BMBAD BJESR Mühendislik Bilimleri ve Araştırmaları Dergisi Journal of Engineering Sciences and Researches

A case study: Understanding The Nature of Memories Architectures in FPGAs to Built-up Bi-CAM

Bi-CAM İnşa Etmek İçin FPGA'lardaki Belleklerin Mimarilerinin Doğasını Anlamak: Bir Vaka Çalışması

[1]Abdelkader LAZZEM, [2]Halit ÖZTEKİN, [3]İhsan PEHLİVAN

[1]*Department of Electrical-Electronics Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye*
[2]*Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye*
[3]*Department of Electrical-Electronics Engineering, Sakarya University of Applied Sciences, Sakarya, Türkiye*

[1]y190004033@subu.edu.tr, [2]halitoztekin@subu.edu.tr, [3]ipehlivan@subu.edu.tr

Araştırma Makalesi/Research Article

ARTICLE INFO

Article history

Received : 13 January 2023
Accepted : 14 March 2023

Keywords:

Field-programmable Gate Array (FPGA), Search Operation, Binary Content Addressable Memory (BiCAM), Flip-flop (FF), Block Random-access Memory (BRAM)

ABSTRACT

This work gives a comparison between two approaches used for improving search operation speed by using FPGA-based Binary Content Addressable Memory (BiCAM), which is a parallel type of computer memory that quickly searches for and retrieves specific data stored within the memory by assigning a unique address to each piece of data. This hardware-based technique is more efficient than traditional software-based techniques. The FPGA-based BiCAM is implemented using two different approaches: using Flip-flops and Block Random Access Memory as the memory element. The performance of these implementations is evaluated through Time complexity analysis, resource utilization, and search speed. The results indicate that both approaches have same Time complexity of O(1) and differ in resources used and power dissipation. As a result, the increasing demand for faster and more efficient search operations, both approaches can play an important role in optimizing search operations according to the application and the resources available.

MAKALE BİLGİSİ

Makale Tarihleri

Gönderim : 13 Ocak 2023
Kabul : 14 Mart 2023

Anahtar Kelimeler:

Alan Programlanabilir Kapı Dizisi(FPGA), Arama İşlemi , İkili İçerik Adreslenebilir Bellek (BiCAM), Flip-flop (FF), Blok Rastgele Erişim Belleği (BRAM)

ÖZET

Bu çalışmada, FPGA tabanlı İkili İçerik Adreslenebilir Bellek (BiCAM) kullanarak arama işlem hızını arttırmak için kullanılan iki yaklaşım arasındaki karşılaştırmayı verir. BiCAM, her veri parçasına benzersiz bir adres atayarak bellekte saklı olan belirli verileri hızlı bir şekilde arayan ve alan paralel bir bilgisayar belleğidir. Bu donanım tabanlı teknik, Linear, Binary ve hash tabanlı gibi geleneksel yazılım tabanlı tekniklerden daha verimlidir. FPGA tabanlı BiCAM'a, Flip-flops ve Block Random Access Memory gibi iki farklı yaklaşımla uygulanmıştır. Bu uygulamaların performansları, zaman karmaşıklığı analizi, kaynak kullanımı ve arama hızı açısından değerlendirilmektdir. Sonuçlar, her iki yaklaşımın da O(1) Zaman karmaşıklığına sahip olduğunu ve kaynak kullanımı ve güç dağılımı açısından farklılık gösterdiğini göstermektedir. Bu nedenle, daha hızlı ve verimli arama işlemlerine olan artan talep göz önüne alındığında, her iki yaklaşım da uygulamanın ve mevcut kaynaklarına göre optimize edilmesinde önemli bir rol oynayabilir.

ORCID: [1]0000-0003-0136-356X
[2]0000-0001-8598-4763
[3]0000-0001-6107-655X

1. INTRODUCTION

The search operation is a process of locating a specific piece of information or data within a larger set of information. The search operation typically involves comparing the search term with the data stored in a particular data structure or database and returning the data if a match is found. The search operation can be done using different algorithms and techniques, such as linear search, binary search, or hash table search [1]. Generally, the efficiency of the search operation depends on the type of data structure used, the size of the data, and the search algorithm employed. Search operations are a fundamental operation in computer science and can be a critical factor in time-sensitive applications. it is used in many applications, such as database querying, text search, and image recognition etc.

In recent years, there has been a significant increase in the demand for faster and more efficient search operations due to the rapid development of technologies and the adoption of new methods. There have been various suggestions for improving search speed using software-based approaches, but these methods may be limited due to the structural constraints of the memory, as they do not involve changing in the hardware itself. In this work, we propose an approach for improving search operation speed within a memory using Binary Content Addressable Memory (BiCAM) as a hardware-based technique, rather than the commonly used software-based techniques. This approach relies on changing the structure of memory to improve search performance. BZK.SAU.FPGA assembler is chosen as the case study of this work to demonstrate the effectiveness of using BiCAM as a replacement for the Random-access Memory RAM to implement its search operations [2]. Figure 1 illustrates an example of a search operation performed on both RAM and CAM. In this example, both types of memory are searched for a specific value, but how the search is performed is different.

