SCIENCE CHINA
Information Sciences

• LETTER •

April 2024, Vol. 67, Iss. 4, 149403:1–149403:2
https://doi.org/10.1007/s11432-023-3964-4

CATCAM: a 28 nm constant-time alteration TCAM enabling less than 50 ns update latency

Chenchen DENG[1], Tianzhu XIONG[2], Zhaoshi LI[3], Zhiwei LIU[3], Yao WANG[3], Jianfeng ZHU[3], Jun YANG[2], Shaojun WEI[3] & Leibo LIU[3*]

[1]*Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing* 100084*, China;*
[2]*National ASIC System Engineering Center, School of Integrated Circuits, Southeast University, Nanjing* 210096*, China;*
[3]*School of Integrated Circuits, Tsinghua University, Beijing* 100084*, China*

Received 10 August 2023/Revised 5 December 2023/Accepted 27 February 2024/Published online 26 March 2024

Ternary content-addressable memory (TCAM) is an indispensable component of lookup tables in switches or routers. However, it suffers from expensive and inflexible update operations and cannot fulfill the demand for rule updates in switches enabling software-defined networking (SDN). The main reason is that TCAM stores rules from top to bottom in decreasing order of priority for disambiguation. Updating a TCAM is similar to the insertion sort that takes $O(n)$ time, where n is the number of inserted rules. This study proposed a constant-time alteration ternary CAM (CATCAM) that can accomplish both lookup queries and update requests in nanoseconds. It decouples rule priorities from physical addresses by encoding the priority ordering between rules separately in an 8T SRAM array. The traversal is enabled by the computing-in-memory technique by writing incoming rules to empty rows without shuffling existing entries.

Architecture. Figure 1(a) shows the overall architecture of the proposed CATCAM. It is composed of three main components, several match segments (16 in this design), the arbiter for search operations, and the scheduler along with the metadata register for update operations. Each match segment consists of a 256 × 320 static random access memory (SRAM) array named the match matrix and a 256 × 256 SRAM array named the local priority matrix as well as the control logic for normal SRAM interfaces and in-memory operations. The match matrix functions as a conventional TCAM that can accommodate 256 entries of 160 ternary bits while the priority matrix replaces the priority encoder for priority decision. The prototype chip has a capacity of 4K entries (640 Kb). According to the type of lookup table tasks, a task allocator dispatches them to different components. Lookup queries are broadcast to all match segments with valid entries. The search results are collected and arbitrated by the arbiter which is made up of a 16 × 16 SRAM array named the global priority matrix and a multiplexer of 16 channels associated with each match segment. Update requests go to the scheduler. According to the metadata of each match segment, memory commands are issued to corresponding match matrices and priority matrices; meanwhile, the metadata is updated. A necessity of the metadata register is the priority store where all the priorities of existing rules are maintained. Finally, the output would be the position of the single highest priority match for lookup or the position of the inserted rule for update.

Methods. To decouple rule priorities from physical addresses for rule insertion, the priority matrix encodes the priority ordering between rules separately in an 8T SRAM array. As shown in Figure 1(b), each row and column in the priority matrix is associated with a rule in the match matrix of its match segment, and each intersection represents the binary relation of the priority ordering between them. A Boolean value $P_{ij} = 1$ indicates the corresponding rule of row i has a higher priority than the corresponding rule of column j, and "0" indicates the opposite. Each Boolean value is therefore stored in a bit-cell of an SRAM array; the mapping is shown in Figure 1(c).

During lookup, the priority matrix is traversed to filter out the entry with the highest priority. After comparing against the valid entries in the match matrix, the match vector indicates those matched entries. We leverage the idea that if the ith entry has the highest priority, its priority has to be higher than any other matched entry. That is to say, any other matched rule has a lower priority than it, i.e., for any matched entry j, $P_{ji} = 0$. With this idea, according to the match vector, the matched rows and columns are selected, as shown in the shaded part of Figure 1(b). Then each column conducts bit-wise NORs iteratively on selected rows. If any matched entry has a higher priority, which suggests there exists a "1" in the column, the result of NOR is "0". Therefore, the resulting one-hot report vector, whose ith bit is "1", indicates that the ith entry in the match matrix is the matched entry with the highest priority. We leverage the computing-in-memory technique to facilitate such a process. To allow bit-parallel NORs, the match vector is applied to the read bit-lines (RBL) and the read word-lines (RWL) of the priority matrix. Given that rules in the priority matrix are arranged in the same order as the match matrix, there is no need for the regular row or column decoder. As shown in Figure 1(c), if the ith bit in the match vector is "1", the ith RBL is pre-charged while the rest is grounded. Then the ith RWL is activated. If any of the bit-cells connected to a pre-charged RBL carries a "1", the

* Corresponding author (email: liulb@tsinghua.edu.cn)

