A 2.46M reads/s Genome Sequencing Accelerator using a 625 Processing-Element Array

Zhehong Wang[1], Tianjun Zhang[1], Daichi Fujiki[1], Arun Subramaniyan[1], Xiao Wu[1], Makoto Yasuda[2], Satoru Miyoshi[3], Masaru Kawaminami[2,3], Reetuparna Das[1], Satish Narayanasamy[1], David Blaauw[1]
[1]University of Michigan, Ann Arbor, MI, [2]Mie Fujitsu Semiconductor Limited, Yokohama, Japan,
[3]Fujitsu Electronics America, Inc., Sunnyvale, CA

***Abstract*—We present an accelerator for seed-extension, a critical and computational intensive step in genome sequencing. The accelerator consists of a triangular array of 25x25 custom design processing elements implementing a string-independent automata. It achieves 2.46M reads/s, a ~1800x performance improvement, and 27x smaller silicon footprint compared to a Xeon E5420.**

Keywords—ASIC, DNA Sequencing, seed-extension

I. Introduction

Production cost of whole human genome sequencing has plummeted by 10,000x from $10 million dollars to $1000 in just the last decade due to advances in next-generation-sequencing technology (NGS). This has led to wide use of DNA testing in both research and daily life. However, for each sequenced human genome, 396 GB of data must be processed (secondary analysis), which poses a significant computational challenge. Since sequencing technology is far outpacing Moore's law, this computation bottleneck is projected to dominate the total cost and processing time of sequencing, becoming a key factor in the growth of this important medical technology.

A DNA sequencer produces approximately 1.5B reads (Fig. 1) which are short DNA strands of 100 base pairs (BP). These reads then proceed through three processing steps: 1) In seeding, a set of possible locations where the read matches the reference is found by matching small fragments between the read and the reference exactly. 2) In seed-extension, these possible match locations are evaluated by exploring alignments between the read and the reference, including possible edits, to determine a final match location. 3) In variant calling, all the reads that are aligned to a particular BP in the reference are evaluated to determine if a mutation occurred at that location. An ASIC for the seeding phase of DNA sequence processing was presented [1], yet, little or no dedicated acceleration hardware has been proposed to address other steps in DNA sequencing.

In this paper, we target the seed-extension, which requires a total of 14 billion alignments for each human genome which takes 2754 CPU hours for one human genome on a single core Xeon E5420 processor using the standard BWA-MEM algorithm [2]. We use a 25x25 triangular array of processing elements (PEs) that implements a string-independent automata for approximate string matching, and also performs match score calculation and generation of the edit string. The proposed alignment accelerator, implemented in MIFS 55 nm DDC CMOS, operates at 670 MHz and achieves 2.46M reads per second with 8 mm^2 silicon area. Marking, to our knowledge, the first seed-extension ASIC, it presents ~1800x performance improvement and 27x smaller silicon footprint when operating on actual Genome read data compared to BWA-MEM on a Xeon E5420 and produces exactly identical output.

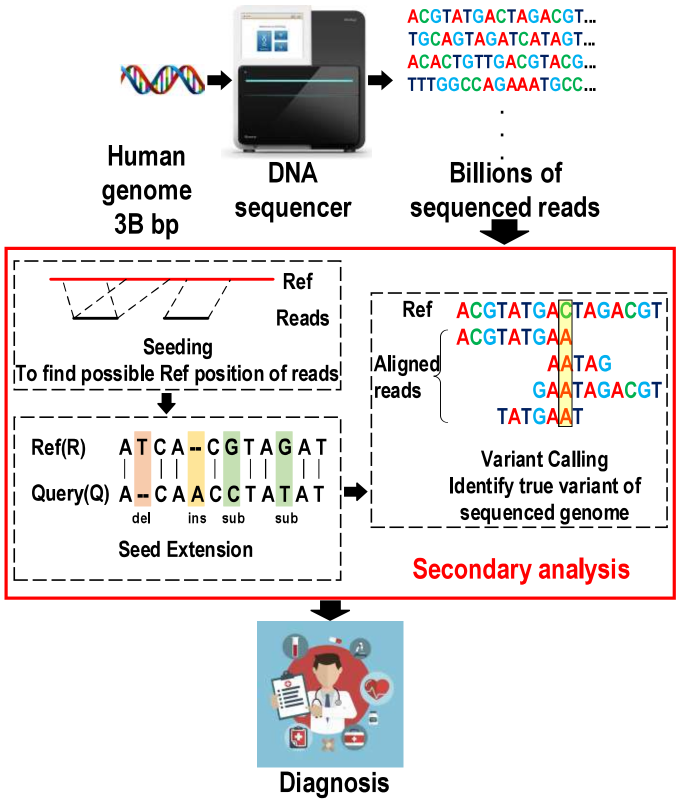

Fig. 1 Reference Guided Sequence Analysis Pipeline with Seed Extension Highlighted.

