

The calculations performed by the algorithm reduce to 6 memory stores, 5 memory lookups, 4 addition/subtractions, and 1 if comparison for a total of 16 operations for each character in a short read. Decreasing the number of calculations to simplified table lookups from the index tables converts the Bowtie-BWT algorithm from a computationally bound kernel to that of a memory bound kernel. Figure 3.5 gives the exact matching table based search algorithm.

The performance increases obtained with the modifications on a GPU base coprocessor achieve a 40x speedup over that of a CPU based solution. The NVIDIA Tesla C2050 GPU used for performance testing is one of the leading GPUs offered for high performance general purpose GPUs available. While the GPU offers 448 processing cores, a significant amount more than standard CPUs, the GPU is designed as a computationally powerful coprocessor achieving high performance with regard to double precision floating point operations capable of 515 Gflops using 247 watts of power for a single GPU in addition to the power requirements of the CPU needed for the remaining

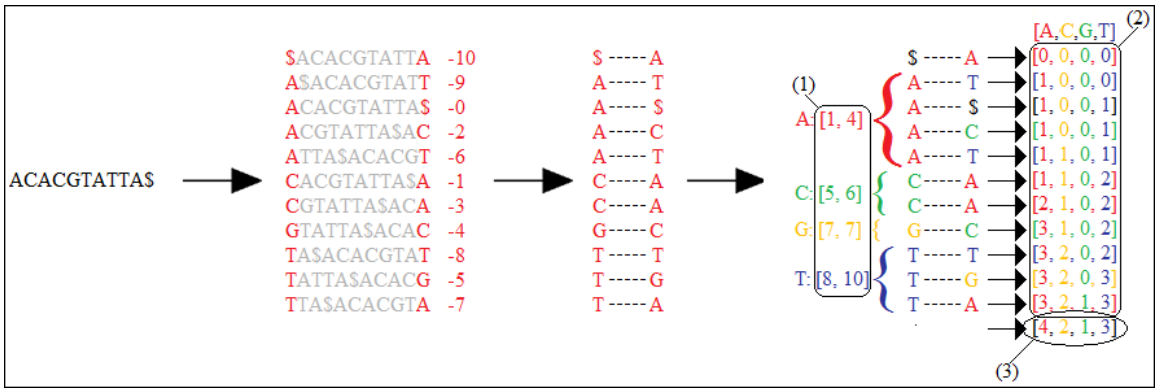

Figure 3.4 Bowtie-BWT Optimization Steps. (1) Find the top and bottom positions for 'A', 'C', 'G', and 'T' in the first column. (2) Count the occurrences of 'A', 'C', 'G', and 'T' above the indicated row in the last column. (3) Sum the occurrences of 'A', 'C', 'G', and 'T' in the last column [4].

```
Constants:
#define    target sequence                                                    as tarSeq
#define    target sequence length                                             as tarLen
#define    the table contains top positions for ATCG in the first column      as ATCG_top
#define    the table for bottom positions for ATCG in the last column         as ATCG_bottom
#define    the table contains the numbers of ATCG above each position in the last column   as upper_sum

Variables:
index      :        index for current character in the target sequence
ch         :        the character pointed by index in the target sequence
tmp        :        a temporary buffer saves the number of ATCG between the up and the bottom positions in the last column

Search () {
        index = tarLen – 1;
        ch = tarSeq[index]
        top = get_top(ATCG_top, ch)
        bottom = get_bottom(ATCG_bottom, ch)
        index = index – 1
        while (index>=0) {
                ch = target_seq[index]
                tmp = get_upper_sum(upper_sum, bottom+1, ch) – get_upper_sum(upper_sum, up, ch)
                if (tmp<=0) {
                        break;
                }
                top = get_top(ATCG_top, ch)
                bottom = get_bottom(ATCG_bottom, ch)
                bottom = top + tmp – 1
                index = index – 1
        }
}
```

Figure 3.5 Exact Matching Table Based Search Algorithm with complexity *O(lg(n))* [4].

operations [18]. The GPU is a powerful coprocessor for its application specific functions, but is not as robust in terms of memory access compared to data calculations, particularly

when considering the power consumption performance benefit of a memory bound application.

3.6 DSPs AS COPROCESSORS

Over the last several years research has expanded into various types of coprocessors for heterogeneous computing acceleration. Select coprocessors provide performance increases for a specific range of applications. Depending on the specific purpose for the acceleration, a coprocessor may be more or less suited for the need. Previous cell processors and FPGAs provided generous speedups for special purpose processing applications. Cell processors have now become outdated and FPGAs require knowledge of hardware design and are difficult to use. The most recent advance in coprocessing technology has been the use of general purpose GPUs for accelerating computationally intense applications with high levels of parallelism. While these processors provide a strong alternative to CPUs, they generally excel only in computationally bound kernels and consume considerable amount of power to the point of being a significant issue that must be addressed with their use. The average supercomputer consumes more than 3.2 megawatts, which translates to a multimillion dollar electric bill and a serious consideration for high performance computing [19].

Digital signal processors are an alternative coprocessor technology not widely explored. One of the world leaders in DSP manufacturing, Texas Instruments, offers numerous packages and has developed processors from ultra-low power DSPs capable of fixed point computation and FFT acceleration to their multicore DSP line that offers a range of processing capability up to 16 GHz in a single multicore package capable of 32 bit fixed and floating point acceleration [13]. With the initial design of the DPS, even the

highest performance processors use a fraction of the power of any other coprocessor technology while maintaining competitive high performance computing ability. The multicore high performance DSPs from Texas Instrument offer multiple DSP core packages with up to 8MB of on chip memory with enhanced memory architecture for faster on and off chip memory access to the available DDR memory bank integrated in the package. A TeraNet switch fabric allows components on the package to communicate through 2 terabits of bandwidth and independently operating high bandwidth peripheral input/output interfaces for simultaneous communication and memory transfer. The simultaneous and high bandwidth communication eliminates memory access bottlenecks for off package needs and allows for real time processing capabilities [13].

