High-Performance FPGA-Based BWA-MEM Accelerator

Binh Kieu-Do-Nguyen, Cuong Pham-Quoc, and Cong-Kha Pham

***Abstract*—There is no denying that Bioinformatics is one of the most important realms for our forthcoming development. As a demonstration of this fact, a plethora of new algorithms that were published over the last decade. Those significantly boost up the processes of biological analysis, especially for DNA alignment. Despite their undeniable contributions, it is still far more to state that DNA alignment has already achieved the ideal performance. In this work, we focus on the DNA alignment system which is based on our improved BWA-MEM algorithm that we have already published. Besides that, we also propose some optimization methods which was applied in order to improve the performance as well as the stability of our entire system. The system offers a speed-up by 46.52x when compared with the other computing platforms.**

***Index Terms*—DNA alignment, BWA-MEM algorithm, FPGA, IP Core Seed extension.**

I. Introduction

Our Computer Age has been marked by the endless development in both computing architectures and techniques. DNA Sequencing also takes grand advantages from the new computational inventions, especially from Nest Generation Sequencing (NGS) techniques. NGS techniques offers a dream low-cost genome sequencing and also stimulate the generation of biological data. As a consequence, the amount of biological data, especially genetic data, will soon become one of the enormous set of Big Data field. In addition, NGS data usually are processed by very complex algorithms that require a number of computational operations. This issue has been being a formidable obstacle that put a high pressure on the traditional computing platforms. In this situation, heterogeneous computing platforms are today highly considered on over the world.

DNA sequences are composed of four kinds of nucleobase, including C (cytosine), G (guanine), A (adenine), and T (thymine). These nucleobases are read by biological techniques and are combined into the short strings (especially with 150 typical length [1]) afterwards. These strings then are aligned against an enormous reference genome (even reaches nearly 3 billion bases for the human genome). Such kind of activities are long-haul iterations and time consuming, easily up to multiple days for mapping jobs. Despite its vast quantity, these jobs, in detail, can be classified into 3 main kinds of operation. This is: (1) addition, (2) subtraction, (3) comparison. Many sequence alignment tools were published, such as Bowtie [2], BWA [3], MAQ [4], SOAP2 [5] and BWA-MEM [6]. Among these tools, BWA-MEM is highly recommended for high-quality queries as it is faster and more accurate [7].

FPGA (Field-Programmable Gate array) is a heterogeneous computing platform, which has been providing a flexible environment for system development. The main feature of FPGAs is application specific. As they can be programmed in order to be integrated with a tremendous number of computing elements such as adders, subtractors, RAMs, etc. In addition, the number of these components as well as the ratio among them can be adjusted for the highest appropriateness of a sole application. The use of FPGAs can yield impressive improvements in both processing frequency and power consumption.

In this paper, we focus on the deployment of our FPGA-based Seed Extension IP Core, which was implemented in our earlier works in [8] and [9], in a CPU-Hardware acceleration system. The main contributions of this paper can be summarized as follows:

1) We analyze the steps of our improved BWA-MEM algorithm
2) Then, we optimized the fragments of this algorithm in order to achieve a highest performance and flexibility.
3) Finally, we advocate a typical CPU-Hardware acceleration model in order to take the advantage from our developed IP Core.

Finally, we advocate a typical CPU-Hardware acceleration model in order to take the advantage from our developed IP Core.

II. Background and Related Work

A. Background

BWA [3] is a read alignment package that is based on BWT (Burrow-Wheeler Transform) to align short sequencing reads against a large reference sequence such as the human genome. BWA and its variants are widely using in both academic and commercial environment thank to their efficiency. BWA consists three descendants, they are: (1) BWA-backtrack, (2) BWA-SW and (3) BWA-MEM where BWA-MEM is the latest with the highest speed and accuracy. In general, the BWA-MEM alignment algorithm involves three main stages:

1) **SMEM generation**: standing for Super-Maximal Exact Matches. In this stage, DNA's data, which were collected and stored beforehand, are read. With each read string, maximal matched positions are located. These spots are called as seeds. A sole seed cannot be extended in both direction and each seed cannot be covered by one in

Manuscript received November 15, 2019; revised October 19, 2020.

Binh Kieu-Do-Nguyen and Cuong Pham-Quoc are with the Ho Chi Minh City University of Technology – VNU-HCM, Ho Chi Minh City, Viet Nam (corresponding author: Cuong Pham-Quoc; e-mail: 1770283@hcmut.edu.vn, cuongpham@hcmut.edu.vn).

