CONFIGURABLE HARDWARE BASED GENOME ALIGNER

by

Mehmet Yağmur Gök

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Electrical and Electronics Engineering

Yeditepe University

2016

CONFIGURABLE HARDWARE BASED GENOME ALIGNER

APPROVED BY:

Prof. Dr. Cem Ünsalan .....
(Thesis Supervisor)

Assoc. Prof. Dr. Sezer Gören Uğurdağ .....
(Thesis Co-supervisor)

Prof. Dr. Işıl Aksan Kurnaz .....

Assoc. Prof. Dr. Fatih Uğurdağ .....

Assoc. Prof. Dr. Duygun Erol Barkana .....

Assoc. Prof. Dr. Oğuzhan Külekçi .....

Assist. Prof. Dr Engin Maşazade .....

DATE OF APPROVAL: ..../..../2016

ACKNOWLEDGEMENTS

First and foremost, I would like to thank both personally and professionally, to my advisor Cem Ünsalan for his guidance and support which made the completion of this study possible. I will remember every word of his valuable advices through all my future studies.

I am grateful to my cosupervisor Sezer Gören for her support, her approach which forced me to clarify and explain each point of study, and the valuable time she spent for me.

I would like to express my deep gratitude to Mahmut Şamil Sağıroğlu for his valuable ideas and encouragement.

I would like to thank EE faculty and Yeditepe University for supporting my education. I am grateful to TUBITAK BILGEM which gave me the chance to pursue my academic studies together with my professional life. I also want to thank to Mete Akgün for his support.

ABSTRACT

CONFIGURABLE HARDWARE BASED GENOME ALIGNER

Next Generation Sequencing (NGS) machines produce enormous amount of data via massively parallelizing DNA sequencing. In bioinformatics, processing and storage of millions of short read data produced is one of the main problems. Hardware platforms, especially Field Programmable Gate Arrays (FPGA), are useful tools to overcome this computational burden. Within bioinformatics, alignment of short DNA read sequencing data to a reference genome sequence has become a standard step in the analysis pipeline for short DNA read sequence data. This is the costliest part of data analysis in terms of computation. Fundamentally, the problem is matching similar parts of two strings which are generally solved by Smith-Waterman (SW) algorithm which is a dynamic programming algorithm. SW algorithm is suitable to run efficiently on FPGA platforms. Besides, an FPGA platform can be used with a PC in a hybrid manner to form a complete system for analyzing and storing the NGS data. In this dissertation, we propose such a hybrid sequence alignment system to obtain the best alignment for short reads. The algorithm, design, and results of this dissertation describe the implementation, as well as improvements in mapping sensitivity and accuracy. The proposed system aligns NGS short reads to reference genome utilizing Phred Quality scores to obtain better mapping accuracy. This scheme results in mapping the reads to the locations that they fit best. This way, the proposed system approximates the optimum solution that can be obtained by dynamic programming. PC side of the system compresses read sequences along with the alignment results. We compare our system with other software and FPGA based systems in terms of sensitivity and accuracy. Based on the experiments, our proposed system provides increased sensitivity and accuracy.

ÖZET

KONFİGÜRE EDİLEBİLİR DONANIM TABANLI GENOM HİZALAYICI

Yeni nesil dizileme platformları (NGS), DNA dizilimini ileri düzeyde paralelleştirerek muazzam miktarda genetik veri ortaya çıkarırlar. Bu büyüklükteki verinin işlenmesi ve depolanması biyoinformatik alanındaki önemli problemlerdendir. Donanım tabanlı çözümler; özellikle de yeniden yapılandırılabilir esnek özellikleri ile Alanda Programlanabilir Kapı Dizileri (FPGA) bu işlemsel yoğunluğun üstesinden gelme konusunda faydalı yapılardır. Biyoinformatik veri analizi zincirinde, NGS çıktıları olan kısa okumaların referans genoma hizalanmaları standart bir basamağı oluşturur ve işlem yükü olarak en ağır basamaktır. Temel olarak problem, aynı alfabeden iki dizinin benzer kısımlarini eşleştirme problemidir. Problemin çözümünde tercih edilen yöntem de esasında bir dinamik proglamlama yaklaşımı olan Smith-Waterman (SW) algoritmasidır. SW algoritması FPGAler üzerinde efektif olarak gerçeklenebilir ve paralellenebilir bir algoritmadır. Bununla beraber FPGA tabanli platformlar, PC'ler ile hibrit olarak çalıstırılarak, NGS datasının analizine ve depolanmasına yönelik bir sistem meydana getirilebilir. Bu çalışmada, kısa genomik dizileri olaslı en doğru pozisyona hizalayabilen hibrit bir sistem önerilmiş ve gerçeklenmiştir. Tezde gerçeklenen sistemin tasarımı, algoritması ve sonuçları gerçeklenmesini ve bunun yanında hassasiyet ve hizalama doğruluğunda, nihai olarak da hizalama kalitesinde elde edilen gelişmeyi tanımlar. Önerilen sistem hizalama kalitesini arttırmak amacı ile hizalama algoritmasında, Phred kalite skorlarını da kullanmaktadır. Bu yaklaşımla beraber referans dizisinin tamamına hizalama yapılarak, diziler en doğru pozisyona hizalanmaktadır. Hizalanan dizi verisi, sistemin PC tarafında hizalama sonuçları ile etkileşimli olarak sıkıştırılarak saklanır. Sistemin performansı yaygın kullanılan yazılım tabanlı hizalayıcılar ve FPGA tabanlı metodlar ile karşılaştırılmış ve denemelerin sonucunda daha yüksek hizalama kalitesi ortaya koyduğu gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

A	Adenine Nucleotide
C	Cytosine Nucleotide
G	Guanine Nucleotide
T	Thyminer Nucleotide
α	Gap Opening Penalty
β	Gap Extending Penalty
Ω	Reward Penalty Comparator
μ	Gap Penalty
BLAST	Basic Local Alignment Search Tool Program
bp	Base Pair
BWT	Burrows Wheeler Transform
CA	Computation Array
CCD	Charge-coupled Device Camera
CPU	Central Processing Unit
de Novo	Assembling Short Reads to Create Full-length Sequences
DDR3	a type of Random Access Memory
DLL	Delay-locked Loop
DNA	Deoxyribonucleic Acid
DPE	Data Processing Element
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
E(i,j)	Left Neighbour Cell Score in Similarity Matrix Computations and Alignment of a Character with a Gap
F(i,j)	Upper Neighbour Cell Score in Similarity Matrix Computations and Alignment of a Character with a Gap
fasta	a Read Data File Format
FASTQ	a Read Data File Format

FIFO	First In First Out Data Channel
FM	Compressed Full-text Substring Index based on BWT
FPGA	Field Programmable Gate Array
FPhQ()	Phred Quality Function
fsn	Read Data File Format of Roche454
GB	Giga Byte
GPU	Graphical Processing Unit
GS	Genome Sequencer 454
H(i,j)	Cell Score in Similarity Matrix Computations
HPG	Human Genome Project
IO	Input-Output
init	Initialize Signal
LUT	Lookup Table
MAX	Maximum Comparator
MB	Mega Byte
MHz	Mega Hertz
NGS	Next Generation Sequencing
NW	Needleman-Wunsch Algorithm
PCI express	a High Bandwidth Data Interface
PCR	Polymerase Chain Reaction
PE	Processing Element
$s(a_i,b_j)$	Similarity Score Function
$S_{i,j}$	Similarity Matrix Element
SBS	Sequencing by Synthesis
shiftRd	Shift Read Signal
shiftRf	Shift Reference Signal
SIMD	Single-instruction Multiple-data
SNP	Single Nucleotide Polymorphism
SRAM	Static Random Access Memory
SUMBASEQ()	Sum of Base Quality Scores in Read Sequence
SW	Smith Waterman

SWA	Smith Waterman Algorithm operation
Xilinx	Mainstream FPGA manufacturer

1. INTRODUCTION

Interest in exploring the genetic data of organisms is growing everyday; especially the data of mankind. Genetic data of *homo sapiens* is coded in a structure called as DNA which is unique for each person. DNA is a helical structure of two conjugate strands. Each strand in a string of three billion organic molecules which are named as nucleotides. There are four known nucleotides: Adenine, Thymine, Guanine, and Cytosine. Although about 99% of the DNA data of all human being is the same, the complete picture is unique. The string of nucleotides tells us much about the human. There are different ways of identifying the whole or part of the DNA string. Nowadays, one of most popular methods is using Next Generation Sequencing (NGS) platforms which dramatically reduce the cost of DNA sequencing. They also increase throughput. Therefore, they make a remarkable improvement[21] in analysis of genetic data. NGS finds application areas in various fields such as biology, ecology, forensics, agriculture, drug development, and individual drug design.

NGS platforms randomly fragment many copies of the genome of an organism. As an output, they give short DNA fragments with determined nucleotide sequences. These are called as *short reads* and usually have length between 20 and 300 base pairs. The NGS process is performed in a highly parallel manner providing massive amount of genomic data in forms of millions of short reads. The bottleneck here is determining the location in the reference genome to which each short read maps best. On the other hand, this is the first step in the DNA data analysis pipeline with three critical aspects. The first is the sensitivity of the mapping algorithm. This is defined as the capacity of the algorithm to map reads which do not have an identical match in the reference genome due to sequencer errors or genetic variances. Mapping non-identical reads is particularly important since the difference may point to a genetic disease. Here, distinguishing sequencer errors and genetic variances is possible since the former is random. The second aspect of mapping is its accuracy. This is defined as the mapping of reads to the best possible location in the reference string. In other saying, accuracy is about approximating the optimum solution to the dynamic programming problem. The third aspect of the mapping problem is the time required to match all reads considering the amount of data to be mapped, which may be hundred millions of short reads.

This is also highly related to the power consumption issue.

Read alignment is traditionally performed via software like Bowtie [1] and BWA [2]. Usually, these are run over a cluster of processors in order to get an acceptable performance. While the sequencing technology is developing, the size of data to be processed is also increasing. Therefore, the need for platforms offering better performance has emerged. Here, Graphical Processing Unit (GPU) and FPGA-based platforms are two widely used candidates [25].

The efficient storage of the processed NGS data is another related problem that requires fast and efficient compression methods. With the help of compression methods, NGS read sequences can be stored and transmitted in an economical manner. Therefore, compression is considered to be a key technology for data management of population-scale genome analysis [7]. The output of an NGS platform is short nucleotide sequences along with quality scores and text data from the input DNA. This is most of the times stored in the FASTQ file format.For FASTQ format, the text data containing the nucleotide sequence and quality values are uncorrelated and they can be compressed separately.

In this thesis, we aim to solve the read alignment and storage problems at once. Therefore, we propose a hybrid PC-FPGA based read aligner and compressor system. In the short read alignment part of the proposed system, we use the quality scores on a hardware platform and make the system more sensitive and accurate [53]. In the compression part of the proposed system, we use Golomb coding [8] for the text data containing the nucleotide sequence. For the compression of quality scores, we utilize a compression technique based on a new model for prediction by partial matching, which was formerly proposed by Akgun and Sagiroglu [9]. In the following section, we will review the related literature. Then, we will explain the proposed system step by step. Afterwards, we provide test results of the proposed system on standard data. Based on this, we analyze the strengths and weaknesses of the proposed system.

2. BACKGROUND

Determination of the genetic basis of human diseases is one of the main goals of molecular genetics. By the publication of the human genome in 2001, considerable progress has been achieved; especially during recent years [10]. Human genome project was a two decades time hard effort. But it motivated the development of new genome sequencing technologies which drastically reduced the amount of time required to produce sequence data from a sample of DNA [11]. Accordingly, they shortened the time and reduced the cost of studies. New sequencing technologies brought possibility of studying human variation at the genetic level [58]. These developments enabled important discoveries for finding out nucleotides and genes in the human genome which keep information about susceptibility to particular diseases.

2.1. DNA SEQUENCING

The human genome is the complete set of genetic information for humans. This information is encoded into nucleic acid sequence structures which are composed of succession of letters that indicate the order of nucleotides within a Deoxyribonucleic Acid (DNA). DNA represents the information which manages the functioning of human body. DNA is arranged in a double helix chain of linked units called nucleotides in which two parallel sugar phosphate backbone strands are linked with nucleotide bases. Nucleotide is an organic subunit composed of mainly a phosphate group and sugar. There are four different nucleotides available: adenine, thymine, guanine, and cytosine which are usually represented by alphabet letters A, T, G, and C. The structure of DNA and the nucleotide bases that hold this bonded, paired structure as illustrated in Fig. 2.1.

The process of DNA sequencing is determination of the arrangement of nucleotide bases. This arrangement represents genetic information. Identification of only one nucleotide of a base pair is enough. This is because; nucleotide A always pairs with T and C always pairs with G. This situation gives the fact that, sequencing one side of the DNA strand makes the other side automatically known.

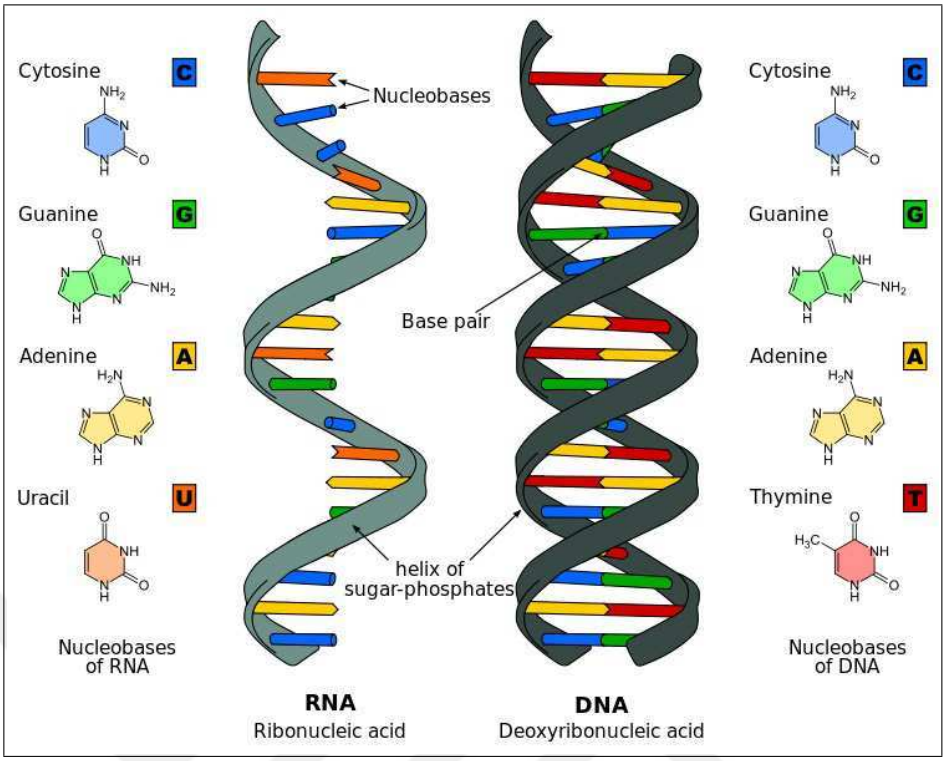


Figure 2.1. Structure of DNA, single stand and double helix strands.

DNA sequencing is mainly a chemical process and several methods have been developed for it. All methods mainly consist four key steps. In the first instance, DNA is removed from the cell. This can be done either mechanically or chemically. The second phase involves breaking up the DNA and inserting its pieces into vectors, cells that indefinitely self-replicate, for cloning. In the third phase, DNA clones are placed with a dye-labelled primer into a thermal cycler. Thermal cycler is a machine which automatically raises and lowers the temperature to catalyse replication. The final phase consists of electrophoresis, in which DNA segments are placed in a gel and subjected to an electrical current which moves them. Initially, the gel was placed on a slab. Today, it is usually inserted into a very thin glass tube named as capillary. When subjected to an electrical current, smaller nucleotides in the DNA move faster than the larger ones. Electrophoresis therefore helps sort out DNA fragments by their size. The different nucleotide bases in DNA fragments are identified by their dyes which are activated when they pass through a laser beam.

DNA sequencing offers valuable asset for a wide range of applications, including medicine,

biological research, and forensics. Personalized medicine has been of great interest in recent years, and genetic information allows the discovery of disease susceptibility and genetic risk factors [12]. This gives hope to locate the cause of genetic diseases, like certain cancer types. Sequencing provides exploiting individual genetic data for prevention purposes. Besides, identifying a malignant gene may be useful in the discovery of treatment and remedies. In forensics, DNA sequencing assists the identification process in challenging cases. DNA sequencing tools can be used on DNA evidence left at crime area to help revealing the suspect. An accurate identification process will not only ease current forensic issues. It also result in deterrence of future crimes.

2.1.1. Human Genome Project and Sanger Sequencing

Considering the benefits of discovering the genetic data more deeply, researchers became interested in sequencing human genome. These efforts resulted in the human genome project (HGP). HGP was an international collaborative research program whose goal was the complete mapping and understanding of all the genes of human beings. HGP researchers have deciphered the human genome in three major ways. The first was determining the order or sequence of all the bases in our genome's DNA. The second was making maps that show the location of genes for major sections of all our chromosomes. The third was producing the linkage maps through which inherited traits can be tracked over generations.

"The International Human Genome Sequencing Consortium published the first draft of the human genome in the journal Nature in February 2001 with the sequence of the entire genome's three billion base pairs some 90 percent complete. A startling finding of this first draft was that the number of human genes appeared to be significantly fewer than previous estimates, which ranged from 50,000 genes to as many as 140,000. The full sequence was completed and published in April 2003 [13].

Sequencing method employed in the project was the *Sanger sequencing* which was developed by Sanger in 1977. This brought him the Nobel prize. This method is also known as *dideoxy* or *chain termination* method. It uses DNA as a template and copies the template DNA repeatedly to generate a set of fragments that differ from each other by a

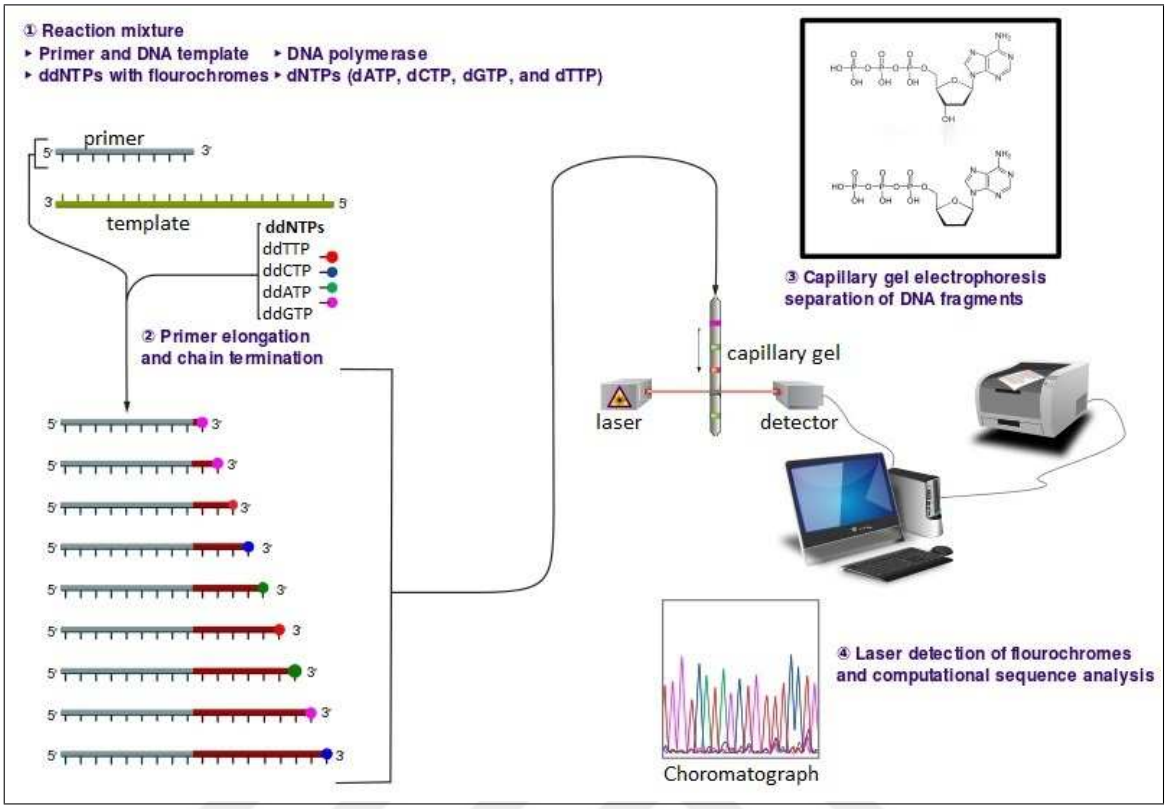


Figure 2.2. The Sanger method for DNA sequencing.

single nucleotide base. Analysis of these copies that vary in length let's us read the entire DNA sequence. A short illustration of the Sanger method is given in Fig. 2.2.

First step in Sanger's method is the separation of DNA strands to obtain a single-stranded DNA template. A short piece of DNA named primer which is complementary to a known sequence is bind to the template. With the injection of chemically altered versions of A, C, G, and T bases to the environment, enzyme, *DNA polymerase*, bind to the primer leads to the formation of new DNA strand complementary to the template. New strand formation process continues to the point of randomly incorporating a florescent labelled nucleotide. Labelled nucleotide is chemically altered and it terminates DNA formation via removing the enzyme. This process is repeated many times resulting many fragments of different length ending with florescent labelled tags. These fragments are the placed into thin glass capillaries and passed through a gel matrix. Applying electric field over gel matrix cause negatively charged DNA fragments to move. Through the motion, longer fragments slow down more, resulting in the sort of fragments by their length. A laser attached to the

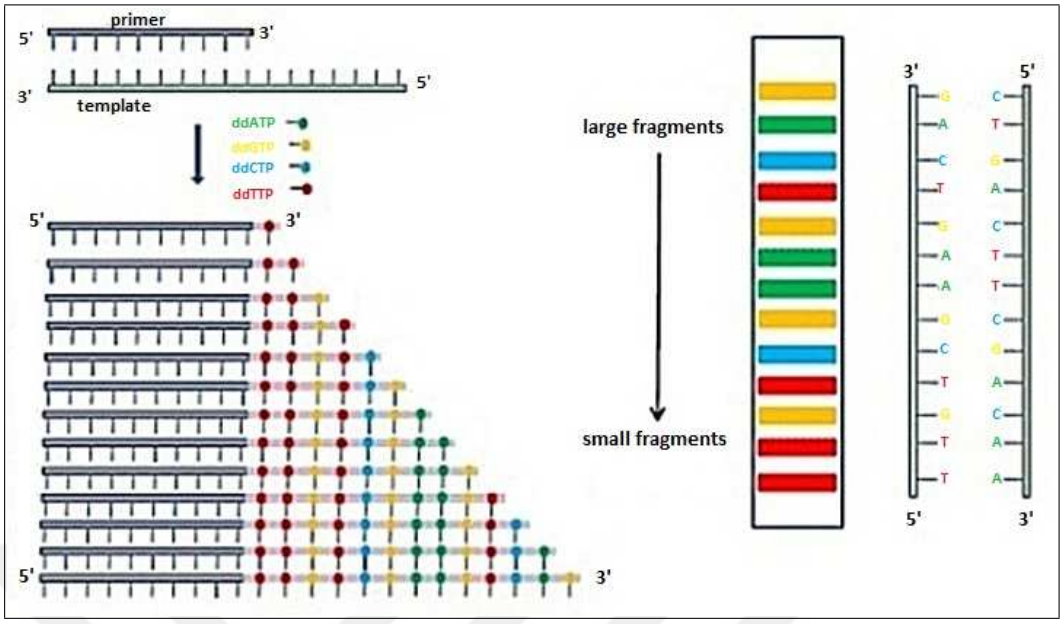


Figure 2.3. Obtaining the decoded DNA in Sanger's method.

end of capillary excites the final fluorescent nucleotide recorded as a coloured bar as in Fig. 2.3. Each coloured bar represents the final base of each fragment ordered from shortest to longest. The sequence of the original piece of DNA can therefore be decoded [14]. Each reaction produces 500 to 800 bases of DNA sequence. Despite being revolutionary, the Sanger method took more than three years to sequence human genome.

2.1.2. Next Generation Sequencers

Since completion of the first human genome sequence, demand for cheaper and faster sequencing methods has increased greatly. This demand has driven the development of second-generation sequencing methods, or next-generation sequencing (NGS). At NGS platforms, the sequence data generated at a single experiment is far greater than capillary electrophoresis-based Sanger sequencers.

NGS sequencer is a scientific instrument which is developed for the purpose of automating the sequencing process. Via taking an extracted sample of DNA as input, sequencer outputs the order of four bases which are reported as text strings called a read. The first automated DNA sequencer was invented in 1987 by Smith. The sequencer was employing the Sanger

sequencing method of human genome project in 2001 [15].

