2016 IEEE International Parallel and Distributed Processing Symposium Workshops

A Novel Associative Memory Based Architecture for Sequence Alignment

M. Ali Mirzaei, Francesco Crescioli,
Giovanni Marchiori, Giovani Calderini
ATLAS group
LPNHE, IN2P3, CNRS, UPMC
4 Place Jussieu, Paris 75005, France
Email: mmirzaei@lpnhe.in2p3.fr

Sebastien Viret, William Tromeur,
Guillaume Baulieu, Geoffrey Galbit
CMS group
IPNL, IN2P3, CNRS, UCBL
4, Rue Enrico Fermi, 69622 Villeurbanne, France
Email: viret@in2p3.fr

***Abstract*—this paper presents a novel hardware architecture based on Associative Memory technology for sequence alignment. The Associative Memory chip (AMchip) architecture employs a huge amount of parallelization to perform real time combinatorial pattern matching. It can be used to perform very fast searches over a large database using hamming weight as a similarity metric. In the presented hardware platform the AMchip communicates with a Xilinx Zynq ARM CPU + FPGA through high-speed (2.4 Gbps) serial links enabling seamless integration between the software and the hardware acceleration. The result shows that low score sections of a database sequence are eliminated rapidly using this architecture which in turn leads to significant speed up. The outcome of a preliminary study on a selected reference protein database shows that all the matches between query and database sequences found by NCBI-BLAST can be successfully found also by the Associative Memory based algorithm.**

***Keywords*-AMchip; Xilinx FPGA; sequence alignment; NCBI BLAST, Hamming Weight**

I. Introduction

The Associative Memory chip (AMchip) is a custom ASIC originally designed to perform real time combinatorial pattern matching for charged particles trajectory reconstruction (tracking) at high-energy physics hadron collider experiments [1]. The first application based on the AMchip technology was the Silicon Vertex Trigger (SVT) track finding supercomputer for CDF experiment [2]. The most recent version of the AMchip (AMchip06) has been developed for the FastTracKer (FTK) [3] [4] [5] processor of the ATLAS experiment [6] at the LHC. FTK is a complex hardware processor based on FPGAs and AMchips which is able to reconstruct O(1000) particle trajectories per event at 100 kHz event rate, processing the data coming from 90 million digital channels. The ability to process in real time the huge amount of data produced by the experiment is fundamental since it is possible to store for later analysis only a tiny fraction of the events. The FTK, which will be commissioned in 2018, will become a key component of ATLAS online event selection procedure. The AMchip plays an extremely important role in the FTK system and its development is therefore driven by the main high-energy physics application.

However, it is also a generic computing element that can be applied into other domains such as image processing [7] and of course bioinformatics, as shown in this document.
In bioinformatics, global/local sequence alignment algorithms are comparing biological sequences such as amino acids of different proteins sequences or nucleotides of DNA sequences. Several sequence alignment algorithms have been presented such as Smith-Waterman (SW) [8]. The Smith-Waterman algorithm is a dynamic programming algorithm with a scoring method. Basic Local Alignment Search Tool (BLAST) [9] [10] is a popular software toolset that uses a heuristic approach to find alignments efficiently without the computational cost of the SW algorithm. The BLAST search enables a researcher to compare a query with a database to identify the database sequences matching the query within a certain threshold. There are several BLAST tools, for example BLASTP is the tool specialized for protein databases.
The genomic algorithms are very time consuming and therefore require intensive computing power when executed on conventional CPUs. Aside from algorithmic acceleration as provided by BLAST, several hardware solutions were proposed in order to speed up alignment algorithms. These can be divided into three main categories:

