ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/285858073

Resistive Associative Processor

Article *in* IEEE Computer Architecture Letters · January 2014

DOI: 10.1109/LCA.2014.2374597

CITATIONS
10

READS
96

4 authors, including:

Shahar Kvatinsky
Technion - Israel Institute of Technology
40 PUBLICATIONS **731** CITATIONS

SEE PROFILE

Ran Ginosar
Technion - Israel Institute of Technology
26 PUBLICATIONS **147** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

In-Data vs. Near-Data Processing: The case for Resistive CAM Storage View project

A Resistive CAM Processing-in-Storage Architecture for DNA Sequence Alignment View project

All content following this page was uploaded by Shahar Kvatinsky on 13 December 2015.

The user has requested enhancement of the downloaded file.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2014.2374597, IEEE Computer Architecture Letters

Resistive Associative Processor

L. Yavits, S. Kvatinsky, A. Morad, and R. Ginosar

Abstract— Associative Processor (AP) combines data storage and data processing, and functions simultaneously as a massively parallel array SIMD processor and memory. Traditionally, AP is based on CMOS technology, similar to other classes of massively parallel SIMD processors. The main component of AP is a Content Addressable Memory (CAM) array. As CMOS feature scaling slows down, CAM experiences scalability problems. In this work, we propose and investigate an AP based on resistive CAM - the Resistive AP (ReAP). We show that resistive memory technology potentially allows scaling the AP from a few millions to a few hundred millions of processing units on a single silicon die. We compare the performance and power consumption of a ReAP to a CMOS AP and a conventional SIMD accelerator (GPU) and show that ReAP, although exhibiting higher power density, allows better scalability and higher performance.

Index Terms—SIMD, Associative Processor, In-Memory Computing, Memristor, Resistive RAM.

1 Introduction

Associative Processor (AP) is a massively parallel SIMD array accelerator [9][10]. AP comprises a Content Addressable Memory (CAM) array and peripheral circuits, and enables storing and processing data at the same location. The execution time of a typical vector operation in an AP does not depend on the vector size, thus allowing efficient parallel processing of very large vectors. The AP is shown to achieve better power efficiency than conventional parallel accelerators [11]. In general, applications with low arithmetic intensity (the ratio of arithmetic operations to memory access [14]), such as sparse linear algebra kernels, are likely to benefit from AP implementation.

As CMOS feature scaling slows down, conventional memory technology experiences scalability problems. In response, resistive memory technologies are explored (*e.g.*, ReRAM based CAM [4][6][12] and STT-MRAM based CAM [5][16]). Resistive memories are expected to scale to much smaller geometries. They are non-volatile, which provides near-zero leakage power. However, ReRAM requires relatively higher write energy and suffers from finite endurance, as compared to CMOS memories.

In this paper, we present the Resistive AP (ReAP), an AP based on resistive CAM. We believe our work is the first to present a resistive CAM based in-memory accelerator that can scale to hundreds of millions of Processing Units (PUs) on a single silicon die, and implement a wide range of massively parallel SIMD workloads. It may not only outperform a CMOS AP, but also conventional massively parallel SIMD accelerators, such as GPUs, in floating point performance and/or power efficiency.

The rest of this paper is organized as follows. Section 2 presents the architecture of the associative processor and principles of associative computing. Section 3 explores the Resistive AP. Section 4 discusses the simulation results and Section 5 offers conclusions.

- *Leonid Yavits, E-mail: yavits@tx.technion.ac.il.*
- *Shahar Kvatinsky, E-mail: skva@tx.technion.ac.il.*
- *Amir Morad, E-mail: amirm@tx.technion.ac.il.*
- *Ran Ginosar, E-mail: ran@ee.technion.ac.il.*

Authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 3200000, Israel.

2 Associative Processing

AP is a non-von Neumann in-memory computing accelerator. The results of basic computing operations are pre-calculated and compiled into a sequence of AP instructions. The operands are stored in the CAM array. In response to an input combination, the result is written into the CAM, typically to multiple locations.

The architecture of an AP is presented in Fig 1(a). The Associative Processing Array (essentially a CAM) comprises of bit cells (Fig 1(b)) organized in bit-columns and word-rows. Typically, a word-row makes a Processing Unit (PU). Several special registers are appended to the associative processing array. The KEY register contains a key data word to be written or compared against. The MASK register defines the active fields for write and read operations, enabling bit selectivity. The TAG register marks the rows that are matched by the compare operation and may be affected by a parallel write. An optional interconnect switch allows the PUs within the AP to communicate in parallel. A Reduction Tree is an adder tree, enabling quick parallel accumulation of TAG bits. It is useful whenever a vector needs to be reduced into a scalar. An example of a CMOS based associative bit cell as well as the tag logic are shown in Fig 1(b). SA is a small sense amplifier. A detailed AP design is presented in [9].

