Original scientific paper
https://doi.org/10.33180/InfMIDEM2022.304

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 52, No. 3(2022), 181 – 189

Energy Efficient and Low dynamic power Consumption TCAM on FPGA

Sridhar Raj Sankara Vadivel, Shantha Selvakumari Ramapackiam

Department of ECE, Mepco Schlenk Engineering College, Sivakasi, Tamilnadu, India

Abstract: Ternary Content Addressable Memories [TCAM] based on Field Programmable Gate Arrays [FPGA] are widely used in artificial intelligence [AI] and networking applications. TCAM macros are unavailable within the FPGA; therefore, they must be emulated using SRAM-based memories, which require FPGA resources. Compared to state-of-the-art designs, the proposed FPGA-based TCAM implementation will save significant resources. This methodology makes use of the Lookup Table RAMS (LUTRAMs), slice carry-chains, and flip-flops (FF) allowing simultaneous mapping of rules and deeper pipelining respectively. The TCAM implementation results in lower power consumption, fewer delays and lower resource utilization. It outperforms conventional FPGA-based TCAMs in terms of energy efficiency (EE) and performance per area (PA) by at least 3.34 and 8.4 times respectively, and 56% better than existing FPGA designs. The proposed method outperforms all previous approaches due to its low dynamic power consumption when considering the huge size of TCAM emulation on SRAM-based FPGAs.

Keywords: TCAM; Software Defined Networking; Artificial intelligence; Networking; Quality of Service (QoS)

Energetsko učinkovit TCAM na FPGA z nizko dinamično porabo energije

Izvleček: Ternarni vsebinsko naslovljivi pomnilniki [TCAM], ki temeljijo na poljskih programirljivih matrikah [FPGA], se pogosto uporabljajo v aplikacijah umetne inteligence [AI] in omrežnih aplikacijah. Makroji TCAM niso na voljo v FPGA, zato jih je treba emulirati s pomnilniki na osnovi SRAM, ki zahtevajo vire FPGA. V primerjavi z najsodobnejšimi zasnovami bo predlagana implementacija TCAM na osnovi FPGA prihranila precej virov. Ta metodologija uporablja pomnilnike RAM s preglednicami za iskanje (LUTRAM), prenosne verige in flip-flope (FF), ki omogočajo hkratno preslikavo pravil oziroma poglobljeno vodenje. Izvedba TCAM omogoča manjšo porabo energije, manjše zakasnitve in manjšo izkoriščenost virov. V smislu energetske učinkovitosti (EE) in zmogljivosti na površino (PA) presega običajne TCAM-e na osnovi FPGA za vsaj 3,34- oziroma 8,4-krat in je za 56 % boljša od obstoječih zasnov FPGA. Predlagana metoda presega vse prejšnje pristope zaradi nizke dinamične porabe energije ob upoštevanju ogromne velikosti emulacije TCAM na FPGA na osnovi SRAM

Ključne besede: TCAM; programsko definirano omrežje; umetna inteligenca; omrežje; kakovost storitev (QoS)

** Corresponding Author's e-mail: sridhars@mepcoeng.ac.in, rshantha@mepcoeng.ac.in*

1 Introduction

Artificial intelligence (AI) is speeding up and becoming more accurate and reliable. The centralized server is used to connect applications from the edge to the cloud. Due to the rapid growth of internet-connected devices and an increase in internet traffic, today's systems require very fast searches. For IP routing and Internet Protocol (IP) forwarding, routers are key components of networking equipment. Routers receive a packet of data and decide where to route it. They must provide fast packet routing by searching through large amounts of data. High-speed searches are also required in CPUs, database engines, and neural networks.

The latest Xilinx and Intel FPGA chips are increasingly being used as data plane accelerators for Software Defined Networking (SDN) [1]. The FPGA industry continually launches software development toolkits to process and classify packets quickly and efficiently [2]. Ethernet/IP forwarding, firewalls, and QoS (Quality of Service) require packet processing and classification. There are three types of matching techniques used in

How to cite:
S. R. S. Vadivel et al., "Energy Efficient and Low dynamic power Consumption TCAM on FPGA", Inf. Midem-J. Microelectron. Electron. Compon. Mater., Vol. 52, No. 3(2022), pp. 181–189

classification. They are Longest Prefix Matching (LPM) [3], Matching with Wildcards [4], and Exact Matching (EM) [5]. Matching with wildcards is the most challenging task.

Switching, Routing, QoS tables and Access Control List (ACL) are all stored in a high-speed memory to allow for forwarding decisions and limits. These memories (lookups) contain information about results, such as whether a packet with a particular destination IP address should be dropped according to an ACL. Cisco Catalyst switches use specialized memory architectures, called CAMs and TCAMs, to store these memory tables.

2 Related works

Content Addressable Memories (CAM) [6] deal only with the binary digits (0's and 1's), whereas the Ternary Content Addressable Memories (TCAM) deal with (0's, 1's, and x), where "x" represents Don't care. TCAMs are not available inside the FPGAs as they must be emulated using memory and logic resources and this leads to a significant resource overhead. Researchers have consequently been working on reducing the resource consumption of FPGA-based TCAMs. TCAMs are made up of three basic parts: storage memory, a priority encoder, and match logic. A major cost component of FPGAs based on SRAM is their storage memories, which comprise the actual TCAM contents to be searched. Bosshart et al. [7] optimize the storage memory needs of TCAMs by combining dual-output LUTs and partial reconfiguration. This saves many storage memory resources.

