KU LEUVEN

FACULTEIT INDUSTRIËLE
INGENIEURSWETENSCHAPPEN

CAMPUS BRUGGE

Genetic sequence alignment acceleration using a FPGA based platform

What methods can be used to accelerate the Smith-Waterman algorithm for genetic sequence alignment with an FPGA equipped platform?

Robin NOLLET

Supervisors: Ing. Václav Šimek
: Ing. Jonas Lannoo

Master Thesis to obtain the degree of
Master of Science in Engineering Technology:
Electronical engineering

Academic Year 2019 - 2020

Acknowledgements

Het voorwoord vul je persoonlijk in met een appreciatie of dankbetuiging aan de mensen die je hebben bijgestaan tijdens het verwezenlijken van je masterproef en je hebben gesteund tijdens je studie.

Thanks to:

BUT Vaclav Simek: promoter en help bij erasmus exchange Thomas martinek: idea BFAST en ondersteuning HLS Vojtech Mrazek: hardware

KUL Jonas Lannoo: ondersteuning vanuit Belgie tijdens erasmus naar thesis toe Sammy Verslype: originele idee accelerator met FPGA gebaseerd platform

AZ sj Friedel Nollet: idee versnellen reference mapping, factchecking biologie gedeelte en aangeven voorbeelddata

kort bedankinkje peuteman, bonte en studena voor administratie erasmus

Summary

De (korte) samenvatting, toegankelijk voor een breed publiek, wordt in het Nederlands geschreven en bevat **maximum 3500 tekens**. Deze samenvatting moet ook verplicht opgeladen worden in KU Loket.

Abstract

Het extended abstract of de wetenschappelijke samenvatting wordt in het Engels geschreven en bevat **500 tot 1.500 woorden**. Dit abstract moet **niet** in KU Loket opgeladen worden (vanwege de beperkte beschikbare ruimte daar).

Keywords: Voeg een vijftal keywords in (bv: Latex-template, thesis, ...)

Contents

List of Figures

List of Tables

List of symbols

Maak een lijst van de gebruikte symbolen. Geef het symbool, naam en eenheid. Gebruik steeds SI-eenheden en gebruik de symbolen en namen zoals deze voorkomen in de hedendaagse literatuur en normen. De symbolen worden alfabetisch gerangschikt in opeenvolgende lijsten: kleine letters, hoofdletters, Griekse kleine letters, Griekse hoofdletters. Onderstaande tabel geeft het format dat kan ingevuld en uitgebreid worden. Wanneer het symbool een eerste maal in de tekst of in een formule wordt gebruikt, moet het symbool verklaard worden. Verwijder deze tekst wanneer je je thesis maakt.

b	Breedte	$[mm]$
A	Oppervlakte van de dwarsdoorsnede	$[mm^2]$
c	Lichtsnelheid	$[m/s]$

List of abbreviations

Chapter 1

Introduction

Chapter 2

Background information in molecular biology

2.1 Biology and DNA

2.1.1 History of genetics and DNA

Genetics For thousands of years, humans have observed the effects of heredity and implemented their knowledge to domesticate plants and animals. However, the science behind heredity was only started to be understood since 1859 with the publication of *on the origin of species* by Charles Darwin.

Gregor Mendel

Around 1865, Austrian monk and botanist Gregor Mendel, who studied at the university in Brno in the current Czech Republic, published his results on the hybridization studies of pea plants. He is often credited as being the father of modern genetics. In his findings, he implemented the role of *factors* that influence the expression of traits. These factors later became known as *genes*.

Molecular biology In 1869, Swiss physician Friedrich Miescher discovered a microscopic substance in the pus of discarded surgical bandages. Later, in 1909, Phoebus Levene named this substance Deoxyribonucleic Acid (DNA) since it is found in the nucleus of a cell and has acidic properties.

The full structure of DNA was discovered by Francis Crick and James Watson at the Cavendish Laboratory at the University of Cambridge.

2.1.2 Structure of DNA

DNA, or Deoxyribonucleic Acid, is what stores the genetic information of all living organisms. It is the information that programs all of the activities in a cell.

Structurally, DNA is a polymer, which means each molecule is built up out of small repeating molecular units. In DNA, these units are called *nucleotides*.

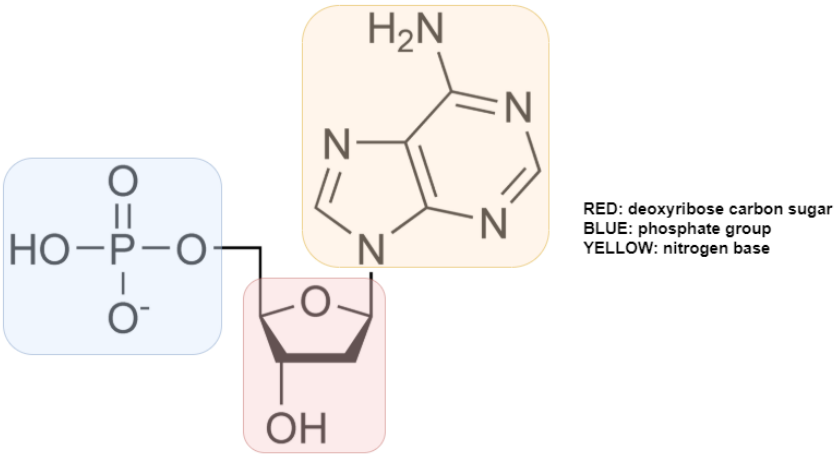

Figure 2.1: The structure of one nucleotide

Each nucleotide consists of 3 parts:

1. A carbon sugar molecule called *Deoxyribose*.
2. A phosphate group to connect the Deoxyribose molecules.
3. One of four possible nitrogen bases: Adenine (A), Thymine (T), Cytosine (C) or Guanine (G).

