

Parameterized SDRAM-based content-addressable memory on field programmable gate array

Binh Dang, Minh Bui, Nguyen Trinh Vu Dang, Linh Tran

Department of Electronics, Ho Chi Minh City University of Technology (HCMUT), Vietnam National University-Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Dec 23, 2022

Revised Mar 13, 2023

Accepted Mar 24, 2023

Keywords:

Collision-avoidance

Data-collision CAM

FPGA CAM

Resource-saving

SDRAM-based CAM

ABSTRACT

Contents-addressable memory (CAM) is a special memory that searches the input data with the entire pre-loaded database and generates corresponding address information. CAMs are advancing to be a core technology in computer networking systems. As field programmable gate array (FPGA) is recently being used for network acceleration applications, the demand to integrate CAM on FPGA is increasing. FPGA-based CAMs are divided into three categories of implementation: register-based, block RAM (BRAM)-based, and distributed RAM-based CAM. However, they come with a cost of excessive resource usage. Besides, the collision ratio is high in FPGA-based CAMs, leading to data loss and failure to produce accurate addresses. Synchronous dynamic random-access memory (SDRAM)-based CAMs, benefiting from the features of high density and low price of SDRAM, solve the limitations of FPGA's on-chip resources. This paper proposes a data-collision CAM hardware implementation using modern FPGA's off-chip SDRAM for data storage. The hardware architecture is customized for massive lookup tables and resource-saving. Furthermore, the architecture is parameterized, which is better for integration. The synthesis results and comparisons show significant advancement compared to other FPGA-based CAM implementations by total reduction of on-chip RAM. The novel architecture shows remarkable improvement in the memory depth and width with the capacity of 128 Mbyte lookup table.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Linh Tran

Department of Electronics, Ho Chi Minh City University of Technology (HCMUT)

Vietnam National University-Ho Chi Minh City (VNU-HCM)

Ho Chi Minh City, Vietnam

Email: linhtran@hcmut.edu.vn

1. INTRODUCTION

With the rapid evolution of the Internet, the networking system has become more sophisticated than ever [1]. Cloud-based services expand and are essential in every aspect, such as web services or in enterprises, which is believed to play a vital factor in computer networking. Also, with the rise of IoT, more extensive and complex databases need effective managing and searching methods [2]. Hence, the demand for a high-speed networking system is necessary; hence, a more reliable solution is preferable.

Simple data lookup solutions can no longer be effective as faster searches and larger table sizes are required. A fast content lookup table, i.e., content-addressable memory (CAM), becomes an indispensable part of any high-speed networking system. CAM is a particular memory type that searches the binary data input with the entire pre-loaded database and generates the corresponding address information with a single clock cycle [3]–[5]. The database can be a routing network table that stores the port information to which the data

packet is being forwarded. The CAM takes the address information (“0 1 1 1”) of the incoming packet and looks up for the appropriate port (Port B), as illustrated in Figure 1 [6]. The high-speed feature of CAM is ideal for intensive search time applications [3]. Frequent and fast lookup operations are required in image processing, encryption information, or database accelerators. CAMs are primarily used for network routers for fast transfer, packet forwarding, Ethernet address lookup, or translation-look-aside buffers (TLBs) [3].

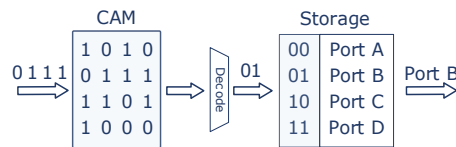


Figure 1. Content-addressable memory

In recent years, field programmable gate array (FPGA) based platforms have increased due to adoption in fields such as security and network processing [7]. Their applications include streaming, data processing, and heavy data flow [8], [9]. FPGA-based hardware accelerators have advanced and been preferred to conventional accelerators in the modern high-performance cloud. Modern FPGAs have integrated static and dynamic storage elements, e.g., SRAM and SDRAM [10], making them ideal for content-addressable advancements. Studies have been proposed with the use of internal memories on FPGA to implement CAM; however, the trade-off for a significant number of on-chip resources [11]–[14]. Improvements in CAM’s performance and resource optimization [15], [16] have been proposed but either required many adaptive logic modules (ALMs) and block RAMs or trade-off for high collision probability to fit a large dataset into them due to the limitation of FPGA’s resources. A DDR-SDRAM packet header lookup circuit, known as the SDRAM-based Hash-CAM [17], was implemented on FPGA. It introduced a method to overcome the limitation of FPGA on-chip resources. Even though block RAMs (BRAMs) were less than on-chip implementations, it still required a substantial number of logics and registers resources to overcome the collision ratio of hashing method. This paper proposes a novel SDRAM-based CAM hardware architecture with advancements compared to conventional CAMs. The architecture saves area costs and reserves on-chip resources such as ALMs and storage elements on FPGA. Furthermore, the architecture reduces collision probability considerably and are parameterized by taking advantage of the high-density feature of SDRAM as well. The rest of this paper is structured as follows. The next section provides information about the preliminaries of FPGA-based CAM research over the years as well as the motivation and contributions of the architecture. Section 3 proposes the hardware system architecture with advancement in collision-avoidance and SDRAM compatibility. Section 4 analyzes the architecture performance and provides synthesis results comparison with related works. Section 5 concludes and discusses the future of large-scale implementation of architecture.

2. PRELIMINARIES AND MOTIVATION

CAMs have become a mature field of research because of their importance on systems. This section provides an overview about the current research state of CAM on FPGA over the years. In addition, we present the inducement of this research.

