Accelerating Charged Single a-helix Detection on FPGA

Sam Khozama

khozama.sam@itk.ppke.hu

Supervisors

Dr. Zoltán Nagy **Dr. Zoltán Gáspári**

Info about the presenter:

Sam Khozama
33 years
Syrian

Bachelor in informatics Engineering, Computer Systems and Networking department.
2009 – 2015. Tishreen University, Lattakia, Syria.

Master in Computer Science Engineering.
2016 – 2018. Pázmány Péter Catholic University, Budapest, Hungary.

Ph.D. in Computer Sciences.
2018 – till present. Pázmány Péter Catholic University, Budapest, Hungary.

1. INTRODUCTION

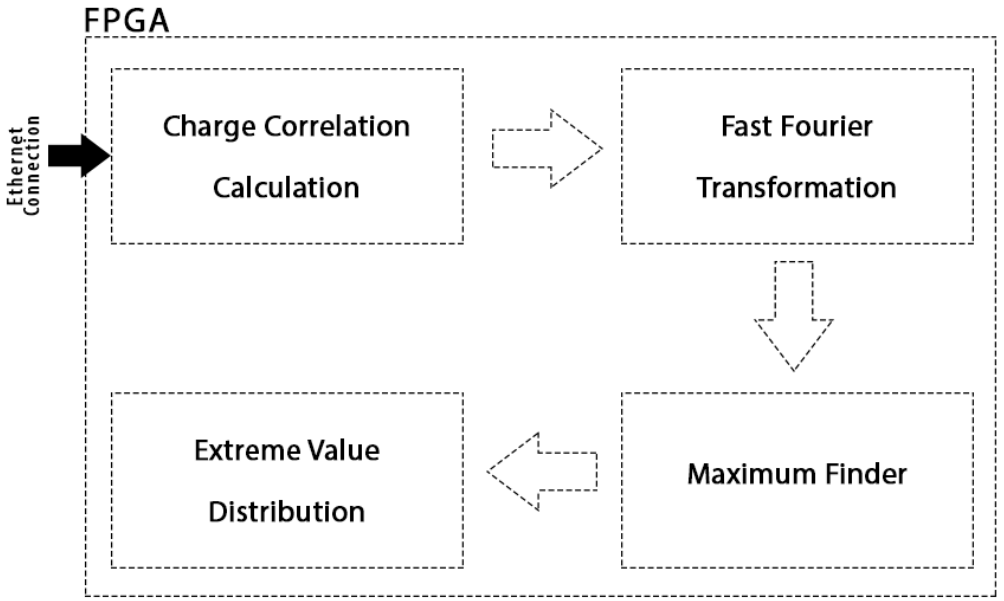
- Charged single a-helix (CSAH) is a unique structural motif in proteins.
- It has been experimentally characterized only in a small number of proteins, therefore, its recognition by prediction methods is of high importance.

- Here we present the speedup of FT_CHARGE, one of the earliest methods based on Fourier transformation.
- Our novel implementation offers higher computational speed to allow processing of very large protein sequence sets like full genomes or even metagenomic samples within hours.

- Both of FPGA and bioinformatics algorithms have parallelism at a fundamental level.

2. FT_CHARGE ALGORITHM

- FT_CHARGE algorithm is a very computationally intensive algorithm, so, it is suitable for accelerating on FPGA.
- Standard FASTA format is the input format.
- The host computer sent the encoded data to the FPGA, where it is processed, and we got back only the filtered results(candidate sequences, which we expect to have a-helix).
- The execution is working block by block.



The block diagram of the system

3. FPGA IMPLEMENTATION

- We have a previous implementation of FT_CHARGE algorithm on a small FPGA. The main goal is optimizing the computation of the charge correlation function to feed the FFT core efficiently.

- The first way to improve that implementation is simply to replicate the system-blocks as much as we can, to fit into a larger ZYNQ device

- We have the possibility to increase the performance nearly **6** times more on a larger FPGA.

TABLE I. AREA REQUIREMENTS OF THE PREVIOUS SYSTEM

	CLB LUTs	CLB Flip-Flops	BRAM	DSP Slices
Complete system	44662	53122	90	157
ZCU102	274080	548160	912	2520
Estimation	6	10	10	16

- Hence, we should take into consideration, having such a huge number of hardware blocks on FPGA.

3. FPGA IMPLEMENTATION

- A more efficient way to extend this solution is to implement several units in parallel, using a larger Xilinx ZYNQ board.

- Our suggested solution is to change the output of the Charge Correlation block in order to compute and send all 32 or 64 elements to FFT in each clock-cycle, and to replace the previous FFT module with more parallel processing units.

