

ASMCap: An Approximate String Matching Accelerator for Genome Sequence Analysis Based on Capacitive Content Addressable Memory

Hongtao Zhong¹, Zhonghao Chen¹, Wenqin Huangfu², Chen Wang¹, Yixin Xu³, Tianyi Wang⁴, Yao Yu⁴,
Yongpan Liu¹, Vijaykrishnan Narayanan³, Huazhong Yang¹, Xueqing Li^{1†}

¹BNRist/ICFC, Electronic Engineering Department, Tsinghua University, Beijing, China; ²Meta Platforms, Inc.;

³Pennsylvania State University; ⁴Daimler Greater China Ltd.; [†]Email: xueqingli@tsinghua.edu.cn

Abstract—Genome sequence analysis is a powerful tool in medical and scientific research. Considering the inevitable sequencing errors and genetic variations, approximate string matching (ASM) has been adopted in practice for genome sequencing. However, with exponentially increasing bio-data, ASM hardware acceleration is facing severe challenges in improving the throughput and energy efficiency with the accuracy constraint.

This paper presents ASMCap, an ASM acceleration approach for genome sequence analysis with hardware-algorithm co-optimization. At the circuit level, ASMCap adopts charge-domain computing based on the capacitive multi-level content addressable memories (ML-CAMs), and outperforms the state-of-the-art ML-CAM-based ASM accelerators EDAM with higher accuracy and energy efficiency. ASMCap also has misjudgment correction capability with two proposed hardware-friendly strategies, namely the *Hamming-Distance Aid Correction* (HDAC) for the substitution-dominant edits and the *Threshold-Aware Sequence Rotation* (TASR) for the consecutive indels. Evaluation results show that ASMCap can achieve an average of 1.2x (from 74.7% to 87.6%) and up to 1.8x (from 46.3% to 81.2%) higher F_1 score (the key metric of accuracy), 1.4x speedup, and 10.8x energy efficiency improvement compared with EDAM. Compared with the other ASM accelerators, including ResMA based on the *comparison matrix*, and SaVI based on the *seeding* strategy, ASMCap achieves an average improvement of 174x and 61x speedup, and 8.7e3x and 943x higher energy efficiency, respectively.

Index Terms—genome sequence analysis, approximate string matching, capacitive multi-level content addressable memory.

I. INTRODUCTION

Recently, genome sequencing has received a lot of attention and triggered new innovations in several applications including precise medical care [1], virus surveillance [2], evolutionary theory [3], etc. With recent advances in sequencing technologies like Next-Generation Sequencing (NGS) [4] or Third-Generation Sequencing (TGS) [5], massive amounts of genomics data can be generated at low cost. Such exponentially increasing bio-data scale much faster than the computing capability [6], which brings severe challenges to the genome sequence analysis.

In genome sequence analysis, small random fragments of the original DNA sequence extracted by machines, called *reads*, are analyzed by a computational process called *read mapping*. Specifically, each read is aligned to one or more possible locations in the reference sequence. Then, at these locations, matches and differences, i.e., *distance*, are determined between the read and the reference sequence segments [7]. Considering the inevitable errors caused by sequencing, genetic mutations and variations, Approximate String Matching (ASM) is necessary in the read mapping.

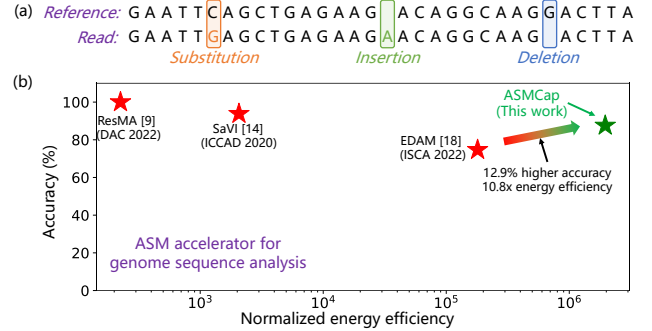


Fig. 1. ASM acceleration for genome sequence analysis: (a) Three types of errors: substitution, insertion, and deletion; (b) Higher energy efficiency is desired for ASM acceleration.

In ASM, as shown in Fig. 1(a), in addition to substitutions, there are two basic errors: insertions and deletions (collectively called *indels*). These three types of errors are typically referred to as *edits*. In the von Neumann architecture, finding the *Edit Distance* (ED), involves many complex operations and produces huge intermediate data with the growing sequence length, which further increases the data movement between the on-chip processors and the off-chip memory or storage [8]. It is a severe challenge towards efficient genome sequence analysis.

ED can be calculated exactly using *Comparison Matrix* (CM), but this iterative process is very costly even with the filtering and PIM techniques in ResMA [9]. Therefore, the *seeding* strategy is frequently utilized in genomics [10], [11], and several corresponding PIM-based accelerators have been developed on different platforms [12]–[14]. These two strategies find the most possible matched location by exactly matching all k -length read fragments, (called k -mers), with the reference and then *extending* or *voting* the matched k -mers. Therefore, significant performance improvement can be achieved over the CM method, but they still suffer from limited throughput due to the computationally expensive k -mer counting and the *extending* or *voting* process [14].

