FAST MAPPING BINARY CAM ON FPGA

Surabhi Nadig
Department of Electronics and Communication Engineering
BMS College of Engineering
Bengaluru, India
surabhi.ec22@bmsce.ac.in

Thrisha M
Department of Electronics and Communication Engineering
BMS College of Engineering
Bengaluru, India
thrisha.ec22@bmsce.ac.in

Vijayalaxmi
Department of Electronics and Communication Engineering
BMS College of Engineering
Bengaluru, India
vijayalaxmi.ec22@bmsce.ac.in

Yuktha Krishna G
Department of Electronics and Communication Engineering
BMS College of Engineering
Bengaluru, India
yuktha.ec22@bmsce.ac.in

Dr Shivkumar M
Assistant Professor,
Department of Electronics and Communication Engineering
BMS College of Engineering
Bengaluru, India
shivkumarm.ece@bmsce.ac.in

Abstract : In modern digital systems, rapid data retrieval is vital for low latency and high throughput. Traditional RAM is inefficient for content-based searches. Binary CAM (BCAM), a simpler and lower-power alternative to TCAM, allows content-based data access but struggles with dynamic updates.

This project presents a high-speed, FPGA-based BCAM architecture using BRAM and LUTs to enable low-latency searches and efficient real-time updates. The scalable design is ideal for data-intensive applications requiring fast associative memory.

I. INTRODUCTION

Traditional memory systems, such as RAM, retrieve data using explicit addresses, requiring prior knowledge of data location. This becomes inefficient when searching for unknown data, as sequential searches or indexing introduce delays. Content Addressable Memory (CAM) offers an alternative by allowing memory to be queried by data content. CAM compares input data against all stored entries in parallel, enabling constant-time search operations regardless of memory size. This makes CAM ideal for applications like network routing, firewall filtering, and database indexing.

Binary CAM (BCAM), a variant of CAM, matches exact binary patterns and is simpler and faster than Ternary CAM (TCAM), which supports partial matching using "don't care" states. While TCAMs offer flexibility, they consume more power and area, making BCAM preferable when exact matching is sufficient. However, BCAMs traditionally struggle with dynamic updates, often requiring complex control logic and memory refresh cycles.

This project focuses on implementing a BCAM architecture on FPGA to deliver low-latency lookups and fast updates. The design aims to support real-time applications in networking and embedded systems by offering a high-speed, flexible, and scalable associative memory solution

II. LITERATURE SURVEY

FMU-BiCAM proposes a novel mapping and updating algorithm for FPGA-based Binary CAMs, reducing update latency to just two clock cycles. Implemented on Xilinx Virtex-6, it outperforms traditional methods that require O(N) cycles, making it ideal for real-time, high-speed applications [1].

This work implements a 512×36-bit TCAM using SRAM on a Xilinx Virtex-5 FPGA. By using BRAM-AUTO synthesis optimization, it achieves low power consumption (2.11 mW @ 100 MHz) while maintaining fast lookup speeds, offering an efficient alternative to conventional TCAMs. [2].

Z-TCAM uses vertical and horizontal partitioning to map TCAM entries onto SRAM on a Virtex-7 FPGA. It supports wildcard searches, achieves a 3-cycle search latency, and enhances memory efficiency, making it a scalable and power-efficient TCAM replacement.[3].

This architecture partitions TCAM tables into smaller SRAM-mapped units to improve scalability and reduce power. It delivers lower latency and better memory utilization, making it suitable for high-speed search operations in networking and databases [4].

REST introduces virtual memory blocks to maximize SRAM usage and reduce energy consumption. Though it slightly lowers throughput, it cuts memory use to 3.5% on Kintex-7 FPGAs,

offering a highly energy-efficient TCAM solution for power-sensitive systems. [5].

This design improves hash join operations for databases using SRAM and FPGA parallelism. It reduces redundant computations and memory bottlenecks, delivering better throughput for real-time analytics than CPU/GPU-based systems [6].

Combining Local Binary Pattern (LBP) with CAM on FPGA, this design accelerates pattern matching for vision and surveillance systems. It reduces latency and overhead, outperforming traditional software-based recognition methods. [7].