In a conventional CAM, a compare operation is typically followed by a read of the matched data word. In AP, a compare is usually followed by a parallel write into the unmasked bits of all tagged rows.

Any computational expression can be efficiently implemented on an AP using line-by-line execution of the truth table of the expression.

1556-6056 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2014.2374597, IEEE Computer Architecture Letters

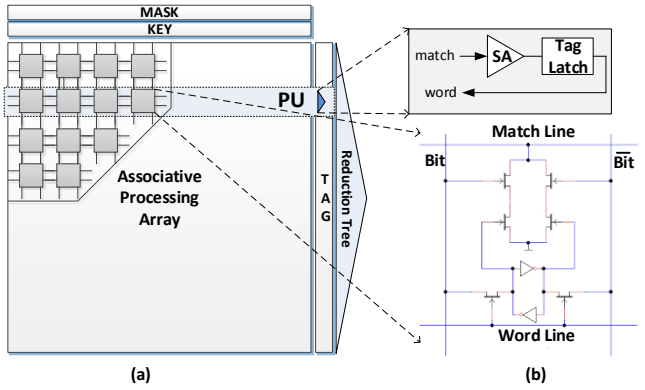


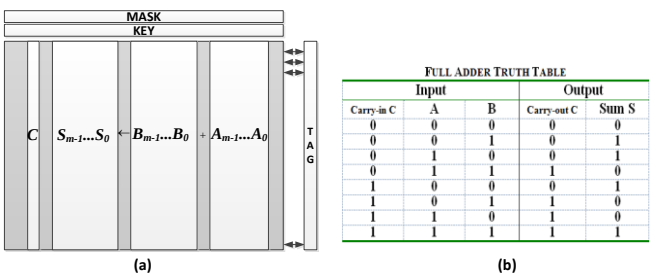
Fig 1. AP: (a) Top-level view; (b) Tag logic and CMOS NOR-type bitcell.

Each argument of the expression is matched with the contents of the entire associative memory, 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. All values of the $f(x)$ truth table are pre-calculated and compiled into a sequence of AP instructions.

More efficiently, arithmetic operations can be performed on AP 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 [2]. Suppose that two m bit columns hold vectors A and B. The sum of A+B is written onto another m bit column S (*cf.* Fig 2(a)). A one-bit column C holds the carry bit. The addition is carried out in m single-bit addition parallel steps (1):

$$c[*] \mid s[*]_i = a[*]_i + b[*]_i + c[*], \quad \forall\, i = 0, \ldots, m-1 \tag{1}$$

where i is the bit index, '$*$' is the word index in the vector, and c and s are, respectively, the carry and sum bits.



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

Fig 2. Vector Addition in AP: (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 (PUs) 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.

A fixed-point m bit addition and subtraction take $16m \propto O(m)$ cycles. Fixed point multiplication and division in AP require $O(m^2)$ cycles [10]. A detailed description of AP architecture and operations can be found in [9].

3 Resistive Associative Processor

Resistive memory (ReRAM) technologies emerge as scalable, long-term alternatives to CMOS memories, including CAM. Resistive memories store information by modulating the resistance of nanoscale storage elements (memristors). Memristors are two-terminal devices, where the resistance of the device is changed by the electrical current or voltage. The resistance of the memristor is bounded by a minimum resistance R_{ON} (low resistive state, logic '1') and a maximum resistance R_{OFF} (high resistive state, logic '0').

A variety of CAM designs, including hybrid CMOS/Magnetoresistive, CMOS/STT-MRAM, and CMOS/memristor, as well as memristor-only schemes have been developed [4][6][8][16][17]. In this paper, we introduce a ReRAM crossbar based Resistive Associative Processing Array (Fig 3(a)), where each bitcell (Fig 3(b)) consists of a pair of neighboring ReRAM bitcells, formed by a nonlinear bipolar memristor that effectively has a diode for preventing sneak paths [4][13]. The second memory bit serves as a complementary bit. The advantage of this approach (relative to previous work) is that it requires no special CAM design. It enables a dual use, where the crossbar array can be operated either as an associative processing array or a conventional ReRAM.

The top level architecture of the resistive AP (ReAP) (Fig 3(a)) closely follows the CMOS based AP architecture of Fig 1, except for the associative processing array which is memristor based. The Word and Match lines, separate in CMOS AP, are combined in the ReAP.

