CAP: A General Purpose Computation-in-memory with Content Addressable Processing Paradigm

Zhiheng Yue, Shaojun Wei, Yang Hu, Shouyi Yin
Tsinghua University, Beijing, China

ABSTRACT

Demands for efficient computing under memory wall have led to computation-in-memory (CIM) accelerators that leverage memory structure to perform in-situ computing. The content addressable memory (CAM) processing is a CIM paradigm that accomplishes general purpose functions, via sequences of search and update operations on CAM. However, the conventional CAM-based CIM is customized for vector-vector operation only and requires long search-update iterations for computing.

To mitigate the drawbacks of prior works, this work proposes a content addressable processor (CAP), improving both functionality and performance. CAP supports general purpose inter-vector and intra-vector operations. Respectively, CAP shortens the search and update step latency. Additionally, the sequence order of the search-update pair is released by CAP to achieve parallel search-update. CAP is implemented in 22nm CMOS technology with 0.6 mm^2 area. By integrating all the techniques, CAP achieves 2.68 $\times$ performance improvement over the baseline and also realizes 11.37 TOPs/W energy efficiency and 1.376 TOP/mm^2 area efficiency.

KEYWORDS

Content addressable memory, Computation-in-memory, Associative computing, General purpose accelerator

1 INTRODUCTION

The computation-in-memory architecture is a promising candidate to mitigate the memory bottleneck of conventional von Neumann architecture. In particular, CIM designs leverage the storage structure itself to perform in-situ computing. Prior works have demonstrated the performance benefits of the CIM in energy and area efficiency. However, the CIM designs are mostly customized for matrix multiplication and support multiplication and accumulation (MAC) function only. Content addressable parallel accelerator is one type of CIM established on the content addressable memory (CAM) and supports a variety of arithmetic and logic operations. The operations are realized by searching matched cells according to the input combinations of the truth table.

Recently, several designs have advocated for leveraging the content addressable accelerator in modern microarchitectures because of its CIM capability, flexibility, and massive parallelism. However, the proposed solutions support parallel vector operation but lack efficient intra-vector operation, like partial sum reduction. Furthermore, conventional CAM-based accelerators suffer from long search iterations and update latency. CAM can only search one pattern in each iteration. A complicated computation may require multiple search iterations and increases the latency. Traditional CAMs are derived from SRAM design and only one row is activated to write. But content addressable accelerator perform parallel operations and multiple partial results wait to be updated. The update latency cancels the benefits of computing parallelism.

To mitigate the challenge of conventional accelerators, this work implements a content addressable processor, named CAP. As Table. 1 shows, CAP improves the functionality of prior works and also boosts the performance by shortening the search iterations and update latency. And it decouples the sequence order of the search-update pair. The CAP is envisioned to be a standalone unit that supports data storage, parallel data search, and multiple logic functions. The contributions of this work are listed below:

- The functionality of the CAM-based accelerator is extended to support both inter-vector and intra-vector operation.
- The search step is optimized with reduced iterations.
- The multi-row update circuit shortens the update to 1-step.
- CAP decouples the sequential order of search-update pair, and hidden update latency with parallel pipeline.

Table 1: CAM-based Accelerator Comparison

Accelerator	Functionality	Performance		
		Search	Update	Order
ATAP[4]	✓*	×	×	×
hyper-AP[5]	×	✓	×	×
Castle[1]	✓*	✓	×	×
CAPE[2]	✓**	✓	×	×
This work	**✓**	**✓**	**✓**	**✓**

* Implement an adder tree for accumulation.
** Implement an isolated digital vector unit.

2 BACKGROUND

2.1 Content Addressable Memory

The Content Addressable Memory (CAM) is widely employed in applications requiring fast data search. For example, In networks like the Internet, the data packets are transferred from the source to the destination node through available routes. The router is responsible for comparing the target address of the packet to all candidate nodes, and decides the feasible path. The CAM is an ideal hardware choice due to its fast search capability.