Recently, EDAM, an ASM accelerator based on the Multi-Level Content Addressable Memories (ML-CAMs) [15]–[17], was reported in [18] with orders of magnitudes higher energy efficiency, as shown in Fig. 1(b). Unlike the conventional CAMs or ML-CAMs, for a certain base, EDAM matches not only the co-located base but also its left and right neighbors. Therefore, it can tolerate *intra-mer* edits, making it possible to directly match the reads with the reference without fragmentation. Combining the *in-situ* parallel match of CAMs, EDAM

achieves much higher performance. However, the current-mode computing and sensing in EDAM are not variation-resistant and scalable, which limits the read length and degrades energy efficiency. Furthermore, the matching method in EDAM may induce misjudgments under certain conditions, which affects the accuracy and limits its application.

From the analysis above, as shown in Fig. 1(b), there is a trade-off between accuracy and energy efficiency in existing ASM accelerators, and it is very challenging to find a good balance. In this paper, we propose ASMCap, a novel ASM architecture based on the capacitive ML-CAMs [17]. ASMCap investigates the matching method proposed in EDAM and makes significant progress: (i) The charge-domain computation with no area cost is proposed to provide linear and stable voltage output scaled by the matching results, and achieve much higher reliability and read length upper bound with lower capacitor variations; (ii) ASMCap proposes two hardware-friendly heuristic misjudgment correction strategies, namely the *Hamming-Distance Aid Correction* (HDAC) and the *Threshold-Aware Sequence Rotation* (TASR), to improve the accuracy with negligible area overheads.

The contributions of this paper are as follows:

- We propose ASMCap, a novel ASM architecture based on the capacitive ML-CAMs with inherent high reliability, throughput, and energy efficiency.
- We propose two hardware-friendly heuristic misjudgment correction strategies, i.e., HDAC and TASR, for higher accuracy with negligible overheads.
- We perform extensive experiments for ASMCap. Results show an average of 1.2x and up to 1.8x higher F_1 accuracy score than EDAM, and an average of 174x, 61x and 1.4x speedup and 8.7e3x and 943x and 10.8x higher energy efficiency than the state-of-the-art ASM accelerators ReSMA, SaVI and EDAM, respectively.

Next, Section II introduces the genome sequence analysis background. Section III presents ASMCap with further software optimizations in Section IV. Section V shows the experimental results and Section VI concludes this work.

II. BACKGROUND

A. Genome Sequence Analysis

Genome sequences consist of four types of bases: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). A-T and C-G are complementary base pairs (bps). To perform the genome sequence analysis, the positions of the reads need to be determined, and read mapping is to align each read sequence to one or more possible locations in the reference sequence. The *read alignment* includes *Exact Alignment* [19] and *Approximate Alignment* [10], [20]. Considering the inevitable errors, *Approximate Alignment*, i.e., ASM, needs to be supported in practical genome sequencing [20].

There are two major error sources in genome sequences: sequencing errors and genetic variations. For example, NGS [4] can produce *short reads* (~ 50 -500 bps) with *low errors* ($\sim 0.1\%$ -1%) [21], while TGS [5] can produce *long reads* (thousands to millions of bps) with *high errors* ($\sim 10\%$ -

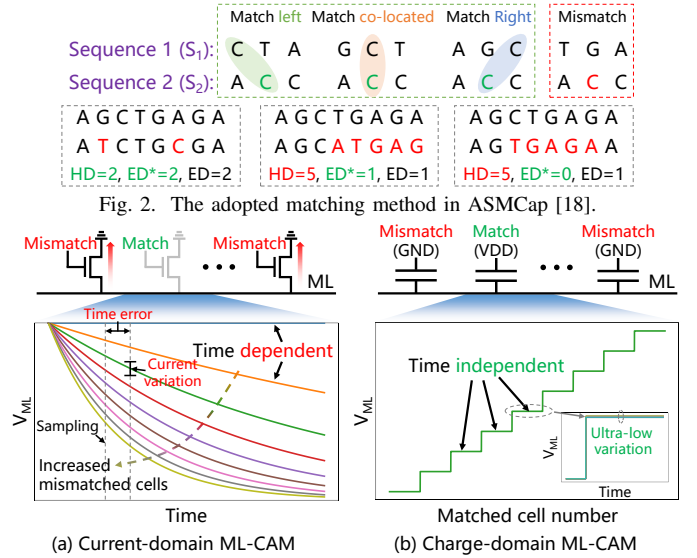


Fig. 3. ML-CAMs in the (a) current-domain [15], [16] and (b) charge-domain [17] for multi-level content matching.

