Electronic Letters on Science & Engineering 19(2) (2023)
Available online at dergipark.org.tr/else

The Power of Computing in Memory for Searching Algorithm

Abdelkader LAZZEM[1*], Halit ÖZTEKİN[2]

[1]Department of Electrical-Electronics Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye
22500409008@subu.edu.tr, ORCID: 0000-0003-0136-356X,
[2] Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye
halitoztekin@subu.edu.tr, ORCID: 0000-0001-8598-4763

Abstract: This work focuses on the critical problem of search algorithm optimization to improve efficiency in a variety of applications within the field of computing. Through the utilization of technology's ongoing advancements, namely in the area of hardware acceleration, the research delves into different approaches meant to improve search algorithm performance. It presents a methodical comparison between the hardware-based Binary Content Addressable Memory (BiCAM)-based search algorithm and conventional software-based search algorithms like Sequential and Binary. The Field-Programmable Gate Array (FPGA), selected for its unmatched adaptability in hardware configurations, is used for implementation. By means of a thorough analysis, the study aims to identify the advantages, disadvantages, and complexity of these algorithms. The overall objective is to contribute to the continuing conversation in computer engineering and digital circuit design by providing nuanced insights into algorithm choices that are suited to particular application objectives. Essentially, the study explores the conditions of different FPGA-based search algorithms, offering a thorough comprehension to direct well-informed decisions for the best results in a range of applications. From the obtained results, it is evident that the BiCAM consumes more power and utilizes more resources but excels in terms of time complexity, making it a favorable trade-off in certain applications where speed is of greater importance.

Keywords: Search Operation, Field-programmable Gate Array (FPGA), Binary Search, Sequential Search, Binary Content Addressable Memory (BiCAM)

Arama Algoritması için Bellekteki Hesaplama Gücü

Özet: Bu çalışma, bilgisayar alanındaki çeşitli uygulamalarda verimliliği artırmak için kritik bir sorun olan arama algoritması optimizasyonuna odaklanmaktadır. Araştırma, teknolojinin donanım hızlandırma alanında devam eden ilerlemelerinden faydalanarak, arama algoritması performansını artırmaya yönelik farklı yaklaşımları incelemektedir. Donanım tabanlı İkili İçerik Adreslenebilir Bellek (BiCAM) tabanlı arama algoritması ile Sıralı ve İkili gibi geleneksel yazılım tabanlı arama algoritmaları arasında metodik bir karşılaştırma sunmaktadır. Donanım konfigürasyonlarındaki eşsiz uyarlanabilirliği nedeniyle seçilen Alan Programlanabilir Kapı Dizisi (FPGA), uygulama için kullanılmıştır. Çalışma, kapsamlı bir analiz yoluyla, bu algoritmaların avantajlarını, dezavantajlarını ve karmaşıklığını belirlemeyi amaçlamaktadır. Genel amaç, belirli uygulama hedeflerine uygun algoritma seçimlerine ilişkin incelikli bilgiler sağlayarak bilgisayar mühendisliği ve dijital devre tasarımında süregelen tartışmalara katkıda bulunmaktır. Esasen araştırma, farklı FPGA tabanlı arama algoritmalarının koşullarını araştırmakta ve bir dizi uygulamada en iyi sonuçlar için iyi bilgilendirilmiş kararları yönlendirmek için kapsamlı bir kavrayış sunmaktadır. Elde edilen sonuçlardan, BiCAM'in daha fazla güç tükettiği ve daha fazla kaynak kullandığı, ancak zaman karmaşıklığı açısından üstün olduğu ve hızın daha önemli olduğu bazı uygulamalarda uygun bir değiş tokuş (trade off) yaptığı açıktır.

Anahtar Kelimeler: Arama İşlemi, Alanda Programlanabilir Kapı Dizisi (FPGA), İkili Arama, Sıralı Arama, İkili İçerik Adreslenebilir Bellek (BiCAM)

Reference to this paper should be made as follows (bu makaleye aşağıdaki şekilde atıfta bulunulmalı):
Lazzem, A., Oztekin, H., 'The Power of Computing in Memory for Searching Algorithm', Elec Lett Sci Eng , vol. 19(2) , (2023), 71-89

[*] *Corresponding author; Abdelkader Lazzem(*22500409008@subu.edu.tr, lazzemabdk@gmail.com)

ISSN 1305-8614

1. INTRODUCTION

In the ever-changing world of computers and technology, the optimization of search algorithms stands as a crucial endeavor, significantly contributing to the efficiency and effectiveness of diverse applications. Search operations are a fundamental form of computer activity used in many different applications [1,2], often consuming a significant amount of the total program execution time. The continuous evolution of technology, particularly in the domain of hardware acceleration, has led to inventive strategies aimed at enhancing the performance of search algorithms. Essentially, the search operation is the basic process of finding specific information or data within a larger dataset. This operation entails comparing a chosen search term with the data stored in a particular data structure or database. If a match is found, the relevant information is then retrieved. Different algorithms and techniques, like linear search, binary search, or hash table search [3], are used to perform this search operation. While many software-based approaches have been proposed to increase search speed, their efficiency may be constrained by the structural limitations of memory *[4-9]*. Notably, these methods primarily focus on optimizing software functionalities without direct alterations to the hardware infrastructure. As a result, the scope of improvement may be restricted by the pre-existing hardware configurations. In this context, the ongoing advancements in hardware acceleration technologies create opportunities for significant improvements in the performance of search algorithms. This paper conducts a comprehensive comparison of traditional software-based search algorithms like Sequential and Binary with the hardware-based Binary Content Addressable Memory (BiCAM)-based search algorithm, all implemented within the framework of a Field-Programmable Gate Array (FPGA). The selection of FPGA as the implementation framework is motivated by its unparalleled flexibility in customizable hardware configurations, facilitating the creation of efficient and tailored solutions across a diverse range of applications. Through a systematic evaluation of these algorithms, our goal is to identify the strengths, weaknesses, and performance nuances associated with different search strategies. This analysis aims to guide the selection of algorithms based on specific application needs, taking into account trade-offs and considerations for optimal performance.

Briefly,This study investigates various search algorithms implemented on FPGA , aiming to identify strengths, weaknesses, and guide algorithm selection based on specific application needs.

