IEICE Electronics Express, Vol.16, No.15, 1–6

LETTER

Pair-HMM accelerator based on non-cooperative structure

Pengfei Wang[1], **Yuanwu Lei**[2a)], **and Yong Dou**[1]

Abstract Pair Hidden Markov Model (Pair-HMM) forward algorithm is gaining increasing popularity in biological research tools. We propose a novel non-cooperative structure of Pair-HMM forward algorithm accelerator on Field Programmable Gate Array (FPGA). We employ a task-level parallel scheme in the structure. We design the non-cooperative Processing Element (PE) to complete Pair-HMM forward algorithm independently. Our three-layer tree topology improves the scalability for different FPGAs. Compared to previous works, our structure reduces the idle cycles which occurs in the systolic array structure and the PE ring structure. Compared with the PE ring, our implementation on Arria 10 can achieve 1.19× speedups.

Keywords: acceleration, FPGA, non-cooperative structure, Pair-HMM forward algorithm
Classification: Integrated circuits

1. Introduction

Gene sequence alignment compares the target gene sequence with the reference one. It identifies regions of similarity which may be a consequence of functional, structural and evolutionary relationships between the sequences. It is mainly applied in precision medicine, screening for the newborn and screening for carriers of disease-causing mutations [1, 2, 3, 4, 5], etc. The Pair Hidden Markov Model (Pair-HMM) has various inference algorithms such as optimal sequence alignment (Viterbi algorithm [6]) and the overall alignment probability (forward algorithm [7]). Pair-HMM forward algorithm is one of the most widely used algorithms in DNA sequence alignment [8]. According to [9], the working stage of Pair-HMM forward algorithm occupied 70% runtime. As a consequence, it is the core and the most time-consuming algorithm in GATK HaplotypeCaller [10, 11].

Gene sequence alignment has the characteristics of a large amount of data and calculation. For example, the human genome includes more than 3 billion base pairs [12]. The number of gene sequence alignment tasks reaches more than one million. Therefore, many researchers use parallel platforms [13] such as multi-core CPUs [9, 14], GPUs [15, 16, 17, 18, 19] and Field Programmable Gate Arrays (FPGAs) [20, 21, 22, 23, 24, 25] to develop parallelism in gene sequence alignment applications. Compared with CPUs and GPUs, FPGA chips serve as custom hardware for the computing-intensive applications with numerous computing and storage resources [26, 27]. Thus, FPGAs could potentially implement gene sequence alignment and provide significantly higher performance.

At present, the systolic array structure [12, 24, 28, 29] and the improved ring structure [25] are adopted in the implementation of the FPGA-based Pair-HMM accelerator. The systolic array structure is a large computation pipeline. It integrates multiple PEs and uses these to cooperatively calculate the elements in the gene sequence alignment task. However, this structure will cause a lot of idleness of PEs in the startup and flush phases. And due to the cooperation needs waiting and synchronization, the idleness of PEs also occurs when the lengths of gene sequences are not an integral multiple of the PE number in the array. In order to overcome these defects, reference [25] has proposed a PE ring structure, which connects the head PE and the tail PE of the systolic array through the buffers and provides high effectiveness for variable lengths of gene sequences. However, the idleness of PEs occurs in the PE ring structure when the lengths of gene sequences are not an integral multiple of the PE number in the ring. The designs of the systolic array structure and PE ring structure show the pursuit of the throughput of gene sequence alignment.

However, the number of sequence alignment tasks is huge and the lengths of gene sequences vary greatly. In view of the overall performance and throughput for the entire application dealing with variable-size gene sequences, we propose a non-cooperative PE structure to complete gene sequence alignment tasks independently. Firstly, each non-cooperative PE can provide high computing efficiency for variable-size gene sequences. Secondly, non-cooperative PEs are integrated into our Pair-HMM accelerator to improve the overall performance with the task-level parallel scheme. Thirdly, a three-layer tree topology of non-cooperative PEs is proposed to improve the scalability of our accelerator.

The experimental result shows that our design can achieve the higher computing efficiency. Compared with the C++ baseline and the PE ring implementation, our work can achieve an acceleration of 575× and 1.19×, respectively.