Seed-extension aligns two DNA strings of ~100 BP: the read or query (Q) and the portion of the reference (R) where the read is expected to align. However, there can be mis-matches between R and Q because of sequencing machine errors or mutations in the individual's DNA. Hence, approximate alignment is needed, allowing for Levenshtein edits: insert (i), delete (d), and substitute (s), as illustrated in Fig. 2. Fig. 3 shows two of many possible alignments for an R and Q pair, each with an edit distance or score. The task of seed-extension is to find the alignment with the best score and reporting this score and its associated string of edits.

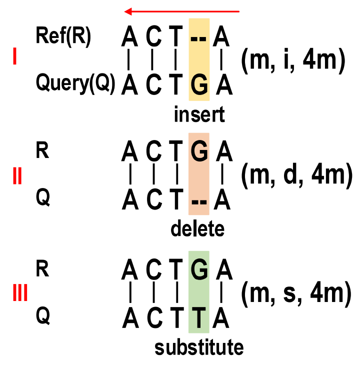

Fig. 2 Levenshtein Edits

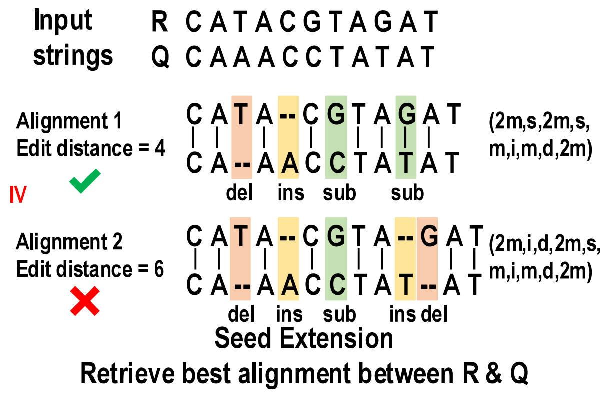

Fig. 3 Example of Alignment

II. Seed-Extension Accelerator

A recently proposed string-independent automata algorithm [3] is adopted. Unlike the standard Smith-Waterman algorithm [2] where R/Q strings remain static and possible alignment paths are explored with an array of PEs, we shift R/Q strings of arbitrary length through a state-machine that simultaneously evaluates all possible alignments between the two strings. The algorithm has the advantage that varying length strings can be processed using same matching hardware, so long as the maximum edit distance remains fixed.

Implementing this algorithm in silicon for the first time, the state-machine for this algorithm consists of a 3D grid of tiny PEs, represented as circles in Fig. 4, each performing a comparison of two BPs. Each PE represents a unique edit cost noted by its indices, e.g., PE120 corresponds to 1i+2d+0s. Note that in terms of edit cost, i = d = s = 1, so, e.g. 1i + 1d = 2s. Initially, in clock-cycle zero (c = 0), only PE000 is activated and compares the first two BPs of the R/Q strings. For clarity, only the activated PEs are shown in Fig. 4. Since the two BPs match, the PE reactivates itself indicating a match (m) edit string so far. On the next clock cycle, the R/Q strings are shifted right/up, and PE000 compares BP (G, T). Since there is a mismatch, PE000, deactivates itself and instead activates its three neighbors, PE100, PE010 and PE001 in c = 2. PE100 evaluates possible alignment of R/Q after 1i and therefore uses a shifted value of R. Since it compares BP (G, G) in c = 2, it finds a match and reactivates itself for the next cycle. Similarly, PE010 represents 1d and compares BP (T, T) and reactivates. PE001 represents 1s and therefore looks at the unshifted values of its inputs and compares BP (T, G) which is a mismatch. It therefore deactivates itself and in c = 3 activates PE101, PE011 and PE002. In c = 3, PE100 and PE101 compare BP (T, C) and find a mismatch. While they both compare the same R/Q BP, PE100 represents 1i and PE101 represents 1i and 1s; hence, their edit distance is not the same, preventing them from being merged. Similarly, PE010 and PE011 compare BP (C, G) and find a mismatch.