It is important to note that in most living organisms DNA does not exist as a single polymer, but rather a pair of molecules that are held tightly together. This is the famous *double helix*.

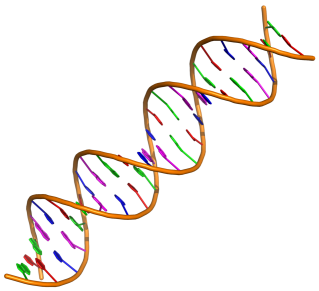

Figure 2.2: The famous double helix

Like in any good structure, there is a need for the main support. In DNA, the sugars and phosphates

bond together to form twin backbones. These sugar-phosphate bonds run down each side of the helix, but chemically in opposite directions.

The first phosphate group, at the start of the molecule, connects to the sugar group's 5th carbon. At the end of the structure, the 3rd carbon of the sugar group is unconnected. This makes a pattern typically noted as $[5' \rightarrow 3']$. Now, since the other molecule in the helix goes in the opposite direction, the pattern of the other backbone is typically noted as $[3' \rightarrow 5']$.

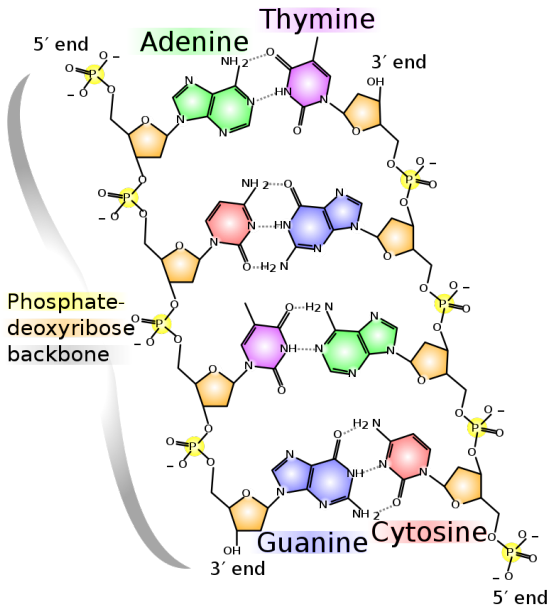

Figure 2.3: the DNA structure

These two long chains are linked together by the nitrogen bases via their relatively weak hydrogen bonds, but there can't just be any pair of nitrogen bases. Adenine can only make hydrogen bonds with Thymine. Likewise, Guanine can only bond with Cytosine. These bonded nitrogen bases are called *base paires*.

It is the order of these bases, which is also called the *sequence*, that allows this DNA to store useful information. In this way, e.g. $AGGTCCATG$ means something completely different as a base sequence than e.g. $TTCCAGATC$.

Since each of the bases in the sequence has only one possible counterpart, you can predict what its matching counterpart will be in the opposite string. For example:

If the following sequence is known

$$[5' - AGGTCCG - 3']$$

we can deduce the sequence in the other direction as

$$[3' - TCCAGGC - 5']$$

2.1.3 DNA in the human body

In human cells, DNA molecules can be found in the nucleus of all cells in the body. It consists of 46 very long molecules, which during cell division condense in what we call *chromosomes*. The only exception is in reproductive cells, which only have 23 chromosomes. These chromosomes are packed tightly together in the nucleus of the cell. If all of these chromosomes are put together, this makes about 3 billion base pairs. These 3 billion base pairs provide the assembly instructions for pretty much everything inside the cell.

These 46 chromosomes, which make up our whole DNA, are always present in pairs in the cells. Each time, the pair consists of one chromosome from each parent.

These 23 chromosome pairs are classified in:

- 22 pairs of autosomal chromosomes. These are marked 1 to 22 according to the length of the sequence. The longest chromosome (chromosome number-1) is 248,956,422 bases long. The shortest (chromosome number-22) is 50,818,468 bases long.
- In each cell, there is also an X chromosome plus an X or Y, dependent on the gender.

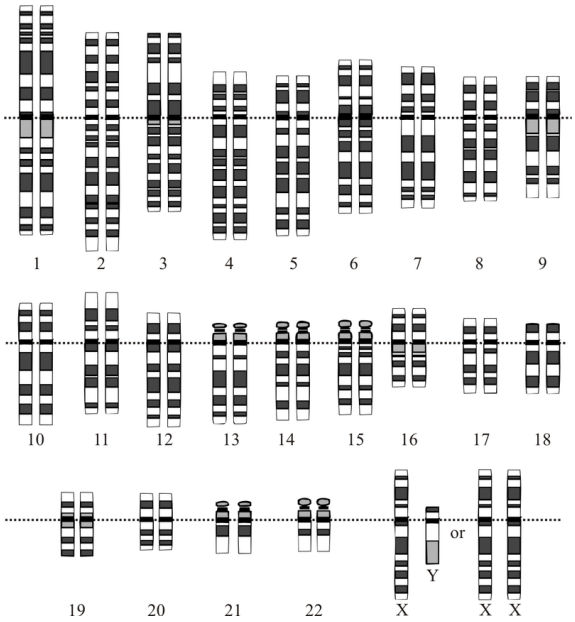

Figure 2.4: the human chromosomes

2.2 The Human Genome Project

In the field of Bioinformatics, an important dataset is the *Human Genome*. This is the full string of DNA found in the Nucleus, ordered from chromosome 1 to 22, followed by the X and Y chromosome.

