A Hardware Approach to Protein Identification

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

***Abstract*—At the basis of proteins identification we have a string matching algorithm, which has a computational complexity that scales with the length of both the searched and the reference string. This complexity, as well as the fact that to match a single protein we need multiple search of different string in the whole database, makes the protein identification a computational intensive task taking tens of seconds to complete. When performing this task with General Purpose Processors (GPPs), as it might be in a large scale installation (such as medical or research centers), this long execution time translates into a high energy requirement which greatly impacts the scalability and maintenance cost of the system. This paper illustrates a possible way to exploit Field Programmable Gate Arrays (FPGAs) to implement a string matching algorithm with an higher energy efficiency, up to 6 times better, than a standard GPP; such solution can be a building block for large-scale installations aimed at improving protein identification.**

I. Introduction

The ability to quickly identify proteins is a major concern in many medical applications such as cancer monitoring and recognition and pharmaceutical research. Due to the fast increment of data available thanks to technological improvements, we need to take proteins identification a step further, improving its identification speed to match the growth of both proteomics and genomics databases. Because of the fast data growth inside both proteomic and genomic databases, protein identification requires an increasing amount of computation. Many researches will benefit from accurate protein identification and the ability to produce more accurate results at faster rates. For instance, disciplines, such as detecting biomarkers within diagnostic field, in order to recognize and monitor cancer disease with serum proteomic [1], bacterial identification, and pharmaceutical research will take advantage of this possibility.

The problem of identifying a protein is similar to the string matching problem, i.e. the problem of finding a substring in another string; in particular in this case the problem consists in matching a string identifying a peptide of an unknown protein, against a string identifying a whole well known protein. After all the peptides of the unknown protein have been searched a score can be assigned to the proteins in the database to find the one that best matches the unknown protein we are trying to identify. Unfortunately the computational complexity of this problem grows as $L_{pep} \times L_{pro}$ if we identify the length of the peptide as L_{pep} and the length of the protein as L_{pro}. Furthermore as we have to search for all the unknown peptides forming the protein, U in the whole database composed of P proteins we have a worst case computational complexity of

$$U \times L_{pre} \times P \times L_{pro}$$

This complexity refers to a single protein identification, and obviously another multiplication factor has to be added to the formula to take in consideration the scenario of multiple proteins identification.

Because of this computational complexity the problem might take long time to be solved for large databases. In this paper we propose an implementation of the string matching algorithm targeting a Field Programmable Gate Array (FPGA) device, and we will show how reconfigurable hardware technology can fit in the proteins identification problem both at small- and large-scale. In our work[1] we focused on the implementation of the well-known *Knuth Morris Pratt (KMP)* algorithms and we targeted real world data targeting the recognition of the Ki-67 protein, which is described as one of the biomarkers currently used to monitor gastric cancer [3], against the isoform proteomic FASTA database, provided by Universal Protein Resource [4].

The remainder of the paper analyzes the current state of the art (Section II), describes 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

Hardware accelerators have been used with benefit in different applications, including in the pattern matching scenario, tackled by this work. As an example an implementation of three different Field Programmable Gate Arrays (FPGAs) for the *regular expression matching* problem has been proposed in [5]; this solution was able to obtain improvements over the SW implementation in all the tested scenarios. The work in [6] developed a custom algorithm, named *Orthogonal Parabix*, adapting the *Parabix* algorithm to its specific needs. This work compares the *Orthogonal Parabix* to other approaches available in literature (Rabin Karp [7], Boyer Moore [8], Parabix [9]) and illustrates how such algorithm can achieve faster results, enough for real time protein identification, for their specific problem. Other researches over the last few years show how, in the proteins identification problem, the usage of FPGA devices leads to good improvements. For instance, in [10] the authors illustrated how a HW/SW codesign approach can help by allowing both the flexibility of the SW as well as the speed of the HW architecture. Another work [11] proposes an automatic implementation of an *exact string matching* algorithm, i.e. *Aho Corasick* [12], that optimizes the string length of the peptides used in the searching process. Finally

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

[13] realized a HW implementation of *Aho Corasick* that aims at optimizing the utilized area on the device, while maintaining good performance, by partitioning the Finite State Machine (FSM) that performs the string matching analyzing a small number of peptides in parallel. Although it has been demonstrated how *Aho Corasick* is the fastest string matching algorithm, it suffers from a great limitation for what concern HW implementation. It relies on the creation of an ad-hoc HW component to match a single peptide, which causes the need to synthesize different HW components for each possible peptide. Given the huge time needed to synthesize an HW component (few hours when the design is not too complex) this approach does not seem the best solution in term of flexibility. In this paper we instead focus on more general string matching algorithms and we show how they can benefit from a FPGA implementation. The algorithms we take into consideration can be used for all kinds of proteins and peptides we want to compare, allowing to use the solution even for future updates of the proteomic database and for new sets of peptides.

