A 2.46M Reads/s Seed-Extension Accelerator for Next-Generation Sequencing Using a String-Independent PE Array

Zhehong Wang, *Graduate Student Member, IEEE*, Tianjun Zhang, Daichi Fujiki, *Student Member, IEEE*, Arun Subramaniyan, *Graduate Student Member, IEEE*, Xiao Wu, *Member, IEEE*, Makoto Yasuda, Satoru Miyoshi, Masaru Kawaminami, Reetuparna Das, *Member, IEEE*, Satish Narayanasamy, *Member, IEEE*, and David Blaauw, *Fellow, IEEE*

***Abstract*— Advances in DNA-sequencing technology have far outpaced Moore's law, imposing significant challenges on the computationally intensive secondary analysis in the sequencing pipeline. An accelerator for seed extension, a critical and computationally intensive step in genome sequencing, is proposed. The accelerator, implementing a string-independent automata, consists of a triangular array of 25×2 custom-designed processing elements. It performs 2.46 million reads per second (MRPS), achieving a 1581× improvement in power efficiency and 165.5× smaller silicon footprint compared to a system setting with dual-socket Xeon E5-2697 v3 server processors.**

***Index Terms*— Application-specified integrated circuit (ASIC), DNA sequencing, Levenshtein edits, seed-extension, string-independent automata.**

I. Introduction

THE completion of Human Genome Project (HGP) [1] has triggered immense interest in the application of whole genome sequencing (WGS), driving the development of next-generation-sequencing (NGS) technology to reduce the cost. Since the advance of NGS, the production cost of whole human genome sequencing has plummeted by 10 000× from U.S. $10 million to U.S. $1000 in the last decade [2], as in Fig. 1. This has led to wide use of DNA testing in both research and clinical diagnosis, creating more personalized patient treatments [3]. For example, genetic tests are now commonly used to predict the effectiveness of specific breast cancer treatments for patients [4]. Furthermore, identifying somatic mutations in the human genome sheds light on the evolution of human cancers, information that can be leveraged to prevent these cancers in individuals [5]. Likewise, large-volume genome testing across diverse samples can provide a better understanding of the cause of Alzheimer disease [6]. As technology speeds up WGS, its use will likely become a standard clinical practice, similar to a blood test, in the coming decade.

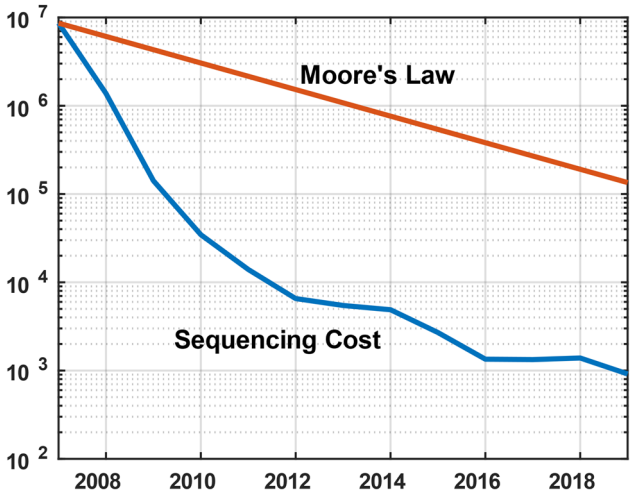

Fig. 1. Sequencing cost plummeting versus Moore's law [2].

The fast reduction of sequencing cost that benefits the ubiquitous application of WGS stems from the rapid improvement of the NGS technology. Compared to HGP, which required 15 years to sequence the first human genome, NGS systems from Illumina can sequence over 45 human genomes in a single day [7], rendering a 200 000× speedup. As shown in Fig. 2, a typical DNA sequencing pipeline of current generation is composed of roughly two main steps. First, a DNA sequencer, backed by NGS technology, splits the input genome into billions of reads, which are short DNA copies of the input genome. Then, the reads are passed into a reference-guided

Manuscript received May 30, 2020; revised July 26, 2020; accepted September 2, 2020. Date of publication September 22, 2020; date of current version February 24, 2021. This article was approved by Guest Editor Mark Oude Alink. *(Corresponding author: Zhehong Wang.)*

Zhehong Wang, Daichi Fujiki, Arun Subramaniyan, Xiao Wu, Reetuparna Das, Satish Narayanasamy, and David Blaauw are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: zhehongw@umich.edu).

Tianjun Zhang was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. He is now with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720 USA.

Makoto Yasuda is with Mie Fujitsu Semiconductor Ltd., Kuwana 511-0118, Japan.

Satoru Miyoshi is with Fujitsu Electronics America, Inc., Sunnyvale, CA 94085 USA.

Masaru Kawaminami is with Mie Fujitsu Semiconductor Ltd., Yokohama 222-0033, Japan.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2020.3023822

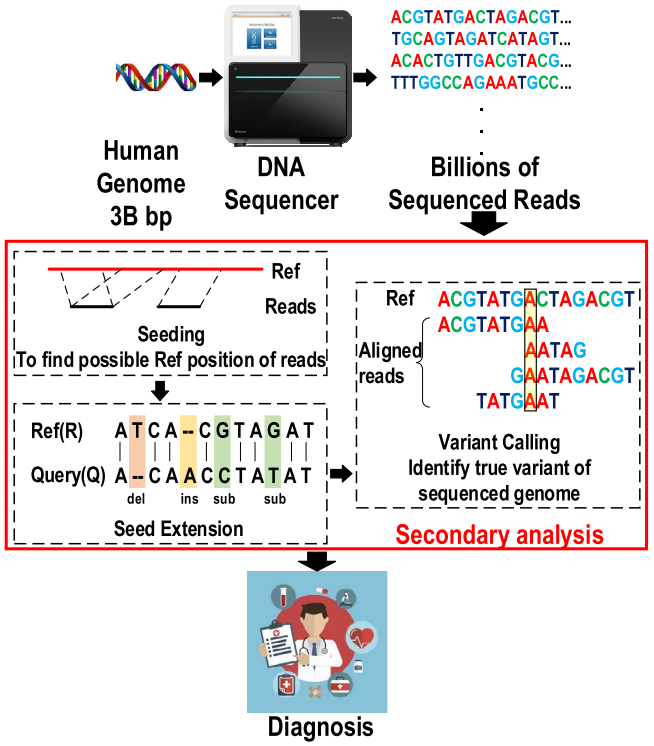

Fig. 2. Reference-guided sequence analysis pipeline with seed extension highlighted.

secondary analysis, where they are assembled into the original genome by aligning to a previously sequenced genome, for example, the one obtained from the HGP. The secondary analysis is far from a simple string comparison in the sense that the input genome does not necessarily match the referenced genome due to DNA mutations, requiring approximate string alignment. Furthermore, the produced reads are not the exact copy of the input genome due to the errors introduced by the DNA sequencer, further complicating the task. Altogether, for each sequenced human genome, on average, 396 GB of data must be processed through the complex secondary analysis [8], which not only poses a significant computational challenge to current general-purpose computing systems but also presents them as a bottleneck in the sequencing pipeline, since the improvement of the DNA sequencer enabled by NGS technology is far outpacing Moore's law. At such rate, this computation bottleneck is projected to dominate the total cost and processing time of sequencing, becoming a key limiting factor in the growth of this important medical technology.

