ReCAM In-Storage Implementation of K-Means

R. Kaplan, L. Yavits and R. Ginosar

With the approaching end of Moore's law, non-von Neumann computing paradigms have gained interest in both academia and industry. One example is associative processing based on resistive content addressable memory (ReCAM). Resistive materials are used to build a CAM bit-cell, resulting in very small cell area, which enables ReCAM application in storage.

This work presents a novel ReCAM based storage architecture with processing-in-storage (PRinS). ReCAM combines nonvolatile data storage and massively parallel processing capabilities, which we demonstrate by implementing k-means clustering, a key machine learning algorithm. K-means groups data samples into disjointed sets so that samples in each set have features closer to each other than to data items in other clusters (Fig 3(a)).

Resistive memories store information by modulating the resistance of nanoscale storage elements. They are nonvolatile, free of leakage power, and emerge as long-term potential alternatives to charge-based memories, including NAND flash. Fig. 1(a) shows the ReCAM crossbar. A bitcell, shown in Fig. 1(b), consists of two transistors and two resistive elements. The KEY register contains a data word to be written or compared against. The MASK register defines the active columns for compare, write and read operations, enabling bit selectivity. The TAG register (Fig. 1(c)) marks the rows that are matched by the compare operation and may be affected by a parallel write. The Reduction Tree is an adder tree, enabling quick parallel accumulation of TAG bits and efficient reduction of a vector into a scalar.

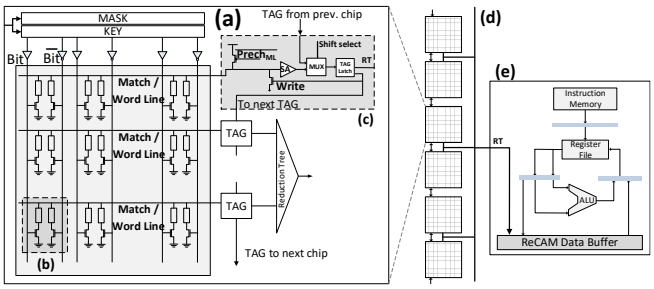
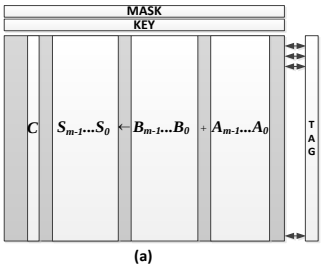


Fig. 1. (a) Single ReCAM crossbar IC. (b) 2T2R ReCAM bitcell (c) TAG logic (d) Daisy-chained ICs (e) Reduction tree and microcontroller

Conceptually, the ReCAM comprises hundreds of millions of rows, each serving as a computational unit. Due to power per die restrictions, the entire array may be divided into multiple smaller ICs, as in Fig. 1(d). The ReCAM storage uses a microcontroller (Fig. 1(e)) which issues instructions, sets the KEY and MASK registers, handles control sequences and executes read requests. In addition, the microcontroller contains the ReCAM buffer, which stores the reduction tree outputs.

ReCAM is a non-von Neumann PRinS accelerator. Any computational expression can be efficiently implemented on the ReCAM using line-by-line execution of the truth table of the expression. Each argument of the expression is matched with the contents of the entire CAM, the matching rows are tagged, and the corresponding expression values are written into the designated fields of the tagged memory rows. For an m-bit argument x, any $f(x)$ has 2^m possible values, therefore the associative computing operation incurs $O(2^m)$ cycles, regardless of the data set size.

More efficiently, arithmetic operations can be performed on ReCAM in a word-parallel, bit-serial manner, reducing compute time from $O(2^m)$ to $O(m)$. For instance, vector addition may be performed as follows. Suppose that two m bit columns hold vectors A and B. The sum of A+B is written onto another m bit column S (Fig 2(a)). A one-bit column C holds the carry bit. The addition is carried out in m single-bit addition parallel steps.



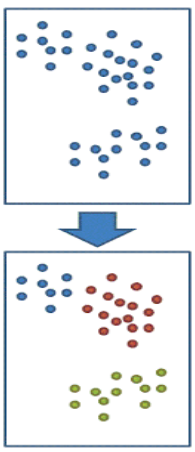
FULL ADDER TRUTH TABLE

Input			Output	
Carry-in C	A	B	Carry-out C	Sum S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(b)

Fig 2. Vector Addition in ReCAM: (a) Memory Mapping; (b) Full Adder Truth Table.

The single-bit addition is carried out in a series of passes, where in each pass one entry of the truth table (a three bit input pattern, Fig 2(b)) is matched against the contents of the $a[*]_i, b[*]_i, c[*]$ bit columns and the matching rows are tagged; the logic result (two-bit output of the truth table as listed in Fig 2(b)) is written into s_i and c bits of all tagged rows. During that operation, all but three input bit columns and two output bit columns of the associative array are masked out in each pass. Overall, eight passes of one compare and one write operation are performed to complete a single-bit addition.



```
Init {
    X → x field in ReCAM;
    c→ x // Initialize each c ∈ C with an x picked randomly from X.
}

Main {
    for i=1 to t {
        // Assignment
        for j=1 to k {  // The assignment loop is  performed in parallel
                        //over all samples.
            d_j = (x − c_j)^2      // This step takes O(m)+O(m^2) cycles:
                                   // subtract a const takes 8m cycles;
                                   // square takes 16m^2 cycles;
            if d_j <d_min {        // copy takes 2m cycles.
                d_min ← d_j
                #cluster ← j
            }
        }
        // Update means
        for j=1 to k {  // This is done in parallel for all samples x ∈ X.
            Reduce: (d_j)      // Reduction tree: log2(N)+m+O(1) cycles
            Reduce: (x)        // for all x assigned to the j^th cluster.
            Calculate new c_j  // Performed by the microcontroller.
        }
    }
}
```

Fig 3. (a) Clustering using k-means, (b) Pseudocode

K-means algorithm pseudocode is presented in Fig 3(b). We simulate k-means on the ReCAM using the cycle-accurate associative processor simulator, employing ReCAM performance figures obtained by SPICE simulations.

To evaluate the efficiency of k-means implementation, we calculate the $Throughput = N \cdot m \cdot D/runtime$, where N is the number of samples (10^6), m is the sample bit width (32), and D is the dimensionality of each sample (1). The simulated ReCAM throughput, along several throughput figures reported by various GPU as well as FPGA-based hardware k-means accelerators is presented in the following table.

Solution	ReCAM	GPU	FPGA1	FPGA2	FPGA3	FPGA4	FPGA5
Throughput	228	0.6	28.7	7.3	5.3	1.6	0.55

ReCAM PrinS k-means implementation may achieve on average 7.5× higher throughput compared with several FPGA based hardware accelerators. It may outperform the high-end GPU implementation by a factor of 377×.

The ReCAM architecture, capable of general purpose associative processing, can also be applied to other challenging problems, such as various machine learning and graph processing algorithms.