CHAPTER 4

RELATED WORK

In order to effectively examine the use of DSPs to accelerate short read mapping, it is important to look at related work done to accelerate sort read mapping through other coprocessors and technologies. With a more rounded view of alternative approaches, a better evaluation of the DSP coprocessor can be made for a potentially higher performance increase and lower power consumption of the previously explained metric, mapped bases per watt-second. Related work covered in this section will explore short read mapping applications with cloud computing, FPGA and GPU coprocessor acceleration and the results obtained from studies done on each.

4.1 CLOUD BASED SHORT READ MAPPING

Cloud based computing has been made recently available since the advent of cloud services. While the capabilities of cloud computing are almost limitless in computational capacity, there is a price associated with the performance gains, as it is a service rendered. The higher the parallelism and performance, the cost of maintenance, power consumption and technology drive up the cost of the service.

A cloud based short read mapping study was completed in 2012 to review the possibility of using these services specifically for accelerating short read mapping. The study introduces a concept of maintaining historical data on the cloud. This allows for mapping to begin the moment short reads start to upload since the reference genome and other needed information already exists on the server. The study ran performance tests

with a 9 server network running Xeon dual core 2.53 Ghz CPUs and 6 GB of memory with 36 base short read lengths [20]. While there were no specific results given, generalities that a single read was able to be done in less than a second on a non-congested system and congested systems were able to send a response in a several seconds indicates that the performance would not be able to provide results from a NGS machine as fast as the reads were generated. In addition a problem was discovered with latency issues with uploading and downloading data from the cloud. The time necessary for a genome's short reads to be uploaded is not listed, but would add additional difficulties which prove to be a substantial setback for this type of technology. Until further improvements are made, cloud based short read mapping would not be competitive with current mapping technologies.

4.2 FPGA Coprocessor Acceleration

Field Programmable Gate Arrays have been used for high performance computing with a great success at large performance speedups. The processors are capable of being reconfigured with a new design to meet multiple needs and are efficient at parallel tasks within the limitations of the hardware blocks available on a FPGA processor. Generally FPGAs are more complex to use, as a specific design must be mapped first before being used as a coprocessor. Often times this can be a daunting task that is quite difficult. Any modifications may cause the entire design to have to be modified. With these limitations, they are still capable of huge performance gains within a certain scope of applications. Several studies were done to explore their use within short read mapping applications.

A study published in 2010 performed hardware and software comparisons of various read sizes and was able to achieve a performance speedup between 2x and 4x

using a naive FPGA mapping method. A naive approach was chosen based on the limitations of FGPAs to handle large data structures. The resulting improvements listed indicate that as the size of the reads increases, the performance gain slows to match that of a CPU implementation. Test results were given with 16, 24 and 36 base read lengths and the speedup dropped approximately 1x speedup with each read size increase on an exact match design. The conclusion listed supports that exploring alternative coprocessor technologies is vital at the long term evolution of short read mapping in order to surpass the performance of CPU implementations to meet the processing needs of NGS technology. The FPGA implementations seem to be limited as larger read sizes require less parallelism because more processing blocks are consumed to process less reads [10].

Related studies over a two year period both evaluated the performance of short read mapping implementations using a majority of registers and lookup tables in their designs with comparisons based on the performance of a CPU implementation of Bowtie. A focus was also made as to the accuracy and percentage of reads mapped. Both studies claim that 100% mapping of reads can be done within the tolerance of the number of mismatches, as they both implement mismatch algorithms. The results obtained indicated that the time to completion of 100,000 and 500,000 reads was 3 minutes 44 seconds and 3 minutes 33 seconds, respectfully. Bowie's performance came in at 3 minutes 45 seconds and 3 minutes 26 seconds, but was only able to map a fraction of the short reads of the FPGA implementations. Results from both are clear. There were no performance time improvements in comparison to Bowtie. A general timing performance increase was obtained with the FPGA design, either through a better design or newer technology. The

FPGA implementation was also able to map a higher percentage with a mapping guarantee [21] [22].

Most recently in 2013 an implementation for FPGAs was developed to split the exact and approximate matching steps into a pipeline approach using a naive mapping compared to popular software tools. The study takes a new approach and performs three comparisons. The first comparison is based on a static FPGA design for an exact string matcher and a software design for the remaining reads with an approximate string matcher. The second uses a static FPGA design for both the exact and approximate string matchers. Finally a comparison is done using a reconfigurable approach in which a first run is completed with the FPGA for an exact string matcher, then the FPGA is reconfigured as an approximate string matcher and run on the remaining data still stored. Performance increases were achieved in the order presented of 2x, 150x and 516x when compared to a software Bowtie implementation [7].

Until recently the FPGA implementation has not been viewed to be a benefit for short read mapping applications. While limited to a specific range of capabilities, the FPGA has been capable of providing quite significant speedups, depending upon the application. The idea of splitting the exact matcher and approximate matcher, as presented in the last of the listed FPGA studies was a portion of the motivation for this thesis. With the significant performance gains, an indication that the speedup improvements were so high based on a reconfigured FPGA method brings to question whether the improvement was based on keeping the memory local to the FPGA. With that in mind, exploring the benefits of DSPs ability to handle memory bound applications may provide promising results.

4.3 GPU COPROCESSOR ACCELERATION

The most recent push for coprocessor technology to shift to the GPU for mainstream use caused many GPU implementations to be developed for short read mapping acceleration. A wide variety of designs for use with short read mapping were created, but many did not achieve performance gains and others only minor speedups less than 3x [11] [12] [23]. GPUs have extremely high power demands that limit their overall benefit, but the high level of parallelism has shown that speedups can be achieved.

An acceleration of the RMAP software implementation for short read mapping achieved speedups from 9x to 14x after developing a pipeline system to eliminate race conditions from simultaneous map updates from GPU threads. The comparison was based on a software version of RMAP with the GPU version of the same algorithm. One limiting factor was placing a limit on the size of the short reads to not exceed 64. While many of the current NGS technologies offer short reads of smaller sizes, this limitation will lead to a decrease in performance as read sizes increase. The limitation stated does not prohibit the GPU implementation from calculating the maps, but decreases the performance gains [24].