Cong-Kha Pham is with the Department of Computer and Network Engineering, Cluster II (Emerging Multi-interdisciplinary Engineering), The University of Electro-Communications, the city of Chofu, Tokyo, Japan (e-mail: phamck@uec.ac.jp).

another. None or more seeds can be produced per read.

2) **Seed extension**: in this stage, the seeds that are already processed in an alignment before are dropped. The others, if they potentially lead to a new alignment, are extended with a banded affine-gap-penalty dynamic programming (DP). After scoring processes, a maximal similarity chain, but not identical, is identified. The scoring process method that is similar to the Smith-Waterman algorithm [10].

3) **Output generation**: among the alignments which were scored, the best one is selected and is stored into memory. The final selection can be performed a global alignment if it is necessary.

Fig. 1 shows BWA-MEM algorithm step-by-step. SMEM generation (including 'read data' and 'seeding') and Output generation (including 'write score') are operated in host computer (i.e., CPU or any other general-purpose processors). Seed extension is partly run on FPGA, which plays a role as acceleration hardware. The process of extending seeds is the most appropriate part in Seed extension stage that can be easily, and also most efficiently, deployed onto heterogeneous computing platform [11].

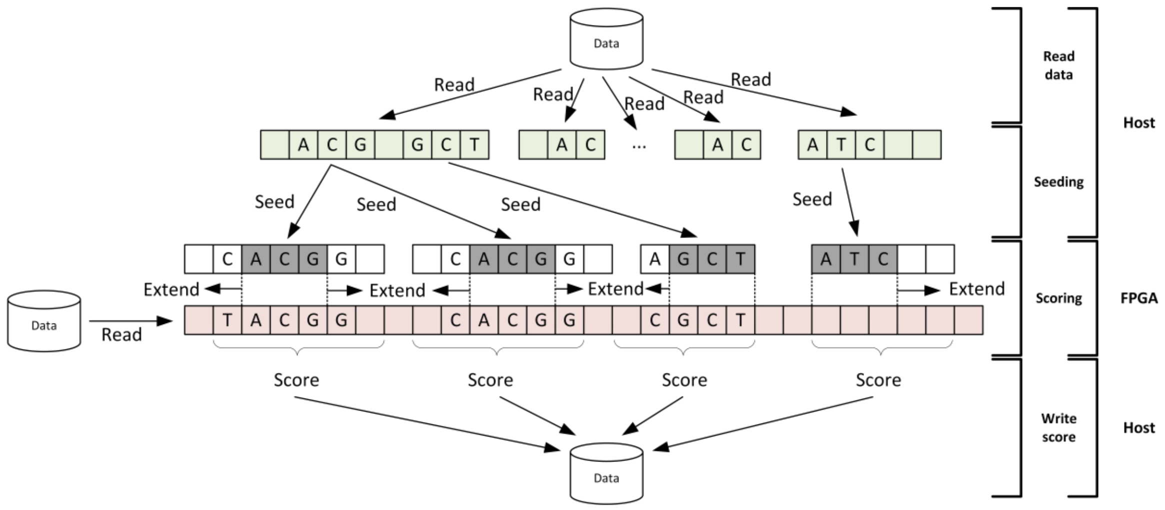


Fig. 1. Stages of BWA-MEM algorithm and the place where each stage is executed.

The exciting BWA-MEM algorithm [11] works well with multithreading programs on general purpose platform (i.e., CPU, GPU) but not on FPGA. Unfortunately, due to a high data-dependency, the original BWA-MEM algorithm does not aim to be easily, and efficiently, implemented into an application-specific platform, especially in FPGA, that can lead to a significant boost [12]. In order to resolve this issue, we promoted a slight rearrangement of the original BWA-MEM algorithm to achieve a well fit with FPGA [9]. Our implementation can work with a higher frequency than one in the previous investigations. Despite having a high performance, the IP Core, based on our improved algorithm, has still needed an appropriate protocol that can handle exchanges of data. In addition, the enhanced algorithm can be more optimized to achieve a better performance.