In principle, the concept behind NGS technology is similar to capillary electrophoresis, or Sanger's method. DNA polymerase catalyzes the incorporation of florescent labeled deoxyribonucleotide triphosphates (dNTPs) into a DNA template strand during sequential cycles of DNA synthesis. During each cycle, at the point of incorporation, the nucleotides are identified by fluorophore excitation. The critical difference is that, instead of sequencing a single DNA fragment, NGS extends this process across millions of fragments in a massively parallel format.

NGS machines randomly slice DNA strands into *short read* components and sequence millions of bases in parallel. The length of short reads have decreased considerably from 650 to 800 base pair(bp) reads of the original Sanger sequencing technique where current NGS machines output short read sequences of 20 to 300 bp long. Shorter reads result in massive parallelization, and therefore result in an efficient method of DNA sequencing. The main drawback of the short read output is that loss of information for reassembly makes the process more costly and brings heavy computation burden.

The first next-generation system on the market was Genome Sequencer (GS) instrument developed by 454 Life Science in 2005. In this system, the whole genome could be sequenced without any cloning step. After random shearing, specific adaptors are ligated to the two ends of the DNA fragment. This way, DNA molecules captured on the surface of a bead can be amplified by emulsion PCRs and be isolated within a emulsion droplet. The aim of emulsion PCR is to get enough light signal intensity for reliable detection. After PCR amplification and denaturation, each bead with its one amplified fragment is arrayed in a well of a fiber-optic slide. By using DNA polymerase to carry out the primed synthesis, sequencing principle of GS is the same as Sanger's method. When a following base incorporated by the polymerase enzyme in the growing chain, a pyrophosphate group is released. Pyrophosphate group carry the potential to be detected with emitted light. Detection is accomplished with a CCD imager coupled to the fiber-optic array. The major drawback of GS was the relatively high error rates.

The second next-generation sequencing technology was Illumina/Solexa sequencing platform released in 2006. A pivotal difference between Solexa and the 454 is that the former uses Sequencing By Synthesis (SBS) technology. In SBS, florescent labeled nucleotides are used as terminating base. Labeled nucleotides can be removed to leave an unblocked 3' terminus, making chain termination a reversible process. The sequencing processing starts with DNA fragments ligated with specific adapters. After denaturation, the sequencing templates are immobilized at one end on a flow cell surface which is coated thickly with adapters and complementary adapters. Following PCR amplification, each single-strand fragment with one end immobilized on the surface creates a bridge by hybridizing with its free end to the complementary adapter.

After solid-phase amplification, up to a thousand identical copies of each single-stranded DNA template molecule are created on the surface. During each sequencing cycle, a single labeled deoxynucleotide triphosphate is added to the nucleic acid chain. After incorporation, the fluorescent dye is imaged by the CCD camera to identify the base and then enzymatically cleaved to allow incorporation of the next nucleotide. Shorter DNA fragments are obtained compared to the 454 sequencer.

Initial read length was about 30 bp. Sequencing instruments improved and more refined laboratory methods being employed by time. NGS outputs now reached to read length of about a few hundred base pairs. The reads are called short reads because they are shorter than reads produced by earlier Sanger-sequencing based DNA sequencing machines. Within the read, each base call has a probability of error associated with it. Error probability refers to the probability that the base call is incorrect and should be some other base. Along the length of the read, probability of error increases gradually with start and increases more rapidly after the initial high quality part of the read is captured. Early in the development of short read sequencing, the high quality part of the read was typically the first 28 bp. The high quality part of the read is now substantially longer. It continues to increase in length as read lengths increase.

Three platforms for massively parallel DNA sequencing read production are in reasonably widespread use at present. These are the Roche/454, Illumina/Solexa Genome Analyzer

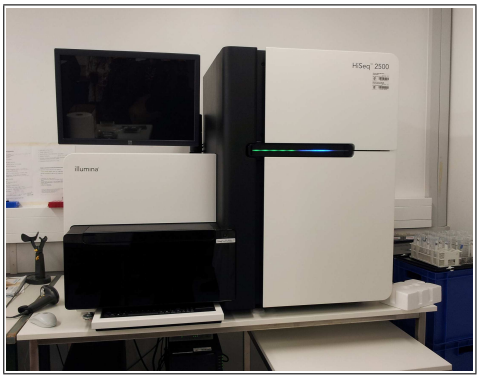


Figure 2.4. Illumina Hiseq 2500 sequencing platform.

and the Applied Biosystems SOLiD System. Recently, two massively parallel systems are on the market. These are the Helicos Heliscope and Pacific Biosciences SMRT sequencer. Popular illumina sequencer Hiseq 2500 can be seen in Fig 2.4.

An important aspect that became available with NGS platforms is the paired-end sequencing. The difference from the classical single read sequencing where sequencing instrument reads from one end of a fragment to the other end. Paired end runs read from one end to the other end, and then start another round of reading from the opposite end. Paired-end DNA sequencing reads whose scheme is illustrated in Fig. 2.5. This gives additional positioning information in the genome and provide superior alignment across DNA regions containing repetitive sequences. They help to produce longer sets of overlapping DNA segments called *contigs*, making it a good choice for *de novo* genome assembly as well as making it easier to resolve structural re-arrangements such as deletions, insertions, and inversions. Experiments designed to study splice variants, epigenetic modifications (methylation) and SNP identification are best served by paired-end runs.

2.2. ALIGNMENT

Discovering the information carried by NGS outputs is a multiple step process, a pipeline whose steps are given in Fig 2.6. Short read sequences have to be arranged and reassembled back to the genome strand for the sake of identifying regions of similarity that are possible consequences of functional, structural, or evolutionary relationships between sequences.

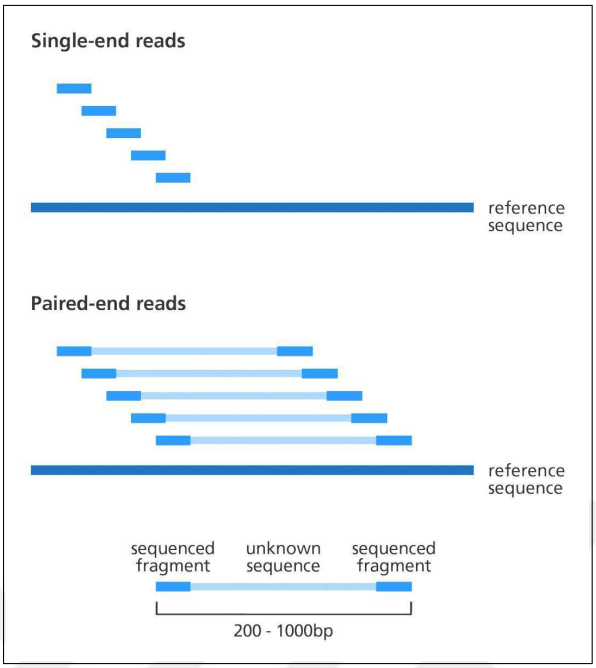


Figure 2.5. Paired-end sequencing provides sequencing of both ends of the DNA fragment. Image credit: Genome Research Limited.

This process is accomplished primarily by two methods: alignment and de Novo. Alignment is the process of rearranging short reads into its 3 billion base pair form by using a string matching technique. An already sequenced and assembled reference genome is compared against short reads.

2.2.1. Alignment Problem

As indicated in the previous sections, the short read alignment problem, also named as genome mapping or read mapping is the problem of finding every position of a reference sequence at which a DNA read pattern appears as a locally-aligned subsequence[16]. Systems which perform this task are called short read aligners or mappers.

The general short read alignment problem is a kind of string matching problem as described at [17] at which it is desired to find the minimum number of steps require to transform a string $r = R_1R_2R_3 \cdots R_N$ into another string $s = S_1S_2S_3 \cdots S_M$. This is equivalent to

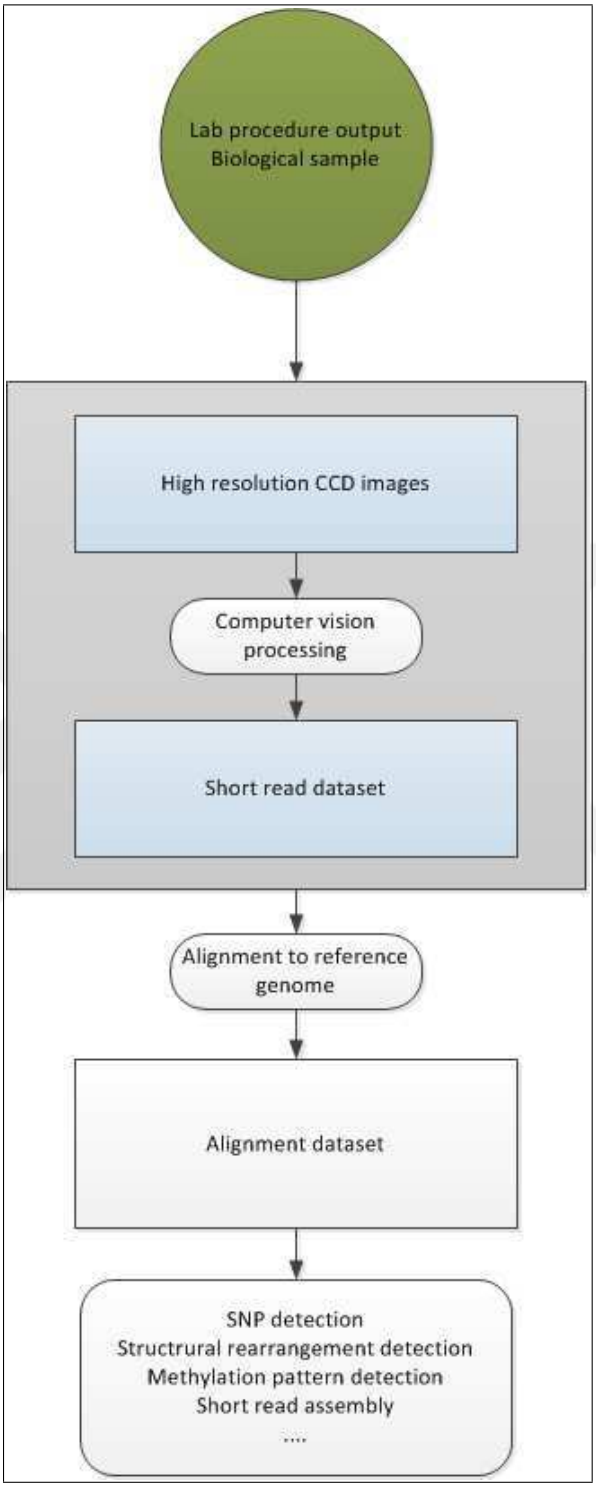


Figure 2.6. Reassembly pipeline stages.

transforming r into s with allowed edits of insertion, deletion, and substitution of characters. A subject of interest is the assessment of similarity between two sequences, number of edits can be defined as a measure of similarity.

As our main interest is on biological sequences, particularly DNA sequences; sequences are of four element alphabet $\Sigma = \{A, T, G, C\}$ with $r_x \in \Sigma$ and $s_y \in \Sigma$ where $x = 1, 2, 3 \cdots, N$ and $y = 1, 2, 3 \cdots, M$. Besides, the alignment idea is the same on another biological sequence type, proteins. Proteins are sequences of twenty element alphabet whose elements are amino acids in which each amino acids is a representation of three nucleotides in a determined order. Protein alignment includes substitution matrices that indicate the possibility of an amino acid being substituted by another one relying on evolutionary similarity.

The problem of aligning two sequences such as a target sequence and a reference sequence has an optimal substructure. The situation is similar to classic example of shortest paths. If you know that the shortest path from A to B goes through C, then you know that the shortest path from A to C is part of the path to A to B. The bigger question can be set aside and main focus can be placed on the smaller problem of getting from A to C, with the intend to be able to construct the solution to the bigger question out of a solution to the smaller problem. If it is already not known which of Bs neighbors the shortest path passes through, it is wise to make a list of the shortest paths from A to all of Bs neighbors. This can be achieved via an exhaustive search to figure out the shortest path from A to B. Pretending to have the knowledge initially that shortest path goes through C, seem to be artificial. It is much less contrived to pretend knowing that the path goes through some point near B. This is the place imaginary list of shortest paths to all the points near B show its usefulness; where paths can be tested one by one via adding those shortest paths with the distance to B in order to determine the one resulting in the shortest distance.

To summarize, dynamic programming is a commonly used technique to solve problems that are composed of multiple overlapping sub-problems which offers the possibility of computing the solution in a reduced number of steps by keeping track of and using solutions to already-computed sub-problems. Dynamic programming is much faster than any naive algorithm and promises highly accurate solutions. But dynamic programming is not fast

enough to use exclusively for alignment in standard CPU based machines.

Developing algorithms which solve the short read alignment problem can make use of the existing string matching algorithm literature. String matching algorithms are old and well established field of study with a very large literature containing a wide variety of algorithms. String matching literature is mainly formed around 1970s and 1980s with important publications including many good textbooks like Crochemore, Hancart, and Lecroq. These offered algorithms typically used in compilers and word processors. In recent years, there has been renewed interest in string matching algorithms due to the vast datasets used in bioinformatics.

Dynamic programming algorithms can be utilized in two category of alignment algorithms as local and global. In global alignment, the optimal alignment is the alignment where the total number of matches is maximal. All characters in both sequences take place in the alignment. Local alignment aims to find related regions in two sequences. The difference between global and local alignments can be explained with an example. Given two sequences to be aligned, ”TGATCG” and ”ATAGACGGCA”, there can be two alignments obtained as

$$\begin{array}{cccccccccc} A & T & A & G & A & C & G & G & C & A \\ & | & & | & | & & & & | & \\ - & T & - & G & A & T & C & G & - & - \end{array} \tag{2.1}$$

which is described as the global alignment and

$$\begin{array}{cccccccccc} A & T & A & G & A & C & G & G & C & A \\ & | & | & & | & | & & & & \\ T & G & A & T & C & G & & & & \end{array} \tag{2.2}$$

which is described as local alignment.

2.2.1.1. Needleman-Wunsch Global Alignment Algorithm

The Needleman-Wunsch algorithm is an application of dynamic programming to the global alignment problem that is mainly the problem of transformation of a sequence A to a sequence B [18]. It was presented in 1970 by Needleman and Wunsch for similarity search between amino acid sequences. Alignment of two sequences with maximum number of matches is returned by building a similarity score matrix as a result of dynamic programming approach. Optimum global alignment is found from the scoring matrix. The algorithm uses a linear gap penalty function representing the cost of inserting a gap in one of the aligning sequences. Linear gap penalty with function with as gap penalty $\mu \geq 0$ is defined as

$$w(k) = \mu k, k \geq 0 \tag{2.3}$$

In order to obtain the local alignment of two sequences $A = \{a_1 a_2 a_3 \cdots a_n\}$ and $B = \{b_1 b_2 b_3 \cdots b_n\}$ the similarity score matrix S is constructed via the initialization equation as

$$S_{i,j} = \begin{cases} w(j) & i = 0, j \geq 0 \\ w(j) & i > 0, j = 0 \end{cases} \tag{2.4}$$

and the recursion equation as

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + s(a_i, b_i) \\ S_{i,j-1} - w(1) \\ S_{i-1,j} - w(1) \end{cases} \tag{2.5}$$

with $s(a_i, b_j)$ defined as the similarity score function with

$$s(a_i, b_j) = \begin{cases} > 0 & a_i = b_j \\ < 0 & a_i \neq b_j \end{cases} \tag{2.6}$$

Optimal global alignment is achieved by tracing back from the maximum value in the last row/column via iterative selection of the highest score of previous rows/columns to the point of origin.

2.2.1.2. Smith-Waterman Local Alignment Algorithm

The Smith-Waterman algorithm (SWA) is a derivative of the Needleman-Wunsch algorithm for computing local alignments which was presented in 1981 by Smith and Waterman [5]. Allowing for the alignment of parts of sequences SWA is the most popular sequence alignment algorithm. This is mainly due to the fact that in global alignment, it is usually

not likely to specify parts of sequences that are similar which is a result of that alignments identifying such similarities will typically be heavily penalized by misalignments of other parts of the sequences.

In this algorithm, a similarity function $s(a_i, b_j)$ is used with a linear gap penalty function $w(k)$ in a similar way in Needleman-Wunsch in order to find the optimal alignment of two sequences A and B. Similarity score matrix S is constructed with the initialization equation

$$S_{i,j} = \begin{cases} 0 \leq i \leq n \\ 0 \leq j \leq m \end{cases} \tag{2.7}$$

and recursion equations

$$S_{i,j} = max \begin{cases} 0 \\ S_{i-1,j-1} + s(a_i, b_i) \\ S_{i,j-1} - w(1) \\ S_{i-1,j} - w(1) \end{cases} \tag{2.8}$$

Optimal local alignment is found by a trace-back procedure. The maximum value in similarity score matrix is determined and traced back to origin from that point picking up the highest value cell to a point of encountering zero. Alignment is obtained via following the path. In the path where each step corresponds to a replacement, insertion or deletion, we insert an element to sequence, a gap to sequence A or a gap to sequence B correspondingly.

Let's assume that two sequences are given as $A = "CCACGCTATA"$ and $B = "CGCACCCT"$. Let the similarity score function $s(a_i, b_j)$ rewards the match with a constant +2 score and penalizes the mismatch with -1 score. The linear gap penalty function

	_	C	C	A	C	G	C	T	A	T	A
_	0	0	0	0	0	0	0	0	0	0	0
C	0										
G	0										
C	0										
A	0										
C	0										
C	0										
C	0										
T	0										

Figure 2.7. SW algorithm similarity score matrix initialization.

	_	C	C	A	C	G	C	T	A	T	A
_	0	0	0	0	0	0	0	0	0	0	0
C	0	2	2	1	2	1	2	1	0	0	0
G	0	1	1	1	1	4	3	2	1	0	0
C	0	2	3	2	3	3	6	5	4	3	2
A	0	1	2	5	4	3	5	5	7	6	5
C	0	2	3	4	7	6	5	4	6	6	5
C	0	2	4	3	6	6	8	7	6	5	5
C	0	2	4	3	5	5	8	7	6	5	4
T	0	1	3	3	4	4	7	10	9	8	7

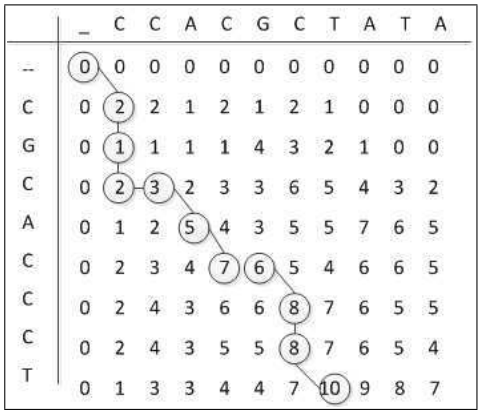
Figure 2.8. SW algorithm similarity score matrix filled.

is defined as $u = 1$. The similarity score matrix S is initialized as in Fig. 2.7.

Forward iteration and filling the similarity matrix is accomplished via the application of Eqn. 2.8. The final matrix is obtained as in Fig. 2.8.

Tracing back from the maximum value of the similarity matrix to origin via selecting the cell with highest score at each step is performed and trace-back path is selected. Tracing back is illustrated in Fig. 2.9.

Along the path, as step diagonally up corresponds to a replacement indicating match/mismatch. A step left towards means a deletion indicating a gap in sequence A and a step upwards means a insertion indicating a gap in sequence B. This all together constructs



	_	C	C	A	C	G	C	T	A	T	A
_	0	0	0	0	0	0	0	0	0	0	0
C	0	2	2	1	2	1	2	1	0	0	0
G	0	1	1	1	1	4	3	2	1	0	0
C	0	2	3	2	3	3	6	5	4	3	2
A	0	1	2	5	4	3	5	5	7	6	5
C	0	2	3	4	7	6	5	4	6	6	5
C	0	2	4	3	6	6	8	7	6	5	5
C	0	2	4	3	5	5	8	7	6	5	4
T	0	1	3	3	4	4	7	10	9	8	7

Figure 2.9. SW algorithm traced back similarity matrix to obtain local alignment.

the alignment in Eqn. 2.9.

$$\begin{array}{ccccccccccc} C & G & - & C & A & - & C & C & C & T \\ & & & | & | & & & & | & \\ - & - & C & C & A & C & G & - & C & A \end{array} \tag{2.9}$$

as optimal alignment result of SWA for sequences A and B.

2.2.1.3. Gotoh Local Alignment Algorithm

Gotoh is an improved version of the SW for aligning biological sequences using an affine gap penalty scheme. It was presented by Gotoh in 1982 [51]. This was due to the need that a mutational event is possible to insert or delete multiple letters at a time; meaning that an alignment containing a gap of a few bases may be more biologically plausible than few gaps of one base. The gap can be defined now as a run of nulls in a sequence aligned with bases in the other sequence. In order incorporate this into alignment algorithm, algorithms need to be modified.

At this scheme, for a cell in the similarity matrix, at a node (i, j), the paths from nodes $(0, j), (1, j) \cdot \cdot, (i-1, j)$ and $(i, 0), (i, 1), \cdot \cdot, (i, j-1)$ besides $(i-1, j-1)$ has to be taken into account. But this normally results in a complexity of $O(mn2)$ for two sequences of length m and n. Fortunately, a relatively simple modification of SWA makes it possible to implement affine gap scheme.

The key point for a node in similarity score matrix (i, j) is to store the best score $H(i, j)$ for entering the node from the left neighbour, the best score $E(i, j)$ for entering the node from the upper neighbour and $F(i, j)$ for entering the node from any path with a modified affine gap score defined as $g(k) = -(a + bk)$ which with $a > 0$ favour alignments with fewer total gaps, not just alignments with a smaller number of insertions and deletions that are commonly called as *indels*. Modified recursion formulas to go through the similarity score matrix can be written as

$$\begin{aligned} H(i,j) &= max\{H(i,j-1) - b, F(i,j-1) - a - b\} \\ E(i,j) &= max\{E(i-1,j) - b, F(i-1,j) - a - b\} \\ F(i,j) &= max\{H(i,j), E(i,j), F(i-1,j-1) + s(x_i, y_j)\} \end{aligned} \tag{2.10}$$

where $s(x_i, y_j)$ is the similarity score function.

2.3. ALIGNMENT TOOLS

As explained previously, the first step in the analysis pipeline of bioinformatics data is the alignment or mapping. Besides, alignment is the most computationally heavy part of the pipeline. Many efforts and different tools have been presented to solve this problem. Most efforts up to date are software programs running on CPU or CPU clusters. Dedicated hardware aligners are also designed and used for alignment purposes. Especially, FPGA

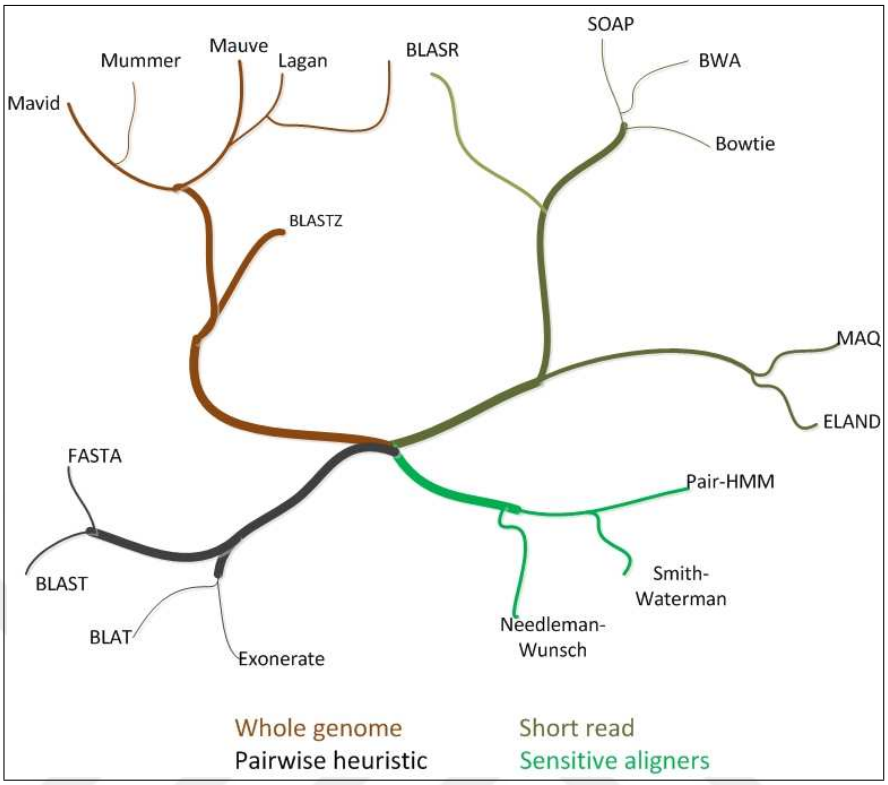


Figure 2.10. Aligner phylogeny.

platforms which provide flexible reconfigurable hardware platforms are popular. GPUs are also valuable platforms for alignment process by means of performing numerous parallel computing similar to FPGA platforms with lower cost. Though there are studies and implementations, most frequently used tools are software aligners; which are represented in a wide range of implementations. Mainstream implementations are shown in Fig. 2.10 with relationships taken into account.