A large contributor to the motivation for this thesis comes from the next paper based on an exact matching approach with the use of the BWT on GPUs. Improvements of the exact matching method on a GPU compared to a CPU version achieved improvements of 40x to 45x speedups. A linear relationship between the numbers of short reads compared to the performance gain for an improvement of 1x speedup for every 4.3 million target sequences. An upper limit to the performance gain was not calculated, but shows that such gains are a motivating factor. Higher performance

achievements were claimed to have been limited when compared to a CPU implementation with respect to increased memory access. Because CPUs performance is higher with memory bound applications than GPUs, a DSP coprocessor implementation may be capable of even higher performance gains on a memory bound kernel, provided similar improvements can be made with the remaining portions of the exact matching approach [4].

CHAPTER 5

DSP DESIGN

Exploring the use of DSP base coprocessors for short read mapping offers the opportunity for significant performance gains. The goal of this thesis is to develop a DSP implementation of the exact match short read mapping algorithm to further improvements for the next stage of the genetic sequencing system. This goal will allow for further improvements with remaining stages to ultimately develop a system that is capable of sequencing a genome on demand. This chapter explains the hardware selected, the DSP and CPU implementation development, test cases used for results and the performance metric selected for comparisons.

5.1 ADVANTECH DSPC-8681 BOARD

A multi-core DSP board was selected to maximize the potential for both performance gains on a single DSP as well as parallelism performance enhancements that can only be gained by the use of multiple DSP chips. The Advantech DSPC-8681 board contains 4 Texas Instruments TMS320C6678 DSPs running at 1GHz per DSP. Each of the DSPs provides 8 processing cores, giving a total of 32 available cores. The board receives power through a PCI Express connection and consumes a maximum of 54 watts. For control of the DSP framework, a JTAG emulator is used for communication to the device for both programming and debugging [25].

Software development of the binary loaded to the DSP system was compiled on Code Composer Studio 5.1 developed by Texas Instruments on an Eclipse development

studio framework. This software package allows for code to be developed in a simulation environment before being applied directly to the hardware. Once the binary file has been compiled from C code, the targeted cores from the DSP system are then connected and the binary file loaded. For any memory needs, such as loading the index data and short reads into memory, a memory browser is used to load directly to the memory architectures available. The selected cores may then be executed to run the loaded binary instructions. In a production environment, the short reads would be able to be transferred through one of the DSP's many high throughput communication modules available. This system of streaming short reads from the NGS frees up additional memory from the DSP.

Unlike alternate coprocessors, the design of the DSP allows for an implementation to entirely run on a self-contained board. The Advantech board used in this thesis requires that power be supplied over a PCI Express connection provided by a computer, but no other functionality is used from that connection. Ideally, testing would be completed with an alternate power source and the board would operate outside of a computer environment, but precise power measurements are still able to be obtained for testing purposes of this thesis.

5.2 DSP DEVELOPMENT

The algorithm developed for the DSP design is based upon the exact match table based search of the GPU implementation previously described [4]. Several aspects were investigated in order to develop further improvements over the algorithm's performance including caching DDR memory into level 1 and level 2 on-chip cache, exploring the result of larger short read lengths, varying sizes of the reference index, and compiler optimizations for parallel processing of 8-way VLIW instructions and SIMD instructions.

With the C6678 DSP containing 32KB of L1 cache and 512 KB of L2 cache, investigating the performance of enabling caching to both levels of on-chip memory provided a significant benefit to overall performance. Enabling caching resulted in between 2x and 2.5x speedup in the amount of time necessary to process a block of short reads. Caching the data for reduced off-chip memory access outweighed the cost of cache misses for index lookups. Significantly increasing performance, caching was enabled for use with the algorithm on the DSP system and used when calculating the overall results.

Short read lengths of 35 bases were chosen as a more widely used standard of comparison between other algorithms. Even as short read sizes are expanding with improving technology, a more accurate comparison can be made by keeping the similarity of the same sized reads for the purposes of algorithm and processing comparisons. More accurate reads are still being computed from current NGS through the use of shorter reads, because longer read lengths often cause the bases farther from the focus of the NGS to be less accurate. To better understand the performance of the algorithm explored by this thesis, the effect of longer short reads was investigated to provide an overall view of the performance in relation to other algorithms, but not used in the result calculations.

A linear relationship was discovered between the size of the short read and the time necessary to process the data. Increasing the short read to 70 bases would require 2x the time to process when compared to a 35 base short read. No modification of the design was needed for this preliminary test. The design of the algorithm for both CPU and DSP implementations accepts any number of short reads and does not require reprogramming or redesign to change the expected read size.

The size differences of varying indices used does not have the same relationship to process time as the length of short reads. The search function in the algorithm is only dependent on the length of the short read and not the size of the index. While a trivially sized index has the potential to reduce the processing time of short reads, this is simply because a very small index could cause a greater amount of short reads to return no remaining matches after reduction iterations occur. With an appropriately sized index, the size only affects the number of remaining matches reduced each iteration. The total processing time is still entirely dependent on the number of short reads and the frequency in which mismatches occur.

Additional improvements to the algorithm were investigated through the use of varying compiler optimizations. The compiler's ability to control the use of the 8-way VLIW and SIMD instructions of the DSP architecture for better parallel performance exceeded attempts to control the functionality for manual performance increases. The compiler was effective in taking advantage of the core's ability of parallelism and the compiler settings chosen by the Code Composer Studio were used for the calculated results.

5.3 Software Development

In developing a CPU implementation, considerations were taken to assure that an equal comparison to the DSP could be made. The same algorithm was used to develop a CPU implementation to illustrate the acceleration improvements of the DSP. The implementation was written in C and compiled with the standardized "gcc" compiler. No compiler optimizations were used to build the binary. The implementation was maintained as closely to the DSP implementation as possible, keeping the searching

portion of the code identical to that of the DSP. The platform used to run the CPU implementation was a Dell Optiplex 980 Enterprise-level Minitower with 3GB of DDR3 memory and an Intel Core I5-650 processor running Ubuntu 12.04.

