# Scalable Hardware Accelerator for Comparing DNA and Protein Sequences

Philippe Faes, Bram Minnaert, Mark Christiaens, Eric Bonnet, Yvan Saeys, Dirk Stroobandt, Yves Van de Peer

*Abstract*— **Comparing genetic sequences is a well-known problem in bioinformatics. Newly determined sequences are being compared to known sequences stored in databases in order to investigate biological functions. In recent years the number of available sequences has increased exponentially. Because of this explosion a speedup in the comparison process is highly required. To meet this demand we implemented a dynamic programming algorithm for sequence alignment on reconfigurable hardware. The algorithm we implemented, Smith-Waterman-Gotoh (SWG) has not been implemented in hardware before. We show a speedup factor of 40 in a design that scales well with the size of the available hardware. We also demonstrate the limits of larger hardware for small problems, and project our design on the largest Field Programmable Gate Array (FPGA) available today.**

## I. Introduction

In this paper we present a scalable accelerator for comparing protein sequences. Comparing protein sequences is a very computationally expensive operation that is often performed in the field of bioinformatics. A typical operation would be to compare one newly determined sequence with each sequence in a database, and calculate their similarities. One such database, the NCBI protein database [1], contains almost 3 million protein sequences with lengths ranging from 6 to 36805 amino-acids. The computational complexity of one comparison is of the order $O(N_1 N_2)$, the product of the length of the two sequences.

An algorithm that is often used for comparing sequences is the Smith-Waterman[2] algorithm, which was extended by Gotoh[3]. Our implementation of this Smith-Waterman-Gotoh (SWG) algorithm can compare two sequences of length 1024 in 50 ms on a modern desktop computer. We have been able to accelerate this operation with a factor 40, using a FPGA.

Previous hardware solutions [4], [5] simplify the Smith-Waterman algorithm by setting many of the parameters to a fixed value, by only allowing DNA comparisons, and by not implementing the Gotoh extension. We put only very weak restrictions on the length of the protein sequences, support the Gotoh extension and provide the full flexibility of arbitrary substitution matrices and insertion penalties.

Even with the current FPGAs, we can accelerate the algorithm by a factor 40.

We demonstrate the scalability by projecting our design onto an FPGA which is seven times bigger than the FPGA we used for our initial development.

## II. Protein comparison

In this section we will present the basic principles of protein comparison, the Smith-Waterman algorithm and its extension presented by Gotoh.

### A. Alignment

A well known problem in bioinformatics is the comparison of Desoxyribonucleic Acid (DNA) sequences and of protein sequences. The former are represented by strings of nucleotides, where possible nucleotides are A, C, G, and T, an alphabet of four symbols. The latter are represented by a string of amino-acids. The 20 possible amino-acids are also represented by roman capital letters, forming an alphabet of 20 symbols. We will focus on comparing protein sequences, but the algorithm and the acceleration we describe is identical when comparing DNA sequences. The calculation for DNA sequences is considerably easier, because of the shorter alphabet.

Comparing genetic sequences is based on the properties of mutations. Over time a sequence mutates and three elementary mutation operations are considered: nucleotides or amino acids can be inserted, deleted or substituted.

Mutations can be visualized by an *alignment*. For example a possible alignment between the protein sequences `AGTTAT` and `TTATATTT` is

```
--AG-TTAT
  |  || |
TTATATT-T.
```

The *edit distance*[1] of an alignment expresses the similarity of the two sequences. It is calculated as the sum of the matching scores of individual characters (positive for matches and usually negative for substitutions) and the penalties of the gaps. For the above example the edit distance is

$$M(\mathtt{A},\mathtt{A}) + M(\mathtt{G},\mathtt{T}) + 3M(\mathtt{T},\mathtt{T}) - 2G(1) - G(2), \qquad (1)$$

with $M$ the so-called *substitution matrix*, which contains the matching scores and mismatch penalties and $G$, the *gap penalty function*.

### B. Smith-Waterman Algorithm

The Smith-Waterman algorithm [2] uses dynamic programming to find the *optimal*, *local* alignment. This is the alignment of two subsequences which produces the maximum edit

[1]This is a misnomer in existing literature: the value we calculate does not have the axiomatic properties of a "distance". A better term would be *similarity*, since higher values represent better matches. We will stick with the conventional term *distance*.