Compare in ReAP is similar to compare operation in resistive CAM [4]. The Match/Word line is precharged and the key is set on Bit and Bit-not lines. In the columns that are ignored during comparison, the Bit and Bit-not lines are kept floating. If all unmasked bits in a row match the key (*i.e.*, when Bit line '1' is applied to an R_{ON} memristor and Bit-not line '0' is applied to an R_{OFF} memristor, or vice versa), the Match/Word line remains high and '1' is sampled into the corresponding TAG bit. If at least one bit is mismatched, the Match/Word line discharges through an R_{OFF} memristor and '0' is sampled into the TAG.

Write operation is performed in two phases. First, the $V > V_{ON}$ voltage (where V_{ON} is a threshold voltage required to switch to the "on" state) is asserted to applicable Bit lines (to write '1's) and Bit-not lines (to write '0's). Second, the $V < V_{OFF}$ voltage (where V_{OFF} is a threshold voltage to switch to the "off" state) is asserted to Bit-not lines (to complement the '1's) and Bit lines (to complement '0's). The write affects only the tagged rows.

1556-6056 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2014.2374597, IEEE Computer Architecture Letters

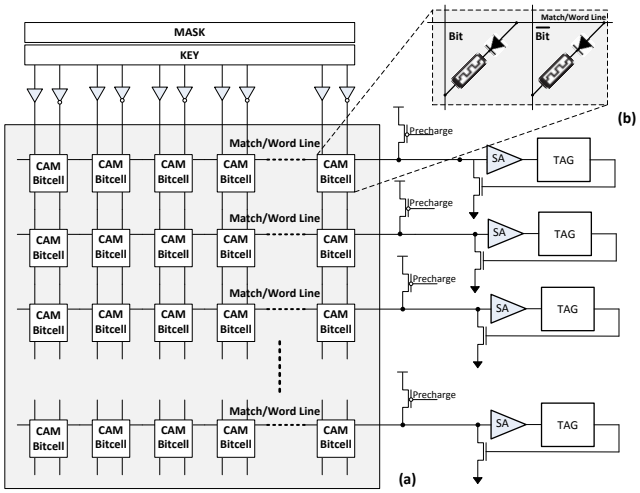


Fig 3. (a) Resistive AP, (b) ReAP bitcell

Compare followed by a write operation are illustrated in Fig 4, which shows a fragment of ReAP storing '0110' in the first row and '0101' in the second row; The ReAP content is compared with the '011x' key and a new '1xxx' key is written in the tagged (first) row.

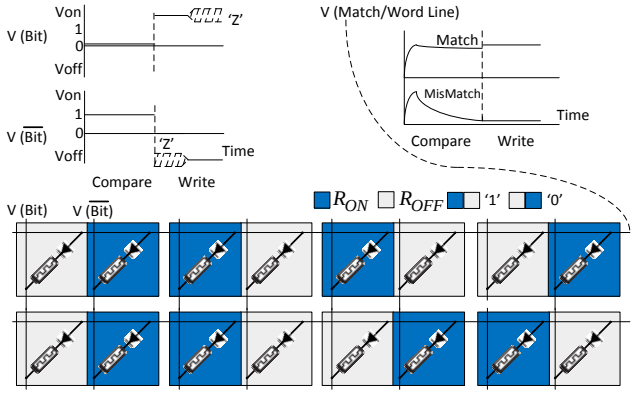


Fig 4. Compare and Write in ReAP

In ReAP, sneak currents affect the compare operation (rather than read operation in a standard ReRAM crossbar). More specifically, there are sneak paths leading from a matching Match/Word Line (which is supposed to retain '1') through neighboring mismatching Match/Word Lines to the ground. The purpose of per-cell diode [4][13] is to terminate such sneak path, so that current can only flow from a Match/Word Line to the ground (through a Bit Line) in one direction.

ReCAM behavior is verified and its performance and energy figures are obtained by SPICE simulations using memristor TEAM model [15].

Utilizing a ReRAM crossbar as CAM enables a ReAp bitcell of $8F^2/k$ footprint (where k is the number of vertically integrated memristor layers [4]). Such level of integration allows placing 100M (k=1) 256-bit PU AP on a single silicon die. To compare, the CMOS based 10-transistor AP bitcell (*cf.* Fig 1(b)) area is approximately $250F^2$ [9], limiting the AP to 4M PU in the same silicon area.

The switching time of memristor may reach the range of a hundred picoseconds [1], allowing GHz AP operation. The energy consumption during compare is less than 1fJ per bit. Unfortunately, the write energy is in the range of few tens of fJ per bit [8], which is prohibitively high for a 100M PU ReAP, as we show below in Section 4. However, write energy is dependent on material the memristor produced of. Discovering more efficient memristor materials will likely result in lower write energy consumption.