5.4 Test Cases

Data used for testing and calculating the results were based on the human genome chromosome 22. The chromosome database, hs_ref_GRCh37.p10, was obtained from the National Center for Biotechnology Information repository and contains 34.9 million bases [26]. The index was calculated from a C program developed to perform the BWT and to generate the improvement tables. The tables were written to 2 sets of files. The CPU implementation used a binary format that stored the tabling information containing the index, while the DSP implementation used a separate format developed by Texas Instruments to allow for blocks of data to be loaded directly into memory of the DSPs.

Short reads were generated from the chromosome 22 database to ensure that matches were possible and mismatches would be at a controlled rate. Files were generated for short reads with length of 35 bases. The number of reads for the files are as follows: 10K, 100K, 1M, 2M, 2.5M, 3M, 4M, 5M, 10M, 100M, and 1B. For each grouping of numbers of reads, a percentage of mismatches were introduced to better understand the relationship between mismatched search terminations and processing time. Each category was also generated to have 0%, 25%, 50%, 75%, and 100% of the reads that are mismatches in order to confirm that there were no unexpected performance results or data generated. Table 5.1 shows the number of matched and mismatched reads for each category.

Table 5.1 Generated Matched and Mismatched Short Read Data

Num of Reads	% Mismatch	Expected		Num of Reads	% Mismatch	Expected	
		Match	Mismatch			Match	Mismatch
10,000	0	10000	0	4,000,000	0	4000000	0
	25	7507	2493		25	3000692	999308
	50	4977	5023		50	2002077	1997923
	75	2445	7555		75	1001015	2998985
	100	0	10000		100	0	4000000
100,000	0	100000	0	5,000,000	0	5000000	0
	25	75148	24852		25	3750030	1249970
	50	50234	49766		50	2500790	2499210
	75	24991	75009		75	1250030	3749970
	100	0	100000		100	0	5000000
1,000,000	0	1000000	0	10,000,000	0	10000000	0
	25	750605	249395		25	7499756	2500244
	50	501785	498215		50	4999306	5000694
	75	250897	749103		75	2499454	7500546
	100	0	1000000		100	0	10000000
2,000,000	0	2000000	0	100,000,000	0	100000000	0
	25	1500292	499708		25	74997560	25002440
	50	999913	1000087		50	49993060	50006940
	75	498887	1501113		75	24994540	75005460
	100	0	2000000		100	0	100000000
2,500,000	0	2500000	0	1,000,000,000	0	1000000000	0
	25	1874394	625606		25	749975600	250024400
	50	1249759	1250241		50	499930600	500069400
	75	623712	1876288		75	249945400	750054600
	100	0	2500000		100	0	1000000000
3,000,000	0	3000000	0				
	25	2250311	749689				
	50	1500743	1499257				
	75	750086	2249914				
	100	0	3000000				

Mismatched short reads were generated randomly based on a percentage basis. With randomization, the mismatches are not an exact number based on the percentage. The total number of mismatched reads and matched reads will still result in the expected number of reads to process. In addition to a percentage based random number of mismatches, each read has a random number of mutated bases between 1 and 3. The randomization helps to ensure that an outside pattern does not form, which would have an influence over the performance results. Terminating reads at random points also aids in a realistic performance result.

The generated short reads for the 25% mismatch categories are the focus of the performance results for this thesis. Having 70% to 80% of short reads matching exactly to a human reference genome, as was previously explained, coordinates directly with the 25% mismatch category having 75% of the short reads matching and gives a more realistic result. The remaining categories are used for calculations to ensure that the results are as expected, eliminating the possibility of erroneous outliers corrupting performance statistics.

5.5 Performance Metric

The most widely used comparison for short read mapping is the amount of time taken to process a number of reads. Investigating the implementations of this thesis has led to the development of a model to derive the performance bounds of a DSP implementation to examine whether the use of a DSP provides a benefit and the significance of any benefits determined.

The same model was used in the comparison of the DSP's performance to the CPU implementation and indicates the number of mapped bases per watt-second. Adding the concept of power consumption allows varying architectures to be compared in a competitive method, while the addition of read length in the metric eliminates improper comparisons of algorithms that handle differing sized short reads. This performance metric allows for more accurate analysis when a comparison is done, especially as technologies lead to more power aware applications.

CHAPTER 6

RESULTS

Evaluation of an architecture's ability to outperform another must be calculated in a way to reduce the possibility of bias with an oversight of an unfair comparison between varying technologies. This thesis strives to achieve this goal with the indicated performance metric based on the results calculated as will be described by this chapter.

The timing measurements calculated with both CPU and DSP implementations were conducted measuring the time difference only between the start and end of the search function for each of the categorized numbers of short reads generated. Power measurements were taken using a Yokogawa WT500 power analyzer sampling at 0.1 second intervals. In cases where the timing of the search function completed in less than 0.1 seconds, an average of the data collected from the power analyzer for the few samples before and after that interval were also collected and averaged to reduce the effects of momentary power spikes from voltage variances as well as subroutines and background processes that may have had a minute effect on the power usage of the computer. All nonessential peripherals were disconnected from the system during CPU power measurements and the DSP card was removed. Due to the DSP board's PCI express power connection, the same computer system used for the CPU calculations was measured over a 5 minute timeframe prior to the calculations to generate an idle power consumption baseline. The baseline was subtracted from the power measurements taken during the DSP testing.

The performance gains of the DSP when compared to the CPU implementation indicate a significant speedup. The system of measuring performance purely based on the number of mapped reads per second is indicated in table 6.1 along with the reflected speedup relative to CPU performance. Tests were conducted with the CPU, 1 core of a single DSP, 8 cores utilizing all of a DSP's logic, and all 4 DSPs of the board with a total of 32 cores. The smallest improvement comes from a single processing core of the DSP when compared to the CPU using 1 Intel I5-650 core, ranging from speedups between 1.13x to 1.39x with an average speedup over the differing number of short reads being 1.18x. The performance increases as more cores are introduced. The 8 core averaged a 9.18x speedup while the 32 core averaged a 36.58x speedup.