Earlier SRAM TCAMs used hashing to reduce cost and power, but suffered from collisions and inefficiencies. These limitations highlighted the need for better approaches, influencing designs like FMU-BiCAM and Z-TCAM. [8].

Hybrid RAM-CAM methods aimed to balance speed and efficiency using hierarchical structures, but often faced complexity and scalability issues. These limitations pushed newer, FPGA-friendly CAM designs focused on power and real-time performance.[9].

III. METHODOLOGY & IMPLEMENTATION

In a Binary CAM (BCAM) system, the Input CAM Word is the binary pattern that needs to be searched within the memory. This word is fed into the system and compared against stored entries. An Address Decoder assists in routing control or data signals to the appropriate memory blocks or logic units.

The core of the BCAM lies in the Control Logic, which manages mapping (data organization), lookup (searching), and update (modifying entries) operations, often implemented using a finite state machine (FSM). Data is stored in LUT-Based Memory, where each entry holds a binary word and is implemented using FPGA resources like distributed RAM or LUT RAMs.

The Matching Logic performs bitwise comparisons between the input word and stored entries, flagging any match. These results are processed by the Lookup Logic, which detects if a match occurred and identifies the highest-priority one—typically the lowest address—using encoders. When updates are required, the Update Mechanism clears outdated data and writes new words, controlled by enable signals and address decoding.

Finally, the Output Matched Address stage outputs the address where a match is found; if no match exists, a default value may be output. The parallel comparison nature of the LUTs and matching units ensures high-speed operation, making BCAMs efficient for fast search applications..

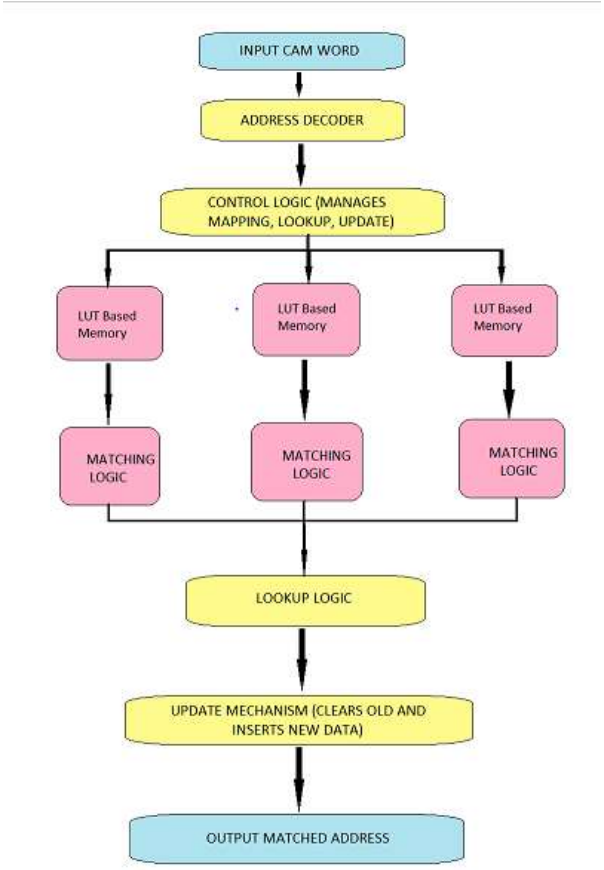

Fig 1: Block diagram

The system begins operation upon a reset, either from power-up or a manual reset trigger. During initialization, key registers such as `boolean_board` (which stores final match results), `match_found` (indicating if a match was detected), `match_index` (storing the index of the first match), and `search_data` (the value to be searched) are all cleared. The system then waits for a clean, debounced press of the central button `btnC` to ensure stability against mechanical bounce noise.

Once triggered, the system loads the search data from the input switches `sw[1:0]`, extending it to a 4-bit format with leading zero padding. Temporary variables like `temp_boolean_board`, `match_found`, and `match_index` are reset to prepare for a fresh search. A for-loop begins, iterating through all rows of the CAM memory. For each row, the system compares the stored value against the `search_data`; if a match is found, the corresponding index in `temp_boolean_board` is set to 1.