1) Acceleration using Graphic Processing Units (GPUs). For example Feng et al [11] enhanced SW protein sequence alignment algorithm on NVIDIA GPUs by optimizing memory access bandwidth and thread allocation. He achieved 32%-52% better performance comparing to CUDASW++2.0 implementation [12]. CUDASW++ is a bioinformatics solution based on Smith-Waterman algorithm developed on NVIDIA Tesla GPUs with significant speed-ups over BLAST under certain query length conditions. The latest version is CUDASW++3.0 [13].
2) Acceleration using Field Programmable Gate Array (FPGA) parallel computing power. Varieties of architectures were presented which accelerate SW algorithms. An example is Causapruno et al [14], who developed an architecture based on systolic array and

/16 $31.00 © 2016 IEEE
DOI 10.1109/IPDPSW.2016.21

interleave pipelining. Nawaz et al [15] proposed a parallel architecture for any of-the-shelf FPGA development board. Another interesting approach is Muriki et al [16] who developed a solution for BLAST acceleration based on embedded ARM processor and FPGA to take into account not only computing power, but also power efficiency. They used PCIe interface for data communication purpose.

3) Acceleration using an Application Specific Integrated Circuit (ASIC), a dedicated silicon chip to execute as fast as possible a portion of the algorithm. An example is the commercial accelerator Parcel [17]. Although it provides a very fast processing, the usage of ASIC is often discouraged by the high initial development cost with respect to GPU or FPGA solutions.

This paper presents a solution mixing the two last categories. In this approach, sequence alignment is performed by a novel system based on an AMchip (ASIC) interconnected to a FPGA. The main aim of this paper is to demonstrate how ASIC-AMchip can reduce the number of low score sections (non-candidate) very rapidly by exploiting its highly parallelized structure. Reducing the number of high score sections (candidate) from thousands to few units by adjusting threshold speeds up the alignment algorithm dramatically. Since the AMchip is an available chip already developed for physics experiments, an AMchip-FPGA approach would have the computing power benefit of a dedicated hardware system without the high development cost inherent in this kind of solution.

The paper is organized as follow: section II will briefly introduce the internal architecture of an AMchip in general and explains the architecture and the operational sequence of the AMchip06 in more details. The hardware environment used to develop the algorithm is illustrated in section III. A similarity region finding algorithm for AMchip will be explained in section IV. The data set used for the data analysis is described in section V and finally, the results will be presented and discussed in section VI.

II. Associative Memory architecture and AMchip implementation

A. Internal architecture of an Associative Memory

The function of the Associative Memory is pattern recognition. For the AM a pattern is a structured data made by a sequence of values. The AM can store a database of patterns and then it can be used to find occurrences of these patterns in a query data set exploring all possible combinations of the data set.

The Associative Memory basic building block is the Content Addressable Memory (CAM) cell [18]. A CAM cell is a memory element with a comparator: it is possible to write a data value in the CAM cell and then query whether a certain input data value is equal to the internally stored value. The

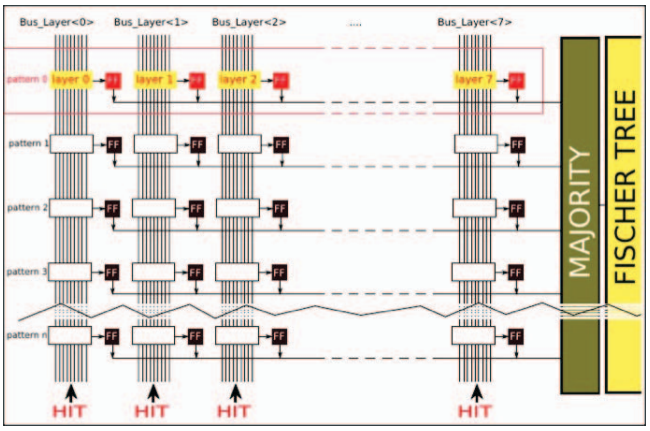

Figure 1. AMchip functional description.

Associative Memory is made of many CAM cells organized in a matrix, as shown in Figure 1. Each column is associated to an input data bus: all CAM cells in the column compare at once with the data sent on the bus. Each row represents a pattern: the sequence of values is stored in the CAM cells of the row.

