



Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

# 37. OTDK BIOLÓGIA SZEKCIÓ

## Optimizing single $\alpha$ -helix detection algorithm on FPGA

KERÉKGYÁRTÓ, Márton

Bioinformatics MSc

Year I

Supervisors: Dr. GÁSPÁRI, Zoltán, Dr. NAGY, Zoltán

2025



# Contents

|          |                                                                                        |           |
|----------|----------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                    | <b>1</b>  |
| 1.1      | Single $\alpha$ -helices and other low complexity motifs . . . . .                     | 1         |
| 1.2      | The architecture and programming of FPGAs . . . . .                                    | 3         |
| 1.3      | The FT_CHARGE method and the architecture of its implemen-<br>tation on FPGA . . . . . | 3         |
| <b>2</b> | <b>Optimizing the FPGA solution</b>                                                    | <b>6</b>  |
| 2.1      | Theoretical maximum speed of the FPGA . . . . .                                        | 6         |
| 2.2      | Optimizing the client . . . . .                                                        | 7         |
| 2.2.1    | Accelerating gzip decompression . . . . .                                              | 7         |
| 2.2.2    | Parallelization and optimization . . . . .                                             | 7         |
| 2.2.3    | Achieved performance . . . . .                                                         | 9         |
| 2.2.4    | Scalability of the client . . . . .                                                    | 9         |
| 2.3      | Optimizing the server . . . . .                                                        | 10        |
| 2.4      | Speed and efficiency of the FPGA compared to other architectures                       | 11        |
| <b>3</b> | <b>Optimizing the parameters of the algorithm and evaluating the<br/>results</b>       | <b>13</b> |
| 3.1      | Double peaks in the spectra . . . . .                                                  | 13        |
| 3.2      | Optimizing parameters . . . . .                                                        | 14        |
| <b>4</b> | <b>Analysis of large datasets using the FT_CHARGE method</b>                           | <b>19</b> |
| 4.1      | The Uniprot Archive database . . . . .                                                 | 19        |
| 4.2      | Analyzing metagenomic databases . . . . .                                              | 22        |
| <b>5</b> | <b>Conclusions</b>                                                                     | <b>25</b> |
| 5.1      | Summary . . . . .                                                                      | 25        |

|     |                        |           |
|-----|------------------------|-----------|
| 5.2 | Results . . . . .      | 25        |
| 5.3 | Future plans . . . . . | 25        |
|     | <b>Bibliography</b>    | <b>29</b> |
|     | <b>A Source codes</b>  | <b>30</b> |

# Chapter 1

## Introduction

### 1.1. Single $\alpha$ -helices and other low complexity motifs

In contrast to globular proteins, some sequences contain low variability of amino acids or repeating sections. These are considered low-complexity sequences, and are characterized by the different amino acid distribution compared to globular proteins. Although the motifs themselves are low complexity, there is no exact definition and multiple methods with different results are available for their identification.[1]

Two low complexity motifs are the polyproline II helix and the coiled-coil. Both contain repeating patterns that are regular enough that they can be described by mathematical equations. Coiled coils are built up from 2 to 12 helices coiled around each other, the individual chains contain repeating segments that are usually 7 amino acids long[2]. The polyproline II helix consists of 3 polypeptide chains containing repeating proline-proline-glycine residues, which is the only sequence capable of creating a stable triple helix[3].

Single  $\alpha$ -helices are another type of low-complexity motifs. Their most notable property is that – in contrast to other proteins – they are capable of forming stable single helical folds in water even in their monomeric form, which was first described in the case of myosin X [4], and is usually proved by circular dichroism (CD) spectroscopy. The reason for this behaviour is the high abundance of charged

residue repeats that stabilize the structure. The three most common charged residues in SAHs are arginine (R), glutamine (E) and lysin (K), which can form salt bridges. It has been demonstrated that salt bridges between Arg and Glu result in more stable SAHs, than Glu-Lys bridges. This means that based on the ratio of Lys and Arg, the stability of the SAH can be predicted to some extent. These ratios for four SAH regions in different proteins are provided in table 1.1[5].

| Protein name | ID     | SAH start | SAH end | K  | R  | R/K   | Reference |
|--------------|--------|-----------|---------|----|----|-------|-----------|
| CAF-1        | Q12495 | 136       | 226     | 24 | 14 | 0.583 | [6]       |
| Myosin 6     | Q9UM54 | 915       | 980     | 12 | 15 | 1.25  | [7]       |
| SFPQ         | P23246 | 538       | 604     | 2  | 16 | 8     | [8]       |
| Myosin 7     | P97479 | 866       | 935     | 11 | 11 | 1     | [9]       |

Table 1.1: The ratios of arginine and lysine residues in case of different, well-examined proteins

915 KKKQQEEEAE<sup>+</sup>RLRR<sup>+</sup>IQEEM<sup>+</sup>EKERKR<sup>+</sup>REDEKRR<sup>+</sup>  
947 RKEEEE<sup>+</sup>RRMKLEMEAKRKQ<sup>+</sup>EEEE<sup>+</sup>RKKREDDEK<sup>+</sup>R

Figure 1.1: The sequence of the SAH in Q9UM54. The numbers on the left refer to the positions in the sequence, red letters show positively, blue letters show negatively charged amino acids

Being relatively recently discovered, and hard to examine using traditional methods like NMR or X-ray crystallography, the number of experimentally characterized SAHs is low. Their role is mostly reported to be mechanical, such as lever arms on myosins IV and X[10].

There are several available methods for predicting low-complexity motifs. These usually use hidden Markov-models (HMM) or position specific scoring matrices (PSSM), like the COILS method for the prediction of coiled coils. Waggawagga is a method that combines the results from multiple prediction methods, and can be used for coiled coils and SAHs[11]. The FT\_CHARGE method, specifically developed for the detection of SAHs, and subject to my work described here, will be detailed below.

## 1.2. The architecture and programming of FPGAs

Field Programmable Gate Arrays (FPGAs) are integrated circuits containing basic logic elements and configurable interconnects. The advantage of these devices is that their hardware can be freely modified to implement the most optimal configuration for any given task. Their building blocks are look-up-tables, which can be configured to implement any logic gate with  $n$  inputs (usually 6) and one output; flip-flops, which can store 1 bit of data; block RAMs, which are higher in capacity, but can only access one element each clock cycle; and DSP blocks, which are highly efficient multipliers, as multiplication is a very common task, and by hardening it, usage of the reconfigurable fabric can be reduced. In addition to the programmable logic (PL), many FPGAs nowadays have a CPU embedded into the same chip to ease interfacing with the PL. This CPU is usually called the processing system (PS)[12].

The biggest drawback of FPGAs is the need to design hardware rather than using high-level programming languages to describe the algorithm. Initially, designing hardware meant the time-consuming process of manually drawing schematics of the hardware elements and their physical layout. In order to automate the design process, hardware descriptor languages (Verilog, VHDL) have appeared, where only a register transfer level (RTL) description of the logic is required, from which a computer can generate the schematics and physical layout of the circuit. High-level synthesis (HLS) aims to simplify this process even more by taking a high-level (C/C++) code as an input, from which – with the guidance of directives – an RTL-level description is generated. In the case of this project, the output of the HLS is a hardware description, which – when programmed onto the FPGA – is visible by the processing system of the FPGA as an OpenCL kernel.