15%) [20]. The error rate of genetic variations is relatively low ($\sim 0.1\%$ for human [6]).

B. Approximate String Matching and Related Works

ED between two sequences is the minimum number of edits required to transform one sequence to the other one. Given a read sequence $R=[r_1r_2\dots r_m]$, a reference sequence $Q=[q_1q_2\dots q_n]$, $m=|R|$, $n=|Q|$, $m\leq n$ and a ED threshold T , the ASM goal is to find all possible subsequences S in Q and their positions such that $ED(S, R)\leq T$.

ED can be calculated exactly using the *Comparison Matrix* $M[i, j]$, but the time complexity of this ED computation is $\mathcal{O}(N^2)$, which is very costly [9]. ReSMA [9], a Resistive Random Access Memory (RRAM) based PIM accelerator, utilizes the RRAM crossbars to exploit anti-diagonal parallelism in $M[i, j]$ and the RRAM CAMs for filtering, which gains great performance improvement. However, the iterative process during CM computation incurs massive intermediate data and updates the crossbars frequently, which heavily degrades the performance and energy efficiency.

A more common method in genomics is the *seeding* strategy, including the *seed-and-extend* strategy [10] and the *seed-and-vote* strategy [11]. The *seeding* strategy splits the read into many k -mers, and then finds all locations of the reference for each k -mer by *exact matching*, and finally, extends or votes the matched k -mers to find the best possible match between the read and the reference. The *voting* process is faster than the *extending* process but suffers from the accuracy loss [11].

There have been several existing accelerators for the *seeding* strategy based on 3D RRAM, DIMM, TCAMs, etc. [12]–[14]. These two strategies can both be much faster than the CM computing, but still suffer from limited throughput due to the memory-hungry and time-consuming k -mer counting [13]. In addition, both *extending* and *voting* processes involve complex computation, which further degrades the performance.

The recent ASM accelerator EDAM [18] is based on ML-CAMs [15]–[17]. As illustrated in Fig. 2, EDAM performs

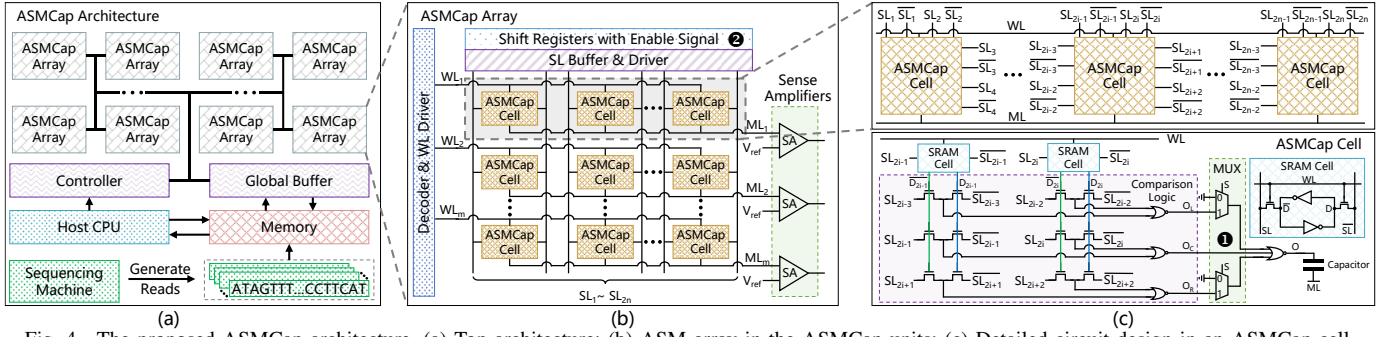


Fig. 4. The proposed ASMCap architecture. (a) Top architecture; (b) ASM array in the ASMCap units; (c) Detailed circuit design in an ASMCap cell.

the search operation for a certain base with not only the co-located base but also the left and right neighbors of the co-located base. If there is at least one match, the matching result of this base is ‘match’. Otherwise, the result is ‘mismatch’. The distance estimated by EDAM is defined as ED^* in this paper. From the examples in Fig. 2, it is seen that ED^* can be more close to ED compared with *Hamming Distance* (HD) when insertions or deletions occur. In other words, EDAM can tolerate *intra-mer* edits, support much larger k , and even match the read directly without fragmentation. In this way, the k -mer counting and the *extending* or *voting* process can be reduced and even eliminated. With the *in-situ* parallel match of CAMs, EDAM can achieve ultra-high throughput.

C. Current Domain and Charge Domain ML-CAMs

ML-CAM is one type of CAM that can not only perform *exact matching* or *not*, but also detect the *degree of match*. EDAM uses the current-domain ML-CAM [15], as shown in Fig. 3(a). The matching result of each base controls a transistor: a ‘match’ turns off the transistor, and a ‘mismatch’ turns on the transistor. The matchline (ML) is pre-charged, and the decreasing slope of the ML voltage V_{ML} scales with the number of mismatched cells. However, this dynamic timing-dependent sensing method is inherently vulnerable to device and timing-control variations and power-consuming, which results in poor scalability and limits the maximum read length that EDAM can support. Besides, the pre-charge operation in EDAM also consumes significant power.