2.3.1. Software Alignment Tools

CPU based alignment typically prefers seed-and-extend techniques at the price of sacrifices in accuracy. Seed-and-extend techniques which are widely employed in aligners, typically do not align the whole read, but just the high quality part of it. The low probability of sequencing errors in the high quality part of the read means that hits generated from the high quality part of the read are less likely to be false positives. Once aligned the high

quality part of the read, one can extend it to check whether the alignment is a real alignment or just a false positive. We sometimes call this high quality part of the read as seed, a term which came from the BLAST algorithm literature, which is basic local alignment search tool [44]. Places where the seed aligns to the reference are called hits.

The short read alignment is implemented in programs running on CPUs are generally presented in three phases as depicted in Fig 2.11. First phase is the ”indexing” phase at which a data structure named ”the index”, containing all the seeds or the whole reference. The second phase is named as ”hit finding”, where the reference genome is scanned against the index. The seed dataset is scanned against the index and all the hits for each seed are found with a method keeping number of false positive hits small. The last phase is ”hit extension”. At this phase, each hit is extended to be examined whether it is a false positive. This procedure brings the concept of error model for the matching algorithm. This is defined as the criteria indicating mapping of a seed pattern to reference sequence is an alignment or hit. Usually, it is allowed one or two mismatches between the seed and reference at an alignment position that is allowing for SNP's and individual base sequencing errors in short read sequences. Several software short-read aligners depending on these concepts were designed to run on microprocessors.

2.3.1.1. Efficient Large-scale Alignment of Nucleotide Databases

Efficient Large-scale Alignment of Nucleotide Databases (ELAND) was the first alignment program coded specifically for alignment of whole-flow cell sized datasets of Illumina reads to a mammalian-sized reference genome [20]. The source code and binary for ELAND are only distributed as part of the Genome Analyzers software package.

Query seeds and the reference genome is represented with 2 bits-per-nucleotide encoding where each nucleotide is simply to be one of the four possible 2-bit words. Which nucleotide is assigned to which 2-bit word does not make any difference as long as assignments are held consistent during whole alignment implementation. In terms of space efficiency, this approach is the most appropriate. It is also suitable for FPGA implementation.

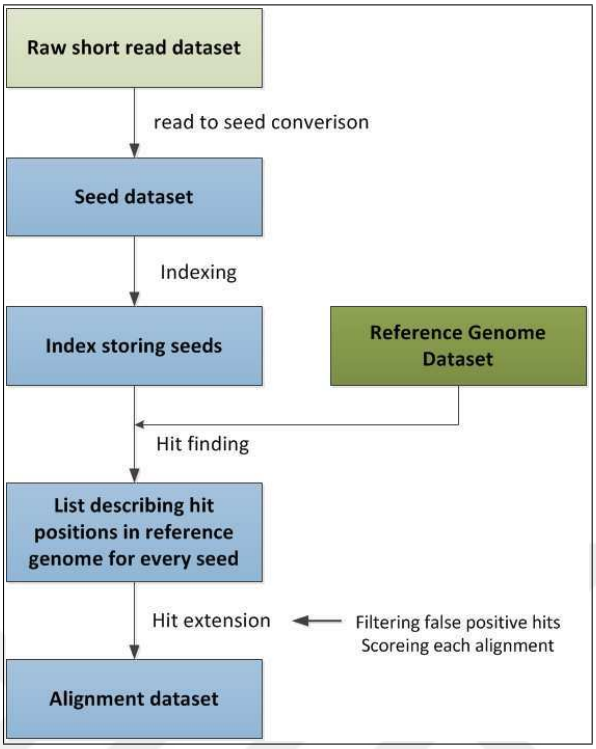


Figure 2.11. Data process pipeline stages of CPU based alignment

In the hit finding phase of the alignment, ELAND generates every subsequence position of the reference sequence which are each looked up in turn in the index to pickup hits. ELAND uses the sliding window technique in order to obtain these positions [22]. Sliding window technique is a well known method in the string matching algorithm literature.

Given a text $T[1 \cdots n]$, the algorithm in Table 2.1 is performed to generate all sub-sequences of length *m* one after another in the character array window $Q[1 \cdots m]$. The iteration step requires only time $O(1)$ as Q is produced from Q by applying a 1-character shift and then a 1-character insert rather than an m-character insert, which would require O(m) time. This process can be visualized as the window $Q[1 \cdots m]$ sliding over the character sequence $T[1 \cdots n]$, hence the name sliding window.

Seed and extend based methods often use seed length of 28 bp, so as ELAND. This is because the per-base probability of error of the position of the read increases rapidly after 28 bp. Longer seeds are being used for longer reads which have a longer high quality segment. When performing hit finding with a sliding window, second part of question to be

Table 2.1. ELAND's sliding window algorithm.

Step	Computation step	Process time
Initialization	$Q[1 \cdots m] := T[1 \cdots m]$; i = m;	$O(m) \simeq O(1); m << n$
Iteration	$Q[1 \cdots m] := Q[2 \cdots m, T[i+1]]; i = i+1;$	$O(1)$
Termination condition	$(i = n)$	266

answered is weather this 28 bp subsequence of the reference is a member of seed dataset. It is able to generate every 28 bp subsequence of the reference sequence via sliding window method for example and check for each one.

Employment of the direct-address table data structure is a way to answer this problem [19]. This data structure has the very desirable property of being adequate enough to provide a solution to the question in $O(1)$ time. The table has 256 slots, numbered 0 to 256 1 each of which is simply a 1-bit boolean value. Depending on that the 56 bit seed, *s*, is in the seed dataset, the corresponding slot is set to TRUE or left as FALSE. It is practical to keep an unsigned integer stored to indicate the number of times a particular seed occurs in the dataset, as each seed sequence can occur more than once in a particular seed dataset. The method is illustrated in Fig 2.12.

This method is feasible for seeds of small lengths. For longer seeds, for example seeds of length 28 bps, 56-bit, it requires tables with 2^{56} slots. That seed length result in a table size requiring 2^{23} 8000000 GB of memory even with slot size of 1 bit. This makes it impractical to implement. Hashing each seed into an integer and use the integer as the key value in place of the raw seed makes the method more practical to use. For an integer of 24 bit number of slots in the hash table reduces to $2^{24} = 16$ MB. This makes it possible for hardware implementation.

In this scheme, collision conditions where multiple different seeds have the same integer hash value and not possible to identify the seed causing collision, is hard to handle. This results in the solution where table is modified to store unique identifiers of seeds in slots instead of a boolean value, which is called as exact match hashing method.

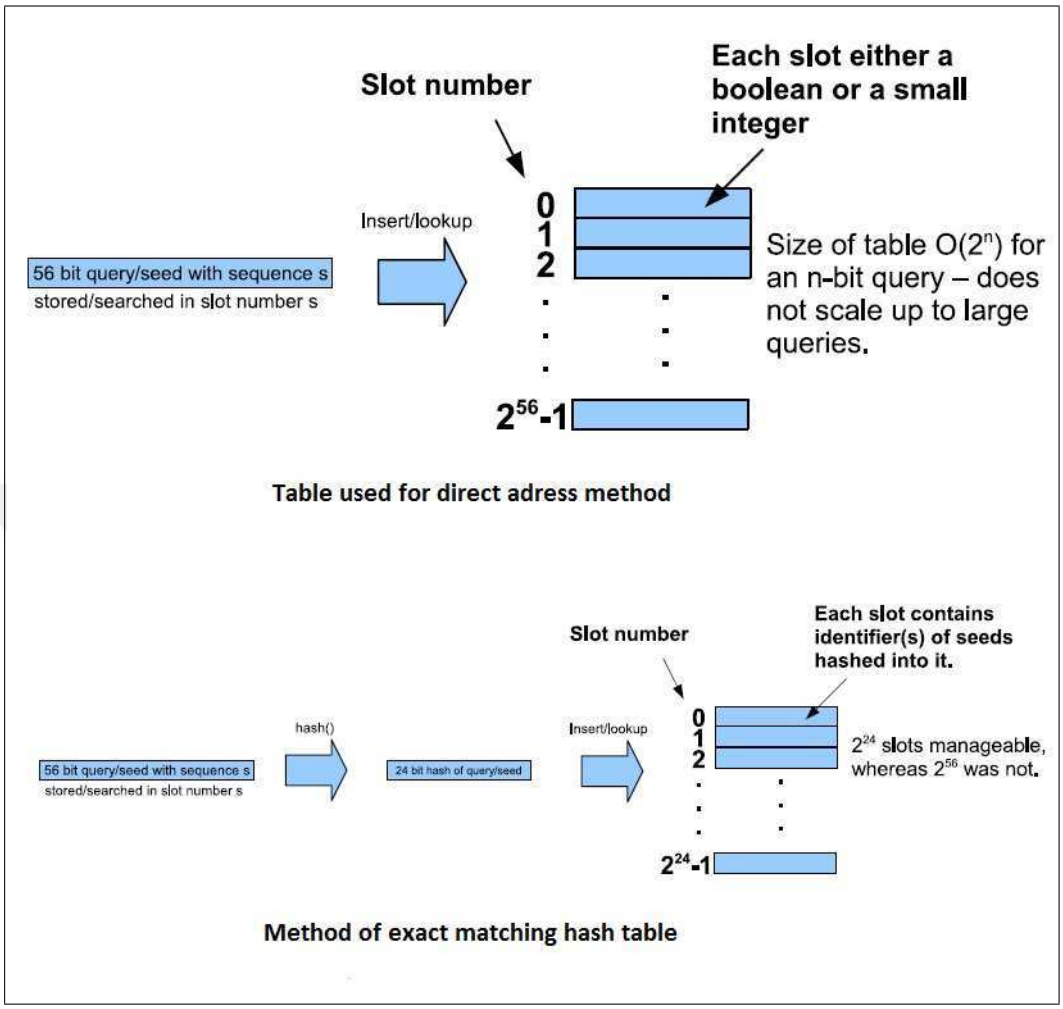


Figure 2.12. ELAND's hash method.

This exact-matching hash-table method is practical to find exact matches. But for alignment purposes, hits with a few mismatches, seed differs one or two bases from reference sequence has to be found. ELAND modifies the hash table to work with inexact matches. Method simply relies on dividing the compared sequences each two to equal length parts and on an observation. That is, given two sequences A, B and each divided to two equal length segments as A1, A2 and B1, B2. If A and B exact match, both segments match meaning at least one match. If A and B are one base different from each other, exactly one of two segments match. Again at least one segment gets matched. If X and Y differ from each other by two or more characters, at least one of two segments mismatches. This means at most one segment matches. From these observations, finding alignment positions where at least on segment of the seed matches the corresponding segment of the query sequence results in discovering exact match hits, all one mismatch hits and some multiple mismatch hits. This 1-of-2 sub-segment matching procedure can be extended to 2-of-4 sub-segment matching. This is the algorithm working in ELAND. This 2-mismatch hash table method is known as *pig-on-hole* principle. It is first implemented in ELAND for alignment purposes.

2.3.1.2. MAQ

MAQ is a free and open source suite of softwares for short-read alignment [57]. MAQ uses the same hit finding method with ELAND. The main difference between MAQ and ELAND is that MAQ generates a mapping quality value for each read aligned. This represents the reliability of read alignment. That is significantly important with its potential to affect the accuracy of the detection of variations. The most convenient way to measure the reliability is to define uniqueness that a read is said to be uniquely mapped if its second best hit contains more mismatches than its best hit. MAQ regards the position that a read is mapped to as a random variable. Reliability of an alignment can be naturally interpreted as the likelihood of the read being mapped to the correct position. It is defined as the *mapping quality* of a read via an estimate of the probability predicting the alignment is a false positive. That is to say that the read not coming from the position to which it has been aligned. Each mapping quality value is a phred-scale probability score that can be computed from a probability value [21]. For a particular short sequence read, consider its best alignment in the genome.

For this alignment, calculate the sum of base quality scores at mismatched bases and define a quantity $SUMBASEQ(best)$. Also, consider all other possible alignments for the read. For the alignment i, define $SUMBASEQ(i)$ as the sum of base quality scores at mismatched bases for that alignment position. Then, the mapping quality is formulated as

$$MAPQ = -log_{10}(1.0 - \frac{10^{-SUMBASEQ(best)}}{\sum_i 10^{-SUMBASEQ(i)}}) \tag{2.11}$$

The probability that the hit is a false positive, meaning its mapping quality, is zero in case that, every character position of the seed matches the corresponding character position of the reference sequence at the position it is aligned. If at the position where the seed is aligned it has one mismatch, mapping quality of the seed is the probability that character position in question is incorrect. That is to say, the probability of error of the read at the mismatching character position. In the situation of multiple mismatching characters, the probability of the read being incorrectly aligned is the product of error probabilities of the mismatching read characters. Here, it is assumed by the program that sequence position of the reference is 100% accurate and only the possibility or errors in reads are questioned.

MAQ is superior to ELAND in terms of incorporating the mapping quality scheme in alignment and handling ambiguous bases in reference genome files. MAQ also has the advantage of pair-end read alignment capability.

2.3.1.3. SOAP

Short Oligonucleotide Alignment Program (SOAP) is another sliding-window method based alignment tool [23]. Its exact algorithm is a little bit different from MAQ and ELAND. It does not hash the reads. But, it indexes and hashes the reference sequence. SOAP is capable of searching for either hits with a maximum number of mismatches, or hits with a single contiguous gap of up to a certain length. SOAP offers a maximum number of two mismatches for searching for hits with mismatches. When searching for hits with a

maximum gap size, SOAP allows a maximum gap sizes of 1 to 3 bps.

2.3.1.4. Bowtie

Bowtie is a fast memory-efficient short read aligner tool. It uses a more sophisticated hit finding algorithm than the sliding window approach. It uses the Burrows-Wheeler Transform which an algorithm originally developed as a compression boosting Method and presented by Adjeroh, Bell, and Mukherjee. Bowtie indexes the reference genome using a scheme based on the Burrows-Wheeler transform (BWT) [3] and the FM index [4].

In order to use in short read alignment, Bowtie modified the common method of Ferragina and Manzini for searching an FM index. This in fact is an exact-matching algorithm to cover sequencing errors and genetic variations. Bowtie extended the algorithm with a quality-aware backtracking algorithm that allows mismatches and favors high-quality alignments. It adds 'double indexing', a strategy to avoid excessive backtracking. The Bowtie aligner follows a policy similar to MAQ's, in that it allows a small number of mismatches within the high-quality end of each read. It places an upper limit on the sum of the quality values at mismatched alignment positions [1].

In the original Bowtie, being an index-aided aligner, it was a very inefficient process to try to perform gapped alignments. Gaps greatly increase the size of the search space and reduce the effectiveness of pruning and slows down aligners relying on index-assisted alignment. Bowtie 2 extended the full-text minute indexbased approach of Bowtie to permit gapped alignment by dividing the algorithm broadly into two stages. The initial state of ungapped seed-finding stage benefits from the speed and memory efficiency of the full-text minute index. The second gapped extension stage employs dynamic programming and has the efficiency benefit of single-instruction multiple-data (SIMD) parallel processing available on modern processors. The combination of full-text minute indexassisted seed alignment and SIMD-accelerated dynamic programming achieves an effective combination of speed, sensitivity, and accuracy [24].

2.3.1.5. BWA

BWA is another popular alignment tool which relies on backward search with BurrowsWheeler Transform [2]. BWA intends to align short sequencing reads, efficiently against a large reference sequence such as the human genome while allowing mismatches and gaps. BWA excepts data in different formats with base space reads and color space reads. BWA claims to be faster than MAQ up to 20 times and achieve a comparable accuracy. In BWA software package, three algorithms are presented. BWA-backtrack algorithm is designed to map Illumina sequence reads up to 100 bps. BWA-SW and BWA-MEM algorithms have longer read mapping capability and faster in mapping time at high-quality queries.

2.3.2. Efforts on Graphical Processing Units

In the literature, studies on graphical processing units (GPU) are popular nowadays. GPU based aligners mainly run the CPU alignment algorithms on graphical processors benefiting their parallel computation ability with low hardware cost. BarraCuda [56], SOAP3 [25], UGENE [26] are some examples of GPU based aligners. Some of those aligners are exact match aligners and some permits gapped alignment schemes. SOAP3 is the first short read alignment program running on GPUs. It uses the Burrows-Wheeler transform compressed full-text index. SOAP3 is reported to be at least 7.5 times faster than BWA and 20 times faster than Bowtie. Besides for alignments up to four mismatches, SOAP is able to align more reads compared to BWA or Bowtie. Therefore, it is reported to be more sensitive.

BarraCuda is another widely used GPU based short read aligner which is mainly a GPU accelerated implementation of Burrows-Wheeler transform/FM-index based short read aligner*BWA*. BarraCuda employs accelerated FM-index inexact alignment and permits gapped alignments with gap opening and extensions, ie implements affine-gap scheme. BarraCuda loads the full BWT suffix array from disk into cached texture memory of GPU [27]. Then, reads to be aligned are streamed in memory batches packed like a single memory block where data is bound to texture cache size in sake of data throughput. BarraCuda is based on a backward search string-matching algorithm to search alignments as for all other

Burrows-Wheeler based aligners. It has a modified and unique version of it.

UEGENE genome aligner is an efficient GPU aligner. EUGENE relies on application of suffix arrays [27]. A suffix array is constructed for reference sequence for the purpose of finding the seed portion of the reads. EUGENE allows up to three mismatch gapped alignments.

2.3.3. Efforts on Reconfigurable Hardware

A reconfigurable computer is a class of computer that includes a special-purpose processor that can perform specific tasks far more efficiently than a general-purpose processor can, but that is also reconfigurable. Field Programmable Gate Array (FPGA) technology enables building custom, special-purpose reprogrammable processors which are able to provide the flexibility of software and the efficiency of Application Specific Integrated Circuits (ASICS). FPGAs in general do not meet current CPU processing speeds in terms of frequency but are capable of performing multiple parallel computations. A typical Xilinx FPGA structure, which is also the FPGA manufacturer of the chip used in implementing our system, is given in Fig 2.13.

FPGA logic blocks are of logic elements and registers. Combinatorial logic is performed by lookup tables (LUTs) which are represented via a truth table. Registers implement sequential logic in order to perform state-holding operations. Programmable logic is achieved by using the LUTs and registers together. Apart from logic blocks, FPGAs contain DSP blocks for efficient multiplication, internal memories for data access, multiple clock speeds for user flexibility, and input/output ports for external communication.

As bioinformatics data is huge, it brings large amount of computation overhead. Due to the property of FPGAs, that they offer parallel and pipelined processing, they are suitable for handling computations in the field of bioinformatics. FPGA also has advantage of lower power consumption compared to CPU clusters which is the most common platform for processing bioinformatics data. One main application on FPGAs designed for bioinformatics is the short read alignment. As techniques relying on indexing and

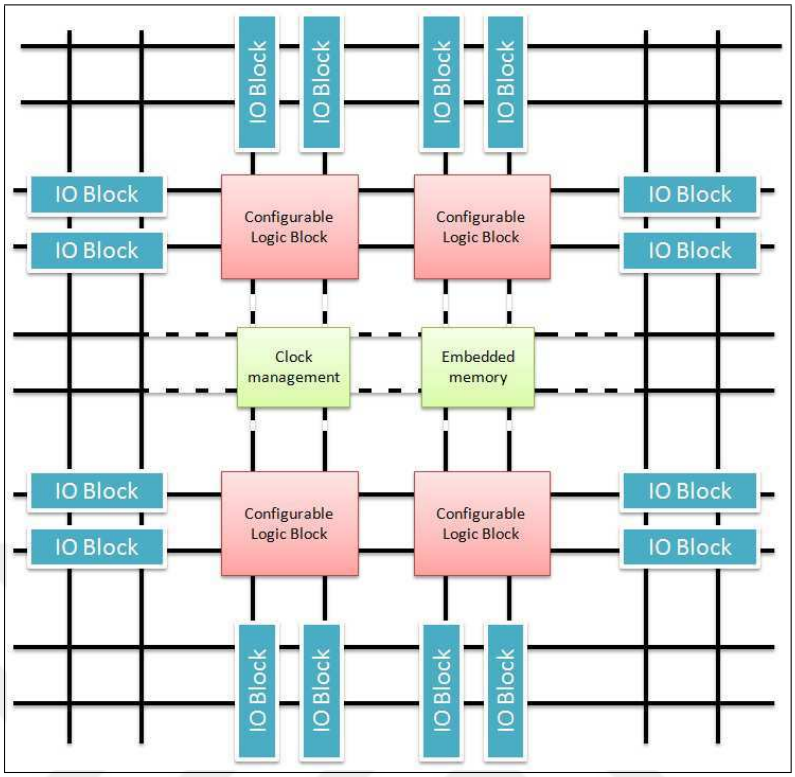


Figure 2.13. FPGA structure.

suffix array perform exact matching very efficiently, there is no need to implement those algorithms in FPGA. Besides those algorithms need significant amount of memory. This makes them impractical for FPGA implementation. FPGAs show their value at the question of dynamic programming when aligning two sequences that differ from each other for multiple points. FPGAs are used both for nucleotide and protein alignment purposes.

There are examples in the literature where the Smith-Waterman and Needleman-Wunsch dynamic programming algorithms have been implemented in reconfigurable hardware. Hoang and Lopresti [28] presented a study where they used systolic array structures and dynamic programming for sequence alignment on reconfigurable logic array structures. Li *et al.* [45] accelerated SW algorithm using FPGAs. Jacobi *et al.* [29] made similar studies. Masuno *et al.* [30] offered a system for multiple alignment of three sequences which is a three dimensional sequence alignment. Many publications describe attempts to accelerate BLAST-type algorithms with FPGA implementations [32, 33, 34]. Xia *et al.* [35] implement an existing RNA secondary structure prediction algorithm in FPGA hardware. Besides there

are systems offered for whole short read alignment. Strengholt [31] offered a short read alignment system implemented on FGPA. They tried and tested various options for hardware platform and communication interfaces with PC host for read and result streaming. Kim and Olson [49] implemented a system on pico computing FPGAs. They combined the index based methods with dynamic programming on FPGA. Hall followed a similar method to Kim and Olson.

Method of combining index based methods and dynamic programming on hardware, significantly improves the overall computation speed. A successful implementation example of this method is Kim and Olsons work which is mainly based on the algorithm by BFAST mapping software [36]. This is similar to the algorithm of software aligner MOSAIK. The algorithm relies on the fact that genome of two individuals of same species slightly differ from each other. Therefore, it is likely that some shorter subsequences of a short read, that are referred as seeds, will exactly match some part of the reference genome of that species. The first step in the algorithm is constructing an index of the reference genome that maps every seed occurring in reference sequence to the location it occurs. Then, aligning a short read begins with looking up all seeds of short read in the reference index. This process yields a set of candidate alignment locations (CALs) for the short read sequence to be mapped. The final step is running the SW algorithm for the short read with the CALs and highest score gives the alignment location for the read.

Kim and Olson uses a reference index of the form modified hash table composed of a pointer table and a CAL table. The pointer table is directly indexed using only a portion of the seed. The CALs for these seeds are stored along with the corresponding seed as a list in CAL table. To look up the CALs for a seed, the pointer table is consulted to find the beginning and end of the relevant bucket in the CAL table. Bucket is searched for the CALs associated with the read. Index is formed via making the list of all seed-location pairs in genome and a pointer table described above is created. As an example for the reference sting ”AGTACTGCGA”, constructed index is given in Fig. 2.14.

The second step in the algorithm is determining CALs for read sequences. They extract seeds of 22 base pairs long from read sequences. Most significant bits of the seeds are used

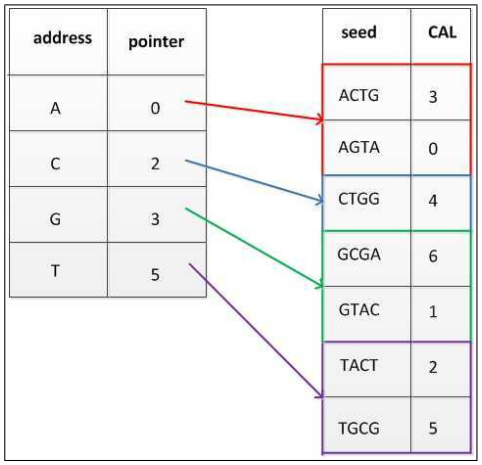


Figure 2.14. The index consists of the pointer table (left) and the CAL table (right). The pointer table is an array of pointers to the start of hash table buckets in the CAL table.

to find CALs and are called as address bits. Remaining bits are named as key bits yielding a pointer to the correct CAL table bucket in the CAL table. Trough a linear scan, they add CALs of entries with key bits pointing to current seed to the set of CALs for that read. An example access to index for a seed is shown in Fig. 2.15. Colored columns of pointer and CAL tables are visited entries.

