2016 IEEE International Parallel and Distributed Processing Symposium Workshops

Parallel Protein Identification Using an FPGA-Based Solution

Fabiola Casasopra, Gea Bianchi, Gianluca C. Durelli, Marco D. Santambrogio
Politecnico di Milano, Italy
Dipartimento di Elettronica Informazione e Bioingegneria
{fabiola.casasopra, gea.bianchi}@mail.polimi.it {gianlucacarlo.durelli, marco.santambrogio}@polimi.it

***Abstract*—The ability to rapidly identify a given protein from small subsamples (i.e. peptides) is at the basis of fundamental applications in the medical field. At the basis of protein identification we have a string matching problem which is computational intensive if we consider that the complexity of the algorithm scales with the length of the string and the number of sweeps of the database that are needed. In this paper we present an improvement for the FPGA-based string matching solution available in the literature improving the amount of parallelism exploited by the solution achieving a $1.63\times$ reduction of the energy needed for the task over the literature and a $5.75\times$ reduction when compared with high-end workstation.**

I. Introduction

One of the most common protein identification procedure is named tandem mass spectrometry (also known as MS/MS) and it includes a final string–matching phase. This stage involves a multiple peptide research against the entire human protein database in order to ascend to the original protein in the analized compound. The *mass spectrometry* technique allows the recognition of complex organic mixtures and also multi–elementary inorganic analysis with very high sensitivity. Within the medical field this technique is used to find and track specific protein biomarkers which is a major concern in various medical application such as cancer monitoring, pharmaceutical research, drug use or abuse. The recent technological improvements created a big growth of the protein and genomic databases, included the human databases. This improvement makes the tandem mass spectrometry much more accurate but also requiring a computational intensive string matching task. Many researches could benefit from a faster and energy efficient computation, keeping the accuracy high. For instance, within the field of serum proteomics there are many researches oriented to cancer monitoring, such as gastric cancer [1], that are obtaining promising results and that would benefit from a faster and more efficient procedure.

The energy requirement of the string matching task performed on a General Purpose Processor (GPP) is high and it grows with the number of identification performed. If we consider a large scale installation such as medical or research centers, we notice that this behavior greatly impacts the scalability and the mantainance cost of the system. Since the number of serum proteomics application has a rising trend, such as the number of the researches oriented in the biomarkers' area, we believe that reducing the energy demand could allow the big centers to perform more parallel analysis with a lower cost.

Recognizing an unknown protein's backbone structure starting from its peptides or from many fragments is analogue to a string matching problem: finding a substring in another string. In this particular application the substrings are the peptides, while the string is composed of the proteins included in the targeted human protein database. After all the peptides of the unknown protein have been searched, a score can be assigned to every matched protein in order to retrieve the best result. The complexity of this task grows with both the length of the peptide/substring L_{pep} and the string L_{pro}. For a *naive* algorithm, the worst case complexity is $O((L_{pro} - L_{pept} - 1)L_{pro})$; given that $L_{pro} >> L_{pept}$, it is approximable to $O(L_{pro}^2)$. Furthermore, in order to complete the identification, we need to compare the entire U number of peptides against the whole P number of proteins in the database. This means that for a single protein identification we need to use $U \times P$ times an algorithm that is $O(L_{pro}^2)$. If we think about a research center or a serum proteomics laboratory, we can imagine how many compound's analysis need to be executed every day. That number is the multiplicative factor to complete the complexity of this problem. If we name it $Num_{analysis}$, we obtain the following problem worst case complexity:

$$Num_{anaysis} \times U \times P \times O(L_{pro}^2)$$

In order to achieve the best accuracy in the string matching phase we need to use the complete proteomic human database, increasing the complexity and the execution time of a single research up to tens of seconds. Our proposal is an hardware–software approach with the implementation of a string matching algorithm, the well-known Knuth Morris Pratt (KMP), targeting an Field Programmable Gate Array (FPGA) device. This solution allows to well manage the complexity of the problem and to remain versatile, in order to keep the database updated and to research multiple unknown proteins. This versatility also enables to perform only once the FPGA system design phase, allowing to remove the development time from the applications overhead. This system is also designed to mantain the accuracy high as a software only implementation, and to parallelize the peptide's research instances against the database. In our work[1], we are investigating the benefit that reconfigurable hardware technology can have in terms of power efficiency when compared to a GPP when used to solve the protein matching problem.

