Resistive Address Decoder

L. Yavits, U. Weiser, and R. Ginosar

Abstract—Hardwired dynamic NAND address decoders are widely used in random access memories to decode parts of the address. Replacing wires by resistive elements allows storing and reprogramming the addresses and matching them to an input address. The resistive address decoder thus becomes a content addressable memory, while the read latency and dynamic energy remain almost identical to those of a hardwired address decoder. One application of the resistive address decoder is a fully associative TLB with read latency and energy consumption similar to those of a one-way associative TLB. Another application is a many-way associative cache with read latency and energy consumption similar to those of a direct mapped one. A third application is elimination of physical addressing and using virtual addresses throughout the entire memory hierarchy by introducing the resistive address decoder into the main memory.

Index Terms—Address Decoder, RAM, CAM, Cache, TLB, Virtual Address, Physical Address, Memristors, Resistive memory.

1 Introduction

Content Addressable Memory (CAM) plays an important role in computer architecture. That role could have been even more significant had it not been for the CAM's major disadvantage when compared to RAM: conventional CMOS CAM is highly, and even prohibitively, power-hungry.

We propose a different approach for the rehabilitation of CAM: a Resistive Address Decoder. Consider a typical NAND address decoder (Fig 1), where the address of a memory row is hardwired by connecting either address bitline or inverse bitline to the gate of a NMOS transistor. In a row where the address matches the hardwired pattern, all NMOS transistors "open" and the row is selected. We suggest adding two resistive elements, such as STT-MRAM or memristive devices, to each NMOS transistor in a voltage dividing manner (Fig 2), achieving a two-fold effect: the address pattern in each row becomes programmable rather than hardwired, and the resistive element pair forms a XOR gate, allowing comparison of the input address bit to the bit "programmed" into the resistive elements. These effects enable content addressability, effectively turning an address decoder into a CAM.

In this paper, we present the Resistive Address Decoder. Since it offers read latency and energy consumption very similar to those of a hardwired address decoder, it allows creating fully-associative memory structures at the similar price to one-way associative or direct mapped caches.

Three Resistive Address Decoder applications are presented. A fully associative TLB can be implemented at the same silicon area, read delay and energy consumption as a one-way associative TLB. Likewise, a fully associative cache can be implemented incurring the same area, read delay and energy as a direct mapped cache. Physical addresses may be replaced by virtual addresses in various parts of computer architecture, including memory hierarchies.

The rest of this paper is organized as follows. Section 2 explores the Resistive Address Decoder, Section 3 discusses its potential applications, and Section 4 offers conclusions.

2 Resistive Address Decoder

A conventional address decoder is depicted in Fig 1(a). Typically, address decoding is decomposed into column selection and row decoding; we focus on the latter, being more area- and time-critical. The row decoder consists mostly of the NAND decoder conceptually shown in Fig 1(b). The address bits are hardwired per each word line. The timing of dynamic NAND decoder is defined by the signal propagation time through the series of NMOS devices. Actual NAND decoder design may differ from that concept. For example, it can be implemented by a cascade of NAND segments of three transistors each. During address lookup, such NAND segments are activated sequentially, one after another. Such implementation might be slower but more energy efficient, since in every step, only segments that are selected in the previous step are activated.

While NAND decoders are typically used in non-volatile memory designs [3], other types of address decoding schemes can be implemented in other RAM designs. For example, address decoders of high-speed on-chip SRAMs are often implemented using random CMOS logic.

The Resistive Address Decoder employs resistive programmable devices rather than the hardwired NMOS transistors (Fig 2). The resistive elements are used to program every address bit in every row of the decoder, as well as form XOR gates to allow content addressable search.

Similar approach can be applied to an address decoder implemented using CMOS logic gates.

- *Leonid Yavits, E-mail: yavits@tx.technion.ac.il.*
- *Uri Weiser, E-mail: uri.weiser@ee.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.