Obtaining the candidate alignments of seeds in reference sequence, they convert those locations indicating the start position of the seed to the candidate alignment location of the short read containing the seed. This is done via subtracting from the CAL, the offset of the seed with respect to the starting position of the short read. Details of the algorithm up to here is given in [49].

The last step in the algorithm is the SW step which is run on an FPGA platform. In their application, they employ the affine-gap model. They implemented the system on a M503 pico computing FPGA card with DDR3 SODIMMs running at 300 MHz and preferred PCIe interface for communication between the FPGA and host PC. They compared their system with Bowtie and BFAST alignment softwares and reported improved runtime performance and sensitivity.

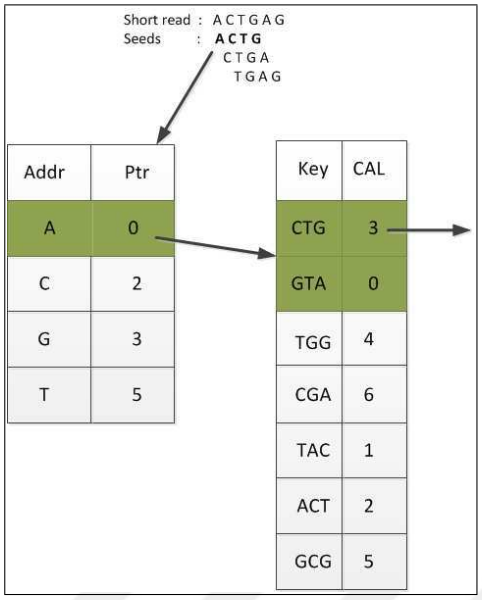


Figure 2.15. Pointer table and CAL table entry example for seed ACTG. Key CTG in the CAL table bucket matches that of seed making 3 a valid CAL for the seed.

3. PROPOSED ALIGNER SYSTEM

The high level design of a typical alignment process flow is presented in Fig 3.1. In this figure, components in dark color are the parts of the alignment pipeline. Light coloured components are part of the system responsible for data flow and timing. They are not direct processes of the pipeline. The reference sequence is stored in DRAM blocks in our implementation. Therefore, the storage block is shown as DRAM. Reference sequence interface is the component responsible for poping the reference sequence to the system. Since the short read data is huge, it is usually stored at PC host directories in FPGA implementations. They are fed to the system via fast data interface like ethernet or PCI express. The controller is responsible for the flow of the whole process. It is also responsible for streaming of the read data and alignment results. Data interfaces are typically hardware FIFOs.

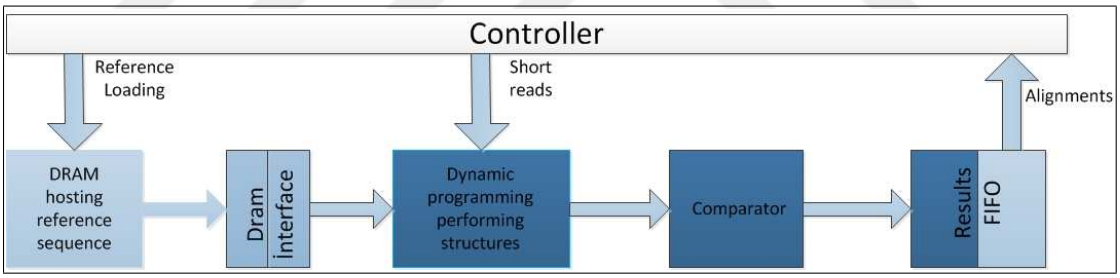


Figure 3.1. High level alignment pipeline design.

In this study, we implemented the aligner system on a PC-FPGA hybrid platform. Some system functions take place on PC. Main processes of the alignment pipeline runs on the hardware FPGA platform. PC part of the system is responsible for hosting the reads, passing them to the alignment pipeline on the FPGA, and compressing the aligned reads. FPGA platform hosts hardware units for performing alignment which incorporate phread quality scores to alignment computations. This provides better mapping performance by means of offering better precision in mapping positions. The controller module is used for controlling data flow, system timing, and synchronization requirements. Proposed read aligner and compressor system architecture is shown at component level in Fig 3.2. In this figure, (1)

stands for read stream from PC to FPGA platform. (2) stands for alignment scores. (3) stands for comparator results. (4) stands for parallel reference data. (5) stands for serial reference data fed to computation arrays. (6) stands for mapping position, score, read id, tags from the comparator. (7) stands for control path to synchronize read stream, reference data stream, and computation triggering. (8) stands for control path to signal result data and compressors.

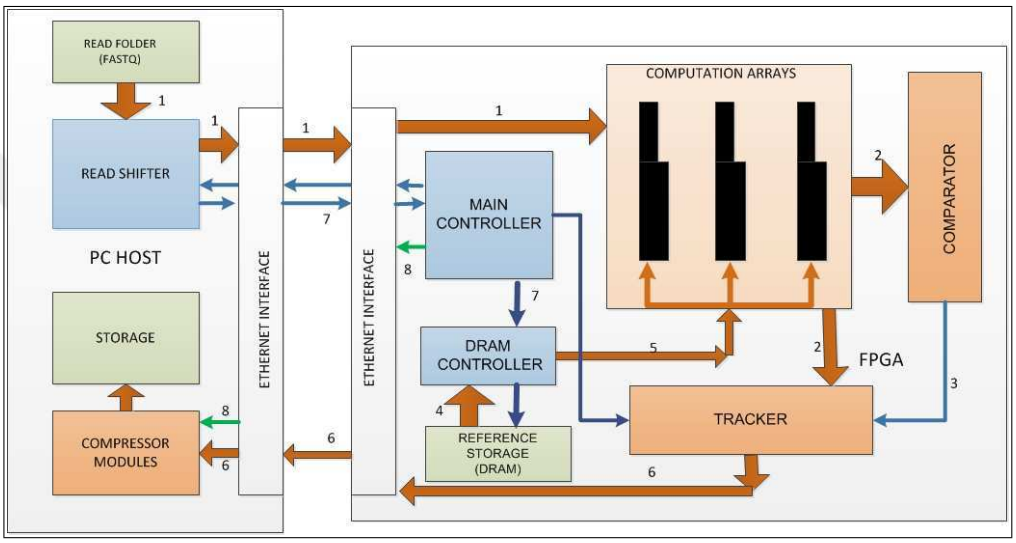


Figure 3.2. Proposed short read aligner system.

3.1. RECONFIGUARABLE HARDWARE SYSTEM

In the proposed short read aligner system, the heart of the FPGA subsystem is composed of computation arrays which perform the SW algorithm. FPGA hosts a main controller for signalling the dataflow over the system and a DRAM controller to access DRAM which holds the reference genome. For data flow, an Ethernet IP core is placed in order to communicate with the host PC. There is a data bus to load the reference genome to the onchip memory of the FPGA board sequentially, a broadcast bus to drive reads to the computation arrays, and a result bus to collect results from systolic arrays to the PC via Ethernet interface. The main controller is responsible for fetching reads from the PC and sending them to computational arrays. The computation arrays run the SW algorithm on the reads up to three hundred base pairs, fill similarity matrices, and obtain alignment

scores. These are then compared to predetermined thresholds to eliminate reads with very low mapping scores and to identify perfect matches. Finally, mapping positions and scores are passed back to the PC together with the *read IDs* and an indicator showing whether the read passes the threshold or not and a perfect match indicator.

3.1.1. FPGA Implementation of SW

Whatever method is applied for read alignment, whether in software, hardware or GPU, the core of the algorithm is dynamic programming. This is responsible for the most computation heavy part of bioinformatics data analysis pipeline. As short read alignment is a local alignment problem, the main concern is the SW algorithm. Therefore, the first thing to accomplish is the implementation of the SWA algorithm on the FPGA platform.

SW algorithm is implemented in FPGA via structures called systolic arrays. Systolic arrays provide systolization of the pairwise distance computation algorithm. These arrays have proven their high efficiency for pairwise sequence alignment using dynamic programming [42]. Systolic array is a homogeneous network of tightly coupled data processing elements (DPEs) called cells or nodes. Each DPE independently computes a partial result as a function of the data received from its upstream neighbours, stores the result within itself and passes it downstream. Systolic arrays are often hard-wired for specific operations, such as multiply and accumulate, to perform massively parallel integration, matrix multiplication, correlation and data sorting tasks.

In our implementation, we call systolic arrays as computation arrays(CA). CAs are read-length long systolic arrays of processing elements (PEs). The SW algorithm is implemented by the CA's in hardware. CA operations construct the similarity matrix in the SW algorithm. The SW algorithm is mainly based on calculating a similarity matrix table and then tracing back for the optimal alignment. To obtain an improved level of similarity between two sequences, the affine gap model based on the SW algorithm is introduced by Gotoh [51]. In this model, a gap is used to compensate for insertion or deletion. This makes the algorithm more condensed in satisfying an expected model. The gap is usually a consecutive null character string in a sequence and it should be as long as possible. In the affine-gap model,

the penalty score for the first gap is called as gap-open. The cost for the following gaps is called as the gap-extend. This is a lower penalty compared to opening a first gap. The affine gap model is described mathematically in initialization as

$$H(i,0) = 0, 0 \leq i \leq n \tag{3.1}$$

$$H(0,j) = 0, 0 \leq j \leq m \tag{3.2}$$

with recursion equations

$$H(i,j) = max \begin{cases} 0 \\ H(i-1,j-1) + \Omega(S[i],T[l]) \\ E(i,j) \\ F(i,j) \end{cases} \tag{3.3}$$

$$E(i,j = max \begin{cases} H(i,j-1) - \alpha, 1 \leq i \leq n \\ E(i,j-1) - \beta, 1 \leq j \leq m \end{cases} \tag{3.4}$$

$$F(i,j = max \begin{cases} H(i-1,j) - \alpha, 1 \leq i \leq n \\ F(i-1,j) - \beta, 1 \leq j \leq m \end{cases} \tag{3.5}$$

where $H(i,j)$ is the similarity of two sequences ending at position i and j of the two sequences S and T. $E(i,j)$ and $F(i,j)$ represent the alignment between a character in one string and a gap in the other string, respectively. α denotes the gap opening, whereas β denotes extending penalties. In dynamic programming, the systolic array is mapped to the anti-diagonal line of the similarity matrix. Hence, every element in the same anti-diagonal

line is computed at once. The mapping of the systolic array to the similarity matrix of dynamic programming is given in Fig 3.3. In this figure, PE stands for a processing element in the systolic array. S and T are the mapped strings and the matrix is the similarity matrix of the SW algorithm.

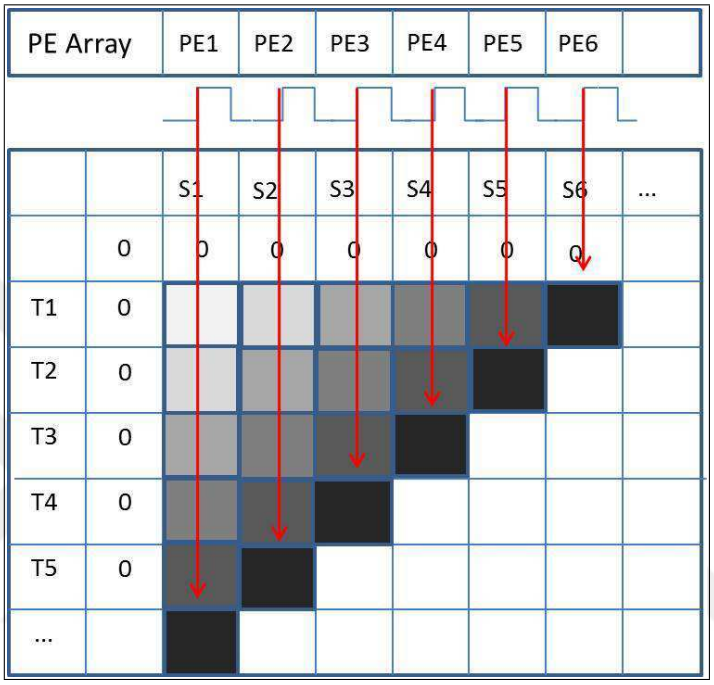


Figure 3.3. Mapping similarity matrix to systolic array.

In Fig 3.3, elements of same color in the similarity matrix on the same anti-diagonal are computed in parallel. This process takes place at every CA clock cycle. Each element in the score matrix has three dependencies as the upper left, left, and the upper neighbour cell score. In design, the systolic cell is compromised of PEs which are connected in series. Each element operates at every CA clock as a shifter [52]. The systolic array for the SW algorithm is given in Fig 3.4. The systolic array enables each base of the short read to be scored against a base of the reference simultaneously. This changes the runtime of the SW computation for an *m* length reference and *n* length short read from $O(m \times n)$ to $O(m+n)$.

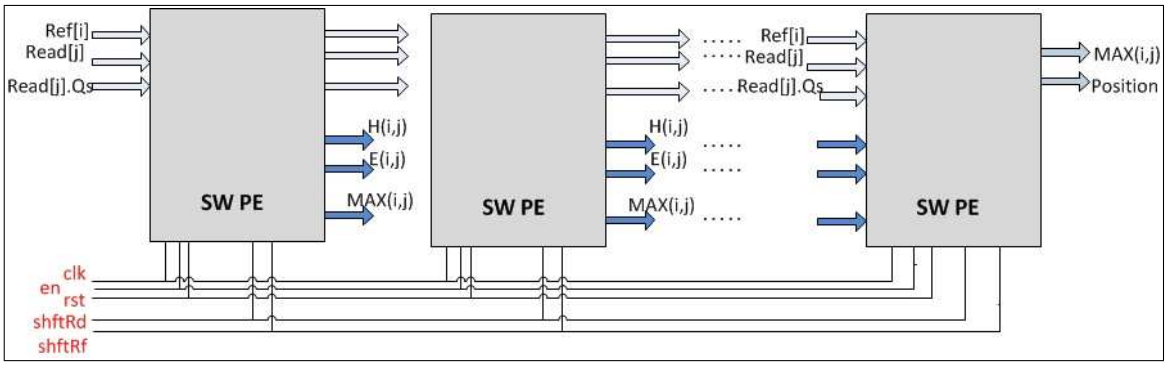


Figure 3.4. Systolic CA structure composed of PE's.

3.1.1.1. Computation Arrays

Computation arrays(CA) are used to compute the dynamic programming algorithm score of the reference and read sequences. Prior to SW calculations, the read sequence is loaded and stored in an internal register. The reference sequence is shifted in each cycle from the DRAM interface as shown in Fig. 3.5. To replicate the dynamic programming algorithms score matrix behaviour, each read base meets with each reference base in the CA as the reference sequence is sliding over the read sequence. As soon as the entire reference sequence shifts through, the computation is finished for the two sequences. A unit at the bottom of the CA outputs the best *Score* and *Position* of the alignment. Besides the dynamic programming score, various flow control signals are generated by the array. These signals interact with the controller so that new sequences can be sent for computation.

Systolic array mapping of SW algorithm was described at beginning of the section. There, SW algorithm with proper cell set allows similarity score matrix cells to be computed in a parallel fashion. As explained in the SW algorithm and systolic array structure, for a single update of the similarity score matrix cell, a value for cell (k, l) for the column l corresponding to the anti-diagonal line in process have to be calculated by the processing engine k of the computation array. In the similarity score matrix, cells on anti-diagonal lines have no dependencies to each other. Therefore, anti-diagonal set of cells can be computed simultaneously and independently which is referred as *wavefront parallelism*.

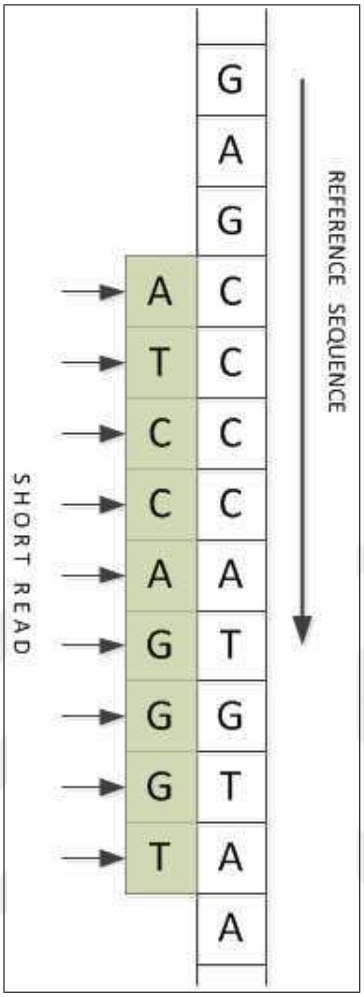


Figure 3.5. SW computation component implemented in a systolic array structure, so called CA.

This technique is introduced by Lipton and Lopresti [28] for constructing custom processor for fast string comparisons.

In the wavefront parallelism of the similarity matrix, considering the anti-diagonal scheme, let a cell of the matrix corresponding to row k and column l be on non-diagonal cell t. This means base-pair k of read faces base-pair l of reference for the non-diagonal t cell. For the update of the cell at row k and column l, values from cells $C(k-1, l-1)$, $C(k-1, l)$ and $C(k, l-1)$ are required. Those are calculated at non-diagonal $t-1$ and $t-2$ updates. Therefore, it is enough to keep cell states in three non-diagonal cells.

As with the SW equations, PE_k responsible for the calculation of the value of cell $C(k, l)$ of column l on non-diagonal t has inputs reference base-pair r_l, short read base-pair s_k, cell values $C(k-1, l-1)$,$C(k-1, l)$ and $C(k, l-1)$ and cell value $C(k, l)$ is computed with recursion equation of SW. Hence, the computation goes as:

(i) Compare the reference and short read base-pairs for penalty/reward $\lambda(S_k, R_l)$
(ii) Compute score from diagonal $C(k-1, l-1)$+ $\lambda(S_k, R_l)$
(iii) Compute score from left with gap penalty $C(k, l-1)$+ α
(iv) Compute score from up with gap penalty $C(k, l-1)$+ β
(v) Compute minimum of above steps

Here, one can calculate steps 1, 3, and 4 simultaneously. For calculating step 2, result of step 1 is required and step 5 follows the result of step 2. This way of calculation makes it impossible to update similarity score matrix at every FPGA clock, but staged calculations. Each PE stores its current value, previous value, the value before that and passes to next PE in array. At every update of CA clock, each PE receives the previous and before previous value of the preceding PE and passes its own to next PE. Besides, sliding reference sequence base-pair is shifted by PE to next PE. These computation steps can be represented in three computation stages relying on their data dependencies as in Fig 3.6.

Computation of the first step, $C(k-1, l-1)$ + $\lambda(S_k, R_l)$, is performed in one clock cycle. Step 2 and 3 calculations require two clock cycles in affine gap scheme but one cycle in

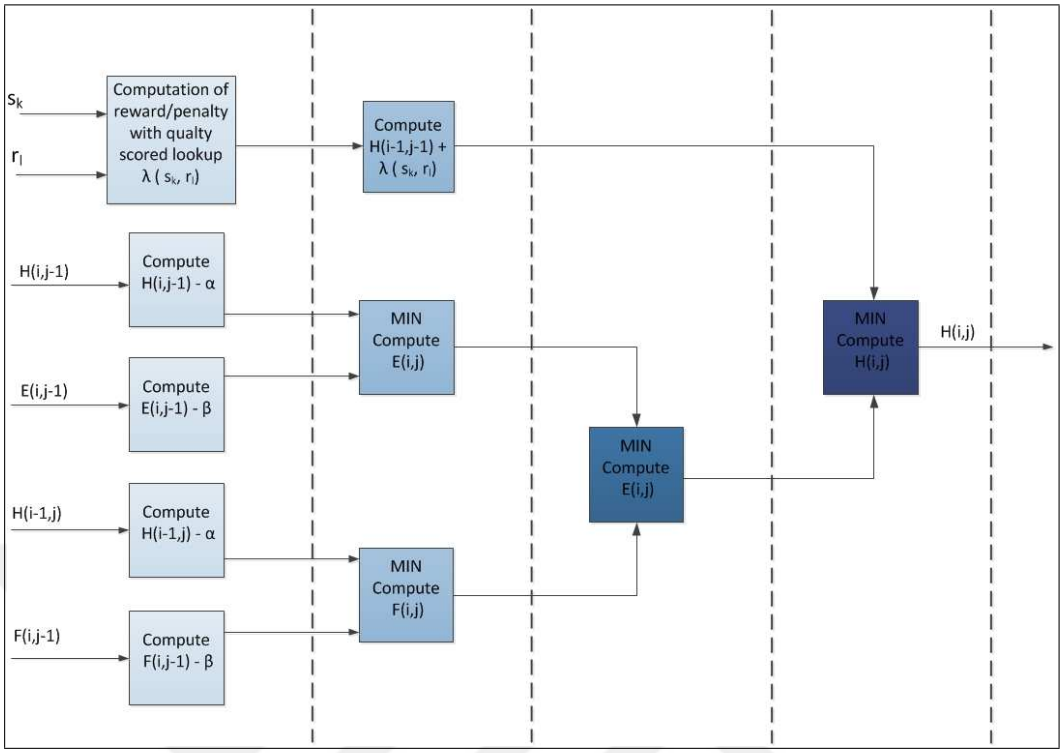


Figure 3.6. Score computation stages of a single PE in systolic array.

regular SW algorithm. Computation of the cell value at step five takes another clock cycle; resulting in the update of a cell on non-diagonal in four clock cycles. Each PE has an enable input for synchronization connected to a clock source of 1/4 frequency of FPGA clock. For the reset of the values at the start of each short read sequence alignment, each PE has reset input line. PE schematic was given in Fig 3.12. Details of the PE operation is given below.

3.1.1.2. Employing Quality Scores on FPGA

Classical SW scheme is fine but it does not seem much appropriate from biological standpoint. In biological sequences we do not observe lots of minor variants. It is more likely that a one big gap of length 10 occurs in a sequence, than 10 small gaps of length 1. Simple gap scoring scheme in which every letter inserted or deleted cost the same amount is not a good model of biology because large gaps tend to occur fairly often. For better handling of that phenomenon, affine gap model which is mentioned in background section is usually preferred. Affine gap model , gap penalties favor longer gaps over single gaps of

the same total length. They use a gap opening penalty greater than gap extension penalty to encourage gap extension rather than gap introduction. affine gap penalties are considered biologically more appropriate than the simple scoring scheme. Affine gap's matrix cells presented in Fig 3.7 are modified version of SW cells. They consider gap extend penalties together with gap open costs which both come from cell neighbours.

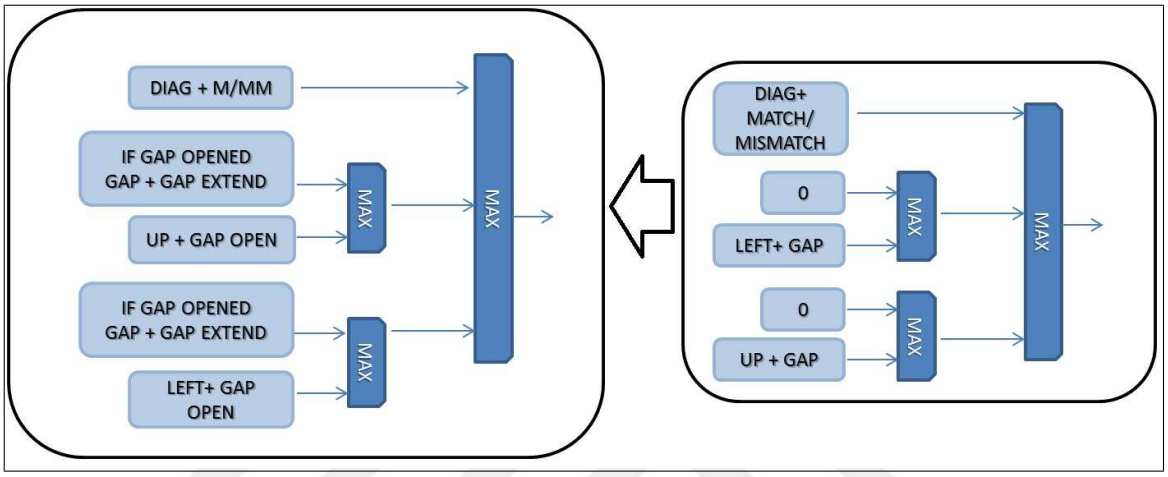


Figure 3.7. Modification of SW cell to affine gap cell for a more appropriate biological model

Quality scores were originally developed by the program $Phred$ to help the automation of DNA sequencing in the Human Genome Project. Phred quality scores are assigned to each base call in automated sequencer traces [21]. Phred quality scores are used as a metric to indicate the reliability of the base read from the sequencing platform. They have become widely accepted to characterize the quality of DNA sequences. Hence, they can be used to compare the efficiency of different sequencing methods. In this study, we exploited systolic arrays of PEs which take Phred quality scores into account during the calculation of the similarity matrix.