When the query dataset is sent to the AM all CAM cells compare with the data at once. If a match at CAM cell level is found it is recorded and this information is retained during the examination of the whole dataset.

For each pattern (row) there is a logic element called Majority: the function of this logic element is to decide whether the pattern is present in the examined dataset or not. In order to make the decision it looks at the amount of CAM cells of the pattern that had a match in the data set, if this amount is over the user specified threshold then the pattern is declared matched and it is sent in output to the AM. The patterns are sent in output by a priority encoder called Fischer Tree starting with the matched pattern with the lower memory address to the pattern with the higher address. The output of the AM is the internal memory address of each pattern found in the input data set along with its hitmap: which input bus had a match.

The power of the AM is the ability to find patterns matching any combination of the input dataset during dataset readout, this without any further computation.

B. The AMchip06 architecture

There are several implementations of the AM concept described before with different features in terms of storable number of patterns and pattern length. The latest of these implementations is AMchip06, a 65 nm custom ASIC with several features, performance and power consumption improvements with respect to the previous versions.

The AMchip06 contains 8 CAM cells in each pattern row, each CAM cell can store up to 16 bit of information with a variable amount of ternary logic bits (possibility to encode 0,

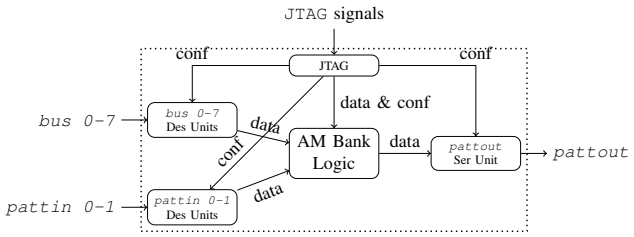

Figure 2. AMchip I/O and internal logic blocks.

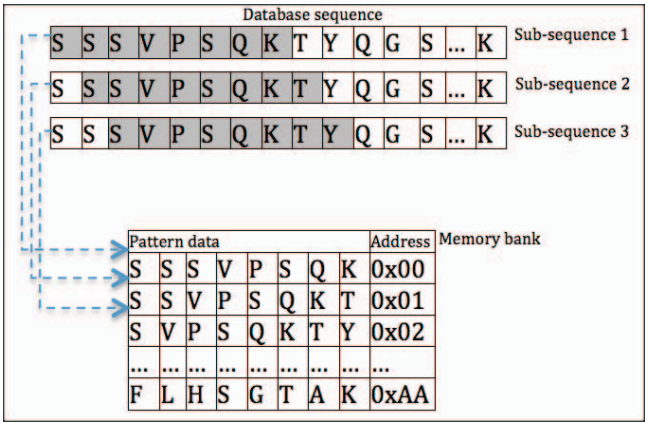

Figure 3. Database sequence reformatting of sequence = SSVPSQKTYQGSYGFRLGFLHSGTAK, m=26, n=8.

1, and either 0 or 1). The AMchip06 has a capacity of 128K patterns in 8-word pattern mode. It can also be configured in 16 or 32 words decreasing the number of bits per word and the total number of patterns available (15 bit * 16 words pattern for 64k patterns or 14 bit * 32 words for 32k patterns). It has the ability to handle data incoming at 2 gbps per input bus and it can send matching sequences in output at a 2.4 gbps data rate. It has the possibility to collect the output patterns of two neighbouring chips enabling the possibility to build multi-chip boards with the same I/O interface of a single chip. The schematic internal structure of the chip is sketched in Figure 2. The features of the AMchip06 are detailed in [19] [20].