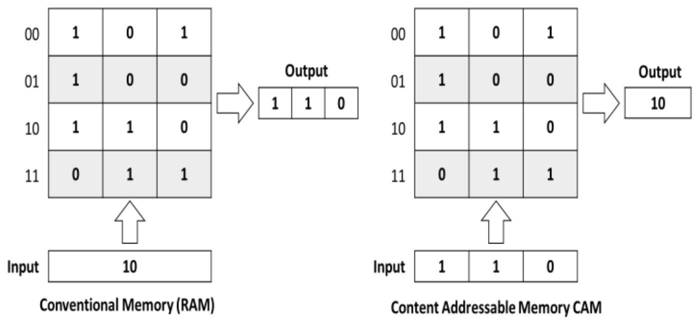


Figure 1. RAM vs. CAM for reading operation.

A Content Addressable Memory (CAM) is a highly parallel type of computer memory that is used to quickly search for and retrieve specific data stored within the memory. It works by assigning a unique address to each data, rather than using the index as in traditional memory systems [3]. When a search is performed, the CAM compares the search data with all the data stored in the memory. If a match is found, the data is retrieved and returned to the user. CAM eliminates the need to search sequentially through the entire memory, making it a faster way to access data. It is commonly used in networking devices, such as switches and routers, to quickly look up and forward data packets based on their destination addresses and applications that require search operations [4,5]. There are two types of Content Addressable Memory (CAM): Binary CAM (BiCAM), in which each memory unit cell can only have binary states of low ‘0’ or high ‘1’, and Ternary CAM (TCAM), which has the same binary states as BiCAM in addition to a third state which is a don’t care “x” state [6, 7]. CAMs are typically implemented using specialized hardware, such as dedicated Application-specific integrated circuits (ASICs) or field-programmable gate arrays (FPGAs) [8]. While FPGAs are more flexible and reconfigurable than ASICs, they do not have a dedicated hardware CAM block unit [8]. Therefore, several methods have been developed to implement CAMs on FPGAs, including using Block RAM (BRAM) [9,10], Distributed RAM (DRAM) [11], or Flip-flops (FFs) [12]. For this study, BiCAM was chosen as we are interested in the exact match cases and the don't care state is not needed.

Briefly, the aim of this work is to enhance the understanding of the construction of FPGA-based BiCAMs, which can be used as a hardware-based improvement for searching operations in computing system applications rather than software-based enhancement. To achieve this goal, the study focuses on exploring the diverse memory types available on FPGA boards and how they can be utilized to develop BiCAMs in a more comprehensive and detailed manner and implement the BiCAM using two different approaches. The first approach uses Flip-flops (FFs) as the memory element and the second one uses Block Random access memory BRAM as the memory element. To evaluate both approaches Time complexity analysis, resource utilization, and power efficiency is presented.

The remainder of this thesis is organized as follows: in section 2 a short explanation of the BZK.SAU.FPGA assembler and why a search operation is required. In section 3, an overview of FPGA-based Content Addressable Memories (CAMs) is provided. The section begins with a general introduction to FPGAs and their main components. Then, an explanation of the types of memory that can be utilized in FPGAs to create a BiCAM with their lecturer review is presented. In section 4 the implementation of both approaches is explained. section 5 explains the evaluation of both approaches. In chapter 6 the results are displayed and discussed. Finally, in chapter 7 the conclusion of this work is given.

2. RELATED WORK

The use of FPGA-based CAMs is increasingly popular in the field of computer engineering, particularly with the current advancements in technology and the growing need for fast access to information. Several studies propose using various memory types to construct CAMs, including BRAM-based, LUTRAM-based, and FF-based CAMs. Zahur et al. (2012) proposed a 512x36 HP-TCAM design that utilized 36k x 56 BRAMs and was implemented for the first time on a Virtex-6 Xilinx FPGA device [13]. Weirong (2013) proposed a 1024x150 Scalable TCAM design using a Virtex-7 FPGA device that considered redesign and utilized 512x272 BRAMs [14]. Zhuo and Martin (2014) proposed a 504x180 Hierarchical TCAM design on a Virtex-6 FPGA device that utilized 36K x 140 BRAMs and demonstrated increased hardware efficiency and low latency [15]. Zahur et al. (2015) proposed a 512x36 Z-TCAM design on a Virtex-6 Xilinx FPGA device that reduced search latency but did not consider redesign or data preparation [16]. Inayat et al. (2018) presented a 512x36 multi-pumping TCAM design on a Virtex-6 FPGA device that utilized 36K x 16 BRAMs, demonstrated increased hardware efficiency, and low latency [17]. Moreover, several studies in the literature propose using LUTRAM. Muhammad et al. (2019) proposed a 512x36 Zi-CAM design implemented on a Virtex-7 FPGA device. The design featured low power consumption for certain data sets but had a low speed and did not support ternary bits [19]. Pedro et al. (2019) proposed a 512x40 PR-TCAM design also implemented on a Virtex-7 FPGA device. The design included the feature of partial reconfiguration but had a slow update process [20]. Muhammad et al. (2019) also proposed a 512x36 D-TCAM design implemented on a Virtex-6 FPGA device. The design had high throughput but was unable to update modules [20]. Inayat et al. (2019) presented a 512x36 DURE design implemented on a Virtex-6 FPGA device. This design had easy dynamic updating but low throughput [21]. Finally, for FF-based CAMs, Hassan et al. (2019) proposed a 64x36 RE-TCAM design implemented on a Virtex-6 FPGA device. The design reduced hardware resource utilization but required more I/O pins and needed the input data to be masked [22]. Zahur et al. (2017) proposed a 64x36 LH-CAM design implemented on a Virtex-6 FPGA device. The design had improved speed compared to BRAM-based CAMs but had complex routing and was not scalable [12]. Muhammad et al. (2017) also proposed the first FF-based TCAM design, called G-AETCAM, which was implemented on a Virtex-6 FPGA device and had a size of 64x36. The design had better performance compared to BRAM-based CAMs but was unscalable [23]. Additionally, CAMs can be constructed using a combination of these memory types. For example, Irfan et al. (2022) introduced a type of TCAM called Comp-TCAM, which is constructed by using both BRAM and LUTRAM [24].