The organization of CAM is shown in Fig. 1, which is derived from SRAM. The row/column peripheral activates corresponding cells for read/write. Based on the intrinsic 6T cell, the CAM cell includes the search logic. Typically, the NOR/NAND gate performs a comparison between the search bit and stored data. Then the

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655689

comparison results are merged on the horizontal matchline and quantized by the sense amplifier. Once one of the cells mismatches, the matchline is discharged to the ground (GND). Only when the entire row matches the search key, the matchline state hold at high. The search key is applied to all rows, thus CAM achieves parallel comparison between the search key and all data rows.

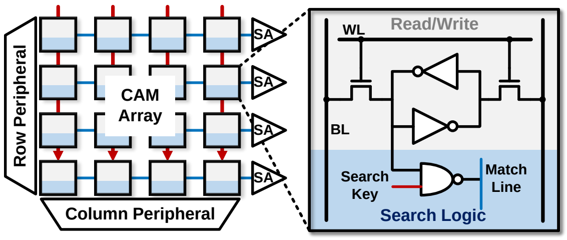


Figure 1: Content addressable memory architecture

2.2 CAM-based Associative Processing

The CAM-based associative processing is one of the computation-in-memory paradigms realized based on CAM architecture. Vector data is stored in parallel rows. And the associative processor performs various operations with 'search'. Specifically, all logic operations can be represented by the truth table. The CAM searches all input combinations that have 'truth' output. For example, AND logic is realized by searching all-1 input combinations. Without changing hardware configuration, CAM supports variable logic functions by altering the search key according to the truth table.

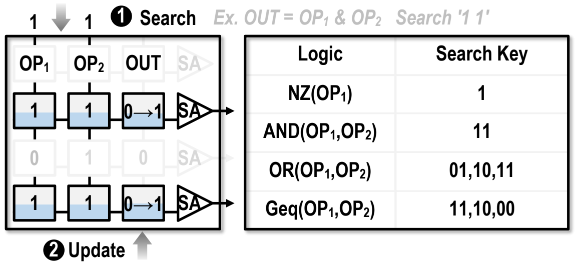
Next, the 'update' step writes computing results into the designated cell if the row matches with the search key. Each search-update pair is viewed as an atomic iteration. When the number of operands increases, various input combinations may have the same result, and multiple search-update iterations are required to complete one operation. CAM takes three iterations to complete the 'OR' logic, search '01'-update, search '10'-update, search '11'-update.

3 CAP ACCELERATOR

To resolve the limitation of conventional CAM accelerators, this section introduces the overall architecture of CAP and details the specific hardware for functionality improvement.

3.1 Accelerator Architecture

The overall architecture is shown in Fig. 3. The CAP accelerator is composed of a 32Kb CAM array, controller, voltage supply, and other BIST units. The CAM array contains 16 banks, each including cell array, row/column peripheral, and multi-stage intra-vector unit (MIU). Compared with conventional 6T SRAM, each cell of CAP includes 3 extra transistors for comparison. The search key is applied on the search line (SL) and shared by all rows. The SL data compares with the data in the bit cell. If they are the same, the voltage at node B equals 0 and N3 cuts off. The voltage of the match line (ML) is maintained at VDD. Otherwise, the N3 is turned on and a leakage path discharges the voltage of ML to GND. The comparison results of an entire row are merged on ML and quantized by a sense amplifier (SA). The row/column peripheral selectively activates the target cell for read/write. A multi-stage intra-vector unit is exploited to realize efficient intra-vector operations.



Logic	Search Key
NZ(OP_1)	1
AND(OP_1,OP_2)	11
OR(OP_1,OP_2)	01,10,11
Geq(OP_1,OP_2)	11,10,00

Figure 2: CAM-based associative processing.

