Some Initial Results on Hardware BLAST Acceleration with a Reconfigurable Architecture

Euripides Sotiriades Christos Kozanitis Apostolos Dollas

Department of Electronic and Computer Engineering
Technical University of Crete
73100 Chania, Crete
Greece
(Contact author: E. Sotiriades: esot@mhl.tuc.gr)

Abstract

The BLAST algorithm is the prevalent tool that is used by molecular biologists for DNA Sequence Matching and Database Search. In this work we demonstrate that with an appropriate reconfigurable architecture, BLAST performance can be improved with a single-chip solution 5 times over a specialized and optimized computer cluster, or 37 times over a single computer. These initial results account for I/O and are very encouraging for the development of a large scale, reconfigurable BLAST engine.

1. Introduction

DNA Sequence Matching and DNA Database Search are two computationally demanding problems. These problems are addressed since the early 1970's and still occupy substantial computational resources to date. One of the first applications of Field Programmable Gate Arrays (FPGAs) in the early 90s was in this area [10][11]. Several algorithms have been developed and a lot of computational power has been devoted for this problem. Previous reconfigurable (i.e. based on FPGA logic) platforms for boosting DNA Sequence Matching and Database Search have used primarily dynamic programming algorithms. This paper presents a new architecture for effective use of reconfigurable logic for BLAST algorithm speedup over general-purpose computers. The new architecture is described in [20]. The contributions of the present work vs. [20] are:

I. A detailed description of how the elements of the BLAST algorithm can be speeded up by using reconfigurable computing
II. Detailed performance results and comparisons of the new architecture vs. conventional computing platforms (single computers or clusters), even highly optimized ones for BLAST execution

Limited information from [20] has been included in this work for readability purposes, and is indicated accordingly.

1.1 Previous Work

The genome of many organisms has been sequenced to date but the process is due to be performed for many others. Databanks with sets of genome information have been created and categorized according to their contents (DNA or proteins). Major databanks are GenBank [1] at NCBI which maintains all DNA sequences that are made public; and EMBL [2] which is a large DNA archive in Europe. In addition, important DNA archives are kept in DDBJ [3], and GSDB [4]. Regarding protein archives, PIR [5] in the USA and Swiss-Prot [6] in Europe are the most important databases. This process started over the last two decades and continues to date, with an exponential growth over the entire period. In 1982 the first 606 sequences and 680,338 base pairs were added to GenBank. The total number of sequences at 2004 was 40,604,319 and of base pairs 44,575,745,176, giving an average *annual* growth rate of 65.71% for sequences and 65.54% for base pairs.

It has been calculated [8] that NCBI databank size grows at a more rapid rate than Moore's law indicates for the transistor number in a chip, and consequently than the growth in single-processor computer power. For that reason, and despite the increasing efforts of many institutions, DNA Sequence Comparison and Database Search remains a challenging problem. The BLAST algorithm [9] is considered to be the basic tool for molecular biologists, and its acceleration is a worthwhile endeavor.

1-4244-0054-6/06/$20.00

2. The BLAST Algorithm

This section includes information from [20] and has been included for purposes of readability of this work. BLAST is the acronym for Basic Local Alignment Search Tool and was first presented in [9]. A family of implementations has been developed, depending on the nature of data to be processed. Depending on query and database data types, each BLAST implementation is named BLASTp when the query is an amino acid and the database is a protein, BLASTn when both the query and the database are nucleotides, BLASTx when the query is nucleotide translated and the database is protein, tBLASTn when the query is an amino acid and the database is a nucleotide translated, and finally tBLASTx when the query and the database are nucleotides translated. The inputs of the algorithm are the genetic sequence database (or a part of it such as the human genome) and a query which tries to find areas of similarity in the database. The outputs of the algorithm are the positions of the areas of these two strings that have similarity, as well the score of these similarities. Each of these pairs, comprising of a database area and a query area, is called a High Score Pair (HSP). The score has significant value for biologists because it is used to compute several variables, of which the e-value is the most important.