Match logic generates a flag for each incoming key, and this consumes a significant amount of resources because it must be done simultaneously for all memory locations at high speed. Ullah et al. [8] propose a novel idea for efficiently mapping the matching logic in Xilinx FPGAs by exploiting the built-in carry-chain resources. As the size of the key and rules to be stored in the TCAM increases, the storage and matching logic requirements increase as well. As a result, it is worth looking into optimizing both storage and matching logic resources at the same time.

TCAM emulation on SRAM-based FPGAs has been examined using four types of resources: block RAMs (BRAMs), LUT RAMs (LUTRAMs), lookup tables (LUTs), and flip-flops (FFs). Slice FFs is used as TCAM storage memory in FF-based TCAMs [9] – [12]. Because each FF holds a single bit of data and the architectural limitations require the use of a LUT–FF pair, many of the LUTs will be used as pass-through, wasting resources. Researchers [13], [14], and [15–21] have extensively investigated BRAM-based TCAM emulation using SRAM-based FPGA. However, the efficient use of BRAM for TCAM emulation is restricted by theoretical limits, which require at least an SRAM/TCAM bit ratio of $2^9/9$ and, when contrasted to LUTRAM or LUTs based TCAMs, which requires $2^6/6$ [22], or 5 × more.

Reviriego et al. [23] used 5 × 2 LUTs to emulate TCAMs as well as modern SRAM-based FPGAs with their reconfiguration capabilities for storing and updating TCAM rules. Compared to PR-TCAM [23], BPR-TCAM [8] uses a slice built-in carry-chain to reduce matching logic in TCAMs. Both approaches rely on partial reconfiguration for updating TCAM stored rules. Another resource for TCAM-emulation, in addition to LUTRAM, is distributed RAM [22], [24], [25]. Ullah et al. [21] used distributed RAMs in a 6 × 1 configuration for resource allocation in the same slice to obtain greater performance per area (PA), in addition to using carry chains for the match-logic reduction.

The D-TCAM [26] structure uses LUTRAMs on a 6 × 1 Xilinx template to store TCAMs and pipeline fine-grain by using its built-in slice register to gain higher throughput (TP). The previous work using the LUTRAMS in the 5 x 2 configuration and all of the FFs in the SLICEM were used to improve the throughput and performance per area.

To implement broader TCAM words, the partial match results must be transferred from the current slice's carry chain to the next slice's carry chain [24]. By utilizing the TCAM ANDing logic in the carry-chain, it is possible to achieve the desired TCAM bit density while saving a significant amount of LUT resources. It increases area performance by at least 67 percent and energy efficiency by at least 2.5 times. Frac-TCAM [27] utilizes RAM32M to construct the 8 × 5 TCAM compared to 4 × 6 TCAM used in DURE, thus almost doubling the utilization density. Moreover, LUTRAM outputs can be pipelined via in-slice registers. In comparison to existing approaches, logic utilization and TP can be enhanced, resulting in improved PAs.

By combining BRAM and LUTRAM, Comp-TCAM [28] can implement the TCAM architecture regardless of the type of memory and can be adapted to meet the system requirements. A decrease of 41.6% in hardware resource utilisation has no effect on the functionality.

In this paper, a TCAM emulation on Xilinx SRAM-based FPGAs to achieve a storage reduction in LUTRAMs and a match reduction in logic resources is presented. To accelerate the arithmetic operations, the match bits from

the distributed RAMs are efficiently AND-cascaded using the FPGA's built-in carry chains. Ullah et al. [8] used only one built-in carry-logic for matching one of the rules. The proposed work used only one built-in carry logic for matching two of the rules (i.e)., dual-output LUTs are connected to the two built-in carry logic compared to LH-CAM [10]. It is capable of mapping a single output LUTRAM matching logic. This makes the delay time shorter and the design clock rate faster because it doesn't use any logic or routing resources.

The main contributions of the paper are listed below:

1. An FPGA resource-saving TCAM emulation scheme has been proposed that significantly reduces the resources needed to emulate an individual TCAM.
2. The mapping of two rules using dual-output LUTs and then using the built-in carry-chain to implement the match logic. Thus, additional logic or routing resources are not required for the matching logic. This reduces the delay time and achieves a high clock speed.
3. TCAM is designed to be scalable in terms of lookup rate, power consumption, device utilization, and energy-efficiency.

3 Proposed TCAM architecture

Consider the TCAM emulation on SRAM-based FPGAs. For example, N = 4 and W = 4, i.e., a 4 × 4 TCAM, where W denotes the key size or width, and depth is denoted by N. The key size is 4, and each 4 × 1 SRAM has two input address lines. The TCAM can be divided into two blocks, as shown in Fig.1 and each of the four rules r0, r1, r2, and r3 is mapped to a 4 × 1 SRAM. The top block is indexed by b0 and b1, whereas the bottom block is indexed by b2 and b3. The outputs of the SRAM are combined using AND gates known as match logic. The choice of SRAM implementation primitives, as well as its width and depth extension, is essential to the efficient TCAM designs on FPGA.

