Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics
& Bioengineering (BIBE), Larnaca, Cyprus, 11-13 November 2012

Opportunities from the Use of FPGAs as Platforms for Bioinformatics Algorithms

Grigorios Chrysos
Euripides Sotiriades
Christos Rousopoulos
Apostolos Dollas

Technical University of Crete
Chania, Greece

dollas@ece.tuc.gr

Agathoklis Papadopoulos
Ioannis Kirmitzoglou
Vasilis Promponas
Theocharis Theocharides

University of Cyprus
Nicosia, Cyprus

ttheocharides@ucy.ac.cy

George Petihakis
Jacques Lagnel
Panagiotis Vavylis
George Kotoulas

Hellenic Center for Marine Research
Heraklion, Greece

gpetihakis@hcmr.gr

Abstract—**This paper presents an in-depth look of how FPGA computing can offer substantial speedups in the execution of bioinformatics algorithms, with specific results achieved to date for a broad range of algorithms. Examples and case studies are presented for sequence comparison (BLAST, CAST), multiple sequence alignment (MAFFT, T-Coffee), RNA and protein secondary structure prediction (Zuker, Predator), gene prediction (Glimmer/GlimmerHMM) and phylogenetic tree computation (RAxML), running on mainstream FPGA technologies as well as high-end FPGA-based systems (Convey HC1, BeeCube). This work also presents technological and other obstacles that need to be overcome in order for FPGA computing to become a mainstream technology in Bioinformatics.**

Keywords— Bioinformatics algorithms, FPGA based systems, high performance

I. INTRODUCTION

Bioinformatics applications are characterized by immense computational loads and extremely large datasets. At the same time, technologies such as reconfigurable computing, also known as FPGA (Field Programmable Gate Array) computing, are reaching a high level of maturity; also, modern FPGA devices offer substantial hardware resources. Reconfigurable computing, is the field in which algorithms are mapped directly to configurable hardware resources. Despite clock speeds that are typically 1/10th of those in general-purpose computing, by exploiting parallelism at all levels, speedups of up to three orders of magnitude can be achieved vs. software executing the same algorithms, whereas even 2x speedups are considerable due to the immense computational times of these algorithms. The cost per computation and watts per computation are also quite favorable for reconfigurable computing, and hence it is worth examining this form of computing as a platform for bioinformatics applications.

New, powerful FPGA-based platforms [14] have emerged during the last two years, ones that combine general-purpose computers and FPGAs. These platforms emphasize on the high-speed data transmission between the FPGA device and the CPU's main memory, the availability of a conventional CPU and the usage of the network for I/O, thus offering integrated solutions for the execution of I/O- and memory-intensive problems, in which the FPGAs form a tightly coupled co-processor to the conventional one.

The contributions of this work include:

- the presentation of several bioinformatics algorithms, which have been mapped on FPGA stand alone platforms
- the presentation of FPGA platform technology barriers that need to be overcome before reconfigurable technology can offer usable, high performance bioinformatics systems
- the presentation of two modern FPGA-based platforms for data intensive problems, such as bioinformatics problems, and,
- the presentation of two case studies for bioinformatics algorithms running on modern high-end FPGA-based platforms, which show the benefits of this approach.

Several standalone FPGA-based systems on which bioinformatics algorithms have been mapped are presented in Section II. Section III presents the state of the art of the new generation of FPGA based systems; two bioinformatics applications running on these platforms are presented at Section IV. Lastly, Section V has conclusions regarding the potential of this approach. It should be noted that this work gives emphasis on results developed by the authors, as these results form the basis for an FPGA-technology-based supercomputer for bioinformatics. Additional research results from other research groups exist as well, many of which are cited (briefly, due to space limitations).

II. FPGA BASED ARCHITECTURES

This section presents various FPGA-based implementations of bioinformatics algorithms. The main parts of the implemented FPGA-based architectures are described and their performance is evaluated. Moreover, this section describes the advantages and the disadvantages of the FPGA technology for the specific bioinformatics algorithms.

A. Sequence comparison

Sequence comparison algorithms compute the degree of matching between two or more biomolecular sequences. Biologists may use sequence comparison results either as a proxy to infer sequence homology or as part of larger computational pipelines. Two FPGA-based implementations that map sequence comparison algorithms are presented, below:

1) BLAST

BLAST (Basic Local Alignment Search Tool) [1] is the most well-known and widely used algorithm in Bioinformatics. BLAST is used to find similarities between genetic sequences (queries) and sequence databases. A typical BLAST application is to search whether a genetic sequence (query), like a gene, is part of a complete genome (database).