III. Experimental hardware setup

A dedicated hardware platform is under development using a Xilinx Zynq mini-ITX board [21] and dedicated AMchip FMC mezzanine. The Zynq is a System on Chip that contains an ARM Cortex-A9 processor and a Xilinx Kintex FPGA integrated in a single chip. LINUX can be executed on the ARM processor that is integrated with the FPGA in such a way to facilitate data communication between CPU, FPGA and AMchip on the FMC mezzanine. The Zynq + AMchip FMC is similar to a desktop PC and all hardware details are hidden to the user.
The software running on the LINUX system on the CPU side of the Zynq will be able to upload the data to the

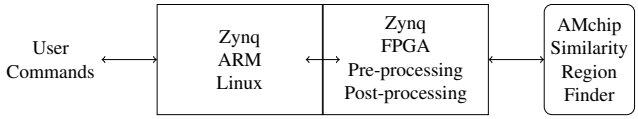

Figure 4. AM-based setup data flow.

FPGA side of the processor using the internal high-speed bus. In turn the FPGA side will be able to communicate with the AMchip hosted on the FMC mezzanine using 8 high-speed serial links at 2 gbps to send query data and receive the response from the AMchip using one high-speed serial link at 2.4 gbps. The FPGA will be able to execute further processing before sending the data back to the LINUX side for final computation and presentation to the user. This data flow architecture is show in figure 4.

IV. Similarity regions search with the AMchip

This section outlines how the AMchip can be used for fast finding of similarity regions between a query sequence and a large database using a very simple algorithm. This algorithm is preliminary and more refined analysis can be added in the FPGA and CPU.

A. BLASTP algorithm

Since we are going to compare with the BLASTP tool of the BLAST suite we briefly describe below the algorithm used by the software tool. The BLASTP tool of the BLAST suite is specialized for protein database query and is divided into three stages:

1) Seeding: small fixed length similarity regions are found
2) Ungapped extension: each seed is extended in both directions without allowing for gaps (insertion or deletion of symbols from the sequence), this stage reject regions with poor ungapped alignment score
3) Gapped alignment: an algorithm similar to SW on the regions found by the previous stage perform the local alignment provining the final score

The quality of the alignment is determined by a score computed as follows: for each pair x, y of aligned symbols is assigned a score $M(x, y)$ where M is a $N \times N$ matrix, N being the number of possible symbols. This matrix is user defined and assigns a high score to identical or biological similar symbols while it assigns zero or negative score to different symbols.
In the setup described below, the AMchip is performing the seeding stage with large fixed word length.

B. Associative Memory algorithm

A gene bank or a protein database is reformatted before loading into AMchip memory bank as shown in Figure 3. In this method the database sequence is reformatted with respect to the AM pattern length. Let's assume m to be the

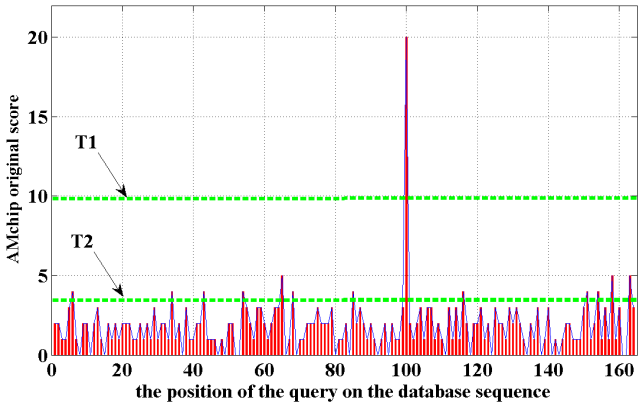

Figure 5. Hamming Weight of sub-sequence comparison for every sub-sequence position from table I. Possible thresholds are highlighted.

database sequence length and n the AM pattern size. The database sequence is then divided into m-n sub-sequences of length n, obtained by running a sliding window over the database sequence. Each symbol in the database sequence is encoded in such a way to be written as a word of the AM pattern. In the case of the AMchip06 there are 14 bits per word available in 32-word pattern mode, more than enough to encode a nucleic acid or a protein amino acid. Each sub-sequence of size n is then loaded as one pattern on the AM memory bank. In order to load the full database in the AM it is therefore necessary to have at least m-n patterns, for example the biggest database loadable in a single AMchip06 in 32-word pattern mode (n=32) is 32k.