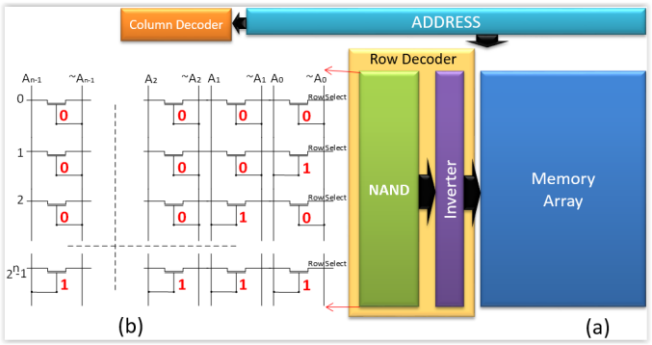

Fig 1. (a) Address Decoder [3]; (b) Dynamic NAND decoder.

2.1 Functionality

Resistive elements are two-terminal devices; their resistance changes by changing the direction of the current through them. That resistance is bounded by a minimum resistance R_{ON} (low resistive state, logic '1') and a maximum resistance R_{OFF} (high resistive state, logic '0').

While a variety of resistive elements exist, one that seems well suited for the Address Decoder Design is a ReRAM element, or memristor. It has off/off ratio of 10^{11}, endurance of 10^{12} and switching speed of 100ps [4].

The programmable Resistive Address Decoder is shown in Fig 2. The wires of Fig 1 are replaced by two resistive elements to the gate of each NMOS transistor. One of the resistive elements should be programmed to R_{ON} while the second (complementary) element should be programmed to R_{OFF}. If the element connected to the bit-line is R_{ON} (and the other one is R_{OFF}), the gate is controlled by the bit-line ('1' at the bit-line enables the transistor). Alternatively, if the other element is R_{ON}, the gate is controlled by the inverse bit-line. The former case is considered as address bit '1', and the latter case is address bit '0'.

To program (write) the decoder, the PE (Program Enable) line is enabled, connecting all mid-points to ground. Appropriate (positive or negative) voltage levels applied to the bit lines (and inverse bit lines) induce programming currents through the resistive elements and achieve parallel programming of all resistive elements in a row.

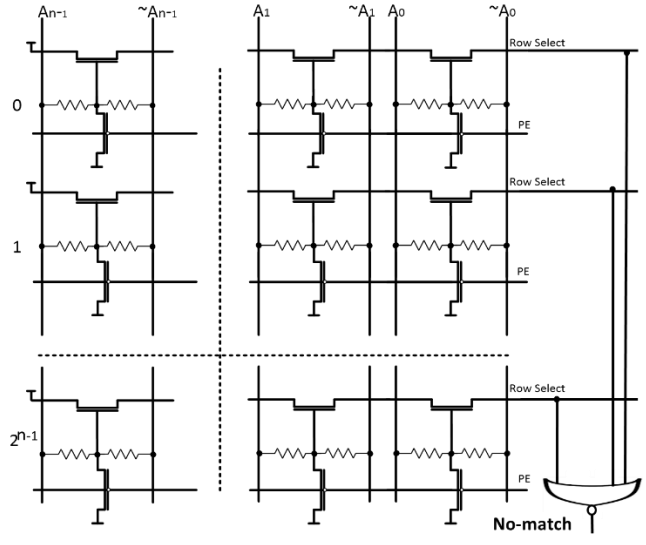

Fig 2. Resistive Programmable NAND Address Decoder

Write operation of each row requires two phases. In one phase, appropriate voltage levels are applied only to bit-lines, and all inverse bit lines are kept disconnected, to assure that only the enabled row is affected. In the second phase, voltage levels are applied only to the inverse bit-lines, while all bit-lines are kept disconnected. To enable the ternary implementation, the "don't care" state can be encoded by programming both resistive elements to R_{OFF}.

During read operation, only the row where all address bits match the address pattern placed on bit- and inverse bit-lines are enabled, and select the corresponding memory rows. Clearly, at most one row should be programmed with a given address. If no matching address is found, a "no match" is signaled (generated by wired-ORing of all address rows). Such "no match" signal can be used to generate page faults.