## 1.3. The FT\_CHARGE method and the architecture of its implementation on FPGA

The traditional, similarity-based algorithms, such as BLAST are optimized for globular proteins, not low-complexity regions. A possible way of identifying SAHs

is the SCAN4CSAH method. This identifies SAHs based on scoring the identically and oppositely charged amino acids separated by given distances. A more restrictive algorithm is the FT\_CHARGE method, developed by Gáspári et al.[13]. This method applies Fourier transform (FFT) to the charges of amino acids of a sequence, and the amplitude and position of the maximum can be used to determine whether or not a SAH is present. The algorithm uses sliding 32- and 64-element wide windows to cover the whole sequence in one-element steps. To increase the sensitivity of the method, the Fourier transform is applied to a charge correlation function instead of the sequence itself. The correlation function is in equation 1.1.

$$U(n) = \sum_{i=1}^m c(i) \times c(i+n) \quad (1.1)$$

where  $m$  is the window size and  $c(i)$  is the charge of the  $i$ th residue. As in single  $\alpha$ -helices, 3 to 4 identically charged amino acids are usually followed by 3 to 4 amino acids of the opposite charge, a peak inside the  $1/9$  and  $1/6$  frequency window is considered a possible SAH. The maxima of the spectra are fitted to an Extreme Value Distribution (EVD) table to determine if the section is a single  $\alpha$ -helical motif by calculating a p-value. Due to the 64-element wide FFT, proteins shorter than 64 amino acids are filtered out, but finding SAHs in such short sequences is very unlikely[14].

The input of the FPGA implementation is a vector consisting of 2-bit wide elements. Each element represents the charge of an amino acid from the sequence of the proteins. As the possible charges are 1 (glutamic acid, aspartic acid), -1 (arginine, lysine), 0.5 (histidine), and 0 (all other proteinogenic amino acids), a significant reduction by a factor of 4 is achievable compared to using amino acids stored as 8-bit characters.

The 32- and 64-element wide FFTs are calculated parallelly. First, the correlation is calculated, where, taking advantage of the sliding windows with one element stride, a cumulative sum over the entire sequence is used. This means that for every new index, only one addition is required to calculate the cumulative sum, and by subtracting the value of the cumulative sum 32 or 64 positions before the current position, the desired sum over a window can be acquired. This means that for every clock cycle, a new element of the charge correlation can be fed into the FFT calculator, which has a pipelined architecture to ensure it can also accept

a new vector every clock cycle. On the FPGA, 18-bit fixed-point numbers are used, as the DSP slices support  $27 \times 18$  bits of input. To synthesize a block capable of executing these calculations, 668 DSP slices are required, which is likely going to be the limiting factor for the number of implementable blocks on most FPGAs.

The programmable logic of the FPGA is connected by an AXI bus to an Arm Cortex A53 CPU. Since the capabilities of the Arm CPU wouldn't be sufficient to fully utilize the capabilities of the FPGA, it merely receives the charge sequence on a gigabit Ethernet interface and sends back the results calculated by the programmable logic. A separate x86 computer is running the client, which extracts the gzip compressed sequence files, converts the characters coding amino acids into charges encoded in 2 bits, and sends them to the server in blocks of 64 megabytes which contain the sequences (the ones longer than 64 elements) concatenated after each other, without any headers.

## Chapter 2

# Optimizing the FPGA solution

The algorithm was optimized for the AMD ZCU102 FPGA, as this was available during development, while the client was optimized on a Ryzen 5 3600 CPU. These are mid-range devices from a few years ago, on more powerful hardware further acceleration would be possible.

### 2.1. Theoretical maximum speed of the FPGA

The ZCU102 has enough hardware elements to implement three separate instances of the calculating blocks. The limiting factor is the number of DSP slices required for the calculations of the multiplication-intensive FFT. With each block requiring 668 of them, the utilization of the 2520 slices on the FPGA is 87.6 %. Each block is capable of processing one element of a sequence every clock cycle, and the achievable maximum operating frequency was 200 MHz, meaning the theoretical maximum of the FPGA is processing 600,000,000 amino acids each second. This cannot be achieved due to the maximum speed of the interface between the PC running the client and the server on the FPGA being only 1 Gb/s (125 MB/s), whereas  $200 \text{ MHz} \times 2 \text{ bits} \times 3 / 8 = 150 \text{ MB/s}$  bandwidth would be required for perfect utilization. Scaling this implementation is relatively straightforward; on a larger FPGA, more instances can be placed which run independently, the only limiting factor is the size of the FPGA.

A possible solution to the bottleneck of the network interface would be encoding charges using variable length codes. As the charge of most amino acids is zero,

coding this as a single 0 bit, while coding other values using 2 to 3 bits would reduce the required bandwidth, but would require more complex encoding and decoding algorithms.

## **2.2. Optimizing the client**

### **2.2.1 Accelerating gzip decompression**

Although the task of the client doesn't require huge amounts of CPU resources compared to the actual calculations, a bottleneck was still present on the client side. Data downloaded from sequence databases is compressed with gzip in most cases. The original implementation used single-threaded gzip decompression and couldn't send data fast enough to the FPGA. For many years, gzip decompression was deemed unparallelizable due to the back-references inserted by the LZ77 step of the DEFLATE algorithm, which means that in order to decompress any given byte, one might need the very first byte of the file.

Fortunately, a parallel gzip decompressor called Rapidgzip already exists[15], which was easy to embed into the existing code by redirecting the decompressed data to the standard output and using the `popen()` C function to read it. The `popen()` function returns a `FILE*` pointer, which means that it is not possible to use C++ constructs, such as streaming with it, but in this case, this is actually not a drawback, because at the required speed the overhead of streams would be a bottleneck. By using `FILE*` pointers, it is possible to optimize the code for fasta files, without any unnecessary operations that would slow the program down.

### **2.2.2 Parallelization and optimization**

As the speed of the client still was not sufficient to fully utilize the FPGA, the client had to be further parallelized. The tasks of the client are reading the data from the decompressed file, converting it to a sequence of charges coded as 2-bit wide elements, and sending the data. Reading and sending the data cannot be parallelized as the file and Ethernet interfaces are inherently serial, but they can be run by separate threads, utilizing the multi-core architecture of the CPU. As

the client sends the data in blocks of 64 megabytes, it is possible to parallelize the middle step, by converting multiple blocks at the same time. In order to be able to achieve this, the boundary between the blocks has to be known before starting the conversions.

The input data is split into 256-megabyte parts, and the processing of each is executed parallelly. As it is probable that the sequence boundaries are not aligned to 256 megabytes in the file, the blocks are somewhat shorter than 256 MB, and the leftover data is copied to the start of the next block. Finding the start of the last sequence before the 256 MB boundary is required, but as the length of each sequence is negligible compared to the size of the block, this isn't a concern regarding the speed of the program.