In October 1990, biologists in the relatively new field of molecular biology started the Human Genome Project. The goal of this project was to determine the sequence of the 3 billion base pairs that make up human DNA. This project was completed in 2003, So nowadays we have a good idea of how the human genome is built up.

The Human Genome is easily found on the internet since it is publically available. One of the most often used is *HG19*, which was published in 2009. Since DNA has only 4 possible bases (A, T, C or G), this can be encoded in a 2-bit representation. If this encoding is used, the Human Genome is approximately 750 megabytes.

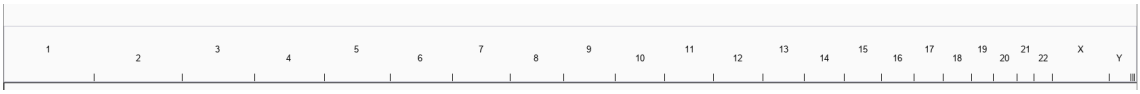

Figure 2.5: the order of the human genome

2.3 Sequencing

2.3.1 The sequencing technology

The term *Sequencing* is used for all techniques to read and decipher the DNA code from a given snippet of DNA. During the last years, the techniques that sequence human DNA has changed quite a lot. For about 15 years the *Next Generation Sequencing (NGS)* is the technique most often used. The biggest advantage of NGS, in comparison with other techniques, is the speed of the sequencing since it can sequence billions of short DNA molecules in parallel. In practice, this sequencing is most often done by the instruments of Illumina, which dominates the market (around 90% market share).

How NGS works

1. The DNA to sequence is isolated from the cells. Most often this is the whole genome.
2. The isolated DNA can now be copied enzymatically. This step is repeated until there are enough copies of the same DNA, usually this is in the millions or billions of copies.

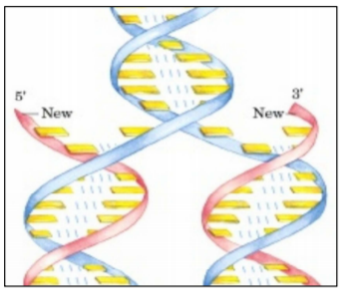

Figure 2.6: the enzymatic copying of a string of DNA. The original is unzipped, thus allowing new nucleotide bases to attach to the exposed bases.

3. The full DNA sequence is now broken apart into small DNA molecules (100 to 1000 bases long). This is done using enzymes or high-frequency sound waves.

4. Now the sequencing can start: a *flow cell* is used where these small DNA molecules can bind to a glass surface.

5. Different enzymatic and chemical reactions can now be done on this flow cell through an automatic flow of reagents. The following steps are iterated until the full read has been filled in:

   (a) The entire flowcell is filled with nucleotides, all with different nitrogen bases. Important is that at each of these nucleotides there is a fluorescent group attached to the phosphor group. This makes sure no other nucleotide can bind.

   (b) The fluorescent groups have a different color, dependent on the nitrogen base attached (A, G, T of C). At this time a camera picture of the flowcell is taken and stored.

   (c) after the flowcell is emptied of the loose nucleotides, another reagent flows in this flowcell. This reagent splits the fluorescent group from the phosphor group, so that in the next iteration a new nucleotide group can bind with the read.

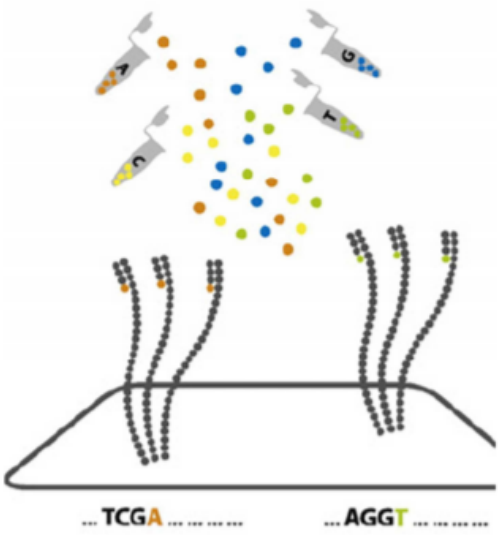

Figure 2.7: Sequencing technology used by Illumina attaches a nucleotide with a fluorescent tag to the next base in the read, captures a picture the read to determine the base, and removes the fluorescent tag so a new nucleotide group can bind in the next iteration.

6. After the whole DNA snippets have been filled in, the machine deduces the sequence in the DNA snippet. The pictures taken during the operation are in order the colors released in a specific spot, and by extent the attached nitrogen base. By the means of some image processing techniques, it is quite easy to get all the sequences in the flowcell. this is called the *Primary processing*.

Figure 2.8: From left to right is the pictures taken at each iteration in the flowcell. The color at that specific spot marks which nucleotide has been bound. With the use of some image processing techniques the exact sequence in that spot can be identified.

7. In the *secundary processing*, the sequence is trimmed by quality, etc. The operations that are done on the read in this step are outside the scope of this thesis.

As a result of the NGS, we get a file in the FASTQ format.