The rest of this paper is organized as follows: Section 2 outlines the basic principles of the Pair-HMM forward algorithm. Section 3 describes the design of our forward algorithm accelerator based on the non-cooperative structure. Section 4 illustrates the experimental results. Section 5 summarizes our work.

[1]National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China
[2]College of Computer, National University of Defense Technology, Changsha 410073, China
a) yuanwulei@nudt.edu.cn

DOI: 10.1587/elex.16.20190402
Received June 23, 2019
Accepted July 9, 2019
Publicized July 19, 2019
Copyedited August 10, 2019

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

2. Pair-HMM forward algorithm

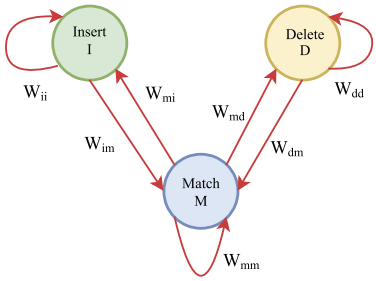


Fig. 1. Pair-HMM states transition diagram.

Pair-HMM gives a probability distribution over certain sequences of pairs of observations including the target gene sequence and the reference sequence. There are three hidden states in Pair-HMM, called Match (M), Insertion (I) and Deletion (D). The state transitions are shown in Fig. 1. It shows that the transition relationship among the three states and that each weight W on the line represents the probability of state transitions. We define the matrix M by $M_{r,c}$ which indicates the total likelihood of all paths from the beginning of the Read sequence to position r in the match state. $D_{r,c}$ and $I_{r,c}$ are defined likewise. And the recursions

$$M_{r,c} = Prior(r,c) \cdot (W_{mm} \cdot M_{r-1,c-1} + W_{im} \cdot I_{r-1,c-1} + W_{dm} \cdot D_{r-1,c-1}) \quad (1)$$

$$I_{r,c} = W_{mi} \cdot M_{r-1,c} + W_{ii} \cdot I_{r-1,c} \quad (2)$$

$$D_{r,c} = W_{md} \cdot M_{r,c-1} + W_{dd} \cdot D_{r,c-1} \quad (3)$$

define the entire Pair-HMM forward algorithm used in the GATK HaplotypeCaller tool [30]:

Algorithm 1: Pair-HMM forward algorithm

Input: $Read\_base{:}\mathcal{R}$, $Haplotype\_base{:}\mathcal{H}$, $base\_quals{:}bq$, $ins\_quals{:}iq$, $del\_quals{:}dq$.
Output: Total likelihood:$result$

1 Initialize $M_{0,c} = I_{0,c} = 0$ and $D_{0,c} = 2^{120}/|\mathcal{H}|$ for $0 \le c \le |\mathcal{H}|$, $M_{r,0} = I_{r,0} = D_{r,0} = 0$ for $1 \le r \le |\mathcal{R}|$, $result$=0 and the error rate sequence $\epsilon[q] = 10^{-q/10}$, for $0 \le q \le 128$.
2 **for** $1 \le r \le |\mathcal{R}|$ **do**
3 **for** $1 \le c \le |C|$ **do**
4 Calculate $M_{r,c}$ via Eq. (1).
5 Calculate $I_{r,c}$ via Eq. (2).
6 Calculate $D_{r,c}$ via Eq. (3).
7 Total likelihood result is $\sum_c (M_{|\mathcal{R}|,c} + I_{|\mathcal{R}|,c})$.

As shown in Eq. (1,2,3) and Algorithm 1, the critical path for the calculation of M is longer than the critical paths for calculating I and D. In the FPGA filed, it means idleness and waiting. As a consequence, in reference [25], two temporary variables ta, tb are added, which help to complete part of the calculation operations in advance when M is updated, so as to achieve regularizing the length of the critical path. Therefore, when we implement the improved algorithm, the memory cost will increase accordingly and we will analyze the storage requirement in section 3.2.3.

On this basis, we get our improved version of the Pair-HMM forward algorithm as shown in Algorithm 2.