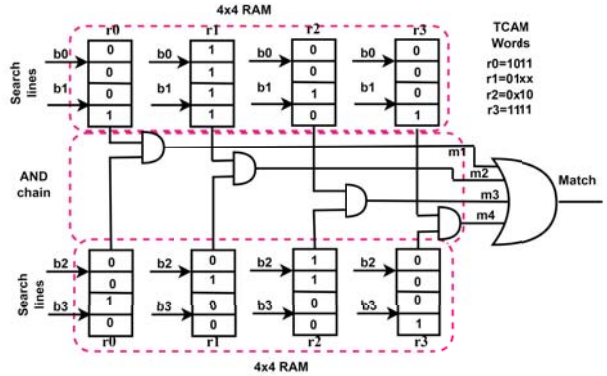

Figure 1: TCAM emulation using SRAMs.

The proposed TCAM makes use of the distributed LUTRAM and carry-chain logic present in the SLICEM of Xilinx FPGAs. Consider a 2 × 5 LUT and carry chain as shown in Fig. 2, which has a key width of five and two rules, i.e., O5 and O6.

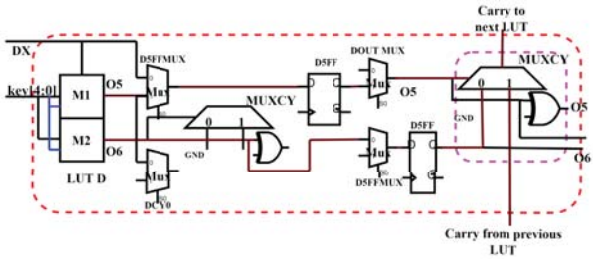

Figure 2: An architecture for mapping a LUT to a carry chain

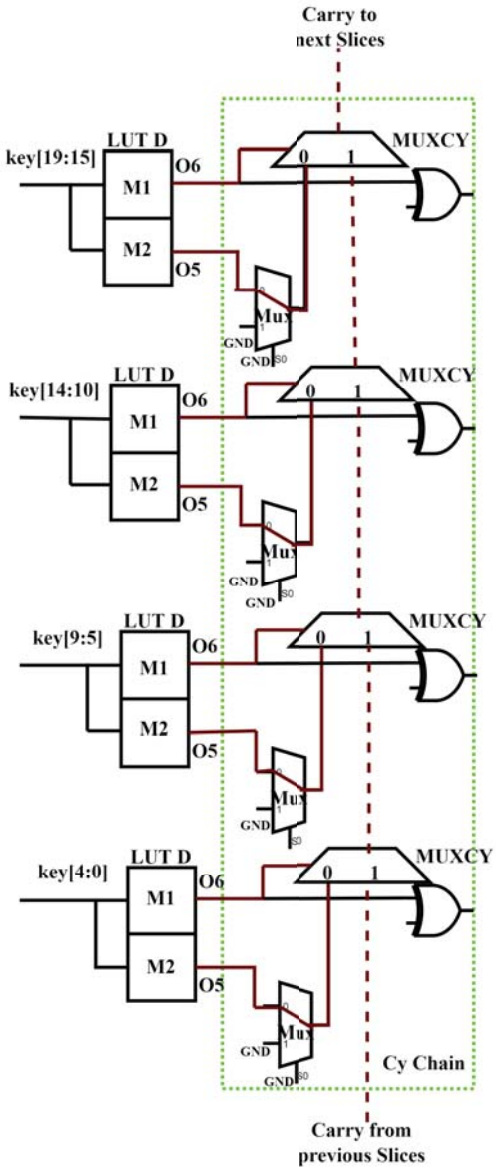

Figure 3: A 1 × 20 TCAM was mapped to from 5 × 2 LUTs

Two rules (Rule 6 and Rule 7) are read from O5 and O6 as the keyword and connected to the 5-bit LUT input (A4:A0). Rule 6 is stored in memory M1, and Rule 7 is stored in memory M2. The rules are updated using the write address inputs as shown in Fig. 2. This LUT output is connected to the built-in carry-chain for implementing the match logic. Note that this is worth mentioning as the proposed TCAM utilized the one carry-chain to reduce the routing resources and extra logic needed, resulting in a higher design clock rate and less delay time.

To implement 1×20 TCAM as shown in Fig.3, LUTs (LUTA, LUTB, LUTC, and LUTD) are stacked with different keywords [19:0] and have eight different rules through O5 and O6. These eight rules are connected to the single carry chain logics i.e., LUT output O6 is connected to the select line of carry-chain logic, and the LUT output O5 is connected to the input of the same carry-chain logic. With the proposed TCAM, six input LUTs in a dual-output mode are combined with eight flip-flops as well as the carry chain in a single slice to provide an 8 × 5 configuration (compared to 4 × 6 for single output LUTs). As shown in Fig. 2, O5 is connected to the select signal of the carry chain through D5FFMUX, D5FF, DOUTMUX and LUT output O6 is connected to the data inputs of the carry chain via DCY0, MUXCY, D5FFMUX and D5FF. In this manner, a fully pipelined TCAM structure is designed, resulting in improved performance such as TP and EDP, while resource utilization is the same as in a non-pipelined structure.