In this paper we propose an improved implementation for the protein matching core with respect to our previous work

[1]Source code and other resources related to the paper are available at [2]

978-1-5090-3682-0/16 $31.00 © 2016 IEEE
DOI 10.1109/IPDPSW.2016.170

that demonstrated the benefit of using an HW implementation in terms of energy efficiency when comparing to a high-end GPP [3].

We conducted the evaluations targeting real world data for both the unknown protein and the database. We targeted the recognition of the Ki–67 protein, known as a gastric cancer biomarker[4] against the isoform proteomic FASTA database, provided by the Universal Protein Resource Consortium[5][6]. In order to simulate the real MS–MS Spectrometry procedure we obtained the Ki–67 fragments thanks to the ExPASy tool[7][8][2].

The remainder of the paper analyzes the current state of the art (Section II), describes in details the solution proposed in this work (Section III and presents the result obtained on the target device (Section IV). Finally Section V concludes the discussion and illustrates possible future works stemming from the topic discussed in this paper.

II. Related Work

The actual State Of the Art related to Field Programmable Gate Array (FPGA) designs for string matching applications is rich of studies which demonstrate the improvements obtainable with these hardware accelerators. For example, we can consider the work described by Bonesana et al. in [9] that introduces an implementation for the regular expression matching problem using three different FPGAs and obtaining improvements over the SW implementation.

As far as the specific contest of our paper, that is the protein identification problem whithin the tandem mass spectrometry method, we can consider the Orthogonal Parabix algorithm, illustrated in [10]. It is an optimization of the Parabix algorithm [11] to be used for real time protein identification. Furthermore, this work shows that the proposed algorithm achieves the best results in comparison with the classic approaches that can be found in literature: Rabin Karp [12], Boyer Moore [13] and Parabix [11].

On the other hand, a research group describes in [14] how they can obtain good results in the context of protein identification with the development of a HW/SW co–design approach targeting a FPGA device. In this way, they take advantage both of the flexibility of the software and the speed of the hardware. To further improve their work, they studied a method, explained in [15], that optimizes the string matching algorithm, in their case Aho-Corasick [16], by organizing the peptides that has to be searched based on their length. To continue their research, they made a HW implementation for the Aho-Corasick algorithm: they want to mantain a good performance optimizing the area used in the FPGA. To achieve this results, they decided to partition the Finite State Machine (FSM) that implements the string matching algorithm and then to check each time a small amount of peptide in parallel. However, the Aho Corasick algorithm has a big limitation in this HW implementation: we may have to create a new HW component when we have a new protein to identify. In this way, every time we have to check if the new protein of the unknown compound can be encoded with the FSM tiles that we already have. Otherwise, we have to synthesize a new HW components, but this means lot of time spent in the place and route operation, up to few hours.

As a result, we thought we could achieve improvements in speed and energy efficiency but remaining far more flexible and versatile as the protein identification application requires. In order to obtain this result, in our previous work [3] we focused on the implementation of the Knuth Morris Pratt (KMP) algorithm using a AVNET ZedBoard Development board as targed device. In this work, we proposed a solution that allows the match of every peptide with the proteins' database, obtaining good performance and energy consumption making our method viable for both small and large scale systems. The solution proposed did not have the need to generate a new HW solution for each set of peptides to match, but just send new data at each string matching request. In this paper, we illustrate a further optimization w.r.t. our previous work in the way the database and the overall data transfer is managed, greatly improving w.r.t our previous solution.

III. Implementation

In our work we decided to implement the Knuth Morris Pratt (KMP) exact string matching algorithm with C++ language. Its complexity, for a single pattern research, grows with both the length of the string and the pattern; the worst case complexity reaches $O(L_{pro} + L_{pept})$ thanks to a pre–elaboration on the pattern (the peptide sequence). This algorithm allows to reduce the worst case complexity of the entire application compared to the previously described *naive* approach as

$$Num_{analysis} \times U \times P \times O(L_{pro})$$

KMP works adding a sentinel character at the end of the strings and building a map based on the pattern, in order to jump to the next matching candidate as soon as the previous one ends. This algorithm choice permits to keep the application general also on Field Programmable Gate Array (FPGA)'s implementation. There are faster approaches available, such as Aho Corasick algorithm, but instead they loose versatility when we need to change the database and/or the pattern we need to match. The reminder of this section will now analyze the hardware design and its integration with the entire architecture, as well as the data transfer mode chosen. We designed a solution in which the SW side, with an host processor, handles all the control part of the program and only the string matching part is handled off to the HW.