The Technical University of Crete (TUC) has proposed several different architectures and implementations of standalone FPGA-based platforms for BLAST [2] [3] [4]. In particular, [4] presents a software–hardware system implementing the complete algorithm. This system produces a superset of the official BLAST software distribution results. The official BLAST implementation is distributed by the National Center for Biotechnology Information (NCBI). In addition to the TUC architectures, other research groups have looked into the BLAST algorithm [5] [6] [7] [8].

The most recent TUC architecture takes as inputs a query that initializes the system and a streaming database. The database is compared against the query in a cache controller -like scheme, where exact matches between small fragments of the database and the query are found. After an exact match is found, there is tree-like structure of embedded Microblaze™ processors that resolve the final step (extension stage) of BLAST. This system has been implemented on a Xilinx Virtex-5 FPGA (XC5VTX240). The critical resource for the implemented architecture is the internal to the FPGA static memories (Block RAMs - BRAMs), due to their use for the cache controller-like scheme and the MicroBlaze's main memory. BRAMs were also used to implement FIFOs for the communication between several stages of the architecture. The clock rate of the implementation was roughly 100 MHZ. Actual runs showed that the proposed architecture can achieve execution time speedups from 10x to 1000x (depending on algorithm variation and dataset size) when compared to the execution on a high end server. An additional issue in order to realize the above speedup is the input transfer speed. The FPGA device does have 64 Gbps aggregate bandwidth, however, there is no standalone platform that supports such a speed.

The proposed architecture proved that BLAST can run significantly faster vs. a conventional computer but there are certain problems that should be faced. Also, the system produces a small superset of the NCBI results and some of the output metrics are not calculated.

2) CAST

One of the most important factors that can affect execution speed of BLAST and the quality of its results is the occurrence of low complexity regions (LCRs) in protein sequences. LCRs are very common in protein sequence databases [9-12] and some of them are clearly preserved even between evolutionarily distant organisms and have been associated with important cellular functionality and diseases [e.g. 10-11]. Despite their biological significance, LCRs have been routinely masked out of query sequences in BLAST searches to avoid the inclusion of large number of spurious hits in the results. Traditionally, this procedure was performed by SEG [9], an algorithm utilizing Shannon entropy measures to detect LCRs. While fast and accurate, SEG is also very aggressive as it was designed to detect and mask whole regions that may often cover a very large portion of the query sequence. On the other hand, CAST [12] was originally designed to perform more sophisticated masking by "surgically" removing only the amino acid residues skewing the query sequence composition. Therefore, in terms of the quality of the results returned by BLAST, CAST usually outperforms SEG. The main disadvantage of CAST is its computationally intensive nature and the large amount of necessary I/O operations. As such, it is an ideal algorithm that can be potentially accelerated through the use of FPGA technologies.

An FPGA-based approach for the CAST algorithm was proposed in [15], where the inherent parallel characteristics of the algorithm were fully exploited in order to accelerate execution time. The system receives streams of protein sequences in FASTA format and iterates the computation until all LCR regions are discovered for each of the input sequences. Emphatically, a single instance of the proposed FPGA architecture outperforms a multi-threaded software version of CAST running on a high-end PC, by 100-1000x depending on the benchmark dataset. Even the considerably faster SEG algorithm is benchmarked slower, behind the FPGA-based CAST, as the FPGA accelerated CAST runs more than 50x faster in most cases. The proposed CAST architecture occupies just 10% of an average Xilinx commercial FPGA chip, thus allowing multiple instances to be utilized on a single FPGA device to process multiple protein sequences in parallel.

B. Multiple Sequence Alignment

This section describes FPGA-based implementations of multiple sequence alignment (MSA) algorithms.

1) MAFFT

The MAFFT algorithm [16] is a progressive MSA method based on the Fast Fourier Transform (FFT). Software analysis of the MAFFT algorithm showed that the sequence alignment process, which is the final step of the algorithm, takes up to 80% of total algorithm execution time, making it suitable for execution on an FPGA. Lakka *et al.* [17] proposed an FPGA-based IP core that implements the final step of the MAFFT algorithm. It takes as input the segments of the sequences with their weights and it outputs the aligned sequences with their alignment scores. Each MAFFT IP core consists of six-pipelined stages. First, the input sequences are loaded to the internal FPGA block memories (BRAMs). The internal memories were used due to the non-streaming nature of the algorithm and due to the lack of fast I/O on the specific FPGA. The first four implemented stages calculate the gap penalties and the weight matrices among the input sequences. The fifth stage of the architecture calculates the final alignment score between the aligned sequences and it outputs the homology matrix of the input sequences where the sequences are aligned. The final stage implements the final modification of the MSA. The implemented IP core uses single precision floating point arithmetic without losing any accuracy when compared to the software implementation, and the results are identical.