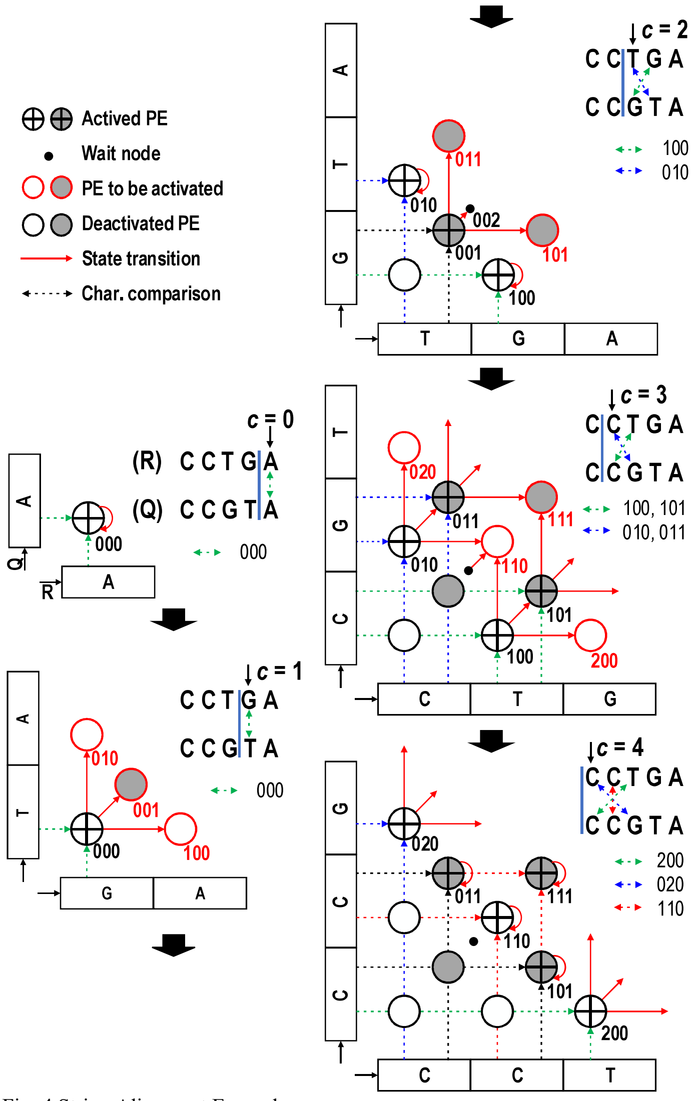

Fig. 4 String Alignment Example

PE002 represents 2s and would compare BP (C, C), which is a match. However, if we follow this pattern, a full 3D grid of PEs develops that would result in O(k^3) complexity. To reduce the complexity to O(k^2) instead, we use the observation that PE002 evaluates the same BP position in c = 3 as PE110 will evaluate in the next cycle c = 4. In our example, PE110 also compares BPs (C, C) in c = 4. Furthermore, the edit distance of both PEs = 2. Hence, they can be merged and instead of an actual PE002, there is only a wait state (indicated by the black dot in Fig. 4) which only activates PE110 in c = 4. Hence, in c = 4, PE110 finds a match (C, C) and after reactivating itself, again finds the final match (C, C) in c = 5 (not shown in Fig. 4), marking the optimal alignment of 2 edits. Note that several other PEs are also active in c = 4. However, all have higher edit distances and are sub-optimal. The resulting array has dimension (k, k, 2), where k is the maximum edit distance.

The overall architecture of the accelerator and PE array is shown in Fig. 5(a). The PEs are extremely simple, consisting of only 6 gates, which OR incoming activations and activate self/neighbors depending on the comparison result. The BP comparisons are performed at the shift registers and is then passed diagonally to neighboring PEs. This makes all communication local allowing for very high-speed operation. Fig. 5(b) shows that the 25x25 triangular array can be decomposed into three smaller triangular arrays of 12x12, enabling a trade-off between throughput and maximum edit distance.

The standard seed-extension algorithm typically uses a more sophisticated score scheme in addition to edit distance, based on empirical statistics [4]. This includes an affine gap penalty used in BWA-MEM, which favors consecutive i/ds over new i/d, preventing merging confluence paths, which can occur with simple edit distance, as shown in Fig. 6. We accommodate this in our PE by adding score calculation logic, which is passed from PE to PE along with the state activations (Fig. 7). This also complies with the clipping heuristic used in BWA-MEM.