To address the inherent challenge of the current-domain ML-CAM, charge-domain ML-CAM, i.e., the capacitive ML-CAM, was proposed [17] for one-shot learning. As shown in Fig. 3(b), the matching result of each base outputs a certain voltage (V_{DD} for ‘match’ and GND for ‘mismatch’) to the bottom plate of a capacitor, and the top plates of all capacitors are connected to ML. It shows that V_{ML} scales linearly with the degree of match, and is stable and timing-independent. Besides, capacitor variations are much less than the current variations, which enhances the accuracy and scalability.

III. PROPOSED ASMCAP ARCHITECTURE

This section presents the proposed ASMCap, including the top architecture and detailed building blocks. Operation mechanisms and performance analysis are also included.

A. Top Architecture

Fig. 4(a) shows the top architecture of ASMCap. The reads generated by the sequencing machine are firstly stored in the

memory and the global buffer can fetch the entire reads or k -mers for the subsequent match according to the read length. The controller receives instructions from the host CPU and controls the process. The reads or k -mers from the global buffer are sent to the ASMCap arrays through the H-tree.

B. ASMCap Array

As shown in Fig. 4(b), each ASMCap array includes $M \times N$ ASMCap cells, the decoder, the wordline (WL) driver, the searchline (SL) buffer and driver, the sense amplifiers (SAs) on the matchline (ML), and the shift registers with an enable signal. Each row stores a reference segment of the same length as the input reads or k -mers. The decoder and the WL driver obtain the address and select rows to execute write and search operations. SL buffer and driver transform the input data to the corresponding voltage and drive the SLs and \overline{SL} s.

The SAs compare V_{ML} with the reference voltage (V_{ref}). Different from EDAM [18], the sample and hold circuit and the timing are not required in ASMCap, thanks to the stable output voltage on the ML, which reduces the search latency. the SAs are required to output ‘1’ (‘match’) when $V_{ML} \leq V_{ref}$ corresponding to $ED^* \leq T$.

The shift registers with enable signal can rotate the input reads or k -mers left or right base-by-base, and are utilized in the proposed TASR strategy in Section IV-B.

C. ASMCap Cell

Fig. 4(c) shows the circuit design of the i^{th} ASMCap cell in a row of the ASMCap array. The i^{th} base of the reference segment is stored in the two 6T SRAM cells. The comparison logic compares the stored base with the co-located base and its left and right neighbors in the read, and produces the partial matching results, i.e., O_C , O_L , and O_R , respectively. If match occurs, the corresponding partial matching result is ‘1’. Then, two multiplexers (MUXs) control the matching mode. If the select signal of the MUXs (S) is ‘1’, the output matching result (O) is $\overline{O_C} + O_L + O_R$. Otherwise, $O = \overline{O_C}$. In this way, the ASMCap array can perform the search operation using both ED^* and HD, which is utilized in the proposed HDAC strategy in Section IV-A. Note that the select signal S is shared by all MUXs in the ASMCap array, and MLs are not required to be pre-charged during the search operation as EDAM [18], which also reduces the search latency.

For the matching mode using ED^* , if at least one partial matching result is ‘1’, O is ‘0’ and the i^{th} cell is a matched cell. Otherwise, O is ‘1’, and the i^{th} cell is a mismatched cell.

Therefore, $V_{ML} = \frac{n_{mis}}{N} VDD$, where n_{mis} is the total number of mismatched cells equivalent to ED^* . Therefore, $ED^* \leq T$ corresponds to $V_{ML} \leq V_{ref}$ when V_{ref} is set to $\frac{T}{N} VDD$.

Furthermore, if all capacitors in the ASMCap array follow the independent and identically distributed normal distribution $\mathcal{N}(\mu_C, \sigma_C^2)$, as [17], the energy consumption per search operation (E_S) and the V_{ML} variance can be estimated as:

$$E_S \approx \frac{M n_{mis} (N - n_{mis})}{N} \mu_C VDD^2 \quad (1)$$

$$Var(V_{ML}) \approx \frac{n_{mis} (N - n_{mis})}{N^3} \left(\frac{\sigma_C}{\mu_C} \right)^2 VDD^2 \quad (2)$$

It is seen that E_S and $Var(V_{ML})$ are small when n_{mis} is close to 0 or N . Considering the genome sequence characteristics, n_{mis} is close to N for most rows, making ASMCap superior to EDAM with much lower power consumption and variations.

IV. MORE SOFTWARE OPTIMIZATIONS

Although EDAM can tolerate edits and is efficient in estimating ED when the indels occur, misjudgments limit the accuracy of EDAM. Because of the huge design space, it is challenging to correct those misjudgments with low hardware overhead. This section focuses on two common kinds of misjudgments and describes two proposed corresponding hardware-friendly heuristic strategies, i.e., HDAC and TASR, for misjudgment correction with the overhead analysis.