The MAFFT IP core was mapped on a Xilinx Virtex 6 (XCV6SX475T) FPGA. As described above, the critical resource for the implemented IP core is the internal FPGA BRAMs, which are used to store the input sequences. Thus, depending on the size of the input sequences 1 to 15 parallel MAFFT IP cores were mapped on a single FPGA. The clock rate of the proposed architecture is 146 MHz for the full mapping. Many datasets were used for the validation and the evaluation of the MAFFT IP core results and performance. The input datasets ranged from 5 sequences with 366 nucleotide bases long up to 100 sequences with 1403 nucleotide bases long. The speedup that the IP core achieves ranges from 10x up to 50x compared to the sequential execution of the algorithm on a high end PC.

2) *T-Coffee*

T-Coffee [18] is a progressive constraint-based method for MSA and it comprises of three steps. The first step constructs a library with the pairwise alignments between all input sequences. The second step extends the library by assigning a weight for each pair of residues. The final step implements the progressive MSA strategy. Profiling of the T-Coffee method showed that the most time consuming part of the algorithm is the calculation of the alignment score among the input sequences. Lakka et al. [17] presented two IP cores for the score calculation either for the pairwise alignments or for the scores between the lists of nucleotide bases. The calculation of the score is based on a 3-dimensional lookup table which is pre-loaded into the internal FPGA BRAMs to the IP-core for each new input dataset of sequences. Fixed-point arithmetic was used for the normalization stage of the final score, without losing accuracy vs. the software. The T-Coffee IP core was mapped on a Xilinx Virtex 6 (XCV6SX475T) FPGA, achieving 146 MHz clock rate. As described above, the internal BRAMs are used for the storage of the input sequences weights, thus it eliminates parallelism that can be achieved by reducing the number of parallel IP cores up to maximum 22. The performance of the implemented system depends on the size and the nature of the input sequences. The speedup achieved by the T-Coffee IP core compared to the sequential software implementation varies from 1x up to almost 10x without taking into consideration the I/O data transmission.

C. *RNA and Protein Secondary Structure Prediction*

Prediction of RNA and protein secondary structure is of great importance in Medicine and Biology as it may highlight structural and functional properties of molecules. This section presents two FPGA-based architectures for the Zuker and the Predator secondary structure prediction algorithms.

1) *Zuker algorithm*

A typical genetic sequence consists of few thousands of bases, which leads to a huge search space for secondary structure prediction. The Zuker method decomposes a sequence into independent sub-sequences, achieving a huge reduction of search space [19]. First, the coefficients for three upper triangular matrices are calculated. Second, a back-trace algorithm is implemented to extract the actual pairs of the RNA input sequence. The values of the matrices are integers representing the optimal sub-sequence's free energy. The official edition of the Zuker algorithm is included in the UNafold package [20], which was used for this work. Software analysis of the algorithm showed that more than 99% of the algorithm's execution time is consumed for the calculation of the three matrices.

Smerdis *et al.* [21] proposes an FPGA-based IP core that implements the first stage of the algorithm on a reconfigurable platform. First, the input sequence (with some experimental parameters needed for the calculations), are loaded in the internal FPGA BRAMs. The implemented system is fully pipelined and consists of five basic modules that are used for the calculation of the matrices coefficients. Also, the implemented system offers high throughput due to its accessibility of the proper data for the coefficient calculations. The main drawback when mapping the Zuker algorithm on reconfigurable technology is the high memory usage. The implemented method uses portions of internal memory to store the input sequence. It is important that for double sized genetic sequence the memory size increases by about four times. Smerdis *et al.* [21] propose the use of external memory (DDR) to circumvent the above limitation. The Zuker system architecture was fully designed on a Virtex 5 Xilinx (XS5VSX240T) device. The clock rate of the design is 100 MHz with the FPGA utilization almost at 100%. The critical resource for the implemented system is the internal BRAMs, which eliminates the parallelization of the implemented system up to 58 parallel computation cores. The test datasets varied from 100 genetic bases up to 800 genetic bases. The implemented system offered execution speedup from 3x up to 10x compared to the sequential software execution on a high-end PC. In addition to this work, performed at TUC, there exist other results of FPGA implementations of the Zuker algorithm [22].