In summary, BRAM-based CAMs may have difficulties with the pre-processing of data and may require it to be in a certain order. LUTRAM-based CAMs may have issues with wide bitwise ANDing and high routing complexity. FF-based CAMs use FFs as their memory elements, which can reduce hardware costs per bit, but they may struggle with scalability and high-power consumption. However, they tend to be more effective in terms of throughput, energy consumption, and hardware resource utilization for small CAMs. In the future, CAM designs may be able to reduce the number of resources required per cell, potentially increasing their scalability. This is an area of potential future research.

3. CASE STUDY BZK.SAU.FPGA

BZK.SAU.FPGA is an FPGA-based microcomputer architecture design that was implemented on Altera's Cyclone II Development board by using a Computer Architecture Simulator known as BZK.SAU [25]. BZK.SAU assembler was mainly designed as an educational tool forwarded to undergraduate students to improve their understanding of computer science courses. It has been used by the Computer Engineering Department at Sakarya University since 2009 [13]. It has its own assembler that was designed especially for it. The design has a drawback of not providing a floating-point arithmetic unit, which has been addressed and improved through the study cited in [26]. The Brute-force algorithm was used as a search operation algorithm to convert the user source code to the machine code [27]. BZK.SAU.FPGA, like other programmable devices that use assembly language, relies on RAM to store its instruction sets in mnemonic form and convert them into machine code for internal operations and to communicate with the outside world. However, the search operation using RAM can be slow due to its hardware constraints, which can be problematic in time-sensitive applications. To address this issue, it has been suggested that the RAM be replaced with an FPGA-based BiCAM as its storage unit, that can perform search operations more quickly. Table I. gives a design summary of BZK.SAU.FPGA Microcomputer architecture. For more details about BZK.SAU.FPGA microcomputer architecture references [28, 29] can be checked.

Table 1. BZK.SAU.FPGA Microcomputer architecture design summary [17].

Feature	Explanation
Built-in media	Altera DE2-70 FPGA
Built-in	Schematic design
Keyboard	Full-stroke “clicky”. keys
Text modes (display)	24 lines × 40 columns
Graphics modes (display)	320 × 384
COLO rsc	Monochrome in VGA mode
RAM	64 KB
ROM	4 MB
Memory endianness	Big-Endian
CPU design	CISC
CPU architecture	Von-neumann (SISD)
Address and data bus	16-bit
The number of GPRs/data and address registers	16
Control unit	Hardware control
The processing of instructions	Non-pipeline
ALU	16-bit (only integers)
Data representation	2's complement
OS	Single user-single task
Written in language	BZK.SAU assembly language
File system	FAT

4. FPGA- BASED CONTENT ADDRESSABLE MEMORY (CAM)

Field Programmable Gate Arrays, which are abbreviated as FPGAs, are a type of integrated circuit that can be programmed to implement various digital circuits. They are made up of configurable logic blocks (CLBs) connected by programmable interconnects [30]. FPGAs have been used in the digital circuit industry since they were first developed in 1985 [31], and they are known for their ability to be configured to design and implement a wide range of digital circuits at various levels of complexity, from high level down to transistor-level architectures. FPGAs can be programmed using Hardware Description Languages (HDL) such as Verilog or VHDL [32]. These programmable integrated circuits have a wide range of applications in various industries due to their ability to perform tasks in parallel, including high-performance computing and data storage, wireless communications, and video & image processing [33].

The use of ASIC-based CAMs, which may be called circuit-level or transistor-level CAMs comes with a difference in cost based on the number of transistors used per CAM cell. This can be caused by the difference in hardware architecture of BiCAM or TCAM and depending on the type of logic gate used. Also, the custom production feature of ASIC-based CAMs makes them unusable in case of manufacturing error as well as unscalable. On the contrary, we find that FPGA-based CAMs are becoming more popular due to their FPGAs' hardware-like performance and software-like reconfigurability, especially in complex reconfigurable systems. This technology could be beneficial for implementing CAMs because it allows fast and parallel search operations that can retrieve the searched word in only a single clock cycle. Although CAMs are useful in certain applications, such as networking (routing), most current FPGA boards do not have a hard intellectual property (IP) core in their architecture [8]. However, modern FPGAs do offer an enormous number of logic and memory resources that can be used to design and implement CAM-based applications, plus some FPGA boards manufacturers such as Xilinx offer CAM soft IP cores [34,35].