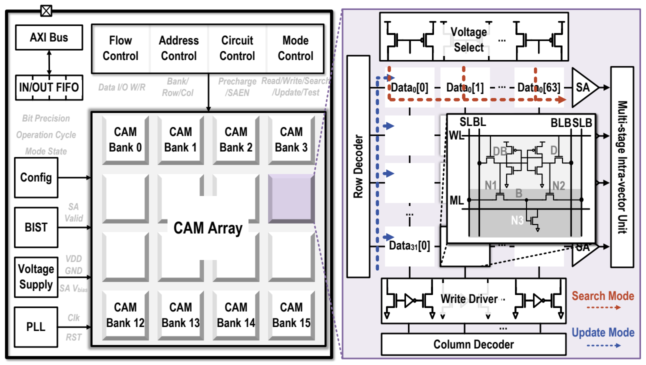


Figure 3: Overall architecture of CAP

3.2 Inter-vector & Intra-vector operation

The CAM-based accelerator is naturally suitable for inter-vector (vector-vector) operation as it takes advantage of row-wise parallelism. Vectors are allocated in multiple columns and all rows perform logic functions in parallel. However, the accelerator is inefficient in processing data from the same column, also known as the intra-vector operation. The intra-vector operation is essential in various applications, especially accumulation. The multiplication and accumulation (MAC) in convolution and fully connected layers account for over 99% of total operations in state-of-the-art neural networks[3]. The hamming distance employs XOR and accumulation (XAC) is widely applied in image processing, genome sequence and encryption. The other intra-vector operations, like leading 1 detection, data rank are also necessary in statistic analysis.

To efficiently perform intra-vector operations, CAP implements a multi-stage intra-vector unit. A hierarchical tree consumes input from the sense amplifier for post-processing. The two neighboring rows share the first-stage vector unit. Without introducing chunk area overhead, the unit supports basic 2-operand boolean logic, including the AND/OR/XOR logic gate. The intra-vector unit obeys the 'Turing Complete' since all functions can be derived from basic boolean logic. For example, the AND gate and XOR gate together perform the accumulation of two bits, producing Carry out and SUM. The valid detection function can be realized by the OR gate, the MIU generates 1 when a match row exists. The reconfigurability

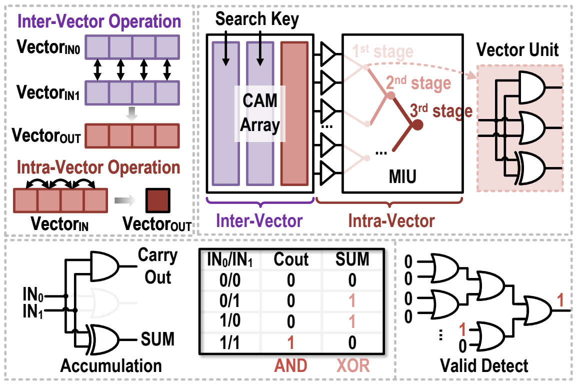


Figure 4: Inter-vector/intra vector operation and architecture of multi-stage intra-vector unit.

of the intra-vector unit realizes the required functions without changing the hardware. Next, the 2nd-stage vector unit accumulates the results from two neighboring 1st-stage units until hierarchical vector units merge results from all sense amplifiers.

4 SEARCH & UPDATE OPTIMIZATION

The conventional associative accelerator performs computing with two primitives, search and update. From this setting, CAP shortens the latency of the search and update procedure with the corresponding hardware design and further decouples these two steps to boost the performance.

4.1 Search Optimization

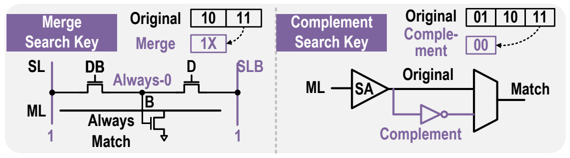
The associative processor improves the performance with element-wise parallelism. The search key is shared by all elements in a column. However, multiple search iterations are required if different input combinations have identical output. With the increasing number of input operands, the number of iterations also increases as more combinations may produce the same output. To reduce the number of iterations, we propose two hardware mechanisms.

This is first optimized by merging search keys with an ‘X’ key that matches with both 0 and 1 data, which is realized by driving SL and SLB to 0 simultaneously. The voltage at node B is fixed to 0 no matter what the stored data is, thus the comparison result is always-match. Next, if search combinations are more than 50% of all cases, the complement key is adopted with fewer iterations. A simple MUX selects the inversion of the SA output.