Before dealing with the bottleneck, it is worthwhile taking a close look at the secondary analysis (see Fig. 2). As mentioned above, a large number of reads, ∼1.5 B, produced by the DNA sequencer, are passed through the secondary analysis, which is further divided into three processing steps [9].

1) In the seeding step, a set of possible locations where the read matches the reference is found by exactly matching small fragments (seeds) between the read and the reference.
2) The seed-extension step evaluates these possible match locations by exploring approximate alignments between the read and the reference, including possible edits, to determine the final match location.
3) In the final step, the variant calling step, all the reads that are aligned to a particular base in the reference are evaluated to determine whether a mutation occurred at that location.

Various software packages have been devised to handle these steps since early twenty-first century. Some of them cover the first two steps, such as BWA-MEM [10], Bowtie2 [11], and SOAP2 [12], while some focus on the third step but use the above libraries for the first two steps, like GATK [13] that incorporates BWA-MEM. There are also other generic string/sequence analysis libraries such as SeqAn [14] and SSW [15]. Among them, BWA-MEM has become a standard for the analysis pipeline and is advocated by GATK best practice guide.

BWA-MEM, as well as other libraries, adapts two algorithms, FM-Index [16], based on Burrows–Wheeler transform (BWT) [17], and Smith–Waterman (SW) algorithm [18], for the two steps, seeding and seed extension, respectively. GPU implementations [19], [20] of these software packages and algorithms were released within the software community to cope with the computation bottleneck due to the increasing amount of data to be processed. However, acceleration for them gained little interest until recent years, following the tapering-off of Moore's law. Field-programmable gate array (FPGA) solutions were proposed at first for its versatility and better power efficiency compared to GPU systems, such as [21] for the seeding step and [22] and [23] for the seed-extension step. With the software stack getting stable, application-specified integrated circuit (ASIC) becomes appealing in terms of form factor and power efficiency. An ASIC designed for the seeding step was also presented in ISSCC 2018 [24] with 7.84-mm^2 die size and 135 mW of power, achieving ∼1000× and ∼400× improvement in terms of power efficiency and area efficiency compared to GPU implementation. However, little or no dedicated acceleration ASIC has been proposed to address other steps in DNA sequencing.

In this article, we target at the seed-extension step, which requires a total of 14 billion alignments for each human genome, which takes ∼5/∼273 h for 56-thread/single-thread workload on a server equipped with dual-socket Xeon E5-2697 v3 processors, using the optimized SeqAn library [25]. Seed extension aligns two DNA strings of ∼100 bases: the read or query (Q) and the portion of the reference (R) where the query is expected to align. However, as mentioned above, there can be mismatches between R and Q due to sequencing machine errors or mutations in an individual's DNA. Hence, approximate alignment is needed, allowing for the following Levenshtein edits as shown in Fig. 3: insert (i), delete (d), and substitute (s). Fig. 4 shows two of many possible alignments for an R and Q pair, each with an edit distance or score. The goal of seed extension is to find the alignments with the best score and to report the score and its associated strings of edits.

We use a 25 × 25 triangular array of processing elements (PEs) that implements a string-independent automata

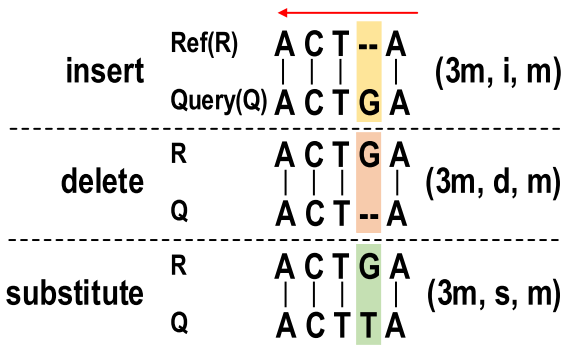

Fig. 3. Levenshtein edits.

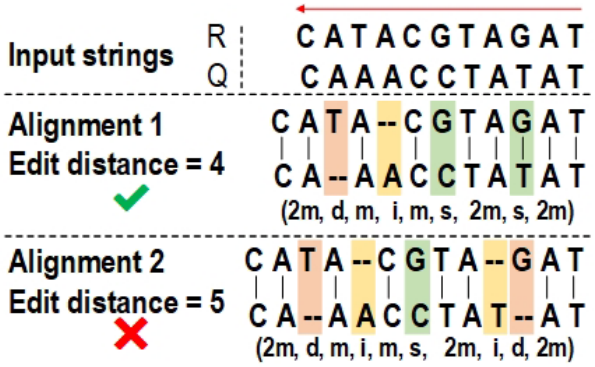

Fig. 4. Example of alignment.

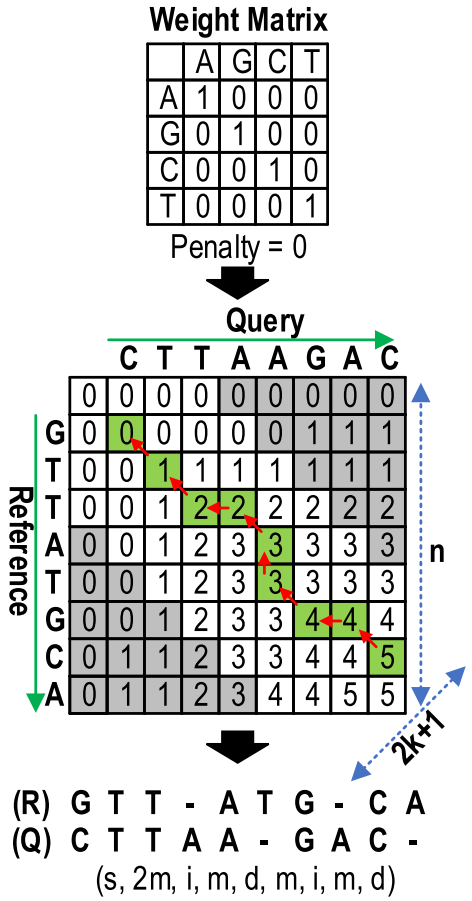

Fig. 5. Basic SW algorithm.

algorithm 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 deeply depleted channel (DDC) CMOS, operates at 670 MHz and achieves 2.46 million reads per second (MRPS) with 8-mm^2 silicon area [26]. Marking, to our knowledge, the first seed-extension ASIC, it achieves an $\sim$1581$\times$ power efficiency improvement and 165.5$\times$ smaller silicon footprint compared to deploying SeqAn library on a server with dual-socket Xeon E5-2697 v3 server processors [25], operating on the same Genome data set and producing the same output.

The remainder of this article is organized as follows. Section II introduces the core operation of the proposed accelerator and compares it with the standard SW algorithm. Section III describes the detailed implementation of the test chip. Section IV presents the measurement results of the design, and finally, Section V concludes this article.