The 256 MB blocks contain the raw fasta files, meaning both sequence and header data. The 256 MB size is chosen because after converting the data from 8-bit to 2-bit elements, the resulting block must be no more than 64 MB. Depending on the ratio of the length of sequences and headers (and sequences shorter than 64, which need to be skipped too) in the input fasta files, the resulting block is going to be less than 64 MB. This can be a problem, because the different-sized blocks take different amount of time to be executed on the FPGA, leading to it not being fully utilized. This is a good compromise, however, as creating blocks of exactly 64 MB can't be parallelized, and the under-utilization problem would still persist.

The usage of dynamically resized containers was also a bottleneck. By default, for every element appended to a C++ vector or string, the size of the container has to be compared to the number of elements already in it, and, if required, a larger memory area is allocated, and the existing data is copied to it. In a general case, this is required to ensure safe memory management, but for this specific case, it is possible to have safe memory management using fewer checks, which means faster program execution, at the cost of having to use pointers and manually ensure correct memory management. To use as little memory bandwidth as possible, the program stores blocks of the fasta files in memory. These blocks contain both headers and sequence data, and a separate index is created for fast access to any sequence. A drawback of this solution is that in fasta files, a single sequence can be stored over multiple lines, meaning when using the stored data, handling the

line breaks inside sequences is required.

A known limitation is that if the server can't accept data as fast as the client can send, the step blocking the execution is the server dropping the TCP packets sent by the client. This works, but wastes resources by retransmitting a packet until the server can receive it, so a packet to signal if the server is ready to accept data would be favorable. To implement this, a new socket would be required, and as for now this isn't a problem, it remains unchanged.

### 2.2.3 Achieved performance

To measure the maximum performance of the client, a modified server was used, which accepts the data, but immediately sends back an empty result set, simulating an infinitely fast server. The simulated server and the client run on the same computer to eliminate the bottleneck of the network interface. On an AMD Ryzen 5 3600 CPU, processing the 2024\_03 release of the UniProt TrEMBL database (which contains 86,585,019,224 amino acids) was finished in 3 minutes, 12.5 seconds. This means the achieved speed was 449,792,308 amino acids per second, while the theoretical maximum speed of the gigabit ethernet interface would be  $1000 \times 1000 \times 1000 / 2 = 500,000,000$  amino acids per second. It has to be noted that the actual number of amino acids sent is lower due to the exclusion of short sequences, but still the performance of the client on this CPU is probably sufficiently high not to be a major bottleneck.

### 2.2.4 Scalability of the client

The client is probably capable of higher speeds on CPUs with more cores, although to what extent could not be tested with the current equipment. To be able to scale the implementation even further, offloading the task of the client to the FPGA has also been considered. Although there exist implementations of FPGA based gzip decompression, a single decompressor – according to AMD[16] – on an FPGA is only capable of 750 MB/s under ideal conditions (this claim could not be tested, as it would require rewriting their example code which currently only supports files smaller than 4 GiB), while rapidgzip was able to achieve 1250 MB/s on the TrEMBL database. This means that scaling decompression on the FPGA is only

possible if the input data is split across multiple files. This isn't the case for the UniProt TrEMBL database, but larger ones, like the UniProt Archive or the MGnify metagenomics database, can be scaled that way. The other problem is that, in the current implementation, the FPGA only handles charges as inputs, but in order to be able to use the result, the sequence headers, and the sequence in the region of SAHs are also required. This is currently handled by the client, and modifying the FPGA circuit to use variable-length data is not trivial. Nonetheless, if using a larger FPGA, this might be necessary, but for now, the performance of the x86 client is sufficient.

Another option would be to keep the client on a more powerful x86 computer, but using a faster network connection. The drawback of this is that since 10 gigabit and faster interfaces are only connected to the programmable logic, a circuit would also have to be designed to receive the data. Another option would be to use a PCIe FPGA card, like an AMD Alveo U250, but this would require a powerful computer, with access to sufficiently large and fast storage, that also has an FPGA card inside it.

## 2.3. Optimizing the server

The task of the server is to receive the data from the client, move the data to the programmable logic of the FPGA, start the calculation, and copy back the results that are sent back to the client.

To be able to utilize the 3 computing blocks on the FPGA, the data movements and calculations need to overlap. Currently, the server receives three blocks of data, starting the calculation of each block before receiving the next one. After receiving the third and the completion of the first calculation, the results are sent back. This serial implementation is not perfect, it would be possible to have three separate threads for sending and receiving the data, and the handling of the programmable logic. As the performance improvements wouldn't be huge, so far this isn't implemented, but is planned in the future.

## 2.4. Speed and efficiency of the FPGA compared to other architectures

Currently, the comparable implementations are running on an x86 CPU and an FPGA. The possibility of a GPU implementation has been previously rejected by a previous student working on the same topic[17], because the very short windows could not be efficiently handled by the cuFFT library. Since then, a new library called cuFFTDx (CUDA FFT Device eXtension) has been released, which allows greater control, and it might be possible to achieve speeds that are on par with the FPGA. Due to the time requirements of such an optimization, this is yet to be tested.

The CPU implementation uses FFTW to calculate the Fourier transforms, which is a very well-optimized library. However, it has to be noted, that the time required to analyze sequences does not grow linearly. The runtime for a single long sequence can be significantly longer than if the same sequence was split up into parts, this is likely due to cache of the CPU being able to fit the shorter sequences, but not the longer ones. It would be possible to fix this, but further optimization of the CPU implementation was outside the scope of the current project. Based on this, a rough estimate for the speed of the CPU can be acquired, but it has to be noted that it would probably be possible to have a somewhat faster solution.

To analyze the 2024\_03 release of the UniProt TrEMBL database, the FPGA took 384.35 seconds (6 minutes, 24.35 seconds), while the CPU was running for 56518 seconds (15 hours and 41 minutes), meaning the FPGA was significantly faster by a factor of 147. Although in most cases it would be advisable to take the average of multiple runs, the database used here is large enough that the averaging happens during runtime. According to the reports of the synthesis, power dissipation of the FPGA circuit was 10.8 W, the TDP of the CPU was 65 W, based on which a rough upper limit for the consumed energy can be estimated. The FPGA in itself used 1.15 Wh, but the power consumption of the client (6.94 Wh) also has to be taken into account, which gives 8.09 Wh total. The CPU used 1020 kWh, here the FPGA has a similarly impressive advantage.

The TrEMBL database contains 86,585,019,224 amino acids. With perfect

utilization of the FPGA, the runtime would be 145 seconds. Based on this, the current utilization of the FPGA is 37.72 %.

## Chapter 3

# Optimizing the parameters of the algorithm and evaluating the results

### 3.1. Double peaks in the spectra

One problem encountered during development was the differing results between the x86 and the FPGA implementations. The reason for this is that while the x86 implementation used double-precision floating-point numbers, the FPGA didn't have enough DSP slices to fit even one instance of the calculating block when using floating-point numbers, so 18-bit wide fixed-point numbers had to be used instead.