Another factor potentially limiting the use of ReAP is the resistive memory endurance, which is in the range of 10^{12} [7]. Given that during arithmetic operations (*cf.* Section 2), write operation typically occurs each second cycle, with an average of 2 out of 256 bits of 1/8th of the AP rows being written (with a close to uniform temporal and spatial distributions of written bits), the probability of a single bit to be written is $(1/2)\cdot(1/8)\cdot(2/256)\approx 5\cdot 10^{-4}$ per cycle. At 1GHz, it limits the endurance-driven MTBF of a ReAP to $\sim 2.05\cdot 10^{6}$ sec, or ~570 hours. However, recent studies predict that the endurance of resistive memories is likely to grow to the $10^{14}-10^{15}$ range [7][8], which extends the endurance-driven MTBF of a ReAP to a number of years.

4 Simulation

We simulate the ReAP using the cycle-accurate AP simulator introduced in [10], employing the resistive CAM performance and power figures obtained by SPICE simulations.

To evaluate the performance and power consumption of ReAP, we use the following set of workloads [3]: N-pairs Black-Scholes option pricing (BSC), N-point Fast Fourier Transform (FFT), and Dense Matrix Multiplication (DMM) of two $\sqrt{N}\times\sqrt{N}$ matrices, where N is the data set size, for simplicity scaled to the AP size (the number of PUs). The parallel portion of each workload is simulated. We simulate all workloads for 16 different values of N, ranging from 2^{12} to 2^{26}. In all simulations, the PU size is 256 bits. We assume the 22nm technology node, operating frequency of 1GHz, and single precision floating point arithmetic.

The simulated performance as a function of the AP area for the CMOS based AP is presented in Fig 5(a). The power consumption results for the CMOS based AP are presented in Fig 5(b). The simulated performance and power consumption as functions of the AP area for the ReAP is presented, respectively, in Fig 5(c) and Fig 5(d). Both performance and power in both CMOS AP and ReAP grow linearly with the AP area (and data set size). Two constraint envelopes are displayed in Fig 5: the maximum area ($250mm^2$) and maximum power ($200W$), defined for example by the chip's thermal design point. Additionally, we show the performance and power consumption of three workloads on GTX480 [16], which area is scaled to 22nm to compare with CMOS AP and ReAP areas.

Our analysis shows that CMOS AP is limited by its maximum area (corresponding to 4M AP PUs), with maximum DMM performance reaching 670 GFLOPs/s.

1556-6056 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCA.2014.2374597, IEEE Computer Architecture Letters

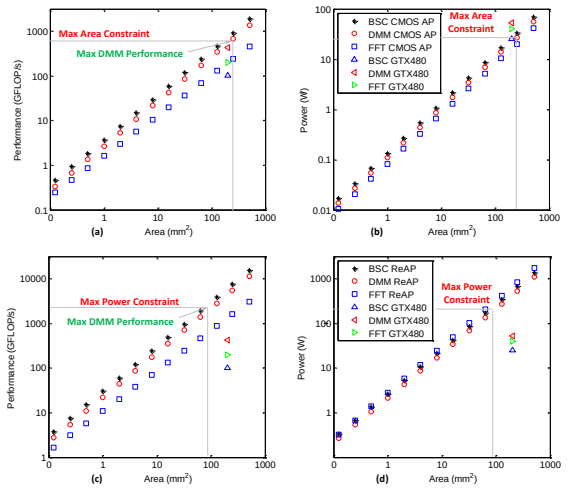


Fig 5. Simulated CMOS AP (a) performance and (b) power, and ReAP (c) performance and (d) power

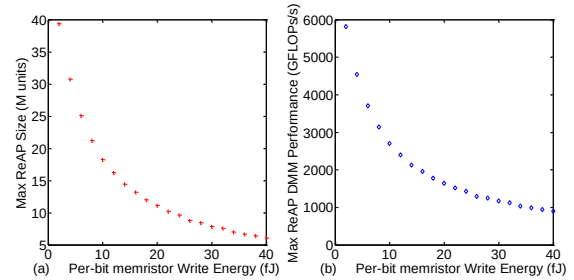


Fig 6. Max ReAP (a) size and (b) performance *vs.* memristor write energy

Due to the dense nature of resistive memory, ReAP scales much better than CMOS AP. However, due to ReAP's relatively high power density (2.7 W/mm^2 vs. 0.14 W/mm^2 for CMOS AP) it is limited by the maximum power. The maximum DMM performance of the ReAP is slightly above 2 TFLOPs/s, achieved at ReAP size of 14M PUs. Both CMOS AP and ReAP exhibit better DMM power efficiency (18 GFLOPs/W and 8 GFLOPs/W respectively) than GTX480 (5 GFLOPs/W).