III. Implementation

As stated above, the computational complexity of string matching grows with the lenght of the strings involved (i.e. the peptide and the protein), in a different way depending on the selected algorithm: from the product between the strings in a naive approach, down to their sum for the Knuth Morris Pratt (KMP) [14] algorithm. This algorithm puts sentinel values at the end of both the peptide and the protein under analysis and, thanks to a preprocessing of the peptide, is able to construct a map that allows it to jump to the next candidate match string when a previous match failed. Faster approaches are available, as cited in Section II, such as the Aho Corasick algorithm based on building trie for the peptides to match. However this approach is not suitable for Field Programmable Gate Array (FPGA) device since it needs to go through the long synthesis, map, and place and route steps required to derive the description of the circuit in form of a bitstream capable of configuring the device. For this reason we focused on the KMP algorithm which can be used to check any peptide the user desires. The rest of the description will focus on the HW core and its integration with the entire system, illustrating the key points of the design.

The solution we propose is an HW/SW combined approach to the string matching problem exploiting the KMP algorithm. The HW core uses local BRAM for effective computation and the overall string matching problem against a proteomic database is split in smaller instances. In the current implementation we analyzed the possibility to perform the comparison between a single protein and a peptide. The HW will then receive the input data one at the time and will then find all the position at which the peptide is contained in the protein (more than one if it is the case) and then it sends the results back to the host processor. The host processor is then responsible of merging the obtained data and compute scores to find out which is the most suitable match in the database (note that all the functions we left in SW have a linear complexity and do not greatly impact on the overall execution time).

The HW architecture we target, as final device, is an ARM based chip using an AXI/AMBA as communication bus that transfers 32bits at a time. This detail, although it is a low level detail decoupled from core functionality, it is important to understand an important implementation detail. An important thing of our design is the fact that we exploit the possibility to send 32bit at the time to a HW device to copy all the input strings without the need of adapting the input before sending the data. Instead of sending everything as an array of characters (8 bits) we use the whole bus-width sending 4 characters at the time, which are then unpacked by the HW core in an array of characters representing the input strings. This solution allows us to use the whole bus bandwidth during transmission and to not lose any time to convert data in the format wanted by the HW accelerator. Considering in fact that the KMP method complexity increases almost linearly with the size of the strings, the process of reshaping data (which is generally linear itself) will not permit the HW core to reach any speedup with respect to the SW solution, since reshaping data will take as much time as computation.

The computational part of the core itself adhere with the original KMP algorithm and it has been implemented using the Vivado High Level System (HLS) tool [15]. We pose particular attention during the design of the core on how the data is exchanged between the host and the HW core. At first we implemented the core such that it uses *AXI stream* interfaces that allows to stream data into the HW core, and to connect both input and output to a DMA component to move data from and to host memory. This solution is by far the most efficient since moving one character at the time writing the value on the bus will incur in a massive overhead. At the same time we optimized the data movement treating the array of characters representing the proteins and peptides to be managed and transferred as an array of 32 bits data. In this way each transfer over the bus, which is 32 bits wide, utilizes the whole data path and we can use less transfers to move all the data. Starting from this, we need to split back data in the HW core upon the reception of 32 bits data to obtain 4 different 8 bits value representing the characters that has to be used for the string matching. Finally we implemented two versions of the algorithm that interact with the host processor in two different ways. The first solution acts as the standard KMP algorithm for a General Purpose Processor (GPP) processor: in this case at each comparison between a protein and a peptide, both of them are sent to the accelerator. The second solution instead tries to reduce the amount of data transfer caching the protein on core local BRAM for whole time it is needed. This means that with this second solution the protein, which is the longer string, is sent to the HW core only upon the comparison with the first peptide and then it is retained in local BRAMs. For the comparison with all the other peptides no information about the protein is send, and only peptides are moved from the host processor to the HW core.