The input of the Pair-HMM forward algorithm consists of two parts: The first part is the reference gene part, called the Haplotype sequence. The second part is the target part, including the Read sequence and three quality factor sequences (base-quals, ins-quals, and del-quals). The intermediate results of the algorithm include M, I, D, ta and tb. The output is an overall similarity value of the Read sequence compared with the Haplotype sequence. The parameters in Pair-HMM forward algorithm are defined as follows:

$$W_{mm}(r) = 1 - \epsilon[iq[r-1] + dq[r-1]], \quad (4)$$

$$W_{im}(r) = W_{dm}(r) = 0.9, \quad (5)$$

$$W_{mi}(r) = \epsilon[iq[r-1]], \quad (6)$$

$$W_{ii}(r) = W_{dd}(r) = 0.1, \quad (7)$$

$$W_{md}(r) = \epsilon[dq[r-1]], \quad (8)$$

$$Prior(r,c) = \begin{cases} 1 - \epsilon[bq[r-1]] & , if(\mathcal{R}_r = \mathcal{H}_c), \\ (1/3) * \epsilon[bq[r-1]] & , if(\mathcal{R}_r \neq \mathcal{H}_c), \end{cases} \quad (9)$$

Algorithm 2: Pair-HMM algorithm improved version

Input: $Read\_base{:}\mathcal{R}$, $Haplotype\_base{:}\mathcal{H}$, $base\_quals{:}bq$, $ins\_quals{:}iq$, $del\_quals{:}dq$.
Output: Total likelihood: $result$

1 Initialize $M_{0,c} = I_{0,c} = 0$ and $D_{0,c} = 2^{120}/|\mathcal{H}|$ for $0 \le c \le |\mathcal{H}|$, $M_{r,0} = I_{r,0} = D_{r,0} = 0$ for $1 \le r \le |\mathcal{R}|$, $result$=0 and the error rate sequence $\epsilon[q] = 10^{-q/10}$, for $0 \le q \le 128$.
2 **for** $0 \le k < (|\mathcal{R}| + |\mathcal{H}|)$ **do**
3 **for** $max(1 + k - |\mathcal{H}|, 0) \le r \le min(k+1, |\mathcal{R}|)$ **do**
4 $c = k - r + 1$
5 $ta_{r,c} = W_{im}(r) \cdot I_{r,c-1} + W_{dm}(r) \cdot D_{r,c-1}$
6 $tb_{r,c} = W_{mm}(r) \cdot M_{r,c-1}$
7 $M_{r,c} = Prior(r,c) \cdot (ta_{r-1,c} + tb_{r-1,c})$
8 $I_{r,c} = W_{mi}(r) \cdot M_{r-1,c} + W_{ii}(r) \cdot I_{r-1,c}$
9 $D_{r,c} = W_{md}(r) \cdot M_{r,c-1} + W_{dd}(r) \cdot D_{r,c-1}$
10 **if** $r = |\mathcal{R}|$ **then**
11 $result = result + (M_{r,c} + I_{r,c})$

$\epsilon[q]$ is the error rate implied by the phred-scaled quality q. The input data becomes the indeices of ϵ to generate the weight parameters used in the calculation.

To illustrate the algorithm, we define two concepts firstly.

Task: Complete calculation of the alignment of a pair sequences including Read and Haplotype sequences.

Computation matrix: In a task, the computation matrix represents the calculation process where each position in the Haplotype and the Read sequences is compared. The size of the matrix is $length(Haplotype) \times length(Read)$. Each element in the matrix represents operations of line 5 to 9 in Algorithm 2.

As shown in Fig. 2, the Pair-HMM forward algorithm is performed with diagonal traversal order. k represents the k^{th} diagonal. The element $[r,c]$ of the computation matrix indicates that the r^{th} position of the Read sequence is

compared with the c^{th} position of the Haplotype sequence. The final result, shown in Algorithm 2 line 10 to 11, is obtained by accumulating the M and I values in all elements of the last row in the computation matrix.

From Algorithm 2 and Fig. 2, we can conclude that:

1) The calculation of each diagonal element depends on the calculations of previous diagonal elements. The element $[r, c]$ depends on the elements $[r, c-1]$ and $[r-1, c]$.
2) There is no data dependence among elements on the same diagonal, of which the elements can be executed in parallel.