Once the reference database is loaded into the AM it can be used to search matches with the query data. To let the AM find the similarity regions between the query and the database it is sufficient to put the query in input to the AM, one symbol of the query for each input bus of the AM and then ask the AM for the matching patterns over a certain threshold T. If the query length is more than the pattern size n then it is sufficient to repeat the query operation every n symbols of the query. As explained in section II the AM compare the input with all the patterns at the same time, the time to execute this operation is therefore independent of the pattern bank size. Since the patterns in the AM bank are the database sequence in every possible position this operation is equivalent to compute the hamming weight of the binary comparison of the query with each sub-sequence of the database. By definition [22], Hamming weight (HW) of a binary number is equal to the number of '1' of this number's representation. For instance, the HW for a binary number such as 1000110010011010 is 7. The HW represents a measure of similarity between the query and the database sequence. The output of the AMchip is then equivalent to the first stage (seeding) of the BLASTP algorithm using a score matrix M(x, y) equal to the identity matrix.

A preliminary study has been conducted on two short

Table I
QUERY AND AN EXAMPLE DATABASE SEQUENCE

Query Sequence
HLMRVEGNLQAYYMEDVNSGRHSVCVPYEGPQ
Database Sequence
SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCP VQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHHERCSDSDGLAPP QHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNS SCMGGMNRRPILTIITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENL

Table II
SELECTED SEQUENCE FOR THE PRIMARY OBSERVATION

Sequence (gi)	Reference (ref)	Sequence (gi)	Sequence (ref)
8922077	NP_061172.1	768020113	XP_011527730.1
70167113	NP_001020278.1	768020129	XP_011527734.1
578800406	XP_006711172.1	768020121	XP_011527732.1
70167032	NP_056656.2	4501919	NP_001103.1
70166944	NP_056655.2	237681091	NP_001153702.1
767908090	XP_011507364.1	7669479	NP_056649.1
70166852	NP_001102.2	578836418	XP_006724017.1
767908088	XP_011507363.1	7669477	NP_056648.1
767908086	XP_011507362.1	768020101	XP_011527727.1
530377181	XP_005262802.1	768020117	XP_011527731.1
226530771	NP_001152767.1	302058269	NP_036223.2
21245124	NP_640336.1	530423621	XP_005255871.1
226529650	NP_001152757.1	223972688	NP_631913.3
223972690	NP_001138872.1	767989508	XP_011521221.1
767989510	XP_011521222.1		

sequences as shown in table I. Figure 5 shows the HW calculated by AMchip for the two given sequences. According to the threshold value it is effectively possible to select only the regions with the desired similarity. For example if the threshold is set to T1 only one candidate will be selected, the best match, while if threshold is set to T2 13 candidates will be selected. To select interesting region the threshold should be set over the expected random matches between two sequences of length n, for example in the case of this protein data and a sequence of length 32 a threshold >5 is effective. The computing cost of this algorithm is negligible, the only parameters that determine the execution speed are the readout speed of the query (2 gbps in the case of the AMchip06), the output speed (2.4 gbps in the case of the AMchip06), and the number of similarity regions found.

V. TEST DATA (QUERY AND DATABASE SEQUENCES)

In order to check further the validity of this algorithm a larger set of sequences was selected and results were compared to the matches obtained by the BLASTP analysis tool. From the reference protein sequence database (Refseq_protein [23]) we extracted some sequences of Homo sapiens database as listed in II. : 29 sequences with length ranging from 200 to 1200 symbols. The search query of length 32 is the same as in I. .