Our previous work [3] implemented a HW core which received as input a protein and then a list of all the peptides that have to be matched. With respect to a naive implementation, also presented in [3], this solution allowed to greatly reduce the number of times each protein, and so the entire database, has to be sent to the HW core.

In this work we propose a modification of the core in such a way that each core is able to cache in local BRAM a portion of the database (i.e. a sequence of strings) and then, upon the arrival of a peptide, to match the peptide with the whole local database. A design of this type requires a lot of BRAM in order to store the database. Our target board however does not allow to store the whole database in local BRAM, so in designing the code we should take into account the fact that

[2]We simulated the chemical protein fragmentation with the trypsin enzime.

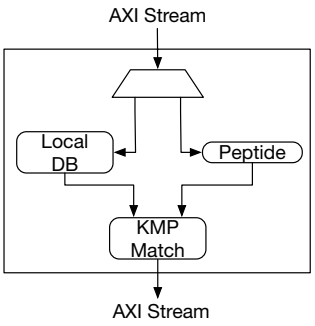

Fig. 1. High level representation of the implemented core.

we have to provide features to permit the update of the local database.

The layout of our cores is the one represented in Figure 1 where the four main parts of each core are sketched. Every core fetches data through a single AXI stream connection which receives data from a DMA core for efficient data transfer. A protocol has been devised such that the input stage of the core determines if the incoming data has to be used to update the local database or if the data represent a peptide to be matched. The Finite State Machine (FSM) controlling one core activity is represented in Figure 2. The core is initially waiting for a command, if a *update db* command is sent the core enters in the update process. It first expects to receive the length of the database and then starts reading the whole database from the input stream. When the transfer is completed the core returns in the initial state waiting for a command. Upon the *match* command, the core waits for the number of peptides to match, and then for each one of them it reads the actual peptide (first the length then the data) and performs the actual match using the KMP algorithm. After having matched all the peptides it returns in the initial state.

A single core implementation, however, can't lead to any particular performance benefit because the data transfer adds a huge overhead to the process. To improve performance we decided to instantiate multiple copies of the aforementioned core allowing them to be executed independently and in parallel. For this reason, each core has been synthesized with a local database capacity of 125KB, instead of the 500KB maximum[3]. We used a total of four cores in order to have each one of them connected directly to each one of the four

[3]The ZedBoard has more BRAM, but the remaining are used for the rest of the system (i.e. DMA cores, AXI interconnects, etc.).

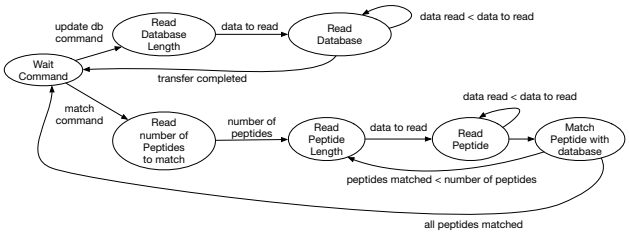

Fig. 2. Control flow of the HW core implemented in this work.

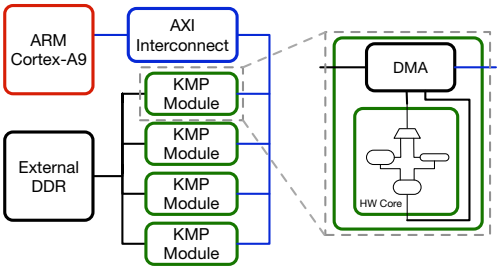

Fig. 3. Final system implemented on the AVNET ZedBoard.

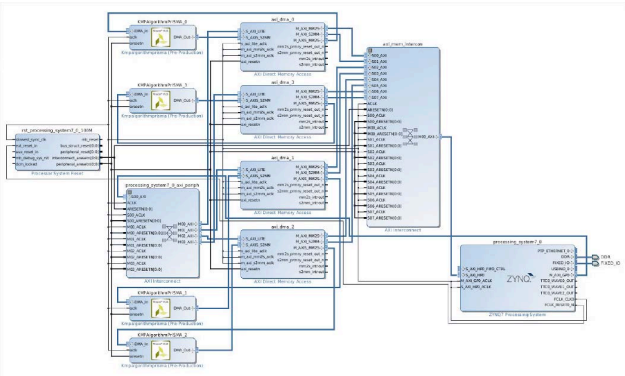

Fig. 4. Actual implementation of our latest design on the AVNET ZedBoard.

high performance memory access ports available on the *Zynq* processing system of the ZedBoard. Each port has been linked univocally to a Direct Memory Access (DMA) in order to mantain the best data exchange efficiency possible. As a result, we have one DMA for each core integrated inside the design.

