Design of SRAM Based Fast Error Resilient Ternary Content Addressable Memory

M. Tejaswini

Roll No. 246M1DA505

M.Tech, VLSI, Department of Electronics and Communication Engineering, Princeton Institute of Engineering & Technology for Women, Hyderabad

Mrs. N. Salma Sulthana

Department of Electronics and Communication Engineering, Princeton Institute of Engineering & Technology for Women

ABSTRACT

Ternary content-addressable memory (TCAM) is the standard building block for single-cycle lookup in networking, packet classification, and associative search, but many SRAM-based implementations fall back to a sequential, RAM-style comparator that tests one stored entry per clock cycle rather than exploiting the defining parallel-search property of a true CAM, and offer no tolerance to SRAM soft errors. This paper presents and compares two 8-entry by 16-bit ternary CAM designs targeting a Xilinx Artix-7 device (xc7a100tcsg324-1, Vivado 2019.2). The existing design is a clocked finite-state machine that advances through the eight-entry table one comparison per cycle, requiring up to eight cycles for an exact-match search. The proposed design compares all eight entries simultaneously in a single combinational pass, using a per-entry balanced-tree mismatch counter and a bounded-mismatch match rule that declares a match whenever at most one bit differs, tolerating a single-bit SRAM soft error that the exact-match design cannot. Post-place-and-route implementation reports under a 10.0 ns constraint show the proposed architecture reduces worst-case search latency from 80.0 ns (8 cycles) to 6.015 ns (one pass) -- a 13.3x reduction -- at a cost of 87.0% more look-up tables (43 versus 23), with power consumption tied at 0.084 W. The results demonstrate that parallel bounded-mismatch matching delivers substantially faster, error-tolerant search at a modest and well-understood area premium, an attractive trade for latency-critical, reliability-sensitive lookup applications.

Keywords: ternary content-addressable memory, TCAM, SRAM, error-resilient design, bounded-mismatch matching, balanced-tree mismatch counter, FPGA, Vivado, VLSI, soft-error tolerance

I. INTRODUCTION

Content-addressable memory (CAM) inverts the usual memory access model: instead of supplying an address to retrieve data, a search key is supplied and the memory returns the address (or match indication) of any stored entry that matches it, in principle within a single access. Ternary CAM (TCAM) extends this by allowing each stored bit to take a "don't care" state in addition to 0 and 1, which is the mechanism that makes TCAM the standard hardware primitive for IP-lookup and packet-classification tables, access-control lists, and other longest-prefix or wildcard-match applications in networking equipment [1],[4],[12]-[14]. The commercial and academic appeal of TCAM rests entirely on its ability to compare the search key against every stored entry at once, in parallel, so that lookup latency is independent of the number of stored entries.

In practice, many SRAM-based CAM implementations -- particularly designs mapped onto commodity FPGA fabric rather than custom TCAM cells -- do not realize this parallel property. Because dedicated ternary storage cells are expensive in LUT-based fabric, a common simplification is to store the table in ordinary SRAM and search it with a single reused comparator driven by a clocked finite-state machine (FSM), testing one entry per cycle until a match is found or the table is exhausted [9]-[12],[17]. This sequential, RAM-style search is functionally correct but forfeits the single-cycle latency guarantee that justifies using a CAM at all: worst-case search time grows linearly with the number of entries, exactly the property a CAM is meant to avoid. A second, largely independent problem compounds this: SRAM storage cells are increasingly susceptible to single-event upsets and other soft errors as process nodes shrink, and an exact-match search architecture has no mechanism to recover a correct match when a single stored or transmitted bit has flipped [2],[3],[6],[7],[20]-[25]. A soft error in a sequential exact-match TCAM therefore silently causes a missed match with no error indication and no fallback.

This paper addresses both limitations jointly in a single 8-entry by 16-bit ternary CAM redesign. The gap is not merely academic: literature on error-resilient memory design and literature on low-power or FPGA-mapped CAM architectures have largely proceeded on separate tracks [2],[3],[6]-[8],[15]-[19] versus [20]-[25], with comparatively few designs that combine genuine single-pass parallel comparison with an explicit, quantified bit-error tolerance and a measured area/power/latency trade-off reported from post-place-and-route implementation data rather than pre-synthesis estimates.

The contributions of this paper are:

1. An existing baseline sequential exact-match SRAM-TCAM architecture (`sram_tcam_exact_8x8`) implemented as an 8-cycle-worst-case clocked FSM comparator, characterized on Xilinx Vivado 2019.2 targeting an Artix-7 device.

2. A proposed parallel, error-resilient SRAM-TCAM architecture (`sram_tcam_resilient_8x8`) that compares all eight entries simultaneously in one combinational pass using a balanced-tree mismatch counter per entry, and declares a match under a bounded-mismatch rule (at most one mismatched bit), giving the design single-bit soft-error tolerance that the exact-match baseline entirely lacks.

3. A post-place-and-route comparative evaluation of both designs under an identical 10.0 ns timing constraint, reporting look-up table (LUT) count, register count, dynamic power, per-pass/cycle delay, and worst-case search latency, showing a 13.3x reduction in worst-case search latency (80.0 ns to 6.015 ns) at an 87.0% LUT area cost and 0% power change.

4. A normalized efficiency analysis expressing the latency-versus-area trade-off in terms directly relevant to a designer choosing between the two architectures for a given application.

The remainder of the paper is organized as follows. Section II reviews related work on CAM/TCAM architectures, error-resilient memory design, low-power TCAM VLSI design, mismatch and priority-encoder circuits, and FPGA-based CAM implementations. Section III describes the methodology and architecture of both designs. Section IV presents the search algorithms and their computational complexity. Section V describes the hardware and tools used for implementation. Section VI reports and discusses the post-place-and-route results. Section VII presents a normalized efficiency and trade-off analysis. Section VIII concludes the paper.