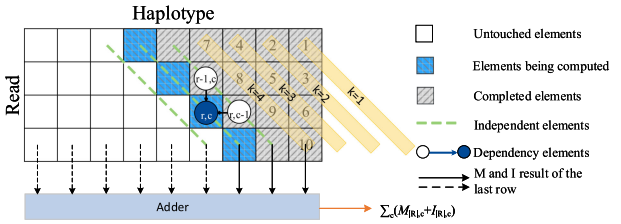


Fig. 2. The features of Pair-HMM forward algorithm.

3) The Pair-HMM forward algorithm is a computing-intensive algorithm. Time complexity is $O(n * m)$ and memory access overhead is $O(n + m)$, where n and m represent the length of Read sequence and Haplotype sequence respectively.
4) The tasks are independent of each other. Thus, task-level parallelism scheme can be used to improve performance.

3. Non-cooperative structure Pair-HMM accelerator

Taking all the above information into consideration, we design the non-cooperative PE and the overall structure.

The non-cooperative PE is shown in Fig. 3(c). It is an independent task processing unit, which can process one task individually without any cooperation with others. The PE contains the calculation unit, the PE controller, the result adder and data storage components. The PE controller realizes the management of the state machine, the control of other functional components, and the signal interaction with the upper module. Shown in Fig. 3(d), the calculation unit accomplishes operations of line 5 to 9 in Algorithm 2.

Fig. 4 shows the calculation process of PEs. There are N_P PEs, and each PE performs a single sequence alignment task independently. The block named $Raw^i_{j,k}$ represents the raw data from the j^{th} element of the Read sequence and the k^{th} element of the Haplotype sequence in the i^{th} task. These raw data are used as the indices of the error rate sequence to generate the weights shown in Eq. (4–9). The block named IR^i_j represents the j^{th} element of the intermediate result data in the i^{th} task, where the order of the indices is incremented along the diagonal elements. The updated element overwrites the element with the same index after its calculation because the elements with data dependence have been used in the calculation. Therefore, the intermediate result data can be stored in the form of sequences instead of matrices. The controller controls storage components to feed data into the calculation pipeline cycle by cycle. After the data of the last element of the current diagonal have been fed into the pipeline, if the calculation of the first element of this diagonal is not finished, idle operations will be inserted. Otherwise, the calculation of the next diagonal will be started immediately. The data in one task are only stored in one PE by the control of the upper module. Without any cooperation among PEs, tasks can be processed independently.

On this basis, we propose a three-layer tree topology structure to manage a large number of integrated PEs on FPGAs. The overall design of the pair-HMM accelerator is shown in Fig. 3. As shown in Fig. 3(a), there are N_A top modules named Allocation. To realize load balancing, one Allocation utilizes a FIFO to share the high bandwidth with a time-sharing multiplexing scheme. SuperPE is the module of the middle layer of the three-layer topology. Fig. 3(b) shows the structures of the Allocation and the SuperPE. The structures of these two modules are similar and realize the transmission to the lower layer module with the distribution module. Through multiplexers, their results are transmitted to the corresponding buffer. Each Allocation can run independently with great flexibility and has N_S SuperPE modules inside. SuperPEs are executed sequentially. Its main function is to buffer and distribute the input data of the upper layer. There are N_P PEs inside a SuperPE and these PEs work in a polling manner. Adopting the polling task allocation strategy ensures load balancing among PEs and simplifies control. The three-layer tree topology structure has the advantages of facilitating layout, routing and obtaining better scalability. We can deploy PEs according to the hardware resources on the FPGA to achieve the best performance.