- We proposed a new FFT implementation based on the Cooley-Tukey algorithm to accept 32 or 64 elements in each clock-cycle.

- This is the motivation behind trying to improve the speed of computation by parallelizing the algorithm.

3. FPGA IMPLEMENTATION

- We decreased the number of bits and saved lots of space because of using fixed-point numbers instead of floating-point, in the multiplication processes inside the FFT. The accuracy of the solution is almost the same.

- We had two possibilities to prepare the input-output interfaces:
  i. either loading the input sequence by sequence or
  ii.loading many sequences to a large buffer.

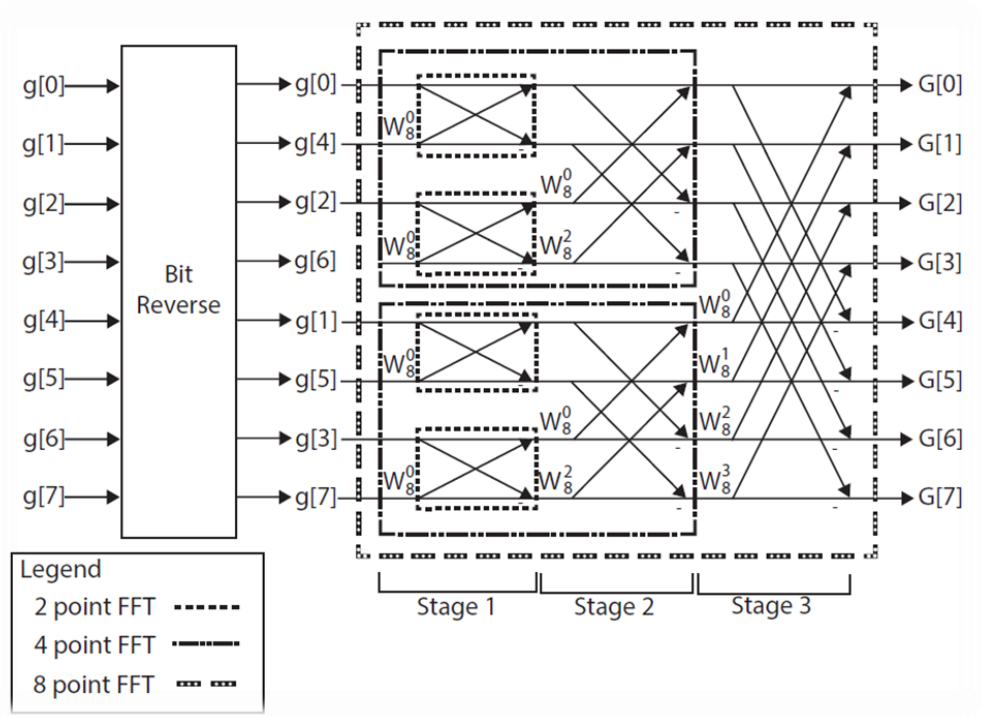
- Our improvement here was, to work on larger buffers where one buffer contains several sequences concatenated one after each other and instead of giving the length of one sequence, we will give an array of lengths to the FT_CHARGE algorithm.

- By this step, the utilization of the hardware will be greatly increased.

4. FAST FOURIER TRANSFORMATION

- **Cooley-Tukey algorithm**

- In our case, we changed **Charge Correlation** part to have parallel output.
- The computation could be increased by 32 times.
- The most important improvement here is to change the sequential processing to parallel.
- So, the new system accept new 32 or 64 element in each clock-cycle. Charge correlation Calculation output changed to parallel to feed these streams into FFT.



Cooley-Tukey algorithm: An 8 point FFT built recursively.

4. FAST FOURIER TRANSFORMATION

- Table II, shows three different representations could be used in the current system.
- The original used 18-bit word length inside FFT. Second one used 23 and 24 bit in case of 32 and 64 element windows. These designs were running on 250MHz clock-frequency and processing one window in each clock-cycle. The last one was a single-precision floating-point version.

TABLE II. AREA REQUIREMENTS OF THE CURRENT SYSTEM WITH DIFFERENT NUMBER REPRESENTATIONS

	CLB LUTs	CLB Flip-Flops	CLBs	Block RAM	DSPs
ZCU102	274 080	548 160	599 550	912	2 520
Fixed-point 18bit	44 043	42 344	8 155	13.5	668
Fixed-point 23bit	51 572	50 378	9 417	13.5	768
Floating-point	225 157	321 469	34 092	13.5	2 146

4. FAST FOURIER TRANSFORMATION

- Using single-precision floating-point numbers requires 6 times more DSP slices.
- To fit the floating-point version into the ZCU102 board, the Initiation Interval of the FFT block is increased to two.
- In this case, the FFT is computed in two clock-cycles and the floating-point MACs are shared between two operations, effectively halving the DSP slice requirement of the circuit.
- The clock-frequency of the design is reduced to 150MHz for stable operation to avoid timing issues caused be FPGA resource utilization.