Every core has been made using the Vivado High Level System (HLS) tool, included in the Vivado Design Suite[17], version $v2015.1$ (64 bit), coding with the C/C++ language. Starting from the basic implementation of KMP presented in [3], we modified it to fit the new version of the matching presented here. We have managed to modify KMP to execute the matching of one peptide against the entire database, as if it was a single string. The four cores have been implemented in a full architecture for the ZedBoard making use of *Xilinx DMA* cores and *AXI memory interconnects* as represented in Figure 3. As we can see the inputs and outputs of the KMP module in the figure are connected only to the DMA, through the AXI interconnect the ARM processor sends command to the DMA, and the DMA can transfer data to and from the memory. It is then the DMA to feed data in and fetch data from the HW core we realized internally to the KMP module. The entire implemented design whithin the Vivado layout can be observed in Figure 4.

IV. Results

In this section we compare the system presented in this work with the results published in [3] improving the performance measurements obtained in that work.

The full system occupation of the ZedBoard's area after the *place and route* phase can be found in Figure 5. The *red*

TABLE I. RESOURCE UTILIZATION BREAKDOWN ACROSS DIFFERENT RESOURCES FOR OUR IMPLEMENTATION COMPARED TO [3]

	BRAM 18K	Flip Flop	LUTs
[3]	152 (54%)	19000 (16%)	15467 (26%)
This work	280 (100%)	22000 (20.68%)	20680 (38.87%)

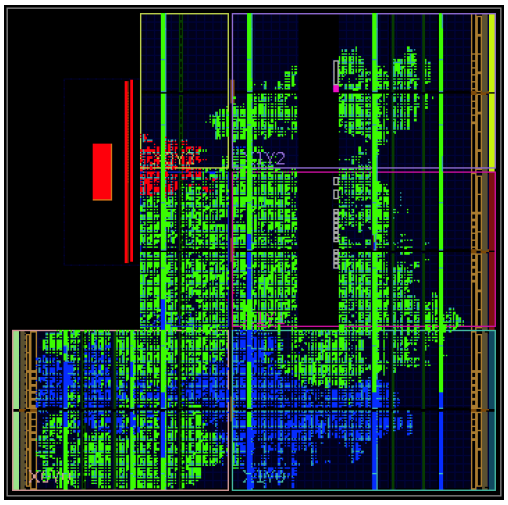

Fig. 5. Layout of the system highlighting different components. ARM processor and interconnection in red, a single KMP module in blue, remaining KMP modules in green.

area represents the ARM processor, the *blue* one represents one single HW accelerator's instance while the *green* ones are related to the other three cores. In the picture the columns in the device spanning all the height fully utilized are the BRAM, which are fully utilized in our design in order to fit the most of the database in HW at one time. Table I describes the resource usage of the programmable logic of the core presented in this work comparing to the one presented in [3]. It can be noticed that we saturated the BRAMs resources, in order to host the maximum number of proteins of the database in one single execution.

We then compared the performance in terms of execution time for a match of a reference protein fragment with a reference human database. The comparison has been made between the following 4 systems (they all implement the Knuth Morris Pratt (KMP) algorithm):

- *i7*: 2,3 GHz, quad core, Intel Core i7–4850HQ processor (single thread);
- *ARM*: Dual ARM® Cortex™–A9 processor (available on ZedBoard device);
- *BioCAS*: our previous implementation of KMP algorithm published at BioCAS [3]. It represents a 4 cores implementation with a different data exchange system;
- *Proposed*: the HW accelerated version of our system, with 4 HW accelerators, as described in the previous section.