By employing search optimization, the iteration numbers of key operations are reduced effectively. The conventional CAM takes three iterations to complete ‘OR’ logic, which is reduced to 1 cycle by adopting the complement search key. The majority function (MAJ) has multiple input combinations that have the same output. The ‘X’ key merges similar keys to reduce the latency.

4.2 Update Optimization

After searching the matched rows, the accelerator will in-situ update the results. Conventional CAM derived from SRAM updates the data in a row-wise manner, i.e. one WL is activated and data is written into the corresponding row. However, more than one



Logic	Description	Original Key	Optimized Key
OR(OP$_1$,OP$_2$)	OR Logic	01,10,11	00
Geq(OP$_1$,OP$_2$)	Greater & equal	00,10,11	01
Leq(OP$_1$,OP$_2$)	Less & equal	00,01,11	10
MAJ(OP$_1$,OP$_2$,OP$_3$)	Majority	011,110,101,111	011,110,1X1

Figure 5: Search key optimization.

row may match with the search key, thus the accelerator has to write computing results into cells from multiple rows, and a time-consuming row-by-row update is required to fill out all the cells.

To shorten the latency brought by the update procedure, CAP proposes to update multiple rows within 1 step. The optimization is based on one observation that the update results of all match rows are identical because of the same input combinations. Therefore, this procedure is simplified to write single value into matched rows.

To successfully write in data, a configurable voltage selector is implemented, which controls the voltage supply of the bit cells from one column. At normal mode, the PMOS of the voltage head is activated and the supply voltage of each cell is equal to VDD, the data is held within each cell safely. When writing cells in compute mode, the NMOS header is activated and has a threshold voltage (V$_{th}$) drop from VDD. After searching the input combinations, the WLs of all match rows are activated by the sense amplifier and maintained at VDD. The data is written to the cell by the write driver. During the write process, the cell voltage is reduced by V$_{th}$, which weakens the force of the transistor P1. Therefore, the data is successfully written into multiple rows with only 1 cycle.

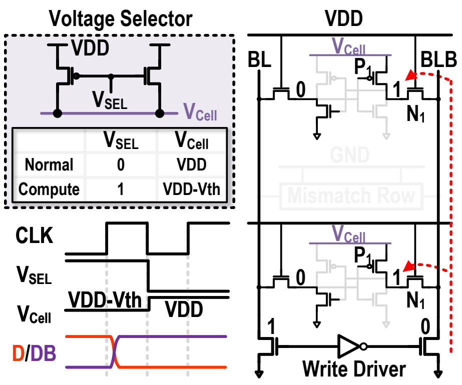


Figure 6: Multi-row update circuits.

4.3 Search-Update Decoupling

In a conventional CAM-based associative processor, the search-update pair is viewed as an atomic operation. Only when the previous pair completes, the following search operation can be issued.

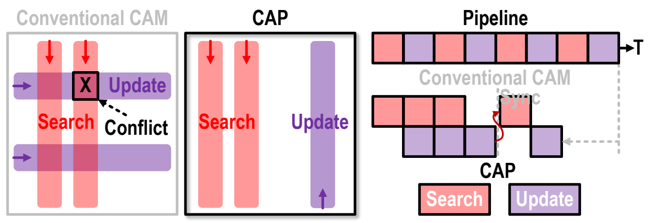


Figure 7: Search update decouple scheme & pipeline.

This sequence order is constrained by the search and update direction. In detail, the search key is shared by entire columns, but the update occurs in a row-by-row manner. Therefore, the orthogonal direction forces the update to wait until the search is complete.

Instead, CAP adopts the column-wise update, which is parallel to the direction of the search. Therefore, the update step is decoupled from the search step and can be performed in parallel. As Fig. 7 shows, the parallel search-update effectively reduces the computing latency when compared with the search-update sequential pair. The synchronization point is required if the data being updated will participate in the following computing, the update-search should follow strict sequential order.