A. Hamming-Distance Aid Correction

Observation: As illustrated in Fig. 5, a common kind of misjudgment happens when there are many substitutions but few or even no indels. If performing the search operation using HD, most edits can be discovered. However, EDAM performs the search operation for a certain base with the left and right neighbors, many edits are likely to be hidden. It leads to the misjudgment, i.e., False Positive (FP), when the threshold T is between ED^* and ED (The ED^* matching result is ‘match’ but the result should be ‘mismatch’ due to $ED > T$).

Solution: We propose the HDAC strategy to address this issue. As shown in Fig. 5, we perform the search operation using both HD and ED^* , and then select the HD matching result with a p possibility. The pseudo codes of the HDAC are listed in Algorithm 1. The main challenge is how to design the function $f()$ in **line 1** such that p is large enough when edits are mainly from substitutions, and decreases rapidly when more indels occur. In this paper, the function is designed as $\frac{e_s}{e_s + e_{id}} e^{-(\alpha e_{id} + \beta T)}$, where e_s is the substitution error rate, e_{id} is the indel error rate ($e_{id} = e_i + e_d$, where e_i is the insertion error rate, and e_d is the deletion error rate), and α and β are constants. The $\frac{e_s}{e_s + e_{id}}$ part is to enlarge p with larger proportion of substitutions in edits. The $e^{-\alpha e_{id}}$ part is to exponentially alleviate the effect of more indels due to larger e_{id} . The $e^{-\beta T}$ part is to exponentially alleviate the effect of a larger threshold T because many matching results of indel-induced large HD are ‘mismatch’, but more of them with small ED should be ‘match’ with larger T , which leads to another misjudgment, i.e., False Negative (FN). Besides, $\frac{e_s}{e_s + e_{id}} e^{-(\alpha e_{id} + \beta T)}$ can be pre-processed off-line to reduce the

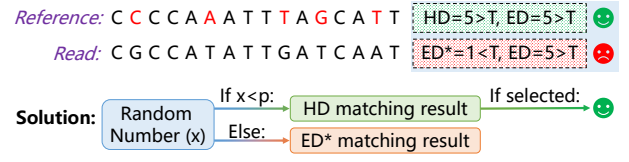


Fig. 5. The proposed HDAC strategy for substitution-dominant edits (5 substitutions, no indels, and $T = 4$ in this example).

Algorithm 1 Hamming-Distance Aid Correction

Input: O_{HD} : The HD matching result; O_{ED^*} : The ED^* matching result; e_s : The substitution error rate; e_{id} : The indel error rate; T : The threshold

Output: O : The final matching result

- 1: Let $p \leftarrow f(e_s, e_{id}, T)$; // Can be pre-processed off-line
- 2: **if** $O_{HD} \neq O_{ED^*}$ **then**
- 3: Generate a random number $\mathbb{X} \sim U(0, 1)$;
- 4: **if** $\mathbb{X} < p$ **then**
- 5: Let $O \leftarrow O_{HD}$;
- 6: **else**
- 7: Let $O \leftarrow O_{ED^*}$;
- 8: **end if**
- 9: **end if**

computation overhead. It is also necessary to point out that this function is only an example. More accurate functions can be designed analytically or obtained by other powerful tools such as neural networks.

Overhead Analysis: HDAC involves two additional MUXs per ASMCap cell in ① of Fig. 4. The MUXs are achieved by two NMOS transistors controlled by S and \bar{S} , respectively. Estimated from the layout, two MUXs induce about 0.1% area overhead through layout optimization. Besides, the HDAC strategy also induces one more cycle to perform the search operation using HD, we can disable the HDAC strategy when p is less than a certain threshold (e.g., 1%).

B. Threshold-Aware Sequence Rotation

Observation: As illustrated in Fig. 6, another common kind of misjudgment happens when several consecutive insertions or deletions occur, which causes ED^* to be much larger than ED. If T is between ED and ED^* , the FN happens.

Solution: An effective way is Sequence Rotation (SR) [18]. SR rotates the read base-by-base N_R times (can be the left or right rotation), and the reference segment is compared with the original read and N_R rotated reads. If there is at least one ‘match’, the final matching result is ‘match’. However, as illustrated in Fig. 6, the ED^* s between the reference segment and some rotated reads may be smaller than ED, so the FP occurs if T is between ED and these ED^* s, especially when T is relatively small. In this paper, we propose an improved strategy, i.e., TASR, and the pseudo codes are listed in Algorithm 2. We set a lower bound of T , called T_l , such that the rotations are triggered only if $T \geq T_l$. In this way, the FP is avoided when T is relatively small. T_l also involves huge design space exploration, and here we design T_l as $\lceil \frac{\gamma}{e_{id}} m \rceil$, where γ is a constant and m is the read length, considering T_l should be small when e_{id} is high to improve accuracy and be large when e_{id} is low to save time and power consumption.