The idea was to incorporate the quality scoring scheme to the computation of similarity score matrix and obtain a better alignment of short read sequences with a model whose scheme is given in Fig 3.8. In the standard affine-gap model, reward/penalty values used for match and mismatches are constant. This results from the fact that calculations are done without the knowledge of the identity of the nucleotide. This scheme keeps us away from reaching the

optimum alignment. For example, assume that we will map the read sequence ”AAATTT” to the reference sequence ”AAAAAAGCCGGCCGCTTTTTT” with A bases having low quality scores and T bases having high quality scores. Standard affine-gap algorithm would map the read sequence to ”AAAAAA” and ”TTTTTT” parts of the reference sequence with equal probability. As T bases are more likely to be T, it is logically more meaningful to map the read sequence to ”TTTTTT” part of the reference sequence. Standard affine scheme is not able to handle such a situation.

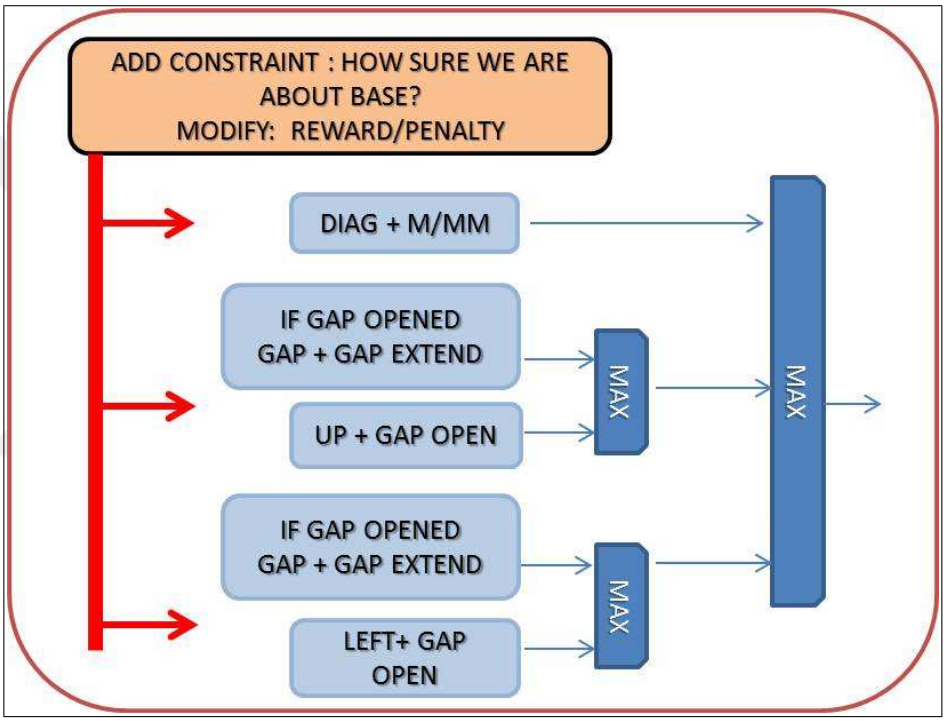


Figure 3.8. Proposed model incorporating the quality score concept for better alignment quality

As an another example one can imagine such a situation. For the sequence ”AAATTT”, A bases and the first T base low quality scores and last two T bases have high quality scores. Think of the situation where this read sequence will map to ”TTAATT” and ”AAATAA” with match score of +2, mismatch penalty of -2, gap opening penalty of -2, and gap extend penalty of -1. At this situation, read will be mapped to second sequence due to that similarity score is higher as in Fig 3.6.

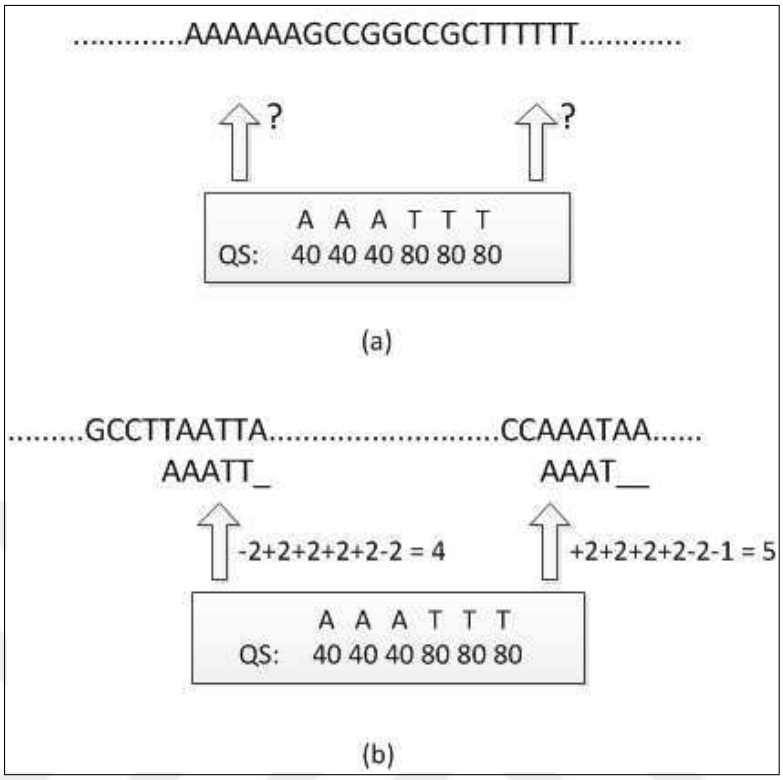


Figure 3.9. (a) Mapping two positions with equal probability. (b) less reliable mapping position with higher similarity score on fixed penalty/reward scheme.

In the proposed system, we do not directly use the quality score itself but a function of it. The quality score range lying between 33 and 83 (represented in six bits) is divided into five levels. The function $FPhQ()$ simply quantizes the quality score value according to those levels producing three bit values. The phred quality score based implementation of the affine gap model (given in Eqn. 3.1-3.5) is re-formulated in initialization equations:

$$H(i,0) = 0, 0 \leq i \leq n \tag{3.6}$$

$$H(0,j) = 0, 0 \leq j \leq m \tag{3.7}$$

and in recursion equations:

$$H(i,j) = max \begin{cases} 0 \\ H(i-1,j-1) + FPhQ(S[i]) * \Omega(S[i],T[l]) \\ E(i,j) \\ F(i,j) \end{cases} \tag{3.8}$$

$$E(i,j = max \begin{cases} H(i,j-1) - \alpha FPhQ(S[i]), 1 \leq i \leq n \\ E(i,j-1) - \beta FPhQ(S[i]), 1 \leq j \leq m \end{cases} \tag{3.9}$$

$$F(i,j = max \begin{cases} H(i-1,j) - \alpha FPhQ(S[i]), 1 \leq i \leq n \\ F(i-1,j) - \beta FPhQ(S[i]), 1 \leq j \leq m \end{cases} \tag{3.10}$$

In this implementation, initialization and recursion formulae are similar to the affine gap model. In the recursion formula, the expression presents the quality score look-up value of the read base in the operation. This is the base of the reference sequence in that step

of the calculation. We provide the internal architecture of the proposed SW with the new PEs based on quality scores in Fig 3.12. The systolic array is composed of N identical PEs connected in series as in Fig. 3.4. N is the read length from sequencing platform. Signals *shiftRd* and *shiftRf* are for driving the read and reference characters to the array. Signal *init* is for telling the PE whether it will participate in calculation or not. Signal *en* is to stop the block operation during loading of the characters and then start. The NGS platform that we use in this study gives reads up to a hundred bases long. Therefore, we implement systolic arrays of hundred PEs originally but reconfigured for different read lengths in experiment section for algorithm comparisons.

3.1.1.3. Atomic Processing Elements of CA using Quality Scores

The CA is composed of multiple PEs as explained previously. Number of processing elements is simply the amount of bases in the short read sequence to be aligned to the reference sequence. Each PE demonstrates an element in the SW similarity or score matrix. It computes a score for one-to-one comparison of short read and reference bases at every CA clock cycle. Each processing receives scores from neighboring units as in similarity matrix calculation of SW, produces its own cell score, and feeds that score to its neighbor PEs. Considering the similarity matrix computation flow, each cell has three neighbours as left, upper, and above/left diagonal cell. That is presented in Fig 3.10.

Each PE has five inputs from neighbouring PEs. Three of the inputs are registered as the values of final score of the PE, gap extended score form upper neighbour, and gap extended score form the left neighbour. Open gap scores are computed from the final scores of neighbouring elements, while the extended scores are computed from the extended scores of neighbouring elements. Diagonal score depends on the current units reference base and read base. A bonus score is given; otherwise, a penalty is taken away from the derived diagonal score coming from the diagonal neighbour. This is accomplished by a match bonus or mismatch penalty depending on the Phred quality score of the read base being aligned. Final score of the PE is the maximum of computed scores which is obtained via comparison.

As can seen in the systolic array structure in Fig. 3.4, PEs are concatenated one after

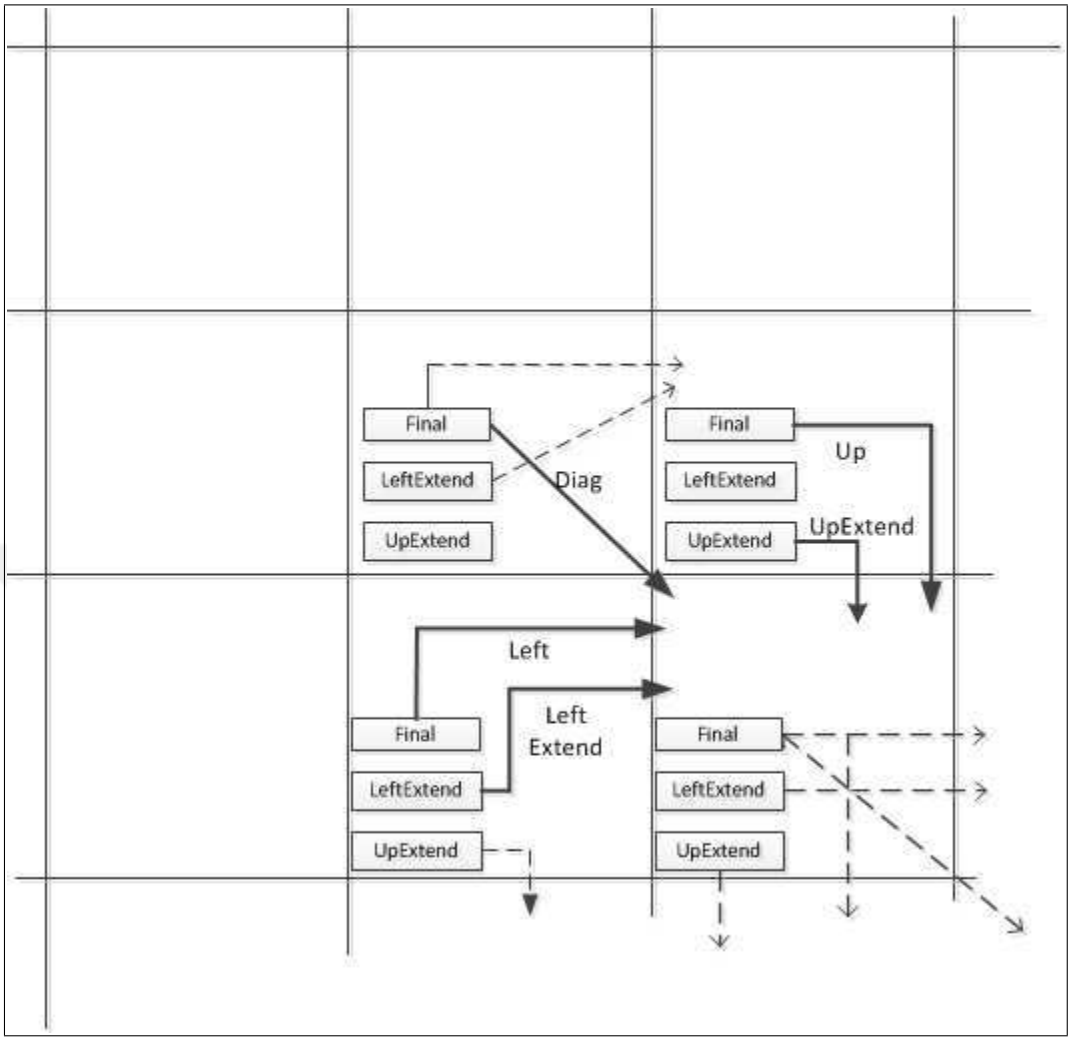


Figure 3.10. Similarity matrix PE neighbours and inputs.

other configuration where each PE has a left and right neighbour. This structure is the one accommodated in all hardware implementations of SW. Structure is functioned via registering the values from left neighbour and implemented as follows. The *Left*, *LeftExtend*, *Up*, and *UpExtend* of each unit is computed in the previous clock cycle. Diagonal value *Diag* is computed and comes from two cycles ago. The unit mentioned calculates *Left* and *LeftExtend* within the unit one cycle earlier. *Up*, *UpExtend*, and *Diag* come from the cell above the unit. Like the score matrix, the computation unit directly receives the *Up* and *UpExtend* scores from the unit above. On the other hand, the *Diag*, *Left*, and *LeftExtend* scores are registered values because a clock cycle delay is equivalent to moving one unit to the left of the matrix which is equivalent to sliding across the reference sequence. Therefore, the old final value is used as *Left*, while the old *LeftExtend* value is used as the input *LeftExtend* value. In addition, the old up value is used as the *Diag* value, which is equivalent to the above left value in the score matrix. Input score connections are shown in Fig. 3.11. Detailed PE architecture which incorporates the Phred Quality Scores into calculations used in our study is given in Fig 3.12.

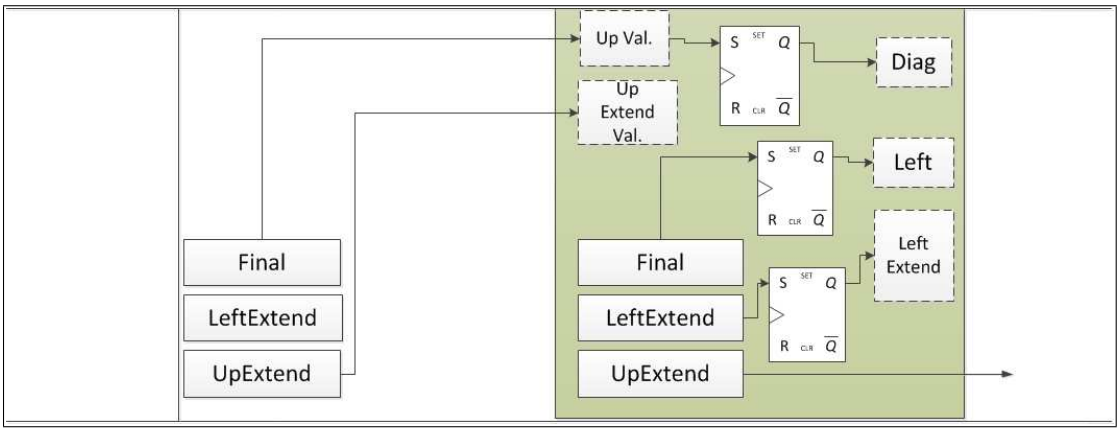


Figure 3.11. Inputs from preceding neighbours. Systolic array implemented version of Fig. 3.10 similarity matrix scheme.

This scheme operates as follows. Initially, the PE has the value of zero for H and MAX to represent Eqns. 3.6 and 3.7. S and T D flip-flops (DFFs) are used to store string sequence elements $S[i]$ and $T[j]$ to be aligned. They are shifted to the next PE in the next clock cycle. The input of DFF *HD* comes from the previous PE as $H(i-1, j)$. It is registered

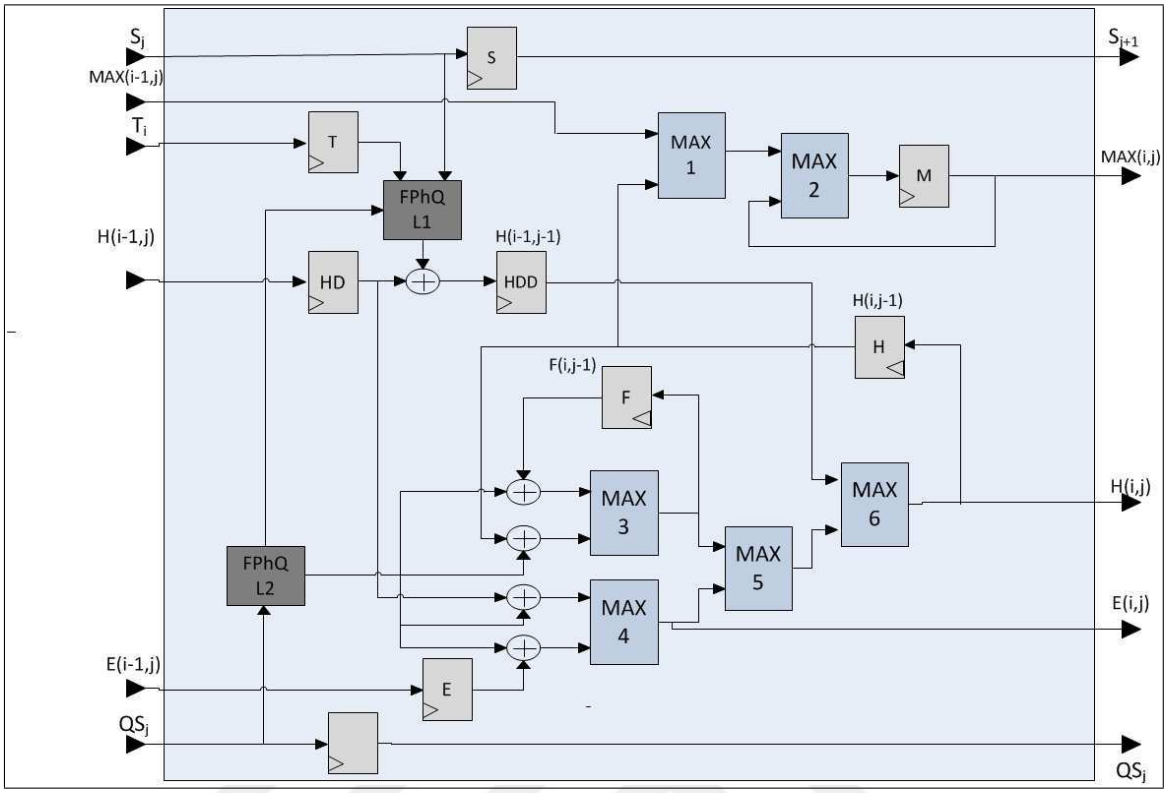


Figure 3.12. Proposed PE block.

for one cycle before being used to represent the value of the upper-left neighbor element $H(i-1, j-1)$. The sixth *MAX* block selects the maximum of H(i-1,j-1) and the fifth *MAX* block which is comparison of F and E values as in Eqn. 3.8. DFF *F* input comes from the same PE generated in the previous clock cycle. It represents the value from the upper neighbor element and is for storing $F(i, j)$ to be used by the same PE in the next clock cycle as $F(i-1, j)$. These are compared to select the max of gap extend or gap opening by the third *MAX* block as in Eqn. 3.10. Similarly, DFF *E* input comes from the previous PE to store $E(i, j)$ to be used by the next PE. It represents the value of left neighbor element. Its gap opening and gap extend score comparison is done by the fourth *MAX* block as in Eqn. 3.9. Look-up Table (LUT) *FPhQ L1* is the implementation of the quality score included comparison for $S[i]$ and $T[j]$ for reward/penalty. *FPhQ L2* is the LUT for the quality score of $S[i]$. Therefore, it determines the reward/penalty score. MAX blocks in the PE structure are simply minimum of two comparators as given in Fig 3.13. FPhQ L1 serves as a comparator and adder block to reward/penalize the comparison based on the quality score. Its architecture is given in Fig 3.14.

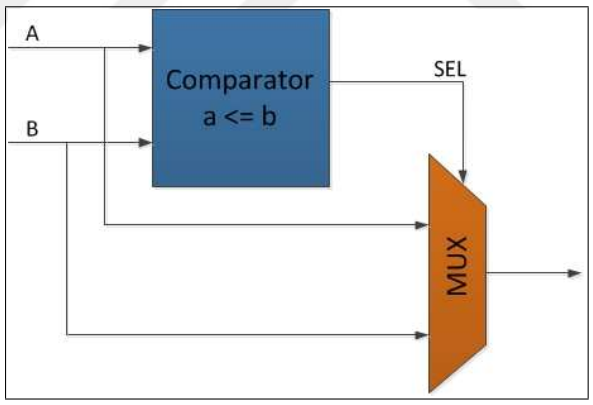


Figure 3.13. PE MAX block.

Human reference genome is composed of three billion base-pairs. With the *2bit* per nucleotide representation, human genome requires 750 MB of memory space. This is obviously too large to hold in the distributed RAM or SRAM of the FPGA block. It is a design choice to store reference sequence in host PC and feed the system via fast communication interface like ethernet and PCI express; or store the reference in a DRAM block. We preferred the second option. CAs receive the stream of reference base-pairs via DRAM interface module.

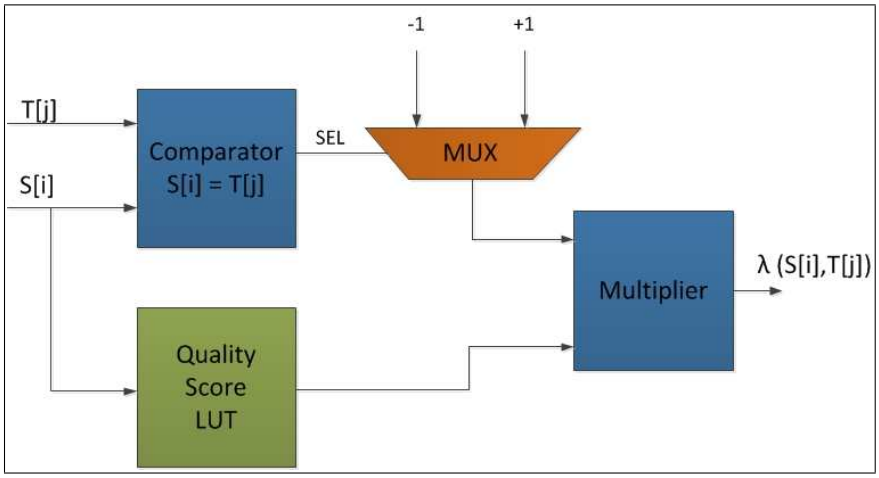


Figure 3.14. Reward/penalty module based on the quality score of the read base-pair.

CA being systolic arrays do not allow gaps in input. Therefore, keeping the reference sequence stream continuous has great importance. CA output goes to a comparator module which compares the alignment score to a predetermined threshold to decide whether alignment does make sense. Alignments with low scores are marked as non-mapped reads. The comparator module attached to the end of computation array in pipeline is given in Fig 3.15. In case of calculated score is above the determined threshold, the result is written to a shared FIFO where it is read by the main controller which is the control unit responsible of data flow, system synchronization and communication interfaces with the host PC.

A new score from the systolic array is computed at every four clock cycles. FIFO write operation has to take place at the time of new score. Counter of comparator module in Fig 3.15 is incremented at every fourth clock cycle. This is responsible for synchronization of FIFO write operation.

3.1.2. DRAM Serializer Interface

Reference sequence of the system is stored in DRAM as explained previously. A DRAM provides a parallel wide data bus which conflicts with the input policy of CAs (having a serial stream of reference base-pairs). This brings the requirement to design the DRAM interface to function as a controller and a serializer. DRAM interfaces main job is to feed

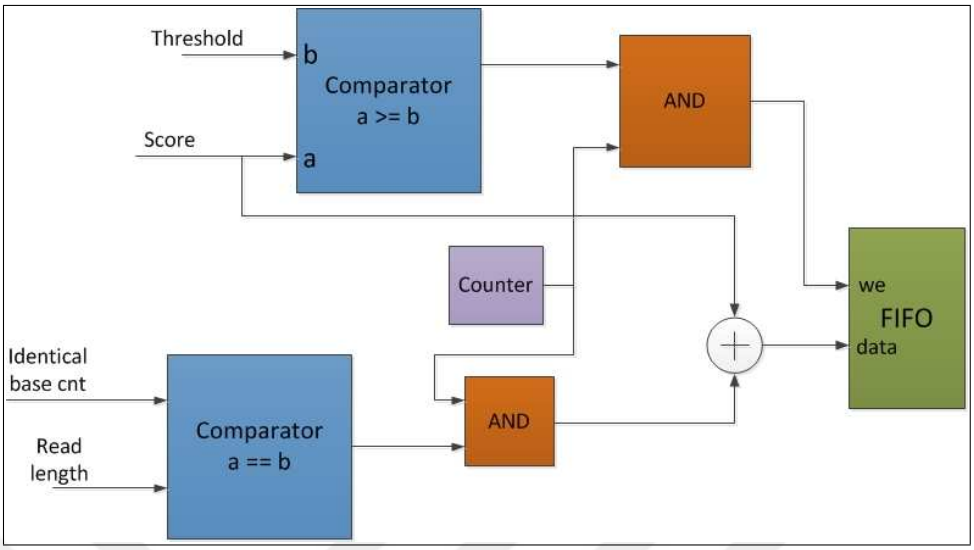


Figure 3.15. Comparator module attached to the end of CAs.

the computation arrays with an uninterrupted stream of two-bit base-pair at update clock of CAs. That is $1/4$ of the FPGA clock. Block diagram of the DRAM interface is shown in Fig 3.16.