The fact that the small differences due to the precision of number representation were enough to decide whether some hits were considered positive or not meant that the robustness of the algorithm could be improved. Upon examining the spectra of such cases, where the x86 and FPGA results didn't match, it became obvious that the reason for the difference is that – in most of the cases – the spectra contained two distinct peaks with very similar amplitudes, an example for which can be seen on figure 3.1. While one of the peaks was inside the frequency window where positive hits are considered to be, the other was not. Further examination showed that this problem is not exclusive to the FPGA implementation. The problem is present regardless of the datatypes, only in the case of double precision numbers it remained unnoticed.

As in this case basically only the inaccuracy of number representation decided

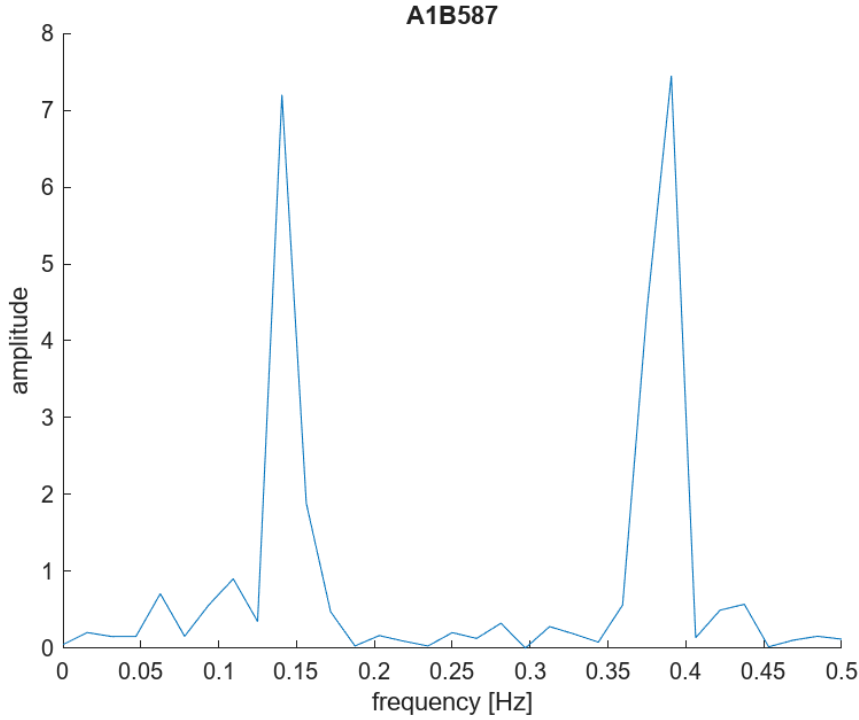


Figure 3.1: The spectrum of a protein (Translation initiation factor IF-2) containing two nearly identical peaks (window between amino acids 85 and 149)

whether there would be a hit or not, it was decided to modify the algorithm to only consider the values inside the  $\frac{1}{9} - \frac{1}{6}$  frequency window; if the maximum of them is larger than 7, it is considered a possible hit, regardless of the other values outside the window. The reasoning behind this is that the existence of a peak means that the stabilizing structures are probably present in the sequence.

In some cases, there were multiple peaks outside the frequency window, and some were considerable larger than the one inside the window, especially near the ends of the spectrum. The number of these is small compared to the overall number of sequences. Regardless, it might be informative to examine these sequences.

## 3.2. Optimizing parameters

As SAHs are a relatively newly discovered motif, there aren't many experimentally characterized segments that can be used for optimization. The list of currently known SAHs is provided in table 3.1. Previous studies have concluded that the optimal parameters for the algorithm are 7 for the minimum amplitude of the

| <b>Protein name</b> | <b>UniProt ID</b> | <b>SAH start</b> | <b>SAH end</b> | <b>Reference</b> | <b>Original parameters<br/>(<math>A \geq 7</math>,<br/><math>P \leq 0.05</math>)</b> | <b>New EVD parameters</b> |
|---------------------|-------------------|------------------|----------------|------------------|--------------------------------------------------------------------------------------|---------------------------|
| Caldesmon           | A0A1L1RXH5        | 196              | 252            | [18]             | 174-316,<br>408-521                                                                  | 174-317,<br>408-521       |
| GCP60               | Q9H3P7            | 183              | 238            | [13]             | 158-274                                                                              | 155-274                   |
| INCENP              | P53352            | 503              | 715            | [19]             | 502-679                                                                              | 502-655                   |
| MAP4K4              | O95819            | 417              | 480            | [13]             | 374-503                                                                              | 374-503                   |
| MFAP1               | P55081            | 267              | 344            | [20]             | 225-259                                                                              | 225-259                   |
| Moysin 10           | Q9HD67            | 813              | 909            | [21]             | 773-870                                                                              | 772-872                   |
| Moysin 7            | P97479            | 866              | 935            | [9]              | 848-929                                                                              | 857-928                   |
| Myosin 6            | Q9UM54            | 915              | 980            | [7]              | 932-995                                                                              | 909-1015                  |
| Snu23               | G0S6R0            | 131              | 164            | [20]             |                                                                                      |                           |

Table 3.1: The list of experimentally characterized SAHs

highest peak and 0.05 for the probability value of that peak compared to other sequences.[14]

The first step in optimizing the parameters was the regeneration of the EVD parameters used for the calculation of the p value of a hit. As the amplitude of a peak in the spectrum has no biological meaning in itself, it has to be compared to amplitudes from other sequences. The amplitude mostly depends on the number of charged residues in a sequence, so in order to get meaningful results, sequences with similar charge compositions have to be compared.

In the original solution, extreme value distributions were fitted to randomly generated sequences containing differing amounts of positively and negatively charged amino acids in 10 % increments. In order to get more accurate results, the EVD parameters were regenerated, but for every possible combination of positive and negative charges, rather than 10 % increments. The benefit of this is that every segment is now compared to one with exactly the same number of charged residues. Both 32- and 64-bit comparisons need their own set of values. For both, 5000 random sequences were generated for every charge composition by first creating a vector containing the required number of elements, then using the Fisher-Yates algorithm to shuffle the elements. In contrast to generating the sequences randomly

one element at a time, this method ensures that the number of charged residues is exactly as required.

Although the start and end positions of experimentally characterized SAHs are marked in the table, it would not be reasonable to expect the FT\_CHARGE algorithm to be able to fully reproduce these – such a strict boundary cannot be exactly determined, the different researches most probably used different methods.

In the case of Caldesmon, the start and end positions designate the part of the sequence that was synthesized for examination[18], this does not mean, however, that the SAH cannot continue beyond it. In another paper by the same group, it is stated that the helical structure continues to what is position 324 in A0A1L1RXH5[22]. The second hit can be omitted, as based on its position it is in a part of the protein that was not discussed in the papers cited.

The shortest known SAH, in Snu23 is missing from the results. According to the authors, the helical propensity of Snu23 is 20 %, which is very low, and with p values around 0.35 at the required positions, tuning the algorithm to find this would result in a huge drop in specificity.