The algorithm consists of three steps. In the *first step* the query is compiled to form a list of length w substrings. These substrings are called W-mers and are all the contiguous substrings of length w of the query sequence. The *Second step* is the search of the database for "hits". After the word list generation, the database sequences are searched for an **exact** match between any substring of the w-mers list and the database sequence. Every word of the word list found in the database is called hit and it is possible to be part of a High Score Pair (HSP). The list of the generated "hits" is processed to the *third step*. Each substring, which made a match in the second step, is extended locally in both directions as long as the score of this substring no longer gets improved following the scoring rules.

3. Reconfigurable Computing for DNA Sequence Comparison and Database Search

Reconfigurable logic has been used for well over a decade as a means for solving computationally intensive problems. FPGA devices vary according to the manufacturing, but invariably each device consists of several programmable resources such as I/O blocks, a core of configurable logic (e.g. configurable logic blocks - CLBs) and their interconnection. In addition there may be resources such as memory, or even embedded processors.

The devices are mostly SRAM based which means that all the resources (including the logic and the interconnect) are programmed by means of a protocol that takes memory contents and maps them to look-up table values or interconnect control signals. Figure 1 shows the general architecture of a typical FPGA device.

In the last decade several platforms have been built with FPGAs to boost performance of algorithms for DNA, to be described in the next section.

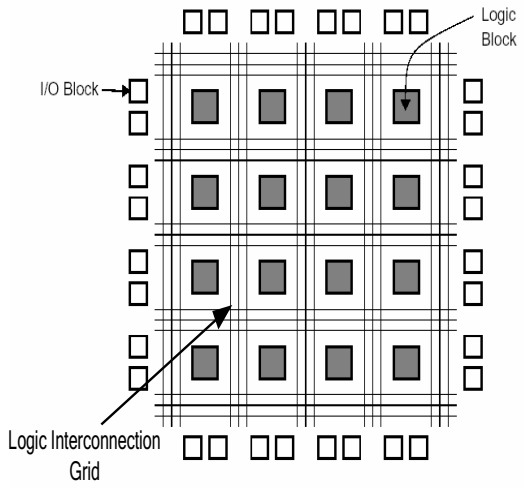

Figure 1 Generic FPGA Structure

3.1. Reconfigurable Hardware for Bioinformatics

To date, FPGAs have been proved to be a powerful platform for performance boosting for string matching types of problems such as Intrusion Detection Systems in computer networks or data mining. The DNA Sequence Comparison and Database Search differ from other string matching problems due to the limited alphabet on one hand, and the exceptionally long database size on the other.

In the specific domain of Sequence Comparison and Database Search Hoang et. al. [10][11] use the FPGAs of the venerable SPLASH 2 machine to implement the Needleman Wunsch algorithm [12]. Using JBits S. Guccione et. al. [13] implement the Smith Waterman [14] matching algorithm. The same algorithm was implemented at Virginia Tech. [15] and more recently at Nanyang Technological University [16]. All these implementations take advantage of the dynamic programming algorithm features that can be efficiently solved with a systolic array processor. In general, FPGA internal structures are well suited for systolic processor design. There has been limited work on BLAST using reconfigurable logic, however. Several efforts towards

BLAST speed up for special cases can be found, with the latest being miBLAST [24].

Heuristic algorithms FASTA [17] and BLAST don't seem do be suitable to FPGAs. Muriki et. al. [18] is the only detailed bibliographic reference for FPGA BLAST implementation. Their purpose was to improve performance of NCBI BLAST implementation with the use of FPGAs maintaining the open source feature of NCBI. At their work they profile the execution of NCBI software and they found at which point (file) of the software the CPU consumes most of the time. The detected code was then replaced with a call to a procedure that was using an FPGA platform for execution boosting. It was projected that code would run 35 times faster but actual measurements show almost 5 times slower execution. This result is not discouraging given the specifics of the technology that the authors used, but it does highlight the difficulties in developing high performance architecture. The contribution of this work was that there was a detailed study of CPU execution time distribution during the execution of the BLAST algorithm, an actual run of BLAST on FPGAs and the open source feature that this project managed to keep. On the other hand, the very old FPGA technology that was used, in combination with limited architectural and design efforts towards optimum performance led to a poor system performance, largely due to Input/Output (I/O) bottlenecks.