5 EVALUATION

5.1 Hardware Configuration

We complete the circuit design, placement and layout of the CAP accelerator based on 22nm CMOS technology. The CAM array, row/column peripheral, SA, and MIU are implemented from the transistor-level schematic, the entire chip is 0.6 mm^2. The functionality and performance are evaluated by the Cadence Spectre. Fig. 8 shows the layout of one CAM bank.

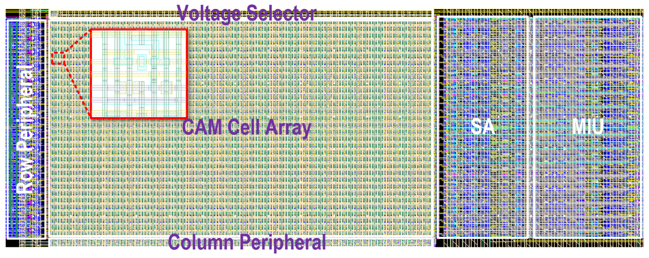


Figure 8: CAP floorplan.

Based on the post-layout simulation, the area and energy breakdown can be acquired as Fig. 9 indicates. The cell array accounts for more than half of the CAM array. To estimate the power breakdown, we activate all the CAM banks. The cell array consumes around half of the power consumption as the mismatch path conducts the current from the ML to the ground. The sense amplifier consumes one-third of power consumption because parallel cross-coupled latches drive ML to VDD or GND. Assuming activating the entire array, the peak energy efficiency and area efficiency of CAP can be acquired as 11.37 TOPs/W and 1.376 TOP/mm^2, respectively.

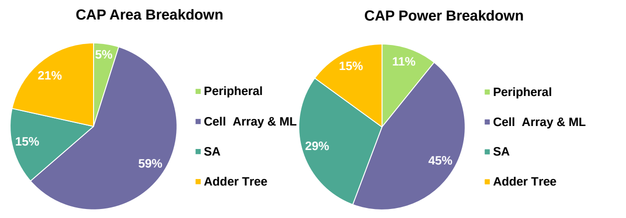
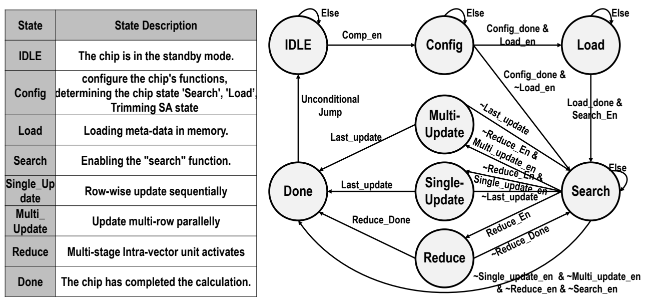


Figure 9: Area and energy breakdown of CAP.



State	State Description
IDLE	The chip is in the standby mode.
Config	configure the chip's functions, determining the chip state 'Search', 'Load', Trimming SA state
Load	Loading meta-data in memory.
Search	Enabling the "search" function.
Single_Update	Row-wise update sequentially
Multi_Update	Update multi-row parallelly
Reduce	Multi-stage Intra-vector unit activates
Done	The chip has completed the calculation.

Figure 10: Finite state machine of CAP.

5.2 Finite State Machine

The CAP contains multiple functional modes, including IDLE, configuration, read, inter-vector search, intra-vector reduce, single-row write, and multiple-row write. As Fig. 10 shows, a finite state machine is employed to switch the mode of CAP, and specific instruction triggers state transition.

6 EXPERIMENTS

This section presents the performance of CAP at the circuit level and application level.

6.1 Circuit Analysis

The CAP has proposed multiple circuit techniques to improve the performance of search and update. Various experiments are conducted to prove the circuit-level functionality and efficiency.

6.1.1 Search circuit. The search operation compares the search key with the stored data. The matched cell maintains cut-off and the matchline holds at high voltage. Instead, the mismatch rows exist conducting path and the ML is discharged to the GND. Fig. 11 proves that the match and mismatch state can be easily differentiated. 10K Monte Carlo tests present that the average ML voltage of the matched rows is 40.276× higher than the mismatch rows.