The algorithm gave a shorter hit in case of the inner centromere protein (INCENP). Interestingly, in this case, p values throughout the entire sequence were mostly higher than before. According to the research paper, the SAH consists of two separate parts, out of which the first one is reliably detected, but the end of the second one (starting from position 598) is missing. This might be due to the different distribution of charged residues in the second part, as it is visible on figure 3.2, where the amino acids have been coloured by the charge. A possible explanation for this is that although the sequence at the end of the SAH in itself would not be capable of forming a stable fold, the preceding SAH stabilizes this part as well. This is not a concern, as the existence of a SAH can be identified, only its length will be shorter.

Another subtle but important effect of the new EVD parameters was that in most of the sequences, the p value near the start and end of SAHs increased, but in the middle, it decreased. This means the specificity increased, which is in accordance with the goal of reducing the number of false positive hits.

To evaluate the specificity of the changes, a set of negative control sequences

503 KEKERQKLQALRKKEEAEQLRKQVEEEKKRRQEEAKLRR  
 543 EERLRKVLQARERAEQLEEEKRRIEQKLALFDEKTEKAR  
 583 EERLAEKIKKRAAAKKMEEAEARRRQDEEARKQKALQQE  
 623 EEERRHKELMQKKKEEQERARKIAEQRQAEQereqqlaa  
 663 ereqerkkeqerkkeee

Figure 3.2: Comparison of the detection results for the SAH in sequence P53352 (INCENP) showing the difference between the old and the new parameters. The numbers on the left refer to the positions in the sequence. The section only identified by the old one is marked with lowercase letters.

was used. The tables 3.2 and 3.3 contain the lowest p values and highest amplitudes found anywhere in the sequences. As the amplitudes are low for every sequence, it is not a concern that a false negative hit would occur. The only exception is the case of Q8IXT5, which contains large amounts of proline residues, which inhibit the protein's ability to form a stable SAH. These kinds of sequences are filtered out at a later step, it is acceptable that FT\_CHARGE gives high amplitudes and low p values.

| Protein name                         | UniProt ID | min p | A for min p | max A   | p for max A |
|--------------------------------------|------------|-------|-------------|---------|-------------|
| Myoglobin                            | P02144     | 0.023 | 1.068       | 1.653   | 0.039       |
| GCN4                                 | P03069     | 0.066 | 0.686       | 0.777   | 0.499       |
| Myosin 2                             | Q9UKX2     | 0.007 | 1.514       | 3.869   | 0.014       |
| Kinesin-related protein 12           | Q6S000     | 0.034 | 0.732       | 0.732   | 0.034       |
| p53                                  | P02340     | 0.083 | 1.124       | 1.153   | 0.199       |
| Golgi SNAP receptor complex member 1 | Q62931     | 0.054 | 0.578       | 0.826   | 0.143       |
| RNA-binding protein 12B              | Q8IXT5     | 0.000 | 100.476     | 109.663 | 0.000       |
| Zinc finger protein 609              | O15014     | 0.006 | 1.824       | 2.568   | 0.196       |
| Calpastatin                          | P20810     | 0.023 | 0.886       | 1.569   | 0.216       |

Table 3.2: Results for negative control sequences using the original parameters

Comparing the results between the original and the modified parameters, in most cases, the amplitudes grew, which is the result of changing the charge of histidine to 0. Even so, p values increased in most cases, meaning that although

| <b>Protein name</b>                     | <b>UniProt ID</b> | <b>min p</b> | <b>A for<br/>min p</b> | <b>max A</b> | <b>p for<br/>max A</b> |
|-----------------------------------------|-------------------|--------------|------------------------|--------------|------------------------|
| Myoglobin                               | P02144            | 0.081        | 1.217                  | 1.217        | 0.081                  |
| GCN4                                    | P03069            | 0.052        | 0.526                  | 1.023        | 0.107                  |
| Myosin 2                                | Q9UKX2            | 0.025        | 0.940                  | 3.869        | 0.044                  |
| Kinesin-related protein 12              | Q6S000            | 0.009        | 0.327                  | 0.932        | 0.587                  |
| p53                                     | P02340            | 0.049        | 0.002                  | 1.215        | 0.212                  |
| Golgi SNAP receptor<br>complex member 1 | Q62931            | 0.151        | 0.944                  | 0.987        | 0.204                  |
| RNA-binding protein 12B                 | Q8IXT5            | 0.000        | 10.401                 | 98.914       | 0.000                  |
| Zinc finger protein 609                 | O15014            | 0.005        | 0.695                  | 2.666        | 0.269                  |
| Calpastatin                             | P20810            | 0.078        | 0.060                  | 1.569        | 0.253                  |

Table 3.3: Results for negative control sequences using the modified parameters

even the original parameters were sufficient, using the new parameters increased the specificity further.

## Chapter 4

# Analysis of large datasets using the FT\_CHARGE method

### 4.1. The Uniprot Archive database

The UniParc database contains protein sequences from multiple public databases. Each entry contains cross-references to every database in which the given entry exists or existed. Entries are never deleted from UniParc, even if they are removed from the original database. This means, that even if a sequence is less reliable and therefore removed from the source database, it stays in UniParc[23].

In the database 1,039,679 SAH regions were identified from 983,988 different sequences. As UniParc contains 833,496,420 entries, around 0.12% of them contained SAHs.

Compared to other, previously analyzed databases, the longest sequences in UniParc were significantly longer than any previously identified SAH. With lengths between 10,000 and 5,000 for the ten longest ones, it is possible that these sequences would not be able to form stable SAHs. The sequences contain long identical repetitions, such as KKKxEEEx in the case of UPI001982BFB1. These – although in shorter sequences might form stable SAHs – are very long, and – the source being the UniParc – might stem from not very reliable data, and the existence of such proteins is questionable. The R/K and D/E ratios are provided in table 4.1. In all cases, the number of arginine residues is lower compared to lysines, and this high R/K ratio might not be compatible with a very long stable SAH. The ratio of

D and E residues is always less than one, this means that there are more glutamine residues in the sequence, which are more capable of forming helices than aspartic acid.

| UniParc ID    | length | K    | R   | E    | D    | R/K   | D/E   |
|---------------|--------|------|-----|------|------|-------|-------|
| UPI001982BFB1 | 10775  | 3972 | 29  | 2464 | 1085 | 0.007 | 0.440 |
| UPI00206DFE49 | 10453  | 3854 | 1   | 2076 | 1289 | 0.000 | 0.621 |
| UPI00204A42D6 | 9951   | 3630 | 2   | 1969 | 1201 | 0.001 | 0.610 |
| UPI00255F9B37 | 7802   | 1063 | 870 | 2936 | 44   | 0.818 | 0.015 |
| UPI00256BD006 | 6853   | 1023 | 728 | 2513 | 59   | 0.712 | 0.023 |
| UPI0025626E47 | 6113   | 923  | 651 | 2259 | 38   | 0.705 | 0.017 |
| UPI000BE674EF | 5813   | 1807 | 29  | 1280 | 375  | 0.016 | 0.293 |
| UPI002E8F61C7 | 5329   | 1635 | 3   | 1308 | 186  | 0.002 | 0.142 |
| UPI00255FB2CB | 5292   | 746  | 589 | 1951 | 32   | 0.790 | 0.016 |
| UPI00255E9FAF | 5033   | 714  | 535 | 1866 | 44   | 0.749 | 0.024 |

Table 4.1: The 10 longest identified SAHs in UniParc, with their R/K and D/E ratios.