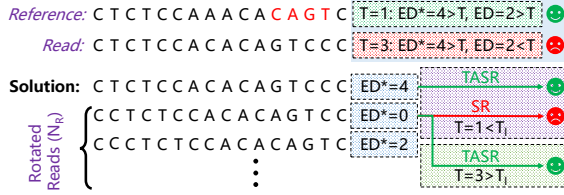


Fig. 6. The proposed TASR strategy for consecutive insertions or deletions (delete consecutive 'AA' in the read and $T_l = 2$ in this example).

Algorithm 2 Threshold-Aware Sequence Rotation

Input: R : The read; S : The reference segment; N_R : The total rotation number; T : The threshold; T_l : The lower bound of T to trigger the sequence rotation
Output: O : The final matching result
if $T < T_l$ **then**
 2: Let $O \leftarrow \text{ED}^*(S, R) \leq T$;
else
 4: **for** i from 0 to N_R **do**
 Let $R_i \leftarrow \text{Rotate}(R, i)$; // Rotate left (right) i bases
 6: Let $O \leftarrow O$ **or** $\text{ED}^*(S, R_i) \leq T$;
end for
 8: **end if**

Overhead: TASR requires additional shift registers with enable signals in ② of Fig. 4, but the average area overhead per cell is only 0.2%. Besides, the *rotation-and-comparison* process also induces N_R more cycles. With the T_l limitation, the average latency overhead can be significantly reduced.

V. EXPERIMENTS AND DISCUSSION

A. Experimental Setup

Genome Sequence Analysis: The datasets used in the experiments are the human genome from NCBI [22]. In this paper, we mainly focus on *short reads* considering the capacity and the sensing ability of the ML-CAMs. The reads are set to 256-base length within the typical range of the read length (~ 50 -500 bps [21]), and extracted from random positions in human DNA sequences. Then, edits are randomly injected, which creates metagenomic datasets with the following two kinds of mixed error rates for edits considering the typical error range of the *short reads* ($\sim 0.1\%$ -1%) [21]:

- **Condition A:** $e_s = 1\%$ and $e_i = e_d = 0.05\%$;
- **Condition B:** $e_s = 0.1\%$ $e_i = e_d = 0.5\%$;

Each 256-base-long read in the metagenomic datasets is directly sent to the EDAM and ASMCap arrays without fragmentation. The references extracted from the human genome are segmented and stored in the EDAM and ASMCap arrays.

To compare the accuracy, we evaluate the F_1 score as EDAM, and the F_1 score is calculated by:

$$\text{Sensitivity} = \frac{TP}{TP + FN}, \text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$F_1 = \frac{2 \times \text{Sensitivity} \times \text{Precision}}{\text{Sensitivity} + \text{Precision}} \quad (4)$$

where the TP is True Positive that the matching result is 'match' and it indeed should be 'match'. Besides, as EDAM, we use the popular tool Kraken2 [23] as a baseline and the normalized F_1 score is normalized by $F_1(\text{Kraken2})$.

TABLE I
CIRCUIT-LEVEL COMPARISON BETWEEN ASMCAP AND EDAM

	EDAM [18]	ASMCap
ML-CAM Mode	Current domain	Charge domain
Technology	65nm	65nm
Cell Area	$33.4\mu\text{m}^2$ (1.4x)	$24.0\mu\text{m}^2$ (1x)
Supply voltage	1.2V	1.2V
Search time	2.4ns (2.6x)	0.9ns (1x)
Average Power per cell ¹	1.0μW (8.5x)	0.12μW (1x)

¹ The average power under two conditions.

EDAM and ASMCap Configurations: ASMCap is designed in a commercial 65nm CMOS process, and 2fF MIM capacitors are used in ASMCap cells. The search voltage, the array size, and the array number of EDAM and ASMCap are both set to 1.2V, 256×256 , and 512, respectively. For the proposed two strategies of ASMCap, α and β are set to 200 and 0.5 in HDAC, respectively, and N_R and γ are set to 2 and 2×10^{-4} in TASR, respectively.

Other ASM Solutions: We compare ASMCap with other state-of-the-art ASM solutions mentioned in Section II-B, including the CM computation using i9-10980XE CPU (CM-CPU) as the baseline and the CM computation using RRAM-based PIM accelerators (ReSMA [9]) and the *seed-and-vote* strategy using TCAM (SaVI [14]). All these ASM solutions also process 256-base-long reads for a fair comparison.

B. Area and Power Breakdown of ASMCap

For a 256×256 ASMCap array, the area and power are 1.58mm^2 , and 7.67mW, respectively. For area, more than 99% of the area is occupied by the ASMCap cells. For power, the ASMCap cells, the shift registers, and SAs occupy 75%, 19%, and 6% of power, respectively.