Consideration was taken to compare the memory management system of the CPU and the direct memory access of the DSP. In order to ensure a proper comparison, the page faults were calculated on the CPU implementation to investigate if hard drive access time was a factor in the performance measurements taken. For all tests, there were no major faults causing a disk access to occur. There were a total of 6 minor page faults for each test, which did not change with index size or the number of short reads. These minor page faults occur when a new page is allocated in memory and can be attributed to the function variables initialized in the mapping algorithm. As these faults are not hard drive accesses, they would not affect the performance or alter the results shown.

As page faults are not an affecting factor to the performance gains of the DSP, the effective bandwidth of both the CPU and the DSP implementation was calculated to better understand the capability of the memory systems. The average effective bandwidth was calculated by counting the number of memory accesses during each run. While the

Table 6.1 DSP and Software Mapped Reads per Second. DSP core speedup comparisons are relative to the 1 core CPU implementation performance.

Number of Reads	Mapped Reads per Second				Speedup over CPU			
	Cores				Cores			
	32	8	1	Software	32	8	1	CPU
10K	4298145.99	1094382.38	138640.45	100000.00	42.98	10.94	1.39	1.00
100K	4390926.27	1096580.71	138565.56	119047.62	36.88	9.21	1.16	1.00
1M	4317765.86	1081169.33	138702.50	118764.85	36.36	9.10	1.17	1.00
2M	4289152.57	1074429.33	138760.75	122699.39	34.96	8.76	1.13	1.00
2.5M	4275801.23	1070609.09	138797.10	122970.98	34.77	8.71	1.13	1.00
3M	4272912.23	1069581.50	138769.88	118811.88	35.96	9.00	1.17	1.00
4M	4267621.43	1069119.06	138762.25	117785.63	36.23	9.08	1.18	1.00
5M	4266514.04	1068309.72	138774.21	117702.45	36.25	9.08	1.18	1.00
10M	4256039.73	1065363.81	138772.08	118231.26	36.00	9.01	1.17	1.00
100M	4246645.04	1063742.14	138757.24	117910.63	36.02	9.02	1.18	1.00
1B	4245745.84	1063203.36	138643.64	117909.79	36.01	9.02	1.18	1.00
Average	**4284297.29**	**1074226.40**	**138722.33**	**117439.50**	**36.58**	**9.18**	**1.18**	**1.00**

listed maximum bandwidth of the Intel Core I5-650 is 21 GBytes per second, the CPU was calculated to have only achieved 96.15 Mbytes per second. One possibility of the greatly reduced throughput may be a limitation of a single core on the processor, which was used for a better comparison to the DSP. While still less than the maximum stated 16 GBytes per second, the DSP was able to achieve 3.46 GBytes per second throughput. Table 6.2 shows the calculated effective bandwidth for the CPU and DSP.

Introducing the developed performance metric indicates an even stronger gain between the DSP implementation and the CPU. With the power consumption of the CPU ranging from 131 watts to 140 watts and the DSP using between 45 and 51 watts, the gains can clearly be seen as illustrated in table 6.3. The single core performance was able to provide an average improvement of 3.59x. Subsequently the 8 core and 32 core implementations show an average performance increase of 25.94x and 102.76x, respectively. Figures 6.1 and 6.2 illustrate a graphic representation of the mapped reads

Table 6.2 Effective bandwidth of the CPU and DSP.

Number of Reads	Bytes per Second			
	Cores			
	32	8	1	CPU
10K	3,515,549,352.48	893,687,536.76	113,215,677.78	81,661,360.00
100K	3,524,246,053.64	882,471,830.82	113,211,726.80	97,264,533.33
1M	3,499,641,341.06	882,471,830.82	113,211,726.80	100,113,918.78
2M	3,499,641,341.06	876,657,388.22	113,218,833.38	100,113,918.78
2.5M	3,488,008,473.56	873,355,280.00	113,224,502.59	100,314,255.39
3M	3,486,538,153.00	872,738,896.29	113,231,086.40	96,946,095.52
4M	3,482,355,704.29	872,395,291.28	113,229,238.37	96,112,431.57
5M	3,481,092,035.53	871,644,724.25	113,227,282.31	96,034,620.06
10M	3,472,501,145.32	869,229,915.12	113,224,089.25	96,464,836.74
100M	3,149,850,928.55	789,006,178.03	102,919,979.61	96,203,234.46
1B	3,429,804,325.06	858,878,422.82	111,999,299.03	96,389,495.97
Average	3,457,202,623.05	867,503,390.40	112,173,949.30	96,147,154.60

Table 6.3 DSP and Software Mapped Bases per Watt-Second. DSP core speedup comparisons are relative to the 1 core CPU implementation performance.

Number of Reads	Mapped Bases per Watt-Second				Efficiency over CPU			
	Cores				Cores			
	32	8	1	Software	32	8	1	CPU
10K	3247735.53	821432.20	104218.55	24957.22	130.13	32.91	4.18	1.00
100K	3285216.32	809540.71	101672.84	31382.59	104.68	25.80	3.24	1.00
1M	3140519.64	777819.66	107449.92	31362.38	100.14	24.80	3.43	1.00
2M	3151140.64	789855.63	110907.20	32433.19	97.16	24.35	3.42	1.00
2.5M	3194983.84	784411.10	111470.83	32534.46	98.20	24.11	3.43	1.00
3M	3252543.02	805581.07	114442.65	31353.51	103.74	25.69	3.65	1.00
4M	3107275.85	780704.51	113819.52	31188.51	99.63	25.03	3.65	1.00
5M	3072592.42	776065.59	111811.63	31194.80	98.50	24.88	3.58	1.00
10M	3045622.38	809546.97	112849.05	31076.10	98.01	26.05	3.63	1.00
100M	3224133.97	805690.87	113895.48	31001.14	104.00	25.99	3.67	1.00
1B	2989360.38	801294.52	113244.51	31090.85	96.15	25.77	3.64	1.00
Average	**3155556.73**	**796540.26**	**110525.65**	**30870.43**	**102.76**	**25.94**	**3.59**	**1.00**