B. Related Work

There are a number of published proposals in the literature that accelerate the DNA alignment algorithm. However, according to our knowledge, there are a minority of them that tried with heterogeneous computing platforms. The authors in [13] firstly suggested the idea where the Seed extension stage is conducted by using an array of processing elements. Their contribution is widely inherited in the later investigations. The authors also reported an up to 26x speed-up achieved for the kernel. Authors in [14] used a systolic array model in order to accelerate Seed extension stage of BWA-MEM algorithm. They promoted the design in [13] by their Variable Logical Length and Variable Physical Length models that help their design to cope with the varied-length string. Their reports reveal that their Seed Extension kernel achieved 2.82x and their next improvement reached 5.7x speed up. All of of these implementations only concentrate on finding the ways by which a sequence alignment algorithm can be implemented into FPGA, despite the fact that they are not designed to work well on such platform. Therefore, the performance of these did not reach a level that it should attain.

III. Optimization

A. Optimized BWA-MEM Algorithm

Algorithm 1 is our improved BWA-MEM algorithm. We already broke off the origin in order to ameliorate the parallelism. Operations in loop are divided into 3 states, each state can be conducted by a specific hardware-block, and each block can be activated simultaneously. In addition, operations inside the loop are divided not only by order but also by its functions.

Algorithm 1: Optimized BWA-MEM scoring algorithm

```
input: Query string Q, length of query string |Q|, Target
       string T, length of target string |T|, initialscores
output: Scored matrix m
1 Allocate scoring matrix m: array [1..|Q|, 1..|T|];
2 Allocate scoring matrix m: array [1..|Q|, 1..|T|];
3 Allocate H: array [1..|Q|, 1..2];
4 ZerosFill(&m);
5 ZerosFill(&E);
6 ZerosFill(&E);

7 for row ← 1 to |T| do
8   | Allocate substitution array P;
```

```
9   Allocate and initialize $F$: array [1..2];
10  Initialize(&$Q$, initialscores);
11  ZerosFill(&$F$);
12  Assign(&stage, preparing);
13  for col ← 1 to |$Q$|
14      if stage is preparing then
15          Select(&candH, H) ;
16          Select(&candE, E, 0);
17          Select(&candF, F, 0);
18          Assign(&stage, scoring);
19      end
20      if stage is scoring then
21          Calculate(&H[1], candH);
22          Calculate(&H[2], candE);
23          Calculate(&H[3], candF);
24          Calculate(&E[1], candH);
25          Calculate(&E[2], candE);
26          Calculate(&F[1], candH);
27          Calculate(&F[2], candF);
28          Assign(&stage, writeback);
29      end
30      if stage is writeback then
31          Update(&m[row][col], candH, candE, candF);
32          Assign(&stage, preparing);
33      end
34  end
35 end
```

B. Performance Optimization

In Algorithm 1, by breaking off the original algorithm into 3 stages with less data-dependency, our deployment into FPGA platform easily achieved a higher frequency than previous implementations. In order to further increase the performance of system, we applied the following enhancements:

1) We organized each stage in Algorithm 1 as a 3-stages pipeline model. As it allows more data to be processed at a same time, the throughput of the whole system is nearly tripled.
2) In Algorithm 1, we realized that the inner loop (from line 13) ought to be executed in parallel. But, because there are data-dependency among different loops, a system-level pipeline mechanism should be applied. Fig. 2 depicts the abstract behaviors of the system when the concurrency and pipeline are both indicated. When the inner loop is declared to be simultaneously processed, $|Q|$ Processing Units (PUs), where $|Q|$ is the number of characters in query string, are generated. Each PU conducts an iteration of the loop. After finishing an iteration, a PU gets a new character in target string. A PU accomplishes its jobs only if the whole target string passes through it. For instance, at time T, PU[1] generates 3 array of data (involving H, E and F). These set of data will be forwarded for itself at $T + 1$, for its neighborhood at $T + 1$ and $T + 2$. By applying this optimization, we reduced the complexity of our adjusted algorithm from $|Q| * |T|$ to $|Q| + |T|$, where $|T|$ is the length of target string.

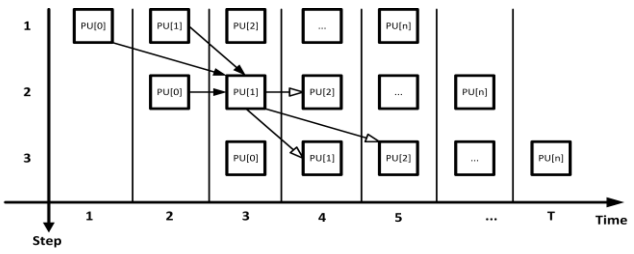


Fig. 2. Dataflow among Processing Units (PUs) over time.