2.1. Preliminaries

One of the most outstanding examples of CAM application was the STARAN [18], built by Goodyear Aerospace Corporation in 1972. It was a single instruction, multiple data array processor with a 4×256 1-bit processing element computer based on CAM. It’s astounding performance was proved by its image-processing ability [19]. It executed many sophisticated image processing algorithms interactively under ten seconds of response time, and the throughput for batch processing systems experienced a giant leap. Therefore, with its strength, an integrated CAM that consumes low hardware resources but has high reliability and expansion capability is an essential part of state-of-the-art computing systems.

2.1.1 Conventional CAM techniques

The original development of CAM based on application-specific integrated circuits (ASICs) has slowed down [15] because of the low storage density and challenging configurability [6]. Field-programmable gate array (FPGA)-based CAMs are becoming popular due to high-integration density, power consumption, and their application in the state-up-the-art networking system such as SDNs [20]. Researchers tended to

implement CAM using on-chip resources on FPGA. FPGA-based CAM implementation can be categorized into three ways of implementations: register-based, BRAM-based, and distributed RAM-based [4]. The register-based BCAMs could read and compare all the patterns simultaneously due to the flexibility of reading and writing flip-flops [13]. Distributed RAM, i.e., lookup table random access memory (LUTRAM)-based CAM, provided better performance, low power, and resource utilization due to exploiting the paired nature of LUTs and FFs in modern FPGAs [21]. BRAMs, utilized by FPGA vendor IP libraries, paved the way for single-cycle pattern match searching [13]. However, the method and architecture have disadvantages that make it hard to integrate widely in many systems. For example, the drawbacks of BRAM-based CAM are the pre-processing of mapping data, the constraint of data order, and the large SRAM/CAM bit ratio. Furthermore, LUTRAM-based CAM is in trouble with wide bitwise AND-ing and high-routing complexity, which makes them require shallow and wide RAM blocks to implement large-size RAM-based CAMs. Another candidate is FF-based CAM, which improves the resources per CAM bit but needs more scalability and is hard to implement due to routing complexity [4].

2.1.2 SDRAM-based CAM

Since IoT has rapidly grown with larger and more complex databases, FPGA's on-chip resources have become an unsuitable choice for data management due to the limited memory storage, high resource consumption, and challenging integration. To relax these resource-related constraints, CAM research has been shifting to using off-chip memory components on modern FPGAs to save resources and increase storage capacity. Yang *et al.* [17] first proposed a DDR-SDRAM-based data lookup circuit setting a landmark for the deployment of SDRAM-based CAM. It shows that the CAM using external memory is superior in size while maintaining competitive throughout compared to conventional CAM techniques. Furthermore, the SDRAM-based Hash-CAM can easily fit 64 K entrants with 32-bit data, a remarkable number compared to other logic-based CAM. The design, however, comes against an inevitable collision issue using the CRC-16 polynomial hash function; thus, it still required a register-based Hash-CAM circuit to accommodate collided keys. Moreover, with the requirement of a CAM logic cell, the timing performance is restricted by the number of CAM entries and takes up considerable on-chip resources.

2.2. Motivation and contributions

2.2.1. Motivation

CAM is now an indispensable element in any modern networking system. It is a reliable routing table for packet classification and forwarding. Moreover, state-up-the-art FPGA technology is distinct in high-speed and complex networking processing due to its rich resources and reconfigurability. Modern FPGAs have efficiently integrated many hardware resources, including off-chip storage elements such as SDRAM, to support FPGA applications [10]. Furthermore, the innovation of 5G networks requires massive database processing; thus, the implementation and advancement of CAM on FPGA are crucial for any complex system. Table 1 shows CAM's on-chip resource consumption of three representatives for distributed RAM-based, BRAM-based, and Register-based CAM methodologies, respectively. The resource consumption of each method is presented with the size of 16 K lookup entrants for 36-bit data.

Table 1. CAM on-chip resource consumption

	ALMs	BRAMs (20 K)	Speed (MHz)
I2D-BCAM [13]	70,000	400	200
BF-BCAM [12]	80,000	2100	160
Reg-based CAM [13]	260,000	-	160

The register method used up more than 250 K logic cells to accommodate data, a tremendous logics consumption. BRAM-based CAM provides better logic utilization but takes up to two thousand 20 K RAMs. The LUTRAM-based solution helps balance the logic and block RAMs usage. Despite such improvements in balancing resources, it is only possible to integrate such a 16 K×36-bit CAM into small FPGA family devices, e.g., the Intel FPGA Cyclone V 5CSEMA4U23C6 with an average number of 15 K ALMs.

Many algorithms have been proposed to tackle area consumption issues, but resource optimization solutions face another inevitable problem, i.e., the collision phenomenon. Such a phenomenon has been acknowledged in various CAM research [15], [16], [22], [23] as a crucial trade-off issue when intending to make CAM less resource-consuming. Users must choose between the demand of the area and the data loss. The trade-off comes as an even more significant issue when users intend to integrate CAM on small FPGA lineups. In this architecture, we propose a novel SDRAM-based CAM hardware architecture using an external storage element (SDRAM) that efficiently utilizes available on-chip resources on FPGA and increases the depth and width of CAM to catch up with the innovation required of complex systems.

2.2.2. Key contribution

Key contributions of the proposed work are as follows:

- This paper proposed a novel hardware architecture of data-collision CAM by using DDR SDRAM that can store massive databases without losing the integrity of data while consuming modest amounts of resources so that even small FPGA lineups that support DDR SDRAM can integrate large-size CAMs easily.
- The proposed CAM design, SDRAM-based data-collision CAM, shows significant advancements compared to the available FPGA on-chip-based BCAMs in logic utilization on FPGA and shows a remarkable improvement in look-up-table depth and width.
- The architecture shows better SDRAM utilization and improvement in resource utilization by 97.2%, 93.6%, and 100% reduction in logic, registers, and memories, respectively, compared to the related work [17].