Looking at the distribution of the lengths of the identified SAHs on figure 4.1, the presence of two peaks for the 32 and 64 elements wide FFT windows is apparent. Obviously, sequences shorter than 32 or 64 cannot be identified by the respective window, this is the reason for the abrupt fall in the numbers. The X axis is logarithmic, to account for the very long sequences discussed above.

The ten most common species among the results are provided in table 4.2. It is important to note that the UniParc database is not nonredundant, meaning the same (or a very similar) sequence can appear multiple times, therefore these numbers don't necessarily reflect the actual number of distinct SAH containing proteins in each species. Although it would be tempting to conclude that the abundance of SAHs is high in these species, a much more probable explanation for their high occurrence is that these are the species most commonly used for genomics studies, and therefore proteins belonging to them are overrepresented in databases. A possible solution to this problem would be to normalize the occurrences by the overall number of sequences belonging to a given species, but this would introduce the possibility of skewing the results in the other direction, as there could be species with few proteins deposited, but containing SAHs.

Sequences in UniParc with the highest numbers of species associated to them

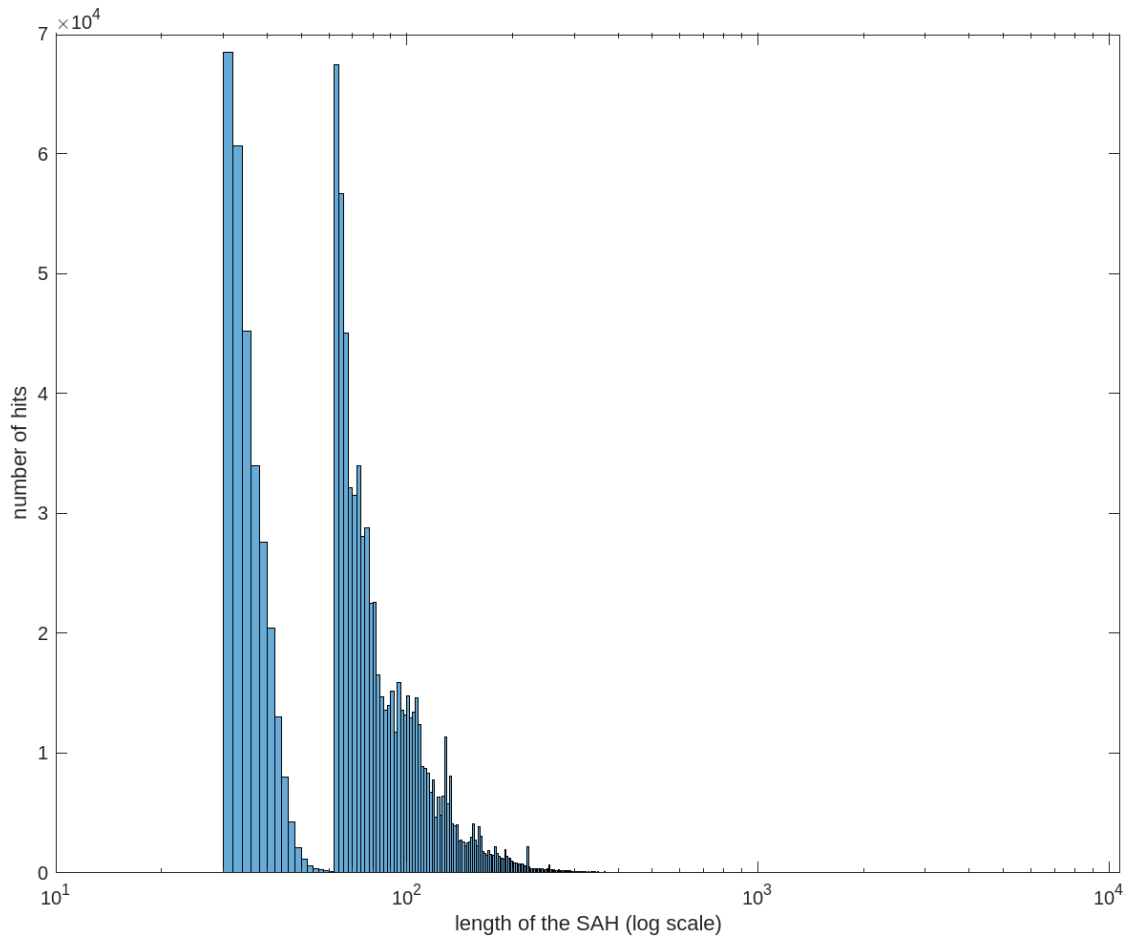


Figure 4.1: The histogram of the lengths of identified SAHs in UniParc

| occurrences | name                          |
|-------------|-------------------------------|
| 17397       | <i>Streptococcus</i>          |
| 9770        | <i>E.coli</i>                 |
| 7634        | <i>Bacillus cereus</i>        |
| 6316        | <i>synthetic construct</i>    |
| 6169        | <i>Helicobacter pylori</i>    |
| 5791        | <i>Salmonella enterica</i>    |
| 4379        | <i>Sus scrofa (pig)</i>       |
| 3653        | <i>Homo sapiens</i>           |
| 3402        | <i>Bacillus thuringiensis</i> |
| 3216        | <i>Cyprinus carpio</i>        |

Table 4.2: The number of occurrences for the 10 most common species containing SAHs

are UPI0000696BCC and UPI00000D7681 with around 500 species, but most of these are the subspecies of *Staphylococcus aureus*, so these results are not very interesting. Among the ones with most species is UPI00000373ED (152 references), annotated as Arginine and glutamate-rich protein 1. This is associated with many mammals, meaning that there is probably an evolutionary relationship behind this phenomenon. For further analysis, a way of filtering out the subspecies is required to get meaningful results without having to manually assess every case.

## 4.2. Analyzing metagenomic databases

Metagenomics is the study of samples from specific locations containing genetic material from a community of organisms. The amount of available data is vastly larger in contrast to curated databases, such as SwissProt, but the data is not well annotated, and the probability of sequencing errors is higher. Identifying SAHs in metagenomic data makes it possible to gather information about their prevalence in different taxonomical groups, which can be used to increase the precision of annotation of sequences.

The problem with using metagenomic data, is that the result of sequencing has to be translated from nucleotides to amino acids. There are many protein databases that are created using automated pipelines. A possible drawback of them is that they are optimized for globular proteins, and the low-complexity SAH containing sequences might be underrepresented. To overcome this, a possible solution is using the DNA sequences by translating them. The problem with this approach is deciding the exact way of the translation, such as the codon table to use, or what should happen with stop codons and non coding regions.