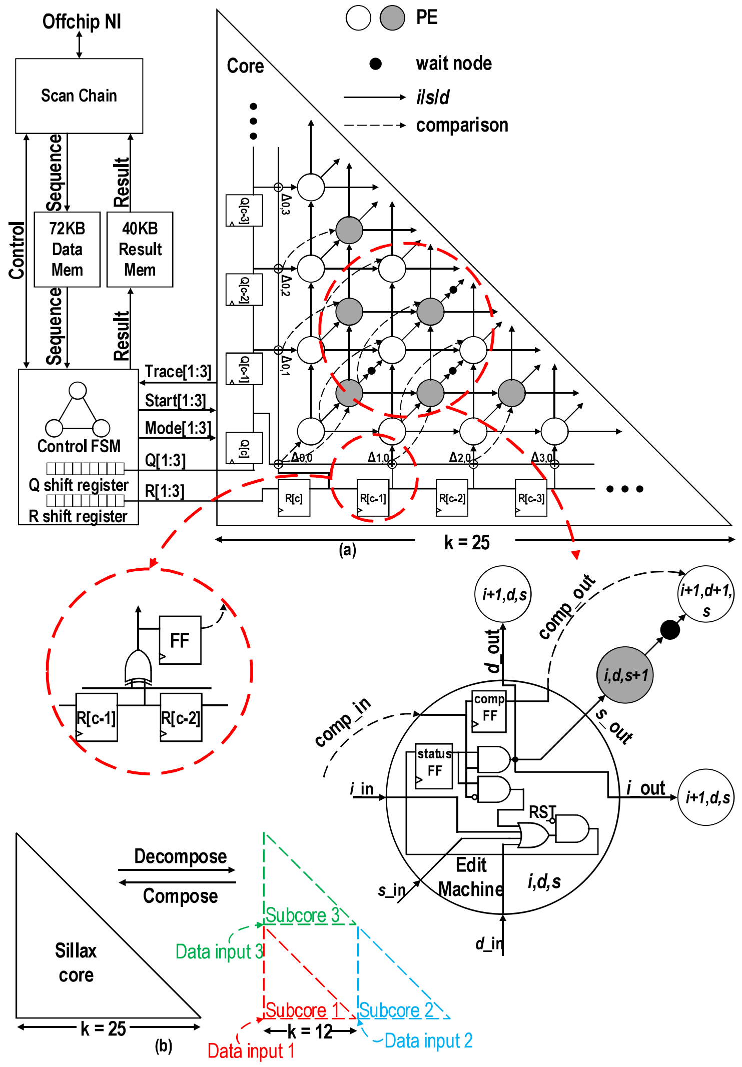

Fig. 5 (a) Overall Architecture of test chip and details (b) Composable Structure