3. PROPOSED HARDWARE IMPLEMENTATION

This section proposes a novel SDRAM-based content-addressable memory hardware architecture that is customized for resource-saving and collision-avoidance purposes. The proposed method is presented with the feature of SDRAM-compatibility and collision-avoidance method. The hardware system architecture is described in detail with each functional block.

3.1. Proposed method

Sato *et al.* [16] proposed a method called data-collision, an effective algorithm to compensate resources for constructing a CAM. The algorithm initially used block RAMs to store data addresses (DA) and decimal figures. The decimal figure is divided into fragments with pre-configured bits each; based on the similarity of fragments, DA are stored in the exact memory location. With this approach, area costs are reduced significantly. The method also showed superior performance on high-speed searching with lower power consumption due to the address-to-data search algorithm.

Regarding accuracy, when the number of inputs is the same as the register address number, the collision probability is conducted to 0.26 in a normalized segment, so the collision probability is 0.26 power the number of segments [16]. Hence, they claimed that their algorithm has a negligible probability for all the collisions in an extensive random database. However, the problem arises when we want to increase the size of CAM due to the limitation in the length of address bits of FPGA's on-chip block RAMs. For instance, on a small FPGA family, the maximum depth a single M10 K memory can support is 512×20-bit [15], which mean the CAM's depth is limited to 512 entries to keep a negligible ratio. Thus, when constructing 2 K entries of RAM-based data-collision CAM, the collision probability is now conducted to approximately 0.6 in a normalized segment, i.e., it is over-entry.

3.1.1. Collision phenomenon

Many researchers have proposed effective resource-saving architectures for CAM; however, they come against the collision phenomenon. This phenomenon occurs varying by different implemented algorithms. A graph illustrating the collision probability of each function in write, collision, and empty entries is built in [16]. If more than one DA is stored at the same register address (RA), that location will be marked as collided, which means a false segment when searching. Figure 2 illustrates faulty consequences resulting from the collision phenomenon with the original dataset with DA and RA in the top left. Data address "D" collides with "B" at register address "11"; with "C" at "11"; and so on. After setting all datasets into the storage element, the CAM table is in the bottom left. Thus, when searching for data address "D," CAM-like element cannot produce accurate results due to the collision incident. Collision phenomenon scales with the large ruleset. BRAM-based segment mat requires many memories when intending to reduce collision ratio. Nevertheless, a novel architecture advancement must minimize collided segments [16].

3.1.2. Collision-avoidance method

By utilizing SDRAM, this study proposed a new method to overcome the collision problem for [16], the so-called collision-avoidance method. The method to minimize collided segments is illustrated in Figure 3. Two different fragmentation methods present an example of 10-bit keys with corresponding addresses. First, binary data are cut into fewer fragments to increase the width. The DA are then distributed far from each other, reducing collision probability. However, since the fragment's pre-defined bits are now longer, memory storage capacity increases, which leads to excessive use of block RAMs.

The architecture takes advantage of the excessive address bits of SDRAM. With lengthy address bits, a 512 Mbyte SDRAM supports up to 28-bit address width with 16-bit data width, which helps scale down

many collided segments. As in Figure 3, cutting binary data into two fragments with five bits each, instead of five fragments with 2-bit each, helps reduce collided segments by five times. It shows superior results in reducing collision probability. Thus, when searching for the data address “D,” the exact result is produced for the system to continue its operation.

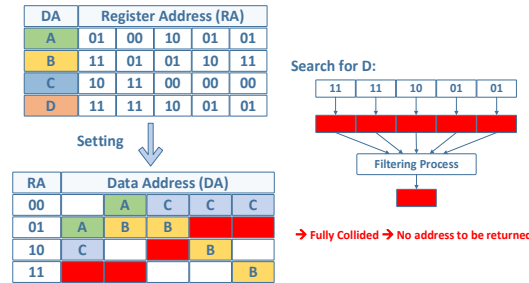


Figure 2. Collision phenomenon

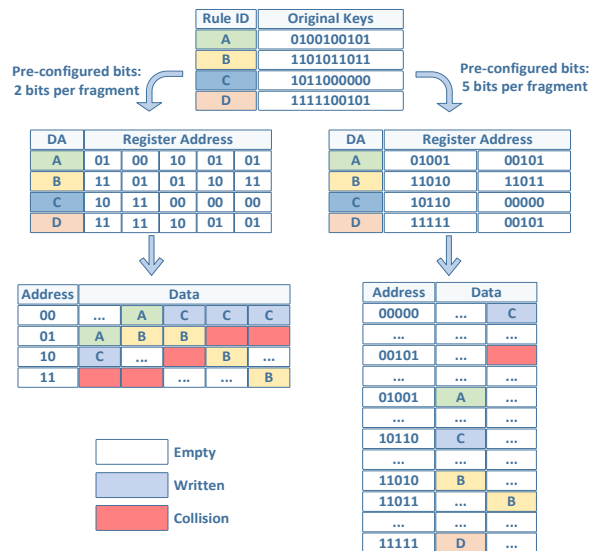


Figure 3. Collision-avoidance method

3.1.3. SDRAM compatibility

Improvements to SDRAM compatibility are required. Figure 4 presents data arrangement with conventional BRAMs and the proposed SDRAM method. Figure 4(a) illustrates the traditional algorithm requires block RAMs connected in parallel to store data (each color indicates a RAM with its addresses). Block RAMs share the same addresses, known as RA, and data width, known as data address (DA). This conventional method is high-speed but very RAM-consuming and unable to be reconfigured.