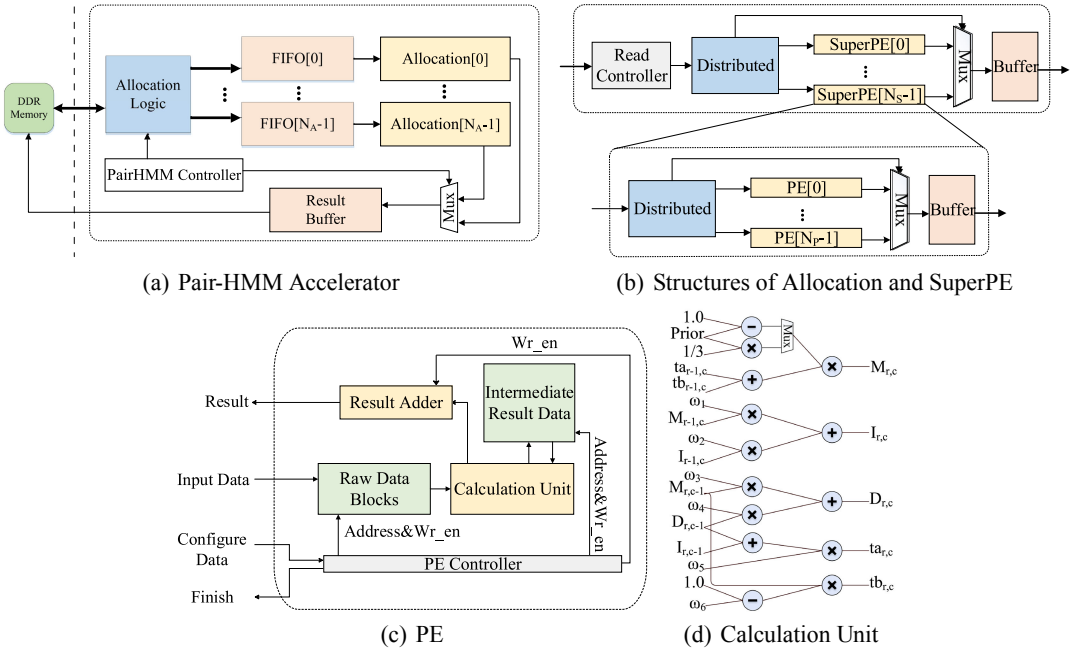
4. Experiments and results

4.1 Experimental environment

Our test data is from the Whole Genome Sequence data set [31], generated by the HaplotypeCaller from GATK version 2.7. The baseline consists of three data sets with different lengths. Each data set contains aligned sequences consisting of the Read and Haplotype pairs ranging in length from 10 to 302 bases. In our experiment, our design is implemented on the Xilinx's xc7vx485tffg1761-2L, xc7vx690tffg1761-2L, xc7vx980tffg1930-2 and Altera's Arria 10 (10AX115H1F34E1SG).

4.2 Implementation on FPGAs

The resources and frequency situation in our design implemented on FPGAs are shown in Table I. On the Virtex-7 platform, the maximum frequency in our design achieves 246.91 MHz. On the Arria 10 platform, the maximum frequency achieves 230 MHz. And the clock cycles spent on the testing data sets with 64 PEs and 128 PEs implementations are shown in Fig. 5. In the "10s" and "1m" data sets, performance on the implementation with 128 PEs achieves two times than that with 64 PEs. In "tiny" data set, the performance improves 1.64×. This is because when the number of sequences is small, it is difficult to balance



(a) Pair-HMM Accelerator (b) Structures of Allocation and SuperPE

(c) PE (d) Calculation Unit

Fig. 3. Implementations use data sets for testing

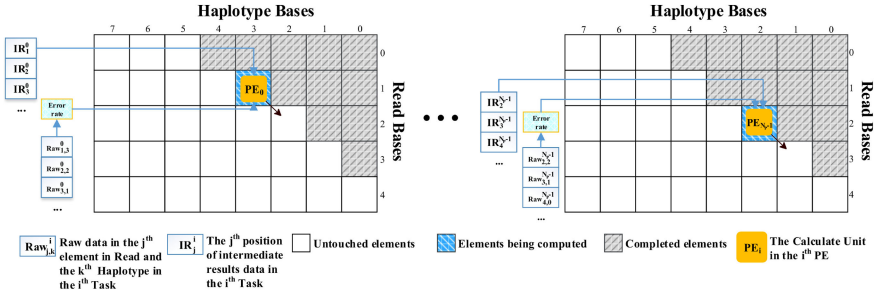


Fig. 4. The calculation process of the structure

the load of PEs and hide the overhead of the initialization of PEs. Thus, the performance of 128 PEs is not fully utilized.