2.3.2 The FASTQ file format

Since the color in the camera pictures in the primary processing can have a light shift, there is a specific ”uncertainty” what the base is in that spot. This is called the *quality* of the base.

The *FASTQ* file format has become the de-facto standard as output from sequencing instruments. It is a text-based format for storing both the bases in the sequence and their corresponding quality. FASTQ has become the de facto standard for storing the output of sequencing machines.

A FASTQ file uses four lines per sequence:

1. a ’@’ character followed by a sequence ID, plus an optional description.
2. The sequence of letters identified by the machine. This is either A, G, C, T or n when the base cannot be identified with a specific threshold certainty.
3. a ’$+$’ character, optionally followed by the sequence ID (again) and an optional description.
4. the quality values for each respective base in line 2. The length of this line must be the same as the number of bases in line 2

The quality score in memory is a value in the range $0x21$ (lowest quality) to $0x7e$ (highest quality). Since this value is represented in ASCII in the file format, this ranges from the ’!’ character to the ’$\sim$’ character. Hereunder is a complete list of the possible values of the quality score:

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Important to note is that this quality score is logarithmic. Also, the '@' and '+' character is contained in the possible values for the score, so when implementing the interpreter for this file, this is something to look out for.

A FASTQ file containing a single sequence might look like this:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

Keep in mind that most of the time a FASTQ file consists of multiple of these sequences, all stacked under each other.

Chapter 3

Methods for genetic sequence alignment

3.1 Genetic sequence aligning

The human genome (e.g. $HG19$) is used as a reference genome for all sequenced human DNA. However, The genetic code of all humans is slightly different. Genetic sequence alignment is the science where you try to align 2 sequences with each other so that the amount of differences is minimal. In this chapter, the most frequently used algorithms are examined.

3.1.1 Alignment in general

In genetic codes, there are 3 types of differences between the given sequence and the reference:

- Insertion: one or more bases have been added in the genetic code in a specific spot.
- Deletion: one or more bases have been removed from the genetic code in a specific spot.
- Substitution: one or more bases have been substituted by other bases.

Inserts and deletions are often described by a single term, *indel*. In literature, this is most often represented with a '$-$' character.

For example: if we want to align the following sequences:

```
Seq1: ATATCGGC
Seq2: ATCG
```

The alignment itself can now be done in different ways. Possible alignments are:

```
Alignment 1
Seq1: AtaTCgGc
Seq2: A--TC-G-
Alignment 2
Seq1: atATCGgc
Seq2: --ATCG--
```

Which alignment that is the actual output, depends on the algorithm and the given parameters.

Keep in mind, there is no one ”correct” alignment. The core of the alignment algorithms is the same each time, but the parameters of these algorithms are changed depending on the application.

3.2 Local VS global alignment

To explain the difference between local and global alignment, we can take a look at the following example:

```
The 2 DNA sequences:
Seq1: TCCCAGTTTGTGTCAGGGGACACGAG
Seq2: CGCCTCGTTTTCAGCAGTTATGTGCAGATC

Alignment 1 :
Seq1: -----------tccCAGTT-TGTGTCAGgggacacgag
Seq2: cgcctcgttttcagCAGTTATGTG-CAGatc-------

Alignment 2 :
Seq1 : tcCCa-GTTTgt-GtCAGggg-acaC-GA-g
Seq2 : cgCCtcGTTTtcaG-CAGttatgtgCaGAtc
```

Both alignments are valid but different. The first alignment is *locally aligned*. This means that the similarities are prioritized in the same region, with the similarity as high as possible. On the other hand, the second alignment is *globally aligned*. Here the similarities over the full length of the sequences are used for the alignment.

In practice, the local alignment is used most often, since it can give you information of 2 sequences that do not have (approximately) the same length.

3.3 Commonly used algorithms

In this section, we will take a look at some algorithms that are used most often for genetic sequence alignment.

The algorithms that are used most often are categorized in 2 ways:

- local alignment VS global alignments
- dynamic algorithms VS heuristic algorithms: dynamic algorithms are exact but slow and computationally demanding, whereas heuristic algorithms are faster but are approximations and the best alignment is not guaranteed.

Hereunder is a schematic view of some algorithms that are used in practice:

	Dynamic programming	**Heuristic programming**
Local alignment	Smith-waterman	FASTA, BLAST
Global alignment	Needleman-Wunsch	X

Table 3.1 Classification of genetic alignment algorithms

Keep in mind, a lot of other claimed ”algorithms” (for example BFAST, ...), are accelerated versions of the Smith-Waterman algorithm.

3.3.1 Needleman-Wunsch

Needleman and Wunch proposed a new algorithm for genetic sequence alignment in 1970, now known as the *Needleman-Wunsch* (N-W) algorithm. Since this algorithm is meant for global alignment. Since global alignment is seldomly used in practice, further analysis of the algorithm will not be done. However, N-W has a lot of similarities with the Smith-Waterman algorithm, discussed in the next section.

3.3.2 Smith-Waterman

The *Smith-Waterman* (S-W) algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981. It is a variation on (N-W), adapted for local alignment. It is a dynamic programming technique, so the optimal local alignment is guaranteed.

The core of the algorithm is a matrix fillup, with data dependencies on the previous cells. Hereunder an analysis of the algorithm:

1. Symbols used in the analysis:

   Let sequences $A = a_1a_2a_3 \dots a_n$ and $B = b_1b_2b_3 \dots b_m$ be the sequences that need to be locally aligned. Here n and m are the lengths of sequence A and B

2. Define the parameters:

   - Define $s(a,b)$ be the *similarity matrix* (sometimes also called the *substitution matrix*) for the two sequences. It is used for ”rewarding” when $a_i = b_j$ and ”punishing” when $a_i \neq b_j$.