If it is the first match, the system records the index and flags the match. After the search completes, `boolean_board` is updated with the results, and LEDs (`LED[3:0]`) are lit to visually indicate which memory locations matched the input data. The system then enters an idle state, waiting for the next valid button press to repeat the search. This design performs parallel-style

search behavior using sequential logic and provides real-time visual feedback. It's optimized for small-width binary data, and the outputs like `match_index` and `match_found` can also be integrated into larger systems for further data-driven decisions.

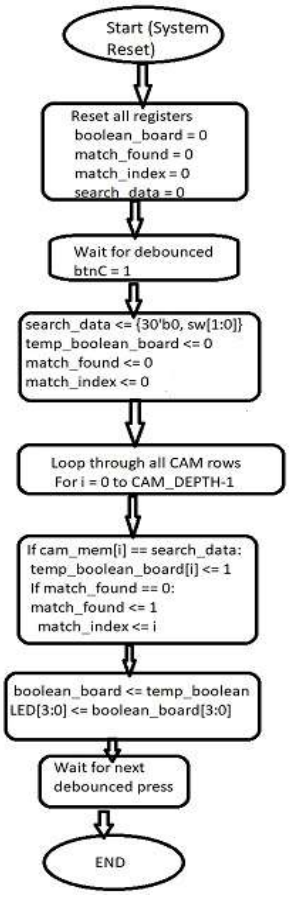

Fig 2: Work flow

IV. RESULTS AND DISCUSSIONS

A. RESULTS

The block diagram represents a top-level hardware integration of a Binary Content Addressable Memory (BCAM) module with input and output logic designed for FPGA implementation. The system accepts user inputs through physical switches (sw[15:0]) and buttons (btn[2:0]). The switches provide both the write pattern (wPatt) and the match pattern (mPatt), which are stored in dedicated registers. The wPatt_reg and mPatt_reg hold these 16-bit values, and their values are updated synchronously with the clock when triggered. The write address (wAddr) is managed by a register that takes its input from an adder block (RTL_ADD) that automatically increments the address value for sequential write operations. When writing is enabled via a button signal, the BCAM module stores the input pattern into the location specified by wAddr. During the search phase, the mPatt register provides the pattern to be matched, and the BCAM checks its memory contents for a match. If a match is found, the match output signal goes high, and the corresponding match address (mAddr) is driven out. The match address or result is then displayed on a set of output LEDs (led[15:0]), allowing real-time visual verification of the match operation. Overall, this system enables easy testing and demonstration of BCAM functionality on an FPGA board, supporting both data storage and pattern matching operations through simple user interaction.

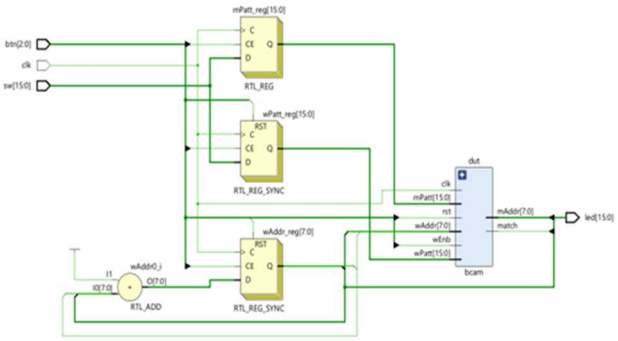

Fig 3: Circuit Diagram

The image shown is a layout view of a VLSI (Very Large Scale Integration) circuit, likely representing a module such as a Binary Content Addressable Memory (BCAM) designed using an EDA (Electronic Design Automation) tool like Cadence Virtuoso or Synopsys.

The layout is divided into multiple rectangular regions, each outlined with colored borders, indicating the presence of hierarchical blocks or standard cell placements. The fine grid of vertical and horizontal lines across the layout represents the metal routing layers used to interconnect different logic components.

The repeated structure and dense regions suggest memory array organization, which is characteristic of CAM designs where each cell performs parallel comparisons. The layout likely includes bitcell arrays, matchline logic, priority encoders, and peripheral control circuits. This physical layout is crucial in the final stage of chip design as it verifies correct placement, routing, and compliance with design rules before fabrication or FPGA synthesis.