2) *Predator algorithm*

The Predator method [23] predicts the secondary structure of a protein sequence. The algorithm consists of four basic steps. The first part calculates the secondary structure propensities, based on amino acid tendency to form parallel b-bridges, antiparallel b-bridges, hydrogen bonds and turns. The second part of the algorithm searches a protein database with 560 proteins of known structure for homologs to the input sequence. The third part of the method takes as input the protein sequence and scores its non-homologous parts. The last stage of the method takes as input the output of the first three parts and applies some simple rules in order to find the final prediction. The Predator algorithm is a computationally demanding method in cases of very long input sequences. Software analysis of the Predator algorithm showed that 90% of the total execution time is spent on the search of homologs in the protein database and the scoring of non-homologous protein parts.

Smerdis *et al.* [21] presented the parallel implementation of the Predator algorithm on an FPGA. The implemented system consists of six independent parallel computing modules. Each one calculates one of the six possible formations which are needed for the first and the second part of the algorithm. Also, there is a module that gathers all the results from the previous modules and outputs the final results. The input sequence and the 560 proteins of known structure are stored in internal FPGA BRAMs. This architecture of the Predator system was mapped on a Xilinx Virtex 5 FPGA device (XC5VSX240T). The system was evaluated with several input sequences and the secondary structures predicted were verified against the original software implementation. The clock rate that the system achieved is 298 MHz. The performance speedup that the implemented system achieved when compared to the sequential software execution varies from 30x up to 50x.

D. *Gene Prediction*

Gene identification is one of the most important steps in the process of understanding the information contained in (complete) genome sequences. The gene prediction problem refers to the identification of biologically functional stretches of sequences (genes) in genomic DNA.

1) *Glimmer algorithm*

Glimmer [24] is a well known gene prediction algorithm for prokaryotic genomes. The main characteristic of Glimmer is the use of probabilistic models for gene extraction and evaluation. The Glimmer method consists of three algorithmic stages. First, the algorithm takes as input the coding regions of known genes and trains three (according to the reading frame position) independent Interpolated Markov Models (IMMs). Second, using the above models it calculates a score for the processed sequence. Finally, it resolves candidate gene sequences that overlap. Profiling the original Glimmer distribution with Intel's VTune tool showed that the scoring phase takes up to 80% of the total execution time.

Chrysos *et al.* [25] presented a reconfigurable architecture implementing the Glimmer algorithm. The proposed architecture consists of three independent parallel tree-structured Markov models that are used for the gene evaluation process. Each tree-structured Markov model was fully pipelined and each level of the tree consisted of BRAMs that kept the information needed for tracing down to the next tree level of the probabilistic model. Single floating point arithmetic was used, as the final result of the system was a probability score. This architecture was mapped on Virtex 5 FPGA device (XC5VSX240T) and its clock rate was at 150 MHz, achieving execution speedups from 1.14x up to 2.37x when it was compared with the official software implementation on a general purpose PC.

2) *GlimmerHMM algorithm*

GlimmerHMM [26] is a successor of the Glimmer algorithm, which is used for eukaryotic gene prediction. The GlimmerHMM algorithm is divided into three phases: training phase, identification phase and resolving overlap phase. The GlimmerHMM software is Unix-based and is designed for single-core systems;90% of its total execution time is spent in the scoring phase. The main differences with the Glimmer algorithm are that it constructs and uses 4 Markov chains, it uses slightly different processing and it takes longer genetic chains, which means longer execution times. Chrysanthou *et al.* [27] presented a parallel accelerator of the GlimmeHMM algorithm, implementing the four hidden Markov Models of the algorithm in parallel. Also, a fully pipelined structure with throughput one clock cycle was used for implementing each one of the HMMs. The proposed system used a PCI-Express communication interface, which was built for the data exchange between the implementation platform and the PC.

The FPGA-based GlimmerHMM system was implemented on a Virtex-5 FPGA (XC5VLX110T). The critical resource for the implemented architecture is the internal BRAMs, which eliminated the accuracy of the final results by using 16-bit floating point arithmetic. The module was clocked at 126 MHz, whereas the clock rate for the total system was 62.5 MHz, which was the operation frequency of the PCI-Express. The bottleneck of the implemented system is the intercommunication of the FPGA with the CPU. The FPGA implementation, despite the low-rate data transmission, outperforms both the GPU and CPU implementations of the official GlimmerHMM software, offering total execution speedup of about 2x [27].