TCAMs with large dimensions can combine multiple basic blocks. To increase the depth of a TCAM, more basic blocks have to be stacked vertically, where each basic block implements a 1 × 20 TCAM. All the basic blocks have the same keyword. As shown in Fig. 4(c), TCAM's width can also be extended by configuring multiple basic blocks with the same depth simultaneously to produce the final match signals.

Fig. 5 shows the proposed TCAM update logic (highlighted in blue dots). The Write Enable "WE" line is short and connected to all the LUTRAM blocks. In the current write cycle, "WE" lines are demultiplexed with the row ID to determine which row needs to be updated. For columns with the same key lines, column update logic takes care of blocks in the same column. Serial shift registers are implemented as SRL32 in SLICEM for column update logic. For the depth varying from 64 to 1024 × 20 columns, only 32 SRL32 is required for implementation. Similarly, for the depth varying from 64 to 1024 × 40 columns, only 64 SRL32 is required for implementation. It is noted that, when the key size is increased from 20 to 40, the SRL32 utilization is doubled. An incoming key value is compared with the 5-bit

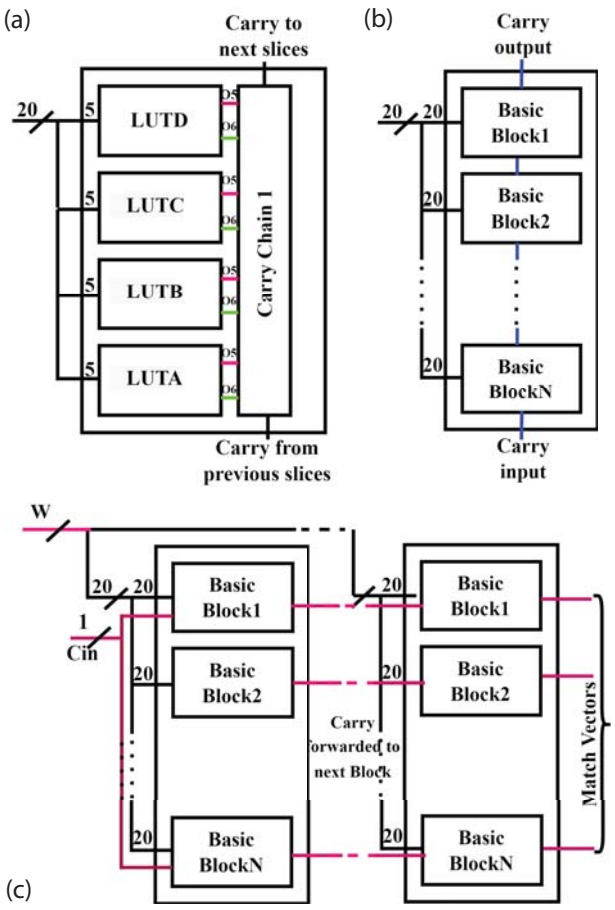

Figure 4: a Architecture of Basic Block combining four LUTs and carry chain into 1 × 20 TCAM; b Depth Extension; c Width Extension

global counter and the binary value is written into the SRL32. A 3-bit counter is present inside the SRL fill logic that controls the demultiplexer, and it increments once every 33 times the global 5-bit counter.

4 Results and discussion

The Xilinx Virtex-7, 28-nm, XC7V2000TFHG1761-2L FPGA device is used to implement the proposed TCAM architecture with a –2 speed grade. There are 1,221,600 LUTs, 344,800 LUTRAMs, 2,443,200 FFs, and 305,400 SLICEs on this device. Performance evaluations of different TCAM sizes are also done using the Vivado HLx 2017.3 design suite. TCAM has a key size of 20 to 160 bits and several rules ranging from 64 to 1024 bits. A SLICE capable of implementing an 8 × 5 TCAM is the fundamental building block. As a result, keys multiply by 5 and rules multiply by 8. The results are based on the implementation of post-place and post-route.