The contributions made in this work can be summarized as follows:

- **Objective**: Enhancing efficiency in various applications through the optimization of search algorithms.

- **Innovative Strategies**: Exploring inventive approaches, particularly in hardware acceleration, to improve algorithm performance.
- **Fundamentals of Search Operations**: Explaining the basic process of search operations, including the use of different algorithms.
- **Limitations of Software-Based Approaches**: Highlighting constraints and drawbacks in software-based methods, such as memory limitations and the exclusive focus on software optimization.
- **Algorithmic Comparison**: Conducting a thorough comparison between traditional software-based search algorithms (Sequential and Binary) and the hardware-based BiCAM-based search algorithm within the context of FPGA implementation.
- **Rationale for FPGA Selection**: Choosing FPGA as the implementation framework due to its flexibility in customizable hardware configurations.
- **Systematic Evaluation Goal**: Identifying strengths, weaknesses, and performance difference associated with different search strategies through systematic evaluation.
- **Guidance for Algorithm Selection**: Providing guidance for selecting algorithms based on specific application needs, considering trade-offs for optimal performance.

The remainder of this work is organized as follows: Section 2 reviews relevant work in search algorithms, summarizing key findings and approaches in the field. Section 3 introduces the FPGA-based Content Addressable Memories (CAMs), detailing th FPGA-based CAMs features and thier types. Sections 4 and 5 covers the implementation and evaluation of the proposed approaches. Section 6 presents and discusses the results, while Section 7 concludes the work.

2. Related Works

In the field of computer science and information retrieval, Search algorithms are essential for locating particular elements within datasets in the fields of computer science and information retrieval. Many researchers have explored ways to speed up searches using various techniques in both hardware and software. They often prioritize speed, measured in clock cycles. A comprehensive literature review has been conducted to enhance our understanding of these search techniques. Sequential search and binary search are two popular techniques among these algorithms, each with unique features and uses. Balogun (2019) introduces the bilinear search algorithm as a novel approach to address limitations observed in the linear search. Unlike the unidirectional linear search, the bilinear search conducts searches from different directions in an organized manner. The study evaluates five forms of bilinear search and linear search by searching for specific numbers in a 50-element array. Results demonstrate that the bilinear search, in its various forms, is more efficient than the linear search. Despite variations in the number of search directions, the time efficiency of the modified linear search consistently

outperforms the traditional linear search. The time complexity of the modified linear search algorithm is expressed as O(n/2), but when constants are disregarded, the complexity remains O(n) [10]. Parmar and Kumbharana (2014) present a comparative analysis of linear search and binary search algorithms, highlighting their implementation across diverse data structures. They introduce a modified binary search specifically designed for linked linear lists. The analysis reveals that, while binary search is generally more efficient, linear search may be preferable in specific scenarios [11]. Arora et al. (2014) introduce the Two Way Linear Search algorithm, a modified version of the linear search designed for finding a specific element in an unordered array. Unlike binary search, ordering of the array is not required. The Two Way Linear Search algorithm compares elements from both ends of the array to enhance search efficiency. The implementation and analysis were conducted using MATLAB 8.0, comparing the CPU time taken by both algorithms for input sequences of various lengths (10,000, 50,000, 100,000, and 5,000,000). Their results indicate that their introdued algorithm performs well for all input values, particularly showing improved efficiency when searching for an element beyond the middle of the array, while maintaining similar performance to linear search otherwise [12].

Irmayana et all (2021) have conducted a systematical study aimed to compare the performance of linear, binary, and interpolation search algorithms by evaluating the search time and number of passed-nodes for desired items in a dataset of 3946 records. Their results demonstrated that the binary search algorithm surpassed both linear and interpolation searches, requiring less time and fewer passed-nodes [7]. Nowak (2008) introduced a generalized version of the classic binary search problem, extending it to learn more general binary-valued functions. His study presented an algorithm that, based on queries that are maximally discriminating at each step, can determine the correct function in a logarithmic number of steps, specifically when certain geometrical relationships between the set of target functions and queries are satisfied [13]. Jacob et al. (2017) presented a discussion on commonly used search algorithms: binary search and linear search. They highlighted the respective advantages and disadvantages of each method and introduced a novel algorithm that combines the strengths of both, offering an effective way to search for a key element in an unsorted array within a limited time frame. Unlike binary search, which requires sorted data, and linear search, which can be time-consuming, the proposed algorithm overcomes these limitations and demonstrates improved efficiency in searching unsorted arrays. Their results and analysis indicate that the combined algorithm performs significantly faster than the individual techniques, making it applicable to various input scenarios [14]. Tiong et al. (2017) propose a method for determining the maximum

power point (MPP) in a photovoltaic (PV) system under partial shaded conditions, employing the Binary Search Algorithm. The Binary Search Algorithm facilitates MPP tracking by dynamically adjusting search boundaries to eliminate infeasible regions. Their results show that the binary search algorithm offers rapid convergence with zero steady-state oscillation, showcasing its effectiveness in MPP tracking for partially shaded PV systems [15]. Jin and Finkel (2020) implemented a binary search algorithm using OpenCL on an Intel Arria 10 FPGA platform, focusing on irregular memory access patterns. They optimized the grid search in XSBench by incorporating techniques like vectorization and kernel replication. To address computational overhead, they grouped work-items into work-groups, resulting in a 1.75x performance improvement in the grid search using the classic binary search on the FPGA. The study applied these optimizations to the grid search in XSBench as a case study, demonstrating minimal changes needed in the OpenCL host program [16]. Bennett Et all. (2015) introduce an Enhanced Binary Search algorithm aiming to address inefficiencies like starvation and unnecessary space searches in traditional binary search algorithms. By optimizing the search space, particularly when the search key is outside that region, the Enhanced algorithm achieves a runtime of O(log n). Comparative analysis with the traditional Binary Search algorithm across various input sizes shows that the Enhanced Binary Search consistently outperforms, demonstrating an average runtime of O(1) when the search key is outside the feasible search region. This enhancement signifies faster search capabilities, highlighting the effectiveness of the Enhanced Binary Search algorithm [17].