E. *Phylogenetic Tree Reconstruction*

A phylogenetic tree or evolutionary tree is a tree-like structure that shows the inferred evolutionary relationships among biological species. The Phylogenetic Likelihood Function (PLF) is the most widely used optimality criterion to score and, thus, choose among distinct evolutionary scenarios (phylogenetic trees). The PLF is used by many program packages, like RAxML [28]. The PLF method is a highly compute- and memory-intensive a two stage process, taking as input a tree structure of the taxonomic units with fixed topology and fixed parameters. First, it tracks down a pair of child nodes in the given tree for which the likelihood vector at the common ancestor has not already been computed. Second, it calculates the likelihood vector entries of the common ancestor and it prunes out the child nodes. These steps are executed recursively until the likelihood vector of the root node has been calculated. Software analysis showed that the programs that use PLF spent more than 90% of their execution time on PFL calculation. Alachiotis *et al.* [29] proposed a dedicated FPGA-based architecture that computes the PLF. Their implementation exploits the fine grained instruction level parallelism of the PLF, despite the fact that, both ML and Bayesian phylogenetic inference also provide a source of coarse-grained parallelism. Their basic computing cell performed floating point additions and multiplications for each parent node. Double precision floating point arithmetic was used for consistency with the software implementation. The proposed architecture was mapped on a Xilinx Virtex 5 FPGA device (XC5VSX240T) and its clock rate was at 101 MHz. The critical resources for the implemented architecture were the BRAMs and the DSPs, which were used for the double floating point arithmetic. The performance of the proposed system was up to 8 times faster than the multi-threaded code (up to 16 parallel threads).

F. FPGA-Based Implementations: Conclusions

The previous sections described several architectures, which mapped well-known bioinformatics algorithms on standalone FPGA platforms. The described implementations showed that reconfigurable technology offers significant advantages on the execution performance of CPU intensive bioinformatics algorithms. More specifically, FPGA technology offers substantial speedups vs. conventional computers. Although it was not discussed in this paper, reconfigurable technology also offers significantly lower energy requirements for the entire calculation when compared cluster or grid platforms Running the same algorithms.

However, there are still several open issues before FPGA technology can be widely adopted by the bioinformatics community. These issues are not inherent problems of FPGA devices but of FPGA based platforms. Such issues are:

- The proposed systems are not integrated with the official software distributions and in some cases they produce similar but not identical results.
- Inherent FPGA capabilities, such as fast serial I/O interfaces, are not fully exploited yet.
- Internal memory (BRAM) is the critical resource of most of the proposed architectures. External memory needs to be used in order to offer higher levels of parallelization, with the associated I/O memory problems appearing in general-purpose computing.

III. State of the Art Platforms

Besides standalone systems, typically boards with a cost of a few hundred Euros, there exist complete multi-FPGA platforms which can be used as application-specific hardware accelerators. We present two such platforms and their usage in bioinformatics applications.

A. CONVEY HC-1

The Convey HC-1 is a hybrid-core computer platform that combines an FPGA-based coprocessor with a 64-bit host dual core Intel Xeon processor running at 2.13 GHz. The HC-1 host runs a 64-bit Linux kernel with a modified virtual memory system, which retains the memory coherence between the host processes and the FPGA coprocessor.

The Convey HC-1 coprocessor consists of three main components: the Application Engines (AEs), the Memory Controllers (MCs), and the Application Engine Hub (AEH). The Application Engine Hub (AEH) is the interface between the coprocessor and the host processor. Eight memory controllers (MCs) are used to support the bandwidth demands of the coprocessor. Each memory controller is implemented on a different non-user controlled FPGA and it is connected to two standard DDR2 memory dimms.

The Application Engines (AEs) are four user-programmable Virtex-5 XC5VLX330 FPGAs, which implement the extended instructions. Each extended instruction consists of a “personality” that is a particular configuration of these FPGAs. The AEs are connected to the AEH by a command bus and to the memory controllers via a network of point-to-point links that provide very high sustained bandwidth. The AEs are interconnected with 668 Mbytes/s full duplex links. Each AE has a 2.5 GB/s link to each memory controller which leads to a theoretical peak bandwidth of 20 Gbyte/s per AE when striding across the eight different memory controllers.