II. Seed-Extension Algorithms

A. SW Algorithm

The canonical solution for the approximate string alignment problem is a dynamic programming (DP) algorithm called the SW algorithm [18], which is adapted as a submodule of BWA-MEM and other software libraries. The basic version of the algorithm that only identifies the edit distance is illustrated in Fig. 5.

To align two input strings' query (Q) and reference (R), first a DP matrix of size n^2 is initialized as all zeros, where n is the length of the input strings (assuming w.l.o.g. the input strings are of the same length). Then all the cells identified by coordinate (i, j), except for the ones in the first row and column, are filled by an alignment score calculated from the neighboring cells, according to (1). The weight matrix denoted by $W(r_i, q_j)$ in (1) assigns each pair of the input characters in the input strings a score for match or amismatch. For example, here, the match score is set to 1 and mismatch score is set to 0. The penalty in the equation is reserved for the gap penalty calculation explained further in Section III and is set to 0 for simplicity in this example. Thus, (1) is a rule of reward and penalty that accumulates the highest alignment score between the two input strings up to position (i, j).

In the meantime, each cell also keeps track of which one of the neighboring cells its score is calculated from. Finally, from the highest scores, the traces that hold the best scores are extracted backward. One of the best traces is shown in the example, which translates to an edit distance that equals 5

$$E[i, j] = \max \begin{cases} E[i-1, j-1] + W(r_i, q_j) \\ E[i, j-1] - \text{penalty} \\ E[i-1, j] - \text{penalty}. \end{cases} \quad (1)$$

The time and space complexity of the SW algorithm is $O(n^2)$, where n is the maximum length of the input string pair, rendering a string-dependent space complexity and preventing efficient hardware implementation. In practice, one

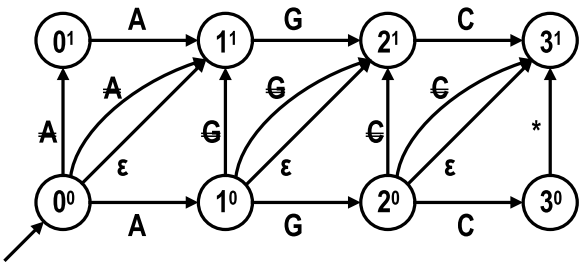

Fig. 6. State diagram of an LA for sequence s = AGC and edit distance $k = 1$.

often focuses, dynamically or statically, on a band of fixed width along the diagonal of the DP matrix [27], leaving out the corners colored in gray in Fig. 5. The parameter that sets the width of the band is the maximum target edit distance k. This banded DP matrix correctly generates output as long as the edit distance of the input string pair is no greater than k. This optimization potentially reduces the complexity to $O(kn)$ since k is much less than n in practice. However, it is still string dependent.

B. String-Independent Automata

Another solution to sequence matching is based on Levenshtein automata (LA). An LA is a finite-state automaton that is defined on a sequence s and a number k. It can recognize the set of all sequences that are at most k edit distances away from the string s. For example, the state diagram of an LA, defined on string automatic gain control (AGC) and edit distance $k = 1$, is shown in Fig. 6. Each state is denoted by n^e, which means that n characters consumed and e edit distance encountered so far. The state machine starts from state 0° and takes the characters of input sequence one by one. Also, it determines whether the input sequence is at most $k = 1$ edit distance away from the sequence s = AGC. Discussion about the finite-state automaton theory is beyond the scope of this work, and the reader is referred to [28] for further information. Although the complexity of LA is $O(kn)$, which is comparable to SW, it suffers from the fact that it is specific to a given sequence, as the LA in Fig. 6 is only able to compare strings to AGC. Thus, LA is rarely utilized in practical sequencing software libraries.

However, our design adopts a recently proposed string-independent LA algorithm [25], the algorithm decouples the state machine from the specific sequence and has the advantage that varying length strings can be processed using the same matching hardware as long as the maximum edit distance remains fixed. Unlike the standard SW algorithm, where R/Q strings remain static and possible alignment paths are explored with an array of PEs, the R/Q strings of arbitrary length are shifted through a state machine that simultaneously evaluates all possible alignments between the two strings.

The state machine for this algorithm consists of a 3-D grid of tiny PEs, represented as circles in Fig. 7, each performing a comparison of a base pair (BP), while the BPs of the input R/Q reads are supplied through the two shift registers,

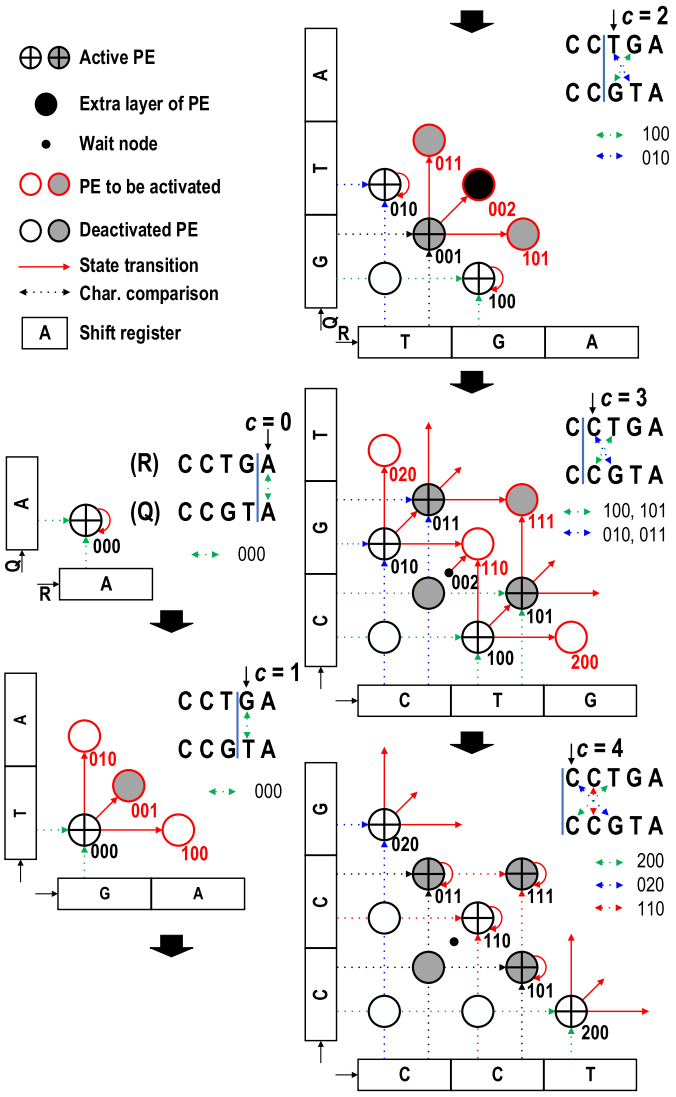

Fig. 7. String alignment example.

rightward and upward, one BP at a time. Starting from the PE at the bottom left, each PE reactivates itself when the BP matches, otherwise, it simultaneously activates the neighboring PEs in a 3-D fashion. The 3-D represent insertion, deletion, and substitution, thus, each PE is uniquely assigned a state identifier (i, d, s) corresponding to the edit cost seen so far, e.g., PE120 corresponds to $1i + 2d + 0s$. So, the notations "PE" and "state" are referred interchangeably in the rest of this article. In summary, the process propagates from PE000 diagonally into the 3-D until the input R/Q reads are shifted out of the registers.