CAMs are also commonly employed techniques for searching. The main fundamental objective of CAM is to have an exact matches between the stored information and the queried data. However, there has been an increasing interest in approximate search and comparison for diverse data-intensive applications. The adoption of FPGA-based CAMs is increasing, especially within the field of computer engineering. This happen due to recent technological advancements and the escalating demand for rapid information access, emphasizing the requirement for parallel data access. Garzón et al. (2023) present a review study exploring the applications of approximate CAM in various domains, including big data, genomics, and other data-intensive applications. The study highlights the potential of approximate CAM in accelerating the processing of large data workloads [18]. Öztekin (2022) introduces the utilization of CAM to evaluate the digital circuit drawings, aiming to reduce the time complexity involved in scoring hand-drawn or digitally created logic circuits. The study explores the integration of machine learning methods for circuit diagnosis and the incorporation of netlist files into CAM,

demonstrating an enhanced verification speed compared to alternative algorithms. Despite the increased resource consumption associated with CAM, the study views its faster logic circuit evaluation process as a valuable trade-off. This work highlights the potential applications of CAM in improving the efficiency of educational evaluation processes and encourages further exploration of resource-efficient CAM types for this purpose [19]. Garzón et al. (2023) proposed a design named AM4, which integrates spin-transfer torque magnetic tunnel junction (STT-MTJ)-based technologies, including Content Addressable Memory (CAM), Ternary CAM (TCAM), approximate matching CAM (ACAM), and in-memory Associative Processor (AP). This design holds potential applications in data-intensive scenarios [20]. Lazzem et al. (2023) present a comparative study that evaluates two approaches for improving search operation speed using FPGA-based BiCAM , a parallel computer memory. The two approaches involve utilizing Flip-Flop (FF) and Block Random Access Memory (BRAM) as memory elements. Both implementations show a consistent time complexity of O(1) for the search operation, regardless of the number or length of entries. However, they diverge in terms of resource utilization and power dissipation due to their hardware structures. Despite scalability concerns, both designs offer speed enhancements suitable for time-sensitive applications, presenting a valuable trade-off [21]. Öztekin et all. (2023) introduce a hardware-based approach to enhance the search operation for Opcode tables in assemblers by replacing conventional RAM with BiCAM. BZK.SAU.FPGA assembler was used as case study to demonstrate the effectiveness of parallel search operations which can be implement within a single clock cycle, . The BiCAM approach exhibits a fixed time complexity of O(1) regardless of memory or input size, outperforming software-based algorithms. While the BiCAM increases resource utilization and power consumption due to its hardware structure, its speed enhancement makes it a reasonable trade-off for time-sensitive applications, suggesting a novel assembler approach [22].

3. FPGA- BASED CONTENT ADDRESSABLE MEMORY

Field Programmable Gate Arrays (FPGAs) are programmable integrated circuits composed of configurable logic blocks (CLBs) connected by programmable interconnects [23]. First released in 1985 [24], FPGAs are widely recognized for their adaptability, as they can be used to create a wide range of digital circuits at different levels of complexity. Using Hardware Description Languages (HDL) such as Verilog or Very High Speed Integrated Circuit Hardware Description Language (VHDL) [25], FPGAs are programmed to execute parallel processing tasks in a variety of industries, including high-performance computing, data storage, wireless communications, and image/video processing [26].

Content Addressable Memory (CAM) functions as a memory type that quickly stores and retrieves data by employing a lookup-table function in a single clock cycle, utilizing dedicated comparison circuitry [27]. Described as hardware-based search engines, CAMs are adapted for applications requiring ultra-fast searching, achieved by comparing input data with stored data content rather than addresses, returning the matching data's address in just one clock cycle [28]. CAMs operate by associating data directly with unique keys, enabling direct retrieval based on content rather than memory addresses. CAMs typically consist of two main components: registers and a memory array. Registers are directly connected to each corresponding bit in the memory array and are primarily utilized for searching operations by by comparing their values with the target data . The memory array, on the other hand, serves as the storage component. CAM is often preferred over traditional memory types like Random Access Memory (RAM) in scenarios prioritizing speed. Unlike RAM, which retrieves data based on addresses and may require multiple clock cycles, CAM retrieves the address associated with a specific word in just one cycle [27]. Fig. 1 illustrates the search operation comparison between RAM and CAM (The choice of 64 bits is due to the memory block size used in many systems). In Table I, a summary of the important differences between RAM and CAM is provided. Moreover, FPGA-based CAMs are gaining popularity due to the hardware-like performance and software-like reconfigurability of FPGAs, particularly in complex reconfigurable systems. There are two primary categories of CAMs, each with a specific set of features. When working with binary data, Binary CAM (BiCAM) is used to match whole words. In addition to its binary states, Ternary CAM (TCAM) adds a "don't care" state for more flexible matching conditions [29, 30]. BiCAM can be preferred for search operations due to its fast parallel searching capabilities as it is able to reduce search latency and suitabile to operate in complex reconfigurable systems.

Table I Summary of Differences Between RAM and CAM.

Memory Type	**RAM**	**CAM**
Application	Utilized to execute programs and store the necessary data for them while they execute	Utilized in applications where extremely quick searching is required
Suitability	Appropriate for the sequential search technique	Duitable for parallel searching
Operation	Address is provided, RAM returns the data that is stored at that address.	A search key is provided, returns the address of the found data
Cost	less costly than CAMs	More expensive than RAM

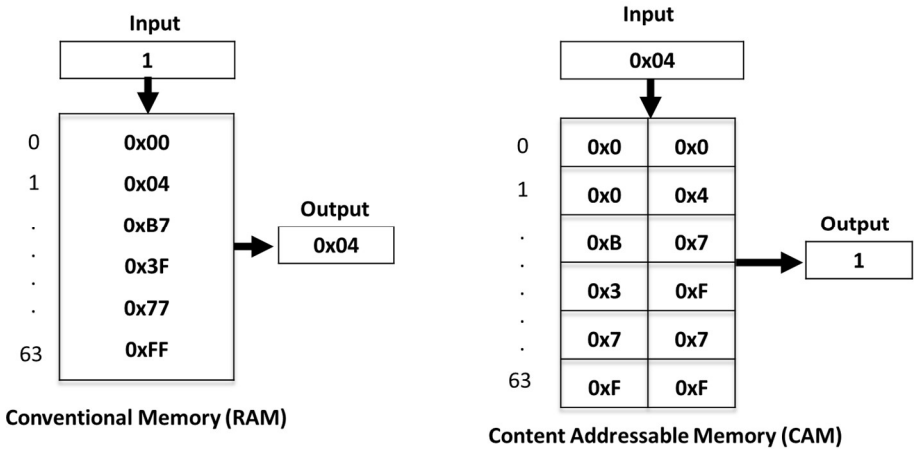

Fig. 1. Search operation RAM vs. CAM.