The ZedBoard SW has been written in C++ and the board boots with PetaLinux exploiting custom DMA drivers and user space libraries to allow a general user to control the HW

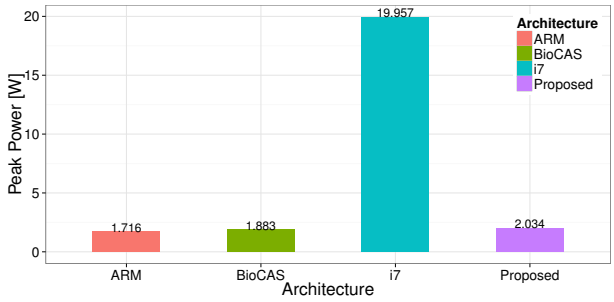

Fig. 6. Maximum power consumption of the systems under comparison.

cores and the full system. The usage of the custom drivers (available here [18]) allows to hide the complexity of the accelerator and data transfer management from the final user, which does not need to be involved in the design process. The targeted board Zynq AVNET Zedboard provides an embedded ARM® processor (Dual ARM® Cortex™–A9 MPCore™), 512 MB DDR3 memory and a 4 GB SD slot. Every HW core has been developed thanks to the Xilinx Vivado toolchain version $v2015.1$, while all the software related to the proposed version and the BioCAS implementation has been written in C++ and compiled with the Xilinx g++ compiler (arm-xilinx-linux-gnueabi-g++) together with our aforementioned support libraries. The software version on the i7 processor of the KMP algorithm has been written in C++ (the same as ARM processor) and compiled with a g++ compiler (version C++11).

The time measurements have been collected over multiple runs using Linux *gettimeofday* function, while the power consumption has been collected with Intel Power Gadget utility which reads HW energy counters, or from post-implementation power estimation for the ARM and HW accelerated versions. Please note that [3] reports the power consumption of the i7 as 37 Watts since the Thermal Design Power (TDP) of the processor has been used in that work instead of performing a runtime measurement, here we used the power measured at runtime as reference instead.

Figure 6 reports the power needed to perform the string matching of the reference protein against the reference database. Regarding the i7 implementation we collected a trace with the Intel Power Gadget and computed a mean value, while the values of the others systems have been obtained thanks to the Vivado tool estimations. As we can see the power of the proposed solution is about $10\times$ smaller than the one used by the i7 and just a little higher than the solution proposed in BioCAS and the sole ARM. Figure 7 reports the average execution time needed for the experiment. In this case the i7 implementation is clearly the best solution, however the proposed architecture is able to achieve a $1.76\times$ speedup with respect to [3], while consuming only a small amount of power more. Finally we want to analyze the energy needed to perform the experiment which shows the benefit of the proposed solution. The energy has been computed by multiplying the average execution time by the power consumption and the result is shown in Figure 8. As we can see from the figure the proposed solution is clearly better than the i7 one and also improves on [3]. In particular it needs $1.63\times$ less energy than [3] and $5.75\times$ less energy by the i7.

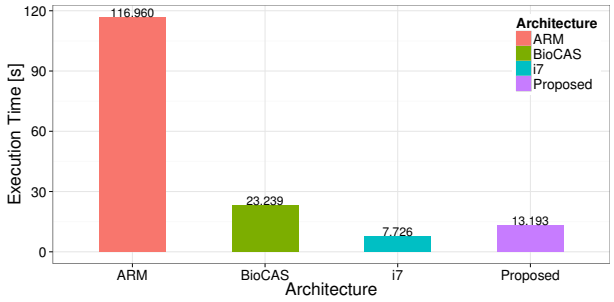

Fig. 7. Execution times of the systems under comparison.

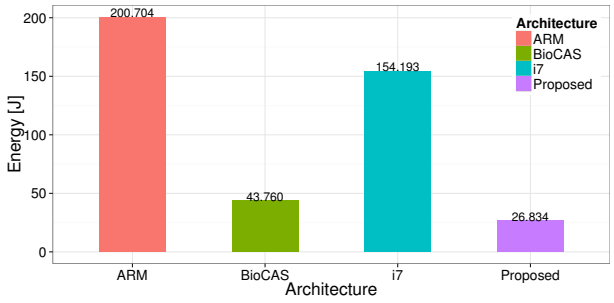

Fig. 8. Energy dissipated by the systems under comparison.

V. Conclusion

In this paper we presented a HW architecture to perform string matching in the context of the problem of protein identification. We designed a SW/HW implementation by leveraging on the Vivado Design Suite and targeting a AVNET ZedBoard. The solution proposed in this paper makes use of local memory to cache part of the database to perform the match against in order to minimize the amount of data moved across the HW/SW boundary. The obtained system has been compared with a previous work proposing another implementation of the same algorithm [3], and with SW implementations on both the ZedBoard ARM processor and on a high-end workstation featuring an Intel i7 core. The results show a reduction of $1.63\times$ and $5.74\times$ of the energy consumption over the literature and the high-end workstation respectively.