The example in Fig. 7 performs as follows. Initially, in clock cycle zero ($c = 0$), only PE000 is activated and compares the first BP of the R/Q strings (A, A). For clarity, only the activated PEs are shown in Fig. 7. Since the BP matches, the PE reactivates itself, indicating a match (m) edit string so far. In the next clock cycle ($c = 1$), 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. PE100 evaluates

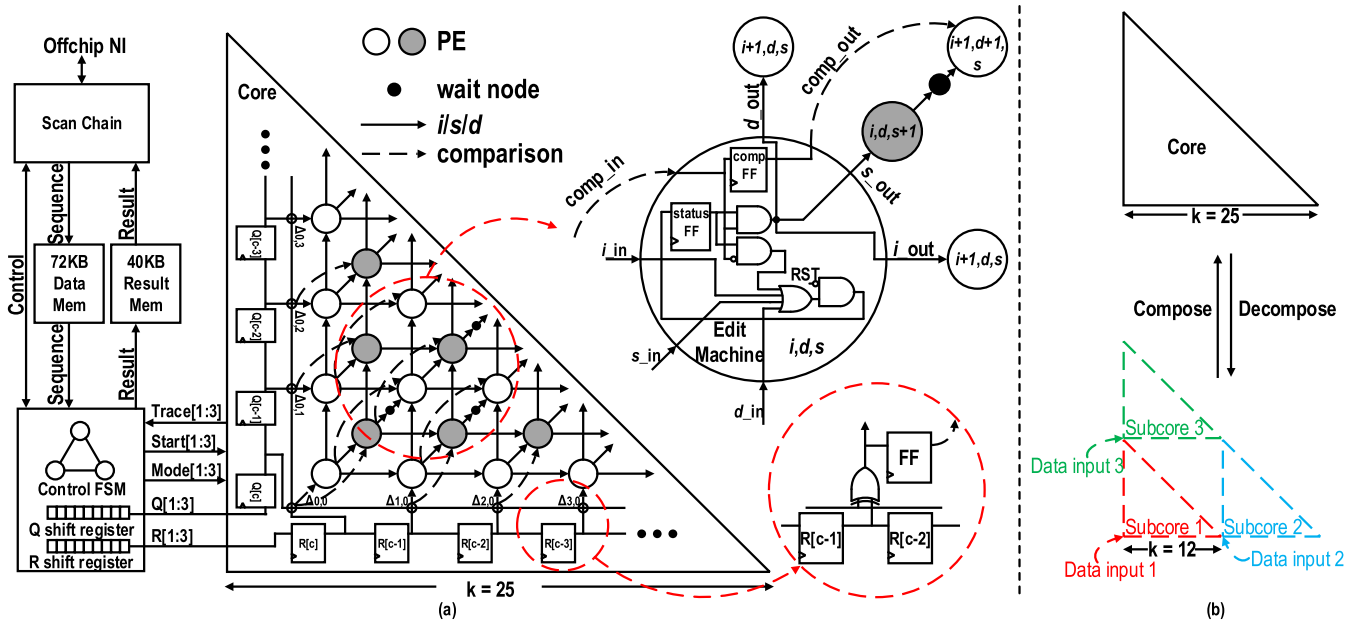

Fig. 8. (a) Overall architecture of test chip and details. (b) Composable structure.

possible alignment of R/Q after $1i$ and, therefore, uses the shifted base 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 bases of its inputs and compares BP (T, G), which is a mismatch. It, therefore, deactivates itself and 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 distances are not the same, preventing them from being merged. Similarly, PE010 and PE011 compare BP (C, G) and find a mismatch. PE002 represents 2s and compares BP (C, C), which is a match. However, if this pattern is followed, a full 3-D grid of PEs develops, as shown by the large black circle in $c = 2$, which 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, since the R and Q strands are shifted right and up, respectively, in $c = 4$, PE110 also compares BP (C, C) in $c = 4$. Furthermore, in terms of edit distance, $i = d = s = 1$, so the edit distances represented by both PE002 and PE110 are equal ($1i + 1d = 2s$). Hence, they can be merged by replacing PE002 with a wait node that does nothing but activates PE110 in $c = 4$ (indicated by the small black dot in $c = 3$ and 4). By offloading the evaluation of PE002 to PE110 in $c = 4$, a full 3-D structure is collapsed into a 2-D structure. Hence, in $c = 4$, PE110 finds a matched BP (C, C) and after reactivating itself, again finds the final match (C, C) in $c = 5$ (not shown in Fig. 7), marking the optimal alignment of two edits. Note that several other PEs are also active in $c = 4$. However, they all have higher edit distances and are suboptimal. Finally, the process ends after both the input strings are shifted out of the shift registers.

Therefore, the resulting array has dimension $(k, k, 2)$. Compared to SW, the space complexity of the automata array, $O(k^2)$, is only quadratic in terms of the maximum edit distance k, making the size of the array much smaller and string independent. On the other hand, the process starts with R/Q shifted into the register and finishes with them shifted out, resulting in an additive linear time complexity $O(n+k)$ instead of multiplicative time complexity $O(kn)$ in the banded SW. Thus, the string-independent automata array renders a higher throughput for a given space.

III. Seed-Extension Accelerator

A. Implemented Architecture

The overall architecture of the accelerator and PE array is shown in Fig. 8(a). The PEs are extremely simple, consisting of only six gates, which OR incoming activations and activate self/neighbors depending on the comparison result. The BP comparisons are performed at the shift registers and are then passed diagonally to neighboring PEs since diagonal PEs use the same comparison result with a one-cycle delay as explained in Section II. This makes all communication local, allowing for very high-speed operation. The 25×25 triangular array can be decomposed into three smaller triangular arrays of 12×12 [see Fig. 8(b)], enabling a tradeoff between throughput and maximum edit distance, which is further discussed in Section IV.

B. Affine Gap Penalty and Score Machine

The standard seed-extension algorithm typically uses a more sophisticated scoring scheme in addition to edit distance, based on empirical statistics [29]. This includes the affine gap penalty used in standard BWA-MEM software [10], complying

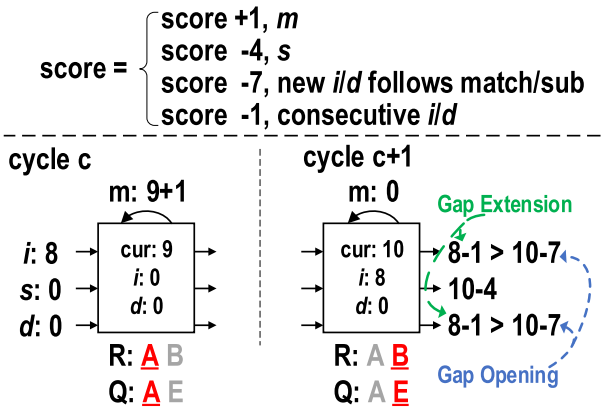

Fig. 9. Score scheme (top) and delayed merging (bottom).