With SDRAM-based CAM, only one big memory is available as data storage. As a result, complex adjustments are necessary for the SDRAM-based CAM to distribute data as the RAM-based version. An address prefix algorithm is developed; by taking advantage of the lengthy address bit, this functional block helps the CAM see the SDRAM as a group of BRAMs, as illustrated in Figure 4(b). The red color indicates the prefix that partitions SDRAM. A distinct color indicates each region of the memory location. Not only does this approach allow CAM to be implemented using one unified memory, but it also helps the architecture to be reconfigurable. Another significant improvement of SDRAM-based design is the new comparator algorithm that enables the architecture to be parameterized and more resource-saving. The time for comparison is fixed and independent from searching time; in other words, even if it performs plenty of segment searching, the result is always received within fixed cycles after the last search operation. As a result, the SDRAM-based data-collision CAM is implemented using the following parameters: a total fragment number, a key width, and an ID width. The use of parameters allows different size CAMs to be easily synthesized and adapted to different systems.



Figure 4. Database arrangement using (a) conventional SRAMs and (b) the proposed SDRAM method

3.2. Hardware system architecture

The overall architecture of SDRAM-based data-collision CAM is illustrated in Figure 5. The architecture supports updating original keys and their corresponding address information in the memories. As setting keys and their address arrive, the address prefix generator cuts the keys into a user-defined number of fragments and calculates the appropriate address location to store data. Simultaneously, the Segmentation block communicates with DDR SDRAM through the SDRAM Controller block to decide the proper status bits for each address to store them in memories.

When search data arrives, the segmentation block performs a straightforward operation. First, data input is cut into a pre-defined number of fragments with pre-configured bits, which addresses the SDRAM through the SDRAM controller block. A vector of segments, including addresses plus their status, is then read, and presented at Segmentation to prepare for the comparison process. In the comparison phase, the comparator will sequentially take segments from SDRAM and compare them to eliminate empty or collided ones. Then, the comparator passes the result for confirmation with the pre-set original search data. The confirmation logic block outputs the reference address if a match occurs. To compensate for SDRAM read latency, which is known as the time from the read signal is asserted to the read-data-valid signal is asserted and read data is presented at the interface of the controller, a parallel shift register is used to delay the input data; therefore, data is synchronized for confirmation.

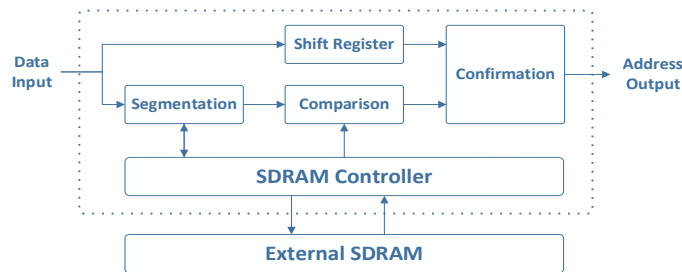


Figure 5. Overall system architecture

3.2.1. Segmentation

The segmentation can be seen as the heart of data-collision CAM. It is responsible for directly communicating with SDRAM controller interface to perform setting or searching segments into or from memory. It consists of a big state machine, known as a segment controller, to control the read and write operation of SDRAM and two data paths: Fragmentation and segment datapath, as in Figure 6.

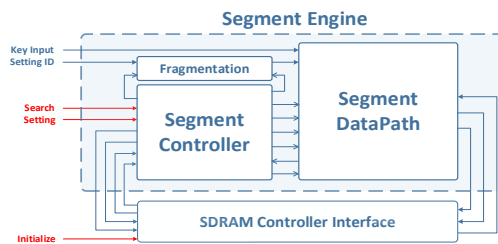


Figure 6. The segment engine architecture

As mentioned above in section 2, a prefix is generated and added to the addresses to mark our segment mat by the address prefix generator (i.e., APG). The APG logic block, located in the fragmentation block, is a parameterized counter that partitions the SDRAM into separated regions. When a key fragment is produced, it is then concatenated with the appropriate auto-generated prefix, as illustrated in Figure 7. The output of the concatenation process is the SDRAM memory address for data arrangement. The inside of the fragmentation is illustrated in Figure 7.

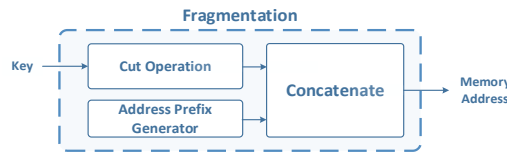


Figure 7. The fragmentation logic

3.2.2. Comparison

An effective comparison algorithm is crucial in CAM-like elements. Since segments are sequentially read from the SDRAM controller, the comparison process needs a new algorithm; otherwise, the whole architecture consumes numerous clock cycles. Figure 8 shows a sequential comparator, which operates concurrently with the segment engine. To optimize supported speed, the comparator is pipelined with three stages. It produces an address for final confirmation process within 3 clock cycles after all segments are read from SDRAM. The new comparator is a key factor that makes SDRAM-based CAM parameterized.

3.2.3. Confirmation

Confirmation process is the last stage of CAM which compares the final address with its original database to increase the reliability of the output. Figure 9 shows the implementation of confirmation process to ensure the accurate output. The final segment for confirmation contains information of the final address, its validity status, and corresponding original key data. It is extracted to obtain the original key, and the module performs a comparison with the search key making sure the final address is correct.