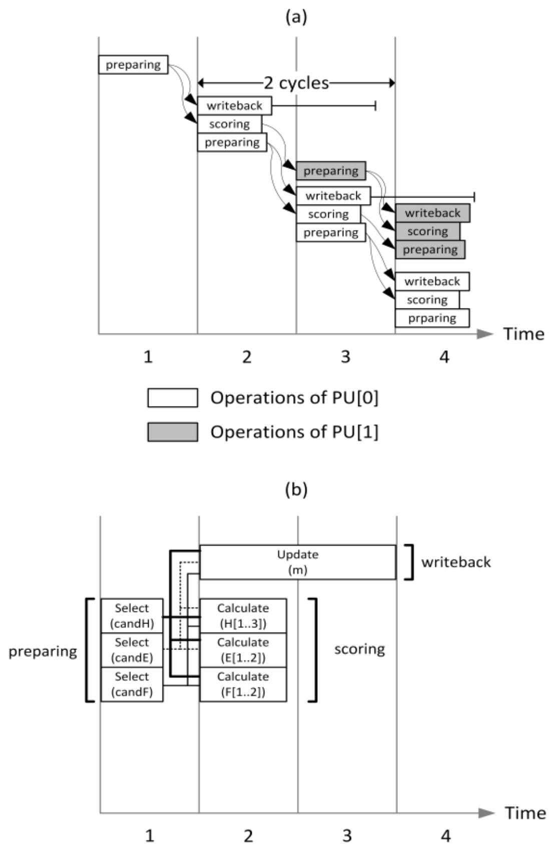


Fig. 3. Operation of BWA-MEM IP Core after optimization.

3) In Algorithm 1, the first goal that motivated us to rearrange the original algorithm is separating complex operations and grouping these fragments that can be executed in parallel. Although there are data-dependency among iterations of inner loop and among stages in each iteration, the statements inside each stage are completely independent. Thank to this feature, these operations in a

same stage can be concurrently processed. Fig. 3(b) shows the order by which operations in each stage are executed. By applying a high density of concurrency, the system just requires 3 period for each iteration.

Fig. 3(a) shows the behaviors of the system, assume that it has 2 PUs, when all the above optimization methods are applied. Firstly, input data are selected by PU[0]. Secondly, the selected data are stored into memory of PU[0] and are used to generate the data for PU[1]. At the same time, new input data also be selected by PU[0]. At the third period, the scored data from PU[0] are selected by PU[1] in order to update in score in the next. It is obviously that, from the third period, the number of PU that participates into the assembly line increase by 1 per period. It means that the density of data that are simultaneously processed, and also the throughput, increase rapidly over time.

IV. Proposed System

There is no doubt that the sequence alignment is a data-intensive application, the number and size of string in this domain can be considered as a big data problem. Input data and final score are There is no doubt that the sequence alignment is a data-intensive application, the number and size of string in this domain can be considered as a big data problem. Input data and final score are frequently transferred between host and FPGA. Moreover, the IP Core has to be adapted with different platform where it would be deployed. Therefore, on our work, we only provide a communication protocol instead of building a specific communication bus. This approach helps us to eliminate technology-dependency of our design, which means our IP Core can be deployed into different FPGA's families that come from various vendors.

An alignment process in our proposed system follows the below steps:

1) Data are loaded from database into CPU. A memory access enhancement mechanism, such as Direct memory access (DMA) or cache, can be added in order to improve the performance of transmission at this step.
2) CPU executes consecutive operations before the Seed extension stage. These operations do not take advantage from heterogenous devices because they cannot be simultaneously processed. CPU, with high frequency, show their strength with such kind of operations.
3) At the Seed extension step, CPU transfers data to the accelerator (implemented on FPGA) and actives a seed extension process. Fig. 4 shows the connections in detail between CPU and FPGA device.

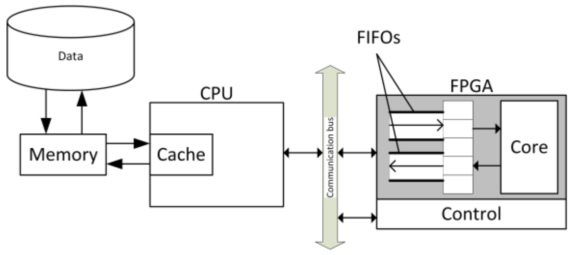


Fig. 4. Connections between GPP and BWA-MEM IP core.