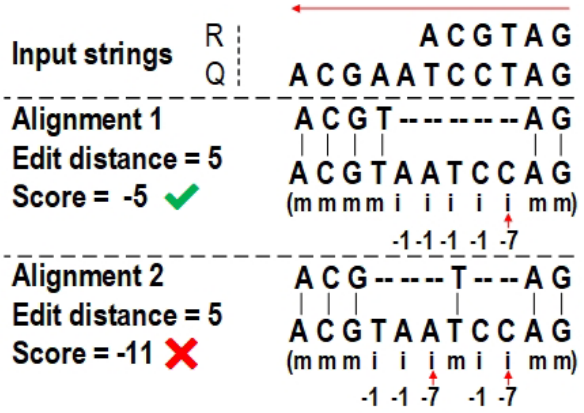

Fig. 10. Affine gap scoring example.

to the clipping heuristic. Accordingly, the scoring scheme, as shown in the top of Fig. 9, is adopted in our design. The penalty scheme favors consecutive insertions and deletions (penalty -1) over new insertions and deletions (penalty -7), preventing merging confluence paths, which can occur with simple edit distance scoring.

According to this rule, two sample alignments (see Fig. 10) have the same edit distance (five insertions) but with different scores. Alignment 2 has two gaps separated by the matched T and T BP in the middle, which imposes a doubled gap penalty compared to alignment 1 (two -7 penalties versus one), making the first alignment more desirable. To support the affine gap penalty, a delayed merge of two converging paths is required as in Fig. 9. When two paths converge, they cannot merge immediately based on the current state and input scores alone since the future score depends on whether the path opens a new gap. In the example, although the incoming score from the preceding insertion edge, 8, is lower than the current score, $9+1$, the incoming score is not discarded immediately to merge the two paths. Instead, we latch it until a new gap opens in the next cycle to decide whether to take the incoming path and pass the higher score to the following PEs. Score calculation logic is introduced in addition to the basic PE to accommodate the aforementioned scheme, which passes the calculated score from PE to PE along with the state activations (see Fig. 11).

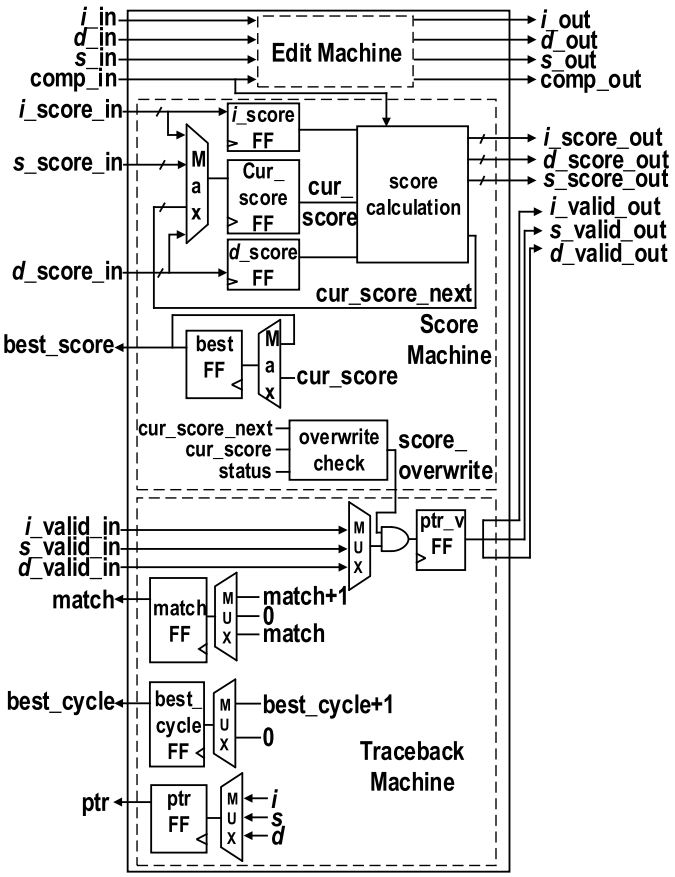

Fig. 11. PE augmented with score machine and traceback machine.

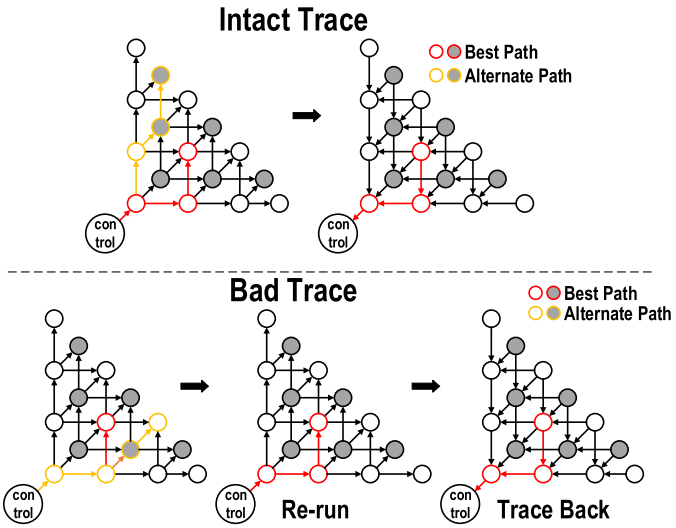

Fig. 12. Traceback of intact trace (top) and bad trace (bottom).

C. Collision Resolution and Traceback Machine

After the forward process of the input read pair as described above, the best trace is then shifted backward and collected by the controller to generate the final output (Fig. 12, top). This in-place trace back is supported by augmenting the edit machine with a traceback machine and a score machine as shown in Fig. 11. The principle is to keep track of the best

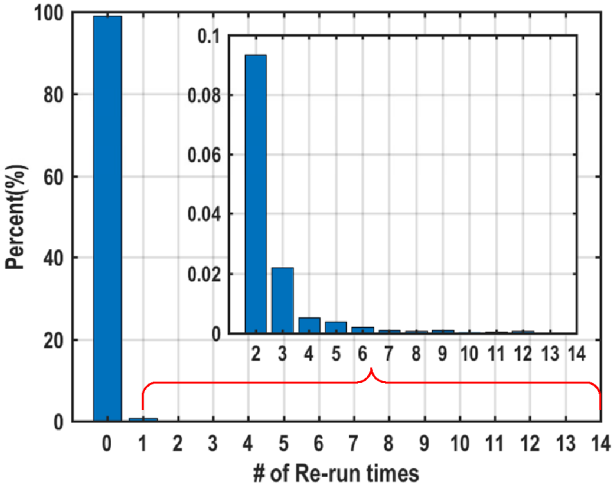

Fig. 13. Distribution of Re-run times across data set.