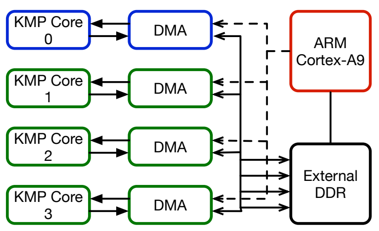

Fig. 1. Complete system layout, realized with Vivado [15]

After realizing the HW core by mean of HLS, we integrated it in our target embedded system in order to enhance the processing capability of the platform. The reference target platform for this work is the AVNET Zedboard development board, which features an ARM dual core processor and programmable logic. We realized a system that exploits multiple copies of HW cores to process different proteins and peptides in parallel. In particular our device provides 4 *high performance memory access ports* that a Direct Memory Access (DMA) component can use to exchange data in an efficient way with the programmable logic and our HW cores. Figure 1 reports the high level view of the system we designed. This hardware design allows to optimize both the HW core, customizing it to the computation that has to be done by means of HLS tools, and the overall system allowing to distribute the computation across different units to speedup the execution.

IV. Results

The HW core has been implemented using the Xilinx toolchain and we used the ZedBoard as a target device. The HW accelerator uses only stream interfaces as input and output meaning that it computes upon the reception of a given amount of data. The SW processor can offload computation to the HW core by issuing data transfers controlling the Direct Memory Access (DMA) connected to the core. Through the Vivado HLS tool we converted the initial C code, adapted to the C-like description supported by the tool itself, into VHDL by means of an High Level System (HLS) process. We focused on the optimization at data transfer level and no finer tuning in the algorithm itself had been explored. The core so design has been integrated into a full embedded system comprised of an ARM Cortex-A9 dual core processor following the structure depicted in Figure 1, where 4 copies of our core (that can work in parallel) have been integrated. Figure 2 illustrates how the system we designed has been mapped on the target device. The red blocks are the one related to the ARM processor and the AXI/AMBA bus; the blue cells represent one instance of the HW accelerator for string matching; while the green part highlights the other three accelerators. For a more straightforward illustration of resource usage, Table I illustrates the resource usage across different resources for both the HW core itself and the full system.

The ZedBoard platform can run Linux on top of the ARM cores, and we compiled a kernel starting from the one released from Xilinx adding the DMA drivers we developed. On top of these drivers further user space libraries can be devised to allow a general user to control the HW cores. These libraries might take as input the proteins and peptides to compare, manages the transmission of data to the core, and collects the results. All the complexity of the HW core management and data transfer management is then hidden from the final user which does not have to be involved with any stage of the HW design process. The ability to run Linux, and the support for C++, allowed us to run develop the application on our laptops for debug and tests and then simply cross-compile the same application for the ARM processor. In this way we are sure that a fair comparison can be drawn since the code executed is exactly the same.

We tested our solution using, as input for identification, the results obtained from the virtual digestion of a protein (Ki-67)

TABLE I. Resource utilization breakdown across different resources for our implementation. Utilization for both single core (after the synthesis) and the whole system (after place and route) are reported.

	BRAM 18K	Flip Flop	LUTs
HW accelerator	34 (12%)	987 (1%)	1958 (3%)
Full System	152 (54%)	19000 (18%)	15467 (29%)

obtained with the tool ExPASy [16], simulating the digestive process by means of the *tripsina* enzyme (one of the most common enzymes used for the Tandem mass spectrometry). The system is tested with a real case scenario trying to identify a unknown protein (i.e. set of peptides) obtained through the previous process against a reference database of human proteins that stores the information of 62573 proteins. We perform the test on 4 different systems:

- *i7*: a 2,3 GHz Intel Core i7 processor;
- *ARM*: an ARM Cortex A9 processor (the one mounted on the Zedboard);
- *HW*: the first implementation of our solution, which needs to be provided each time with one protein and one peptide;
- *HW2*: the second version that receives the protein one time for every unknown peptides.

Figure 3 illustrates the results obtained for the execution of the Knuth Morris Pratt (KMP) algorithm in these four scenarios. The plot shows the performance in terms of number of proteins in the database analyzed per second, and it shows that if we focus only on performance the *i7* implementation is the best solution when compared to any of the implementation we carried out on the Zedbord. Focusing on the Zedboard we can see how *HW2* can obtain faster results than *HW* especially for shorter proteins; while both of the HW solutions perform better than the *ARM* implementation.