One last notable BLAST acceleration system is the DeCypher commercial product by Time Logic, Inc. The system is a PCI card that can be attached to a server and it is based on FPGA technology [22]. Several impressive but not detailed results of DeCypher have been announced [23]. Unfortunately, lack of information about the architecture itself (number of chips, I/O, architecture type, etc.) as well as how the performance is calculated (types of queries, size of database, version of BLAST, etc.) do not allow for comparisons with our present work.

4. The TUC Architecture

The Technical University of Crete (TUC) architecture, described in more detail in [20] and analyzed in this work, was designed for BLASTn small query implementation (1000 letters) regardless of the data base size. NCBI codes consist of several hundreds of files calculating the BLAST algorithm and exporting several numbers which have biological meaning. All these numbers are calculated based on the score of HSP. These calculations produce substantial computing load but the most significant part of the computation power is consumed to find every HSP and extend it, calculating its score. Previous efforts for hardware implementation of BLAST using profiling show that almost 80% of CPU time spend for these calculations [18].

The TUC architecture is divided into N identical computing machines, each one of which has two main components and implements all three steps of the algorithm. Input data have a width of 2N bits, which are part of N different channels. Every channel drives one of the N computing engines. At the first component of each machine, the W-mer list is created and stored in the memory. As the data stream of the database passes, a search for a hit (match) is being performed. Every hit that is found activates the second component of the architecture which starts to extend it, implementing the third step of the algorithm. The general design of the architecture is shown at Figure 2.

5. Recent Technology FPGA Features for BLAST Implementation

The Input Output (I/O) of data proved to be the bottleneck of performance of previous efforts to implement hardware for BLAST [18]. Present generations of FPGAs offer an extremely wide I/O bandwidth. To illustrate, FPGA's of Xilinx corp. have the so-called ROCKET I/O transceiver which is embedded in the device families Virtex-II Pro and Virtex-4. Such a transceiver allows up to 10.3125 Gb/s [19] transfer rates per transceiver with a maximum number of 20 transceivers on a single device. That feature gives an aggregate baud rate of 206.25 Gb/s per device. This is the baud rate of a single chip according to datasheets but the actual bit rate is lower and estimated to be about 8 Gb/s per channel or 160 Gb/s per device. These baud rates are significantly higher than the PCI baud rate which was the bottleneck in [18] and even the DDR2 baud rate from main memory to CPU. A typical value for PCI baud rate is 1 Gb/s and for DDR2 (main memory to CPU) it is 8.5 Gb/s. The significant difference between the baud rates between a FPGA device and what a conventional processor can receive from main memory or a PCI-based co-processor proves that even if the performance bottleneck of the BLAST algorithm is the I/O, as in [18], it can be overcome. However, there needs to be proper use of FPGA technology and the appropriate memory subsystem architecture, from which the database will be streamed into the processor, because I/O is non-trivial.

A very important feature of modern FPGAs is the existence of embedded Blocks of RAM (BRAM for Xilinx) which can offer a large amount of RAM with flexibility, using them in any design as small RAMS of some Kbits each or as one bigger RAM of some Mbits. For the BLAST algorithm it is very important to keep inside the processor the data of the database stream under examination, and indeed keep it for a time proportional to