C. Circuit-Level Comparison with EDAM

We evaluate the circuit-level simulations using Cadence Virtuoso, and Table I lists the circuit-level comparison between ASMCap and EDAM. Results show that ASMCap achieves 1.4x cell area reduction, 2.6x less search time, and 8.5x power reduction compared with EDAM.

Through layout estimation, the area reduction results from both the reduced transistors for the discharge process in EDAM and layout optimizations. Although the area of a 65nm 2fF MIM capacitor is about $1.4\mu\text{m}^2$ [17], the capacitors can be placed on top of the cell, which induces no area penalty considering the large cell area.

The less search time is from the skipped pre-charge operation and the sampling operation in EDAM, as analyzed in Section III. Besides, the smaller parasitic effect due to the smaller area also helps ASMCap reduce the search time. The power reduction is mainly from the charge-domain computation mode, as analyzed in Section III.

D. Accuracy Comparison with EDAM

To perform the accuracy comparison, Monte Carlo simulations are carried out for both ASMCap and EDAM. The estimated current variation in EDAM is 2.5%, supporting at most 44 distinguishable states under the 3σ constraint for V_{ML} . Meanwhile, the capacitor variation in ASMCap is only 1.4%. Combined with Equation 2, ASMCap can support 566 distinguishable states even with the worst case.

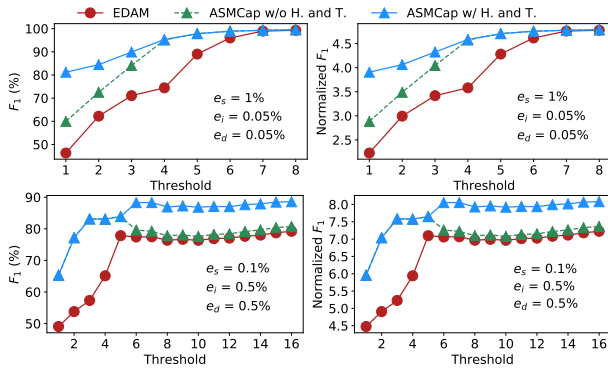


Fig. 7. Accuracy (F_1) comparison between ASMCap with the proposed strategies and EDAM. H. and T. represent HDAC and TASR, respectively.

In this way, as shown in Fig. 7, ASMCap without the HDAC and TASR strategies achieves an average of 1.13x and 1.11x higher F_1 in Condition A and B, respectively (1.12x higher F_1 on average). Furthermore, the proposed HDAC strategy achieves an average of 1.07x further higher F_1 in Condition A, and the proposed TASR strategy achieves an average of 1.08x further higher F_1 in Condition B. Therefore, ASMCap with the HDAC and TASR strategies achieves an average of 1.2x higher F_1 (from 74.7% to 87.6%), and up to 1.8x higher F_1 (from 46.3% to 81.2%) when $T=1$ in Condition A. Finally, compared with Kraken [23] with *exact matching*, ASMCap can achieve an average of 4.5x and 7.7x higher F_1 in Condition A and B, respectively (6.6x higher F_1 on average).

E. Comparison with Existing ASM Accelerators

To evaluate the system-level performance and energy efficiency, 512 ASMCap arrays are utilized with 64Mb memory capacity, which can entirely store some small virus sequences (e.g., SARS-CoV-2 in coronavirus pandemic [18]). As shown in Fig. 8, without the proposed HDAC and TASR strategies, ASMCap achieves an average of 9.7e4x, 362x, 126x and 2.8x speedup and an average of 5.1e6x, 2.3e4x, 2.4e3x and 28x energy efficiency compared with CM-CPU, ReSMA, SaVI, and EDAM, respectively. Considering the average effect of the proposed HDAC and TASR strategies, ASMCap achieves an average of 4.7e4x, 174x, 61x and 1.4x speedup and an average of 2.0e6x, 8.7e3x, 943x and 10.8x energy efficiency compared with CM-CPU, ReSMA, SaVI, and EDAM, respectively.

For accuracy, the CM computation can reach 100% accuracy, while the *seed-and-vote* strategy achieves about 93.8% accuracy on average [11]. Therefore, with ultra-high throughput and energy efficiency with comparable accuracy, ASMCap is more suitable for the task-intensive but accuracy-insensitive scenarios such as fast testing.