We then compared the energy efficiency of the solution we devised using reference Thermal Design Power (TDP) for the *i7* taken from Intel specification, which is 37 W, and values obtained using Vivado power report tool for the Zedboard (we executed the tool with default parameters). The power report

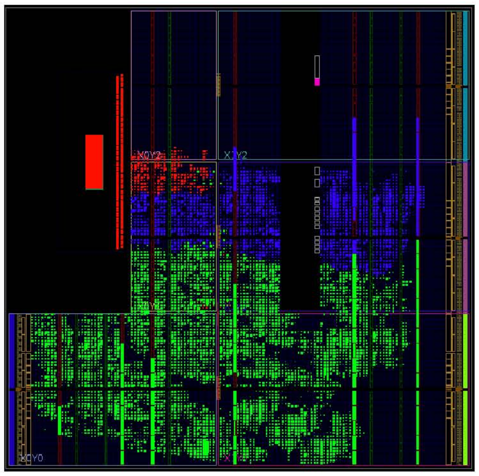

Fig. 2. FPGA resource mapping after *place and route* phase for the ZedBoard.

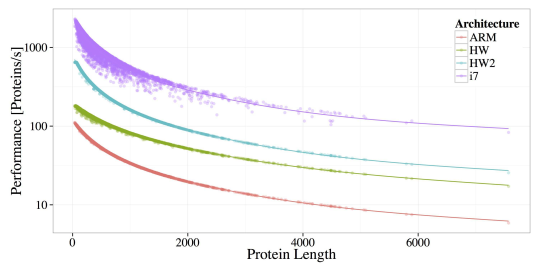

Fig. 3. Performance of the different KMP implementations

on Vivado assigned a total power (static and dynamic) of 1.883 W for the entire board, where 1.716 are due to the ARM subsystem and device static power. For *HW* and *HW2* we used 1.883 W as power consumption value, while we used 1.716 W for the *ARM* solution. A plot of the energy efficiency of the different solutions (measured as proteins analyzed per joule) is reported in Figure 5. As we can see in this case the results are flipped and *i7*, even if is the fastest solution, it is not the most energy efficient. If we look at the figure we see that all the implementations on the Zedboard have an energy efficiency which is higher than the *i7* one for every possible protein length with an energy efficiency gain ranging from 5x to 10x. This increase in energy efficiency ends up in the possibility to perform a comparison between the unknown reference protein and the whole database consuming less energy. Figure 4 reports the total energy needed for a comparison of the unknown protein with the whole database, estimated as the total time needed multiplied by the power consumption of each solution. As the figure shows the most efficient of our solutions (*HW2*) can perform a full scan over the database consuming less then 6.27 times less energy than the *i7*.

V. Conclusion

In this work we presented a hardware implementation for the protein matching problem based on the Knuth Morris Pratt (KMP) algorithm targeting the Zedboard development board. The HW implementation has been focused at optimizing the data transfer overhead between the host processor and the reconfigurable logic. Tests using the human proteomic database [4] has been executed on both the Zedboard and an Intel processor and performance and energy efficiency results have been collected. The analysis of such results shows how the devised solution can improve the increase the energy efficiency of a 6.27 factor with respect to the *i7* implementation, making it a viable solution for larger scale systems.

Future works will investigate the possibility to implement a larger scale solution starting from either a cluster of lower end Field Programmable Gate Arrays (FPGAs), such as the Zedboard we used, or a cluster of high end boards, such as the Virtex 7, connected to PCI express to standard desktop and server processors as the *i7* we used as comparison in this work.

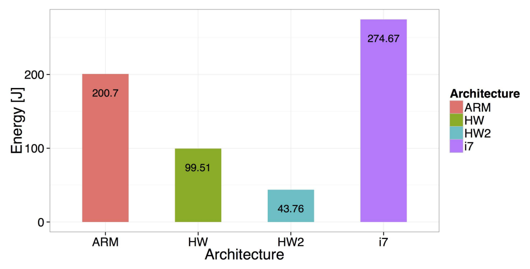

Fig. 4. Total energy needed matching one unknown protein against all the database for the different KMP implementations

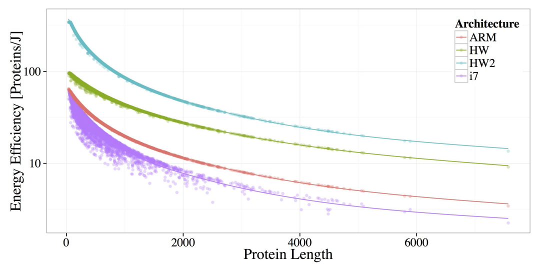

Fig. 5. Energy efficiency of the different KMP implementations

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] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.