DRAM interface is a modified version of the one in [54]. Controller part of the interface is the logic requesting the reference data hold as words in DRAM block. With the request of DRAM word by the controller logic, data is written to dual-port RAMs the first port. To synchronize with the update clock, a clock divider is employed to serve as a clock to a counter cycling addresses on the second port of the dual-port RAM. This double-buffer way is preferred to keep latency of DRAM smaller than required clock cycles and provide a continuous data flow.

3.1.3. Simultaneous Multiple CAs

Creating and simultaneously running multiple independent CAs obviously increases the alignment speed. This method requires the implementation of multiple identical CAs in FPGA logic and synchronization of the data input and output stages. Controlling the reference input is not a big problem as all CAs are fed by the same reference sequence with

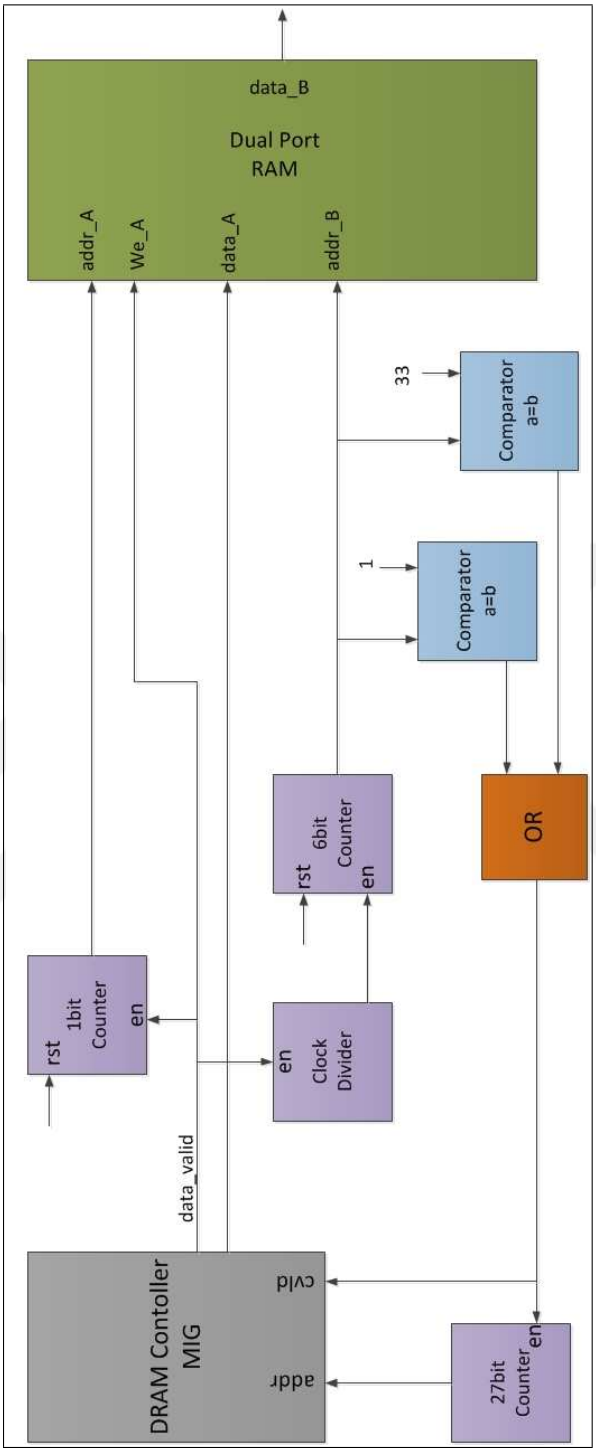


Figure 3.16. DRAM controller and serializer module.

DRAM interface and serializer feeding all the systolic arrays in parallel. Short read input synchronization can be achieved by dividing the short read data into N equal size storage spaces and feed each CA from its own short read storage at the same time. As all CAs are of same length due to the defined short read length and attached to the same update clock, it will be easy to run them in parallel simultaneously where they take the input at same time and give the output at same time. This implementation is shown in Fig 3.17

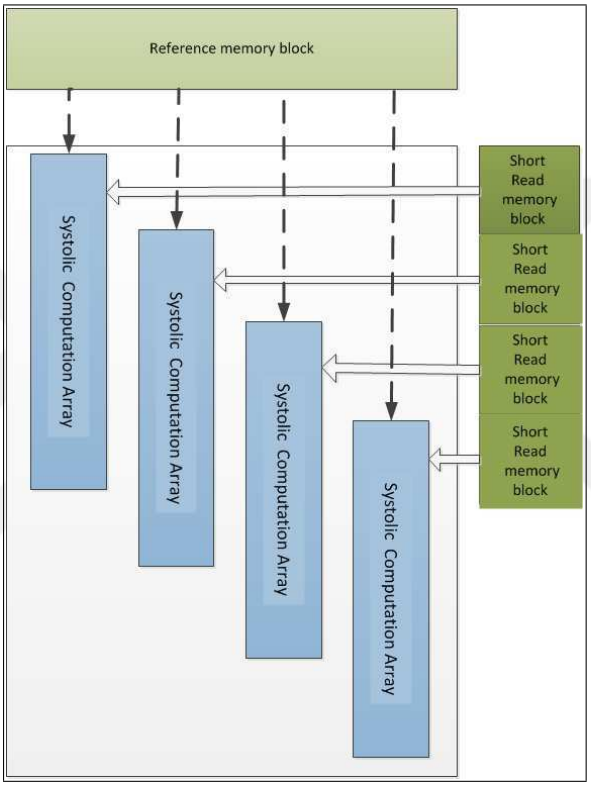


Figure 3.17. Multiple parallel CAs for multiple simultaneous read alignment.

As can be seen from Fig 3.5, each computation array can be thought as a sliding window. Signals with registers holding short read nucleotides in two bits/nucleotide form. Short read sequences are loaded in registers of computation arrays and are aligned to the same reference sequence a the same clock speed. Due to this fact single memory interface is enough to stream the reference sequence in computation arrays. Memory interface simultaneously broadcasts data over an internal bus on FPGA to all functional computation units.

The point multiple implementation differs from the single aligner CA is the output stage. We do not place multiple output FIFOs and comparators into the FPGA. Each CA write its

score to regular FIFO which costs less logic resources. A reader-comparator module cycles through the output FIFOs in round-robin fashion and writes the results to shared FIFO which is accessible from the main controller of the FPGA system. The reader-comparator module is given in Fig 3.18.

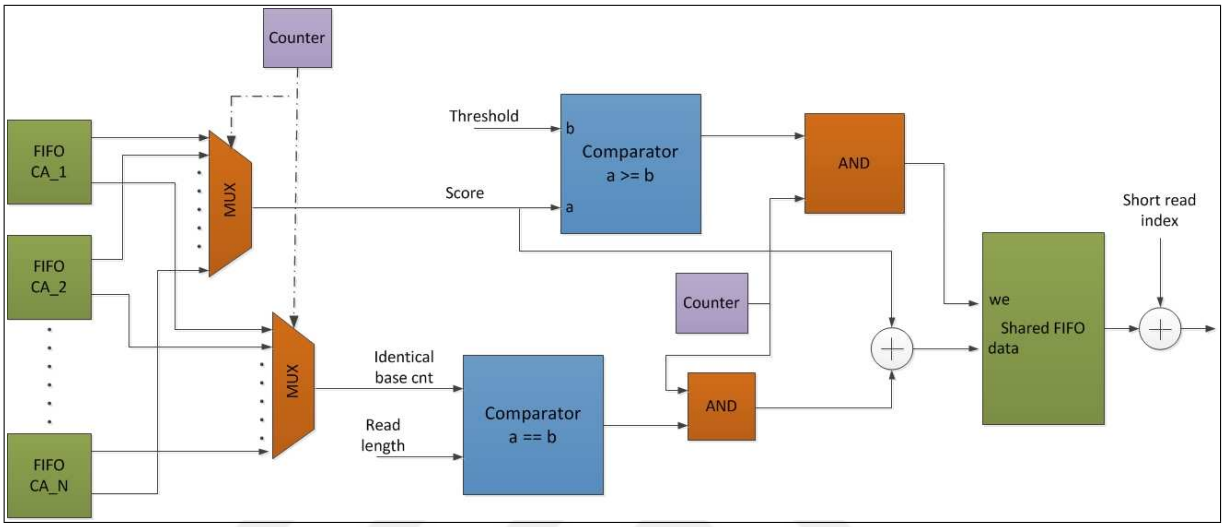


Figure 3.18. Multiple CA shared output comparator and FIFO module based on round-robin scheduling.

Reader-comparator continuously cycle through FIFOs attached to the end of each CA, threshold the value and write it to shared FIFO with two tags of marked as aligned read and being a perfect match. A counter provides the cycle through CA FIFOs with an index value attached to each FIFO. Counter is attached to a multiplexer which to select the FIFO value the counter points to.

3.1.4. Main Controller Module

The main controller is responsible for the flow control, synchronization and host-PC communication interface control of the system. Main controller handles the read sequence streaming from the host-PC and streaming of the result to PC. It also signals for the reference string that streams from DRAM. Main controller handles multiple CAs and the FIFOs to CAs. Main controller also signals the read shifter module to pass read data to CAs and controls result data flow to the PC. Multiple CAs are loaded with read sequences in a Round-

Robin fashion with a counter. Short read sequences are read from the PC memory serially. As the proposed system relies on shifting the entire reference over short read sequences (which takes at least three billion clock cycles), the cost of loading reads to the CAs in a hundred clock cycles has a negligible cost. Therefore, it saves us from implementing a read serializer in hardware. Only the read FIFO is implemented in hardware which is controlled by the controller block. The main controller controls two FIFOs. The output FIFO collecting the mapping score, mapping position, read ID from systolic computation arrays and comparator module, and streams to PC. Finally, the reference FIFO is used to stream the reference data over systolic arrays.

Implementing the controller module in hardware as a state machine is a hard work due to that its fairly complicated to implement. Early versions of our designs accommodated hardware control units and used UART serial communication with the host PC. However, with the use of ethernet interface it makes more sense to implement the controller as an embedded software running on soft-processor in the FPGA. MicroBlaze soft processor core is used for that implementation. Soft-processor provides a library of components including memories, off-chip memory controllers and interfaces. The main control module is responsible for:

(i) Driving the ethernet adapter to transmit packets to the host workstation and receive packets from it.
(ii) Writing the reference sequence data into DRAM at the beginning of the process.
(iii) Writing the short read data into the CA registers in the computation modules.
(iv) Collecting alignment results from the comparator-FIFO module and returning them to the host workstation with the tags.
(v) Overall control of the alignment pipeline.

The controller software is mainly a state machine. States of the system are the $IDLE$ state which it waits a command from the host computer to start; $REFlOAD$ state where the processor loads the reference sequence from the host to the DRAM module; $READLOAD$ state where it drives the short read sequences into the registers of the computation units and finally the $READALIGN$ state where it performs the SW alignment and send the

alignment result to the host PC. State machine diagram of the main controller unit running at soft processor implemented on FPGA is given in Fig. 3.19.

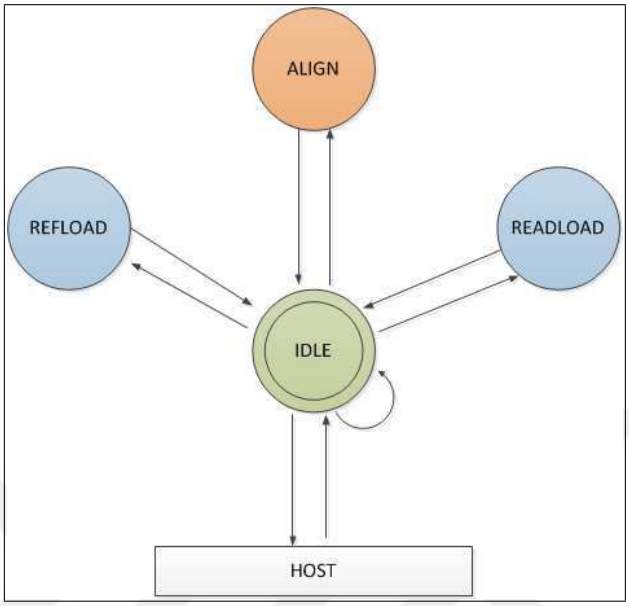


Figure 3.19. State machine diagram of the Main controller.

3.1.5. Ethernet Interface

Ethernet is the communication interface of the FPGA aligner system with the PC-host. Short read data are read from the memory of the PC via internet and alignment results are also reported via that interface. Besides ethernet interface carry control signals for short read input delivery synchronization. Ethernet interface is implemented in logic blocks and controlled by the main controller of the FPGA system. One advantage of internet is that if FPGA aligner is distributed over multiple FPGA boards, it allows an easier communication handling; meaning it is scalable. Some similar systems prefer PCI express but as our systems input flow rate is smaller compared to them, due to the entire shift of reference sequence, we prefer to use ethernet protocol. Also at the PC host side of the system it is easier to observe and process ethernet data. As ethernet interface implementation is beyond the scope of this study, Ethernet implemented is based on 10/100/1000 Mbits/s tri-mode Ethernet core from *opencores* which meets the requirements of our system [59]. In the proposed system, a simple ethernet package setup is enough for implementation. Therefore, it is not required for a complete TCP/IP functionality. This removes the necessity to have an operating system

on the soft-processor.

3.1.6. Clock Domains

CA are the main computation units performing the alignment and they run at 1/4th of the FPGA clock. DRAM controller contains serializer between update clk domain and ram clcok domain. DRAM controller is clocked and requested data when short reads are stored inside registers of CAs. When data is available ready signal is asserted by DRAM controller to signal main controller and CAs. Some clock domains are obtained by clock dividers on design and some are obtained via delay-locked loop (DLL) core. Clocks used by this design are as follows.

(i) 220 MHz FPGA clock
(ii) 55 MHz CA update clock
(iii) 40 Mhz soft processor clock
(iv) 25 Mhz ethernet interface clock
(v) 400 Mhz Dram clock
(vi) 400 Mhz Dram clock delayed

The 400 MHz bus to the DRAM has to be shifted relative to the bus to the DRAM controller to account for the delay on transmitting the signal along the board. A shift has to be applied to the bus to the 167 MHz SSRAM clock for the same reason. The main data bus is clocked at 50 MHz and the ethernet controller at 25 MHz. The DLL produced 55 MHz clock clocks the alignment pipeline generates just one clock. The clock to the alignment pipeline.

3.2. THE PC-HOST SYSTEM

Host-PC mainly plays the role of data storage. Its functional responsibilities are data streaming and compression of bioinformatics data. It hosts read storage areas, read shifter and compressor modules and controls the ethernet port of its side. First of the read storage areas is the memory where short reads sequences outputted from the NGS sequencer are

stored. The reads shifter block is responsible for shifting the reads into the FPGA using the ethernet bus. The compressor block runs reference base compression algorithms on the short read sequences which are mapped to the reference sequence on the FPGA.

3.2.1. Read Shifter

The read shifter module fetches the indexed reads from the short read storage. The read shifter expects ready for load signals from the FPGA controller module.Then, It sends the reads to the CAs of the FPGA module under supervision of the controller module hosted by the FPGA system.

Short reads are stored in FASTQ file format. FASTQ is a text-based format where four lines per sequence are normally used. The first line contains sequence identifier beginning with the '@' character and a description. The second line contains nucleotide sequence letters. The third line starts with the '+' character and optionally contains sequence identifer and description. The fourth line contains quality values for nucleotide sequence in the second line. As FPGA aligner makes computation on two bit values, base-pair characters have to be mapped. Read shifter also performs this mapping operation and shifts the values to the FPGA platform via ethernet interface. Read shifter is coded in Python.

3.2.2. Compressor

Compressor is the unit of the host-PC that is responsible for compression of aligned short reads. From the output of the NGS sequencer, there is no statistical relation between nucleotide sequence and quality score sequence. So text data and quality score sequence of short reads can be compressed separately. Being the bulky part of the data, with six bits representation, quality scores require special attention.

In the standard text format, a file of N short sequences of average length l requires $N(l+1)$ bytes (or $8 \times N(l+1)$ bits) to store, using one ASCII byte per character, and including a character (carriage return) to separate two consecutive sequences. To store and compress this information, short read sequences should be mapped to the reference genome. In this

case, each short sequence can be represented by its address in the reference genome. If the match is not exact, variations from the genomic sequence must also be included by recording their address and type.

The compressor subsystem gets the read IDs and best alignment position of the reads from the read aligner on the FPGA platform. Also, the comparator module on the hardware platform returns an indicator whether the match is perfect or not. This indicator is from the threshold module attached to the back of CAs. Alignment scores above a threshold value are marked as perfect match. This is similar to thresholding and marking reads as unmapped read. At the host PC's compressor block, there is an aligner implemented in software. This runs the SW algorithm on the non-perfectly mapped reads. Target area on reference genome is string of length two times the read length; whose starting position is the alignment location obtained from FPGA aligner for that particular read. This is similar to CAL position alignment of other FPGA aligners or MOSAIK. Information coming from this extra alignment step is only for the compression algorithm we used. For compression, we benefit from the algorithm proposed in [38].

In the compression scheme, the starting position with respect to the reference and strand of the read are stored for perfect matches. The difference between successive positions is stored rather than the absolute value. This means that the positions are relative encoded and stored as a Golomb code. This results in a variable length code. Strand flags require one bit. In case of a non-perfect match, a list of variations are taken into account. Every variation is stored by its position on the read, the variation type (substitution, insertion, deletion), and additional information (the base change in the case of a substitution, the inserted bases, or the length of the deletion) in which positions are relative and Golomb encoded. The variation type (substitution, insertion, or deletion) is encoded in one or two bits. Given the reference base, any substitution to A, C, G, T, or N (other than the reference base) can be encoded in two bits. Inserted bases are encoded in two or three bits (in truncated binary encoding, similar to the variation type). Details of the algorithm can be found at [38].

The massive part of the FASTQ data is quality scores since each quality score is represented with six bits. In the proposed system, quality scores are compressed by a lossless

compression method giving the lowest entropy on quality score data [9]. Prediction by partial matching models are used to predict the next symbol from a varying number of previous characters. In the proposed method, entropy of each quality score is investigated depending on previous scores. It is calculated relying on the information that how previous scores change. This information is kept in binary format such that a one or zero is used depending on whether the quality is the same (different) with the previous one. The entropy coders are compared by their actual and expected compression rates.

It is observed that models with low entropy have lower compression performance than expected values. The reason is that the proportion of escape probabilities of models with higher entropy is much higher than those models with lower entropy. This leads to excessive unused code space and significant reduction of execution speed. Depending on this observation, the previous symbol is used to obtain the best compression result. Besides, it is observed that quality values for a read may be higher, lower or moderate in some parts of a row, with parts of length approximately 20 characters. This leads to a compression technique in which, quality scores are encoded using the previous symbol, location information of the current symbol, the sum of difference between current and previous symbols, and average of last twenty symbols before the current symbol.

4. EXPERIMENTAL RESULTS

In this chapter, we will present the test results of our system. Performance of the system will be evaluated with a few aspects. We will first examine the resource utilization of the system. A very important aspect is the correctness of the system; ie to see alignment algorithm on FPGA coincides with the same algorithm running on CPU. We will present the sensitivity and accuracy of the system. Alignment time is another topic that will be evaluated. Besides we will talk about the scaling ability of the system. The system will be compared to state-of-the art CPU software aligners and with an another FPGA based aligner system.

The proposed short read aligner system is built on a Xilinx Virtex4 XC4VLX80 FPGA board with DDR3 SODIMMs. Xilinx ISE 11.1 tool chain is used for system implementation and resource evaluation. FPGA platform operates in cooperation with a PC with two Quadcore Intel CPUs with clock speed of 3.2GHz and 16 GB of memory. The interface between the host PC and the FPGA board is ethernet.

In terms of resource utilization, CA and comparator modules attached to the end of those structures are costliest parts of the design. As there is a linear relation between the number of computation units and alignment speed; design of those units is particularly important. First parameter that makes significant effect on the design is the length of short read sequences. CAs contain as much PEs as the number of elements in the short read sequences. Clearly as read length increase, number of units in a single FPGA decreases. At first glance, this may be considered as a decrease in alignment time. But as read length increases, total number of reads outputted by NGS sequencers decrease. Besides the number of PEs, the design style itself affects the system runtime. Clock frequency of the computation process in FPGA can be increased by reducing the comparison steps for example. But that kind of design costs more FPGA resources and decreases the effective number of computation arrays in the system. Table 4.1 shows the number of systolic arrays that can be fit in a single FPGA scaling with the number of PEs, i.e length of short read sequences to be mapped.

Our system was originally configured to use hundred base-pair short read sequences. This is

Table 4.1. Dependence of resource utilization on the length of read sequences to be aligned.

Read Length	**Number of CAs fitted**	**Total PEs**
28	12	336
76	4	304
100	3	300
300	1	300

mainly due to that, real data used to evaluate the system during development stage obtained from the Illumina sequencers operating in TUBITAK BILGEM, specifically run by our study group IGBAM. Second costliest part in terms of resource utilization is the main controller which is a soft-processor implemented in FPGA. Soft-processor is implemented simply as a main CPU without peripherals. It runs on a slow system clock enough to run its tasks. The logic resources used to implement a single PE and the entire design is summarized in Table 4.2.

Table 4.2. FPGA resource utilization of hardware modules.

	4 input LUT	**FlipFlop**	**BRAM**
PE	207	179	9Kb
100PE CA with IDR	20817	18025	47x18Kb
Ethernet MAC	1476	1610	-
Main Controller	2932	1219	4x18Kb
DRAM Serializer	376	117	1x18Kb

With the resources available on the board, three systolic arrays (CAs of hundred PEs) can be implemented at most. This means that, three short read sequences are mapped simultaneously to the reference genome. Systolic arrays are implemented as realizing the method using the *Phred quality scores*. CAs are running at 55 MHz. One systolic array utilizes a resource 9710 of 80640 slice registers and 21322 Slice LUTs. The slices are the limiting factor as they limit the number of CAs that can be placed onto the FPGA chip. For data and control signal flow between PC and FPGA, we used a free *10/100/1000 Mbits/s* tri-mode Ethernet core from *opencores*. This Ethernet core utilizes 1476 LUTs.

In terms of functional requirements, a very important aspect to evaluate is the correctness of the system. In order to test the correctness of the proposed system, we followed the same method defined in [54]. To do so, we loaded a hundred million base-pair *Drosophila melanogaster* reference sequence on DRAM of the FPGA board. FPGA system is the configured to align 31 base-pair length reads. This is because short read data set provided for Drosophila was of reads 31 bps. We carried out alignment against thousand different short read sequences. Comparing the results with a conventional CPU implementation written in C++, we observed that FPGA and CPU results exactly match. Correctness of the system is also checked with a negative control test. Hundred thousand 31bp short read sequences of *Drosophila melanogaster* are aligned to Human Hg19 reference genome. As a result, it is observed that 43.6% of total reads are aligned to human genome. Only 6.71% of the alignments were perfect match. In order to test with longer read length, Hundred thousand 100bp long reads are simulated from *Drosophila melanogaster*. Those reads are again aligned to human genome. This time, 21.2% of the reads were aligned and 0.089% of the alignments were perfect match.

In order to see the effect of quality score based cost scheme, we conducted an experiment on simulated data. Illumina single-end reads of 100bp length containing 3,5 and 10 bp INDEL events from Chromosome 20 of Human Hg19 genome are simulated with MUTATRIX genome simulator. A run of alignment is performed with the proposed algorithm where costs are based on base quality scores. Another alignment run is conducted with setting all rewards and penalties to zero. Accuracy of the runs are compared. An alignment is considered correct when it is mapped to the correct position which we know from read simulation stage. Results are presented in Fig. 4.1. Quality score based scheme is observed to be much superior over no cost scheme.