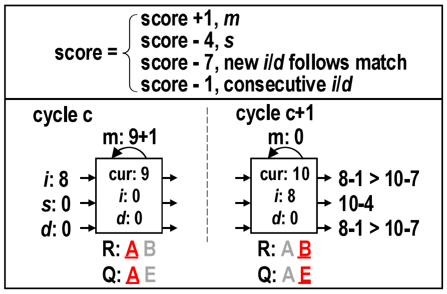

Fig. 6 Score Scheme and Delayed Merging

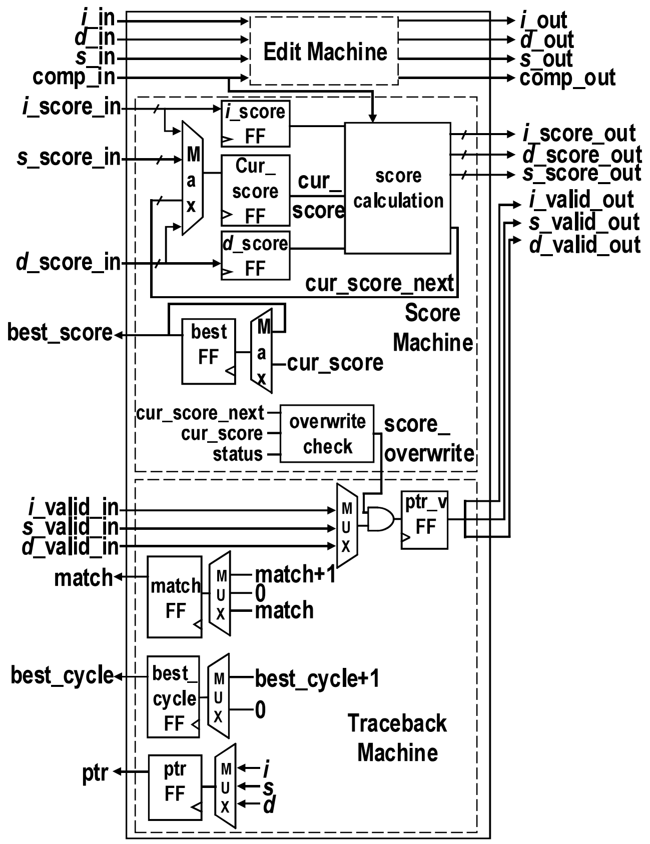

Fig. 7 PE Augmented with Score Logic

Fig. 8 shows the complete sequence of operation: after the full read is processed (phase I), each PE passes the maximum score backward in back-propagation mode, and the best score is retrieved from PE000 (phase II). The array controller also counts the number of cycles until the best-score exits the array and then reverses the machine and propagates the best score into the array again for the same number of cycles (phase III), at which point the node that matches the best score self-identifies. During forward processing, each PE also stores which of the incoming arcs (i, d, s) pass the best score. In phase IV, starting from the final winning state, the traceback pointers are connected in backward fashion till they reach PE000. Finally in phase V, this backward trace is collected by shifting it back to PE000, revealing the edit string. During phase I it is possible for the correct pointer to be corrupted by a later, non-optimal score (due to affine gap scoring) thus breaking the backward trace. When this is detected by the controller, the string is reprocessed up to the point when the broken state occurs and then traced back. Although this requires reprocessing the string, in practice, this is rare (0.93% of reads, Fig. 9), resulting in negligible performance degradation. Edit distance and time breakdown of the processing are also shown in Fig. 9.

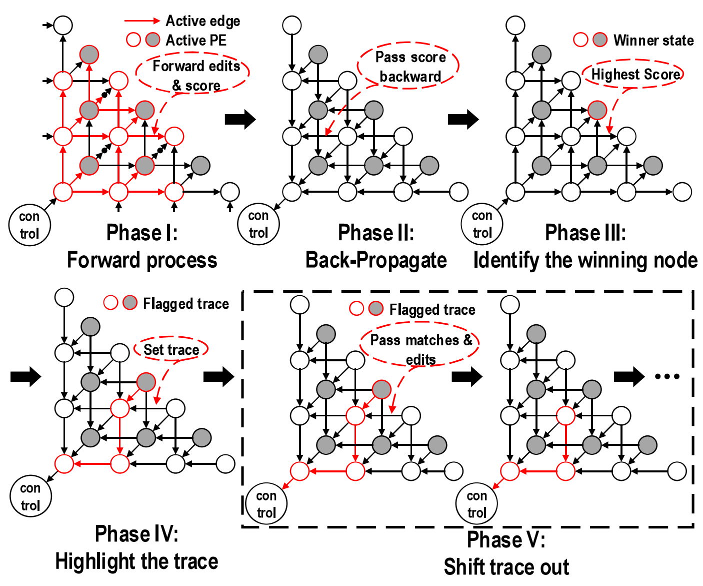

Fig. 8 Process Sequence of the Proposed Accelerator

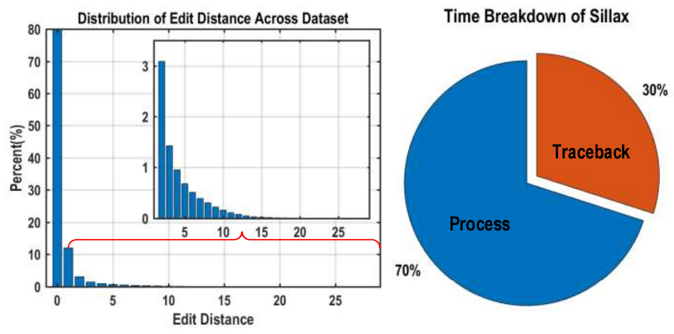

Fig. 9 Edit Distance Across Test Dataset (Left), and Average Time Breakdown of Processing(Right)