Although all these advantages offered by CAM are notable, it's important to acknowledge potential drawbacks, including increased power dissipation and higher hardware costs. To address these issues, various energy-saving designs and technologies for CAM have been proposed in the literature [31-33]. The choice between the advantages and potential trade-offs depends on the specific criteria and circumstances of the desired application.

4. Implementation

This study includes a comparison of three approaches: Sequential Search, Binary Search, and BiCAM-based search. The first two approaches utilize traditional RAM as storage elements and employ software-based algorithms to enhance computing system speed. In contrast, the third approach incorporates BiCAM as a replacement for RAM, representing a hardware-based enhancement. All proposed approaches are developed using the VHDL programming language and implemented at the logic gate level through Altera's Quartus II software. The implementation is carried out on the Cyclone® II 2C70 core FPGA board.

4.1. Sequential Search

Sequential search, referred to as linear search, is a straightforward searching algorithm that checks each element one by one in a list or array until a match is found [34,35]. The search starts at the first index of the list and moves through the list item by item until the desired element is located or the end of the list is reached [34,35]. Mathematically, it can be expressed as List[i] for i = 0, 1, 2, ..., N, where i denotes the position of the current element being examined, and N represents the total number of elements in the list. The algorithm can be simply represented in pseudocode:

	Algorithm I Sequential Searching Algorithm
1:	*Input:* list of elements (List), Target element (TE)
2:	*Output: Index of the TE if found, or 0 if not found*
3:	Search words (W_i) Sequentially, i= 0, 1, 2 …… N
4:	**If** TE matches W_i then
5:	Match occurs and index of the TE is sent to output
6:	**else**
7:	Mismatch occurs and error is sent to output
8:	**end if**

This straightforward algorithm is simple to comprehend and apply due to its simplicity. It is especially useful in situations with small datasets and a preference for algorithm simplicity over search speed. Sequential search does not require the list to be pre-sorted, in contrast to certain other algorithms. To implement this algorithm in VHDL, Algorithm I is utilized. Additionally, a 64x8 RAM is created and initialized with random values to implement the search process. In Fig. 2, a detailed explanation of the Sequential Search algorithm is provided to offer a clearer understanding. Additionally, in Fig. 3, the RTL (Register-Transfer Level) Diagram illustrating the VHDL code written for Sequential Search is presented.

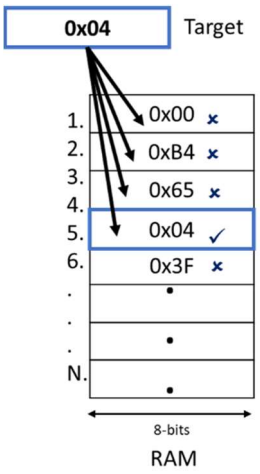

Fig. 2. Sequential Search Explanation

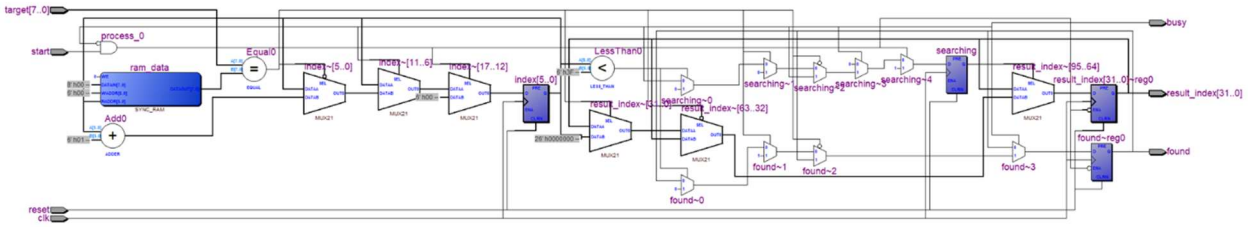

Fig. 3. The RTL Diagram of the Desgined Sequential Search.

4.2. Binary Search

One popular algorithm for quickly finding a target value in a sorted set of data is binary search. Binary search, also known as logarithmic search, works by continually dividing the search space in half to reduce the number of potential locations for the target [36,37]. It works especially well with sorted arrays or lists, as it reduces the number of comparisons required to locate an element. This makes it operate much more quickly than linear search algorithms, especially when dealing with larger datasets, as it drastically reduces the search space with each iteration. Mathematically, the formula for binary search entails iteratively dividing the search space in half and adjusting the boundaries until the target is found or the search space is depleted [36,37]. The formulaic representation is:

$$mid = \frac{low+high}{2} \qquad (1)$$

It is important to note the due to the integer division the result may truncate to the lower side which leds to overflow issues which can be avoided using the folwing formula:

$$mid = low + \frac{high-low}{2} \qquad (2)$$

The algorithm can be simply represented in pseudocode below:

Algorithm II Binary Searching Algorithm	
1:	*Input:* Sorted list of elements (List), Target element (TE)
2:	*Output:* Index of the TE if found, or 0 if not found
3:	*Set : low to 0, high to length of List – 1*
4:	**If** low <= high
5:	*Set :* mid to low + (high - low) / 2
6:	**If** If List[mid] equals TE then
7:	Match occurs and index of the TE is sent to output
8:	**else if** List[mid] < TE then
9:	*Set :* low to mid + 1 and go to step 4
10:	**else**
11:	Set : high to mid – 1 and go to step 4
12:	**end if**
13:	**else**
14:	Mismatch occurs and error is sent to output
15:	**end if**

This algorithm is particularly useful in scenarios where data is pre-sorted, as in databases or sorted arrays. Its advantages lie in its speed and efficiency, making it suitable for large datasets and real-time applications. To implement this algorithm in VHDL, Algorithm II is utilized and a 64x8 RAM is created and initialized with random values. In Fig. 4, a detailed explanation of the Binary Search algorithm is provided to offer a clearer understanding. Additionally, in Fig. 5, the RTL diagram illustrating the VHDL code written for Binary Search is presented.