VI. RESULTS AND DISCUSSION

Figure 6 shows the best matches interval between query (TABLE I.) and each sequence of TABLE II. found by

AMchip and BLASTP. In each case the region found by the hamming weight thresholding and BLASTP is the same. This result shows that AMchip is capable to find the best matches in a very simple case successfully with little difference comparing to NCBI BLASTP tool. The time to perform this operation depends on the data itself, but it is possible to give a general estimation. Currently the slowest operation is loading the reference database on the AMchip. Writing patterns is performed through the slow JTAG interface due to a design constraint and currently takes 700 ms. In case of multi-chip board it is possible to write all chips in parallel. This limitation is not due to the AM internal architecture and it will probably be lifted in future AMchip versions enabling writing at the same speed than the I/O buses. The query operation is done through the 2 gbps high-speed serial links and is very fast: it is possible to send 8 symbols in parallel each 10 ns so for example querying a 32 symbols long sequence takes 40 ns. For each query the system must wait a fixed latency (400 ns) plus 16.6 ns for each matched pattern (2.4 gbps output). If we suppose an average of 1 matched pattern per query it is possible to analyze data with respect to the loaded reference database at 2.2 MHz. In any case, the processing time is independent from the size of the database sequence. It must also be noted that the proposed dataflow is a pipeline: the FPGA stage and the final processing in CPU can proceed in parallel with the AM stage as soon as first matched patterns become available.

VII. Conclusion and Future Work

The Associative Memory was presented and the AMchip silicon implementation was described. An hardware platform based on CPU+FPGA+AMchip was described and a preliminary algorithm for fast search of similarity regions has been described. The comparison between the sequence alignment results achieved by BLASTP tool and the results calculated by AMchip shows that it successfully determined the location of the best matches between a query and a given protein database sequence. The presented sequence matching result is promising, however, larger databases and longer sequences must be studied in order to understand more precisely the computing performances of the AMchip approach. The AMchip step is supposed to be the pre-processing stage of a more refined alignment algorithm implemented in FPGA and CPU. Until now, only protein sequences were considered in the study and no other type of genomes. It will be interesting to know how the proposed approach will apply to DNA sequences.

Acknowledgment

The project received founding from the French ANR project FastTrack (ANR-13-BS05-0011-01). The author and project team would like to be extremely grateful for their generosity, contribution and financial support.

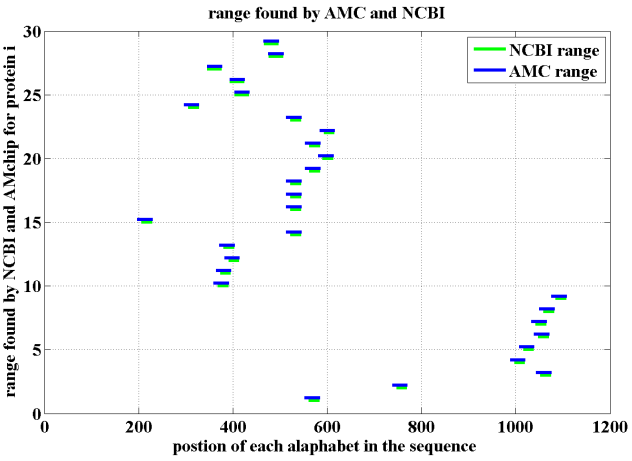

Figure 6. Best match interval between query and a database sequences (sequence extracted from TABLE II) found by AMchip and BLASTP

References

[1] M. Dell'Orso and L. Ristori, "{VLSI} structures for track finding," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 278, no. 2, pp. 436 – 440, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0168900289908620

[2] A. Bardi, S. Belforte, A. Cerri, M. Dell'Orso, S. Donati, S. Galeotti, P. Giannetti, E. Meschi, F. Morsani, G. Punzi *et al.*, "A large associative memory system for the cdf level 2 trigger," in *Nuclear Science Symposium, 1998. Conference Record. 1998 IEEE*, vol. 1. IEEE, 1998, pp. 312–313.

[3] A. collaboration *et al.*, "Fast tracker (ftk) technical design report," CERN-LHCC-2013-007, Tech. Rep., 2013.