This read operation is functionally identical to a search in Content Addressable Memory (CAM). In other words, the programmable Resistive Address Decoder functions as CAM, allowing content addressing, and in this case the content is the address.

With the Resistive Address Decoder, memory addresses no longer need to be consecutive, unlike hardwired address decoders. Data can be written anywhere in the memory array. Additionally, the size of the address space becomes arbitrary rather than 2^n.

A number of NOR based resistive CAM and ternary CAM designs have been proposed in recent years [7][8]. In NOR CAM, the match discharges on a mismatch. Since in an address decoder, all rows, but one, mismatch during a read/lookup, the energy consumption is significant. On the contrary, in NAND Resistive CAM, introduced in present work, the mismatching rows do not conduct current. Consecutively, the energy consumption during read is much lower.

A Resistive Address Decoder using memristors has been designed and SPICE-simulated using the TEAM model [5], obtaining timing and energy figures as follows. Main TEAM parameters are presented in Fig 3(a).

2.2 Timing

Read timing of the Resistive Address Decoder is similar to read timing of hardwired address decoders. A short time is added for signal propagation through the resistive element, as compared with propagation over wire.

Memory write is preceded by the address lookup. If the address exists (programmed into the address decoder), the data is written into that memory row. Otherwise, the new address is programmed into an available empty row of the address decoder, simultaneously with writing the data in the same row of the memory array, to reduce the write latency. For a 512-row memristive NAND address decoder, programming delay is 2 ns [6]. This could substantially increase the write latency relative to hardwired address-decoded memory. However, such increased write latency could be mitigated by dividing the memory into separate modules. If a write is followed by a read but they address different modules, then read and write can be executed in parallel. Another mechanism of the write latency mitigation is a write buffer. It uses a simple queuing mechanism

to write data to memory during its free cycles. If a read comes before the data is written in memory, it is read from the write buffer instead.

Furthermore, as shown in Section 3 below, for most potential applications of the Resistive Address Decoder, writes are quite infrequent compared to reads.

2.3 Area

The bit-cell of the Resistive Address Decoder is two resistive elements and two transistors (Fig 2) whereas the bit-cell of a hardwired NAND decoder is 1T. Memristors have reciprocal density of less than $4F^2$, while SRAM memory cells are typically larger than $140F^2$ [4]. Resistive elements can at least partly be placed above CMOS logic. Hence, although the programmable address decoder is somewhat larger than the hardwired one, it may remain small enough to fit within the pitch of the memory array.

2.4 Energy Consumption

Read dynamic energy consumption remains virtually identical to that of a hardwired address decoder.

Write dynamic energy may also include the resistive element programming energy, which may reach 1pJ for memristors [5]. Static power is consumed by current leaking through the resistive element pair. R_{OFF} spans a 10^4—10^{11} Ω range, depending on specific resistive devices and material [4]. Since typically only one memory module is active at a time, the static power consumption could range from 10nW to 100mW.

2.5 Endurance

Endurance (namely the number of times the resistive element may be programmed until it stops functioning correctly) could limit the usage of the programmable Resistive Address Decoder. While endurance of STT-MRAM is close to that of DRAM, the endurance of memristors is probably limited to 10^{12} [4]. To mitigate such endurance, the frequency of write to each memory cell must be lower.

The probability of a write to a certain memory address equals the probability of a memory write times the probability of specific entry to be selected (which in the case of uniform memory utilization equals *1/number_of_entries*). Given the typical size of L2 TLB or L2/L3 cache and typical write frequency in such devices, such probability can be quite low: For a memory structure with the resistive address decoder (with endurance of 10^{12}) to perform for 10 years at 1GHz, the average frequency of write to each address should be ~$1/315{,}000$ cycles^{-1}, so as not to exceed 10^{12} writes. For a 4MB L2 cache with the line size of 64B and 4MB/64B=8192 entries (yielding the probability of a certain address entry selection of $1/8192$), assuming the fraction of memory access instructions is 20% and L1 miss rate is 10% (yielding the L2 write probability of $1/200$), this condition is safely met: $1/200 \times 1/8192 < 1/315000$.