B. Bee Cube

BeeCube is the third generation of a commercial FPGA-based computer system, evolved from the Berkeley Emulation Engine (BEE) [13] at the University of California (Berkeley). The platform is jointly developed by Microsoft Research, UC Berkeley and BEEcube Inc. The system is built to host large-scale hardware-based computation engines and to perform runtime emulation of complete multicore processor architectures and application-specific systems.

The BeeCube emulation platform is designed using Xilinx FPGAs interconnected in a ring architecture. The FPGAs are used to execute algorithms directly on hardware fabric for both speed and flexibility. Users can interact with the engine through an instance of the TinyOS operating system running on a host Microblaze processor, which is also mapped on the FPGA fabric along with the architecture of accelerated algorithm. Each platform supports high-speed interconnection such as Gigabit Ethernet and SATA, for communication to host workstations and/or other BeeCube platforms. The system can be configured either using a low-level HDL (Hardware Description Language) flow or through high-level flow using the MATLAB and Simulink tools.

IV. Case Studies

A. BLAST Execution on the Convey HC-1

The design process to map BLAST algorithm at Convey HC-1 platform followed the next steps:

- Profile of the NCBI software distribution.
- Map the most time consuming part of the software on FPGA device.
- Integrate NCBI software with the special purpose hardware.
- Validate results of the new SW/HW system.

1) Profiling

A profiling study of NCBI-BLASTn with default parameters using the open source GNU gprof profiler showed that the time spent in each step of the algorithm can vary substantially depending on the input datasets. Several queries and two of the largest NCBI’s genetic databases were used for our profiling tests (nucleotide collection (nt), Environmental samples (env_nt)). Our tests showed that the ungapped extension and the seeding functions are the most computationally intensive parts of the BLAST algorithm. As shown in Figure 1 these two functions consume over 50% of the total execution time for ungapped alignments.

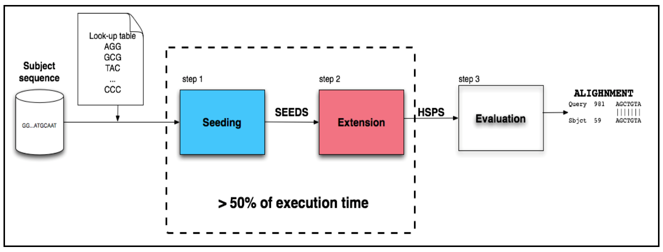

Figure 1. BLAST Profiling

2) Design

A new architecture has been designed, based in previous implementations. At the new architecture, data structures are exactly the same as the official NCBI software distribution making this architecture more demanding on memory resources.

3) Integration

As described above, the next step of the system development was the integration of the FPGA-based system with the software implementation. First, the official BLAST software distribution was downloaded and it was compiled on the HC-1 server. Second, the implemented IP core was mapped on the HC-1's FPGA devices using the corresponding "connections" with the memory and the CPU. The part of code that was implemented by the dedicated IP core was replaced by a "function call" to the HC-1 coprocessor, which produced results identical to the replaced functions. The flowchart of the BLAST algorithm software-hardware co-design is presented in the Figure 2.

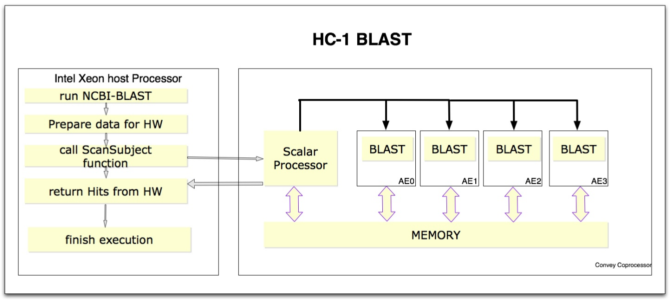

Figure 2. BLAST Execution on the Convey HC-1

4) Validation

Several tests with different datasets took place on the implemented system. The results of the final system were compared and they were found identical to the results of the NCBI-BLAST software distribution. Concluding the implementation of the BLAST algorithm on the new platform offers significant advantages:

- This system offers identical results with the NCBI's software solution.
- High memory bandwidth offers a great solution to the I/O problem for input database streaming.
- It offers high parallelism and speed characteristics and is scalable up to the FPGAs devices size.
- The high-speed external memory can be used in addition to the on-FPGA BRAM.

As this design is an ongoing research project and not fully optimized yet, it is expected to offer at least one order of magnitude speed up against NCBI implementation, whereas the fully identical results of the system vs. the NCBI software have been verified already.