Generally, each FPGA device is composed of a specific number of resources, programmable interconnects, and I/O blocks that allow the implementation of reconfigurable digital circuits [36]. A general FPGA structure is presented in Figure 2. The main components of an FPGA's structure are the Configurable Logic Blocks (CLBs) and routing resources, which are arranged in a matrix format. CLBs are the basic logic units of an FPGA and consist of FFs and Lookup Tables (LUTs) that are used to implement logic functions. The routing resources connect the various logic blocks to create the desired digital system. Each switching node in the routing resources is typically made up of six transistors, which are controlled by an SRAM configuration memory [8]. Figure 3 shows the architecture of a modern FPGA.

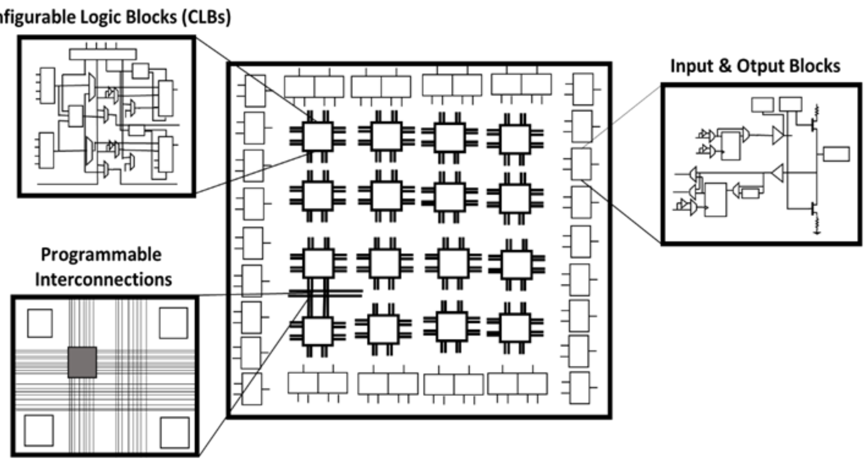

Configurable Logic Blocks (CLBs)
Input & Otput Blocks
Programmable Interconnections

Figure 2. The main components of the conventional FPGAs [36].

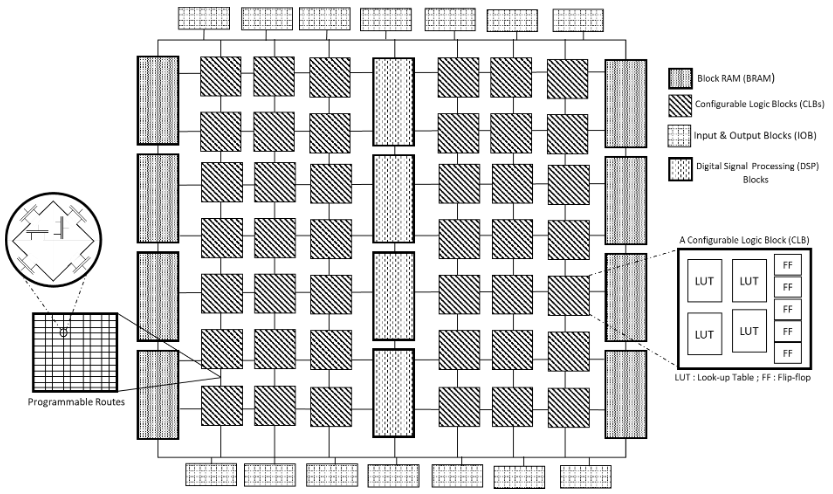

Block RAM (BRAM)
Configurable Logic Blocks (CLBs)
Input & Output Blocks (IOB)
Digital Signal Processing (DSP) Blocks
A Configurable Logic Block (CLB)
LUT
FF
LUT : Look-up Table ; FF : Flip-flop
Programmable Routes

Figure 3. A modern FPGA architecture [8].

4.1. Memories of Modern FPGAs

Understanding the types of memory available in FPGAs is essential as they are typically used to build FPGA-based CAMs. There are three main types of memory that can be found in modern FPGAs: Block RAM (BRAM), Look-up Table (LUT) RAM, and Flip-flops (FFs) [8].

4.1.1. Block RAM (BRAM)

Block Random-access Memory, or BRAM, is a sort of RAM used for data storage that is integrated throughout the FPGA [37]. Figure 4 shows the block diagram of the BRAM. BRAM can be a dual-port memory that allows for fast and efficient access to large amounts of data, making it well-suited for applications that require fast access to large amounts of data. In the literature, many studies on BRAM-based CAMs are proposed.

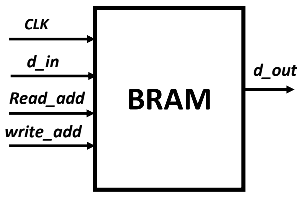
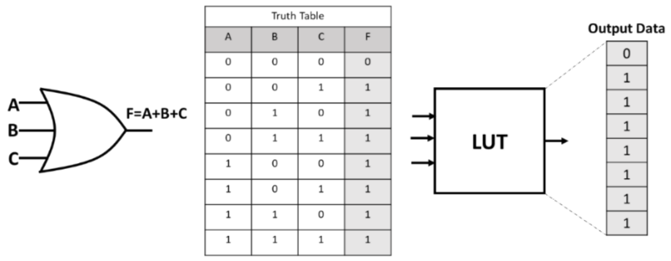


Figure 4. Single-port BRAM block diagram.