VI. CONCLUSION

This paper proposes ASMCap, an ASM acceleration approach based on capacitive ML-CAMs that achieves the highest reported energy efficiency. Compared with prior ML-CAM based accelerators, the charge-domain computation of ASMCap improves the accuracy and power efficiency. ASMCap also embodies two hardware-friendly strategies, i.e., HDAC and TASR, to overcome the accuracy loss challenge in prior state-of-the-art approximate matching methods with negligible

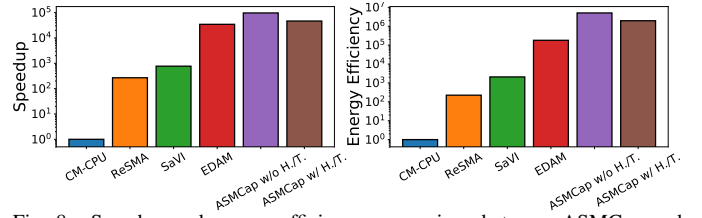


Fig. 8. Speedup and energy efficiency comparison between ASMCap and existing ASM accelerators. H. and T. represent HDAC and TASR, respectively. area cost. Results show that ASMCap outperforms the prior state-of-the-art ASM accelerators in both energy efficiency and speed with comparable accuracy.

ACKNOWLEDGMENT

This work is supported in part by the National Key R&D Program of China (No. 2019YFA0706100), in part by NSFC (U21B2030, 61934005), and in part by Tsinghua University – Daimler Greater China Ltd. Joint Institute for Sustainable Mobility. H. Zhong and Z. Chen make equal contributions.

REFERENCES

- [1] A. E. Guttacher, *et al.*, “Personalized genomic information: preparing for the future of genetic medicine,” *Nat. Rev. Genet.*, 2010.
- [2] R. Lu, *et al.*, “Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding,” *Lancet*, vol. 395, no. 10224, pp. 565–574, 2020.
- [3] J. Prado-Martinez, *et al.*, “Great ape genetic diversity and population history,” *Nature*, vol. 499, no. 7459, pp. 471–475, 2013.
- [4] J. Shendure and H. Ji, “Next-generation DNA sequencing,” *Nat. Biotechnol.*, vol. 26, no. 10, pp. 1135–1145, 2008.
- [5] E. E. Schadt, *et al.*, “A window into third-generation sequencing,” *Hum. Mol. Genet.*, vol. 19, no. R2, pp. R227–R240, 2010.
- [6] S. Canzar and S. L. Salzberg, “Short read mapping: an algorithmic tour,” *Proc. IEEE*, vol. 105, no. 3, pp. 436–458, 2015.
- [7] C. Alkan, *et al.*, “Personalized copy number and segmental duplication maps using next-generation sequencing,” *Nat. Genet.*, 2009.
- [8] H. Wei, *et al.*, “String similarity search: A hash-based approach,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, pp. 170–184, 2017.
- [9] H. Li, *et al.*, “ReSMA: accelerating approximate string matching using ReRAM-based content addressable memory,” in *DAC*, 2022.
- [10] S. F. Altschul, *et al.*, “Basic local alignment search tool,” *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.
- [11] Y. Liao, G. K. Smyth, and W. Shi, “The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote,” *Nucleic Acids Res.*, vol. 41, no. 10, pp. e108–e108, 2013.
- [12] W. Huangfu, S. Li, X. Hu, and Y. Xie, “Radarr: a 3d-reram based dna alignment accelerator architecture,” in *DAC*, 2018, pp. 1–6.
- [13] W. Huangfu, *et al.*, “Nest: Dimm based near-data-processing accelerator for k-mer counting,” in *ICCAD*, 2020, pp. 1–9.
- [14] A. F. Laguna, *et al.*, “Seed-and-vote based in-memory accelerator for dna read mapping,” in *ICCAD*, 2020, pp. 1–9.
- [15] K. Ni, *et al.*, “Ferroelectric ternary content-addressable memory for one-shot learning,” *Nat. Electron.*, vol. 2, no. 11, pp. 521–529, 2019.
- [16] E. Garzón, *et al.*, “Hamming distance tolerant content-addressable memory (hd-cam) for dna classification,” *IEEE Access*, 2022.
- [17] X. Ma, *et al.*, “CapCAM: A Multilevel Capacitive Content Addressable Memory for High-Accuracy and High-Scalability Search and Compute Applications,” *IEEE Trans. Very Large Scale Integr. Syst.*, 2022.
- [18] R. Hanhan, *et al.*, “EDAM: Edit Distance tolerant Approximate Matching content addressable memory,” in *ISCA*, 2022, pp. 495–507.
- [19] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows–Wheeler transform,” *bioinformatics*, 2009.
- [20] D. S. Cali, *et al.*, “Genasm: A high-performance, low-power approximate string matching acceleration framework for genome sequence analysis,” in *MICRO*, 2020, pp. 951–966.
- [21] S. Goodwin, J. D. McPherson, and W. R. McCombie, “Coming of age: ten years of next-generation sequencing technologies,” *Nat. Rev. Genet.*, vol. 17, no. 6, pp. 333–351, 2016.
- [22] NCBI. Genome Database. <https://www.ncbi.nlm.nih.gov/genome>.
- [23] D. E. Wood, J. Lu, and B. Langmead, “Improved metagenomic analysis with Kraken 2,” *Genome Biol.*, vol. 20, no. 1, pp. 1–13, 2019.