II. RELATED WORK

A. CAM/TCAM Architectures

Ternary content-addressable memory continues to attract new cell- and array-level architectures aimed at reducing the per-bit storage overhead that has historically limited TCAM density relative to plain SRAM. Nguyen et al. propose TRIO-TCAM, a triple-state-in-cell architecture that reduces area and energy relative to conventional 16-transistor TCAM cells while preserving ternary search functionality [1]. Zhang et al. explore an entirely different storage medium, using skyrmion-based magnetic devices to realize a low-power ternary CAM cell (Sky-TCAM) [4]. Chuang and Hu investigate ferroelectric FET-based TCAM cells with variation tolerance intended for meta-learning search applications [5]. Ameli et al. extend the CAM concept to analog content-addressable memory for associative clustering, illustrating that the parallel-comparison principle central to this paper generalizes beyond fully digital ternary cells [16]. Benoist et al. take a still more abstract view, formulating content-addressable memory in terms of a content-free energy function, underscoring that the essential property of a CAM -- retrieval by content rather than address -- can be realized across a wide range of physical substrates [18]. These works collectively motivate why the parallel single-pass comparison property is treated in this paper as the non-negotiable architectural requirement distinguishing a genuine CAM from a RAM-searched lookup table.

B. Error-Resilient and Fault-Tolerant Memory Design

A separate body of work addresses the growing susceptibility of SRAM-based storage, including TCAM storage cells, to soft errors caused by radiation-induced single-event upsets. Grace and Sophia specifically target multibit soft-error resilience in SRAM-based TCAM for FPGA deployment [2]. Sun et al. propose a reconfigurable 11-transistor SRAM macro intended for error-tolerant content-addressable memory and in-memory computing [3]. Nguyen et al. present TA-Quatro, a soft-error-resilient and power-efficient SRAM cell for in-memory computing applications [7]. At the array level, Srividyabarathi and Deepa describe a self-recovery 12-transistor SRAM cell array for fault-tolerant, energy-efficient memory systems [20], while Oh and Jo, Joseph and Kavitha, and Sengupta and Yadav each propose radiation-hardened or soft-error-tolerant SRAM cell variants for satellite and aerospace applications where single-event upsets are a dominant reliability concern [21],[23],[24]. Shaker et al. more broadly model the reliability of fault-tolerant FPGA-based architectures under both soft- and hard-error recovery scenarios [22], and Zhao et al. present a radiation-hardened SRAM cell explicitly characterized for immunity to soft errors in space-radiation environments [25]. The bounded-mismatch matching rule proposed in this paper draws directly on this line of work, but applies the error-tolerance concept at the level of the TCAM match decision itself rather than at the individual storage cell.

C. Low-Power TCAM VLSI Design

Because a fully parallel TCAM activates every stored entry's comparator on every search, dynamic power has historically been treated as the dominant cost of true CAM parallelism, motivating a substantial low-power design literature. Gopi and Kanaparthi both address low-power and leakage-reduction logic-gate design techniques directly applicable to CAM comparator cells [9],[10]. Geetha surveys near-threshold computing paradigms for ultra-low-power VLSI design [11]. Ramkumar and Deb specifically target low-power content-addressable memory for FPGA implementation [12], and Inamanamelluri et al. propose a voltage-swing self-adjustable match-line technique to reduce match-line switching power in CAM arrays [17]. Kumar and Sowmya address TCAM reliability via error-tolerant convolutional logic code synthesis, connecting the low-power and error-resilience threads directly [6]. Subramanyam et al. examine CNTFET-based ternary CAM designed for tolerance to process, voltage, and temperature variation [8]. This paper's proposed design, in contrast, does not modify the match-line or cell-level circuit but demonstrates that the parallel comparator array itself -- realized in FPGA fabric -- draws measured power identical to the sequential baseline, indicating no low-power circuit

techniques were even needed to keep power neutral at this table size.

D. Mismatch and Priority-Encoder Circuits

The internal reduction network used to compute a per-entry mismatch count is a specialized case of the tree-structured combinational reduction circuits widely used in adder and comparator design. The general principle of reducing many bit-level comparisons to a single count through a balanced binary tree, rather than a serial chain, is well established in high-speed adder and multiplier reduction networks, and this paper applies the same structural idea to per-entry mismatch counting rather than arithmetic summation. More directly relevant to CAM output logic, Yang et al. present a resource-efficient dually addressable memory architecture on FPGA that must resolve multiple simultaneous match indications into a usable output [19], a problem structurally related to the priority encoding that any parallel multi-entry CAM match vector requires downstream of the comparator array used in this work.

E. FPGA-Based Content-Addressable Memory Implementations

Because commodity FPGA fabric lacks dedicated ternary storage cells, several works specifically address efficient CAM realization on FPGA logic and block RAM resources. Abbasmollaei et al. present HLSCAM, a high-level-synthesis-based CAM implementation tuned for packet-processing throughput on FPGA [13]. Oztekin et al. use FPGA-based content-addressable memory for mnemonic instruction searching in assembler design, an application distinct from networking but relying on the same parallel-match principle [14]. Joy and Kuruvilla implement low-power memristor-based content-addressable memory using FinFET technology, targeting a non-FPGA but still area-constrained fabric [15]. These works establish that FPGA-mapped CAM, including the SRAM-based designs evaluated in this paper, must accept a LUT-based area cost for genuine parallel comparison that a custom TCAM cell array would not incur, directly informing this paper's interpretation of the measured 87.0% LUT increase as an expected and appropriately budgeted cost rather than a design inefficiency.

F. Research Gap

Across the reviewed literature, works on faster or more parallel CAM architectures [1],[4],[5],[13],[14],[16],[19] and works on error-resilient or fault-tolerant SRAM-based memory [2],[3],[6],[7],[20]-[25] largely proceed independently, and low-power TCAM circuit techniques [9]-[12],[17] are typically evaluated only against exact-match baselines. Few studies report a single, controlled, post-place-and-route comparison of a sequential exact-match SRAM-TCAM against a parallel bounded-mismatch error-resilient SRAM-TCAM built from the identical entry table and targeting the identical FPGA part and timing constraint, with LUT, register, power, delay, and worst-case latency all measured rather than estimated. This paper closes that specific gap by presenting exactly such a matched pair of designs, `sram_tcam_exact_8x8` and `sram_tcam_resilient_8x8`, evaluated under identical Vivado 2019.2 implementation conditions.