4.1.2. Look-up Table RAM (LUTRAM)

Look-Up Table Random-Access Memory (LUTRAM) is a type of memory that combines the features of a LUT and a RAM where LUTs are digital circuits that can be configured to implement any Boolean function by storing a truth table [38]. Figure 5 shows an example of a LUT with its truth table and logic gate symbol. When inputs are applied to a LUT, the output value is determined by looking up the corresponding entry in the truth table. LUTRAMs are like regular RAMs, but they are more flexible because they can be configured to implement any

Boolean function. This makes them useful for storing data such as lookup tables and arrays of coefficients that need to be accessed quickly.



A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Output Data
0
1
1
1
1
1
1
1

Figure 5. Lookup tables as a Function Generator.

4.1.3. Flip-flop (FF)

Flip-flops are often used as data storage components or pipeline registers in complex digital systems to increase system speed. They have a clock input for synchronization and are available in different types, including SR, D, and JK flip-flops, which each have their own unique characteristics and can be used in different applications. Flip-flops can be combined in large numbers to provide simultaneous access to stored data. Figure 6 shows a block diagram of FF.

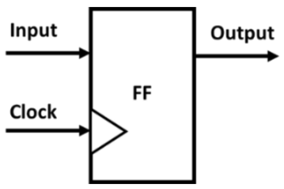


Figure 6. FF’s Block diagram.

5. IMPLEMENTATION

In this work, a comparison between two approaches to designing FPGA-based BiCAMs is presented. The first is FF-based BiCAM and the second is BRAM-based BiCAM. Where they can be used to replace traditional storage elements such as RAM to improve the speed of computing systems’ memory searches. The proposed approaches are designed using the VHDL programming language and implemented at the logic gate level using Altera's Quartus II software. The Cyclone® II 2C70 core FPGA board is selected for implementation because it is the same board used in the implementation of the BZK.SAU.FPGA [25].

5.1. Flip-Flop based BiCAM

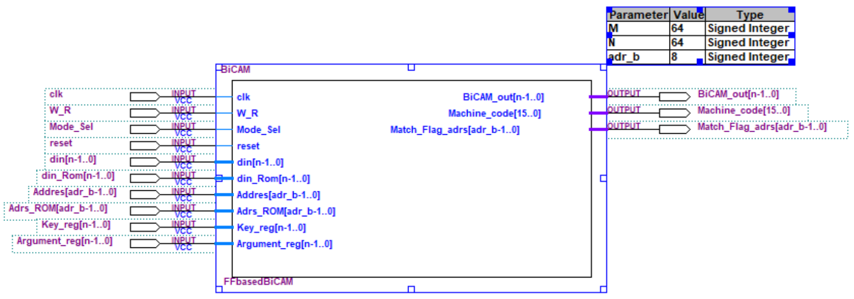
The top module of the proposed design is named as BiCAM that includes multiple submodules, including the Bitcell, Decoder, and Encoder. The Bitcell comprises two main components: a D-Type FF for storing and retrieving data, and a logic-based comparison circuit to check for a match between the stored value in the flip-flop and the search value. The comparison operation of the circuit is performed logically by using OR_gate, AND_gate, and NOT_gate according to Equation 1 that is given below [39].

$$M = A_{reg}F_{out} + \overline{A_{reg}}\,\overline{F_{out}} + \overline{K_{reg}} \qquad (1)$$

Where M is equal to the Output of the comparison circuit, F_out is the output of the Flip-flop, A_reg is the value of the argument register, and K_reg is the value of the key register. The decoder module is used to decode the unique addresses of the BiCAM rows. The Encoder module is used to simplify the address of the match flag register for users. The general block diagram for the proposed approach design is shown in Figure 7. It includes 10 inputs and 3 outputs as following the inputs are the clock (CLK), mode-selection (mode_sel), write or read enable (W_R), reset, 8-bit address, 8-bit Adres ROM,64-bit dinROM,64-bit argument register (Argument_reg), 64-bit key register (Key_reg), and 64-bit data input (din). The outputs are the 64-bit BiCAM output (BiCAM_out), 16-bit machine code (machine_code), and 8-bit address for the match flag (match_flag_adrs).

5.2. BRAM -based BiCAM

The proposed design includes a top module known as Block 1, which comprises several submodules such as a memory module (BRAM), a comparison module, a decoder, an encoder, and a multiplexer. The memory module, which is based on M4k block memory in Cyclone II that is used for data storage and retrieval. The comparison module checks for matches using a logic-based circuit according to Equation 1 mentioned in subsection 4.1.



Parameter	Value	Type
M	64	Signed Integer
N	64	Signed Integer
adr_b	8	Signed Integer

Figure 7. The block diagram of the proposed FF-based BiCAM.

The decoder module decodes unique addresses for BiCAM rows, the encoder simplifies the match flag register address for users, and the multiplexer outputs matched data. The block diagram of the design is illustrated in Figure 8. Where we can observe that the design includes 6 inputs (clock, AddressEnb, 1-bit address, 64-bit argument register, 64-bit key register, and 64-bit data) and 3 outputs (64-bit data_out, 16-bit machine code, and 8-bit matchaddress for the match flag).