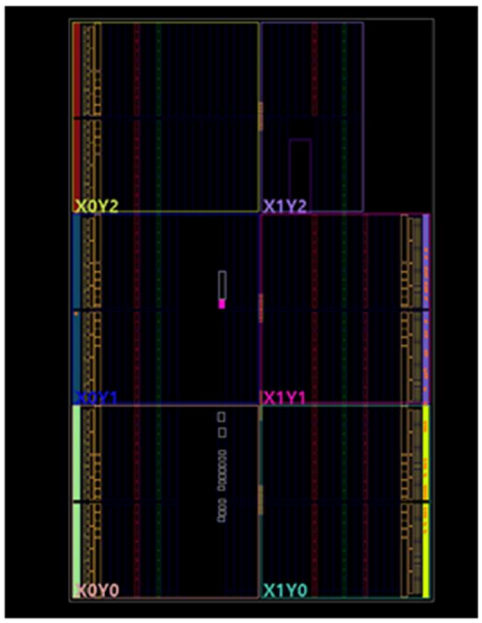

Fig 4: Floor Planning Diagram

The simulation waveform shown represents the functional behavior of a Binary Content Addressable Memory (BCAM) module with a match address output (mAddr) and match flag (match). The primary signals under observation include the system clock (clk), reset (rst), write enable (wEnb), write address (wAddr), write pattern (wPatt), match pattern (mPatt), match flag (match), and match address (mAddr).

Initially, the system is under reset (rst = 1), holding all operations idle. Once reset is deasserted (rst = 0 at around 30 ns), the module starts operating. During the interval where wEnb = 1 (from ~35 ns to ~170 ns), data is sequentially written into the CAM at addresses wAddr = 0 to wAddr = f (hexadecimal 0 to 15). The patterns wPatt being written are 8-bit values, such as 24, 81, 09, etc., shown clearly in the waveform. This demonstrates the loading phase of the BCAM.

Following the write phase, the wEnb signal is deasserted (wEnb = 0), transitioning the CAM to the search phase. During this phase, the input mPatt is updated with different 8-bit patterns (e.g., 0a, c5, aa, etc.), and the CAM performs a search operation to check if any stored entry matches the mPatt. When a match is found, the match signal goes high (match = 1) and the corresponding address of the matching data is output on mAddr. For example, at the final point shown in the waveform, when mPatt = 63, the match signal goes high and mAddr = 6, indicating that the data 63 was previously written to address 6 and successfully matched.

Overall, this simulation confirms the correct functionality of the BCAM: it stores data in write mode and correctly detects matching data with the corresponding address in search mode.

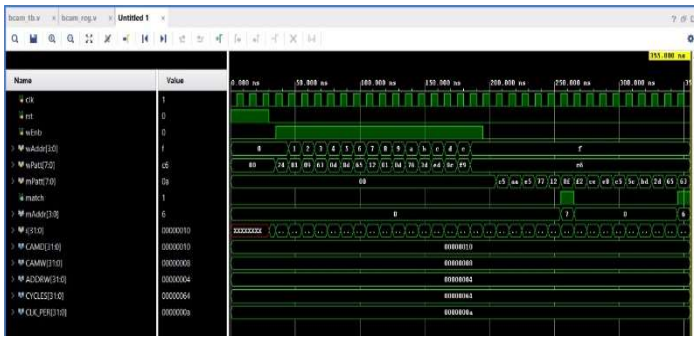

Fig 4: Simulation Waveform

B. DISCUSSIONS

In real-world applications, Binary CAM (BCAM) offers high-speed parallel search capabilities, making it valuable in systems like IP routing tables, cache indexing, and packet classification in networking hardware.

However, **scalability** becomes a challenge as the number of entries and bit-width increases. Since BCAMs rely on parallel comparison logic, the hardware complexity and resource usage (logic elements, power, and routing) grow rapidly with size, making them less practical for large-scale implementations without optimization. In terms of **cost**, BCAMs are relatively expensive in FPGA or ASIC designs due to their high area and power consumption, especially when implemented using distributed logic or flip-flops instead of efficient memory blocks.