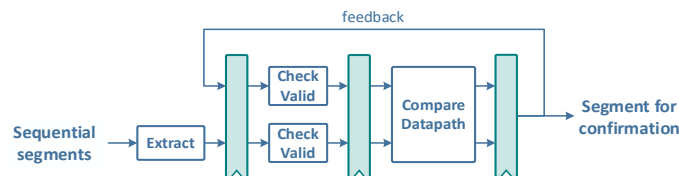


Figure 8. Sequential comparator algorithm



Figure 9. The confirmation process

3.2.4. SDRAM controller interface

Figure 10 shows how external DDR-SDRAM communication has been simplified using a built-in address span extender intel FPGA IP. The IP provided an Avalon-MM master-slave interface in which the master side controls the operation of SDRAM while the slave side connects to FPGA’s portion. The SDRAM controller interface is designed with an avalon memory-mapped master interface [24] to establish interconnection with the address span extender command interface. Prior to proceeding with setting or searching, all contents need to be initialized. This means all segments in the table are set to empty status and ready for setting. Figure 11 shows the data-path in two different processes: Setting and searching. Figure 11(a) illustrates the setting process of the architecture. When data input arrives at the segment engine, it is cut into a pre-defined number of fragments based on pre-configured bits. Each of these is a memory address for the data address to situate.

The process of setting a data address is described as follows. First, the fragment is fed into the APG logic block to be marked appropriately as the exact register address. Then based on this, DDR SDRAM accesses that location to check whether it is empty, written, or collided to set the proper status for the new data address. Since the necessary status is generated, the data address is then written to the location. Figure 11(b) illustrates how the proposed CAM architecture performs searching rule ID. Searching has the key cut in the same way as in the setting process; however, the data from DDR-SDRAM is now redirected and sequentially fed into the comparison stage.

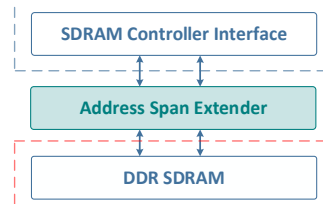


Figure 10. DDR SDRAM communication

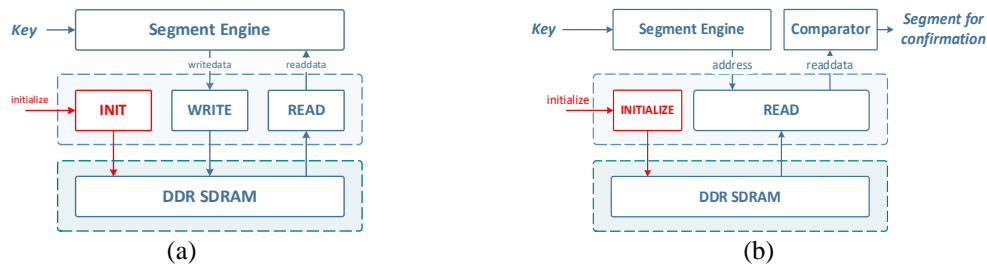


Figure 11. How the architecture processes data in (a) setting and (b) searching mode

4. SYNTHESIS AND ANALYSIS

This section implements a feasible data-collision CAM design with Avalon-MM mapped interface with SDRAM on Altera's DE0-Nano-SoC device (5CSEMA4U23C6)-Cyclone V family [25]. The architecture uses hard processor system's ISSI DDR3 SDRAM memory models for functional verification [26]. Synthesis results and comparisons with various CAM versions are presented.

4.1. FPGA implementation

The entry ratio is derived from [16]. The greater this ratio gets, the higher the collision probability is. If the entry ratio equals one, the collision ratio equals 0.26, which provides negligible collision probability, as mentioned above. Thus, it is recommended to keep the ratio less than one; otherwise, it is over-entry. The entry ratio is expressed in (1).

$$\text{Entry Ratio} = \frac{\text{Number of key entries}}{\text{Number of registers}} \quad (1)$$

Parameterization advancement enables the architecture to be easily created in assorted sizes. The proposed SDRAM-based CAM is parameterized based on a total number of fragments (n_{frag}), a key width (W_k), and an ID width (W_{ID}); thus, the SDRAM depth-width configurations can be calculated using (2)-(4) and selected referring to Table 2.

$$U_{SDRAM} = 2^{AW_{DDR\ SDRAM}} \times DW_{DDR\ SDRAM} \quad (2)$$

$$AW_{DDR\ SDRAM} = \left\lceil \frac{W_k}{n_{frag}} \right\rceil + \lceil \log_2(n_{frag}) \rceil \quad (3)$$

$$DW_{DDR\ SDRAM} = 2 + W_{ID} + W_k \quad (4)$$

Where U_{SDRAM} denotes the SDRAM usage in bits; $AW_{DDR\ SDRAM}$ is the address width, and $DW_{DDR\ SDRAM}$ is the data width of DDR SDRAM. The data width of SDRAM is rounded up to the nearest power-of-two value, e.g., if the total data width is 54-bit, then the selected $DW_{DDR\ SDRAM}$ is 64-bit.

Table 2. SDRAM's depth and width configurations

SDRAM's property	Width (bits)						
$AW_{DDR\ SDRAM}$	28	27	26	25	24	23	22
$DW_{DDR\ SDRAM}$	16	32	64	128	256	512	1024

to evaluate the feasibility of the proposed architecture, multiple configurations of SDRAM-based data-collision CAMs are successfully implemented on the selected FPGA device. Figure 12 presents the synthesis results, SDRAM usage, and collision probability of different CAM sizes. Figures 12(a) and 12(b) shows the FPGA's on-chip logic and registers resource utilization respectively, Figure 12(c) illustrates SDRAM size for each configuration of CAM table, and their corresponding entry ratio in Figure 12(d). The architecture shows significant advancements in the number of entries whilst maintaining a negligible collision ratio and low resource consumption. The architecture supports a capacity of 256-thousand-entry with 144-bit data while consuming less than 1% FPGA on-chip resources and 25% of SDRAM capacity in the selected device. Furthermore, the listed architectures size can operate at 400 MHz.