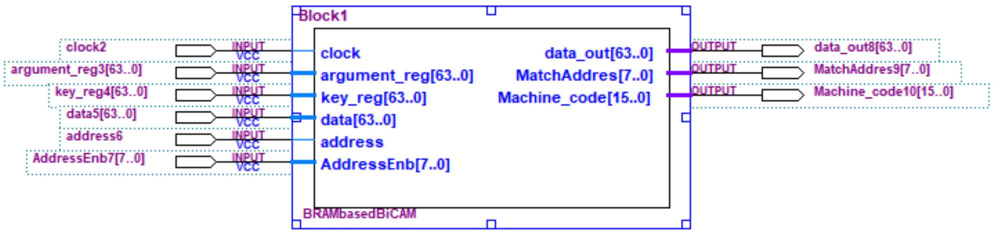


Figure 8. The whole block diagram of the proposed BRAM-based BiCAM.

6. EVALUATION

A bunch of randomly picked assembly codes is used for the implementation of both approaches. After that, they have been converted to their American Standard Code for Information Interchange (ASCII) equivalent in hexadecimal notation, as the same format was used for the BZKSAUFPGA assembler. Then, each of the converted codes was mapped into a 16-bit ROM as the BZKSAUFPGA has a 16-bit architecture. Before it can be used, the random assembly codes that are stored in the 16-bit ROM must be rearranged in the standardized format of the argument register. As the argument register consists of 64 bits, the mapping process begins by taking the most significant bits (MSBs) of the random assembly command and replacing them with the relevant MSBs of the argument register until the address mode character is detected, which indicates the end of the desired portion of the assembly command. The remaining bits of the argument register are set to the "don't care" condition, as they will be ignored in the subsequent search operation. The operand of the assembly command is extracted in the same way that was used for the argument register. The same process was followed when initializing the BiCAM with the assembly instruction sets of the BZKSAUFPGA. For a clearer understanding, Figure 9 provides an example of this process.

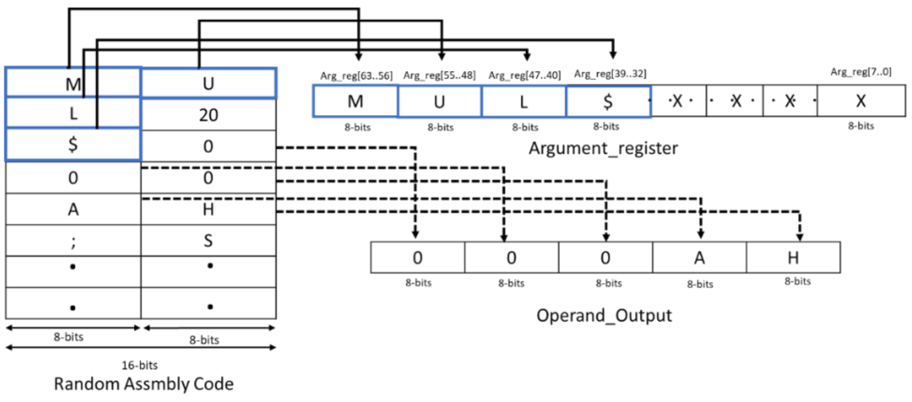


Figure 9. Data preparation of Argument Register.

7. RESULTS AND DISCUSSION

In computer science, it is important to analyze algorithms to determine which one is the most effective for solving a particular problem. While various algorithms can be used to solve the same problem, it can be challenging to determine which one is the most efficient. One common analysis of algorithms is time complexity, which looks at how much time is required to implement an algorithm based on the size of its input [41]. In another words, it evaluates how the running time of an algorithm changes as the size of its input increases. This is usually represented using big O notation, which provides an upper limit on how quickly the running time of an algorithm will grow as the size of its input increases without bound. As expected, after implementing both approaches to design the BiCAM, the time complexity was found to be a constant value of O(1) in all cases. This means that it only takes a single clock cycle to find a match if it exists. To gain a more complete understanding of both proposed approaches, in addition to the time complexity analysis, a source utilization and power efficiency analysis was also conducted. A summary of source utilization is shown in Table 2. below. As we can see, the FF approach utilizes more logical elements, with a threefold increase, and dedicated logic registers, with a 97-fold increase. This is expected as it uses a large number of FFs as storage units and includes decoders and encoders in its design. Additionally, no memory bits are used in the FF-based approach. On the other hand, the BRAM-based approach uses memory bits, as it utilizes the factory-specified onboard memory bits (M4K), as the storage element.

Table 2. Source utilization of both RAM and BiCAM-based methods.

Memory Type	BRAM-based BiCAM	FF-based BiCAM
Total Logic Elements	4,958	14,815
Dedicated Logic Registers	127	12,288
Total Combinational Functions	4,958	8,799
Total Registers	127	12,288
Total Pins	290	364
Total Memory Bits	4,032	0

In Figure 10, a graph of the power efficiency shows us the total thermal power dissipation by both FF and BRAM based approaches. From the graphs, we can notice that the FF-based approach consumes more power by the percentage of 3.6% as it utilizes more logical elements.