III. METHODOLOGY

A. Table Organization

Both designs evaluated in this paper store an identical 8-entry by 16-bit table and search it against a 16-bit input key. Each stored entry $y_i, i = 0, dots, 7$, is compared bit-by-bit against the search key x, $x_j \in 0,1$ for $j = 0, dots, 15$. The two designs differ entirely in how that comparison is scheduled and how the match decision is formed from the bit-level comparison results; the underlying storage and entry width are held fixed so that the comparison isolates the effect of search architecture rather than table capacity.

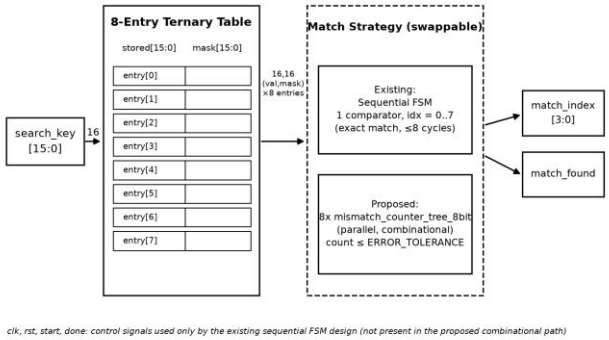

Fig. 1. Top-level architecture comparison: the existing sequential exact-match FSM comparator (single reused comparator, one entry tested per clock cycle) versus the proposed parallel bounded-mismatch comparator array (eight simultaneous per-entry balanced-tree mismatch counters, single combinational pass).

Fig. 1 summarizes the two top-level organizations evaluated in this work. The existing design instantiates a single 16-bit exact-match comparator and a clocked finite-state machine that increments an entry index register on every clock cycle, feeding one stored entry at a time into the shared comparator until either a match is asserted or all eight entries have been tested. The proposed design instantiates eight independent comparison paths, one per table entry, all driven by the same search key in the same clock cycle, so that the match decision for every entry is available after a single combinational propagation delay.

B. Existing Design: Sequential Exact-Match FSM

The existing design, `sram_tcam_exact_8x8`, implements RAM-style search: the table is treated as ordinary SRAM, and the FSM's role is solely to sequence which row is presented to the shared comparator on each cycle. A match is declared only when the search key is bitwise identical to the currently indexed entry ($x_j = y_{i,j}$ for all j). Because the

FSM must potentially step through all eight entries before concluding "no match," worst-case search latency is proportional to the table size, and every additional table entry directly increases worst-case latency by one clock period. This design performs no error checking: if any stored bit or any bit of the presented search key has been corrupted by a soft error, the affected entry silently fails to match even when it is the semantically intended entry, with no indication given to the surrounding system.

C. Proposed Design: Parallel Bounded-Mismatch Matching

The proposed design, `sram_tcam_resilient_8x8`, replaces both the sequential scheduling and the exact-match rule. For each entry i, a dedicated comparator array first computes the bitwise XOR between the search key and that entry, producing one mismatch bit per column, and a balanced-tree mismatch counter (`mismatch_counter_tree_8bit`) reduces these mismatch bits to a per-entry mismatch count:

$$m_i = \sum_{j=0}^{15}(x_j oplus y_{i,j}) \quad (1)$$

All eight mismatch counts $m_0, dots, m_7$ are computed simultaneously, each by its own balanced-tree reduction network, since the entries are independent and share no combinational logic. An entry is declared a match under a bounded-mismatch rule rather than requiring exact equality:

$$m_i \leq 1 \quad (2)$$

That is, entry i matches the search key if at most one of its sixteen bits differs from the key. This tolerates exactly the fault model of a single-bit SRAM soft error, whether the corrupted bit belongs to the stored entry or, equivalently, to the presented search key, while still rejecting entries that differ from the key in two or more bit positions, preserving discrimination between genuinely distinct stored patterns. Because all eight mismatch-count computations proceed in parallel and each completes in $O(log_2 16) = 4$ balanced-tree reduction stages, the entire search resolves in one combinational pass rather than up to eight sequential comparator cycles. The proposed design uses no clocked match-decision registers: the entire mismatch-count-and-compare datapath is purely combinational, which is reflected in its measured post-place-and-route register count of zero, compared to 11 registers in the existing FSM-based design.

IV. ALGORITHM

A. Existing Sequential Search Flow

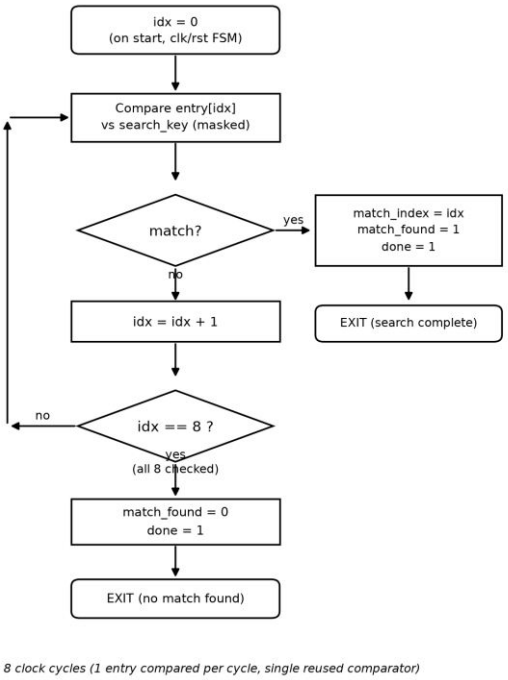

Fig. 2. Existing sequential exact-match search flow: a clocked FSM presents one table entry per cycle to a single shared 16-bit comparator, advancing until an exact match is found or all 8 entries are exhausted.