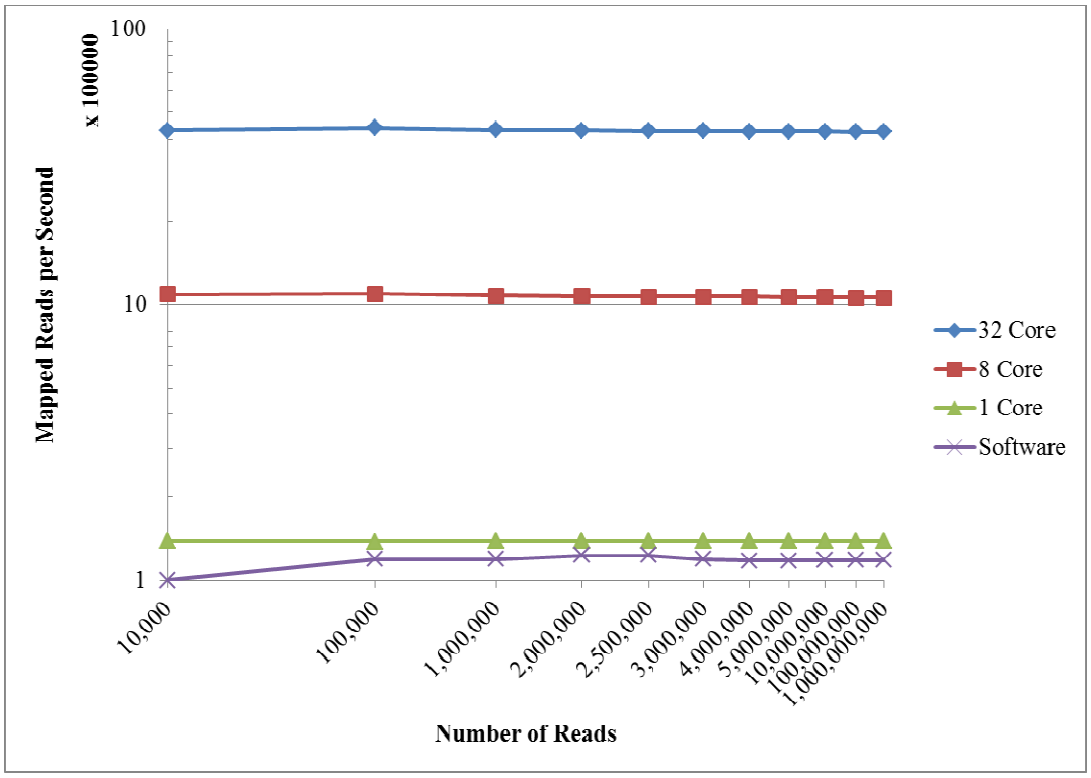

Figure 6.1 DSP and CPU Mapped Reads per Second

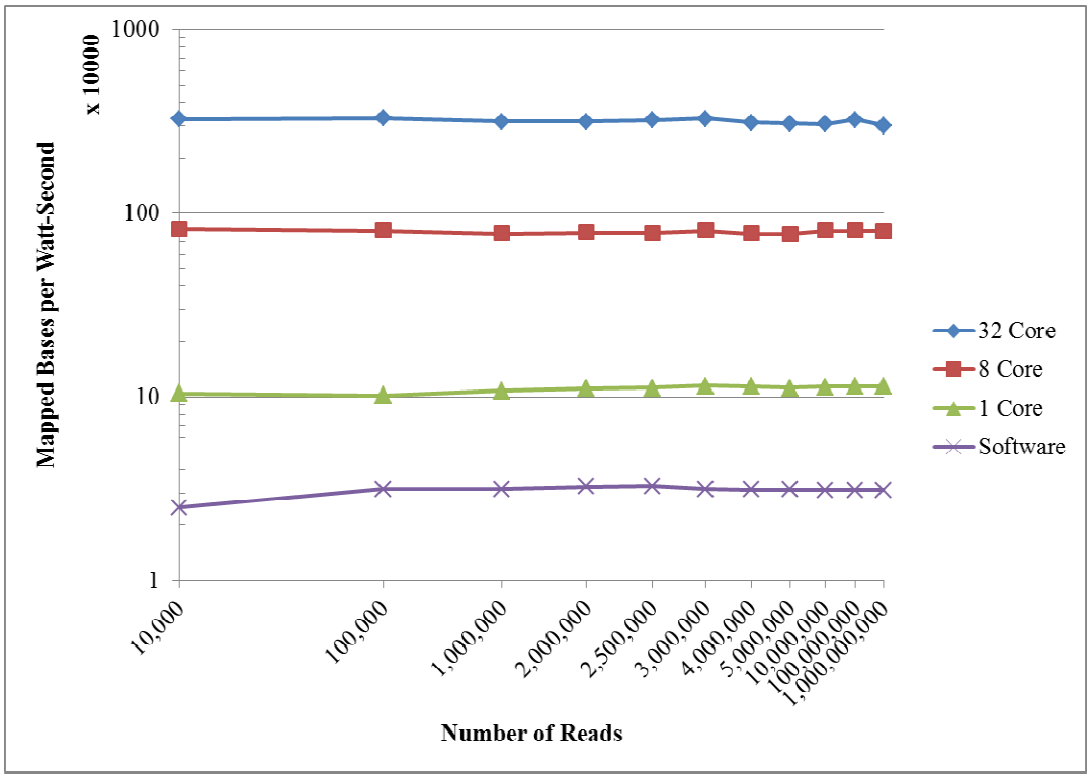

Figure 6.2 DSP and CPU Mapped Bases per Watt-Second

per second and mapped bases per watt-second, respectfully. The improvements for both metrics are shown in the performance comparison of the DSP, GPU, CPU, and FPGA represented by figure 6.3.

The study that was used as a basis for the development of this thesis provided results of the performance for the GPU implementation compared to a CPU. That study achieved a significant improvement using a GPU and results were given for a comparison of 7 million reads processed in 0.432 seconds, which achieves 16.2 million reads per second performance [4]. While this method of performance measurements indicate a 3.78x improvement of the DSP implementation described in this thesis, it fails to account for the difference in architectures used. The hardware described in that study used 2 NVidia Tesla C2050 GPUs, which each contain 448 cores and consume a maximum of 247 watts. NVidia lists the running power consumption of a single GPU at 238 watts [18]. Accounting for the 2 GPUs used, the mapped bases per watt-second of the GPU implementation is 676,742.83. Compared to the DSP, which achieves 3,155,556.73 mapped bases per watt-second, there is a 4.66x improvement over the GPU. The power consumption for the performance metric for the GPU must be estimated instead of measured, only the running power for the GPUs was used in the calculations. More power would be consumed with the controlling CPU needed to operate the GPU, which would decrease the performance of the GPU a small degree more than indicated.