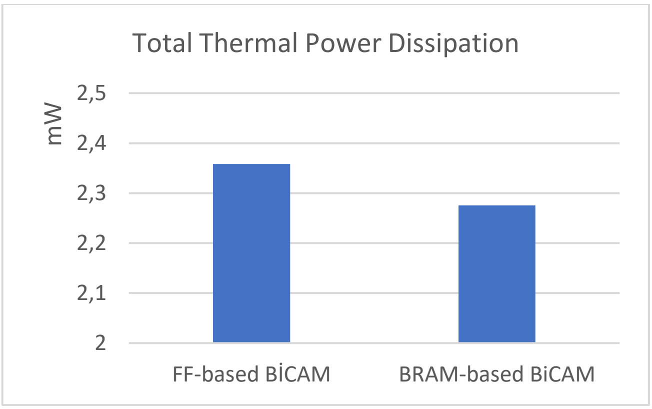


Figure 10. Total thermal power dissipation graph of both BiCAM and RAM.

8. CONCLUSION

Memory is a crucial component of computing systems that can affect the execution time of the system. It is used to store instructions that the CPU or microcontroller needs to execute. To find a desired instruction within memory, a search operation is required. The speed of search operations can be improved using various techniques, however, many of these are software-based and may be limited by their memory's hardware structure. This work presents a comparison of two approaches for designing FPGA-based BiCAM, which is a hardware-based enhancement to the search operation, as an alternative to existing software-based techniques. This study seeks to provide a more comprehensive and detailed understanding of the various types of memory that can be utilized in constructing FPGA-based BiCAMs to be used as a hardware-based enhancement to searching operation in computing systems applications. This can be accomplished by examining the different memory types that are available on the FPGA boards and how to utilize them to develop the BiCAMs. The first approach uses FFs and the second one uses BRAMs as the memory element. The BZK.SAU.FPGA assembler was used as the case study. Both approaches were used to replace the RAM that implements the search operation for the instruction sets of the assembler. They were designed and implemented in an FPGA environment. Both proposed approaches demonstrated excellent efficiency in terms of time complexity, with a constant time of O (1) for the search operation regardless of the number or length of entries. However, the approaches showed different efficiency in terms of resources used and power dissipation due to their different hardware structures. To determine which approach is better, the specific requirements of the application and the resources available on the FPGA board should be considered. To summarize, after implementing both approaches, we found that using FPGA-based BRAM instead of conventional techniques resulted in better search speed despite higher cost. However, the size and cost of both designs increased linearly with the length of the data and the number of entries, particularly for the FF-based approach. As a result, both designs may have scalability issues, which can be a critical factor in applications where larger CAMs are required. Nevertheless, they are suitable for use in applications where large-scale CAMs are not necessary, offering a speed enhancement that can be crucial in time-sensitive applications. Therefore, they can be considered a good trade-off.

Author's Contribution

Abdelkader LAZZEM, Halit ÖZTEKİN, and İhsan PEHLİVAN contributed to the design and implementation of the research, to the analysis of the results, and to the writing, reviewing, and editing of this manuscript.

Conflict of Interest

All authors declare that they have no conflicts of interest.

REFERENCES

[1] R.Schlesinger "Developing Real World Software", Jones and Bartlett Publishers, 2009.

[2] H. Öztekin, F.Temurtas, and A.Gulbag "BZK.SAU. FPGA10.1: A modular approach to FPGA-based microcomputer architecture design for educational purposes", Computer Applications in Engineering Education, vol. 22, no. 2, pp. 272–282, 2014.

[3] R. Karam, R. Puri, S. Ghosh and S. Bhunia "Emerging Trends in Design and Applications of Memory-Based Computing and Content-Addressable Memories", in Proceedings of the IEEE, vol. 103, no. 8, pp. 1311-1330, Aug. 2015,

[4] H. Öztekin "BiCAM-based automated scoring system for digital logic circuit diagrams". Open Chemistry, vol. 20, no. 1, pp. 1548-1556. Dec.2022

[5] S. Hirasawa, H. Yamaki and M. Koibuchi "Packet Forwarding Cache of Commodity Switches for Parallel Computers", 2021 IEEE International Conference on Cluster Computing (CLUSTER), pp. 366-376, Sep.2021,

[6] D. Jothi, and R. Sivakumar "Design and Analysis of Power Efficient Binary Content Addressable Memory (PEBCAM) Core Cells", Circuits, Systems, and Signal Processing, vol. 37, no. 6, pp. 1422–1451, 2018.

[7] M.V. Zackriya , and H.M. Kittur " Precharge-Free Low-Power Content-Addressable Memory", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 8, pp. 2614-2621, 2016.

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

[9] Z. Ullah, K.Ilgon, and S. Baeg " Hybrid Partitioned SRAM-Based Ternary Content Addressable Memory", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, pp. 2969-2979, 2012.

[10] A. Ahmed, K. Park, and S. Baeg "Resource-Efficient SRAM-Based Ternary Content Addressable Memory ", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, pp. 1583-1587, 2017.

[11] J.G. Nash "Distributed-Memory-Based FFT Archite -cture and FPGA Implementations", Electronics, vol. 7, no. 7, pp. 116–145, 2018.