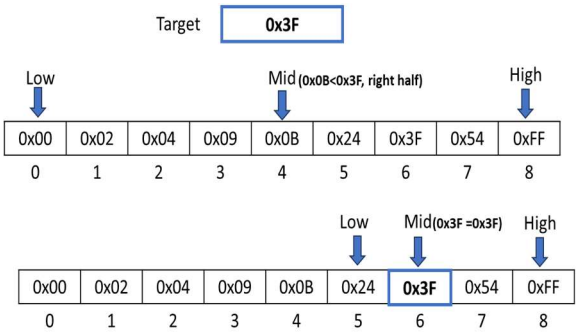

Fig. 4. Binary Search Explanation.

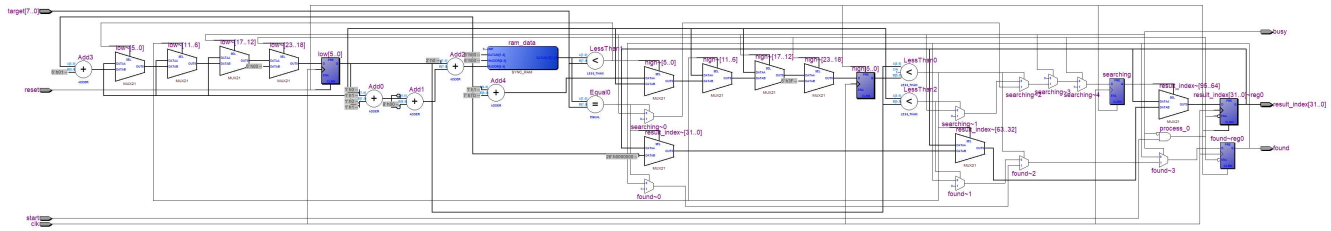

Fig. 5. The RTL Diagram of Desgined Binary Search.

4.3. BiCAM -based Search

The BiCAM is a specialized type of parallel memory architecture. Its objective is to provide an efficient, quick, and parallel search mechanism by searching for and retrieving data based on its content. In BiCAM, each memory location is associated with a unique key, and data is stored based on its content rather than its address. The BiCAM has two registers (Argument and key) used during searching. The argument register stores the value targeted for the search, while the search register is utilized for masking specific patterns that correspond to the desired matches in the search process. There are three main types of memory in modern FPGAs that can be used for designing BiCAM: Flip-Flops (FFs), Lookup Table RAM (LUT RAM), and Block RAM (BRAM) [38,39]. In this work, an FF-based design was chosen for its simplicity [38,39]. BiCAM's search function works by matching the stored data with the argument (search key) content. To identify the precise bit that needs to match, a mask (key) is used. The following is the formula that BiCAM uses for the match [40].

$$M = \mathrm{A}\, F_d \; + \; \overline{\mathrm{A}}\, \overline{F_d} + \overline{\mathrm{K}} \qquad (4)$$

Where M is equal to the Output of the comparison circuit, F_d is the output of data, A is the value of the argument register, and K is the value of the key register. The algorithm used to implement the search operation can be simply represented in pseudocode below.

Algorithm III FF-based BiCAM Searching Algorithm	
1:	*Input:* Argument element (A), key element (k)
2:	*Output: content of the A if match, or 0 if not found*
3:	$M = \mathrm{A}\,F_d + \overline{\mathrm{A}}\,\overline{F_d} + \overline{\mathrm{K}}$
4:	**If** M equal to 1 then
5:	Match occurs and content is sent to output
6:	**else**
7:	Mismatch occurs and 0 is sent to output
8:	**end if**

The VHDL code is written according to Algorithm III, and a 64x8 FF-based BiCAM is created and initialized with random values. For a better understanding, a detailed explanation of the FF-based BiCAM Search algorithm is given in Fig. 6 Moreover, the RTL diagram displaying the VHDL code created for FF-based BiCAM search is shown in Fig. 7.

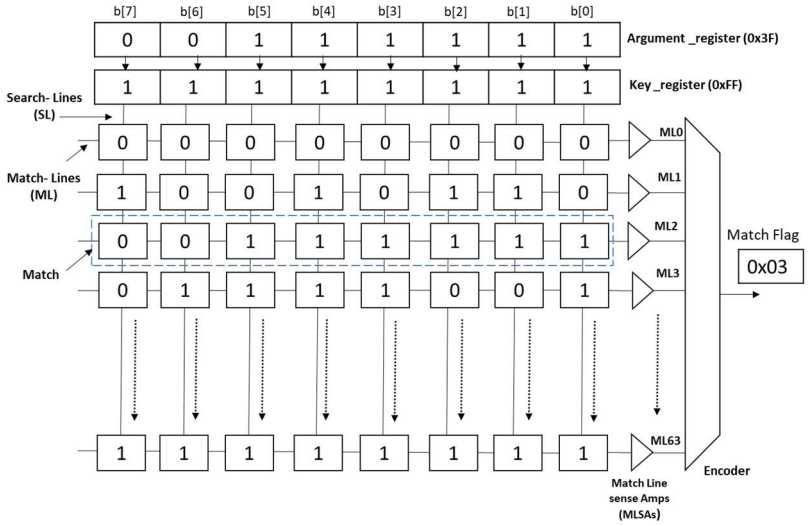

Fig. 6. FF-based BiCAM Search Explanation.

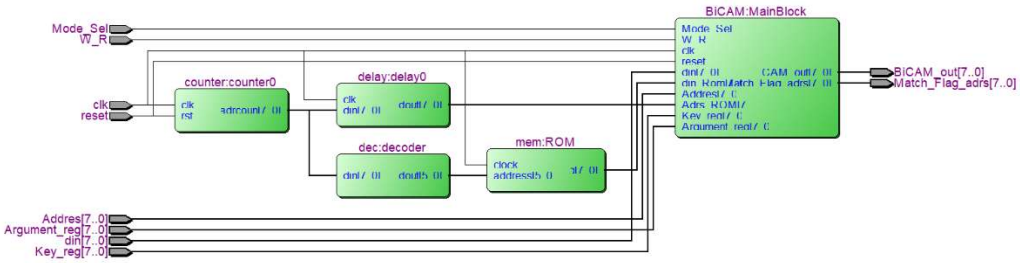

Fig. 7. The RTL Diagram of designed FF-based BiCAM.