Additionally, a comparison with an FPGA was investigated. A study previously discussed in this thesis implemented the RMAP algorithm on an FPGA [10]. While this algorithm is different from those listed above, it still gives an indication of the exact match algorithm performance. That study indicated that for 1 million reads, the FPGA

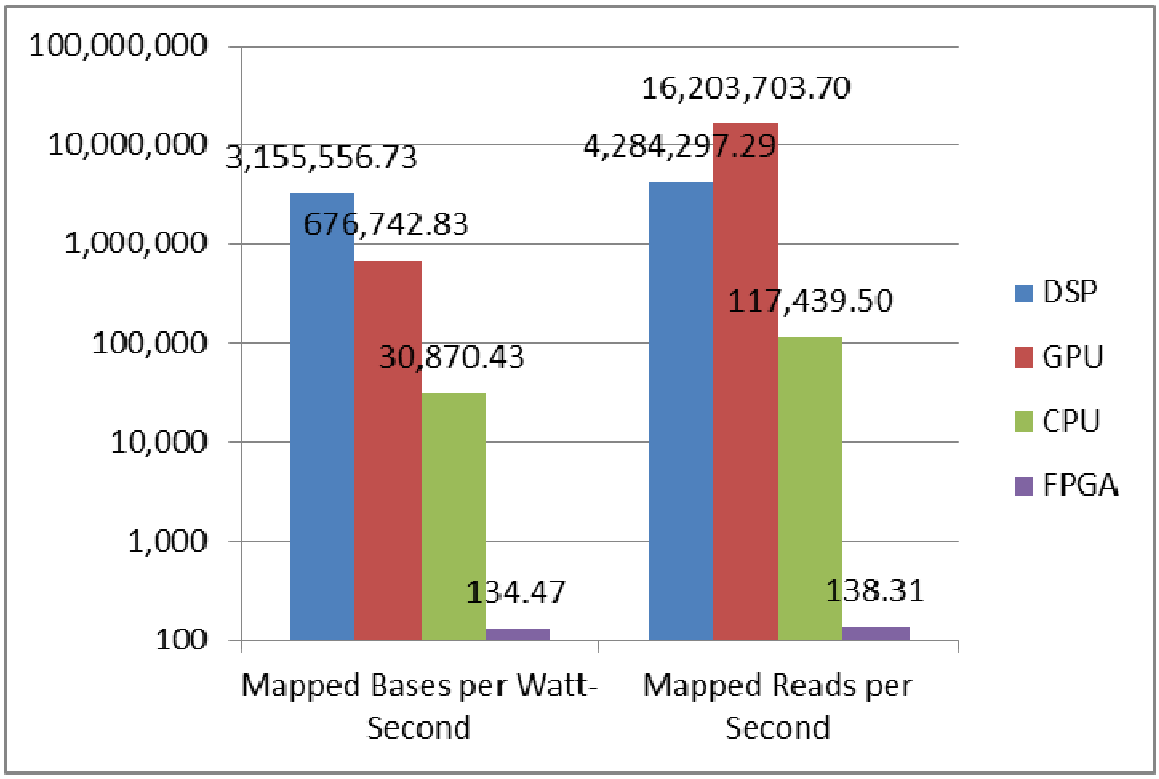

Figure 6.3 DSP, GPU, CPU and FPGA Performance Comparison

took 7,230 seconds to process the data. The resulting mapped reads per second for the FPGA is 138.31. Using the Xilinx estimated 36 watts of running power for the Virtex 5 LX330 FPGA, the calculated mapped bases per watt-second is 134.47 [27]. While there was a 2.36x improvement of the FPGA over the CPU in the study, there is a drastically lower performance when compared to the GPU and even more when compared with the DSP. The DSP in this instance achieves a 23,467x improvement over the FPGA and 102.22x over the CPU, shown in comparison with the DSP, GPU and CPU in figure 6.3.

CHAPTER 7

CONCLUSION

Overall performance comparisons can be viewed in many ways. The performance metric that was developed in this thesis shows a valuable indication of the processing ability while still maintaining an aspect of the architecture's power. Without this significant change, attempting to view the results from comparisons between 2 different architectures does not give the view needed to understand their relationship. The result of a supercomputer's processing ability when compared to a laptop computer would show dramatically better performance when considering only the processing power. When you take into account the power consumed by the supercomputer, the results are leveled out to understand that the laptop's performance may not be insignificant.

The same can be said with any other architecture, particularly as power aware computing becomes more important. The results given in the previous chapter show that the DSP implementation was on average 102x better than the performance of the CPU and 4.66x better than the GPU. While the GPU's performance without respect to power was 3.78x faster than the DSP, this can be explained by the processing difference between the two technologies and the availability of 28x more processing cores in the GPU tests. Using a DSP board with more available cores, such as the Advantech DSPA-8901 with 20 DSP's totaling 160 cores would easily make up for the difference. This reasoning also confirms the use of the performance metric to bring the results of both into the same scope, showing the strong performance potential of the DSP.

With the stated goal of being able to process short reads as they are generated from NGS, the current 4 DSP board's ability to handle 4.1 million reads per second would be more than adequate to handle any of today's generating needs. The Illumina HiSeq 2500, previously mentioned, having the capability to process a whole human genome of 1.2 billion reads in a 27 hour timespan equates to approximately 333,000 reads per second, which would easily be handled by the DSP [1]. Being able to process the reads as they are generated eliminates the need to have available data storage for large volumes of short reads and allows more storage for reference indices. This further allows for multiple smaller chromosomes' indices to be stored and searched by a single DSP, needing less DSPs to process the reads for all chromosomes of the human genome.