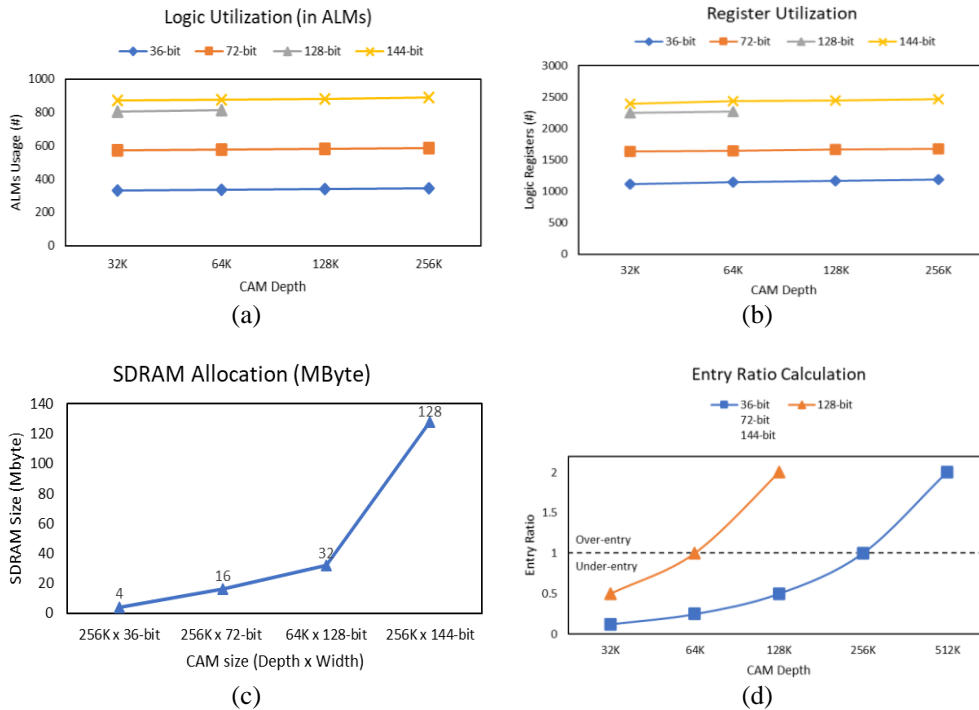


Figure 12. Resource utilization of different configurations in (a) logic (in ALMs), (b) logic registers, (c) SDRAM allocation, and (d) entry ratio of the architecture

As mentioned in section 3, there exist delays for DDR SDRAM to perform proper read and write, which are represented as t_{rd} and t_{wr} , respectively. The following condition (5) needs satisfaction to perform in an appropriate setting. For each search request, the following clock cycles condition (6) must be a guarantee for the proposed CAM to produce the appropriate data address.

$$t_{CAM,setting} = n_{frag} \times (t_{rd} + t_{wr} + t_{seg}) \tag{5}$$

$$t_{CAM,search} = n_{frag} \times (t_{rd} + t_{seg}) + t_{compare} + t_{confirm} \tag{6}$$

Where n_{frag} is the number of fragments that the user defines; $t_{CAM, setting}$ and $t_{CAM, search}$ are clock periods that CAM architecture perform setting or searching respectively; in this design, $t_{compare} = 3$ and $t_{confirm} = 2$ are cycles it takes for address comparison and confirmation. Another critical parameter is t_{seg} , which is the clock period it takes segment engine to perform its operation. For a setting request, $t_{seg} = 6$ cycles for fully extracting a single fragment from SDRAM; while for a search request, for $t_{seg} = 8$ cycles for thoroughly arranging a single fragment into SDRAM as it needs extra cycles to modify data before writing it to the database. The time to perform a read from or write to SDRAM is defined as the sum of the controller latency, command latency, and the CAS latency of the DDR memory. The total latency is simulated as $t_{rd} = t_{wr} = 11$ cycles. In the 64 K key entries CAM with 36-bit each, the matched address presents at the output every 39 clock cycles, and a key is set correctly into the SDRAM every 60 clock cycles.

4.2. Comparison with other CAM architectures

Table 3 compares the novel CAM model with conventional logic-based CAM [14], synthesized with the same 512×36-bit size. The architecture is configured with two fragments which keep the collision probability less than 1%. In addition, the proposed architecture efficiently balances on-chip lookup tables (LUTs) and registers. Table 4 shows a comparison with [12], [13]. The selected configuration is 16 K-depth with 36-bit data width. By effectively utilizing the SDRAM, the proposed model outperforms other conventional CAM architectures by total reduction in BRAMs and significant savings on-chip ALMs and speed compared to the most advanced architecture of all.

Table 5 compares the synthesis results of two SDRAM-based CAM architectures. The selected configuration for both models is 64 K key entries with 32-bit each. The logic utilization of the SDRAM-based data-collision circuit is less than 1% in the selected device with optimal configuration. The data-collision CAM shows better SDRAM utilization compared to the Hash-CAM. The results have proved that the proposed design is superior in reserving logic resources by 97.2% in adaptive LUTs, 93.6% in logic registers, and 100% in memory bits.