In the most general way, we define the similarity score as a matrix of values, e.g.:

	A	**C**	**G**	**T**
A	3	-3	-3	-3
C	-3	3	-3	-3
G	-3	-3	3	-3
T	-3	-3	-3	3

Table 3.2 Similarity matrix example

Often, there are only 2 scores used (equal or not equal). In this case, the similarity matrix can be condensed as follows:

$$s(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$$

- Define d as the *gap penalty* which regulates the score for an insertion or a deletion. This parameter can be:
  - *Linear*: The penalty is constant. So, in this case, it doesn't matter e.g. the previous was also a gap.
  - *Affine*: An affine gap penalty considers gap opening and extension separately. For the sake of simplicity, this analysis will not cover it. The algorithm can be extended to include this affine gap penalty, but this would make the algorithm more complex and we would limit our ability to develop possible accelerations.

3. The initialization: We construct a scoring matrix H with dimensions $(n+1) \times (m+1)$. The first column and first row we initialize with 0.

   For example: if we want to align the sequences A = $TGTTACGG$ and B = $GGTTGACTA$:

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

Table 3.3 Example of the initialization of the scoring matrix

4. Matrix fill in: We fill in the matrix using the following formula:

$$H_{ij} = max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ H_{i-1,j} - d, \\ H_{i,j-1} - d, \\ 0 \end{cases}$$

If we keep in mind that the value of a cell may never be lower than 0, we can represent the data dependencies in the following schematic:

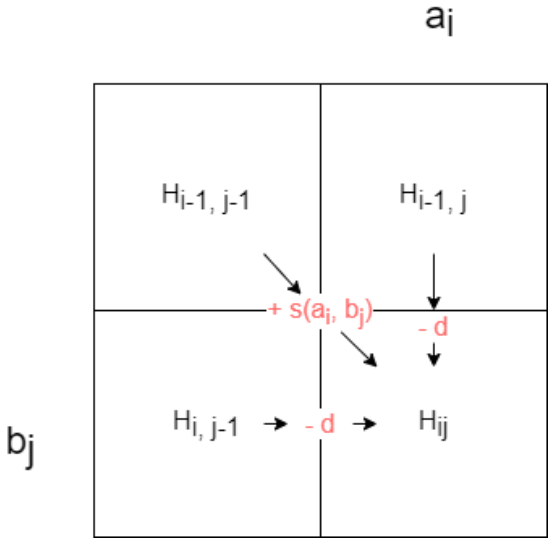

Figure 3.1: Data dependencies in the H matrix

Where $s(a,b)$ and d are the parameters of the algorithm. If we use the following values as an example:

$$s(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases} \qquad \text{and} \qquad d = 2$$

We can now fill up the scoring matrix H:

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Table 3.4 Example of a populated scoring matrix

5. Traceback: We start at the cell with the highest score in the matrix H. Starting here we only move left, up or diagonally (left-up) to the cell on which the value in the cell was based until we hit a cell with value 0.

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Table 3.5 Example of a traceback in S-W

From this traceback we can now deduce the following alignment:

```
GTT-AC
||||||
GTTGAC
```

This alignment is the output of our algorithm.

3.4 Problem definition

3.4.1 Mapping to a reference genome

From the sequencing machines, we get a big amount of reads in the FASTQ format. We should note that all these reads are worthless without a proper interpretation.

In most cases, the first step in the analysis of the reads is knowing from which part of the genome it is from. Typically, the read is compared with the whole genome in a local alignment, for example with the Smith-Waterman algorithm. As an output, we would get the position in the human genome and an alignment with its score (how well the sequence fits in that spot). This practice is commonly referred to as *Mapping to reference genome*.

Since the reads are 100 to 1000 bases long, and the whole human genome is approximately 3 billion bases, this comparison is computationally a very intensive task. If we analyze the S-W algorithm (as we have done in 3.3.2, we can see that the value of each cell in the matrix is only dependent on the left-upmost 3 cells. Therefore, It leads us to believe that this algorithm can be accelerated on other hardware solutions such as an FPGA (which will be discussed in chapter 4) since S-W is heavily parallelizable.

In most clinical applications where mapping to a reference genome is used, the number of reads to be compared with the genome is in the billions.

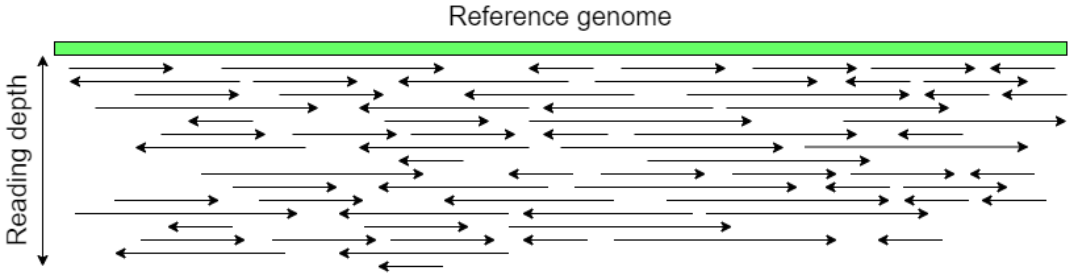

Figure 3.2: Mapping to a reference genome. The direction of the read is represented by arrows

Since the read can be from the complementary DNA molecule in the double helix, the sequences can be in forward $[5' \rightarrow 3']$ direction, or in complementary $[3' \rightarrow 5']$ direction. To transform that read to the used reference genome direction we need to perform the following changes to the read:

1. The bases should be changed to their corresponding base in the base pair;
2. The sequence should be reversed.

We have no way of knowing in which direction the read is taken, both the forward and the backward possibility should be compared with the reference and the best-aligned version of these two should be chosen.