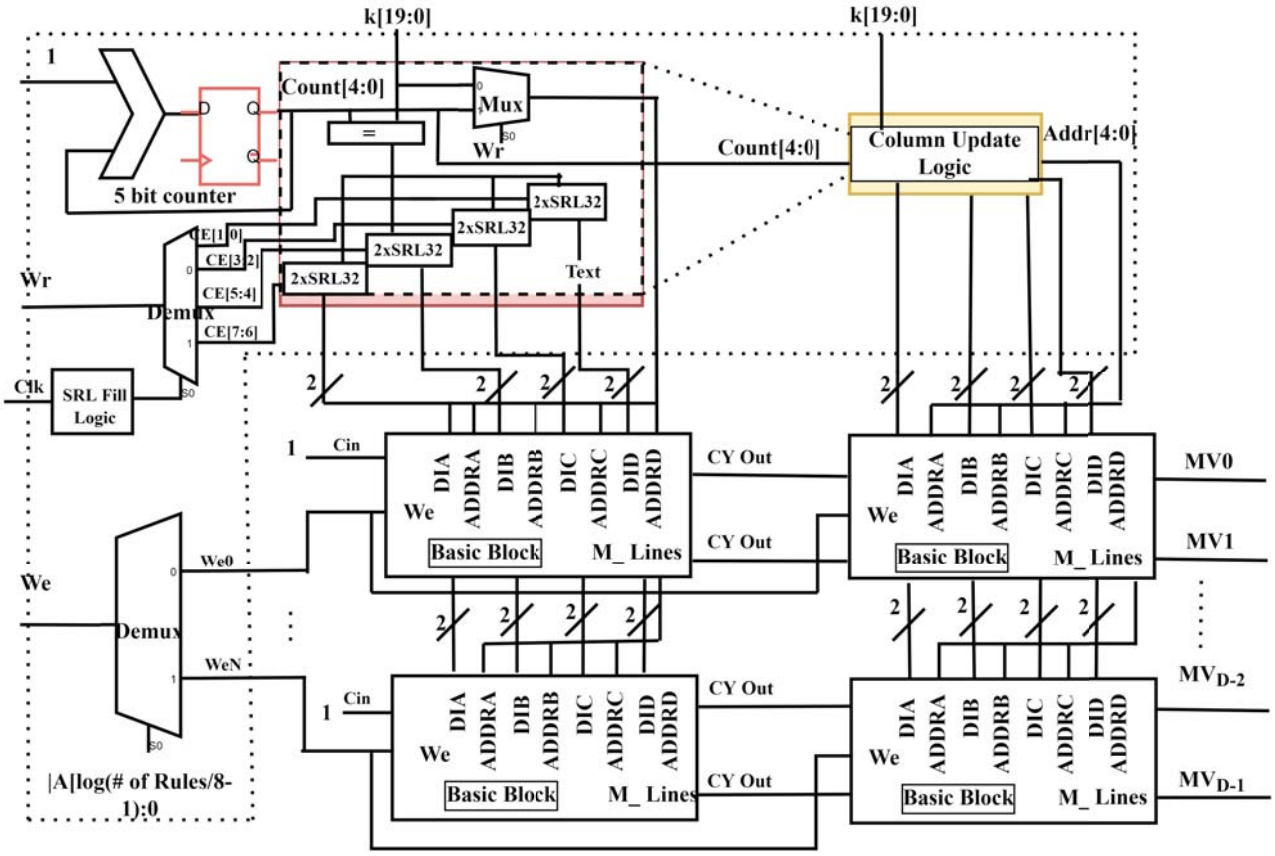

Figure 5: Architecture of the proposed TCAM with update logic.

TCAM storage and update logic resources are required for different configurations, that is, 512 × 20, 512 × 40, 512 × 80, 512 × 160, and 1024 × 160 as mentioned in Table 1. The table shows that the storage part of TCAMs uses a lot more resources than the update logic.

Table 1: Resource Utilization for the proposed TCAM

TCAM Size(D x W)		512 × 20	512 × 40	512 × 80	512 × 160	1024 × 160
TCAM	LUT as logic	0				
	LUT RAM	1024	2048	4096	8192	16384
	FFs	2048	4096	8192	16384	32768
Update Logic	LUT as logic	84	104	144	224	382
	LUT RAM	32	64	128	256	256
	FFs	23	23	23	23	23

Resources utilized for the different TCAM sizes are listed in Table 2, which does not contain a priority encoder or match reduction. The proposed TCAM architecture only utilizes three FPGA resources: LUTRAMs for storing TCAM rules, FFs registers for deeper pipelining, and slice carry chain for match logic. It is important to note that no logic LUTs are needed to implement AND gates since the rules are linked to LUT Carry-chains. Resource utilization can be observed to be directly related to the TCAM's size. It should be noted that the LUT as logic for the entire TCAM will be zero since Match logic can be implemented with the help of the LUT carry chain. As an example, the 64 × 20 TCAM requires 128 LUTRAMs and 256 FF for pipelining, in addition to 32 CARRY4 for data transfer. In the proposed TCAM, FFs and carry chains are used within the same SLICE. FFs are utilized by multiplying the number of blocks by the depth of TCAM. The Virtex-7 FPGA from Xilinx can support eight FFs within a single SLICE. To maximize the utilization of SLICE resources, the proposed TCAM fully exploits the FF available in the SLICE. With this approach, a pipelined TCAM architecture can be implemented without the use of additional SLICEs.