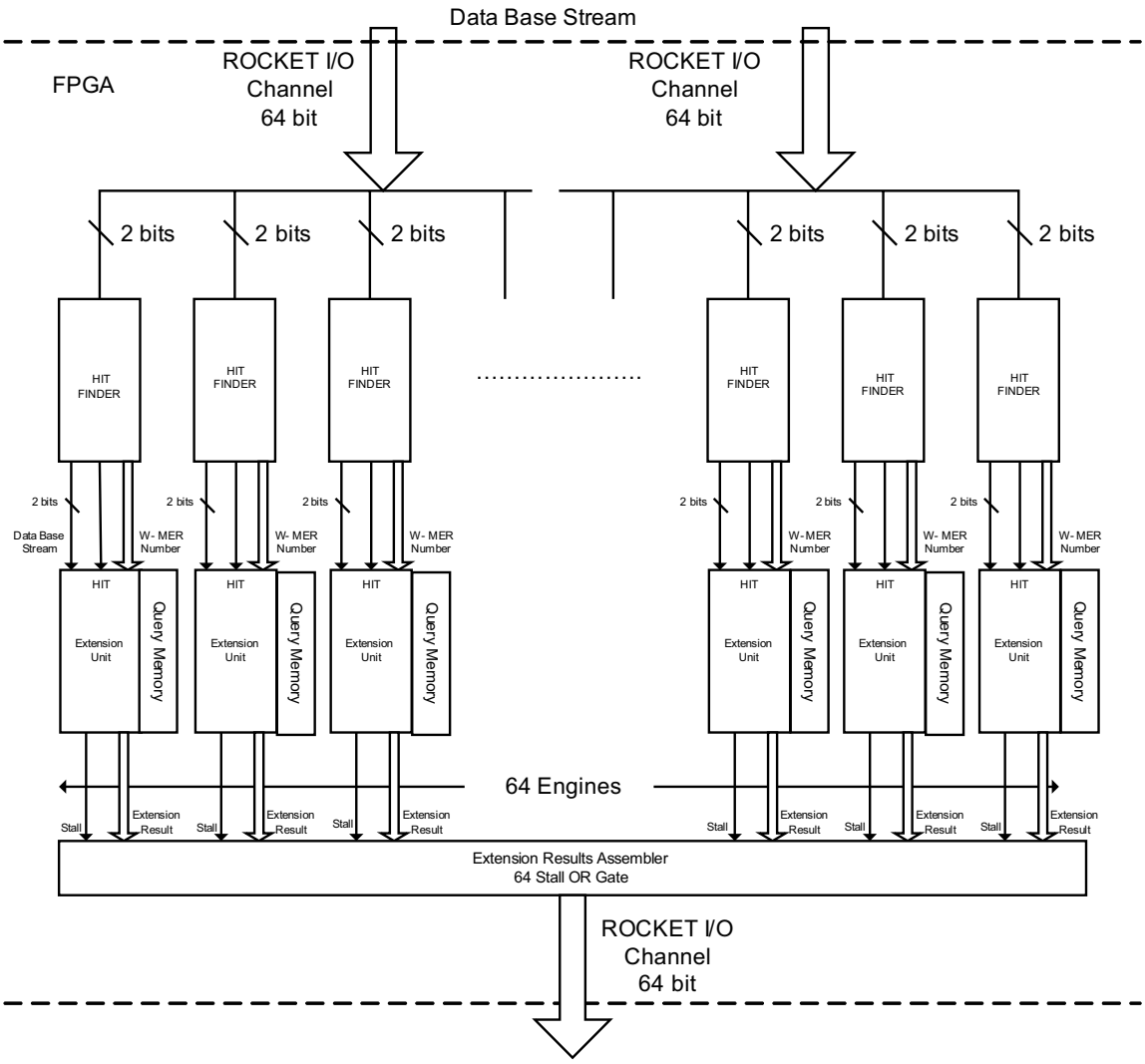

Figure 2 General TUC Architecture

the query length. The BRAMs combined with the existing distributed RAMs, extend the designers capabilities to manipulate larger data sets, which is of very high importance for the BLAST algorithm.