From the perspective of parameter settings, there are some slight differences between the Pair-HMM forward algorithm in our acceleration design and [25]. In our algorithm, the operations of calculating the "Prior" parameter add one more floating-point multiplication operation. It results in a single PE using more computing resources than [25].

Table I. Synthesis results for target FPGAs

FPGA	xc7vx485t	xc7vx690t	xc7vx980t	Arria 10
#PE	64	64	128	128
Frequency	227.27 MHz	246.91 MHz	222.22 MHz	230 MHz
LUT	69.82%	56.81%	88.64%	90%
DSP	91.43%	49.78%	99.56%	93%
BRAM	45.15%	31.63%	57.07%	25%

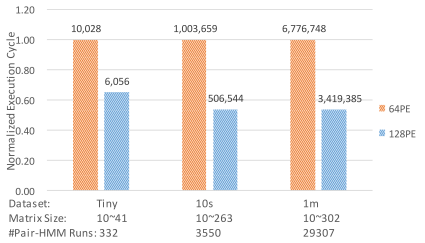
4.3 Impact of PE numbers

When the total number of PEs is constant, the adjustment of the number of Allocations and SuperPEs will affect the performance. As shown in Fig. 6, we adjust the number of Allocations N_A and the number of PEs in one Allocation module, which is $N_S * N_P$. Fig. 6 shows the normalized execution time executed on three testing data sets, where the "$p \times q$" in the figure legend means $N_A = p$, $N_S * N_P = q$. In the case of a certain number of PEs, the larger N_A, the better the performance. But it is difficult to meet the bandwidth requirement with the increasing N_A. Thus, we do not further increase N_A larger than 8.

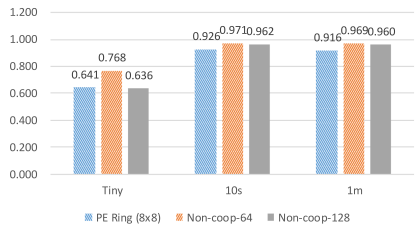
4.4 Performance comparison with related work

Comparing Fig. 6(a) with Fig. 6(b), both in the optimal configuration, the performance of our design with 128 PEs is 1.73× than that of our design with 64 PEs. Because higher memory bandwidth is needed on the 128 PEs version, the performance of the 128 PEs version cannot be doubled.

Fig. 6 and Table II show the performance comparison between our implementation and the other platforms, including multi-core CPUs, GPUs and FPGAs. The "10s" data set is used in Table II. Compared with GPU implementation with Nvidia K40 [29], our design can achieve a speedup with 32.27×. Reference [18] has improved the warp-based GPU implementation on a real data set. How-

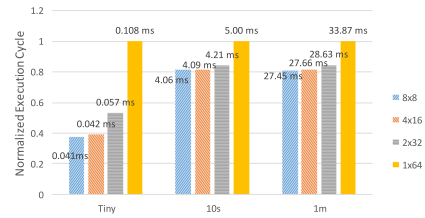


(a) Cycles on three testing data sets in two implementations

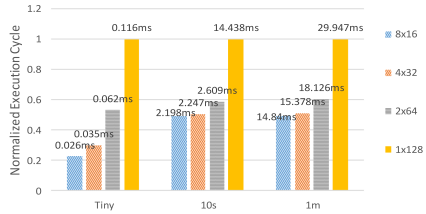


(b) Comparison of effective cycles ratio in different structures

Fig. 5. Implementations use data sets for testing



(a) 64 PEs on XC7VX690T



(b) 128 PEs on Arria 10

Fig. 6. Normalized execution time on three data sets when using different sizes of one Allocation

ever, due to the small number of read-haplotypes pairs and the unbalance load of multiple threading, it is not using the GPU resources efficiently. Thus, our design can achieve a 5.89× speedup in comparison to improved warp-based GPU version.

As shown in Fig. 6, two types of FPGA platforms are used to compare the performances of our non-cooperative structure with that of PE ring structure [25], which has the highest performance among FPGA implementations of Pair-HMM as far as we know. 128 PEs are integrated into Altera Arria 10 respectively both in our design and [25]. Due to less idle cycles and easier to achieve load balancing among PEs for different lengths of read-haplotypes pairs on real data sets, our non-cooperative structure has improved performance by an average factor of 1.19× over the PE ring structure.