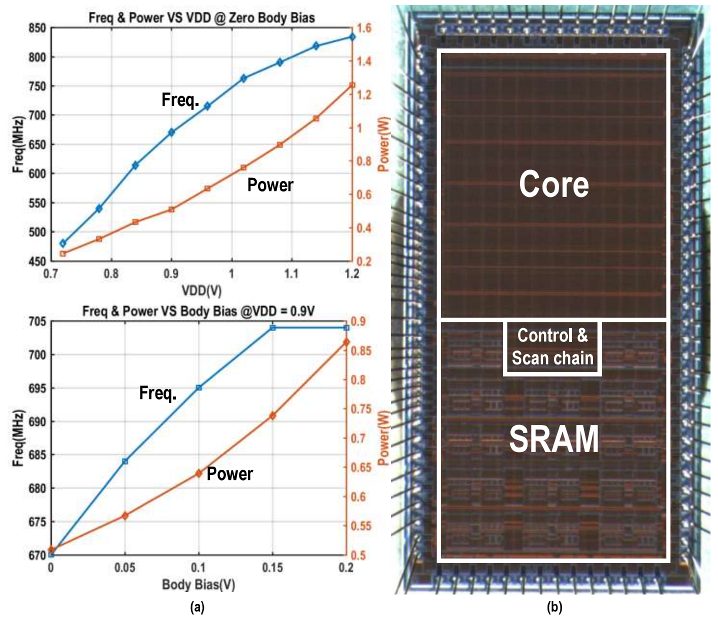

Fig. 10 Performance /Power vs. VDD & vs. Body Bias (Left), and Die Photo(Right)

III. Results

Implemented in Mie Fujitsu 55 nm DDC technology, the test chip achieves 670 MHz core clock frequency at 0.9 V VDD and consumes 508 mW of power. The frequency can scale up to 834 MHz with 1.2 V VDD and 704 MHz with 0.2 V forward body bias at 0.9V VDD (Fig. 10). The measured throughput is 2.46 million reads per second (MRPS) on the Illumina Platinum Genomes ERR194147 dataset with identical results as the software implementation of BWA-MEM. This marks an improvement of ~1000x improvement over a single core Xeon E5420 CPU [1] and ~10x improvement over a GPU [5] and FPGA [6], all of which have much larger silicon footprint yielding a performance normalized by area and technology of > 1000x. Power efficiency is 4.24 MRPS/W also marking a > 1000x improvement over BWA-MEM on the CPU and 70x improvement on the FPGA [6]. A comparison table is shown in Fig. 11.

	This work	CPU Implementation [1]	GPU Implementation [5]	[6]	[7]
Silicon Result	Yes	Software	Software	FPGA	FPGA
Technology	55nm DDC	45nm	28nm	40nm	40nm
Chip Size(mm²)	8	214	551	N/A	N/A
Freq.(MHz)	670	2500	706	200	200
Power(W)	0.508	80	250	4.7	4.7
Throughput* (MRPS)	2.46	0.0013	0.25	0.279	0.032
Power Efficiency (MRPS/W)	4.24	0.00002	0.001	0.059	0.027
Area Efficiency** (MRPS/mm²)	0.3075	0.000004	0.00011	N/A	N/A

* Normalized to single core or single PE array
** Normalized by technology node

Fig. 11 Comparison Table

Acknowledgment

We would thank MIFS for chip fabrication.

References

[1] Y. Wu, J. Hung and C. Yang, "14.8 A 135mW fully integrated data processor for next-generation sequencing," ISSCC 2017, pp. 252-253, 2017.

[2] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows–Wheeler Transform," Bioinformatics, vol.25, no.14, pp.1754-1760, 2009.

[3] D. Fujiki, A. Subramaniyan, T. Zhang, Y. Zeng, R. Das, D. Blaauw and S. Narayanasamy, "GenAx: A Genome Sequencing Accelerator," ISCA, pp. 69-82, 2018.

[4] O. Gotoh, "Optimal Sequence Alignment Allowing For Long Gaps," Bulletin of Mathematical Biology, vol. 52, no. 3, pp. 359–373, 1990.

[5] R. Wilton, T. Budavari, B. Langmead, S. J. Wheelan, S. L. Salzberg and A. S. Szalay, "Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space," PeerJ 3:e808, doi 10.7717/peerj.808, 2015.

[6] J. Arram, K. H. Tsoi, W. Luk and P. Jiangy, "Reconfigurable Acceleration of Short Read Mapping," ISFPCCM, pp.210-217, 2013.

[7] E. J. Houtgast, V. Sima, K. Bertels and Z. Al-Ars, "An FPGA-Based Systolic Array to Accelerate the BWA-MEM Genomic Mapping Algorithm," SAMOS, pp. 221-227, 2015.