TABLE III. RUNTIME PROCESSING OF THE SPROT DATABASE (SECONDS), FPGA AND SOFTWARE IMPLEMENTATIONS

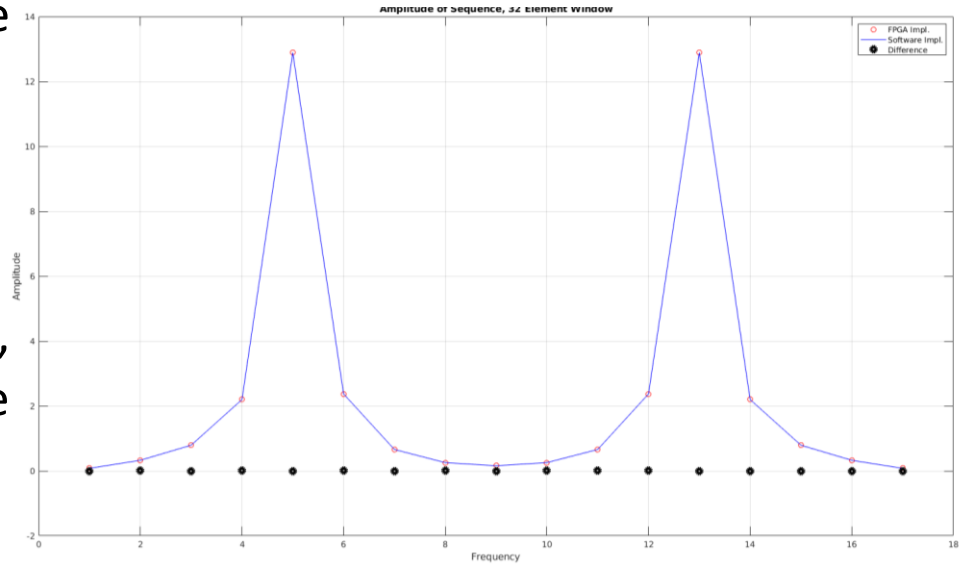
	with communication	without communication
Previous System	275.857	-
Current System		
Fixed-point	20.218	2.24159
Floating-point	23.1604	5.15008
AMD Ryzen 5 3400G	223.52	-
INTEL Core i5-4590	371.731	-

5. System Testing

- We are using a Zedboard SoC, which uses a Zynq system.
- While the circuit is running, we experience a few cases where the sequences are detected on the software side but not on the hardware.

- We could examine two types of errors:
  I. The amplitude was close to the amplitude threshold (7.0).
  II. We have a difference in the third or fourth decimal value, which could not recognize by the human-eye. It would be clear after diving with the numbers.

- We can easily overcome this, by increasing the accuracy and adding additional dedicated bits in ap_fixed type but it requires more resources



Amplitude of FFT, 32 elements window computation

6. Conclusion

- We improved, in this paper, the proposed FPGA based system for speeding up a-helix detection algorithm by replicating the main three blocks as much as they fit a larger FPGA board.
- Implementing these processing units in parallel enables fast search on larger protein databases and runs the whole system at a speed of 30 times higher.
- Using the butterfly diagram and fixed-point representation inside FFT, save more area and time with almost the same accuracy.
- We have also tested this code, which could be implemented on FPGA with real sequence data and we got the same results with the previous version on Matlab.

7. References

- Süveges, D., Gáspári, Z., Toth, G. and Nyitray, L., 2009. Charged single α-helix: A versatile protein structural motif. *Proteins: Structure, Function, and Bioinformatics*, *74*(4), pp.905-916.

- Kovács, Á., Dudola, D., Nyitray, L., Tóth, G., Nagy, Z. and Gáspári, Z., 2018. Detection of single alpha-helices in large protein sequence sets using hardware acceleration. *Journal of structural biology*, *204*(1), pp.109-116.

- Nagy, Z., Gáspári, Z. and Kovács, Á., 2016, January. Accelerating a charged single α-helix search algorithm in protein sequences using FPGA. In *15th International Workshop on Cellular Nanoscale Networks and Their Applications, CNNA 2016* (pp. 117-118). IEEE Computer Society.