5. Evaluation

Random values are used to initialize the RAM for both sequential and binary search, as well as to initialize the BiCAM. Subsequently, one of these values is randomly selected to bed searched for. In computer science, analyzing algorithms is essential to finding the best solution

for a particular issue. Even though there might be several algorithms available for a given problem, figuring out which one is the most effective is still difficult. The correctness of an algorithm determines its acceptability, whereas the execution time of the algorithm determines its efficiency. Comparing algorithms based on the number of steps compared to the size of the input is made easier by time complexity, which is expressed using the 'Big O' notation [41]. Time complexity for search operations is commonly expressed as a function of 'n', which stands for the elements that are being searched. Three scenarios are taken into account in time complexity analysis for search algorithms: best-case, average-case, and worst-case. This study compares the time complexity of each of the chosen algorithms. The scenarios for these search algorithms are compiled in Table II.

Table II Time complexity of the proposed algorithms.

Algorithm	Best case	Average case	Worst case
Sequential search	O(1)	O(n)	O(n)
Binary search	O(1)	O(log n)	O(log n)
BiCAM Search	O(1)	O(1)	O(1)

Examining Table II shows that, when compared to other traditional RAM-based methods, the BiCAM search algorithm has the best time complexity value. Given that its access times are constant O(1). To gain a comprehensive understanding of the proposed approaches, alongside the time complexity analysis, an examination of source utilization and power efficiency was also conducted.

6. Results and Discussion

After implementing the proposed algorithms, they were simulated using the ModelSim HDL simulator. RAM and BiCAM were initialized with random values; however, it's important to note that the RAM for binary search should be sorted, which may require additional time for sorting. This sorting step is not considered in our work as we focus only on the search algorithm. A search value was selected to be searched under various conditions. Fig. 8 illustrates some of the simulation results of the proposed search algorithms. As expected the FF-based BiCAM search approach demonstrated a constant time complexity of O(1), meaning that a match could be found in a single clock cycle, regardless of the BiCAM configuration or input size. On the other hand, the time complexity of sequential search is directly proportional to the array's length, leading to a linear increase in the number of comparisons as the array size grows. Lastly, the time complexity required for binary search is O (log n), indicating a logarithmic increase with

the length of the array, resulting in faster search times for larger arrays compared to linear search. From Fig. 8. we can analyze the simulation results of linear, binary, and BiCAM-based searching operations using memories of the same size and same initialized values. The target value chosen for the search is 12, stored at address 18. In the case of linear search, it took 18 clock cycles to find the target, while binary search took 5 clock cycles. As expected, BiCAM-based search achieved the result in just one clock cycle. Additionally, values 34 and 03 were chosen for demonstration purposes, showcasing the power of the BiCAM-based search.

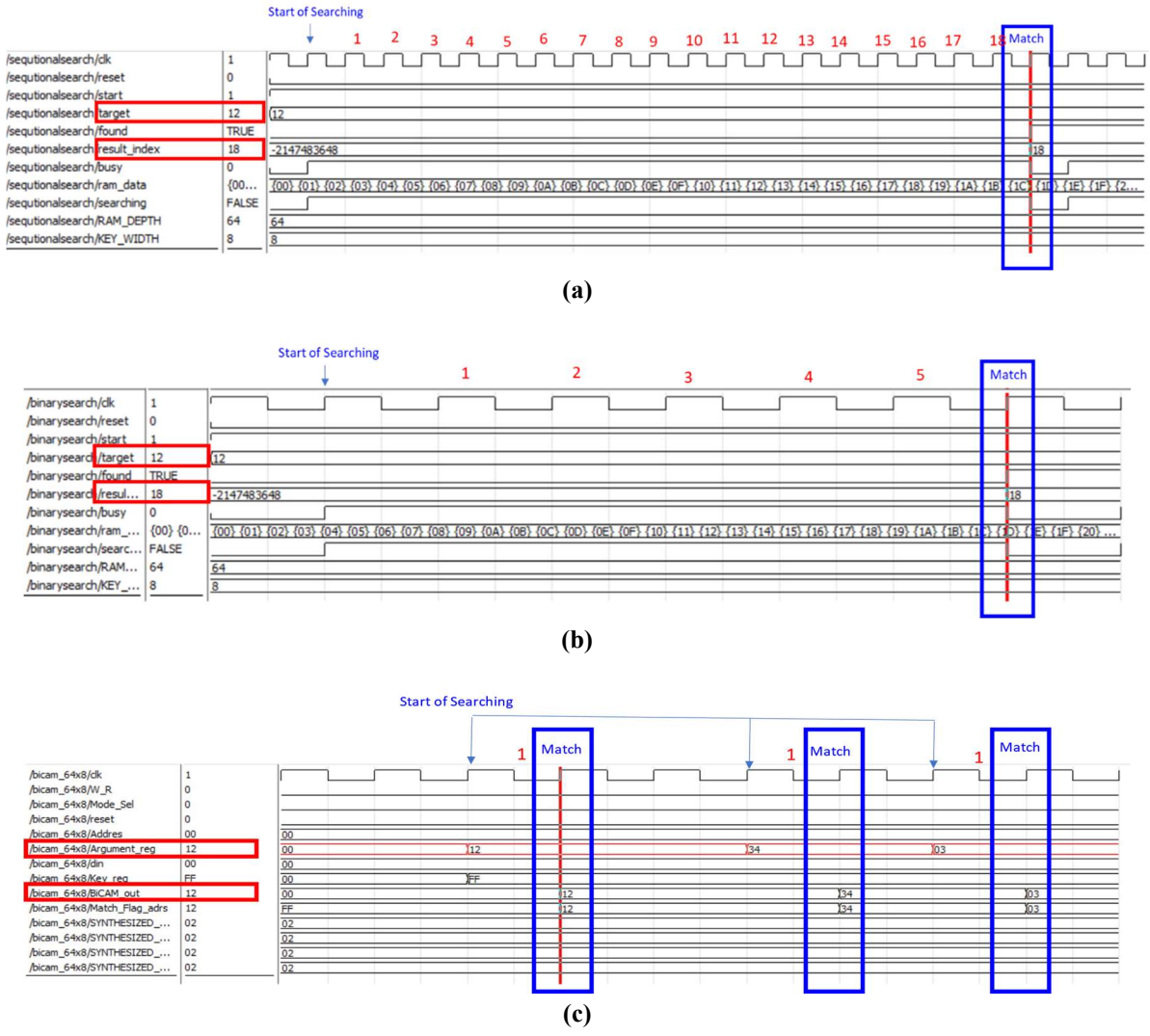

Fig. 8. Simulation results of the proposed search algorithm. (a) Sequential ;(b) Binary;(c) FF-based BiCAM.