Table 3. Logic utilization comparison with on-chip resources-based CAM

CAMs	LUTs	Registers	Memory bits	SDRAM usage	Speed (MHz)
Proposed CAM	287	1,025	0	4 Mbyte	400
LH-CAM [14]	1,259	70	0	—	340

Table 4. Logic utilization comparison with large-size on-chip resources-based CAM

CAMs	ALMs	BRAMs (20 K)	SDRAM usage	Speed (MHz)
Proposed CAM	329	0	4 Mbyte	400
I2D-BCAM [13]	70,000	400	—	200
BF-BCAM [12]	80,000	2,100	—	160
Reg-based CAM [13]	260,000	—	—	160

Table 5. Logic utilization comparison with SDRAM-based Hash-CAM

	SDRAM-based data-collision CAM	SDRAM-based hash-CAM [17]
ALUTs	296	11,221
Logic Registers	1104	17,312
Memory bits	0	21,888
PLLs/DLLs	1/1	1/1
SDRAM utilization	4 MByte	256 KByte

SDRAM-based data-collision CAM is distinguished by its storage capacity that can fit up to 256-thousands-entrants, making it suitable for very-large-size routing table. Meanwhile, the architecture keeps the negligibility of collision probability to ensure the correctness of the outputs. Also, parameterization feature helps the architecture easily synthesized to adapt any system's needs.

5. CONCLUSION AND FUTURE WORK

A novel CAM architecture using SDRAM with a data-collision algorithm has successfully operated in hard processor system, proving the concept of SDRAM-based CAM. It has immense application and development potential. SDRAM-based data-collision CAM hardware architecture can fit a large-sized lookup table while keeping the collision ratio under entry. The proposed architecture is superior in saving on-chip

resources on FPGA; thus, it can easily fit any FPGA device that supports DDR SDRAM and still has enormous logic and storage elements for other FPGA purposes. Furthermore, the architecture is parameterized and advances in reducing collision ratio, which make it great for highly critical system accuracy. However, different parameter configuration affects the performance of CAM. Therefore, it is up to the user to consider the system's demand by using provided parameters, such as size, speed, and collision probability to select the appropriate CAM's parameters. Future upgrades focus on improving search time and throughput of the architecture. Double-data rate (DDR) interface is well known for its high speed and throughput. An advanced CAM-specific DDR controller logic is believed to achieve reduction of read latency of the system. Moreover, ternary content-addressable memory, i.e., TCAM is more versatile. Advancement in supporting ternary value "X" of the architecture is under investigation.

ACKNOWLEDGEMENTS

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.




REFERENCES

- [1] L. Liu, "Discussion and practice of computer network information and network security protection strategy," in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, Dec. 2020, pp. 1810–1813. doi: 10.1109/ICMCCE51767.2020.00397.
- [2] H. Qian, "The application of big data technology in computer network information management," in *2021 2nd International Conference on Big Data Economy and Information Management (BDEIM)*, Dec. 2021, pp. 379–382. doi: 10.1109/BDEIM55082.2021.00083.
- [3] A. Swedha and D. Naveen, "Survey of content addressable memory," *VDE*, p. 1516, Feb. 2018.
- [4] M. Irfan, A. I. Sanka, Z. Ullah, and R. C. C. Cheung, "Reconfigurable content-addressable memory (CAM) on FPGAs: A tutorial and survey," *Future Gener. Comput. Syst.*, vol. 128, pp. 451–465, Mar. 2022, doi: 10.1016/j.future.2021.09.037.
- [5] S. J. E. Wilton, C. W. Jones, and J. Lamoureux, "An embedded flexible content-addressable memory core for inclusion in a Field-Programmable Gate Array," in *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2004, p. II–885, doi: 10.1109/ISCAS.2004.1329414.
- [6] M. Irfan, Z. Ullah, and R. C. C. Cheung, "D-TCAM: A high-performance distributed RAM Based TCAM architecture on FPGAs," *IEEE Access*, vol. 7, pp. 96060–96069, 2019, doi: 10.1109/ACCESS.2019.2927108.
- [7] A. Ullah, P. Reviriego, A. Sánchez-Macián, and J. A. Maestro, "Multiple cells upset injection in BRAMs for Xilinx FPGAs," *IEEE Trans. Device Mater. Reliab.*, vol. 18, no. 4, pp. 636–638, Dec. 2018, doi: 10.1109/TDMR.2018.2878806.
- [8] P. H. W. Leong, "Recent Trends in FPGA Architectures and Applications," in *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, Jan. 2008, pp. 137–141. doi: 10.1109/DELTA.2008.14.
- [9] E. Sotiriades, C. Kozanitis, and A. Dollas, "FPGA based architecture for DNA sequence comparison and database search," in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, Apr. 2006, p. 8. doi: 10.1109/IPDPS.2006.1639443.
- [10] H. Nikolov, T. Stefanov, and E. Deprettere, "Efficient External Memory Interface for Multi-Processor Platforms Realized on FPGA Chips," in *2007 International Conference on Field Programmable Logic and Applications*, Aug. 2007, pp. 580–584. doi: 10.1109/FPL.2007.4380721.
- [11] C. W. Jones and S. J. E. Wilton, "Content-addressable memory with cascaded match, read and write logic in a programmable logic device," *US6622204B1*, Sep. 16, 2003, Accessed: Aug. 19, 2022, <https://patents.google.com/patent/US6622204/en>
- [12] A. M. S. Abdelhadi and G. G. F. Lemieux, "Deep and narrow binary content-addressable memories using FPGA-based BRAMs," in *2014 International Conference on Field-Programmable Technology (FPT)*, Dec. 2014, pp. 318–321. doi: 10.1109/FPT.2014.7082808.
- [13] A. M. S. Abdelhadi and G. G. F. Lemieux, "Modular SRAM-Based Binary Content-Addressable Memories," in *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, Vancouver, BC, Canada: IEEE, May 2015, pp. 207–214. doi: 10.1109/FCCM.2015.69.
- [14] Z. Ullah, "LH-CAM: Logic-Based Higher Performance Binary CAM Architecture on FPGA," *IEEE Embed. Syst. Lett.*, vol. 9, no. 2, pp. 29–32, Jun. 2017, doi: 10.1109/LES.2017.2664378.
- [15] N. Trinh, A. L. Thi Kim, H. Nguyen, and L. Tran, "Algorithmic TCAM on FPGA with data collision approach," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 22, no. 1, p. 89, Apr. 2021, doi: 10.11591/ijeecs.v22.i1.pp89-96.
- [16] Y. Sato, K. Otsuka, K. Kobayashi, T. Kouchi, M. Uwai, and M. Nishizawa, "Novel approach for search engine," in *2016 11th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, Taipei, Taiwan: IEEE, Oct. 2016, pp. 6–72. doi: 10.1109/IMPACT.2016.7799977.
- [17] X. Yang, J. Mu, S. Sezer, J. McCanny, and E. Swartzlander, "High performance IP lookup circuit using DDR SDRAM," in *2008 IEEE International SOC Conference*, Newport Beach, CA, USA: IEEE, Sep. 2008, pp. 371–374. doi: 10.1109/SOCC.2008.4641547.
- [18] E. W. Davis, "STARAN parallel processor system software," in *Proceedings of the May 6-10, 1974, national computer conference and exposition on - AFIPS '74*, Chicago, Illinois: ACM Press, 1974, p. 17, doi: 10.1145/1500175.1500179.
- [19] D. Rohrbacher and J. L. Potter, "Image Processing with the Staran Parallel Computer," *Computer*, vol. 10, no. 8, pp. 54–59, Aug. 1977, doi: 10.1109/C-M.1977.217824.
- [20] R. Karam, R. Puri, S. Ghosh, and S. Bhunia, "Emerging Trends in Design and Applications of Memory-Based Computing and Content-Addressable Memories," *Proc. IEEE*, vol. 103, no. 8, pp. 1311–1330, Aug. 2015, doi: 10.1109/JPROC.2015.2434888.
- [21] Z. Qian and M. Margala, "Low power RAM-based hierarchical CAM on FPGA," in *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, Cancun, Mexico: IEEE, Dec. 2014, pp. 1–4, doi: 10.1109/ReConFig.2014.7032536.