Fig. 12 indicates the sense time when the search key matches with the stored data. Considering that the output is initially assumed to be mismatched, the match case decides the latency of the CAM array, further affecting the clock period of the entire chip. As the match state can be easily distinguished from the mismatch cases, the sense amplifier easily pulls the voltage of output to VDD. The sense time of CAP reaches < 0.75 ns, assisting CAP in achieving 806MHz peak frequency.

6.1.2 Update circuit. To reduce the update latency, CAP introduces a multi-row updating technique. Compared with the conventional

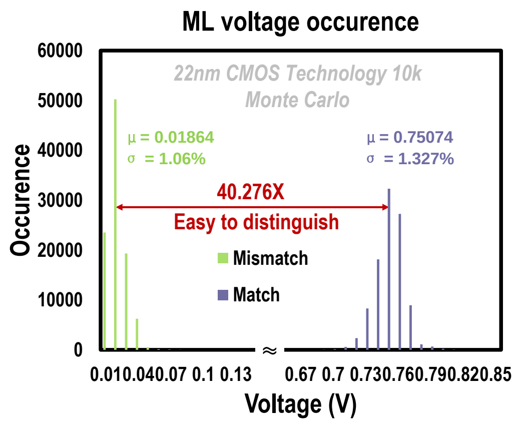


Figure 11: Matchline voltage occurrence.

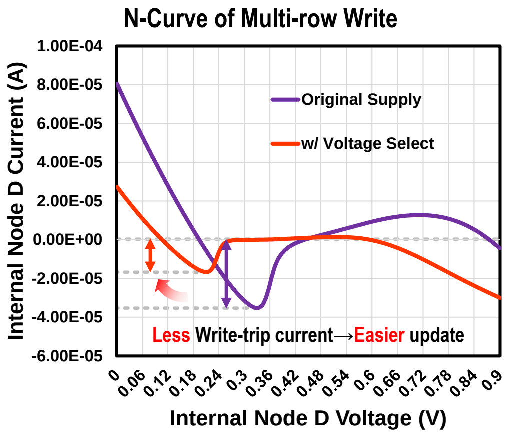


Figure 13: N-curve of the multi-row update mechanism.

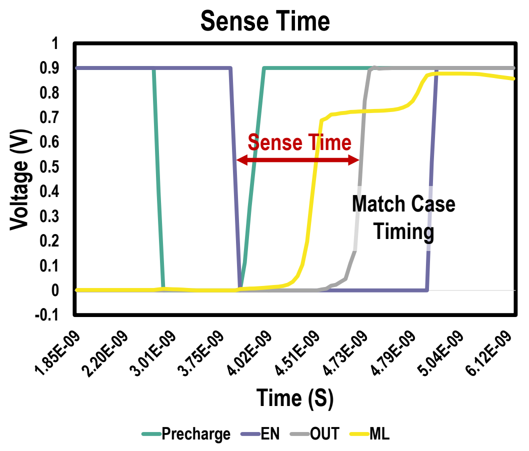


Figure 12: Sense time of CAP at match case.

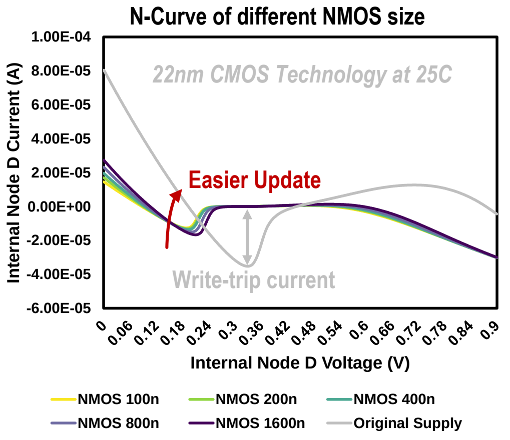


Figure 14: N-curve of circuit with different NMOS sizes.

row-wise write manner, the multi-row update requires more driving force to update the result, and we propose a voltage selector to assist with the writing. Fig. 13 presents the N-curve of the CAP equipped with the voltage selector technique or not. The absolute value of minimum current in the curve, denoted as write-trip current, indicates the ability to write data. Less write trip current means the data in the cell will be easier changed. The result proves that the voltage select technique makes the multi-row writing easier.