Please note, in most normal cases we can assume the distribution of reads is practically uniform. Therefore, each base in the human genome will be covered by a statistically expected amount of reads. This amount is often referred to as the *reading depth*.

3.4.2 The sam and bam file format

As a convention, the output of mapping algorithms is in a *SAM* (Sequence Alignment Map) or *BAM* (Binary Alignment Map) file format. The BAM file format is just a compressed version of the SAM format, but the SAM format is more readable, which makes it easier for troubleshooting. There exists a lot of tools transform a SAM to BAM file already, so in this thesis we will focus on the SAM format. A SAM file constist of two sections, a header and an alignment section.

3.4.2.1 Header

The header is used for information which is independent of the alignments, such as name of the used algorithm, reference genome, used commands during generation and many more. The header must be in the beginning of the file, before the alignment section. Each line of the header field must start with an '@' character, so these lines are easily distinguished from lines the alignment section.

3.4.2.2 Alignment Section

Each line in the alignment section represents one mapped sequence, and consists of 11 mandatory and some optional fields, which are seperated with a tab.

The 11 mandatory fields are:

1. **Qname**: this is the name of the query (the sequence to match) and can be found on the first line of the FASTQ file, which is the input to the mapping algorithm.
2. **FLAG**: a combination of bitwise flags where each bit has a specific interpretation. It consists of 11 bits, but for basic alignments only 2 fields are important:
   - bit 2 (binary: $00000000X00$, integer value 4) is set to 1 if the sequence is unmapped or the map is not found
   - bit 4 (binary: $000000X0000$, integer value 16) is set to 1 if the sequence has been mapped as it's reverse complement.
3. **Rname**: the name of the reference genome. This can be found on the first line of the FASTA file, where the reference genome is stored. If the read is unmapped, this field contains a '*' character
4. **Pos**: the position of the leftmost base in the alignment. Keep in mind that the indexing of the reference starts with 1 for the first base.
5. **MapQ**: the mapping quality, which indicates how good the sequence fits in the specific position. This can be any value between 0 and 254. Value 255 is a reserved value to represent an unavailable quality.

6. **CIGAR**, which stands for *Concise Idiosyncratic Gapped Alignment Report*. It is a string which indicates where the matches (M), insertions (I) and deletions (I). For example, if the CIGAR states $3M1I6M2D10M$ this means from left to right: 3 matches, then an insertion, followed by 2 matches, 2 deletions an finally 10 matches. Keep in mind that the sum of the numbers in this string must be equal to the length of the sequence. In case the sequence is unmapped, this field should be filled with a '*' character

7. **Rnext**: reference sequence name of the primary alignment of the text read in the template. For this thesis we will fill this field with a '*' character.

8. **Pnext**: position of the primary alignment of the next read in the template. For this thesis we will fill this field with a '0' character.

9. **Tlen**: the observed template length, from the first till last mapped base. For this thesis we will fill this field with a '0' character.

10. **Seq**: the full sequence. This can be found in the FASTQ file on the second line.

11. **Qual**: the qualities of the sequence, also given by the FASTQ file on the fourth line.

//TODO voorbeeldje

3.4.3 Clinical application

We will discuss two clinical applications of mapping to a reference genome.

1. ***NIPT* (non-invasive prenatal testing), a test for detecting genetic defects in a foetus.** During or After conception, DNA can be lost or gained in the fertilized egg cell. This can result in a severe syndrome of the child. For example, Down syndrome is caused by a trisomy of chromosome 21. Normally all chromosomes are present twice in each cell, one from the mother and the other from the father. In Down syndrome patients something went wrong during cell division at the very early stage of development, and the fetus has in its cells three times chromosome 21. Because chromosome 21 is quite small and does not contain that many genes, the child can survive, though with typical mental and clinical problems.

Figure 3.3: Trisomy 21 karyotype

Before high throughput DNA sequencing technologies were available like they are today, testing if a fetus has a trisomy-21 could only be done by taking a small amount of amniotic fluid (fluid around the fetus). However, to obtain this fluid there was a need for a risky invasive procedure (called *amniocentesis*) leading sometimes to termination of the pregnancy.

It is known that small amounts of DNA of the fetus are present in the blood of the mother, in the cell-free DNA (*cfDNA*) which we find in the blood plasma (the clear, aqueous part of the blood). The blood plasma is used by our body to transport 'waste', including DNA from cells that were broken down. When fetal cells die, which is a normal process, the building blocks of these cells are transported in the plasma of the blood from the mother, included small DNA fragments from the fetus.

NIPT (non-invasive prenatal testing) is used to analyze DNA derived from the mother's blood. A large number of short cfDNA's are sequenced at random. Then, each sequence is mapped the whole human genome to find out where it comes from. Finally, the distribution of these reads is calculated. If we observed that a higher frequency of reads as compared to normal individuals coming from chromosome-21, it is almost certain that the fetus has Down's syndrome.