We integrate 2 FIFOs/queues in order to ensure the synchronization between two different platforms, one for write data and the other for read results. The lined data in FIFO are stored consecutively into memory and are taken by the IP Core afterwards. On the other side, results that are wrote back to memory are read through FIFO by host computer. In order to optimize the communication between CPU and FPGA device, event-driven technique is applied into the system. Fig. 5 reveals the interactions between CPU and FPGA-based accelerator via event-driven mechanism. When the transmission of data is completed, host rises a request that will activate the aligning process in FPGA device. After that, while the IP Core is running, host prepares new data for next tasks. Whenever host receives an event's signal from device, it interrupts its current activities and then reads results from device. After that, host continues its interrupted works.

4) Whenever the host reads any value from device, it must read the valid flag of this data beforehand. The valid flag is a mechanism that ensures the soundness of data before they are read.

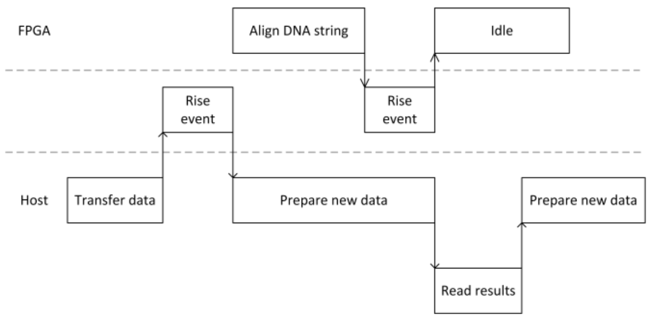


Fig. 5. Event-driven mechanism between Host and FPGA.

Our implemented communication handler provides a flexible interface that help our design can be easily deployed onto different FPGA's platform. By the helping of a general protocol, the implementation can work properly when it connects to host via any bus type. In addition, with kinds of bus that support an addressing mechanism, the system is even able to extend. Multiple acceleration's devices can connect with each other and with host in a single bus, and host will control them via their unique identification.

V. Experiments

A. Experimental Setup

We deploy and test our system in Zedboard. Zedboard is a development kit that is integrated with Zynq-7000 (xc7z020), a new generation FPGA chip from Xilinx. Our rival is i5-4440, a General-Purpose Processor from Intel. The processing time on FPGA is recorded and then is compared with this on i5-4440. The Table I shows our experimental system in detail.

TABLE I. Resources of Testing System

	GPP platform	FPGA platform
CPU	Intel i5-4440	Zynq-7000
Number of Cores	4	30
Frequency	3.30 GHx	200MHz
Storage	16GB DDR3	1GB DDR3

All performances are built based on an identical input data set, which is legally provided by NCBI [15] for testing. Our final goal is providing a Seed Extension IP Core. Therefore, the results only focus on the Seed extension stage, a whole system speed-up can be estimated based on the reports in [11].

B. Results

Tests are run with an identical input set. Input and output are totally stored on RAM. Because both platforms share a same RAM technology, we can avoid any potential discrimination of drive's access technology.

For each scenario, we run it multiple times and the average results are selected. In addition, different lengths of read of query string and target string are considered in order to identify the gap among them. In this contest, we measure execution time of strings with short read length (from 10 to 50 bp), medium read length (from 50 to 120 bp) and long read length (higher than 120 bp). The total size of target strings and query strings of our used dataset is about 135 MB.

Table II informs in detail about the execution time (in minutes) of Intel Core i5 Processor and our recommended system when it is deployed on Zedboard. In addition, Table II also shows the proportion of resources which are used to synthesize an IP Core that has 30 PUs. Recorded data involve the time that the system take to align all input strings of the dataset. The statistic includes time for execution and time for transmission. Despite Intel's Processor has a higher frequency but thanks to the higher throughput and high-level of parallelism, our system is always by far the best.