Table 2: Resource Utilization for different Configurations of proposed TCAM

Width	Parameters	Depth			
		64	128	256	512
20	LUT RAM	128	256	512	1024
	FFs	256	512	1024	2048
	Speed(MHz)	888.4	872.8	815.5	752.6
	Power(mW)	4	9	18	34
40	LUT RAM	256	512	1024	2048
	FFs	512	1024	2048	4096
	Speed(MHz)	685.6	680.2	662.5	597.8
	Power(mW)	9	18	34	60
80	LUT RAM	512	1024	2048	4096
	FFs	1024	2048	4096	8192
	Speed(MHz)	642	628	607	584
	Power(mW)	18	34	60	109
160	LUT RAM	1024	2048	4096	8192
	FFs	2048	4096	8192	16384
	Speed(MHz)	567.8	532.8	395.4	372.5
	Power(mW)	34	60	109	122

The speed and dynamic power consumption achieved by the proposed TCAMs shown in Table 2 for the different TCAM configurations. The TCAM inserts registers between the input and the TCAM module, as well as between the TCAM module and the reduction logic. The proposed TCAM achieves speeds from 372 to 888 MHz for different sizes. However, the proposed TCAM degrades minimally as its size increases, and the degradation does not double as the TCAM's size increases. For example, when moving from 64 × 20 to 128 × 20 and from 64 × 160 to 128 × 160, the speed decreases by 15.16 and 35MHz respectively. Similarly, when moving from 64 × 20 to 64 × 40 and from 64 × 80 to 64 × 160, the speed decreases by 202 and 75.1 MHz respectively. In Table 2, TCAMs proposed in this paper scale well with size, when analysing FPGA resource utilization and clock speed. Vivado's power analyser reports these values for default switching activity, after the post-implementation. From the above table it is evident that, as the TCAM size increases, the power consumption also increases. A 64 × 20 configuration consumes 7mW of power dynamically, while a 512 × 160 configuration consumes 122mW.

Table 3 compares the proposed TCAM architecture's in terms of parameters like the normalized slices, normalized speed, PA, TP, update rate, energy per bit, and EDP with state-of-the-art FPGA TCAMs. By using the following equation, the number of normalized slices can be found:

Normalized slices =# of FPGA Slices +
*(# of 36 KBitsBRAMs * 24)* *(1)*

The normalized speed is calculated to provide a good comparison between different FPGA technology nodes:

$$N\text{speed} = \text{Speed} * \frac{\text{Technology(nm)}}{40\ (\text{nm})} * \frac{1.0}{VDD} \quad (2)$$

Throughput (TP), which is another important factor in TCAMs, is calculated with the help of the following equation:

$$\text{TP}\left(\frac{\text{Gbit}}{\text{s}}\right) = F(\text{MHz}) * \text{TCAM Width} \quad (3)$$

The proposed work has a throughput of 26.64, 50, and 83 Gbit/sec, which is better than the existing work for the TCAM sizes of 512 × 40, 512 × 80, and 512 × 160, as seen in Table 3.

The update rate is defined as the ratio of clock rate (MHz) to the clock cycles, and its unit is a million updates per second (MUPS) as follows:

$$\text{Update rate (MUPS)} = \frac{\text{Clockrate (MHz)}}{\text{clockcycles}} \quad (4)$$

In the literature, Performance per area (PA), represented mathematically is

$$\text{PA}\left(\frac{\frac{\text{Mbit}}{\text{s}}}{\text{slices}}\right) = \frac{\text{TP(Mbit/s)}}{\frac{N_{Slices}}{TCAM_{\text{Depth}}}} \quad (5)$$

Table 3: Performance Comparison with the state of the art FPGAs

Architecture	TCAM Size (D×W)	LUTRAMS (#)	Slice Registers (#)	BRAM (36K)	FPGA slices Usage	Speed (MHz)	Tp (Gbits / s)	Search cycles	Delay (ns)	Update rate	PA (/1K)	P (W)	Ebs (fJ/ bit/ search)	EDP (ns.fJ /bit/ search)
Jiang[22]	1024×150^1	20480	37556	0	20526	199	20.9	6	5.03	4.21	1.04	1.9	180	1290
REST [18]	72×28^1	8	390	1	77	50	0.98	5	20	0.07	0.998	0.11	798	22817
HP-TCAM [14]	512×36^2	0	2670	56	1637	118	4.25	5	8.47	0.23	1.045	0.19	102.2	865
G-AETCAM[9]	512×36^2	NA	NA	NA	NA	358	NC	-	2.79	1/358	NC	NC	-	-
RPE-TCAM [29]	512×36^2	NA	NA	NA	NA	319	NC	-	3.13	1/319	NC	NC	-	-
UE-TCAM [17]	512×36^2	0	1758	32	913	202	7.26	2	4.95	0.4	3.16	0.08	42.3	210
Xilinx Locke [30]	256×32^3	4096	341	0	1406	100	5.2	1	10	9.6	1.05	0.09	68	413
Comp-TCAM [28]	512×36^1	1536	-	16	541	525	-	-	-	-	10.8	-	-	-
D-TCAM[26]	512×36^1	NA	NA	0	968	460	16.56	-	2.17	NA	8.76	NA	NA	NA
	512×72^1	NA	NA	0	2357	214	15.41	-	4.67	NA	3.35	NA	NA	NA
	512×144^1	NA	NA	0	4835	259	37.3	-	3.86	NA	3.95	NA	NA	NA
DURE [24]	512×36^2	4096	1174	0	1668	335	12.06	1	2.99	5.15	3.7	0.05	28	84
	1024×144^2	32768	2700	0	9654	175	25.2	1	5.72	2.7	2.67	0.48	32.8	187
BPR-TCAM [8]	512×40^1	0	1105	0	768	360	10.08	1	2.78	-	6.72	-	-	-
	512×80^1	0	1185	0	1280	188	12.77	1	5.32	-	2.55	-	-	-
	512×160^1	0	1345	0	2560	114	12.768	1	8.77	-	2.55	-	-	-
	1024×144^2	0	3029	0	4608	111.49	16.05	1	8.97	-	3.57	-	-	-
Frac-TCAM [27]	512×40^1	2048	4096	0	768	588	16.46	1	1.7	15.47	10.98	0.065	-	-
	512×80^1	4096	8192	0	1408	473.9	37.91	1	2.11	17.11	13.79	0.12	-	-
	512×160^1	8192	16384	0	2944	254.8	40.77	1	3.92	9.58	7.09	0.15	-	-
	1024×160^1	16384	32786	0	5888	250	39.95	1	4	9.4	6.95	0.19	-	-
Proposed work	512×40^2	2048	4096	0	598	666	26.64	1	1.5	20.18	22.81	0.034	16.60	24.9
	512×80^2	4096	8192	0	1105	635	50.8	1	1.57	19.24	22.86	0.065	15.87	24.92
	512×160^2	8192	16384	0	2222	524	83.84	1	1.91	15.88	19.32	0.110	13.43	25.65
	1024×40^2	4096	8192	0	1374	563	22.52	1	1.78	17.06	16.78	0.070	17.09	30.42
	1024×80^2	8192	16384	0	2428	532	42.56	1	1.88	16.12	17.95	0.120	15.87	29.84
	1024×160^2	16384	32786	0	4315	421	67.36	1	2.38	12.76	15.99	0.160	11.60	27.60