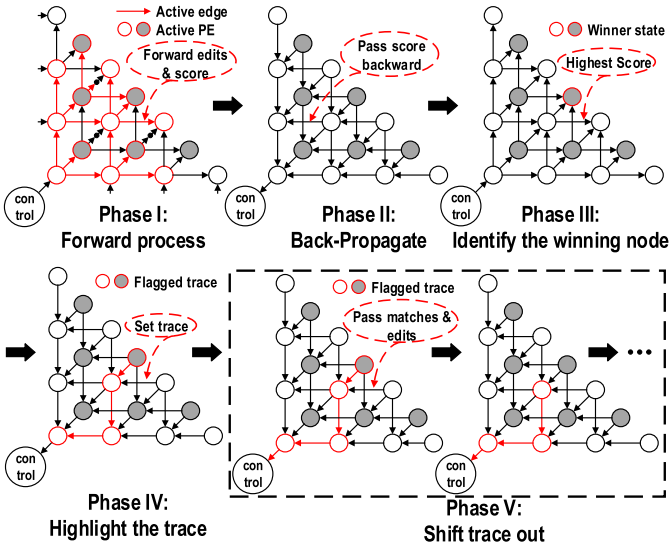

Fig. 14. Process sequence of the proposed accelerator.

score and the incoming state pointer (i, d, s) that activates current state with that best score and also to count the matches that the current state meets before activating the next state. With this information, the traceback machine chains the winning states one-by-one during the backward propagation to form a complete winning path. However, during the forward process, it is possible for the correct pointer to be corrupted by a later, non-optimal path due to greedy affine gap scoring, thus breaking the backward trace of the best path. When such a collision is detected by the controller, the string is reprocessed up to the point when the broken state occurs and then traced back, as shown in Fig. 12, bottom. Although this requires reprocessing the string, in practice, this is rare, less than 1% of the reads (see Fig. 13), resulting in negligible performance degradation.

D. Complete Operating Sequence

Fig. 14 shows the complete sequence of operation: after the full read pair is processed (phase I), each PE passes

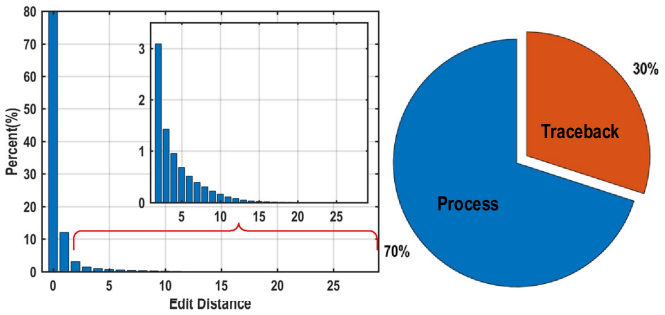

Fig. 15. Edit distance across the test data set (left) and the average time breakdown of the processing (right).

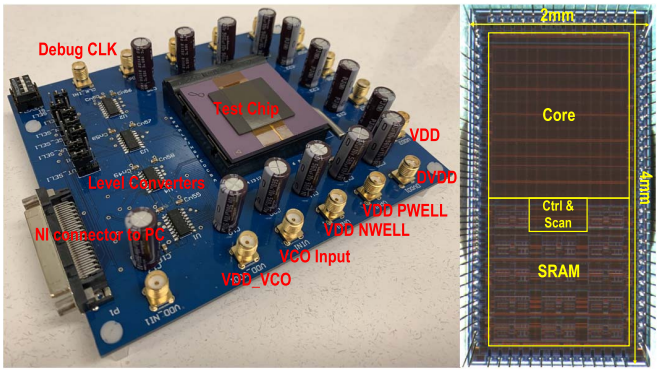

Fig. 16. Test board and die photograph.

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 until they reach PE000. Finally, in phase V, this backward trace is collected by shifting it back to PE000, revealing the edit string.

IV. Measurement

A representative data set, the Illumina Platinum Genomes ERR194147 data set [30], is used to test our implementation. The distribution of edit distance of the reads in the data set is shown in Fig. 15, left. From the distribution, 99.9967% of the reads are of edit distance no greater than 25, so with $k = 25$ for the PE array, 99.9967% of the reads are correctly processed. Since the size of the array is quadratic in maximum edit distance supported, 25 is a good tradeoff point between accuracy and silicon area. For the decomposed mode, where $k = 12$, 99.7907% of the reads are correctly covered. A time breakdown of the processing on the data set (Fig. 15, right) shows that 70% of the processing time is devoted to forward process and 30% is taken by the traceback phase.

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. Fig. 16 shows

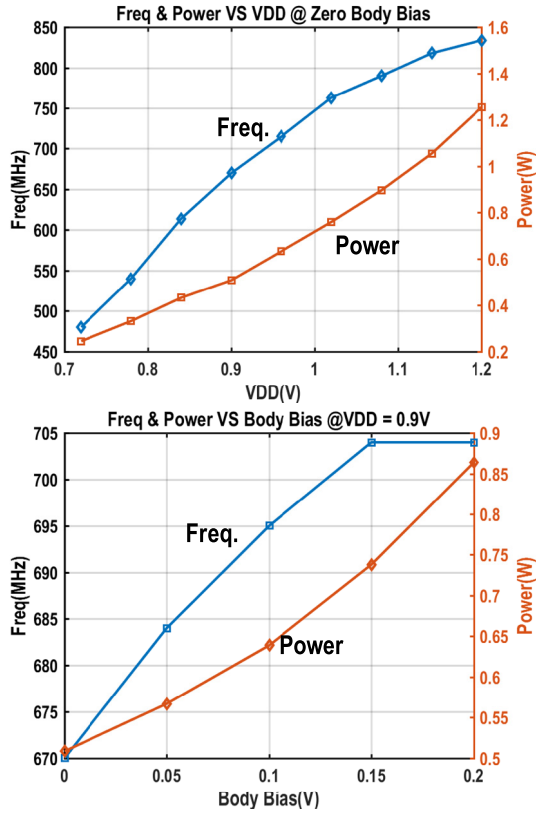

Fig. 17. VDD scaling and body bias scaling plot.

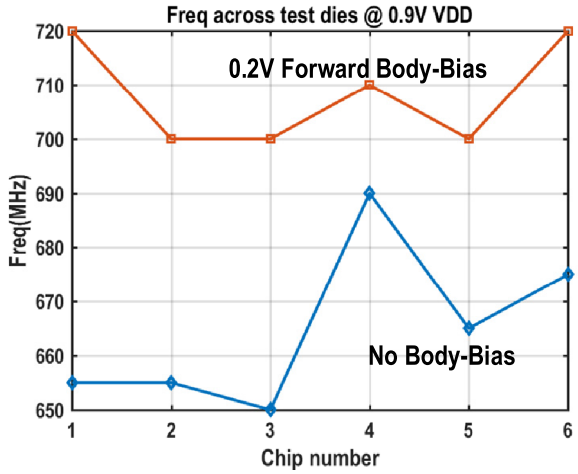

Fig. 18. Frequency distribution across test chips.

the test board and die photograph. As shown in the VDD scaling plot (see Fig. 17), the frequency of the test design can scale up to 834 MHz with 1.2-V VDD and 704 MHz with 0.2-V forward body bias at 0.9-V VDD. Fig. 18 gives a frequency distribution across six different test chips.