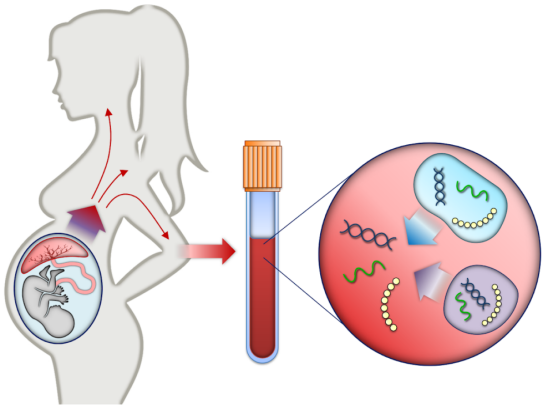

Figure 3.4: DNA of the fetus in the mothers blood

Using the same method, we can also find other defects in the number of chromosomes. For example trisomy 18 (Edwards syndrome), trisomy 13 (Patau syndrome) or even in the sex chromosomes, such as XXY (Klinefelter syndrome) or lack of a second X or a Y chromosome (Turner syndrome).

2. **Shallow whole-genome sequencing of tumor DNA.**
   It is a known fact that damaged DNA can lead to tumor development. This damage can be single bases changes but can also be loss or gain of larger DNA sequences where important genes are located. When someone is diagnosed with cancer, the knowledge of which DNA regions are lost or gained can be important to decide on treatment.

   A new technique to detect all gains and losses of DNA material in one single experiment is *sWGS* (shallow whole-genome sequencing). The technique is performed as follows: DNA

from the tumor is fragmented (it is broken in small pieces, eg. by a fragmentize enzyme or by high-frequency sound). These pieces are sequenced randomly, and with a mapping algorithm to the reference genome, the over- or underrepresentation of reads (as compared with a normal sample) indicates if regions of the DNA have changed, and which regions these are.

Chapter 4

Platforms for accelerating Smith-Waterman

As we saw in the analysis of the Smith-Waterman algorithm in 3.3.2, we can see that the value of each cell in the matrix is only dependent on the left-upmost 3 cells. Therefore, It leads us to believe that this algorithm can be accelerated on other hardware solutions which are better equipped for parrallellism than a normal CPU.

4.1 Overview of possible hardware

4.1.1 CPU

4.1.2 GPU

4.1.3 FPGA

4.1.4 ASIC

4.2 Hardware selection

4.2.1 Recent advances in High Level Synthesis

4.3 Platform communication

4.3.1 USB

4.3.2 Ethernet

4.3.3 Alternatives and selection

bv. SATA, PCI, ...

Chapter 5

Initial Difficulties during the implementation

Disclaimer: The purpose of this chapter is more to show the learning process I had to go through in order to achieve a suitable implementation for the genome mapping problem. I had to learn some new programming techniques such as HLS, SDSoC . . . especially for this thesis, since I did not have any experience in it yet. This chapter might be less applicable if the reader is interested in the science and implementation approach of the genome mapping problem, but can be of interest if the reader is also new to the mentioned programming techniques.

5.1 Using Vivado HLS and Xilinx SDK

5.1.1 learning HLS in examples

Ik moet nog starten met HLS, nog nooit gedaan.

learning HLS

lab1: BASIC WORKFLOW: How to make project, difference between sources and test bench, GUI and it's components, simulation, synthesis (=¿ estimate and Gantt chart), viewing resulting VHDL code, co-simulation, view simulation waveforms, IP Core export, creating multiple solutions and comparing, setting a directive (like unroll a loop).

lab2: DATA TYPES adhv FIR filter: floating point VS fixed point, saturation with fixed point, fixed point with larger internal bit-width (LSB laten vallen),

lab3: INTERFACES: - basic argument-level interfaces (ap_vld, ap_none, ap_hs) - block-level interface of pipelined component (pipeline on the top function)

lab4: ARRAYS: Sequential, pipelining internal loop, rewind parameter, top function pipelining, array map, array partition (cyclic and complete), array reshape technique

5.1.2 Learning how to program target board

hello world via Xilinx SDK =¿ baremetal -¿ zelf kiezen waar uitgevoerd + dataverplaatsingen zelf implementeren

5.2 SDSoC

Op een Linuxdistributie.

Snel functie verplaatsen van PS naar PL.

gebruikt ook delen van HLS

5.2.1 learning process on matrix multiplication example in SDSoC

Eerste test: matrixmultiplicatie

lab1: INTRODUCTION: basic workflow, working with hardware accelerators, running the project, ...

lab2: PERFORMANCE ESTIMATION: rescource utilization, comparing software and hardware implementations, overall speedup comparison (all using LINUX hosted)

lab3: diff between ACP, AFI and GP ACP Hardware functions have cache coherent access to DDR via the PS L2 cache. AFI (HP) Hardware functions have fast non-cache coherent access to DDR via the PS memory controller. GP Processor directly writes/reads data to/from hardware function. Inefficient for large data transfers. + error reporting (opening log files to see more information). Additional: the hardware functions and DMAs can only access the physical address space (not virtual) =¿ DMA that can handle a list of pages for a single array is known as Scatter-Gather DMA. (malloc(), sds_alloc()) Typically sds_alloc = best (area and performance) but uses physical memory (minder van beschikbaar dan virtueel).

lab4: directives to optimize the accelerator

lab5: task level pipelining =¿ speedup x3 theoretical, probably I make some mistakes

lab6: DEBUG

lab7: HARDWARE DEBUG: trace feature: what is the application doing? (software, hardware, transfer, recieve)

lab8: EMULATION

Chapter 6

Software implementation with pure Smith-Waterman

This capter will cover an approach to reference genome mapping using pure smith-waterman. This is a computationally very demanding task, but is more precise than heuristic methods such as FASTA, ...