B. CAST Execution on the Bee CUBE

The capabilities of the highly scalable FPGA-based CAST hardware architecture are indeed limited only by the amount of reconfigurable fabric present on the FPGA used, and the speed of the I/O communication channel that propagates the protein sequence streams from the host PC into the FPGA fabric. As such, using larger FPGA devices and particularly state-of-the-art platforms can further increase performance.

To illustrate this, multiple instances of the CAST hardware architecture where loaded on BeeCube Hardware Emulation platform [13] as a case study for using such engines in high-performance bioinformatics applications. Each of BeeCube's LX155T FPGAs can accommodate up to eight instances of FPGA-based CAST when adding the necessary I/O logic needed (FIFO buffers) to propagate the protein sequence streams to each CAST instance. In the experimental platform, eight instances of the CAST architecture were loaded on BeeCube's FPGA fabric in order to observe initial results. The protein sequences were loaded using a Compact Flash memory, and transferred to the BeeCube memory structure through a MicroBlaze processor.

Initial results showed a near-linear speedup (7.6 times faster) of the BeeCube implementation over the stand-alone implementation demonstrated in [15] and prove that higher performance is indeed achievable when the design is scalable. However, the ideal 8x speedup was not reached despite the fact that the CAST hardware architecture is highly scalable, as a more optimized protein sequence partitioning scheme was not considered in this initial study. Arbitrarily partitioning the sequences into streams to drive each instance fails when applied to the CAST algorithm, in which the amount of iterations for each sequence is dynamically determined at runtime. Given that the algorithm's performance relies heavily on the way that the input sequences are fed to the system, an optimal resource allocation algorithm could jointly be developed amongst biologists and hardware design engineers that will enable linear speedups. This further emphasizes the potential for research and development in the particular field as such large-scale systems can indeed be used as computationally efficient high-end platforms for genomic data processing.

V. CONCLUSIONS

A broad range of bioinformatics algorithms has been shown to have very promising execution times on FPGA standalone systems. These systems operate on the standard datasets and testbenches of the bioinformatics community. In many cases the results are bit-for-bit equal to software executing the same algorithms, whereas in some cases the results are small supersets of the software runs of these algorithms. There still exist open challenges to overcome, such as I/O speeds, standardization of user interfaces, and seamless architectures, to name a few, in order to develop usable and energy-efficient bioinformatics supercomputers for wide use by the scientific community. The use of emerging high-end platforms such as the Convey HC-1 and the BeeCube are expected to overcome I/O, memory, and integration with software issues and result into usable supercomputers for high-throughput bioinformatics.

ACKNOWLEDGMENT

This work is co-financed by the European Union (ERDF) and national resources of Greece and Cyprus.

REFERENCES

[1] S. Altschul, W. Gish, W. Miller, and E. Myers, "Basic Local Alignment Search Tool" *J. Mol. Biol.*, vol. 215, pp 403-410, 1990.

[2] E. Sotiriades, C. Kozanitis, A. Dollas, "FPGA based Architecture of DNA Sequence Comparison and Database Search", Proceedings 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, p 193, ,at the 13th Reconfigurable Architectures Workshop Rhodes, Greece, 25-29 April, 2006

[3] E. Sotiriades, C. Kozanitis, A. Dollas, "Some Initial Results on Hardware BLAST Acceleration with a Reconfigurable Architecture", Proceedings 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, p 251 , at the 5th IEEE International Workshop on High Performance Computational Biology (HiCOMB2006), Rhodes, Greece, 25-29 April, 2006.

[4] E. Sotiriades, A. Dollas "A General Reconfigurable Architecture for the BLAST algorithm", The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Special Issue on Computing Architectures and Acceleration for Bioinformatics Algorithms, Kluwer Academic Publishers Volume 48, Issue 3 Pages: 189 – 208, September, 2007.

[5] T. Oliver, B. Schmidt, D. Maskel ""Reconfigurable Architectures for Bio-sequence Database Scanning on FPGAs", IEEE Transactions on Circuits and Systems II, Vol, 52, No, 12, pp, 851-855, 2005.

[6] K. Muriki, K. Underwood, and R. Sass, "RC-BLAST: Towards an open source hardware implementation," In Proceedings of the International Workshop on High Performance Computational Biology (2005).

[7] M. Herbordt, J. Model, Y. Gu, B. Sukhwani, T. VanCourt, "Single Pass, BLAST-Like, Approximate String Matching on FPGAs"14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06), pp, 217-226, 2006.