To compare this work with the state-of-the-art, BWA-MEM is chosen as the ground truth to compare the output due to its popularity. However, because BWA-MEM also incorporates the seeding step, the SeqAn library is implemented to study the performance of this work independent of the seeding step. The measured throughput is 2.46 MRPS for the test data set, and the results are identical to those obtained with BWA-MEM under the same settings of maximum edit distance k and scoring scheme. This marks a performance improvement of $\sim 3\times/\sim 170\times$ over SeqAn library deployed as 56-thread/single-thread workload on a server with dual-socket Xeon E5-2697 v3 processors under the same configuration [14], [25] and $\sim 32\,000\times$ over SSW library on a 2-GHz AMD processor [15]. It also achieves improvement of $\sim 10\times$ over a GPU implementation [20] or an FPGA implementation [22], all of which have a much larger silicon footprint, yielding a performance improvement normalized by area and technology of over $1000\times$. The power efficiency is 4.24 MRPS/W, also marking an over $1000\times$ improvement over SeqAn on the CPU [25] and $70\times$ improvement on the FPGA [22]. Table I summarizes the above comparison.

TABLE I
COMPARISON WITH OTHER WORK

	This work	CPU [14][25]	GPU [20]	[22]	[23]
Silicon Result	Yes	Software	Software	FPGA	FPGA
Technology	55nm DDC	22nm	28nm	40nm	40nm
Chip Size(mm^2)	8	662x2	551	N/A	N/A
Freq.(MHz)	670	2600	706	200	200
Power(W)	0.508	145x2	250	4.7	4.7
Throughput* (MRPS)	2.460	0.014	0.250	0.279	0.032
Power Efficiency (MRPS/W)	4.24	0.003	0.001	0.059	0.027
Area Efficiency** ($MRPS/mm^2$)	0.3075	0.000096	0.00011	N/A	N/A

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

V. CONCLUSION

An accelerator for seed extension is presented. The accelerator consists of a triangular array of 25×25 custom-designed PEs, implementing a string-independent automata. Marking the first silicon implementation of the string-independent matching algorithm, it achieves 2.46-MRPS throughput and 4.24 MRPS/W power efficiency, providing $1581\times$ power efficiency and over $1000\times$ area efficiency improvement compared to deploying SeqAn library on dual-socket Xeon E5-2697 v3 server processors, while maintaining the same output as standard BWA-MEM library under the same configuration.

REFERENCES

[1] E. S. Lander *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, pp. 860–921, Feb. 2001, doi: 10.1038/35057062.
[2] K. A. Wetterstrand. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)*. Accessed: May 20, 2020. [Online]. Available: https//www.genome.gov/sequencingcostsdata
[3] A. Grada and K. Weinbrecht, "Next-generation sequencing: Methodology and application," *J. Investigative Dermatology*, vol. 133, no. 8, pp. 1–4, Aug. 2013, doi: 10.1038/jid.2013.248.

[4] M. A. Hamburg and F. S. Collins, "The path to personalized medicine," *New England J. Med.*, vol. 2010, no. 363, pp. 301–304, 2010.
[5] E. D. Pleasance *et al.*, "A comprehensive catalogue of somatic mutations from a human cancer genome," *Nature*, vol. 463, no. 7278, pp. 191–196, 2010.
[6] A. Lacour *et al.*, "Genome-wide significant risk factors for Alzheimer's disease: Role in progression to dementia due to Alzheimer's disease among subjects with mild cognitive impairment," *Mol. Psychiatry*, vol. 22, no. 1, pp. 153–160, Jan. 2017, doi: 10.1038/mp.2016.18.
[7] *An Introduction to Next-Generation Sequencing Technology, Illumina, Inc*. Accessed: Jun. 15, 2020. [Online]. Available: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_in troduction.pdf
[8] Z. D. Stephens *et al.*, "Big data: Astronomical or genomical," *PLoS Biol.*, vol. 13, no. 7, Art. no. e1002195, 2015.
[9] G. A. Auwera *et al.*, "From FastQ data to high-confidence variant calls: The genome analysis toolkit best practices pipeline," *Current Protocols Bioinf.*, vol. 43, no. 1, p. 11, Oct. 2013, doi: 10.1002/0471250953.bi1110s43.
[10] H. Li and R. Durbin, "Fast and accurate long-read alignment with burrows-wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, 2010, doi: 10.1093/bioinformatics/btp698.
[11] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with bowtie 2," *Nature Methods*, vol. 9, no. 4, pp. 357–359, Apr. 2012, doi: 10.1038/nmeth.1923.
[12] R. Li *et al.*, "SOAP2: An improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, Aug. 2009, doi: 10.1093/bioinformatics/btp336.
[13] A. McKenna *et al.*, "The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Res.*, vol. 20, no. 9, pp. 1297–1303, Sep. 2010, doi: 10.1101/gr.107524.110.
[14] K. Reinert *et al.*, "The SeqAn C++ template library for efficient sequence analysis: A resource for programmers," *J. Biotechnol.*, vol. 261, pp. 157–168, Nov. 2017, doi: 10.1016/j.jbiotec.2017.07.017.
[15] M. Zhao, W.-P. Lee, E. P. Garrison, and G. T. Marth, "SSW library: An SIMD smith-waterman C/C++ library for use in genomic applications," *PLoS ONE*, vol. 8, no. 12, Dec. 2013, Art. no. e82138, doi: 10.1371/journal.pone.0082138.
[16] P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, Redondo Beach, CA, USA, Nov. 2000, pp. 390–398, doi: 10.1109/SFCS.2000.892127.
[17] M. Burrows and D. J. Wheeler, "A block sorting lossless data compression algorithm," Digit. Equip. Corp., Maynard, MA, USA, Tech. Rep. 124, 1994.
[18] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.
[19] Y. Liu and B. Schmidt, "CUSHAW2-GPU: Empowering faster gapped short-read alignment using GPU computing," *IEEE Des. Test. IEEE Des. Test. Comput.*, vol. 31, no. 1, pp. 31–39, Feb. 2014, doi: 10.1109/MDAT.2013.2284198.
[20] 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*, vol. 3, p. e808, Mar. 2015, doi: 10.7717/peerj.808.
[21] H. M. Waidyasooriya and M. Hariyama, "Hardware-acceleration of short-read alignment based on the burrows-wheeler transform," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1358–1372, May 2016, doi: 10.1109/TPDS.2015.2444376.
[22] J. Arram *et al.*, "Reconfigurable acceleration of short read mapping," in *Proc. IEEE 21st Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, Seattle, WA, USA, Apr. 2013, pp. 210–217, doi: 10.1109/FCCM.2013.57.
[23] E. J. Houtgast, V.-M. Sima, K. Bertels, and Z. Al-Ars, "An FPGA-based systolic array to accelerate the BWA-MEM genomic mapping algorithm," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling, Simulation (SAMOS)*, Jul. 2015, pp. 221–227, doi: 10.1109/SAMOS.2015.7363679.
[24] Y.-C. Wu, J.-H. Hung, and C.-H. Yang, "14.8 A 135 mW fully integrated data processor for next-generation sequencing," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 252–253, doi: 10.1109/ISSCC.2017.7870356.
[25] D. Fujiki *et al.*, "GenAx: A genome sequencing accelerator," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 69–82, doi: 10.1109/ISCA.2018.00017.
[26] Z. Wang *et al.*, "A 2.46M reads/s genome sequencing accelerator using a 625 processing-element array," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Boston, MA, USA, Mar. 2020, pp. 1–4, doi: 10.1109/CICC48029.2020.9075900.
[27] D. Okada, F. Ino, and K. Hagihara, "Accelerating the smith-waterman algorithm with interpair pruning and band optimization for the all-pairs comparison of base sequences," *BMC Bioinf.*, vol. 16, no. 1, p. 321, doi: 10.1186/s12859-015-0744-4.
[28] K. U. Schulz and S. Mihov, "Fast string correction with levenshtein automata," *Int. J. Document Anal. Recognit.*, vol. 5, no. 1, pp. 67–85, Nov. 2002, doi: 10.1007/s10032-002-0082-8.
[29] O. Gotoh, "Optimal sequence alignment allowing for long gaps," *Bull. Math. Biol.*, vol. 52, no. 3, pp. 359–373, May 1990.
[30] M. A. Eberle *et al.*, "A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree," *Genome Res.*, vol. 27, no. 1, pp. 157–164, Jan. 2017, doi: 10.1101/gr.210500.116.