Fig. 6 illustrates the speed-ups of our proposed system with Intel Core i5. In our test, we assume that the speed of Intel Core i5 is 1, and then we calculate the speed-ups of our system. The chart clearly proves that our system provides a signification acceleration in the processing time. Results show that our system archives nearly 45.66x speed-up from Intel Core i5 with the long read's scenario. There is a fact that the grade of acceleration is not the same among different size of input strings. For instance, our system shows the higher performance with input strings that have a longer read length. On the contrary side, it shows a worse performance when the read length is small, the speed-ups are only 4.48x. We have already tried further experiments and conclude that the longer string is, the higher performance the system gets.

The reason that causes this phenomenon is the limitation of flexibility of hardware designs. When hardware systems are deployed, they are only able to execute the functions which they were designed for. In contrast, software's behaviors are able to change more flexibly based on their internal code. In our case, the deployed BWA-MEM's hardware accelerator has a fixed number of cores. When the IP Core deals with short read strings, there is small proportion that target strings can fill all PUs. It means there are some PUs that are still working despite the final result was calculated. These unnecessary operations make the whole process become longer than it should be. Our system achieves its highest performance only if the length of target strings is equal to the number of PUs.

The performance of the system is also affected by the number of PUs. The system achieves its ideal performance only if the number of integrated PUs is equal to the number of characters of query string. In our experiments, due to the limitation of resources, we can only deploy an IP Core with 30 PUs. Therefore, our experimental system does not reach its best. Despite the number of PUs does not equal the length of query string, the system still works properly thank to a mechanism that allows a new scoring process to begin with nonzero initial values. According to the experimental results, we can state that our proposed system has already achieved our final goals: providing an IP Core and an efficient system that can take advantage from the improved BWA-MEM algorithm.

TABLE II. EXECUTION TIME OF OUR PROPOSED SYSTEM IN COMPARISON WITH A GENERAL PURPOSE PROCESSOR (UNIT: MINUTES)

Read length	i5-4440	Zynq-7000	Resource utilization (%)
Short	15.98	3.57	86.7%
Medium	56.31	3.45	
Long	149.32	3.27	

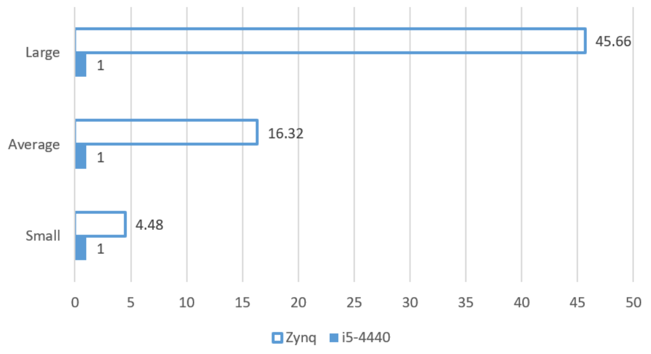


Fig. 6. Speed-up of proposed system (with 30 PUs) when it is compared with Intel Core i5 processor.

VI. CONCLUSION

In this paper, by optimizing one of the three essential stages of BWA-MEM algorithm as well as providing an efficient optimized communication protocol and connections, we have presented an BWA-MEM's hardware accelerator based on FPGA. Our proposed system, as well as the improved BWA-MEM algorithm's hardware implementation, has already reached the following features:

1) High performance: speed-up achieved up to 46.52x for both processing and transmission time.
2) High throughput: 2-level pipeline noticeably increase the throughput of the whole system.
3) Extensibility: the number of IP Cores in single FPGA device can be integrated when a specific bus that supports addressing mechanism is used. Moreover, thank to our designed protocol, as well as its handler, host is able to easily control multiple cores.
4) Soundness: by providing an optimized and reliable communication protocol, we ensure that the transmission is fast and accurately operated.

BWA-MEM algorithm is a fast and accurate aligner that can works well for wide variety of length of sequence. BWA-MEM can be still accelerated by both software and hardware. On our next effort, we will focus on reduce the charge that is caused by the limited flexibility of hardware. This is the reason that lead the less-efficient performance of our system when there is a huge gap between the length of target string and the length of query string.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Binh Kieu-Do-Nguyen implemented the system and conducted experiments. He also contributed in writing the paper. Cuong Pham-Quoc proposed the system architecture and structure of the paper. He also wrote the paper. Cong-Kha Pham discussed and adviced on the system architecture and implementation. He also proofed read the paper.

Acknowledgements

This research is funded by Department of Science and Technology of Ho Chi Minh City under grant number 09/2018/HĐ-QKHCN.