In terms of sensitivity, we put an effort on testing the system not just with Illumina dataset but with various NGS platform outputs. The biggest problem here is NGS sequencers on the market employ different techniques for library preparation, different protocols for paired-end read and DNA sequencing. To do so leads to a range of read lengths, base quality assignments and error profiles. Besides, different technologies present different output data formats. Ion torrent sequencers output short read data in *fasta* format. Roche GS

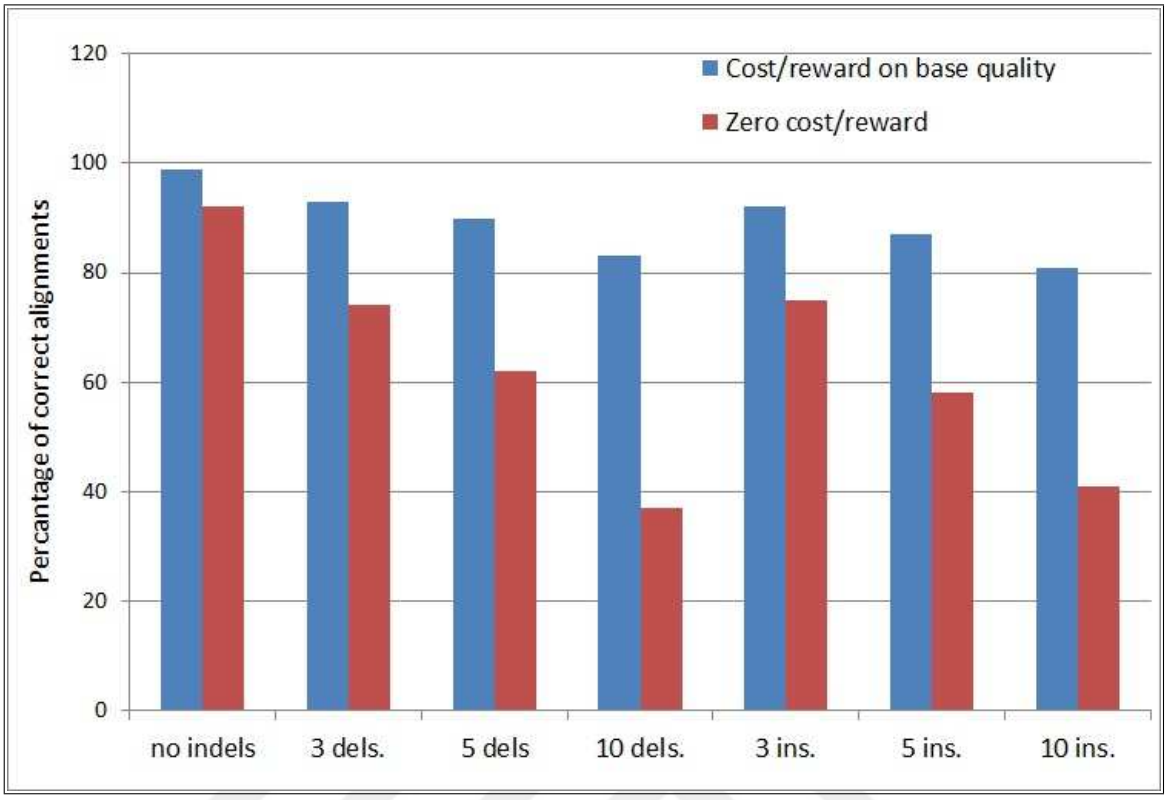


Figure 4.1. Effect of Quality score based cost calculation on alignment accuracy. Simulated reads of 100bp are aligned to Chromosome 20 of Human Hg19 reference genome. Alignments are performed with various INDEL events. Percentage of correct alignments with different INDEL lengths are shown on graphic. Dels and ins stand for deletions and insertions respectively. Quality score based scheme is observed to be much superior over no cost scheme.

FLX/454 platform's data format is *.fna* which is similar to *fasta* format. In order to test data from different sequencers, all formats are converted to *FASTQ* file format which our system accept as input. Fasta file formats are converted using *SAMtools* software. Roche 454 outputs are converted via *Biopython*. Illumina data used is real sequencing data from IGBAM, TUBITAK BILGEM. Roche 454 data if synthetically created using the MASON read simulator [61]. A test is also run via synthetic Illumina data produced by the same tool. For experimenting with ion torrent data, *E. coli* reads provided by Ion Torrent [62] are used. Resultant alignment rates are given in Table 4.3.

Table 4.3. Dependence of resource utilization on the length of read sequences to be aligned.

Technology	**Sensitivity %**	**Read length**	**Reference**	**Dataset**
Illumina SE	86.65	100	Human hg19	IGBAM Sequenced
Illumina SE	95.90	100	Human hg19	Synthetic
454 SE	93.85	266	Human hg19	Synthetic
Ion Torrent	87.58	298	E. coli strain 536	Ion torrent released

Sensitivity comparison of the proposed system is conducted with BOWTIE-2.0-beta5 CPU based aligner and with Kim-Olson's FPGA based solution [49]. Bowtie is configured and tested using eight threads. We prefer this because we also compare with Bowtie in terms alignment time for which eight thread scheme leads to saturation of the memory bandwidth of the system, giving optimum performance in terms of mapping speed. For the comparison run, two million short read sequences of 76 base pairs obtained from MASON read simulator is used. Reads contain SNPs and indels, as well as sequencing errors distributed according to typical non-uniform error profiles of Illumina sequencer. Proposed system is configured to operate 76 PE computation arrays. At the end of the experiment, we observed that our proposed system mapped 94.3% of one million reads. Bowtie managed to align 87.6% using the best option mode. For the test of the other FPGA based solution, PC part of the system is obtained from the author and FPGA part is implemented in VHDL on our system FPGA card. On the same data set, the mapping obtained with this system is 91.7%. This is still lower than the sensitivity value obtained by our system. Fig. 4.2 summarizes the sensitivity results. The main reason for sensitivity is clearly the search of all possibilities both on streaming reference and SW score matrix computations with quality scores. Both

algorithms first search for candidate locations and run SW on those candidate areas. Kim and Olson does not consider base quality scores and keep concern mainly in reducing the total time of alignment.

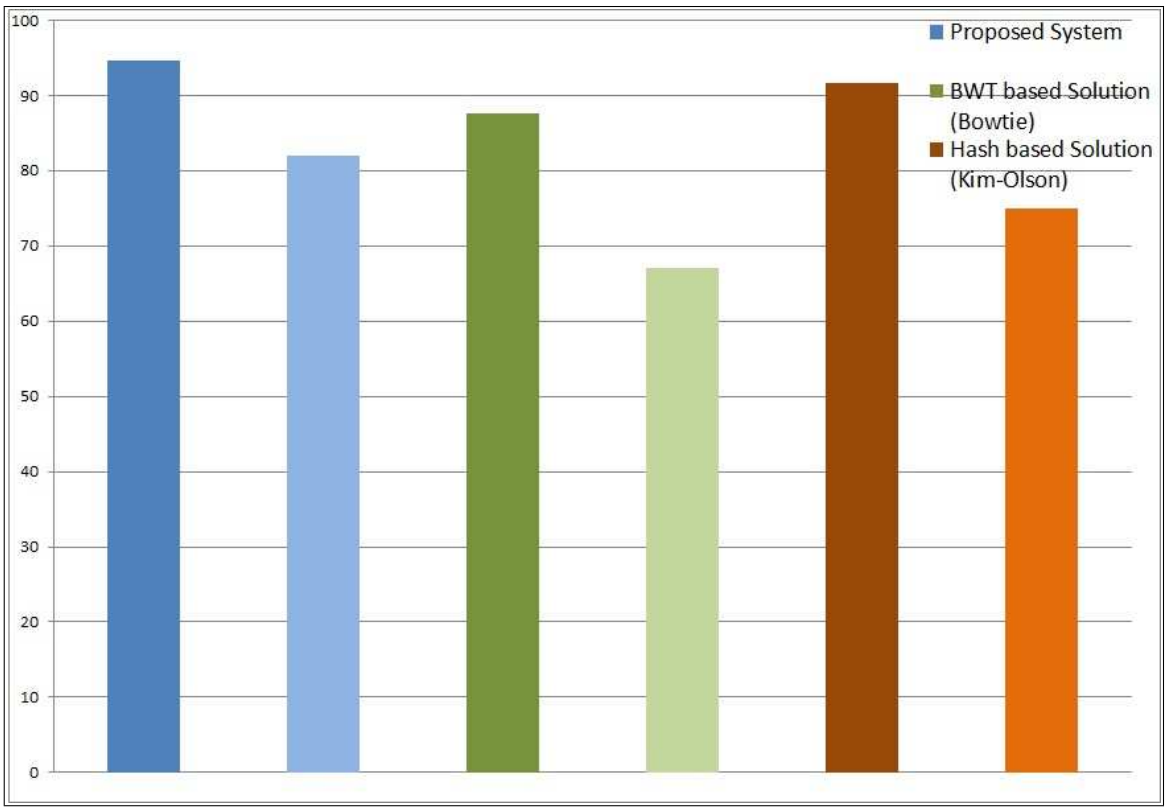


Figure 4.2. Sensitivity and Accuracy of the methods evaluated with small synthetic read data set aligned to Human Hg19 reference genome. Each method is represented by 2 tone of a color. Dark color tone shows sensitivity and light color tone represents accuracy evaluation for the method. Proposed system is compared to a cutting edge BWT based solution and to a hash based algorithm.

Unique feature of the proposed system is its usage of the *Phred Quality Scores* in FPGA aligner system. This scheme increases the ability of the system to make more accurate decisions based on the optimal solution of the alignment problem [53]. To evaluate accuracy or mapping quality, we conducted a two step accuracy testing protocol. For the first step we created hundred synthetic reads as described in [53] and followed the method explained in that study. Read lengths were chosen 76 base-pairs and our system is configured to operate with 76 PE length CAs. The reason of this choice is to make the test compatible with the

other FPGA system whose tests were performed with 76 bp reads in its original paper. We tested the alignment of those reads using Bowtie, Kim and Olson's system, and our system. Our system was able to map 82 of the reads to the correct positions. This mapping dropped down to 75 in Kim-Olson's system. Bowtie was able to map only 55 reads. These results are also presented in Fig. 4.2. Therefore we may claim that quality scores improves mapping quality. It can be argued that the reads synthetically created at this particular experiment is hand crafted and carry ambiguity. But it is practically impossible to find out that kind of reads from real data or simulate from well known read simulators and search for in simulated data.

Second step in testing accuracy is accomplished with using high amount of read data which is more concordant to the nature of the process. Here, a total of two million Illumina single-end reads of length 100 base-pairs from chromosome 20 of the Hg19 human genome are simulated using the MASON read simulator with a haplotype SNP rate of 0.1 %. Reason for choosing MUTATRIX is that it features position specific error rates and base quality values. The reads are aligned against the entire human genome with BOWTIE-2.0-beta5, BWA-0.5.9 aligner and our proposed system. Then we calculated the positive predictive value of each aligner by dividing the number of correctly placed reads, that genomic coordinate of the mapped read agreed with the known location of the read from MASON simulator, with total number of aligned reads. This is the method similar to the one followed by another aligner study, which uses quality scores, named *MOSAIK* [55]. A correct alignment is considered as the one staying in a tolerant window of 20 base-pairs as in [55]. Fig. 4.3 shows the positive predictive value of the tested aligners as a function of mapping quality cutoffs. Mapping quality cutoff is the lower bound for the Phred quality score for the reads used in particular run. For example with cutoff value of 20, short reads of mapping quality value 30 and over, are used. Mapping quality concept was explained in [57]. The reason for using simulated data is that it is possible to estimate both discarded reads and wrongly aligned reads of alignment. On real data, in a shotgun sample, it is not known where the reads come from. It is not practical to calculate wrongly aligned reads as we do not know what the correct alignment is. As a consequence, it is impossible to assess read alignment accuracy on real data and so not appropriate to put an effort on measuring the accuracy of the alignment via real sequencing data.

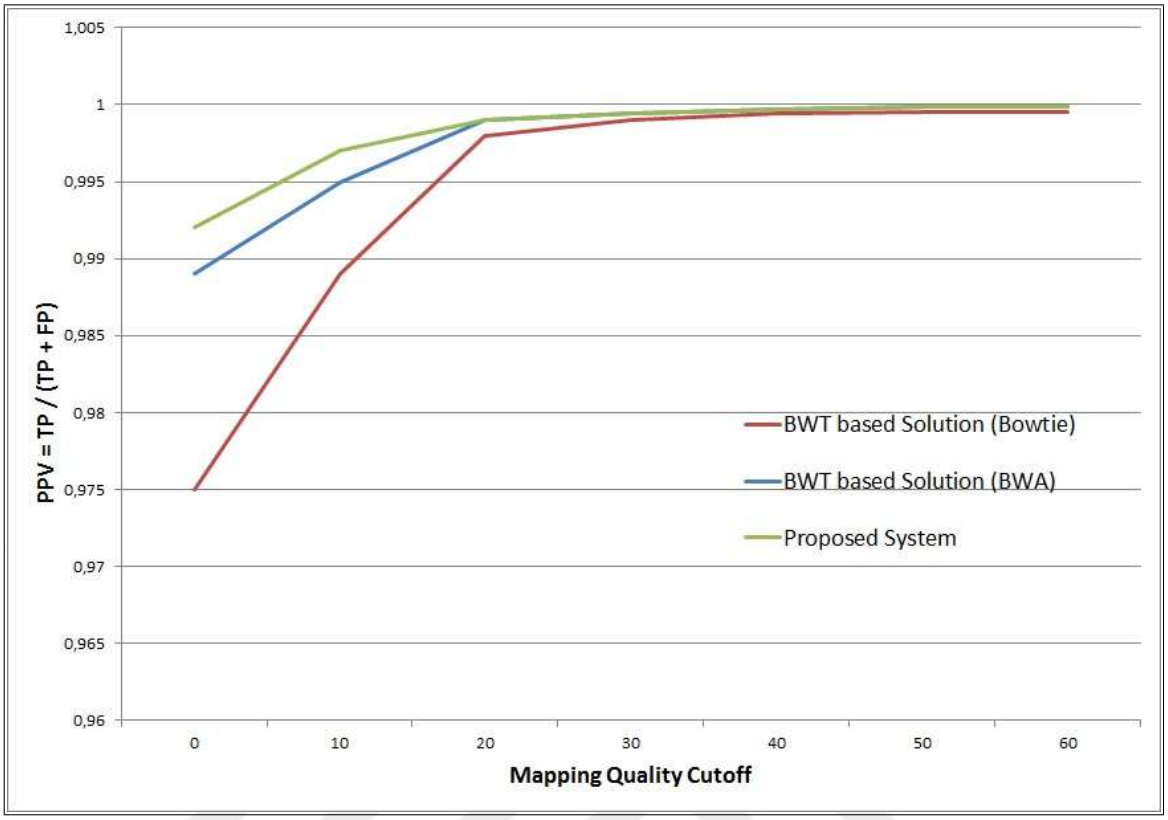


Figure 4.3. The positive predictive value of alignment methods (the number of correctly mapped reads divided by the total number of mapped reads) given as a function of mapping quality threshold. Two million simulated short reads of 100 bp long are aligned to Chromosome 20 of Human Hg19 reference. PPV, TP, and FP stand for positive predictive value, true positive and false positive, respectively. Proposed system is compared to cutting edge BWT based solutions

Receiver operating characteristic (ROC) curves for the data used to calculate the PPV curves are given in Fig. 4.4. The total number of mapped reads (x axis) is plotted against the number of incorrectly mapped reads (y axis). Each point on the curve represents the number of alignments whose mapping qualities are greater than or equal to the indicated value. Proposed system has relatively a smooth and low sloped curve compared to other aligners and outperforms other two alignment softwares especially as the mapping quality cutoff drops. BWA also has a very good performance above mapping quality cutoff values greater that 30. But when quality cutoff drops below 30 there is a steep loss in the mapping performance.

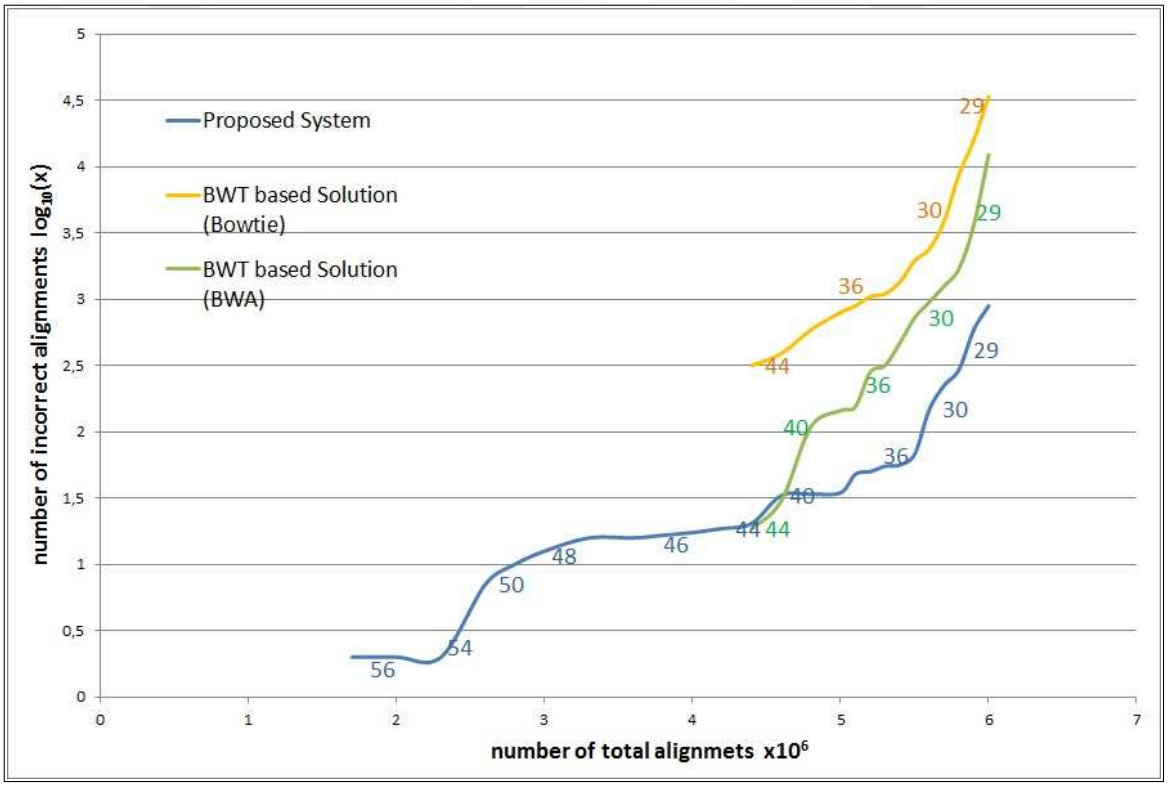


Figure 4.4. The receiver operating characteristic (ROC) curves for the simulated data used in Fig. 4.3. Incorrect alignments are plotted against total alignments with mapping quality cutoffs. Points on the curve represents minimum mapping quality value for that alignment.

The reliable detection of genomic variation for variants larger than a few base-pairs is another challenge. Though they are difficult to align, it is possible to identify read sequences crossing boundaries of structural variation. Proposed system solely relies on SW algorithm which is considered to be the appropriate solution to align gapped sequences with short

INDELs with seeking for all possible frame of alignments with all possible gaps. Effect of INDELs on the alignment sensitivity and its comparison with other aligners are tested on simulated Illumina single-end reads of 100bp length containing 1-10 bp INDEL events. The simulation software was MUTATRIX [60] genome simulator. This is again selected to keep the comparison similar to [55]. For each indel length, an average of 100 events, with approximately 1000 spanning reads are simulated. Sensitivity is expressed as ratio of correct alignments to total reads and plotted as a function of INDEL length in Fig. 4.5.Here, correct alignment is the one mapping to correct position and also containing the matching variant within the alignment. Proposed system expresses better sensitivity than two other aligners in terms of deletion handling. In terms of insertion handling it's performance is between Bowtie and BWA. Fig. 4.6 presents the sensitivity of the system with the correct alignment criteria set to the situation that mapping position of the read is correct.

Aligners provide information on where reads map in the human genome along with information on the confidence of the mapping, However, they do not themselves weigh evidence for genetic variants in the genome being studied. Dedicated variant callers use the information provided by mapper in statistical models to determine if there is enough evidence to report a difference with respect to the reference genome. To determine the effect of the mapping on single nucleotide polymorphism (SNP) discovery, we simulated 1486 SNPs on the human genome chromosome 20 using MUTATRIX. We then used MASON to generate one million 100bp reads from the mutated chromosome carrying SNPs. BWA, Bowtie and proposed aligners aligned those reads back to the entire human reference genome. After the alignment, SNP calling is performed via GATK. Variant caller's sensitivity to SNPs as a function of the false discovery rate (FDR) is evaluated by taking SNP calls with variant quality score over a cutoff value. Sensitivity of the SNP caller to the data aligned by different aligners is given in Fig. 4.7. True positive (TP), false positive (FP), and false negative (FN) are calculated by intersecting SNPs called on each aligner's alignments. SNP call performance of the proposed system is better than the other two aligners.

System's alignment speed tests are conducted in two stages. In the first stage, proposed alignment method is coded in C to run on CPU. Short read sequences from datasets with various read lengths are aligned to Chromosome 20 of Human Hg19 reference genome.

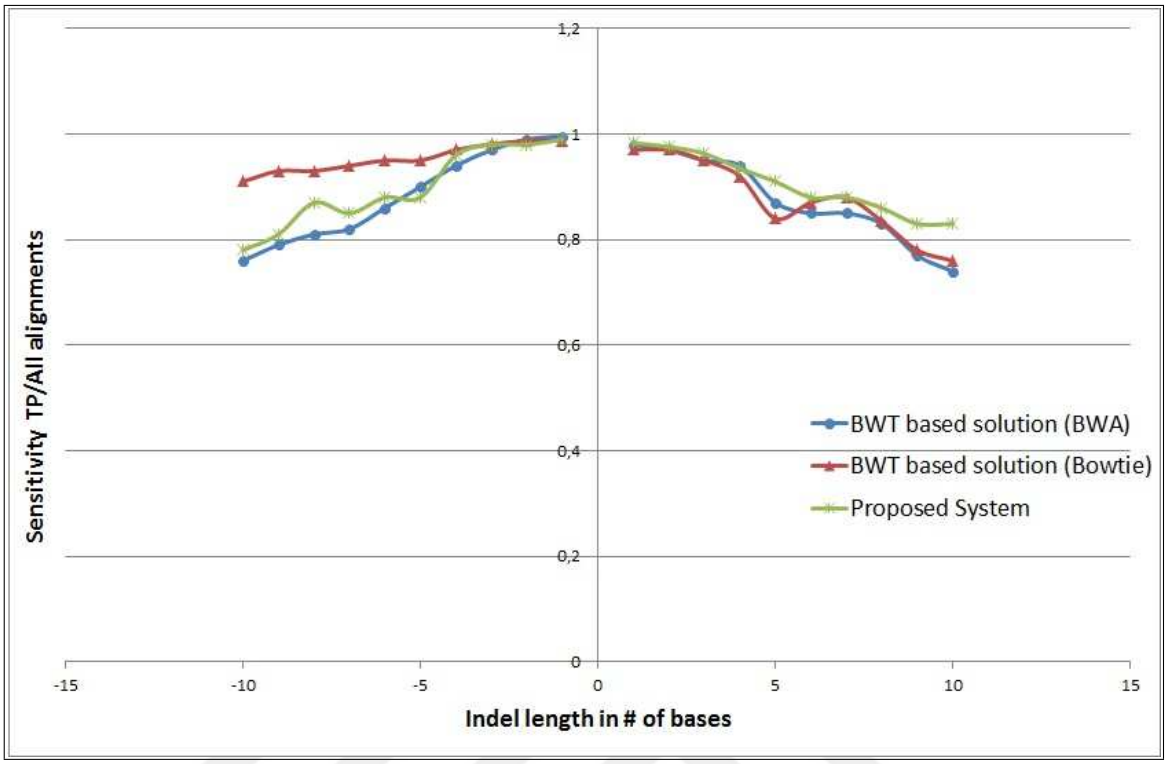


Figure 4.5. Indel sensitivity evaluation with large simulated short read dataset. Short read sequences contain INDELs and are aligned to Chromosome 20 of Human Hg19 reference genome. Results are presented as the ratio of correctly aligned reads to total number of simulated reads. Alignment is considered correct if read id aligned to correct location and includes the variant within. Proposed system is compared to cutting edge BWT based solutions

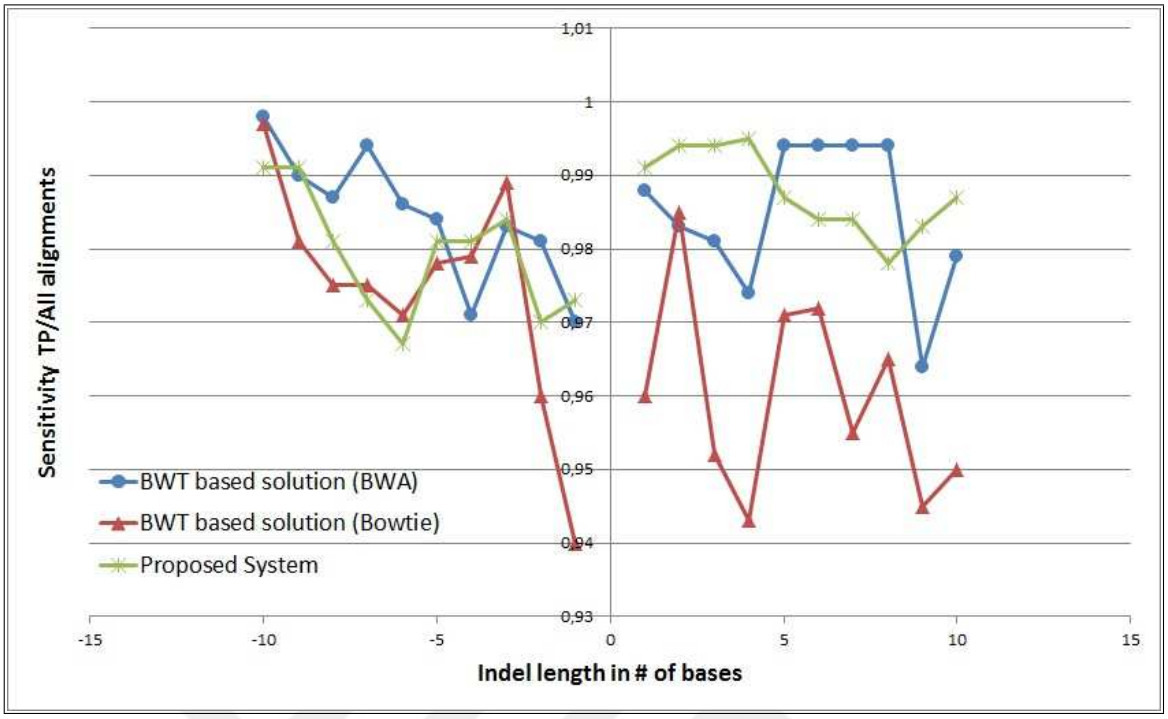


Figure 4.6. Indel evaluation with large simulated short read dataset. Short read sequences contain INDELs and are aligned to Chromosome 20 of Human Hg19 reference genome. Results are presented as the ratio of correctly aligned reads to total number of simulated reads. Correctness criteria is mapping to correct location only.