[4] N. Biesuz, S. Citraro, S. Donati, M. Piendibene, E. Rossi, C. Sotiropoulou, G. Volpi, A. Annovi, A. Andreani, M. Berreta *et al.*, "Highly parallelized pattern matching execution for atlas event real time reconstruction," *accepted for publication on IEEE Transactions on Nuclear Science (TNS)*, 2016.

[5] A. Annovi, R. Beccherle, M. Beretta, E. Bossini, F. Crescioli, M. Dell'Orso, P. Giannetti, J. Hoff, T. Liu, Y. Liberali *et al.*, "Associative memory design for the fast track processor (ftk) at atlas," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*. IEEE, 2011, pp. 141–146.

[6] A. Collaboration *et al.*, "Atlas detector and physics performance technical design report, volume i," 1999.

[7] C.-L. Sotiropoulou, A. Annovi, M. Beretta, P. Luciano, S. Nikolaidis, and G. Volpi, "A multi-core fpga-based clustering algorithm for real-time image processing," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2013 IEEE*. IEEE, 2013, pp. 1–5.

[8] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in applied mathematics*, vol. 2, no. 4, pp. 482–489, 1981.

[9] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[10] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[11] X. Feng, H. Jin, R. Zheng, L. Zhu, and W. Dai, "Accelerating smith-waterman alignment of species-based protein sequences on gpu," *International Journal of Parallel Programming*, vol. 43, no. 3, pp. 359–380, 2015.

[12] Y. Liu, B. Schmidt, and D. L. Maskell, "Cudasw++ 2.0: enhanced smith-waterman protein database search on cuda-enabled gpus based on simt and virtualized simd abstractions," *BMC research notes*, vol. 3, no. 1, p. 93, 2010.

[13] Y. Liu, A. Wirawan, and B. Schmidt, "Cudasw++ 3.0: accelerating smith-waterman protein database search by coupling cpu and gpu simd instructions," *BMC bioinformatics*, vol. 14, no. 1, p. 117, 2013.

[14] G. Causapruno, G. Urgese, M. Vacca, M. Graziano, and M. Zamboni, "Protein alignment systolic array throughput optimization," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 68–77, 2015.

[15] Z. Nawaz, M. Nadeem, H. Van Someren, and K. Bertels, "A parallel fpga design of the smith-waterman traceback," in *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 2010, pp. 454–459.

[16] K. Muriki, K. D. Underwood, and R. Sass, "Rc-blast: Towards a portable, cost-effective open source hardware implementation," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 8–pp.

[17] P. Inc., "Parcel," http://www.parcel.com, 2006.

[18] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: A tutorial and survey," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 3, pp. 712–727, 2006.

[19] F. Crescioli, R. Beccherle, E. Rossi, V. Liberali, M. Beretta, S. Citraro, A. Stabile, M. A. Mirzaei, Y. Piadyk, A. Annovi *et al.*, "Ftk amchip05: an associative memory chip prototype for track reconstruction at hadron collider experiments," ATL-COM-DAQ-2015-083, Tech. Rep., 2015.

[20] M. Beretta, A. Annovi, A. Andreani, M. Citterio, A. Colombo, V. Liberali, S. Shojaii, A. Stabile, R. Beccherle, P. Giannetti *et al.*, "Next generation associative memory devices for the ftk tracking processor of the atlas experiment," *Journal of Instrumentation*, vol. 9, no. 03, p. C03053, 2014.

[21] AVNet, "Zynq-7000 all programmable soc mini-itx development kit," http://zedboard.org/product/mini-itx.

[22] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[23] K. D. Pruitt, G. R. Brown, S. M. Hiatt, F. Thibaud-Nissen, A. Astashyn, O. Ermolaeva, C. M. Farrell, J. Hart, M. J. Landrum, K. M. McGarvey *et al.*, "Refseq: an update on mammalian reference sequences," *Nucleic acids research*, vol. 42, no. D1, pp. D756–D763, 2014.