Table II. Performance comparison on implementations

Platform	Runtime (ms)	Speedup
Original Java version [9]	10800	1×
C++ Baseline [9]	1267	9×
Intel Xeon 24 Cores [25]	15	720×
NVidia K40 GPU [29]	70	154×
Improved warp based [18]	12.8	843×
PE Ring (Stratix V) [25]	5.3	2038×
Alter OpenCL (Arria 10) [25]	2.8	3857×
PE Chunks (Xilinx KU3) [22]	5.0	2160×
PE Ring (Arria 10) [25]	2.6	4154×
Non-coop-64PE (XC7VX690T)	5.0	2609×
Non-coop-128PE (Arria 10)	2.2	4970×

5. Conclusion and future work

In our work, we implemented a Pair-HMM forward algorithm accelerator with non-cooperative PEs that process the individual task independently through a tree topology. The non-cooperative implementation achieved a 575× speedup compared to the C++ program and a 19% performance improvement over the implementation of the PE ring structure.

We analyzed the relationship between the performance and the number of lower modules mounted at each level in the tree topology. For FPGAs with different computing resources, we can choose the correct configuration parameters so that we can flexibly port the accelerator and maximize the use of computing resources for the best performance which improves the applicability of our structure. Our non-cooperative structure is quite different from other structures in the calculation mode and PE schemes. The differences exist not only in providing a task scheduling mechanism for PEs but also in the working mechanism in every PE. Our non-cooperative structure digs out the independence of the data in one comparison task and uses one PE to solve one comparison task to realize multiple PEs solving different tasks independently. It shows higher computing efficiency than FPGA implementations of other structures.

In the future, flexible configuration of PE numbers according to specific different FPGA resources can be one direction of further research. And the idle cycles in the startup and flush phases can be utilized among the PEs for less idleness overall. We believe that with the rapid development of bioinformatics, FPGA will play a more important role.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under No. 2018YFB1003405.

References

[1] M. J. Khoury, *et al.*: “Population screening in the age of genomic medicine,” N. Engl. J. Med. **348** (2003) 50 (DOI: 10.1056/NEJMra013182).
[2] E. T. Juengst: ““Prevention” and the goals of genetic medicine,” Hum. Gene Ther. **6** (1995) 1595 (DOI: 10.1089/hum.1995.6.12-