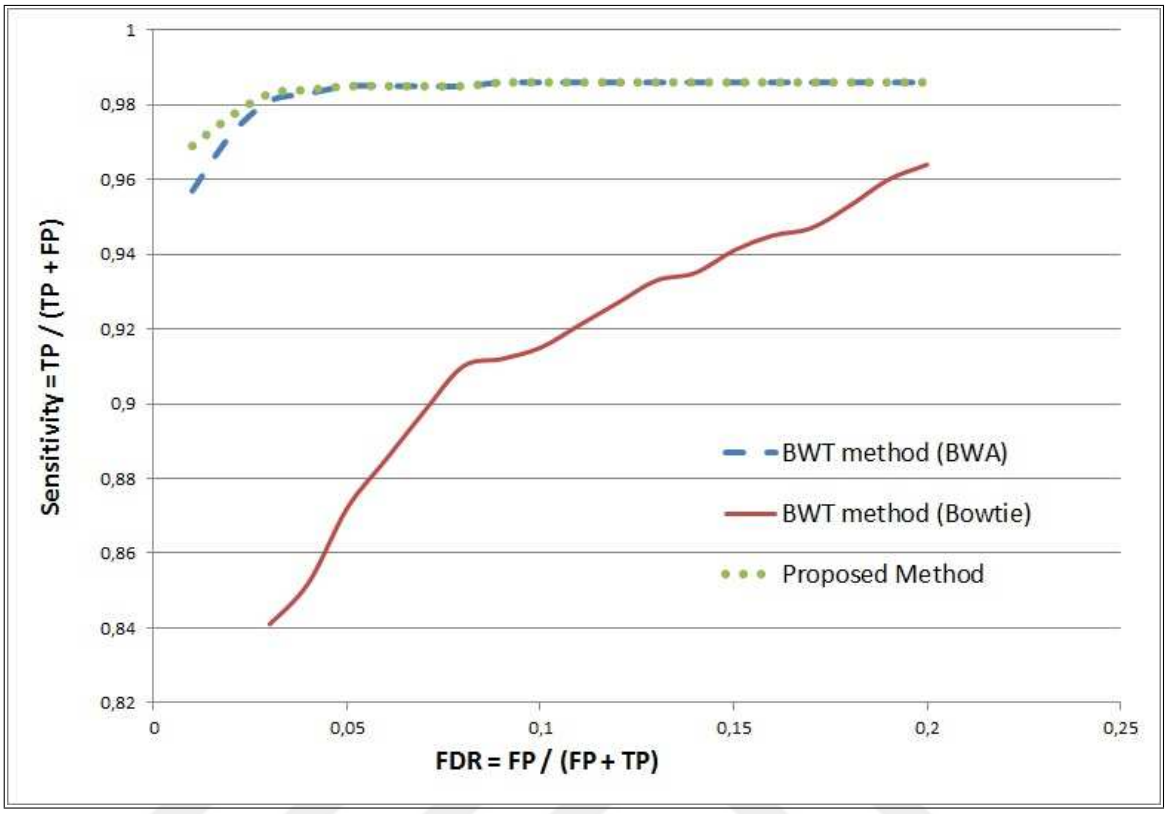


Figure 4.7. SNP call performance of proposed system compared to other two cutting edge BWT based aligners. Short read sequences are created from Chromosome 20 of Human Hg19 reference genome. Chromosome is mutated with SNPs. Short read sequences are simulated with errors and indels from mutated chromosome and are aligned to mutated reference. SNP call is accomplished via GATK variant caller.

Each dataset contains thousand short read sequences which are simulated with errors and variants. Alignments are performed on a PC together with proposed system and runtime results are compared. Alignments are performed with single core on CPU and single CA on FPGA. Alignment speed performance comparison is presented in Table 4.4 as ratio of CPU runtime to FPGA runtime.

Table 4.4. Alignment speed evaluation of proposed algorithm on CPU and FPGA with various read lengths. Simulated read datasets containing SNPs and INDELs are used for evaluation.

Read Length	**CPU runtime / FPGA runtime**
26	1.312
50	2.402
76	3.392
100	4.618
300	14.528

The main downfall of the system is that it has long mapping duration compared to BWT or Hash based approaches. This is mainly due to the entire shift of the three billion base-pair reference genome over each read; which means three billion clock cycles. This is also the reason for the running of sensitivity and accuracy evaluations over a few million reads but not a few ten millions.

In order to test system runtime for reads of various length, we simulated reads of lengths 100, 200, 300, 1000, and 10000 and 50000 base-pairs of total 1000000 base-pair for each read length. Runtime of alignments are given in Fig. 4.8. All reads of length up 300 are practically aligned to Chromosome 20 of Human reference genome Hh19. For read lengths of 1000,10000 and 50000; similarity matrix is partitioned. Submatrix results are computed, stored and merged to obtain results. This is because, FPGA board used does not have enough resources for fitting a monolithic computation array longer that 300bp. Runtime results are compared with an Hash based method running on FPGA and BWT based CPU method. As it can be seen in Fig. 4.8, in terms of computation time, proposed system can deliver practical alignment durations with longer read sequences. Also system is capable of handling reads over 1M long which is not possible with BWT base method BWA.

New sequencing platforms like Pacific Bioscience's PackBio which claim to produce reads up to 15K long and Oxford Nanopor which claims 50K long; proposed system will be much feasible in terms of alignment time. One good point for those platform's outputs is that, they report to have high sequencing error rates like 10 percentage for PackBio and 15 percentage for Nanopor. Full SW is known to be superior at handling errors than hash or BWT based solutions ; so will our system be. Besides system will be faster with more logic resources are available.

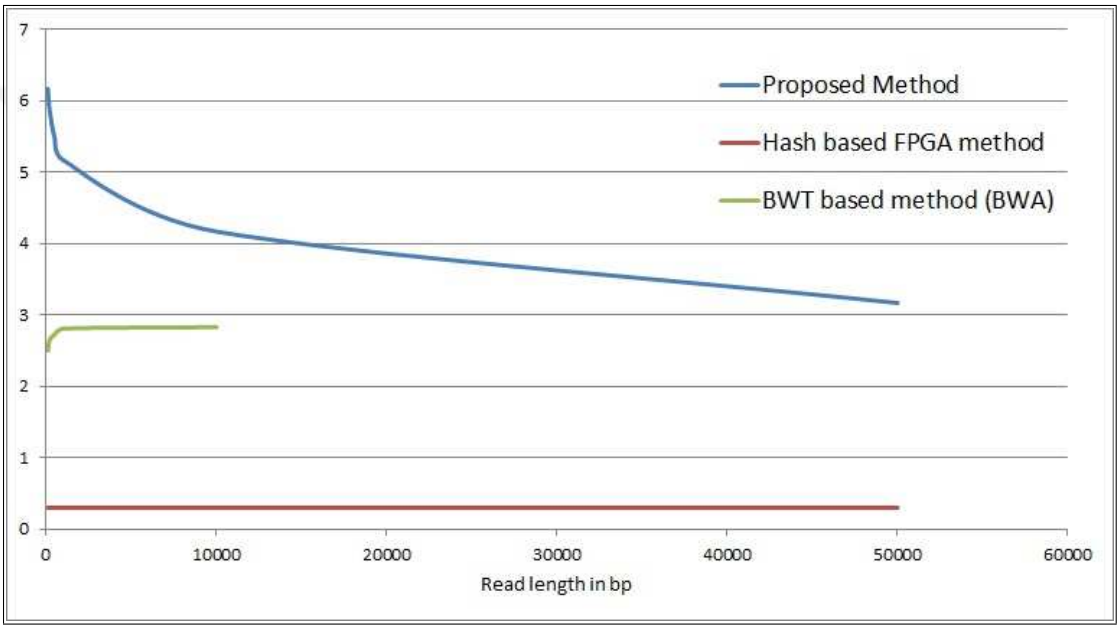


Figure 4.8. Runtime for various read length with a total of 1000000 base pair datasets. Vertical axis represents time in logarithmic scale. As read length increase, total system alignment time decrease. For longer reads, system can handle read lengths beyond 50K where BWT based cutting edge solution BWA does not even accept as input.

There are possible improvements that can be achieved for overall mapping performance. A module can be implemented with a memory area keeping the database of well known SNPs and a method not to penalize alignments against known SNPs can be followed in order to achieve better alignment. A similar way may be followed for *structural variants*. Known-insertion sequences such as mobile element insertions (MEIs), can be included as part of the reference genome for minimizing alignment artifacts. Also pair-end mapping scheme can be incorporated into the PC host system which will help to increase mapping accuracy. It is

hard work to implement a pair-end mapping in FPGA.

Most of the tools relying on SW for better mapping perform band limited SW in which they align two sequences within a specified diagonal band. This way eliminates the need for searching all alignment possibilities and provides an advance in alignment time. Besides it allows longer sequences to be aligned and allows optimization within wider bands, which can include longer gaps [43]. In FPGA implementations, doing so does not decrease processing time or provides better alignment of long-gapped sequences; but costs in additional hardware resources.

5. CONCLUSIONS

In this dissertation, we proposed a reconfigurable hardware based system to align short read data, produced by NGS machines, to a reference genome. We designed a novel processing element using *Phred quality scores* for systolic array structure computing the dynamic algorithm in order to get optimum alignment positions. Specifically attention is paid on increasing the sensitivity and accuracy of the system. Sliding of the whole reference sequence over each short read, full SW implementation and thresholding scheme implemented in hardware leads to high sensitivity of the system.

A significant property of our implementation is the incorporation of *Phred quality scores* of the read sequences into SW matrix computations. For evaluation of the impact that scheme creates, we prepared synthetic reads and conducted tests those reads with an other FPGA based system and with software aligners. It is observed that proposed system is capable of mapping to correct positions; hence it offers logically and practically better solution for alignment. We name this as accuracy and report our system as giving better mapping performance in terms of accuracy over the standard affine-gap model.

FPGA based systems usually process the reference sequence to uncover potential alignment regions with indexing and hash based methods. This method is practical by means of reducing overall computation time allocated to alignment via dynamic programming which is the bottleneck of the analysis pipeline. In fact, the main problem addressed by those systems is the computation time. Our system slides the whole genome sequence over the short read, aiming to increase sensitivity. As whole human genome should be clocked over the short reads, it needs three billion clock cycles for the human genome to align with a single short read sequence. This is the reason for the main drawback of the proposed system; alignment speed.

Unfortunately, due to the nature of the proposed method, it is not possible to deliver mapping time close to software or GPU based aligners. SW computations are the most time consuming stage for all aligners, there are new approaches like SIMD SW which can

be implemented in CPU but not on FPGA. This kind of implementation extend the alignment time gap. Algorithms employing indexing, finding CALs and aligning reads on FPGA over CALs can outperform software aligners; but that time they sacrifice from mapping quality and sensitivity.

Fortunately, proposed design scales linearly with the FPGA resources available. With each iteration of Moore's Law, FPGAs double in size. Our alignment appliance FPGA design should scale up to these larger chips easily. It can simply have twice the number of registers. It is hard to predict the effect on clock speed but it should stay the same or probably increase as the larger, newer FPGAs will be better designed and the signals are moving the same physical distance as in the older FPGAs. Besides, solution proposed offers potential to be scaled and distributed to multiple FPGAs. At the very beginning of this study that was something which we were planning to implement; possibly on a PICO computing board with multiple Virtex4 or Virtex5 FPGAs. But it could not be achieved due to the high cost of custom boards. We implemented the system on a single FPGA board. Scaling the system to multiple larger FPGAs obviously will increase the alignment speed; but brings a new problem to the table; high system cost. Luckily, NGS platforms continue to improve. Hence, they are expected to produce short reads of length a few thousand base pairs. With read lengths of few thousand base pairs, the proposed system can deliver reasonable alignment speed.

There is still room for increasing alignment speed on single FPGA. Each cell update processing engine in the systolic array design currently takes four clock cycles per computation. Therefore, the speed at which the systolic array operates is four times slower than the clock frequency of the FPGA. However, within the processing engine there are four separate stages, each of which only requires one cycle to execute. On any given clock cycle, one of the stages is performing useful work and three of the stagess output is ignored. It is possible to modify this design to take advantage of the fact that on every clock cycle every stage is performing a computation. This can be achieved via four short-read base-pairs to each cell update engine instead of one with a multiplexer placed at the front of the engine which will cycle through the four short-read base-pairs. In this way it will be possible for each systolic array to execute four short-read alignments in the same time it takes for the

current design to execute just one which will lead to an increase in the CA clock rate by a factor of four.

The system still can be practically used without modifications or scaling. One possible usage of the proposed system may be for areas on the genome with a few 10K base lengths for a deeper search than standard aligner tools. Also as the system carry potential to align sequences with large INDELs, or map reads which is not mapped in index based aligners. System can be adopted to work in collaboration with a software aligner and undertake mapping of reads discarded by software tool.

REFERENCES

1. 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*, 10-3, 2009.

2. H. Lii and R. Durbin, Fast and Accurate Short Read Alignment with Burrows Wheeler Transform, *Bioinformatics*, 24-14: 1754-1760, 2009.

3. M. Burrows, D.J. Wheeler. A Block-sorting Lossless Data Compression Algorithm, *Digital Equipment Corporation*, Technical Report: 124, Palo Alto, CA, 1994.

4. P. Ferragina and G. Manzini, Opportunistic Data Structures with Applications, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, Washington, DC, 2000.

5. T.F. Smith and M.S. Waterman. Identification of CommonMolecular Subsequences. *Journal of Molecular Biology*, 147-1:195-197, 1981.

6. N. Homer, B. Merriman and S.F. Nelson, *BFAST: An Alignment Tool for Large Scale Genome Resequencing*, PLoS ONE, 4-11:67-77, 2009.

7. A. Wandelt, A. Rheinlander, M. Bux, L. Thalheim, B. Haldemann and U. Lese. Data Management Challenges in Next Generation Sequencing. *Datenbank-Spektrum*, 12:161-171, 2012.

8. S.W. Golomb. Run-length Encodings. *IEEE Transactions on Information Theory*, 12:399-401, 1966.

9. M. Akgun, M. Sagiroglu. A New PPM Model for Quality Score Compression. *Proceedings of the IEEE 21th Conference on Signal Processing and Communications Applications*, 2013.

10. E.S. Langmead. Initial Sequencing and Analysis of the Human Genome. *Nature*, 409:860-921, 2001.

11. A. Sundquist, M. Ronaghi, H. Tang, P. Pevzner and S. Batzoglou. Whole-genome Sequencing and Assembly with High-throughput Short-read Technologies.*PLOS One*, e484, 2007.

12. E.R. Mardis. The Impact of Next-generation Sequencing Technology on Genetics. *Trends in Genetics*, 24:3, 2007.

13. National Human Genome Research Institute. http://www.genome.gov [retrieved 16 February 2014].

14. F. Sanger, S. Nicklen and A.R. Coulson. DNA Sequencing with Chain-terminating Inhibitors. *Proceedings of the National Academy of Sciences USA*, 74-12:5463-7, 1977.

15. C.A. Hutchison. DNA Sequencing: Bench to Bedside and Beyond. *Nucleic Acids Research*, 35-18:6227, 2007.

16. C. Trapnell and S.L. Salzberg. How to Map Billions of Short Reads onto Genomes. *Nature Biotechnology*, 27:5, 2009.

17. L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*, Cambridge University Press, 2005.

18. S. Needleman and C. Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology*, 48-3:443-453, 1970.

19. C.E. Cormen, R. Leiserson and L. Rivest. *Introduction to Algorithms*. The MIT Press, 2. edition, 2003.

20. A. Cox. ELAND: Efficient Local Alignment of Nucleotide Data. *Unpublished*, 2007.

21. B. Ewing , L. Hillier , M. Wendl and P. Green, Base-calling of Automated Sequencer Traces Using Phred I. Accuracy Assessment , *Genome Research*, 8-3:175185 , 1998.

22. H.L. Crochemore. *Algorithms on Strings*, Cambridge University Press, 2007.

23. R. Li, Y. Li , K. Kristiansen and J.Wang. SOAP: Dhort Oligonucleotide Alignment Program. *Bioinformatics*, 24:713-714, 2008.

24. B. Langmead and S.L. Salzberg. Fast Gapped-read Alignment with Bowtie 2. *Nature Methods*, 9-4:357-359, 2012.

25. C.M. Liu, T. Wong, E. Wu, R. Luo, S.M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K Zhao and T.W. Lam. SOAP3: Ultra-fast GPU-based Parallel Alignment Tool for Short Reads. *Bioinformatics*, 15-6:878-879, 2012.

26. M. Fursov, I.E. Efromov and Y.E. Danilova. UGENE: High Performance Genome Analysis Suite. *Proceedings of the Fifth Moscow International Congress on Biotechnology*, 14-15, 2009.

27. D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.

28. D.T. Hoang and D.P. Lopresti. FPGA Implementation of Systolic Sequence Alignment, In: *Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping*, pages 183-191. Springer, 1992.

29. M. Ayala-Rincon, R.P. Jacobi, L.G.A. Carvalho, C.H. Llanos and R.W. Hartenstein. Modeling and Prototyping Dynamically Reconfigurable Systems for Efficient Computation of Dynamic Programming Methods by Rewriting-logic. *SBCCI 17th symposium on Integrated circuits and system design*, New York, 248-253, 2004.

30. S. Masuno, T. Maruyama, Y. Yamaguchi and A. Konagaya. Multiple Sequence Alignment Based on Dynamic Programming Using FPGA. *IEICE Transactions on*

Information and Systems, E90-D-12:1939-1946, 2007.

31. B. Strengholt. Acceleration of the Smith-Waterman Algorithm for DNA Sequence Alignment Using an FPGA Platform, *MS Thesis*, Delft University of Technology, Computer Engineering Department, 2013.

32. A. Jacob, J. Lancaster, J. Buhler, B. Harris and R. Chamberlain. Mercury BLASTP: Accelerating Protein Sequence Alignment. *ACM Transactions on Reconfigurable Technology and Systems*, 2:9, 2008.

33. D. Lavenier, L. Xinchun and G. Georges. Seed-based Genomic Sequence Comparison using a FPGA/FLASH Accelerator. *IEEE International Conference on Field Programmable Technology*, 41-48, 2006.

34. M. Herbordt, J. Model, Y. Gu, B. Suhwani and T. Van-Court. Single pass, BLASTlike, Approximate String Matching on FPGAs. *IEEE Symposium on Field Programmable Custom Computing Machines*, 217-226, 2006.

35. F. Xia, Y. Dou, D. Zhou and X. Li. Fine-grained Parallel RNA Secondary Structure Prediction using SCFGs on FPGA. *Journal of Parallel Computing*, 36-9:516-530, 2010.

36. N. Homer, B. Merriman and S.F. Nelson. BFAST: An Alignment Tool for Large Scale Genome Resequencing. *PLoSONE*, 4:11, 2009.

37. R. Lipton and D. Lopresti. A systolic array for rapid string comparison. *Proceedings of Chapel Hill Conference on Very Large Scale Integration*, p.363-376, 1985.

38. H.Y.M. Fritz, R. Leinonen, G. Cochrane and E. Birney. Efficient Storage of High Throughput DNA Sequencing Data Using Reference-based Compression. *Genome Research*, 21:734-740, 2011.

39. D.C. Jones, W.L. Ruzzo, X. Peng and M.G. Katze. Compression of Next-generation Sequencing Reads Aided by Highly Efficient De Novo Assembly. *Nucleic Acids*

Research, 40, 2012.

40. P.J.A. Cock, C.J. Fields, N. Goto, M.L. Heuer and P.M. Rice. The Sanger FASTQ File Format for Sequences with Quality Scores, and the Solexa/Illumina FASTQ Variants. *Nucleic Acids Research*, 38-6:1767-1771, 2009.

41. E.V. Koonin, S.F. Altschul and P. Bork. BRCA1 Protein Products: Functional Motifs, *Nature Genetics*, 13: 266-267, 1996.

42. P. Guerdoux-Jamet and D. Lavenier. SAMBA: Hardware Accelerator for Biological Sequence Comparison. *CABIOS*,12-6:609-615, 1997.

43. C. Kun-Mao, W. Pierson and W. Miller. Aligning Two Sequences Within a Specified Diagonal Band. *Computer Applications in the Biosciences*, 8-5:481-487, 1992.

44. S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215-3:403-413, 1990.

45. I.T. Li, W. Shum and K. Troung. 160-fold Acceleration of the Smith-Waterman Algorithm Using a Field Programmable Gate Array (FPGA). *BMC Bioinformatics*, 8:185, 2007.

46. Y. Yamaguchi, T. Maruyama and A. Konagaya, *High Speed Homology Search with FPGAs*, Proceedings of Pacific Symposium on Bio-computing, p.271-282, 2002.

47. R. Sidhuand and V.K. Prasanna, *Fast Regular Expression Matching Using FPGAs*, IEEE Symposium on Field-Programmable Custom Computing Machines, April, 2001.

48. O. Knodel, T.B. Preusser and R.G. Spallek, Next Generation Massively Parallel Short-read Mapping on FPGAs, *IEEE International Conference on Application Specific Systems, Architectures and Processors*, p.195-201, 2011.

49. M. Kim and C.B. Olson, Hardware Acceleration of Short Read Mapping, *IEEE Symposium on Field-Programmable Custom Computing Machines*, p:161-168, 2012.

50. D.T. Hoang, *A Systolic Array for the Sequence Alignment Problem*, Techinal Report, Brown University, CS-92-22, 1992.

51. O. Gotoh, *An Improved Algorithm for Matching Biological Sequences*, Journal of Molecular Biology, 162:705-708, 1982.

52. M. Gok and C. Yilmaz, Efficient Cell Designs for Systolic Smith Waterman Implementation, *Proceedings of the International Conference on Field Programmable Logic and Applications*, p:889-892, 2006.

53. M. Gok, C. Unsalan, S. Goren and M. Sagiroglu, Programmable Hardware based Short Read Aligner Using Phred Quality Scores, *Internation Conference on Social Computing*, p:864-867 , 2013.

54. P. McMahon, *Accelerating Genomic Sequence Alignment using High Performance Reconfigurable Computers*, University of Cape Town, Computer Science Department, 2008.

55. W.P. Lee, M.P. Stromberg, A. Ward, C. Stewart, E.P. Garrison and G.T. Marth, *MOSAIK: a Hash-based Algorithm for Accurate Next-generation Sequencing Short-read Mapping*, PLoS One, 9-3, 2014.

56. P. Klus, *A Fast Short Read Sequence Aligner Using Graphics Processing Units*,Tech. Rep. 5:27, BioMed Central Ltd, 2012.

57. H. Li, J. Ruan and R. Durbin, *Mapping Short DNA Sequencing Reads and Calling Variants Using Mapping Quality Scores*,Genome Research, 18: 1851-1858, 2008.

58. 1000 Genomes, ”A Deep Catalog of Human Genetic Variation”, http://www.1000genomes.org [retrieved 24 April 2014].

59. 10/100/1000 Mbps Tri-mode Ethernet MAC, Open Cores, http://opencores.org [retrieved 11 March 2014].

60. Mutatrix: Population Genome Simulator, https://github.com/ekg/mutatrix [retrieved 14 June 2015].

61. MASON: Read Simulator for Second Generation Sequencing Data, http://www.seqan.de/projects/mason/ [retrieved 14 June 2015].

62. IonTorrent NGS Sequencer Supplied Data, http://www.iontorrent.com/applications-pgm-accuracy/ [retrieved 14 June 2015].