One noticeable power advantage of ReAP is its near-zero leakage power, while static power consumption in GPUs or CMOS AP cannot be neglected [10]. If a way to reduce the write energy of memristor is found, so that the ReAP performance is no longer limited by its power density, it can reach above 6 TFLOPs/s while processing data sets of above 40M elements in a $250mm^2$ or smaller die. The relationship between the memristor write energy and the maximum achievable size and peak theoretical performance of ReAP is presented in Fig 6.

5 Conclusions

This paper explores a Resistive AP (ReAP), which has the potential to scale the AP from a few millions of PUs to a few hundred millions of PUs on a single silicon die, adhering to the ever growing computing needs of big data era. We compare the performance and power consumption of ReAP to those of traditional CMOS AP and conventional SIMD accelerator (GPU). We conclude that although high power density and finite endurance of memristors limit the potential of ReAP, it allows much better scalability and higher performance compared to CMOS AP and conventional SIMD accelerators. Future progress in development of new materials for memristors will ensure continuous scalability, improved power efficiency, and higher endurance for ReAP.

Acknowledgment

We thank Uri Weiser for inspiring this research. Present work was partially funded by the Intel Collaborative Research Institute for Computational Intelligence and by Hasso-Plattner-Institut.

References

[1] A. Torrezan *et al.* "Sub-nanosecond switching of a tantalum oxide memristor." Nanotechnology 22.48 (2011): 485203.

[2] C. Foster, "Content Addressable Parallel Processors", Van Nostrand Reinhold Company, NY, 1976

[3] E. Chung *et al.* "Single-chip heterogeneous computing: Does the future include custom logic, FPGAs, and GPGPUs?" 43rd Annual IEEE/ACM International Symposium on Microarchitecture, 2010.

[4] F. Alibart, T. Sherwood, D. Strukov. "Hybrid CMOS/nanodevice circuits for high throughput pattern matching applications", IEEE Conference on Adaptive Hardware and Systems, 2011

[5] G. Qing, *et al.* "AP-DIMM: Associative Computing with STT-MRAM," ISCA 2013.

[6] J Li, *et al.* "1Mb 0.41 µm 2 2T-2R cell nonvolatile TCAM with two-bit encoding and clocked self-referenced sensing", IEEE Symposium on VLSI Circuits, 2013.

[7] J. Nickel, "Memristor Materials Engineering: From Flash Replacement Towards a Universal Memory," *Proceedings of the* IEEE International Electron Devices Meeting, December 2011.

[8] K. Eshraghian, *et al.* "Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines", IEEE Transactions on VLSI Systems, 19.8 (, 2011): 1407-1417.

[9] L. Yavits, "Architecture and design of Associative Processor for image processing and computer vision", http://webee.technion.ac.il/publication-link/index/id/633, 1994

[10] L. Yavits, A. Morad, R. Ginosar, "Computer Architecture with Associative Processor Replacing Last Level Cache and SIMD Accelerator", IEEE Transactions on Computers, 2014

[11] L. Yavits, A. Morad, R. Ginosar, "Sparse Matrix Multiplication on Associative Processor", http://webee.technion.ac.il/~ran/papers/YavitsSpMMonAP.pdf

[12] O. Kavehei, *et al.* "An associative capacitive network based on nanoscale complementary resistive switches for memory-intensive computing", Nanoscale 5.11 (2013): 5119-5128.

[13] R. Patel, *et al.* "Multistate Register Based on Resistive RAM", IEEE Transactions on VLSI, 2014.

[14] S. Kamil *et al.*, "An Auto-Tuning Framework for Parallel Multicore Stencil Computations", IEEE International Symposium on Parallel & Distributed Processing 2010, pages 1-12.

[15] S. Kvatinsky, *et al.* "TEAM: threshold adaptive memristor model", IEEE Transactions on Circuits and Systems I, 2013.

[16] S. Matsunaga *et al.* "Standby-power-free compact ternary content-addressable memory cell chip using magnetic tunnel junction devices", Applied Physics Express 2.2 (2009): 023004.

[17] W. Xu, T. Zhang, Y. Chen, "Design of spin-torque transfer magnetoresistive RAM and CAM/TCAM with high sensing and search speed", IEEE Transactions VLSI Systems, 18.1 (2010): 66-74.J.

1556-6056 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

View publication stats