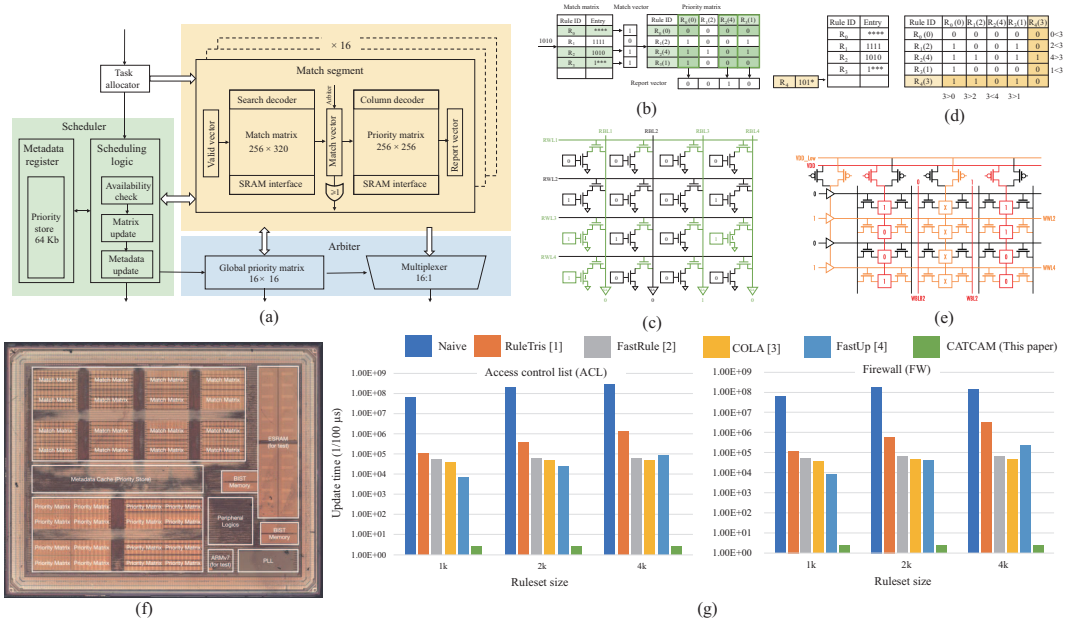

Figure 1 (Color online) (a) CATCAM architecture. (b) The priority relation of rules is encoded in the priority matrix. (c) The priority matrix is mapped to an 8T SRAM array. (d) The priority matrix is updated according to the rule priority and the allocated address. (e) The dual-voltage scheme supports column-wise write. (f) Die photo. (g) Comparison of TCAM update time.

RBL discharges. The RBL stays high only if all the corresponding bit-cells carry "0"s. The results are detected through sense amplifiers (SA) and stored in the report vector. The priority decision is therefore performed in situ and can be finished in a single cycle.

Since the priority ordering is no longer subject to physical addresses, a new rule can be inserted into any available entry. The scheduler then stores and compares its priority against those of existing rules from the priority store using up to 256 comparators to encode the priority ordering in the priority matrix. According to the address allocated for the new rule, the corresponding pair of rows and columns in the priority matrix is to be updated, as shown in Figure 1(d). The row update can use the normal SRAM interface to write the array row-wise. However, the column update requires writing data column-wise, which has to write all bits sequentially under row-wise write. To meet the update time constraint, we adopt a dual-voltage scheme to support column-wise write. The idea is that all "1"s or "0"s in the data can be written to the column concurrently. An additional column decoder selects the column to be written. The data is applied to the write word-line (WWL) instead of the conventional write bit-line (WBL), as shown in Figure 1(e). The WWLs for bit-cells where "1"s ("0"s) have to be written are enabled, and the WBLs and write bit-line bars (WBLBs) for the selected column are driven accordingly to write "1"s ("0"s). Column-wise write takes two cycles to write "1"s and "0"s, respectively. Counting one cycle for row-wise write, updating the priority matrix takes three cycles.

Results. The prototype chip of CATCAM is implemented in the 28 nm CMOS technology and the die photograph is shown in Figure 1(f). The form factor is 2.1 by 2.7 mm. Figure 1(g) [1–4] shows the evaluation performed with ClassBench and various types of rulesets with different sizes. The experimental results show that the rule update time of CATCAM varies from 4 to 40 ns depending on specific tasks, which outperforms the state-of-the-art (SOTA) TCAM update methods by four to six orders of magnitude. With a capacity of 640 Kb, it achieves 470 MHz at 0.9 V and consumes 1.14 fJ/bit per search, which is comparable to the latest TCAM designs. The reason is that SOTA TCAM works still prioritize rules by physical addresses. Constructing and maintaining the dependency relationship with a graph is complicated and time-consuming especially for large rulesets. On the contrary, CATCAM schedules newly inserted rules based on priorities rather than dependencies, eliminates the firmware overhead, and guarantees deterministic update latency that is not subject to the number of existing rules or ruleset characteristics. To our knowledge, CATCAM is the first implementation to fulfill constant time updates and lookup at the same time, making it a promising candidate for future SDN switches.

Acknowledgements This work was supported in part by National Key R&D Program of China (Grant No. 2021YFB3100903) and National Natural Science Foundation of China (Grant No. 62104129).

Supporting information Appendixes A–D. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Wen X, Yang B, Chen Y, el al. RuleTris: minimizing rule update latency for TCAM-based SDN switches. In: Proceeding of IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016. 179–188
2 Qiu K, Yuan J, Zhao J, et al. FastRule: efficient flow entry updates for TCAM-based OpenFlow switches. IEEE J Sel Areas Commun, 2019, 37: 484–498
3 Zhao B, Li R, Zhao J. Efficient and consistent TCAM updates. In: Proceeding of IEEE Conference on Computer Communications, 2020. 1241–1250
4 Wan Y, Song H, Che H, et al. FastUp: fast TCAM update for SDN switches in datacenter networks. In: Proceeding of IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021. 887–897