The work done in this thesis was successful in demonstrating the DSP to be a powerful alternative processor for exact matching short read mapping. The use and implementation of the DSP will provide an important first step in accelerating the exact match stage of the sequencing pipeline with the potential for improving the remaining stages that will form the next bottleneck and need for processing improvements.

REFERENCES

[1] Illumina Inc, "Sequencing Technology," May 2013. [Online]. Available: http://www.illumina.com/technology/sequencing_technology.ilmn.

[2] M. Burrows and D. J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," Digital Equipment Corporation, Palo Alto, 1994.

[3] B. Langmead, C. Trapnell, M. Pop and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, no. 3, p. R25, March 2009.

[4] S. Chen and H. Jiang, "An Exact Matching Approach for High Throughput Sequencing based on BWT and GPUs," *The 14th IEEE International Conference on Computational Science and Engineering*, vol. 163, pp. 173-180, 2011.

[5] US Department of Energy Genome Programs, "Human Genome Project," Biological and Environmental Research Information System, 30 July 2012. [Online]. Available: http://genomics.energy.gov/.

[6] X. Yang, D. Medvin, G. Narasimhan, D. Yonder-Himes and S. Lory, "CloG: a pipeline for closing gaps in a draft assembly using short reads," *IEEE International Conference on Computational Advances in Bio and Medical Sciences*, pp. 202-207, 2011.

[7] J. Arram, K. H. Tsoi, W. Luk and P. Jiang, "Reconfigurable Acceleration of Short Read Mapping," *21st Annual International IEEE Symposium on Field-Programmable Custom Computing Machines*, vol. 57, pp. 210-217, 2013.

[8] Y. Li, J. M. Patel and A. Terrell, "WHAM: A High-Throughput Sequence Alignment Method," *ACM Transacions on Database Systems*, vol. 37.4, no. 28, 2012.

[9] W. Tang, W. Wang, B. Duan, C. Zhang, G. Tan, P. Zhang and N. Sun, "Accelerating Millions of Short Reads Mapping on a Heterogeneous Architecture with FPGA Accelerator," *IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, vol. 39, pp. 184-187, 2012.

[10] E. Fernandez, W. Najjar, E. Harris and S. Lonardi, "Exploration of Short Reads Genome Mapping in Hardware," *International Conference on Field Programmable Logic and Applications*, vol. 78, pp. 360-363, 2010.

[11] H. Li, B. Ni, M.-H. Wong and K.-S. Leung, "A Fast CUDA Implementation of Agrep Algorithm for Approximate Nucleotide Sequence Matching," *IEEE Symposium In Application Specific Processors*, vol. 9, pp. 74-77, 2011.

[12] Y. Liu and B. Schmidt, "Evaluation of GPU-based Seed Generation for Computational Genomics using Burrows-Wheeler transform," *IEEE 26th International Parallel and Distributed Processing Symposium Workshops*, vol. 85, pp. 684-690, 2012.

[13] Texas Instruments Incorporated, "TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor," 22 February 2012. [Online]. Available:

http://www.ti.com/product/tms320c6678.

[14] U.S. National Library of Medicine, "Genetics Home Reference: Your Guide to Understanding Genetic Conditions," 6 May 2013. [Online]. Available: http://ghr.nlm.nih.gov/.

[15] S. Kaisler, F. Armour, J. A. Espinosa and W. Money, "Big Data: Issues and Challenges Moving Forward," *46th Hawaii International Conference on System Sciences,* vol. 645, pp. 995-1004, 2013.

[16] H. Li, J. Ruan and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Research,* no. 18, pp. 1851-1858, 2008.

[17] R. Li, Y. Li, K. Kristiansen and J. Wang, "SOAP: short oligonucleotide alignment program," *Bioinformatics,* no. 24, pp. 713-714, 2008.

[18] NVIDIA Corporation, "High Performance Computing - Accelerating Science with Tesla GPUs," 2013. [Online]. Available: http://www.nvidia.com/object/tesla-supercomputing-solutions.html.

[19] K. Kasichayanula, D. Terpstra, P. Luszczek, S. Tomov, S. Moore and G. D. Peterson, "Power Aware Computing on GPUs," *2012 Symposium on Application Accelerators in High Performance Computing,* no. 26, pp. 64-73, 2012.

[20] D. Dai, X. Li, C. Wang and X. Zhou, "Cloud Based Short Read Mapping Service," *IEEE International Conference on Cluster Computing,* no. 60, pp. 601-604, 2012.

[21] O. Knodel, T. B. Preußer and R. G. Spallek, "Next-Generation Massively Parallel Short-Read Mapping on FPGAs," *IEEE Inernational Conference on Application-*

Specific Systems, Architectures and Processes, pp. 195-201, 2011.

[22] T. B. Preußer, O. Knodel and R. G. Spallek, "Short-Read Mapping by a Systolic Custom FPGA Computation," *IEEE 20th International Symposium on Field-Programmable Custom Computing Machines,* no. 37, pp. 169-176, 2012.

[23] J. S. Torres, I. B. Espert, A. T. Dominguez, V. H. Garcia, I. M. Castello, J. T. Gimenez and J. D. Blazquez, "Using GPUs for the Exact Alignment of Short-Read Genetic Sequences by Means of the Burrows-Wheeler Transform," *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* vol. 9, no. 4, pp. 1245-1256, 2012.

[24] A. M. Aji, L. Zhang and W.-c. Feng, "GPU-RMAP: Accelerating Short-Read Mapping on Graphics Processors," *13th IEEE International Conference on Computational Science and Engineering*, no. 29, pp. 168-175, 2010.

[25] Advantech Co., Ltd., "DSP Processing Platforms," 2013. [Online]. Available: http://www.advantech.com/prodcuts/DSP-Processing-Platforms/sub_DSP_PROCESSING_PLATFORMS.aspx.

[26] National Center for Biotechnology Information, "Human Genome Resources," U.S. Naional Libary of Medicine, Naional Institutes of Health, 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/projects/genome/guide/human/index.shtml.

[27] Xilinx Inc., "Xilinx Vertex 5 FPGA," 2013. [Online]. Available: http://www.xilinx.com/support/index.html/content/xilinx/en/supportNav/silicon_devices/fpga/virtex-5.html.