Zhehong Wang (Graduate Student Member, IEEE) received the B.E. degree in electronics and information engineering from Zhejiang University, Hangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Michigan, Ann Arbor, MI, USA.

His current research interests include application-oriented application-specified integrated circuit (ASIC), such as DNA sequencing, machine learning, fully homomorphic encryption, and emerging memory design.

Tianjun Zhang received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, and Shanghai Jiao Tong University, Shanghai, China, in 2018. He is currently pursuing the Ph.D. degree in computer science with the University of California at Berkeley, Berkeley, CA, USA.

His current research interests include software–hardware co-design of deep neural networks and efficient training/inference architecture for machine learning.

Daichi Fujiki (Student Member, IEEE) received the B.E. degree from Keio University, Tokyo, Japan, in 2016 and the M.S.Eng. degree from the University of Michigan, Ann Arbor, MI, USA, in 2017. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Michigan, Ann Arbor, MI, USA.

He is advised by Prof. R. Das. He is a member of the Mbits Research Group, Computer Engineering Laboratory (CELAB), University of Michigan, which develops *in situ* compute memory architectures and custom acceleration hardware for bioinformatics workloads.

Arun Subramaniyan (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI, USA.

His research interests include in-memory computing and hardware acceleration for genomics.

Xiao Wu (Member, IEEE) received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, the B.S. degree in computer science from Shanghai Jiaotong University, Shanghai, China, in 2014, and the Ph.D. degree in electrical and computer engineering from the University of Michigan in 2019.

Her research includes small sensor node design and high-performance hardware accelerator for genome sequencing.

Makoto Yasuda received the B.S. degree in electronic physical engineering from Hiroshima University, Hiroshima, Japan, in 1992.

He joined Fujitsu Ltd., Kawasaki, Japan, in 1992, where he was a Process Integration Engineer. He is currently engaged in the development of deeply depleted channel (DDC) technology with Mie Fujitsu Semiconductor Ltd., Kuwana, Japan.

Satoru Miyoshi received the B.S. degree in material science from Yokohama National University, Yokohama, Japan, in 1988.

In 1988, he joined Fujitsu Ltd., Kawasaki, Japan, as a Process Integration Engineer. Since 2014, he has been the Technology Marketing Director with Fujitsu Electronics America, Inc., Sunnyvale, CA, USA, where he has been involved in the development of deeply depleted channel technology.

Masaru Kawaminami received the B.S. and M.S. degrees in science and engineering (applied chemistry) from Waseda University, Tokyo, Japan, in 2002.

He was a Technical Marketing Specialist with the Business Development Division, Fujitsu Electronics America, Inc., Sunnyvale, CA, USA, from 2015 to 2019. He is currently a Customer Support Engineer with the Technology Service Division, Mie Fujitsu Semiconductor Ltd., Yokohama, Japan, for expanding foundry businesses to overseas customers.

Reetuparna Das (Member, IEEE) received the Ph.D. degree in computer science and engineering from Pennsylvania State University, University Park, PA, USA, in 2010.

She was a Research Scientist with the Intel Labs, Santa Clara, CA, USA, and the Researcher-In-Residence with the Center for Future Architectures Research, Ann Arbor, MI, USA. She is currently an Assistant Professor with the University of Michigan, Ann Arbor. Some of her recent projects include in-memory architectures, custom computing for precision health and AI, fine-grain heterogeneous core architectures for mobile systems, and low-power scalable interconnects for kilo-core processors. She has authored over 45 articles and holds seven patents.

Prof. Das received two IEEE top picks awards, the NSF CAREER Award, the CRA-W's Borg Early Career Award, the Intel Outstanding Researcher Award, and the Sloan Foundation Fellowship. She has been inducted into IEEE/ACM MICRO and ISCA Hall of Fame. She served on over 30 technical program committees and as the Program Co-Chair for MICRO-52.

Satish Narayanasamy (Member, IEEE) received the B.E. degree in computer science and engineering from Anna University, Chennai, India, in 2001 and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego, San Diego, CA, USA, in 2005 and 2007, respectively.

He has been an Associate Professor with the Electrical Engineering and Computer Science (EECS) Department, University of Michigan, Ann Arbor, MI, USA, since 2014. He is also the President and the Co-Founder of Sequal Inc., Ann Arbor. His current interests include concurrent systems, specialized accelerator architectures and systems for mobile and web applications, big data for program analysis and MOOC tools, and system reliability.

David Blaauw (Fellow, IEEE) received the B.S. degree in physics and computer science from Duke University, Durham, NC, USA, in 1986 and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Urbana and Champaign, IL, USA, in 1991.

Until August 2001, he worked with Motorola, Inc., Austin, TX, USA, where he was the Manager of the High Performance Design Technology Group. Since August 2001, he has been a Faculty Member with the University of Michigan, Ann Arbor, MI, USA, where he is currently the Kensall D. Wise Collegiate Professor of electrical engineering and computer science (EECS) and also the Director of the Michigan Integrated Circuits Laboratory. He has authored over 600 articles and holds 65 patents. He has performed extensive research in ultralow-power computing using subthreshold operation and analog circuits for millimeter sensor systems, which was selected by the MIT Technology Review as one of the year's most significant innovations. For high-end servers, his research group introduced so-called near-threshold computing, which has become a common concept in semiconductor design. Most recently, he has pursued research in cognitive computing using analog, in-memory neural networks for edge devices, and genomics acceleration for precision health.

Dr. Blaauw was the General Chair of the IEEE International Symposium on Low Power and the Technical Program Chair of the ACM/IEEE Design Automation Conference and serves on the IEEE International Solid-State Circuits Conference's Technical Program Committee. He received the 2016 SIA-SRC Faculty Award for lifetime research contributions to the U.S. semiconductor industry, the Motorola Innovation Award, and numerous best paper awards.