6.1 The concept

For reference genome mapping a sequence of 100 to 1000 bases in length should be mapped to the reference genome. The idea behind this mapping and the applications is thouroughly described in 3.

In this implementation, a mapping with ”pure” Smith-Waterman was chosen. Commonly, an algorithm is used to select canditate locations where a small local smith-waterman is performed, or no smith waterman is performed at all. These methods are often less precise but computationally less demanding. Here, the alignment of the sequence is done with the full reference genome.

//figuurtje

6.2 General overview of the implementation

6.2.1 Parameters and Types

Nucleotide base First of all, It seemed important to define the nucleotide bases. There are only 4 possible bases (A, C, G and T), so 2 bits are enough. The following coding was chosen:

Base	A	C	G	T
Code	00	01	10	11

Table 6.1 Encoding for the nucleotide bases

To store these bases the uint8_t type type from the stdint library was used. it is 1 byte in size which is the smallest available type in the C language. If more time would have been available, and an implementation with the full human genome would have been made, it might be a good idea to define a specific type consisting of only 2 bits, which would be a lot more memory efficient.

DNA sequence type store string of DNA struct: length + pointer to first element types for the index: Ref and Seq

6.2.2 The code structure

1. reserve memory for:
   (a) genome
   (b) current read
   (c) reverse of current read
   (d) matrix for during the alignment
2. initialize the first row and first column of the matrix on 0
3. load the genome from file
4. open the FASTQ (for loading unmapped reads) and SAM (for storing mapped reads) file
5. for every read in the FASTQ file:
   (a) load the next read from FASTQ file
   (b) perform the alignment. This is the most computationally demanding step.
   (c) write the mapped read to the SAM file
6. close the open FASTQ and SAM file
7. free the reserved memory again

6.3 Details of the implementation

splitting up the problem.

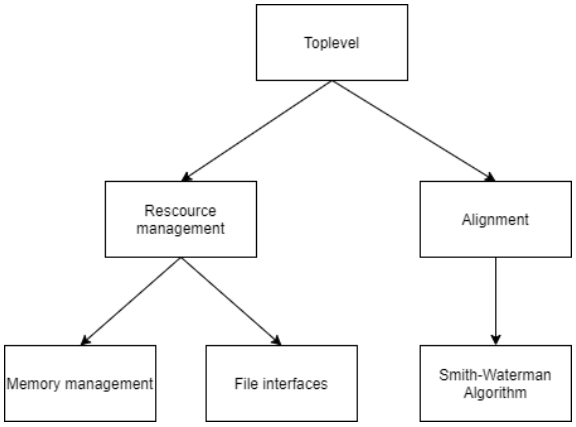

Figure 6.1: an organisation chart of the split up functionalities

6.3.1 File Interfaces

types!

Conversion functions char -¿ base and base -¿ char

FASTA interface =¿ loads genome: splitting functions Genome info Putting Reference info into array

FASTQ interface =¿ loading next read (stream) splitting functions get qname get sequence get qualities

SAM interface init for the defaults (Rnext, Pnext, Tlen) writeSAMline =¿ writes the mapped read in the sam file in the right format

6.3.2 Memory Management

sdsalloc VS malloc choosing sdsalloc so that hardware can also get it easily

6.3.3 The alignment

computational complexity: O(mn)

6.4 Implementation results

6.4.1 Sample data

PhiX galaxy FASTQ: https://github.com/Illumina/Isaac3/tree/master/src/data/examples/PhiX/Fastq coronavirus

//figuurtje IGV uitleggen reading depth + sequence direction desired result

6.4.2 Results

//figuurtje + bespreking

Chapter 7

Acccelerating the software implementation using HLS

Chapter 8

Implementation results

...

Chapter 9

Conclusion and future research

Gap opening and extension

BFAST algorithm

BASE type memory efficient

Bibliography

Aad, G., Abajyan, T., Abbott, B., Abdallah, J., Khalek, S. A., Abdelalim, A., Abdinov, O., Aben, R., Abi, B., Abolins, M., et al. (2012). Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29.

Cottrell, J. A., Hughes, T. J., and Bazilevs, Y. (2009). *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons.

Hughes, T. J., Cottrell, J. A., and Bazilevs, Y. (2005). Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39):4135–4195.

Appendix A

Uitleg over de appendices

Bijlagen worden bij voorkeur enkel elektronisch ter beschikking gesteld. Indien essentieel kunnen in overleg met de promotor bijlagen in de scriptie opgenomen worden of als apart boekdeel voorzien worden.

Er wordt wel steeds een lijst met vermelding van alle bijlagen opgenomen in de scriptie. Bijlagen worden genummerd het een drukletter A, B, C,...

Voorbeelden van bijlagen:

Bijlage A: Detailtekeningen van de proefopstelling

Bijlage B: Meetgegevens (op USB)

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
CAMPUS BRUGGE
Spoorwegstraat 12
8200 BRUGGE, België
tel. + 32 50 66 48 00
iiw.brugge@kuleuven.be
www.iiw.kuleuven.be