As a preliminary analysis, some species were examined using a brute-force method of translating every triplet to nucleotides, regardless of the presence of start or stop codons. The results will of course be less reliable, as, for example, a SAH might be identified after a stop codon, but as metagenomic data might contain sequencing errors, this can be justified as a first approximation.

First, three species were examined from the MGnify metagenomics database. MGnify contains both a nucleotide and a peptide database, the latter being gener-

| MGnify species<br>accession | number of hits |            | number of proteins<br>in genome |
|-----------------------------|----------------|------------|---------------------------------|
|                             | protein        | nucleotide |                                 |
| MGYG000002163               | 3              | 4          | 1662                            |
| MGYG000321269               | 2              | 2          | 1425                            |
| MGYG000331309               | 1              | 1          | 1845                            |

Table 4.3: Identified SAHs

```

Query  2   IDKKWRDTFAQEQLNRKRNEINKQIAQFKKDGENKKFEAAKKESSELKK  51
          ID  WR T  +   L  +RNEI+K +A+ KK+G+N + EA  +E   L  +
A5IMJ4 34  IDADWRTTITKTNELRSRRNEISKNVARLKKEGKNTEAEALIEEGKRLGE  83

Query  52   TVEDLKKQREEFEQKIGAL  70
          ++ L+++ +E ++++  L
A5IMJ4 84   EIKALEEKEKELQKQLNDL  102

```

Figure 4.2: The alignment between the SAH containing sequence from the metagenomic database and the SwissProt protein A5IMJ4

ated from the former and annotated using automated methods. The SAHs found in the three species are shown in table 4.3. There was a single hit, which was only identified in the translated nucleotide sequence (ITWDMSTTEERHQLEE-QIKAEDPKKWEEFKREMIR). The proteins containing the hits were annotated as hypothetical proteins, and a BLAST search on the single new hit resulted in a protein with 100 % coverage, that was also annotated as a hypothetical protein. The existence of such a protein in a database shows that the same or a similar sequence was identified as a protein coding gene. This proves, that there in fact are regions, which might be protein coding, but are missing from the peptide database. In conclusion, the number of hits was low, which was as expected, since SAHs are relatively rare, and the proteins are not very well annotated, which was also expected, as metagenomic data was used.

It is worth noting, that there were also some hits returned by the FT\_CHARGE algorithm that were filtered out, because they contained stop codons. A BLAST search did not return any similar sequences, which – considering the presence of stop codons – is not a surprise, it does mean, however, that, when using this method, false positive hits are to be expected.

A similar preliminary test was done using the genome of *Candidatus Lokiarchaeum sp.*, which was available in the Integrated Metagenomic database. Similarly to the previous case, 4 hits were identified from protein sequences, and 9 from the translated nucleotides, which were also hypothetical proteins. The only notable result was annotated as a serine-tRNA ligase, being somewhat similar (60% query cover, 33.33% identity) to the SwissProt A5IMJ4. Aligning them (shown in figure 4.2) shows some similar regions, and the positions of the possible SAHs (marked in red) overlap (The FT\_CHARGE algorithm did not identify a hit in A5IMJ4, only the less reliable scan4csah method). According the results of the Waggawagga web service, this region in A5IMJ4 is more likely a coiled coil rather than a SAH.

# Chapter 5

## Conclusions

### 5.1. Summary

The first objective of the study was optimizing the FPGA implementation of the FT\_CHARGE algorithm, and using it to examine large databases. This was accomplished by the optimizations described in chapter 2. The other objective was the analysis of large databases, described in chapter 4.

### 5.2. Results

On the FPGA side, a significant speedup was measurable compared to the CPU implementation, while on the biological part, SAHs were identified and analyzed in the UniParc database, and the foundations were laid for the analysis of metagenomic databases.

### 5.3. Future plans

Although further optimization of the FPGA implementation would be possible, but the speed is sufficient for the analysis of large databases. A more interesting next step is the in-depth analysis of the UniParc and other databases, with the possibility of using not only peptide, but nucleotide databases as well.

# Bibliography

- [1] P. Romero, Z. Obradovic, X. Li, E. C. Garner, C. J. Brown, and A. K. Dunker, “Sequence complexity of disordered protein,” *Proteins*, vol. 42, no. 1, pp. 38–48, Jan. 1, 2001, [Online; accessed 2024-11-13], ISSN: 0887-3585. DOI: `10.1002/1097-0134(20010101)42:1<38::aid-prot50>3.0.co;2-3`.
- [2] A. N. Lupas and M. Gruber, “The structure of alpha-helical coiled coils,” *Advances in protein chemistry*, vol. 70, pp. 37–78, 2005, [Online; accessed 2024-11-13], ISSN: 0065-3233. DOI: `10.1016/S0065-3233(05)70003-6`.
- [3] B. Brodsky and A. V. Persikov, “Molecular structure of the collagen triple helix,” *Advances in protein chemistry*, vol. 70, pp. 301–39, 2005, [Online; accessed 2024-11-13], ISSN: 0065-3233. DOI: `10.1016/S0065-3233(05)70009-7`.
- [4] P. J. Knight, K. Thirumurugan, Y. Xu, *et al.*, “The predicted coiled-coil domain of myosin 10 forms a novel elongated domain that lengthens the head,” en, *Journal of Biological Chemistry*, vol. 280, no. 41, pp. 34 702–34 708, Oct. 2005, ISSN: 0021-9258. DOI: `10.1074/jbc.M504887200`. [Online]. Available: <http://dx.doi.org/10.1074/jbc.M504887200>.
- [5] M. Wolny, M. Batchelor, G. J. Bartlett, *et al.*, “Characterization of long and stable de novo single alpha-helix domains provides novel insight into their stability,” en, *Scientific Reports*, vol. 7, no. 1, Mar. 13, 2017, ISSN: 2045-2322. DOI: `10.1038/srep44341`. [Online]. Available: <http://dx.doi.org/10.1038/srep44341>.
- [6] R. Rosas, R. R. Aguilar, N. Arslanovic, *et al.*, “A novel single alpha-helix dna-binding domain in caf-1 promotes gene silencing and dna damage sur-