The NMOS size decides the threshold voltage drop from VDD, thus affecting the writing behavior. Fig. 14 compares the writing behavior of the voltage selector with different NMOS sizes. Compared with the original voltage supply, all sizes of voltage selectors improve the writing performance. And the larger NMOS size achieves easier writing. Considering the available layout area, the largest NMOS (800n) is selected as the header of the voltage selector.

6.2 Performance Improvement

Equipped with the earlier mentioned circuit design, CAP enables to boost the performance. We evaluate the proposed architecture by using comprehensive workloads. These benchmarks cover a wide range of real-world applications, including geomean, count 1(statistics analysis); FFT, FIR (signal processing); RGB2Gray, census transform (Image Processing); fast walsh-hadamard Transform, DCT transform (data compression); Fibonacci sequence, element addition (arithmetic); ResNet-18, Bert-base (neural network). The experiments demonstrate the improvement in search latency and update latency respectively.

6.2.1 Search Optimization. The CAP optimizes the search behavior with two techniques, merging the similar key and utilizing the complement key. Fig. 16 shows the latency reduction with search optimization, CAP reduces latency by an average of 8.23%. This is because several applications require only 1-cycle search iteration, like the accumulation in the geomean, XOR and accumulation in Census, multiply and accumulation (MAC) in the neural network. It should be noticed that the accumulation operation is easily accomplished by the multi-stage intra-vector unit of CAP. Otherwise, conventional associative accelerators have to rely on time-consuming element-wise addition to accumulate all elements. Except for the applications with 1-cycle operation, the proposed technique improves the latency performance by 15.12%.

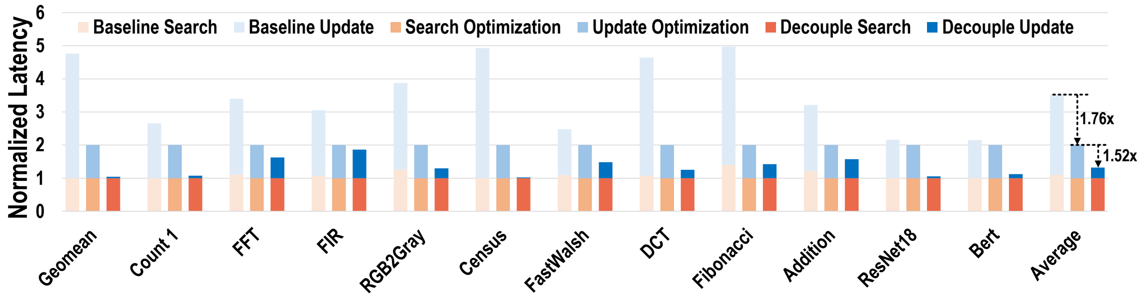


Figure 15: Normalized latency reduction with search-update decouple.

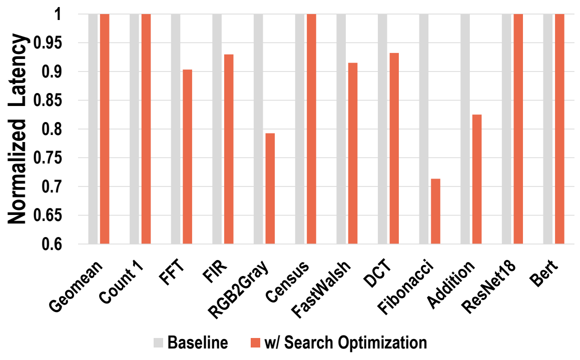


Figure 16: Latency reduction with search optimization.