If a memory structure has only few entries, or few tens of entries (for example some L1 DTLBs, or L2 DTLBs with 1G page size), contemporary memristor (with 10^{12} endurance) is not a suitable building block. Such small memory structures, however, are set to benefit very little from the resistive address decoder anyway, since making them fully associative using conventional approach (comparators) is quite cost effective.

3 Applications of the Resistive Address Decoder

In this section, we suggest three potential applications of the Resistive Address Decoder.

3.1 Translation Lookaside Buffer

The hit ratio of TLB is important since L_2 TLB miss may result in a costly page walk. The obvious way of improving the hit ratio is increasing the associativity of the TLB. However, associativity incurs larger silicon area, higher complexity, longer access delay and higher energy consumption.

We propose replacing the TLB CMOS CAM by programmable Resistive Address Decoder, which provides an "affordable" full associativity, as follows.

Read access delay of a fully associative TLB using Resistive Address Decoder is similar to that of a 1-way associative TLB, which is shorter than the access delay of 4-way or 8-way associative CMOS TLB.

Read energy consumption of a fully associative TLB using Resistive Address Decoder is also similar to that of a 1-way associative TLB.

Writing energy of the Resistive Address Decoder is higher than that of the hardwired TLB due to the need to program resistive elements. However, low write frequency in TLBs, around 1000 or fewer writes per million instructions, is typical for many workloads [2]. In that case, the added energy required for programming the resistive devices may be negligible.

In summary, the Resistive Address Decoder converts a 1-way associative TLB into a fully associative TLB, improving hit ratio and reducing or eliminating page walks. The read latency and energy of such fully associative TLB are very similar to those of 1-way associative TLB. The write energy of such TLB are higher, however since write is typically infrequent in TLBs, the impact on overall energy consumption may be minor.

3.2 Cache Memory

A fully associative cache generally delivers higher hit ratio that a direct mapped one. The Resistive Address Decoder enables a fully associative cache with similar lookup time and energy as a direct mapped cache that uses a hardwired address decoder. The silicon cost of a fully associative cache using the Resistive Address Decoder is likely to be similar to that of a direct mapped cache, since resistive elements can be placed above CMOS.

Caches are usually too large to be designed as a single memory array. They are typically partitioned into a number of separate memory banks, with the higher bits of address selecting the bank and the lower bits selecting the memory row within the bank. Each memory bank has its own NAND address decoder. If we make programmable only the NAND address decoder inside the memory bank, we create a many-way set associative cache, where each memory row of a memory bank is a way, and each memory

bank is a set. This many-way set associative cache architecture is presented in Fig 3(b). A large number of ways is possible, e.g., 256 or 512. The hit ratio of such many-way set associative cache is likely to be similar to that of a fully associative cache.

Read (lookup) timing and energy are similar to those of a direct mapped cache. The only difference is due to the discrepancy in the number of index bits in a direct mapped cache *vs*. the number of tag bits in the many-way set associative cache, which affects the number of NMOS transistors in, and hence the propagation delay of, the NAND decoder.

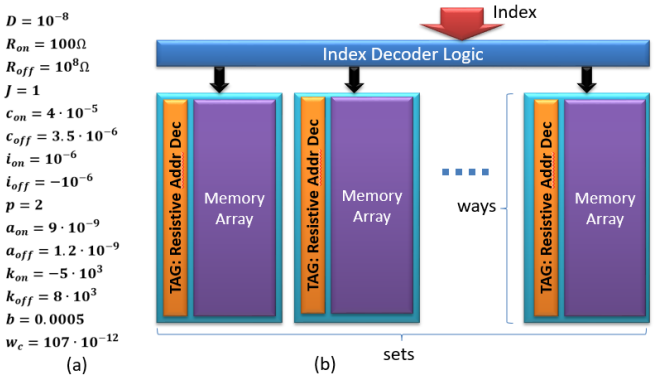

Fig 3. (a) TEAM parameters used in the Resistive Address Decoder simulation (b) The architecture of a many-way set associative cache

Cache replacement could be somewhat costly energy-wise, since per-cell programming energy of a resistive element could reach 1pJ.