References

[1] H. Z. Cao, Y. Wang, W. Zhang *et al*., "A Short-read multiplex sequencing method for reliable, cost-effective and high-throughput genotyping in large-scale studies," *Human Mutation*, vol. 34, 2014.

[2] B Langmead, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *J. Genome Biol.*, vol. 25, pp. 423–433, 2009.

[3] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 25, 1754–1760, 2009.

[4] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Research*, vol. 18, 1851–1858, 2008.

[5] R. Q. Li, C. Yu *et al.*, "SOAP2: An improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, pp. 1966–1967, 2009.

[6] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," ArXiv 1303, 2013.

[7] H. Li. (2010). *Burrows-Wheeler Aligner*. [Online]. Available: http://biobwa.sourceforge.net/

[8] P.-Q. Cuong, K. Binh, and T. N. Thinh, *An FPGA-Based Seed Extension IP Core for BWA-MEM DNA Alignment*, pp. 1–6, 2018.

[9] Cuong Pham-Quoc, Kieu Binh, and Tran Ngoc Thinh, "A high-performance FPGA-based BWA-MEM DNA sequence alignment," Concurrency and Computation: Practice and Experience, 2019.

[10] Temple F. Smith and Michael S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[11] N. Ahmed, V. Sima, E. Houtgast, K. Bertels, and Z. Al-Ars, "Heterogeneous hardware/software acceleration of the BWA-MEM DNA alignment algorithm," in *Proc. 2015 IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 240–246.

[12] K. Benkrid, A. Akoglu, C. Ling, Y. Song, Y. Liu, and X. Tian, "High Performance Biological Pairwise Sequence Alignment: FPGA versus GPU versus Cell BE versus GPP," *Int. J. Reconfig. Comp.*, pp. 752910:1– 752910:15, 2012.

[13] Y. Chen, J. Cong, J. Lei, and P. Wei, "A novel high-throughput acceleration engine for read alignment," in *Proc. 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, 2015, pp. 199–202.

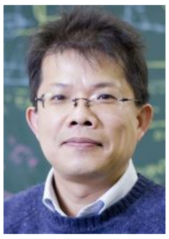
[14] E. J. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "An FPGA-based systolic array to accelerate the BWA-MEM genomic mapping algorithm," in *Proc. 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2015, pp. 221–227.

[15] The National Center for Biotechnology Information. (2019). [Online]. Available: https://www.ncbi.nlm.nih.gov/guide/all/

Binh Kieu-Do-Nguyen graduated with a BS in computer science and engineering from Ho Chi Minh City University of Technology (HCMUT) of Viet Nam in 2017. He obtained his master's degree in computer science from HCMUT in 2019. He is currently a teacher assistant of Computer Science and Engineering Faculty at HCMUT. His current research interests are in the high-performance computing, heterogeneous hardware accelerators, hardware/software co-design, high-level synthesis. He is pursuing his doctoral in the same specialization.

Cuong Pham-Quoc received the BSc degree in 2007, and the MEng degree in 2009, both from the Faculty of Computer Science and Engineering, the Ho Chi Minh City University of Technology (HCMUT). He moved to the Computer Engineering Lab (it now is Quantum & Computer Engineering), the Delft University of Technology, the Netherlands, from May 2011 to pursue the Ph.D. degree. In May 2015, he came back to the Ho Chi Minh City University of Technology, Vietnam. Currently, he has been a lecturer at the HCMUT where he got the position before he left for TUDelft in 2011. He now is serving as the head of the Department of Computer Engineering, Faculty of Computer Science and Engineering.

His research interests are in multi-/many-core architecture, high-performance computing, heterogeneous hardware accelerators, on-chip interconnect, high-speed network security on reconfigurable hardware, hardware/software co-design, polymorphic processor design, high-level synthesis, wireless sensors network, internet of Things, architecture and technology for smart cities.



Cong-Kha Pham is a lecturer in the University of Electro-Communications. He received his B.S. and also his M.S. in Sophia University. He is currently a professor in the University of Electro-Communications in the city of Chofu, Tokyo, Japan.

He specializes in energy harvest power supply and low-power data-centric sensor network system utilizing the energy harvest, development of long-distance transmission / miniaturization equipment of sensor network by low power wireless, super low-voltage device, memory-based information detection system, hardware implementation of hardware system by FPGA and integrated circuit, etc.

Professor Pham is teaching many undergraduate and postgraduate students and has received numerous awards for dissertation.