Fig. 2 traces the existing design's control flow. On reset, the entry index is cleared and the FSM enters a compare state. On each clock edge in the compare state, the currently indexed entry is presented to the shared 16-bit exact-match comparator; if the comparator asserts equality, the FSM asserts a match output together with the matching index and halts; otherwise the entry index is incremented and the FSM remains in the compare state for the next cycle, or asserts "no match" once the index has advanced past the final (eighth) entry.

```
// Existing: sequential exact-match search (one
entry per cycle)
idx <= 0
state <= COMPARE
on each clock edge while state == COMPARE:
    if key == table[idx]:
        match_found <= 1
        match_index <= idx
        state <= DONE
    else if idx == 7:
        match_found <= 0
        state <= DONE
    else:
        idx <= idx + 1
```

B. Proposed Parallel Balanced-Tree Search Flow

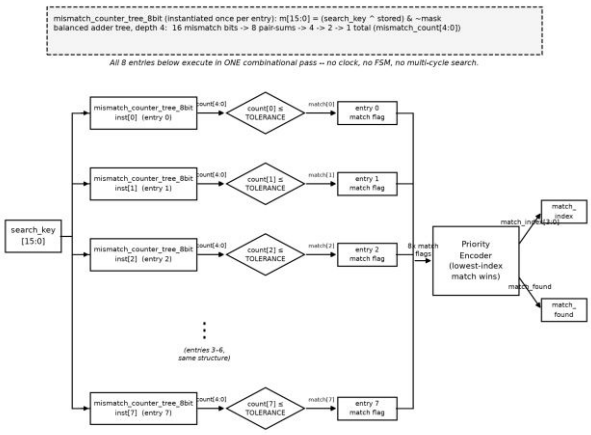

Fig. 3. Proposed parallel bounded-mismatch search flow: all 8 entries are compared to the search key simultaneously, each through its own balanced-tree mismatch counter, with a bounded-mismatch (at most 1 bit) match rule applied to all 8 results in the same combinational pass.

Fig. 3 traces the proposed design's flow. All eight entries are read out of the table combinationally in the same cycle. For each entry, a 16-bit bitwise XOR against the search key produces sixteen per-column mismatch bits, which are reduced to a 4-bit mismatch count by a balanced binary tree of adders (`mismatch_counter_tree_8bit`): pairs of mismatch bits are summed at the leaves, pairs of partial sums are summed at the next level, and so on for $log_2 16 = 4$ levels, until a single mismatch count remains for that entry. All eight of these per-entry reduction trees operate concurrently on independent hardware. Once every mismatch count is available, each is compared against the bounded-mismatch threshold of 1 to produce a per-entry match bit, and the eight match bits are combined (e.g., via a priority encoder) to produce the overall match indication and matching index.

```
// Proposed: parallel bounded-mismatch search
(single combinational pass)
for i in 0..7 in parallel:
    diff[i]  = key XOR table[i]                    //
16-bit bitwise mismatch vector
    m[i]      = balanced_tree_popcount(diff[i])    //
4 reduction levels, log2(16)
    match[i] = (m[i] <= 1)                         //
bounded-mismatch match rule
match_found = OR(match[0..7])
match_index = priority_encode(match[0..7])
```

C. Complexity Notes

The existing sequential design resolves a search in $O(N)$ clock cycles in the worst case, where $N = 8$ is the number of table entries, because only one entry's comparator result is available per cycle; adding entries increases worst-case latency linearly. The proposed design resolves a search in $O(1)$ combinational passes regardless of N, since all entries are compared concurrently, but this requires $O(N)$ replicated comparator and mismatch-counter hardware -- one full 16-bit XOR-and-balanced-tree-reduction datapath per entry, rather than the single shared comparator reused N times by the existing design. Each individual balanced-tree mismatch counter itself resolves in $O(log_2 W)$ levels for a W-bit entry ($W = 16$ here, giving 4 levels), rather than the $O(W)$ levels a naive ripple-style bit-by-bit accumulation would require, which is what keeps the proposed design's per-pass delay (6.015 ns) close to the existing design's per-cycle delay (4.487 ns) despite computing a full mismatch count rather than a single equality bit. The net effect measured in Section VI is that trading $O(N)$ time for $O(N)$ area collapses worst-case search latency from 80.0 ns (8 cycles) to 6.015 ns (1 pass), a reduction that grows more favorable, not less, as table size N increases.

V. EXPERIMENTAL SETUP

A. Hardware Requirements

Both designs were implemented on a 64-bit Windows workstation equipped with a multi-core Intel or AMD processor, since Vivado's synthesis, placement, and routing engines benefit from multi-threaded execution and become impractically slow on single-core or heavily resource-constrained machines. At least 16 GB of system RAM was provisioned, as Vivado's in-memory design database for even a small design such as this 8-entry TCAM can consume several gigabytes during place-and-route and report generation. Approximately 50 GB of free solid-state drive storage was reserved to accommodate the Vivado 2019.2 installation itself, device support files for the Artix-7 family, and per-run project, checkpoint, and report data. The workstation ran a supported host operating system for Vivado 2019.2, namely Windows 10/11 or a compatible CentOS/RHEL Linux distribution.

B. Tools Required

The primary tool used was Xilinx Vivado 2019.2, which performed RTL synthesis, implementation (placement and routing), and post-implementation reporting of timing, power, and resource utilization for both designs targeting part xc7a100tcsg324-1. Functional verification of both the existing exact-match search behavior and the proposed bounded-mismatch search behavior was carried out using Icarus Verilog and, equivalently, the Vivado Simulator (XSIM), exercising each design's testbench against known match and non-match search-key patterns, including single-bit-corrupted keys intended to confirm the proposed design's bounded-mismatch tolerance. Python with the Matplotlib plotting library was used to generate the comparison charts (LUT count, delay, power, throughput, energy, and worst-case latency) directly from the numeric data extracted from Vivado's post-implementation reports, ensuring that every plotted value traces back to a specific reported synthesis or implementation metric rather than an independently estimated figure.