- [22] I. Sourdis, D. Pnevmatikatos, S. Wong, and S. Vassiliadis, "A reconfigurable perfect-hashing scheme for packet inspection," in *International Conference on Field Programmable Logic and Applications, 2005.*, Aug. 2005, pp. 644–647, doi: 10.1109/FPL.2005.1515804.
- [23] G. Papadopoulos and D. Pnevmatikatos, "Hashing + memory = low cost, exact pattern matching," in *International Conference on Field Programmable Logic and Applications, 2005.*, Aug. 2005, pp. 39–44. doi: 10.1109/FPL.2005.1515696.
- [24] *Avalon® Memory-Mapped Interfaces*. Mar. 12, 2023. [Online]. Available: <https://www.intel.com/content/www/us/en/docs/programmable/683091/20-1/memory-mapped-interfaces.html>.
- [25] *Cyclone® V Hard Processor System Technical Reference Manual*. Accessed: Mar. 12, 2023. [Online]. Available: <https://www.intel.com/content/www/us/en/docs/programmable/683126/21-2/hard-processor-system-technical-reference.html>.
- [26] Overview of the Design Guidelines for Cyclone® V SoC FPGAs and Arria® V SoC FPGAs. Mar. 12, 2023. [Online]. Available: <https://www.intel.com/content/www/us/en/docs/programmable/683360/18-0/overview-of-the-design-guidelines-for.html>.

BIOGRAPHIES OF AUTHORS






Binh Dang    received his B.S. in Electrical and Electronics Engineering from Ho Chi Minh City University of Technology (HCMUT), Vietnam (2023). Currently, he is working as a Digital IC Design Engineer in Optical and Copper Connectivity Business Unit, at Marvell Technology Vietnam. His research interests include Ethernet PHY, computer architecture, algorithm development, and hardware design targeting FPGAs. He can be contacted at email: binh.dang.eboy@hcmut.edu.vn.






Minh Bui    received a Bachelor of Engineering degree in Electrical-Electronics Engineering with classification Good from Ho Chi Minh City University of Technology. At the beginning of his third-year, he was a Teaching Assistant of Intro to Computer Engineering, specializing in Embedded C and LC3. From 2020 to 2021, he worked as a Product Engineering Intern of Intel Corporation, in charge of rejected validation Thunderbolt and CPU products. After finished his internship at Intel, he joined Ampere Computing as Design Verification Engineering. In October 2022, he took part in Master of Micro and Nanotechnology at TU Ilmenau. He can be contacted at email: minh.buiminh5052vn@hcmut.edu.vn.



Nguyen Trinh Vu Dang    received the B.S. and M.S. degree in Electronics Engineering from Ho Chi Minh City University of Technology (HCMUT), Vietnam (2019, 2021). He is also working as lecturer at Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology-VNU HCM. He can be contacted at email: nguyentvd@hcmut.edu.vn



Linh H. Tran    received the B.S. degree in Electrical and Computer Engineering from University of Illinois, Urbana-Champaign (2005), M.S. and Ph.D. in Computer Engineering from Portland State University (2006, 2015). Currently, he is working as lecturer at Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology-VNU HCM. His research interests include quantum/reversible logic synthesis, computer architecture, hardware-software co-design, efficient algorithms and hardware design targeting FPGAs and deep learning. He can be contacted at email: linhtran@hcmut.edu.vn.