[8] M. Herbordt, J. Model, B. Sukhwani, Y. Gu, and T. VanCourt, "Single pass streaming BLAST on FPGAs", Parallel Computing, vol. 33, issue 10-11, pp 741-756, 2007.

[9] J.C. Wootton, S. Federhen: Statistics of local complexity in amino acid sequences and sequence databases. *Computers & Chemistry* 1993, 17(2):149-163.

[10] G. Gill, E. Pascal, Z.H. Tseng, R. Tjian: A glutamine-rich hydrophobic patch in transcription factor Sp1 contacts the dTAFII110 component of the Drosophila TFIID complex and mediates transcriptional activation. *Proceedings of the National Academy of Sciences, USA* 1994, 91(1):192-196.

[11] S. Karlin, L. Brocchieri, A. Bergman, J. Mrazek, A.J. Gentles: Amino acid runs in eukaryotic proteomes and disease associations. *Proceedings of the National Academy of Sciences, USA* 2002, 99(1):333-338.

[12] V.J Promponas, A.J. Enright, S. Tsoka, D.P. Kreil, C. Leroy, S. Hamodrakas, C. Sander, C.A. Ouzounis: CAST: an iterative algorithm for the complexity analysis of sequence tracts. *Bioinformatics* 2000, 16(10):915-922.

[13] "BeeCube Hardware Emulation Platform." [Online]. Available: http://beecube.com. [Accessed: 20-Jun-2012].

[14] "Xilinx Products Guide" [Online]. Available: http://www.xilinx.com. [Accessed: 20-Jun-2012].

[15] A. Papadopoulos, V.J. Promponas, T. Theocharides, "Towards systolic hardware acceleration for local complexity analysis of massive genomic data", ACM Great Lakes Symposium on VLSI - GLSVLSI 2012, Salt Lake City, May 2012.

[16] K. Katoh, K. Kuma, H. Toh, T. Miyata, MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, vol 33, pp. 511-518,, 2005.

[17] M. Lakka, A. Desarti, G. Chrysos, E. Sotiriades, I. Papaefstathiou, A. Dollas, "Reconfigurable Computing IP Cores for Multiple Sequence Alignment", In Proceedings of BIOINFORMATICS, pp. 216-221, 2011.

[18] C. Notredame, D.G. Higgins, J., Heringa, T–Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, vol. 302, issue 1, pp. 205-217 2000.

[19] M. Zuker, P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information", *Nucleic Acids Research*, vol. 9, pp. 133-148, 1981.

[20] N. R. Markham, M. Zuker, "Unafold:Software for nucleic acid folding and hybridization", *Methods in Molecular Biology*, vol 453, pp 3–31, 2008

[21] M. Smerdis, P. Dagritzikos, G. Chrysos, E. Sotirades, A. Dollas, "Reconfigurable Systems for the Zuker and the Predator Algorithms for Secondary Structure Prediction of Genetic Data", In the proccedings of Field Programmable Logic (FPL), 2010.

[22] A. Jacob, J. Buhler, R. Chamberlain, "Rapid RNA Folding: Analysis and Acceleration of the Zuker Recurrence," Field-Programmable Custom Computing Machines, Annual IEEE Symposium on, pp. 87-94, 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, 2010

[23] D. Frishman and P. Argos, "Incorporation of long-distance interactions into a secondary structure prediction algorithm", *Protein Engineering*, vol 9, pp.133-142, 1996.

[24] S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models, *Nucleic Acids Research* 26:2 (1998), 544-548.

[25] G. Chrysos, E. Sotiriades, I. Papaefstathiou, and A. Dollas, "A FPGA based coprocessor for gene finding using Interpolated Markov Model (IMM)," in Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09), pp. 683–686, August 2009.

[26] W.H. Majoros, M. Pertea, and S.L. Salzberg, TigrScan and GlimmerHMM: two open-source ab initio eukaryotic gene-finders *Bioinformatics*, Vol. 20, pp. 2878-2879,2004.

[27] N. Chrysanthou, G. Chrysos, E. Sotiriades, I. Papaefstathiou, Parallel accelerators for GlimmerHMM bioinformatics algorithm, In the Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1-6, 2011.

[28] A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.

[29] N. Alachiotis, A. Stamatakis, E. Sotiriades, and A. Dollas, "A reconfigurable architecture for the phylogenetic likelihood function," in Proceedings of the 19th International Conference on Field Programmable Logic and Applications (FPL '09), pp. 674–678, 2009.