In summary, the Resistive Address Decoder converts a direct mapped cache into a fully associative one. The lookup latency and energy of such fully associative cache are similar to those of a direct mapped cache. The write energy of such a cache is higher. However, as we move farther in the cache hierarchy, the miss rate drops, and with it drops the frequency of write. Therefore, the impact of programming the resistive elements at each write on the overall energy consumption should not be critical, especially in higher level caches.

3.3 Elimination of physical address

Virtual addressing is essential to contemporary computers. Unfortunately, virtual to physical address translation takes its toll in terms of performance degradation and excessive energy consumption. TLBs alone may consume up to 13% of a core power [1].

Introducing the Resistive Address Decoder to the main memory may enable the elimination of physical addressing altogether. In a write access, the virtual addresses, together with the corresponding process and thread IDs are transferred to the memory along with the data and are programmed into the Resistive Address Decoder. The delay and energy impact of programming resistive elements can be mitigated by lower write frequency (which could be the case if most memory accesses hit in cache and there is no write-through).

Data is read using the virtual rather than physical address (which no longer exists). Although functionally equivalent to a search in content addressable memory, the read is very similar in terms of access delay and energy consumption to a read from a hardwired address-decoded memory.

If an address that is not programmed in the Resistive Address Decoder is accessed, the "no match" signal is generated, signaling a page fault to the operating system.

Managing memory footprints larger than physical memory also becomes easier. Every newly assigned address is simply programmed into an empty location (which could be marked by a "busy" bit"). On page fault, one page is evicted by the OS, the requested page is loaded in its place, and the virtual address in the Decoder is updated.

4 Conclusions

A Resistive Address Decoder has been presented, where the row address pattern is programmed by resistive elements rather than hard-wired. The Resistive Address Decoder allows comparing the input address with the programmed row pattern, effectively turning the address decoder into a CAM.

The read latency and energy consumption of the Resistive Address Decoder are similar to those of a hardwired decoder. Similarly, its silicon area may be only slightly larger because resistive elements can be placed above CMOS transistors. Thus, the Resistive Address Decoder enables creating fully associative memory structures at the price of direct mapped ones.

We discuss potential applications of the Resistive Address Decoder. One such application is a fully associative TLB at similar silicon area, read delay and energy consumption as with a one-way associative TLB. Another application is a many-way associative cache at the same price as a direct mapped cache. Introducing the resistive decoder to main memory may potentially eliminate physical addressing throughout the entire computer architecture, including the memory hierarchies.

Acknowledgment

We thank Eitan Rosen and Marcelo Yuffe for their valuable input.

References

[1] A. Sodani, "Race to Exascale: Opportunities and Challenges," in MICRO 2011 Keynote

[2] A. Bhattacharjee, M. Martonosi, "Characterizing the TLB behavior of emerging parallel workloads on chip multiprocessors", PACT 2009.

[3] G. Campardo, R. Micheloni, and D. Novosel. "VLSI-design of non-volatile memories". Berlin: Springer, 2005.

[4] J. Yang, D. Strukov, and D. Stewart, "Memristive devices for computing". Nature nanotechnology, 8(1), 13-24

[5] S. Kvatinsky, E. Friedman, A. Kolodny, U. Weiser, "TEAM: threshold adaptive memristor model", IEEE TCAS I, 60(1), 211-221.

[6] L. Yavits, S. Kvatinsky, A. Morad, & R. Ginosar, "Resistive Associative Processor," IEEE Computer Architecture Letters, 14(2), 148-151.

[7] Q. Guo, X. Guo, Y. Bai, E. Ipek, "A resistive TCAM accelerator for data-intensive computing", MICRO 2011.

[8] Meng-Fan Chang et al. "Designs of emerging memory based non-volatile TCAM for Internet-of-Things (IoT) and big-data processing: A 5T2R universal cell", ISCAS 2016.