- vival through tetrasome-length dna selectivity and spacer function,” en, *eLife*, vol. 12, Jul. 11, 2023, ISSN: 2050-084X. DOI: [10.7554/elife.83538](https://doi.org/10.7554/elife.83538). [Online]. Available: <http://dx.doi.org/10.7554/eLife.83538>.
- [7] B. J. Spink, S. Sivaramakrishnan, J. Lipfert, S. Doniach, and J. A. Spudich, “Long single  $\alpha$ -helical tail domains bridge the gap between structure and function of myosin vi,” en, *Nature Structural & Molecular Biology*, vol. 15, no. 6, pp. 591–597, May 30, 2008, ISSN: 1545-9993. DOI: [10.1038/nsmb.1429](https://doi.org/10.1038/nsmb.1429). [Online]. Available: <http://dx.doi.org/10.1038/nsmb.1429>.
- [8] L. Dobson, L. Nyitray, and Z. Gáspári, “A conserved charged single  $\alpha$ -helix with a putative steric role in paraspeckle formation,” en, *RNA*, vol. 21, no. 12, pp. 2023–2029, Oct. 1, 2015, ISSN: 1355-8382. DOI: [10.1261/rna.053058.115](https://doi.org/10.1261/rna.053058.115). [Online]. Available: <http://dx.doi.org/10.1261/rna.053058.115>.
- [9] H.-H. Li, J. Rankin, J. Rinzel, M. Carrasco, and D. Heeger, “An attention model of binocular rivalry,” en, *Journal of Vision*, vol. 17, no. 10, p. 579, Aug. 31, 2017, ISSN: 1534-7362. DOI: [10.1167/17.10.579](https://doi.org/10.1167/17.10.579). [Online]. Available: <http://dx.doi.org/10.1167/17.10.579>.
- [10] J. A. Spudich and S. Sivaramakrishnan, “Myosin vi: An innovative motor that challenged the swinging lever arm hypothesis,” *Nature reviews. Molecular cell biology*, vol. 11, no. 2, pp. 128–37, Feb. 2010, [Online; accessed 2024-11-13], ISSN: 1471-0072. DOI: [10.1038/nrm2833](https://doi.org/10.1038/nrm2833).
- [11] D. Simm and M. Kollmar, “Waggawagga-cli: A command-line tool for predicting stable single  $\alpha$ -helices (sah-domains), and the sah-domain distribution across eukaryotes,” *PloS one*, vol. 13, no. 2, e0191924, Feb. 14, 2018, [Online; accessed 2024-11-13], ISSN: 1932-6203. DOI: [10.1371/journal.pone.0191924](https://doi.org/10.1371/journal.pone.0191924).
- [12] M. Fingeroff, *High-Level Synthesis Blue Book*. Xlibris Corporation LLC, 2010, ISBN: 9781450097253. [Online]. Available: [https://openlibrary.org/books/OL49513944M/High-Level\\_Synthesis\\_Blue\\_Book](https://openlibrary.org/books/OL49513944M/High-Level_Synthesis_Blue_Book).

- [13] D. Süveges, Z. Gáspári, G. Tóth, and L. Nyitray, “Charged single  $\alpha$ -helix: A versatile protein structural motif,” en, *Proteins: Structure, Function, and Bioinformatics*, vol. 74, no. 4, pp. 905–916, Aug. 19, 2008, ISSN: 0887-3585. DOI: [10.1002/prot.22183](https://doi.org/10.1002/prot.22183). [Online]. Available: <http://dx.doi.org/10.1002/prot.22183>.
- [14] Á. Kovács, D. Dudola, L. Nyitray, G. Tóth, Z. Nagy, and Z. Gáspári, “Detection of single alpha-helices in large protein sequence sets using hardware acceleration,” en, *Journal of Structural Biology*, vol. 204, no. 1, pp. 109–116, Oct. 2018, ISSN: 1047-8477. DOI: [10.1016/j.jsb.2018.06.005](https://doi.org/10.1016/j.jsb.2018.06.005). [Online]. Available: <http://dx.doi.org/10.1016/j.jsb.2018.06.005>.
- [15] M. Knespel and H. Brunst, “Rapidgzip: Parallel decompression and seeking in gzip files using cache prefetching,” *arXiv*, 2023. DOI: [10.48550/ARXIV.2308.08955](https://arxiv.org/abs/2308.08955). [Online]. Available: <https://arxiv.org/abs/2308.08955>.
- [16] X. Inc, *Vitis Data Compression Library – Xilinx GZip Compression and Decompression*, [https://xilinx.github.io/Vitis\\_Libraries/data\\_compression/2020.1/source/L2/gzip.html](https://xilinx.github.io/Vitis_Libraries/data_compression/2020.1/source/L2/gzip.html), [Accessed 08-11-2024].
- [17] Á. Kovács, “Magányos töltött alfa-hélix megbízható detektáló algoritmus gyorsítása fpga-n,” Bachelor’s Thesis, Pázmány Péter Katolikus Egyetem, 2015.
- [18] E. Wang and C.-L. A. Wang, “(i,i+ 4) ion pairs stabilize helical peptides derived from smooth muscle caldesmon,” en, *Archives of Biochemistry and Biophysics*, vol. 329, no. 2, pp. 156–162, May 1996, ISSN: 0003-9861. DOI: [10.1006/abbi.1996.0204](https://doi.org/10.1006/abbi.1996.0204). [Online]. Available: <http://dx.doi.org/10.1006/abbi.1996.0204>.
- [19] K. Samejima, M. Platani, M. Wolny, *et al.*, “The inner centromere protein (incenp) coil is a single  $\alpha$ -helix (sah) domain that binds directly to microtubules and is important for chromosome passenger complex (cpc) localization and function in mitosis,” en, *Journal of Biological Chemistry*, vol. 290, no. 35, pp. 21 460–21 472, Aug. 2015, ISSN: 0021-9258. DOI: [10.1074/jbc.](https://doi.org/10.1074/jbc.)

- m115.645317. [Online]. Available: <http://dx.doi.org/10.1074/jbc.M115.645317>.
- [20] A. K. Ulrich, M. Seeger, T. Schütze, N. Bartlick, and M. C. Wahl, “Scaffolding in the spliceosome via single  $\alpha$  helices,” en, *Structure*, vol. 24, no. 11, pp. 1972–1983, Nov. 2016, ISSN: 0969-2126. DOI: 10.1016/j.str.2016.09.007. [Online]. Available: <http://dx.doi.org/10.1016/j.str.2016.09.007>.
- [21] M. Wolny, M. Batchelor, P. J. Knight, E. Paci, L. Dougan, and M. Peckham, “Stable single  $\alpha$ -helices are constant force springs in proteins,” en, *Journal of Biological Chemistry*, vol. 289, no. 40, pp. 27 825–27 835, Oct. 2014, ISSN: 0021-9258. DOI: 10.1074/jbc.m114.585679. [Online]. Available: <http://dx.doi.org/10.1074/jbc.M114.585679>.
- [22] C. Wang, J. Chalovich, P. Graceffa, R. Lu, K. Mabuchi, and W. Stafford, “A long helix from the central region of smooth muscle caldesmon,” *Journal of Biological Chemistry*, vol. 266, no. 21, pp. 13 958–13 963, Jul. 1991. DOI: 10.1016/s0021-9258(18)92796-1.
- [23] A. Bateman, M.-J. Martin, S. Orchard, *et al.*, “Uniprot: The universal protein knowledgebase in 2023,” en, *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, Nov. 21, 2022, ISSN: 0305-1048. DOI: 10.1093/nar/gkac1052. [Online]. Available: <http://dx.doi.org/10.1093/nar/gkac1052>.

# Appendix A

## Source codes

The source codes created, modified or used during the study are accessible in the following repository: <https://dev.itk.ppke.hu/kerekgyarto.marton/ft-charge-public>