[1] Virtex-7 (28 nm), [2]Virtex-6 (40 nm), [3]Virtex-5 (65 nm)

PA results in Table 3 show that the suggested TCAM implementation outperforms prior work by a narrow margin. The lower resource usage is due to the search and matching logic. Then, using equation 6, the energy/bit/search (Ebs) is calculated.

$$Energy = \frac{Power\ (W)}{Frequency\ (Hz)\ *\ Depth\ *\ Width} \quad (6)$$

Another important parameter for comparing TCAMs is Energy Delay Product (EDP), which is determined using the following equation:

*Energy Delay Product (EDP)=Energy*Delay* (7)

It is observed that the proposed TCAM is 22%, 21%, 24%, and 26% more efficient than Frac-TCAM respectively, for the different TCAM sizes in slice resource utilizations. Compared to state-of-the-art designs, the proposed TCAM has less slice utilization due to slice carry chain utilization. Compared with Frac-TCAM, BPR-TCAM, DURE, D-TCAM, and Comp-TCAM, the proposed TCAM achieves higher clock speed due to inbuilt slice carry chain utilization, SLICEM registers, and RAM32M for the different TCAM sizes.

TCAM size 512×40 has a dynamic power consumption of 34mW and a delay time of 1.5 ns. Thus, the energy consumption is 16.60 fJ/bit/search and the EDP is 24.9 ns.fJ/bit/search. The EDP achieved in the proposed

work is 3.37 and 8.4 times lower than that of DURE [24] and UE-TCAM [17] respectively, and is the lowest among the various FPGA-based TCAM architectures. TCAM size 1024 × 160 is a larger TCAM that uses 190mW of dynamic power and has a delay time of 2.38ns. Therefore, its EDP is 27.60 ns.fJ/bit/search, almost 46 times less than that of the 150-kbit TCAM implementation in [22]. Thus, the proposed work is also a very energy-efficient TCAM architecture.

5 Conclusion

An FPGA implementation of a TCAM that uses SRAM for higher energy efficiency and resource efficiency is presented. By leveraging the architecture of Xilinx FPGAs, TCAMs can be emulated efficiently. Utilizing LUTRAMs with dual outputs within the latest seven series FPGAs, as well as built-in slice registers and carry chains, a scalable TCAM architecture is proposed. When compared to the conventional 8 × 5 TCAM, the suggested design can map an 8 × 1 TCAM, virtually doubling the utilization density. In addition, the use of in-slice registers to pipeline LUTRAM outputs allows for high-speed operation, and the utilization of carry-chain logic for match reduction archives lower slice utilization. Hence, both logic utilization and TP are enhanced, resulting in a better PA compared with the existing approaches. It achieved an EE and PA that were at least 3.34 and 8.4 times and 56% better than those of the other FPGA-based TCAM solutions, respectively. The large size of TCAM emulation on SRAM-based FPGAs, this solution outperforms the existing solutions with its low dynamic power consumption.

6 Conflict of Interest

The authors declare no conflict of interest.