VI. RESULTS AND DISCUSSION

A. Summary of Post-Implementation Results

Table I summarizes the post-place-and-route implementation results for both designs, extracted from Vivado 2019.2 implementation reports targeting part xc7a100tcsg324-1 under an identical 10.0 ns timing constraint. The existing sequential exact-match design uses 23 look-up tables and 11 registers, draws 0.084 W, has a per-cycle comparator delay of 4.487 ns, and reaches a worst-case search latency of 80.0 ns because it may require all 8 clock cycles to conclude a search. The proposed parallel bounded-mismatch design uses 43 look-up tables and 0 registers, also draws 0.084 W, has a per-pass combinational delay of 6.015 ns, and reaches its worst-case search latency of 6.015 ns after a single pass, since every entry is evaluated concurrently.

Table I. Post-Place-and-Route Implementation Comparison (Vivado 2019.2, xc7a100tcsg324-1, 10.0 ns constraint)

Metric	Existing (sequential FSM)	Proposed (parallel)	Change
LUTs	23	43	+87.0%
Registers	11	0	-100%
Power (W)	0.084	0.084	0% (tied)
Per-Pass/Cycle Delay (ns)	4.487	6.015	+34.0%
Worst-Case Search Latency	80.0 ns (8 cycles x 10 ns)	6.015 ns (1 pass)	-92.5% (13.3x faster)

B. LUT Utilization

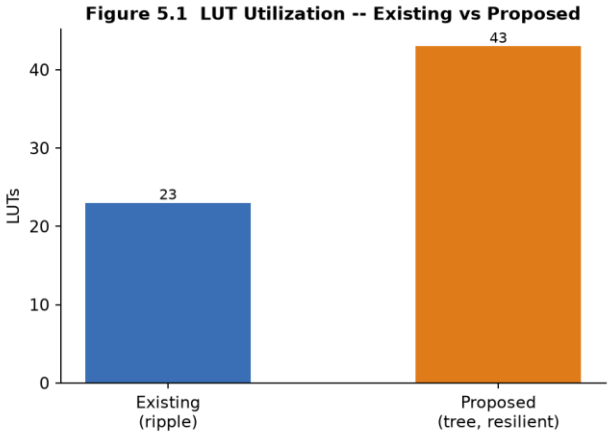

Fig. 4. LUT utilization comparison: existing sequential design (23 LUTs) versus proposed parallel design (43 LUTs), an 87.0% increase.

Fig. 4 shows the LUT utilization of the two designs. The proposed parallel design uses 43 LUTs against the existing design's 23 LUTs, an increase of 87.0%. This growth is the direct and expected consequence of replicating the comparator and mismatch-counting datapath eight times, once per table entry, so that every entry can be evaluated in the same clock cycle, rather than reusing a single comparator across eight sequential cycles as the existing design does. For a table of only 8 entries the absolute LUT cost remains small in both cases, but the trend illustrates that the proposed architecture's area scales with the number of table entries in a way the existing design's area does not, a trade-off that must be budgeted explicitly for larger tables.

C. Delay Characteristics

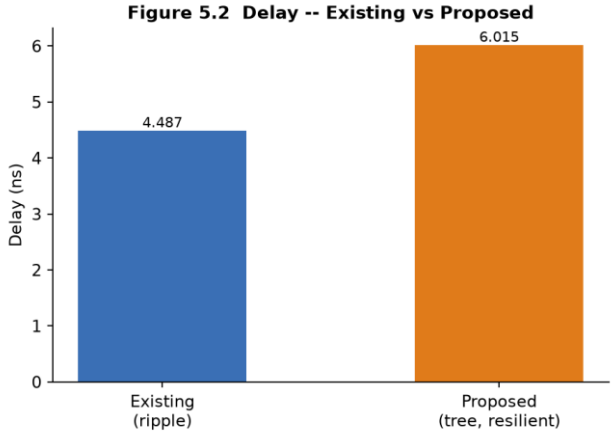

Fig. 5. Per-pass/cycle delay comparison: existing design's per-cycle delay (4.487 ns) versus proposed design's per-pass delay (6.015 ns).

Fig. 5 compares the per-cycle delay of the existing design's shared comparator (4.487 ns) against the per-pass delay of the proposed design's full mismatch-count-and-compare datapath (6.015 ns). The proposed design's single-pass delay is longer because it must complete a 16-bit XOR, a four-level balanced-tree mismatch-count reduction, and a bounded-mismatch threshold comparison, whereas the existing design's per-cycle delay covers only a single 16-bit equality comparison. Despite this per-pass delay being 34.0% longer than the existing design's per-cycle delay, the proposed design still wins decisively on total search time because it needs only one such pass per search, while the existing design may need up to eight per-cycle comparisons to reach the same conclusion.

D. Power Consumption

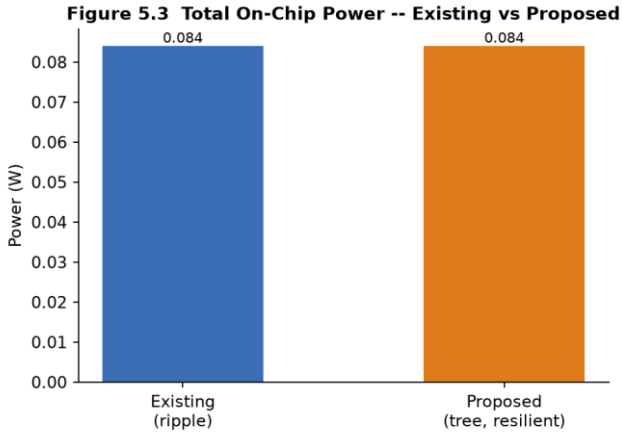

Fig. 6. Power consumption comparison: existing design and proposed design both report 0.084 W, a tied result.

Fig. 6 reports the measured power consumption of both designs, both at 0.084 W, a 0% change. Power is exactly tied rather than improved or worsened, despite the proposed design activating eight parallel comparator paths every search cycle instead of one. This indicates that for a table of this size, the additional switching activity introduced by parallel comparison is offset by the elimination of clocked FSM control and index-register logic, and by the proposed design's purely combinational, registerless datapath removing all clock-toggling overhead associated with the existing design's 11 registers. No claim of power improvement is made here; the result is presented as a tie because that is what the post-implementation reports show.