Whithin the serum proteomics field, the growing number of biomarker analysis requests could bring to a huge energy requirement. This requirement could greatly impact the scalability of large scale installation and their cost of mantainance. Our work could lay the basis for an instrument which should allow a bigger number of exams computed in parallel and reducing the energy consumption needed.

Future works will investigate the possibility to optimize the core caching, not only the input database, but also the results which are now read one at the time from the HW and are the most limiting factor to achieve greater speedups. Furthermore we also would like to investigate how integrating this core in a high-end FPGA connected to PCIe to a host processor would compare in terms of energy efficiency against the embedded solution proposed here.

References

[1] W. Liu, Q. Yang, B. Liu, and Z. Zhu, "Serum proteomics for gastric cancer," *Clinica Chimica Acta*, vol. 431, pp. 179–184, 2014.

[2] [Online]. Available: https://bitbucket.org/necst/proteinidentification-paper

[3] G. Bianchi, F. Casasopra, G. C. Durelli, and M. D. Santambrogio, "A hardware approach to protein identification," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*. IEEE, 2015, pp. 1–4.

[4] M. Calik, E. Demirci, E. Altun, . Calik, O. B. Gundo?du, N. Gursan, B. Gundo?du, and M. Albayrak, "Clinicopathological importance of Ki-67, p27, and p53 expression in gastric cancer," *Turk J Med Sci*, vol. 45, no. 1, pp. 118–128, 2015.

[5] T. U. Consortium, "Uniprot: a hub for protein information," *Nucleic Acids Research*, vol. 43, no. D1, pp. D204–D212, 2015.

[6] [Online]. Available: http://www.uniprot.org/UniProt

[7] P. Artimo, M. Jonnalagedda, K. Arnold, D. Baratin, G. Csardi, E. de Castro, S. Duvaud, V. Flegel, A. Fortier, E. Gasteiger, A. Grosdidier, C. Hernandez, V. Ioannidis, D. Kuznetsov, R. Liechti, S. Moretti, K. Mostaguir, N. Redaschi, G. Rossier, I. Xenarios, and H. Stockinger, "Expasy: Sib bioinformatics resource portal," *Nucleic Acids Research*, vol. 40, no. W1, pp. W597–W603, 2012.

[8] [Online]. Available: http://web.expasy.org

[9] I. Bonesana, M. Paolieri, and M. Santambrogio, "An adaptable fpga-based system for regular expression matching," in *Design, Automation and Test in Europe, 2008. DATE '08*, March 2008, pp. 1262–1267.

[10] R. J. Peace, "String matching and online retention time prediction for real-time information-driven mass spectrometry," Ph.D. dissertation, Carleton University Ottawa, 2011.

[11] R. D. Cameron, K. S. Herdy, and D. Lin, "High performance xml parsing using parallel bit stream technology," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*. ACM, 2008, p. 17.

[12] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987.

[13] R. S. Boyer and J. S. Moore, "Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic," in *Machine intelligence*. Citeseer, 1985.

[14] S. M. Vidanagamachchi, S. D. Dewasurendra, and R. G. Ragel, "Hardware software co-design of the aho-corasick algorithm: Scalable for protein identification?" *CoRR*, vol. abs/1403.1317, 2014.

[15] S. Vidanagamachchi, S. Dewasurendra, R. Ragel, and M. Niranjan, "Tile optimization for area in fpga based hardware acceleration of peptide identification," in *Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on*, Aug 2011, pp. 140–145.

[16] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[17] T. Feist, "Vivado design suite," *White Paper*, 2012.

[18] https://github.com/durellinux/ZedBoard_Linux_DMA_driver.

[19] R. J. Peace, H. A. Mahmoud, and J. R. Green, "Exact string matching for ms/ms protein identification using the cell broadband engine," *Journal of Medical and Biological Engineering*, vol. 31, no. 2, pp. 99–104, 2011.

[20] Y. S. Dandass, S. C. Burgess, M. Lawrence, and S. M. Bridges, "Accelerating string set matching in fpga hardware for bioinformatics research," *BMC bioinformatics*, vol. 9, no. 1, p. 197, 2008.

[21] D. E. Knuth, J. Morris, and V. R. Pratt, "Fast pattern matching in strings," *SIAM Journal of Computing*, vol. 6, no. 2, pp. 323–350, 1977.