Therefore we have seen that key elements for BLAST acceleration on present-day FPGAs are the availability of very fast I/O for streaming the database data into the processor, and the internal RAM which is sufficient to hold the entire string of a query. As the entire database needs to be streamed into the system, we observe that present-day FPGA's overcome the problems of previous generations *vis a vis* the BLAST algorithm. In addition, a resource of great importance is the core of a PowerPC processor that new Xilinx devices have embedded. These embedded processors have high computational power and can be used as processing elements that can manipulate memory and consequently data sets more efficiently.

6. TUC Architecture Implementation

The TUC Architecture has been coded in VHDL and exhaustively post place-and-route simulated for the VIRTEX-4 family using the 4VFX140FF1517-11 device. Such simulations are considered highly accurate by the computer hardware community and their results are quite dependable. The simulations were of actual string matching for existing large sequences, as described in the next section. In this first generation of the architecture three different experiments have been executed. In the first experiment a single computing machine has been implemented. This machine according

to post place and route simulations runs at 121 MHz and consumes 1% of the device resources. In the second experiment, 60 parallel computing machines of the same design have been placed in a single chip, and the clock speed was 103 MHz. The consumed resources were 36% of logic and 86% of BRAMs. In the third experiment the parallelization came up to 69 and the respective measurements were 101 MHz clock speed, 42% of logic resources and 100% of BRAMs. The BRAM is the critical resource that limits parallelization to 69 in the first generation of TUC Architecture.

7. TUC performance – Results

In order to evaluate the TUC architecture performance, measurements of the NCBI software have been made on conventional computers and on the new machine, with identical queries.

Runs of blast-2.2.12 were performed on a 2GHz Xeon with 2GB main memory running SUSE 9.1 Linux and the CPU usage was profiled. Five NCBI data bases of several sizes for a small query of 1000 letters were executed at the 2GHz Xeon and measured.

The same experiment was repeated with a Intel Pentium M 1,7 GHz with 1 GB main memory running Windows XP professional and an Intel P4 2,66 GHz with 1 GB main memory running Windows 2000. For Computers running Windows Intel VTune Performance Analyzer 7.2 was used and every measurement repeated 5 times. Results of these experiments are respectively on Tables 1, 2 and 3.

The averages in the tables are arithmetic averages.

DataBase name	**Database Size (characters)**	**Run Time (sec)**	**Throughput (characters/sec)**
ecoli.nt	4,662,239	0.024	194.25 10^6
drosoph.nt	122,655,632	0.482	258.33 10^6
month.nt	386,242,580	1.753	220.56 10^6
env_nt	1,061,221,997	1.190	891.63 10^6
igSeqNt.ftptemp	44,419,359	1.397	31.77 10^6
Average	**323,840,361**	**0.968**	**319.25 10^6**

Table 1 Measurements on XEON 2 GHz / Linux

DataBase name	**Database Size (characters)**	**Run Time (sec)**	**Throughput (characters/sec)**
ecoli.nt	4,662,239	0.045	102.85 10^6
drosoph.nt	122,655,632	0.364	337.32 10^6
month.nt	386,242,580	1,303	296.50 10^6
env_nt	1,061,221,997	3.670	289.19 10^6
igSeqNt.ftptemp	44,419,359	0.174	255.43 10^6
Average	**323,840,361**	**1.111**	**256.26 10^6**

Table 2 Measurements on Intel M 1,7 GHz / Windows XP

DataBase name	Database Size (characters)	Run Time(sec)	Throughput (characters/sec)
ecoli.nt	4,662,239	0,039	118.45 10^6
drosoph.nt	122,655,632	0.309	396.32 10^6
month.nt	386,242,580	1.022	378.10 10^6
env_nt	1,061,221,997	3.200	331.63 10^6
igSeqNt.ftptemp	44,419,359	0,160	277.40 10^6
Average	**323,840,361**	**0.946**	**300.38 10^6**