1595).
[3] N. J. Wald: "The definition of screening," J. Med. Screen. **8** (2001) 1 (DOI: 10.1136/jms.8.1.1).
[4] L. L. McCabe, *et al.*: "Newborn screening: Rationale for a comprehensive, fully integrated public health system," Mol. Genet. Metab. **77** (2002) 267 (DOI: 10.1016/S1096-7192(02)00196-8).
[5] B. Hill, *et al.*: "Precision medicine and fpga technology: Challenges and opportunities," IEEE MWSCAS (2017) 655 (DOI: 10.1109/mwscas.2017.8053008).
[6] R. Li, *et al.*: "A high-throughput reconfigurable Viterbi decoder," IEEE WCSP (2011) 1 (DOI: 10.1109/WCSP.2011.6096781).
[7] R. Durbin, *et al.*: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge University Press, Cambridge, 1998) 77 (DOI: 10.1017/CBO9780511790492).
[8] J. Shendure and H. Ji: "Next-generation DNA sequencing," Nat. Biotechnol. **26** (2008) 1135 (DOI: 10.1038/nbt1486).
[9] M. Carneiro: "Accelerating variant calling," in Broad Institute Intel Genomic Sequencing Pipeline Workshop Powerpoint Presentation Mount Sinai (2013) Available online: https://hpc.mssm.edu/files/Carneiro_workshop.pdf.
[10] A. McKenna, *et al.*: "The genome analysis toolkit: A mapreduce framework for analyzing next-generation DNA sequencing data," Genome Res. **20** (2010) 1297 (DOI: 10.1101/gr.107524.110).
[11] M. A. DePristo, *et al.*: "A framework for variation discovery and genotyping using next-generation DNA sequencing data," Nat. Genet. **43** (2011) 491 (DOI: 10.1038/ng.806).
[12] C. Rauer and N. Finamore: "Accelerating genomics research with opencl and fpgas," Altera, Now Part of Intel Tech. Rep 2016 (DOI: 10.1145/3078155.3078179).
[13] S. Aluru and N. Jammula: "A review of hardware acceleration for computational genomics," IEEE Des. Test **31** (2014) 19 (DOI: 10.1109/MDAT.2013.2293757).
[14] G. VdAuwera: "Speed up haplotypecaller on ibm power8 systems," Available online: https://gatkforums.broadinstitute.org/gatk/discussion/4833/speed-up-haplotypecaller-on-ibm-power8-systems.
[15] X. Li, *et al.*: "A speculative hmmer search implementation on gpu," IEEE IPDPSW (2012) 735 (DOI: 10.1109/ipdpsw.2012.91).
[16] H. Jiang and N. Ganesan: "Cudampf: A multi-tiered parallel framework for accelerating protein sequence search in hmmer on cuda-enabled gpu," BMC Bioinformatics **17** (2016) 106 (DOI: 10.1186/s12859-016-0946-4).
[17] S. Ren, *et al.*: "Exploration of alternative gpu implementations of the pair-hmms forward algorithm," IEEE BIBM (2016) 902 (DOI: 10.1109/BIBM.2016.7822645).
[18] S. Ren, *et al.*: "Efficient acceleration of the pair-hmms forward algorithm for gatk haplotypecaller on graphics processing units," Evol. Bioinform. Online **14** (2018) 1 (DOI: 10.1177/1176934318760543).
[19] S. Che, *et al.*: "Accelerating compute-intensive applications with gpus and fpgas," IEEE SASP (2008) 101 (DOI: 10.1109/sasp.2008.4570793).
[20] L. W. Howes, *et al.*: "Comparing fpgas to graphics accelerators and the playstation 2 using a unified source description," IEEE FPL (2006) 1 (DOI: 10.1109/fpl.2006.311203).
[21] K. Benkrid, *et al.*: "High performance biological pairwise sequence alignment: Fpga versus gpu versus cell be versus gpp," IJRC **2012** (2012) 752910 (DOI: 10.1155/2012/752910).
[22] D. Sampietro, *et al.*: "Fpga-based pairhmm forward algorithm for DNA variant calling," IEEE ASAP (2018) 1 (DOI: 10.1109/ASAP.2018.8445119).
[23] S. S. Banerjee, *et al.*: "On accelerating pair-hmm computations in programmable hardware," IEEE FPL (2017) 1 (DOI: 10.23919/FPL.2017.8056837).
[24] J. Peltenburg, *et al.*: "Maximizing systolic array efficiency to accelerate the pairhmm forward algorithm," IEEE BIBM (2017) (DOI: 10.1109/BIBM.2016.7822616).
[25] S. Huang: "Hardware acceleration of the pair hmm algorithm for DNA variant calling," ACM FPGA (2017) (DOI: 10.1145/3020078.3021749).
[26] Y. Zhou and J. Jiang: "An fpga-based accelerator implementation for deep convolutional neural networks," IEEE ICCSNT (2015) 829 (DOI: 10.1109/iccsnt.2015.7490869).
[27] J. Shen, *et al.*: "Towards a multi-array architecture for accelerating large-scale matrix multiplication on fpgas," IEEE ISCAS (2018) 1 (DOI: 10.1109/iscas.2018.8351474).
[28] M. Ito and M. Ohara: "A power-efficient fpga accelerator: Systolic array with cache-coherent interface for pair-hmm algorithm," IEEE COOL CHIPS XIX (2016) 1 (DOI: 10.1109/CoolChips.2016.7503681).
[29] S. Ren, *et al.*: "Fpga acceleration of the pair-hmms forward algorithm for DNA sequence analysis," IEEE BIBM (2015) (DOI: 10.1109/BIBM.2015.7359892).
[30] D. Benjamin: "Pair hmm probabilistic realignment in haplotypecaller and mutect," Broad Institute. Available online: https://github.com/broadinstitute/gatk/blob/master/docs/pair_hmm.pdf.
[31] Pair-HMM test data. Available online: https://github.com/MauricioCarneiro/PairHMM/tree/master/test_data.