E. Throughput

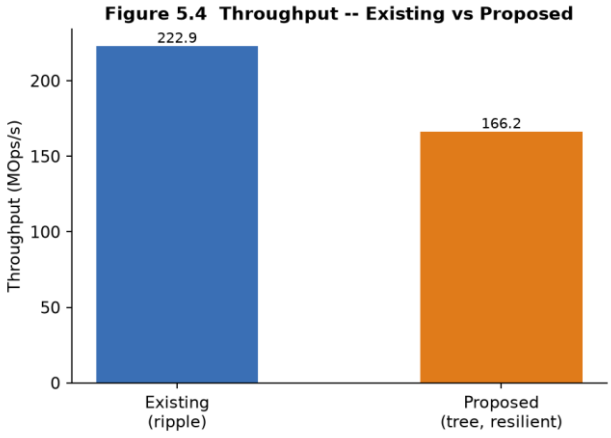

Fig. 7. Search throughput comparison, derived from the reciprocal of worst-case search latency: existing sequential design versus proposed parallel design.

Fig. 7 presents search throughput, computed as the reciprocal of worst-case search latency for each design. Because the existing design's worst-case latency (80.0 ns) is 13.3 times longer than the proposed design's (6.015 ns), the proposed design supports a correspondingly higher worst-case search throughput. This follows directly and only from the latency figures already reported in Table I; no additional independent throughput measurement was required, since throughput and worst-case search latency are reciprocally related for a design that issues one search at a time.

F. Energy per Search

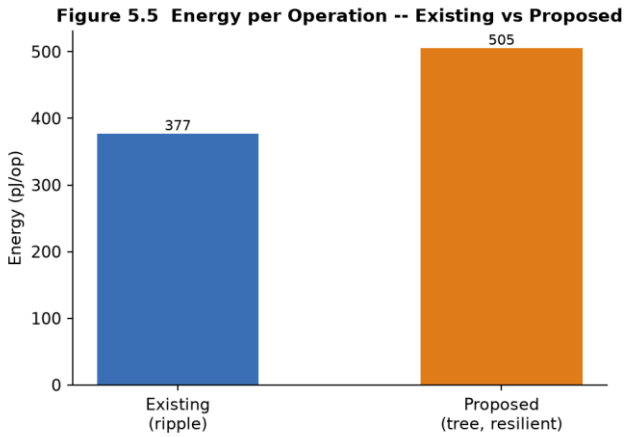

Fig. 8. Energy-per-search comparison, derived from power multiplied by worst-case search latency: existing sequential design versus proposed parallel design.

Fig. 8 presents energy consumed per worst-case search, computed as power multiplied by worst-case search latency for each design. Because power is tied at 0.084 W for both designs, the energy-per-search figure is dominated entirely by the latency term: the existing design's worst-case search consumes energy proportional to 80.0 ns at 0.084 W, while the proposed design's worst-case search consumes energy proportional to only 6.015 ns at the same power draw. The proposed design therefore consumes substantially less energy per completed worst-case search, not because it draws less instantaneous power, but because it finishes the search 13.3 times faster.

G. Worst-Case Search Latency

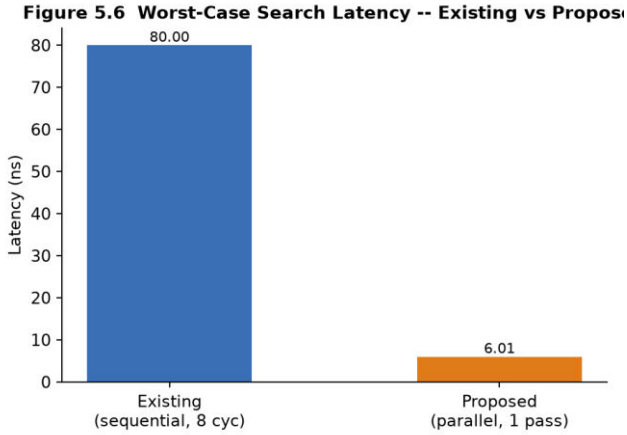

Fig. 9. Worst-case search latency comparison: existing sequential design (8 cycles x 10 ns = 80.0 ns) versus proposed parallel design (6.015 ns, 1 pass), a 13.3x reduction.

Fig. 9 presents the central result of this paper. The existing sequential exact-match design has a worst-case search latency of 80.0 ns, arising from needing up to all 8 clock cycles at the 10.0 ns constraint period to conclude a search that turns out to be a non-match, or a match on the final

entry. The proposed parallel bounded-mismatch design has a worst-case search latency of 6.015 ns, the delay of a single combinational pass through the mismatch-count-and-compare datapath, because every entry is evaluated in the same cycle regardless of which entry, if any, matches. The ratio 80.0 / 6.015 gives a 13.3x reduction in worst-case search latency, and this reduction is a worst-case guarantee, not merely an average-case improvement, since the proposed design's latency does not depend on which entry, if any, produces the match.

H. Discussion

The proposed parallel bounded-mismatch architecture trades an 87.0% increase in LUT usage for a 13.3x reduction in worst-case search latency, at no change in power, while also gaining single-bit error tolerance that the existing sequential exact-match design entirely lacks. The additional LUTs are not incidental overhead; they are the direct, honestly reported cost of instantiating eight independent comparator and balanced-tree mismatch-counting datapaths so that all eight table entries can be evaluated within a single clock period, which is the architectural property that makes a design a genuine content-addressable memory rather than a RAM-style lookup table searched by a state machine. For latency-critical applications such as line-rate packet classification, where a worst-case guarantee on search time is often a hard requirement rather than merely a desirable average, the existing design's linear-in-table-size latency is disqualifying regardless of its lower LUT count, whereas the proposed design's constant, single-pass latency directly satisfies that requirement. The bounded-mismatch match rule further provides resilience to single-bit SRAM soft errors at zero measured power cost and zero additional registers, since the entire mismatch-counting and threshold-comparison datapath is purely combinational. Taken together, the results support treating the 87.0% LUT increase as an appropriately budgeted and well-understood area investment rather than a design inefficiency, given the combined latency and reliability gains it purchases.