Table 3 Measurements at Intel P4 2,66GHz / Windows 2000

Number of Processors	Type of Processors	Time (sec)	Database Size (characters)	Actual System Throughput (characters/sec)	Actual Throughput per Chip (characters/sec)
1	POWER3	43.63	4 10^9	91.68 10^6	91.68 10^6
	Model 681 1.1	21.32	4 10^9	187.62 10^6	187.62 10^6
2	POWER3	24.09	4 10^9	166.04 10^6	83.02 10^6
	Model 681 1.1	11.39	4 10^9	351.18 10^6	175.59 10^6
4	POWER3	14.23	4 10^9	281.10 10^6	70.27 10^6
	Model 681 1.1	6.53	4 10^9	612.56 10^6	153.14 10^6
8	POWER3	9.25	4 10^9	432.43 10^6	54.05 10^6
	Model 681 1.1	4.33	4 10^9	923.79 10^6	115.47 10^6
16	POWER3	7.56	4 10^9	529.10 10^6	33.07 10^6
	Model 681 1.1	3.33	4 10^9	1201.20 10^6	75.07 10^6

Table 4 BLASTn Benchmarks with a Small Single Query and Large Database(from [21])

The NCBI software has been used as a benchmark from IBM for measurement of computing systems such as IBM 375 MHz POWER3-II multiprocessor (SMP) and the 1.1 GHz POWER4 pSeries 690 Model 681[21]. In this work several experiments have been made for several sizes of databases, queries and BLAST version. Out of these results BLASTn results for small queries were selected to be compared with all the other experiments and are presented on Table 4. It can be shown that the fastest system throughput is achieved with the 16 processor Model 681 1.1 system, which has a throughput of 1,201.20 10^6 characters/sec. However, the fastest single chip system is the IBM Model 681 1.1 with 187.62 10^6 characters/sec.

Finally, the TUC architecture performance is determined according to post place and route timing information of Xilinx software 7.1.03 which includes *Device speed data version: "ADVANCED 1.54 2005-05-25"* for the specific device. Table 5 has speed measurements for the three experiments.

Throughputs of all systems are presented at Table 6 and in Table 7 the speedup of TUC architecture against the other system is presented.

8. Conclusions

The presented results show that reconfigurable logic with proper technology exploitation can offer a flexible and effective platform for bioinformatics problems. BLAST seems to be the most popular tool for biologists but several other challenging problems still exist.

For BLAST implementation the second generation of TUC architecture has to be designed for bigger parallelization and broaderbandwidth, and several data mining techniques that are used in intrusion detection systems have to be used for further speed improvement. The designed architecture needs to be verified in commercial platforms with fast I/O,.such as ClearSpeed [25] or CRAY XD1 [26]. For I/O bandwidth, it still has to be proved that the Xilinx FPGA device capabilities can be matched by an external interface at the system

Number of Parallel Machines	Speed (MHz)	Width of Data Stream (characters)	Predicted Throughput (characters/sec)
1	121	1	121.20 10^6
60	103	60	6,192.58 10^6
69	100	69	6,924.84 10^6

Table 5 Speed and throughput of TUC Architecture

System	Predicted Throughput (10^6 characters/sec)
2GHz Xeon	319.25
1,7 GHz Intel M	256.26
2,66 GHz Intel P4	300.38
TUC Architecture N=1	121.20
TUC Architecture N=60	6,192.58
TUC Architecture N=69	6,924.84
IBM single chip	187.62
IBM System	1,201.20

Table 6 Systems Throughput

	SpeedUp of TUC Architecture N=1	SpeedUp of TUC Architecture N=60	SpeedUp of TUC Architecture N=69
2GHz Xeon	0.38	19.39	21.69
1,7 GHz Intel M	0.47	24.16	27.02
2,66 GHz Intel P4	0.40	20.61	23.05
IBM single chip	0.65	33.00	36.90
IBM System (16 chips)	0.10	5.15	5.76

Table 7 TUC Architecture SpeedUp

level. The design of a platform which addresses the I/O issues is an interesting open problem, which we will study next. It would be very interesting to create a flexible platform on which many bioinformatics problems can be executed and measured.