This makes them more suitable for performance-critical, small-footprint tasks. Regarding **accessibility**, while BCAMs are conceptually simple and easy to prototype using HDL languages on FPGAs, deploying them at scale requires careful hardware-aware design and synthesis constraints. Nonetheless, they remain accessible for academic, experimental, and embedded-system-scale uses, especially when small depth and width are sufficient. Optimized versions like SRAM-based or ternary CAMs are often used in industry to balance speed with resource efficiency.

V. CONCLUSION

The proposed work a parameterized, low-latency Binary Content Addressable Memory (BCAM) architecture suitable for presents real-time data retrieval and dynamic updates in embedded and networked systems. By leveraging the parallelism and reconfigurability of FPGAs, the architecture achieves high-speed matching in just one clock cycle and supports flexible write/update operations.

Implemented using Verilog HDL and tested on Xilinx FPGA devices through Vivado simulation and synthesis, the design has been proven both functionally robust and resource-efficient. The BCAM supports:

Parallel comparisons for high-speed search,Dynamic updates at runtime,Scalability for different data widths

and CAM depths,And modular design for seamless integration into larger digital systems.This BCAM can be deployed in multiple real-time applications such as:Packet classification in routers,Rule matching in firewalls,Cache lookup systems,Symbol table search engines,And hardware accelerators for AI inference systems

VI. FUTURE TRENDS

While the current implementation meets the goals of fast matching and update support, several enhancements can be introduced in future iterations to extend its performance, flexibility, and application scope.

1. AXI4-Lite Integration: Enables BCAM control via software (e.g., Python on PYNQ) for easy SoC integration with ARM processors like Zynq-7000.

2. Upgrade to TCAM: Adds support for mask-based matching (X/"don't care" bits), ideal for IP routing, SDN, and security applications.

3. Match Vector & Multi-Hit: Outputs all match indices as a bit-vector, supporting parallel match detection useful in AI and pattern recognition.

4. Partial Reconfiguration: Allows dynamic updates to CAM regions without halting the FPGA—ideal for real-time systems like IDS.

5. Use of BRAM/UltraRAM: Supports larger CAM arrays (e.g., 512–1024 entries) with better density, power efficiency, and performance.

6. Pipelined/Hierarchical CAMs: Improves speed and scalability by reducing critical path and organizing entries in tiers.

7. AI Applications: Suitable for AI accelerators in tasks like label matching, inference lookup, and sparse matrix access in GNNs

VII. REFERENCES

[1.] Azhar Qazi, Zahid Ullah , Member, IEEE, and Abdul Hafeez, "Fast Mapping and Updating Algorithms for a Binary CAM on FPGA", IEEE canadian journal of electrical and computer engineering, vol. 44, no. 2, spring 2021

[2.] A. Madhavan, T. Sherwood, and D. B. Strukov, "High-throughput pattern matching with CMOL FPGA circuits: Case for logic-in-memory computing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 12, pp. 2759–2772, Dec. 2018..

[3.] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "Z-TCAM: An SRAM based architecture for TCAM," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 2, pp. 402–406, Feb. 2015.

[4.] Z. Ullah, K. Ilgon, and S. Baeg, "Hybrid partitioned SRAM-based ternary content addressable memory," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 12, pp. 2969–2979, Dec. 2012

[5.] F. Syed, Z. Ullah, and M. K. Jaiswal, "Fast content updating algorithm for an SRAM-based TCAM on FPGA," IEEE Embedded Syst. Lett., vol. 10, no. 3, pp. 73–76, Sep. 2018.

[6.] A. Ahmed, K. Park, and S. Baeg, "Resource-effcient SRAM-based ternary content addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1583–1587, Apr. 2017.

[7.] Zahid Ullah, Manish Kumar Jaiswal, Y.C. Chan, and Ray C.C. Cheung, "FPGA Implementation of SRAM-based Ternary Content Addressable Memory". 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum.

[8.] Wen-Qi Wu, Mei-Ting Xue , Tian-Qi Zhu, Zhen-Guo Ma and Feng Yu, "HighThroughput Parallel SRAM-Based Hash Join Architecture on FPGA". IEEE transactions on circuits and systems—ii: express briefs, vol. 67, no. 11, november 2020

[9.] Omer Mujahid, Zahid Ullah, Hassan Mahmood, and Abdul Hafeez, "Fast Pattern Recognition Through an LBP Driven CAM on FPGA".