[12] Z. Ullah "LH-CAM: Logic-Based Higher Performan -ce Binary CAM Architecture on FPGA", IEEE Embedded Systems Letters, vol. 9, no. 2, pp. 29–32, 2017.

[13] Z. Ullah, K. Ilgon, and S.Baeg "Hybrid Partitioned SRAM-Based Ternary Content Addressable Memory", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, pp. 2969-2979, 2012.

[14] W. Jiang "Scalable Ternary Content Addressable Memory implementation using FPGAs", 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 71-82. Oct. 2013.

[15] Z. Qian, and M. Margala "Low power RAM-based hierarchical CAM on FPGA", 2014 International Conference on Reconfigurable Computing and FPGAs (ReConFig14), pp. 1-4, Dec. 2014.

[16] Z. Ullah, M. K. Jaiswal, and R. C. Cheung "Z-TCAM: an SRAM based architecture for TCAM", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 2, pp. 402–406, 2015.

[17] I. Ullah, Z. Ullah, and J. Lee "Efficient TCAM Design Based on Multi Pumping-Enabled Multi ported SRAM on FPGA", IEEE Access, vol. 6, pp. 19940-19947, 2018.

[18] M. Irfan, Z. Ullah, and R.C. Cheung "Zi-CAM: A Power and Resource Efficient Binary Content-Addressable Memory on FPGAs", Electronics, vol. 8, no. 5, pp. 584-596, 2019.

[19] P. Reviriego, A. Ullah, and S. Pontarelli "PR-TCAM: Efficient TCAM Emulation on Xilinx FPGAs Using Partial Reconfiguration", IEEE

Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 8, pp. 1952-1956, 2019.

[20] M. Irfan, Z. Ullah, and R.C. Cheung " D-TCAM: A High-Performance Distributed RAM Based TCAM Architecture on FPGAs", IEEE Access, vol. 7, pp. 96060-96069, 2019.

[21] I. Ullah, Z. Ullah, U. Afzaal, and J. Lee "DURE: An Energy- and Resource-Efficient TCAM Architecture for FPGAs With Dynamic Updates", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 6, pp. 1298-1307, 2019.

[22] H. Mahmood, Z. Ullah, O.M. Mujahid, I. Ullah, and A. Hafeez "Beyond the Limits of Typical Strategies: Resources Efficient FPGA-Based TCAM", IEEE Embedded Systems Letters, vol. 11, no. 3, pp. 89-92, 2018.

[23] M. Irfan, and Z. Ullah "G-AETCAM: Gate-Based Area-Efficient Ternary Content-Addressable Memor -y on FPGA", IEEE Access, vol. 5, pp. 20785-20790, 2017.

[24] M. Irfan, H.E. Yantır, Z. Ullah and R.C.C. Cheung "Comp-TCAM: An Adaptable Composite Ternary Content-Addressable Memory on FPGAs", in IEEE Embedded Systems Letters, vol. 14, no. 2, pp. 63-66, 2022.

[25] H. Öztekin, F. Temurtas, and A. Gulbag "BZK. SAU. FPGA10.0: Microprocessor architecture design on reconfigurable hardware as an educational tool", 2011 IEEE Symposium on Computers & Informatics, pp. 385-389, Mar. 2010.

[26] H. Öztekin, H. Kişioğlu, A. Gülbağ, F. Temurtas "The design and implementation of a 16 bit floating point arithmetic unit using BZK.SAU.FPGA microcomputer assembly language", Computer Applications in Engineering Education. vol. 20, no. 6, pp. 1834–1856, 2022.

[27] H. Öztekin, A. Gülbağ, and F. Temurtaş " Assembler Design for BZK.SAU. FPGA Micro Computer Architecture", Electronic Letters on Science and Engineering, vol. 13, no. 1, pp. 1-9, 2017.

[28] H. Öztekin " Embedded Operating System Design on Configurable Modular Hardware for Educational Purposes", Ph.D. Thesis, Sakarya University. Institute of Science and Technology, Sakarya, 2012.

[29] F. Temurtas, and A. Gulbag "EducationalMicrocomputer Architecture and Embedded Operating System Design on Remote Accessible Configurable Hardware", Proj. No. 110E069 , TÜBI TAK-EEEAG, 2012.

[30] A. Boutros, and V. Betz "FPGA Architecture: Principles and Progression", IEEE Circuits and Systems Magazine, vol. 21, no. 2, pp. 4-29, 2021.

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

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

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

[34] Xilinx "Binary CAM Search LogiCORE IP Product Guide (PG317) ", Retrieved from https://docs. xilinx.com/r/en-US/pg317-bcam/Introduction, 2022.

[35] Xilinx "Ternary CAM Search LogiCORE IP Product Guide (PG318) ", Retrieved from https://docs. xilinx.com/r/2.2-English/pg318-tcam/Introduction, 2021.

[36] B. MacCleery, Z. Kassas "New Mechatronics Development Techniques for FPGA-Based Control and Simulation of Electromechanical Systems", IFAC Proceedings, vol. 41, no.2, pp. 4434-4439,2008.

[37] G. Dhanabalan, V. Karutharaja and M. Sakthimohan "Realization of Resource Efficient Block RAM Based Eight Bit Adder in FPGA", 2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), pp. 1-5, April 2019.

[38] M.M. Mano "Computer System Architecture", Prentice Hall, 1993.

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