9. Acknowledgments

We wish to thank several people who contributed to this project. Georgia Adamopolou introduced us to bioinformatics and the specific problem. Support for the FPGA hardware in the lab came from donations from Xilinx, Inc. This work was funded from the Greek Ministry of Education Grand for the project GSRT "IRAKLEITOS: Research Scholarships for TUC", Program. TUC, sub-program #10, "Structures for Reconfigurable Computing".

References

[1] www.ncbi.nlm.nih.gov
[2] www.ebi.ac.uk
[3] www.ddbj.nig.ac.jp
[4] scop.wehi.edu.au/gsdb/gsdb.html
[5] pir.georgetown.edu
[6] www.isb.sib.ch
[7]www.ncbi.nih.gov/Genbank/genbankstats.html

[8] Paul G. Higgs, Teresa K. Attwood. Bioinformatics And Molecular Evolution. Blackwell Publishing, 2005.
[9] S. Altschul et. al. Basic Local Alignment Search Tool. J. Mol. Biol., vol. 215, pp 403-410, 1990
[10] D. Hoang et. al. FPGA Implementation of Systolic Sequence Alignment. Proceedings of the 2nd *International Workshop on Field-Programmable Logic and Applications*, Lecture Notes in Computer Science 705, pp 183-191, 1992.
[11] D. Hoang. Searching Genetic Databases on Splash 2. Proceedings IEEE *Workshop on FPGAs for Custom Computing Machines* (FCCM), pp 185-191, 1993
[12] S. B. Needleman and C. D. Wunsch. A General Method Applicable to the Search or Similarities in the Amino Acid Sequence of Two Proteins. J. Mol. Biol., vol. 48, pp 443-453, 1970.
[13] S. Guccione and Eric Keller. Gene Matching Using JBits. Proceedings of the 12th *International Conference on Field-Programmable Logic and Applications*, Lecture Notes In Computer Science; Vol. 2438, pp 1168-1171, 2002.
[14] T.F. Smith and M.S. Waterman. Identification Of Common Molecular Subsequences. J. Mol. Biol., vol. 147, pp 195-197, 1981
[15] K. Puttegowda et. al. A Run-Time Reconfigurable System for Gene-Sequence Searching. Proceedings 16th *International Conference on VLSI Design* pp 561 – 566, New Delhi 2003.
[16] T. Oliver et. al. Hyper Customized Processors for Bio-Sequence Database Scanning on FPGAs. Proceedings of the 2005 ACM/SIGDA 13th *international symposium on Field-programmable gate arrays* (FPGA), pp 229 – 237.
[17] Pearson, W. and Lipman, D. Improved tools for biological sequence analysis. In *Proceedings of National Academic Science*, 85, pages 2444–2448, 1988
[18] K. Muriki et. al. RC-BLAST: Towards a Portable, Cost-Effective Open Source Hardware Implementation. Proceedings 19th IEEE *International Symposium Parallel and Distributed Processing* (IPDPS), pp 196b-196b 2005.
[19]http://www.xillinx.com
[20] E. Sotiriades, C.Kozanitis, A.Dollas. FPGA based Architecture of DNA Sequence Comparison and Database Search. 13th *Reconfigurable Architectures Workshop* (RAW) 2006.
[21] C. P. Sosa et. al. Some Practical Suggestions for Performing NCBI BLAST Benchmarks on a pSeries™690System.
http://www.redbooks.ibm.com/abstracts/redp0437.html?Open.
[22] Roland Luethy and Christopher Hoover. Hardware and software systems for accelerating common bioinformatics sequence analysis algorithms. BIOSILICO Vol. 2, No. 1 January 2004.
[23]http://www.timelogic.com/benchmark_blast.html
[24] Y. J. Kim, A. Boyd, B. D. Athey, J. M. Patel. miBLAST: scalable evaluation of a batch of nucleotide sequence queries with BLAST. Nucleid Acids Research, 2005, Vol. 33, No. 13 pp 4335-4344.
[25]http://www.clearspeed.com/
[26]http://www.cray.com/products/xd1/