6.2.2 Update Optimization. The CAP optimizes the update behavior by employing a multi-row updating mechanism. Fig. 15 proves the improvement by adopting this optimization. The average update latency is effectively reduced by 49.7%. The obvious latency reduction is because the conventional CAM adopts a row-by-row update manner, which has to write all match rows after one search iteration. Instead, the CAP completes the update within one step no matter how many rows match with the search key.

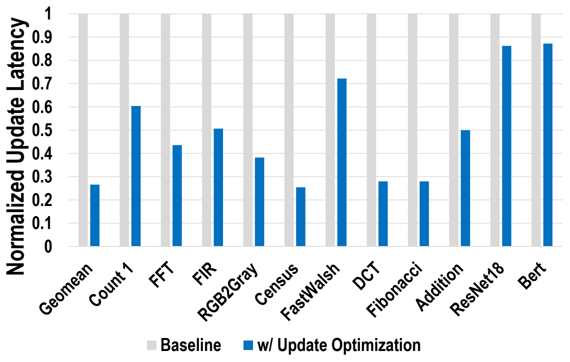


Figure 17: Latency reduction with update optimization.

6.2.3 Search-Update Decoupling Optimization. The conventional associative accelerator follows the search-update sequence order. But CAP supports parallel search-update as it adopts a column-wise updating direction. The latency of the update can be hidden by the pipeline. Fig. 15 normalizes the latency to the design with search optimization and update optimization. Compared with the baseline, the search and update optimizations together improve the performance by 1.76 ×. The search-update decouple pipeline further improves the latency by 1.52 ×.

7 CONCLUSION

In this work, a CAM-based associative accelerator CAP is implemented to achieve efficient general-purpose in-memory computing. Compared with conventional associative processing designs, CAP improves both functionality and performance. Considering the algorithm requirement, CAP first supports intra-vector operation. Next, the CAP reduces the latency of the search behavior and the update behavior of associative processing. And the search-update order is decoupled by CAP to achieve parallel processing. Combining all the techniques, CAP achieves 2.68 × performance improvement over baseline design, 11.37 TOPs/W and 1.376 TOP/mm^2.

8 ACKNOWLEDGMENTS

This work was supported in part by the NSFC under Grant 62125403, and Grant 92164301;in part by the National Science and Technology Major Project under Grant 2022ZD0115201; in part by the National Key Research and Development Program under Grant 2021ZD0114400;Beijing ST Project Z221100007722023;in part by the project funding for the 2022 Special Project on Industrial Foundation Reconstruction and High Quality Development of Manufacturing Industry CEIEC-2022-ZM02-0245 in part by the Beijing National Research Center for Information Science and Technology; and in part by the Beijing Advanced Innovation Center for Integrated Circuits. (Corresponding author: Shouyi Yin.)

REFERENCES

[1] Helena Caminal, Yannis Chronis, Tianshu Wu, Jignesh M. Patel, and José F. Martínez. 2022. Accelerating database analytic query workloads using an associative processor. In *ISCA '22: The 49th Annual International Symposium on Computer Architecture, New York, New York, USA, June 18 - 22, 2022.* ACM, 623–637. https://doi.org/10.1145/3470496.3527435
[2] Helena Caminal, Kailin Yang, Srivatsa Srinivasa, Akshay Krishna Ramanathan, Khalid Al-Hawaj, Tianshu Wu, Vijaykrishnan Narayanan, Christopher Batten, and José F. Martínez. 2021. CAPE: A Content-Addressable Processing Engine. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA).* 557–569. https://doi.org/10.1109/HPCA51647.2021.00054
[3] Tien Ju Yang, Yu Hsin Chen, and Vivienne Sze. 2017. Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*
[4] Hasan Erdem Yantır, Ahmed M. Eltawil, and Fadi J. Kurdahi. 2018. A Two-Dimensional Associative Processor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, 9 (2018), 1659–1670. https://doi.org/10.1109/TVLSI.2018.2827262
[5] Yue Zha and Jing Li. 2020. Hyper-Ap: Enhancing Associative Processing Through A Full-Stack Optimization. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA).* 846–859. https://doi.org/10.1109/ISCA45697.2020.00074