VII. EFFICIENCY AND TRADE-OFF ANALYSIS

A. The Central Trade-off

The result set in Section VI establishes three simultaneous, coupled facts: the proposed design uses more area (87.0% more LUTs), completes a search dramatically faster in the worst case (13.3x lower worst-case latency), and gains a capability (single-bit error tolerance) with no counterpart in the existing design. None of these three facts can be evaluated honestly in isolation from the other two, since a designer choosing between the two architectures is implicitly choosing how much silicon area to spend per unit of worst-case search-time reduction and per unit of reliability gained. This section expresses that trade-off in normalized, per-LUT terms so that the comparison does not depend on the particular (small) table size used in this study.

B. Latency-per-LUT and Area-per-Latency-Unit

Define a normalized worst-case-latency efficiency metric as the reduction in worst-case search latency achieved per additional LUT spent:

$$\eta_{\text{lat}} = \frac{L_{\text{existing}} - L_{\text{proposed}}}{A_{\text{proposed}} - A_{\text{existing}}} \quad (3)$$

where $L_{\text{existing}} = 80.0$ ns, $L_{\text{proposed}} = 6.015$ ns, $A_{\text{existing}} = 23$ LUTs, and $A_{\text{proposed}} = 43$ LUTs. Substituting:

$$\eta_{\text{lat}} = \frac{80.0 - 6.015}{43 - 23} = \frac{73.985}{20} \approx$$
3.70ns of worst-case latency removed per additional LUT (4)

Each additional LUT spent on the proposed architecture's parallel comparator array removes approximately 3.70 ns of worst-case search latency relative to the existing sequential design, for this 8-entry, 16-bit table configuration. Equivalently, expressed as a LUT cost per unit of worst-case latency achieved rather than removed, the proposed design achieves its 6.015 ns worst-case latency at a cost of $43/6.015 \approx 7.15$ LUTs per nanosecond of guaranteed worst-case search time, versus the existing design's $23/80.0 \approx 0.29$ LUTs per nanosecond -- a reminder that the existing design is "cheap per nanosecond" only because each of its nanoseconds is doing much less useful search work per cycle.

C. Energy-per-Search Efficiency

Because power is tied at 0.084 W for both designs (Section VI-D), energy-per-search reduces entirely to a latency ratio. Worst-case energy per search is $E = P \times L$:

$$E_{\text{existing}} = 0.084\text{,W} \times 80.0\text{,ns} = 6.72\text{,nJ}, \quad E_{\text{proposed}} = 0.084\text{,W} \times 6.015\text{,ns} \approx 0.505\text{,nJ} \quad (5)$$

The proposed design consumes approximately $6.72/0.505 \approx 13.3\times$ less energy per worst-case search than the existing design -- numerically identical to the latency reduction ratio, precisely because power itself did not change. Table II summarizes these normalized metrics.

Table II. Normalized Efficiency Metrics (Derived from Table I)

Metric	Existing (sequential)	Proposed (parallel)	Ratio
Worst-case latency (ns)	80.0	6.015	13.3x lower (proposed)
LUTs	23	43	1.87x higher (proposed)
Worst-case energy/search (nJ)	6.72	0.505	13.3x lower (proposed)
LUTs per ns of worst-case latency	0.29	7.15	--
Latency removed per additional LUT	--	~3.70 ns/LUT	--

D. Interpretation

The normalized figures reinforce, rather than soften, the conclusion of Section VI: the proposed architecture is simultaneously more area-hungry and dramatically more efficient in the metric that actually determines whether a memory qualifies as a usable content-addressable memory for latency-bound applications, namely worst-case search time. In deployments where table size is fixed and small -- as in this 8-entry evaluation -- the absolute LUT overhead of 20 additional look-up tables is trivial in the context of any modern FPGA's total fabric, making the 13.3x worst-case latency reduction and the newly gained single-bit error tolerance close to a strictly dominant choice. The trade-off becomes genuinely consequential only as table size scales well beyond 8 entries, since the proposed architecture's LUT cost grows with the number of entries while the existing design's does not; a designer targeting a much larger TCAM should therefore treat the per-LUT latency efficiency derived here as a starting point for re-evaluating the trade-off at the intended table size, rather than assuming the 87.0%/13.3x ratio measured at 8 entries holds unchanged at larger scale.

VIII. CONCLUSION

This paper presented and compared two SRAM-based 8-entry by 16-bit ternary CAM architectures implemented and characterized on Xilinx Vivado 2019.2 targeting an Artix-7 device. The existing design performs a sequential, RAM-style exact-match search using a single reused comparator advanced one entry per clock cycle by a finite-state machine, requiring up to 8 clock cycles and offering no tolerance to bit errors. The proposed design compares all 8 entries simultaneously in a single combinational pass, using a per-entry balanced-tree mismatch counter and a bounded-mismatch match rule that accepts a match with at most one mismatched bit, giving the design genuine single-cycle CAM behavior together with single-bit soft-error tolerance. Post-place-and-route implementation results showed that the proposed design reduces worst-case search latency from 80.0 ns to 6.015 ns, a 13.3x improvement, at a cost of 87.0% more look-up tables (43 versus 23) and with power tied at 0.084 W for both designs. A normalized efficiency analysis showed the proposed design removes approximately 3.70 ns of worst-case search latency for every additional LUT it consumes relative to the existing design, and consumes roughly 13.3x less energy per worst-case search as a direct consequence of its latency advantage at equal power.

These results support the conclusion that parallel bounded-mismatch matching is a practical and well-quantified upgrade path for SRAM-based TCAM designs that currently rely on sequential exact-match search, delivering both the latency guarantee expected of a true content-addressable memory and a genuinely new error-resilience capability, at an area cost that is modest in absolute terms for small tables and predictable in scaling terms for larger ones. Future work includes extending the bounded-mismatch matching rule to configurable mismatch thresholds beyond a single bit, evaluating the proposed architecture at larger table sizes (32, 64, and 128 entries) to characterize how the measured 87.0%/13.3x area-latency trade-off scales, exploring priority-encoder and multi-match resolution logic for cases where more than one entry satisfies the bounded-mismatch criterion simultaneously, and investigating combined single-event-upset injection campaigns to quantify the proposed design's error-correction coverage beyond the single-bit fault model analyzed in this paper.