To gain a deeper understanding of the proposed methods, an analysis of source utilization and power efficiency has been conducted in addition to the simulation results.

6.1. Source Utilization

Table III provides an overview of the source utilization analysis for the proposed approaches. From the table, it is observed that the FF-based BiCAM approach exhibits a higher usage of logical elements compared to the alternative approach, confirming our predictions. The FF-based BiCAM integrates encoders and decoders into its architecture, utilizing FFs as the storage element. In contrast, the alternative method relies on onboard memory bits specified by the factory.

Table III Source utilization of proposed search algorithms.

Search Algorithm Type	Sequential	Binary	FF-based BiCAM
Total Logic Elements	22	65	2,261
Dedicated Logic Registers	14	20	1,550
Total Combinational Functions	16	65	1,598
Total Registers	14	20	1550
Total Pins	45	45	60
Total Memory Bits	0	0	512

From the Table III As we can see that The FF-based BiCAM algorithm stands out with higher usage in total logic elements, dedicated logic registers,Total Pins,Total Registers,Total Memory Bits ,and total combinational functions compared to Sequential and Binary algorithms. Despite a relatively high usage of total logic elements, the FF-based BiCAM demonstrates noteworthy efficiency considering to the other approaches. This highlights the FF-based BiCAM's efficiency and resource utilization strengths in comparison to Sequential and Binary algorithms.

6.2. Power Consumption

A graph depicting the total thermal power dissipation obtained by powerplay power analyzer tool of the proposed search algorithms is presented in Fig. 22. The illustration reveals that the RAM-based technique consumes 19.8% less power compared to the BiCAM-based approach. The increased power consumption of the BiCAM-based approach is primarily related to the utilization of numerous FFs and parallel comparison circuitry. This leads to additional power dissipation since the compare operations circuitries are typically active for the majority of the time, as discussed in section 3. This presents a significant challenge for designers aiming to minimize power consumption in parallel-based compare operations. However, for time-sensitive

applications, the higher power dissipation of the suggested BiCAM may still represent a reasonable trade-off for increased speed.

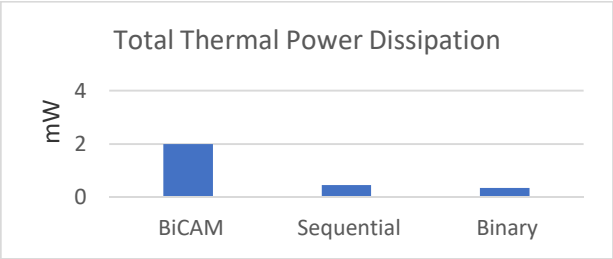

Fig. 9. Total thermal power dissipation of the proposed search algorithms.

7. Conclusion

In the rapid evolution of computer science and engineering, the optimization of search algorithms stands as a critical aspect to enhance application performance. This paper focuses on comparing various FPGA-based search algorithms, including sequential, binary, and BiCAM-based techniques. The fundamental search function is crucial for computer science applications such as text and database searching. Besides the BiCAM, two traditional search algorithms, sequential and binary search, are chosen to be examined. The sequential search represents a straightforward and simple search. In contrast, binary search is a more sophisticated and logarithmically efficient algorithm. The analysis focuses on the strengths and weaknesses of these algorithms, particularly when implemented on FPGAs. An important point of exploration in this paper is the introduction and examination of BiCAM, a hardware-dependent technique for search operations. BiCAM, as part of CAM, facilitates parallel searching, eliminating the need for sequential scanning and significantly enhancing search speed. The study explores various FPGA-based search algorithms, highlighting the versatility of FPGA technology. The construction of FPGA-based sequential, binary, and BiCAMs is discussed, presenting both traditional approaches and a novel one. The evaluation metrics include time complexity, resource utilization, and power efficiency. The results indicate that the FF-based BiCAM outperforms other traditional search algorithms in terms of time complexity, as it can find a match within a single clock cycle regardless of the size of the BiCAM. However, due to its hardware structure and parallelism, it may face challenges compared to other traditional methods. The binary search exhibits better performance, but it requires pre-sorting. Thus, there exists a tradeoff when choosing which algorithm to use. In conclusion, this paper provides insights into FPGA-based search algorithms, highlighting the potential efficiency gains of BiCAM. The proposed methods contribute valuable perspectives to ongoing discussions in computer science, engineering, and hardware design.

References

[1] Cho, S., Martin, J.R., Xu R., Hammoud, M.H., Melhem, R. (2007). CA-RAM: A High-Performance Memory Substrate for Search-Intensive Applications. 2007 IEEE International Symposium on Performance Analysis of Systems & Software, 230-241.

[2] Knuth, D.E. (1973). The Art of Computer Programming, Vol. 3 (Sorting and Searching), Addison-Wesley, USA

[3] Schlesinger, R. (2009). Developing Real World Software, Jones & Bartlett Publishers

[4] Mano, M.M. (1993). Computer System Architecture. Pearson, USA

[5] Levitin, A. (2012). Introduction of the Design & Analysis of Algorithms. Addison-Wesley, USA

[6] Cao, Y., Qi, H., Zhou, W., Kato, J., Li, K., Liu, X., Gui, J. (2018). Binary Hashing for Approximate Nearest Neighbor Search on Big Data: A Survey. IEEE Access 6, 2039-2054.

[7] Irmayana, A., Sy, H., Paulus, Y.T., Aini, N., Aryasa, K.B. (2021). A Systematic Comparative Study of Linear, Binary and Interpolation Search Algorithms. 3rd International Conference on Cybernetics and Intelligent System (ICORIS) 1-5.