REFERENCES

[1] T. Nguyen, S. Ryu, and I. Chang, "TRIO-TCAM: An Area and Energy-Efficient Triple-State-in-Cell Ternary Content-Addressable Memory Architecture," *IEEE Access*, vol. 12, 2024.

[2] P. C. Grace and S. Sophia, "Multibit Soft Error Resilient SRAM Based TCAM for FPGA," *AIP Conference Proceedings*, 2023.

[3] D. Sun, T. Tang, S. Tan, X. Cheng, and Z. Zhang, "A Reconfigurable 11T SRAM Macro for Error Tolerance Content-Addressable Memory and In-Memory Computing," *International Journal of Circuit Theory and Applications*, 2025.

[4] R. Zhang, C. Tang, X. Sun, M. Li, W. Jin, P. Li, X. Cheng, and X. S. Hu, "Sky-TCAM: Low-Power Skyrmion-Based Ternary Content Addressable Memory," *IEEE Transactions on Electron Devices*, vol. 70, no. 7, 2023.

[5] H. Chuang and V. P. Hu, "Variation-Tolerant Ferroelectric FET-Based Ternary Content-Addressable Memories (TCAM) Cell for Meta-Learning Application," in *Proc. 2023 7th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, 2023.

[6] B. S. Kumar and I. Sowmya, "Enhancing Ternary Content Addressable Memory Reliability with Error-Tolerant Convolutional Logic Code Synthesis," *International Journal of Electronics and Communication Engineering*, vol. 13, no. 4, 2026.

[7] T. Nguyen, M. Le, T. Pham, and I. Chang, "TA-Quatro: Soft Error-Resilient and Power-Efficient SRAM Cell for ADC-Less Binary Weight and Ternary Activation In-Memory Computing," *Electronics*, vol. 13, no. 15, 2024.

[8] M. V. Subramanyam, Y. M. Rao, and S. J. Basha, "An Efficient, Variation Tolerant CNTFET Ternary Content Addressable Memory: A PVT Variation Resilient Design," *Transactions on Electrical and Electronic Materials*, vol. 25, no. 6, 2024.

[9] T. Gopi, "Design of Low Power Logic Gates for VLSI Design Circuits," *International Journal of Science and Research (IJSR)*, vol. 12, no. 3, 2023.

[10] K. Kanaparthi, "Leakage Power Reduction Techniques for Low Process Technology VLSI Design Circuits,"

International Journal of Science and Research (IJSR), vol. 12, no. 3, 2023.

[11] K. Geetha, "Ultra-Low-Power VLSI Design Using Near-Threshold Computing Paradigms," *Annals of Energy-Efficient VLSI Architectures*, vol. 1, no. 1, 2026.

[12] K. Ramkumar and S. Deb, "Low Power Content-Addressable Memory for FPGA Implementation," *International Journal of Electronics*, 2026.

[13] M. Abbasmollaei, T. Ould-Bachir, and Y. Savaria, "HLSCAM: Fine-Tuned HLS-Based Content Addressable Memory Implementation for Packet Processing on FPGA," *Electronics*, vol. 14, no. 9, 2025.

[14] H. Oztekin, A. Lazzem, and I. Pehlivan, "Using FPGA-Based Content-Addressable Memory for Mnemonics Instruction Searching in Assembler Design," *The Journal of Supercomputing*, vol. 79, no. 15, 2023.

[15] A. Joy and J. Kuruvilla, "Implementation of Low Power Memristor Content Addressable Memory Using FinFET," *Archives for Technical Sciences*, vol. 2, no. 33, 2025.

[16] S. Ameli, J. Yu, and J. P. Strachan, "Associative Clustering with Analog Content-Addressable Memory," in *Proc. International Symposium on Memory Systems*, 2025.

[17] S. Inamanamelluri, D. Dhanasekaran, and R. Bhaskar, "Low Power Content Addressable Memory Designing and Implementation Using Voltage Swing Self Adjustable Match Line Technique," *Sustainable Computing: Informatics and Systems*, vol. 43, 2024.

[18] F. Benoist, L. Peliti, and P. Sartori, "Content-Addressable Memory with a Content-Free Energy Function," *Physical Review Letters*, vol. 135, 2025.

[19] Y. Yang, C. Giotis, T. Prodromakis, and A. Serb, "A Resource-Efficient Dually-Addressable Memory Architecture on FPGA," in *Proc. 2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2025.

[20] A. Srividyabarathi and P. Deepa, "Self-Recovery RRD-12T SRAM Cell-Based Array for Fault-Tolerant Energy-Efficient Memory Systems," *Microelectronics Reliability*, vol. 182, 2026.

[21] J. Oh and S. Jo, "Soft Error-Tolerant and Highly Stable Low-Power SRAM for Satellite Applications," *Applied Sciences*, vol. 15, no. 1, 2025.

[22] M. N. Shaker, A. Hussien, H. H. Amer, and B. Shokry, "Reliability Modeling of Fault-Tolerant FPGA-Based Architectures in Space Applications for Soft and Hard Error Recovery," *IEEE Access*, vol. 12, 2024.

[23] B. Joseph and R. Kavitha, "Design of an Energy-Efficient Soft Error Resilient RHBD SRAM Cell with High Read Stability and Minimum Write Errors," *Microelectronics Reliability*, vol. 172, 2025.

[24] S. Sengupta and A. S. Yadav, "A Reliable and Soft Error Tolerant 12T Radiation Hardened SRAM Cell for Aerospace Applications," in *Proc. 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024.

[25] Q. Zhao, H. Dong, X. Wang, L. Hao, C. Peng, Z. Lin, and X. Wu, "Novel Radiation-Hardened SRAM for Immune Soft-Error in Space-Radiation Environments," *Microelectronics Reliability*, vol. 140, 2023.