[8] Sultana, N., Paira, S., Chandra, S., Alam, S.S. (2017). A brief study and analysis of different searching algorithms. Second International Conference on Electrical, Computer and Communication Technologies (ICECCT) 1-4.

[9] Fukac, T., Korenek, J. (2019). Hash-based Pattern Matching for High Speed Networks. IEEE z22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS) 1-5.

[10] Balogun, G.B. () A Modified Linear Search Algorithm. African journal of computing & ic 12(2), 43 – 54.

[11] Parmar, V. P., Kumbharana, C.K. (2015). Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List. International Journal of Computer Applications,121 (3),13-17.

[12] Arora, N., Bhasin, G., Sharma, N. (2014). Two way Linear Search Algorithm. International Journal of Computer Applications, 107, 6-8.

[13] Nowak, R. (2008). Generalized binary search. 2008 46th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, 568-574.

[14] Jacob, A. E., Ashodariya, N., Dhongade, A. (2017). Hybrid search algorithm: Combined linear and binary search algorithm," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India.1543-1547.

[15] Tiong, M. C. , Daniyal, H., Sulaiman, M. H. , Bakar, M. S. (2017). Binary search algorithm as maximum power point tracking technique for photovoltaic system under partial shaded conditions,2017 IEEE Conference on Energy Conversion (CENCON), Kuala Lumpur, Malaysia, 10-14.

[16] Jin, Z., Finkel, H. (2020) Performance Evaluation of the Vectorizable Binary Search Algorithms on an FPGA Platform. 2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3), GA, USA, 63-67.

[17] Bennett, E.O., Ejiofor, V.E., Akpan, R.I. (2015). Efficient algorithm for binary search enhancement. 22(1).

[18] Garzón, E., Yavits, L., Teman, A., Lanuzza, M. (2023). Approximate Content-Addressable Memories: A Review. Chips, 2(2),70-82.

[19] Öztekin, H. (2022). BiCAM-based automated scoring system for digital logic circuit diagrams. Open Chemistry, 20(1), 1548-1556.

[20] Garzón E, Lanuzza M, Teman A, Yavits L (2023) AM4: MRAM Crossbar Based CAM/TCAM/ACAM/AP for In-Memory Computing. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 13(1), 408-421.

[21] Lazzem, A., Öztekin, H., Pehlivan, İ. (2023). A case study: Understanding The Nature of Memories Architectures in FPGAs to Built-up Bi-CAM. Mühendislik Bilimleri ve Araştırmaları Dergisi, 5 (1), 47-56.

[22] Öztekin, H., Lazzem, A. & Pehlivan, İ. (2023). Using FPGA-based content-addressable memory for mnemonics instruction searching in assembler design. J Supercomput 79, 17386–17418.

[23] Boutros, A. , Betz, V. (2021). FPGA Architecture: Principles and Progression. IEEE Circuits and Systems Magazine, 21(2),4-29.

[24] Trimberger, S. (1995). FPGA Technology: Past, Present, and Future", ESSCIRC '95: Twenty-first European Solid-State Circuits Conference,12-15.

[25] Kasivinayagam, G., Skanda, R., Burli, A.G., Jadon S., Sidhu,R. (2022) Hardware Description Language Enhancements for High-Level Synthesis of Hardware Accelerators, Advances in Computing and Data Sciences,1613,1-12.

[26] Gandhare, S., Karthikeyan, B. (2019). Survey on FPGA Architecture and Recent Applications", 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN),1-4.

[27] Pagiamtzis, K., Sheikholeslami, A. (2006). Content-addressable memory (CAM) circuits and architectures: a tutorial and survey. IEEE Journal of Solid-State Circuits 41(3),712-727.

[28] Sivakumar, SA., Swedha, A., Naveen, R. (2018) Survey of Content Addressable Memory. International Journal of Creative Research Thoughts 6(1):1516-1526.

[29] Jothi, D., Sivakumar, R. (2018) Design and Analysis of Power Efficient Binary Content Addressable Memory (PEBCAM) Core Cells", Circuits, Systems, and Signal Processing, 37(6),1422–1451.

[30] Zackriya, M. V., Kittur, H. M. (2016). Precharge-Free Low-Power Content-Addressable Memory, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 24(8),2614-2621.

[31] Wasmir, S.H., Venkata, T.M. , Sandeep, S., Anup, D. (2020). Low-Power Content Addressable Memory Design using Two-Layer P-N Match-Line Control and Sensing.Integration, 75,73-84.

[32] Chang, Y.J., Tsai,K. , Cheng, Y.C., Lu, M.R. (2020). Low-power ternary content-addressable memory design based on a voltage self-controlled fin field-effect transistor segment. Computers & Electrical Engineering 81:106528.

[33] Jiang, S., Yan, P., Sridhar, R. (2015). A high speed and low power content- addressable memory (CAM) using pipelined scheme. 2015 28th IEEE International System-on-Chip Conference, 345-349.

[34] Aremu, B. (2023). Introduction to Algorithms and Data Structures: A Solid Foundation for the Real World of Machine Learning and Data Analytics. Nigeria: Ojula Technology Innovations.

[35] Sultana, N., Paira, S., Chandra, S., Alam, S. K. S. (2017). A brief study and analysis of different searching algorithms," 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 1-4.

[36] Sanders, P., Mehlhorn, K., Dietzfelbinger, M., Dementiev, R. (2019). Sequential and Parallel Algorithms and Data Structures: The Basic Toolbox. Germany: Springer International Publishing.

[37] Search Algorithm: Fundamentals and Applications. (2023). One Billion Knowledgeable.

[38] Irfan, M, Sanka, A.I., Ullah, Z., Cheung, R.C.C. (2021). Reconfigurable content-addressable memory (CAM) on FPGAs: A tutorial and survey, Future Generation Computer Systems, 128, 451-465, 2021.

[39] Pagiamtzis, K., Sheikholeslami, A. (2006). Content-addressable memory (CAM) circuits and architectures: a tutorial and survey. IEEE Journal of Solid-State Circuits 41(3):712-727.

[40] Sipser, M. (2012). Introduction to the Theory of Computation", Cengage Learning .

[41] Chatterjee, A., Kiao, U. (2021) Time Complexity Analysis (Coding Interviews: Algorithm and Data Structure Proficiency), Independently published.