7 References

1. Zilberman, N. Audzevich, Y. Covington, G.A. Moore, A.W. "NetFPGA SUME: Toward 100 Gbps as Res for the different TCAM sizes in slice resource utilization search Commodity". IEEE Micro 2014, vol. 34, 32–41.
2. Xilinx. "SDNet Packet Processor User Guide UG1012 (v2018.1)", Xilinx: San Jose, CA, USA, 2018.
3. Reviriego, P. Pontarelli, S. Levy, G. "CuCoTrack: Cuckoo filter based connection tracking". Inf. Process. Lett. 2019, vol.147, 55–60.
4. Sundstron, M. Larzon, L. Åke "High-performance longest prefix matching supporting high-speed incremental updates and guaranteed compression". In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005 Vol.3, pp. 1641–1652.
5. Yu, F. Katz, R. Lakshman, T. "Efficient Multimatch Packet Classification and Lookup with TCAM". IEEE Micro 2005, vol.25, 50–59.
6. Pagiamtzis, K. Sheikholeslami, A. "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey." IEEE J. Solid-state Circuits 2006, vol.41, 712–727.
7. Bosshart, P. Gibb, G. Kim, S.H. Varghese, G. McKeown, N. Izzard, M. Mujica, F. Horowitz, M. "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN". In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '13), Hong Kong, China, 12–16 August 2013.
8. A. Ullah, A. Zahir, N. A. Khan, W. Ahmad, A. Ramos, and P. Reviriego, "BPR-TCAM—Block and partial reconfiguration based TCAM on Xilinx FPGAs," Electronics, vol. 9, no. 2, p. 353, 2020.
9. M. Irfan and Z. Ullah, "G-AETCAM: Gate-based area-efficient ternary content-addressable memory on FPGA," IEEE Access, vol. 5, pp. 20785–20790, 2017.
10. Z. Ullah, "LH-CAM: Logic-based higher performance binary CAM architecture on FPGA," IEEE Embedded Syst. Lett., vol. 9, no. 2, pp. 29–32, Jun. 2017.
11. M. Irfan and A. Ahmad, "Impact of initialization on gate-based area efficient ternary content-addressable memory," in Proc. Int. Conf. Comput., Electron. Commun. Eng. (iCCECE), Southend, U.K., Aug. 2018, pp. 328–332
12. H. Mahmood, Z. Ullah, O. Mujahid, I. Ullah, and A. Hafeez, "Beyond the limits of typical strategies: Resources efficient FPGAbased TCAM," IEEE Embedded Syst. Lett., vol. 11, no. 3, pp. 89–92, Sep. 2019.
13. M. Somasundaram, "Circuits to generate a sequential index for an input number in a pre-defined list of numbers," U.S. Patent 7 155 563 B1, Dec. 26, 2006.
14. Z. Ullah, K. Ilgon, and S. Baeg, "Hybrid partitioned SRAM-based ternary content addressable memory," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 12, pp. 2969–2979, Dec. 2012.
15. Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "Z-TCAM: An SRAMbased architecture for TCAM," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 2, pp. 402–406, Feb. 2015.

16. Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "E-TCAM: An efficient SRAM-based architecture for TCAM," Circuits, Syst., Signal Process., vol. 33, no. 10, pp. 3123–3144, Oct. 2014.
17. Z. Ullah, M. K. Jaiswal, R. C. C. Cheung, and H. K. H. So, "UE-TCAM: An ultra-efficient SRAM-based TCAM," in Proc. TENCON-IEEE Region 10 Conf., Macao, China, Nov. 2015, pp. 1–6.
18. A. Ahmed, K. Park, and S. Baeg, "Resource-efficient SRAM based ternary content addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1583–1587, Apr. 2017.
19. I. Ullah, Z. Ullah, and J.-A. Lee, "Efficient TCAM design based on multipumping-enabled multiported SRAM on FPGA," IEEE Access, vol. 6, pp. 19940–19947, 2018.
20. F. Syed, Z. Ullah, and M. K. Jaiswal, "Fast content updating algorithm for an SRAM-based TCAM on FPGA," IEEE Embedded Syst. Lett., vol. 10, no. 3, pp. 73–76, Sep. 2018.
21. I. Ullah, Z. Ullah, and J.-A. Lee, "EE-TCAM: An energy-efficient SRAM-based TCAM on FPGA," Electronics, vol. 7, no. 9, p. 186, Sep. 2018, https://doi.org/10.3390/electronics7090186
22. W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in Proc. Architectures Netw. Commun. Syst., Oct. 2013, pp. 71–82.
23. Reviriego, P. Ullah, A. Pontarelli, S. "PR-TCAM: Efficient TCAM Emulation on Xilinx FPGAs Using Partial Reconfiguration". IEEE Trans. Very Large Scale Integr. Syst. 2019, 27, 1952–1956.
24. Ullah, I. Ullah, Z. Afzaal, U. Lee, J.-A. "DURE: An Energy- and Resource-Efficient TCAM Architecture for FPGAs With Dynamic Updates". IEEE Trans. Very Large Scale Integr. Syst. 2019,vol. 27, 1–10.
25. P. Maidee, "Multiplexer-based ternary content addressable memory," U.S. Patent 9 653 165, May 16, 2017.
26. 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.
27. Ali Zahir, Shadan Khan Khattak, Anees Ullah , Pedro Reviriego , Fahad Bin Muslim, and Waleed Ahmad "FracTCAM: Fracturable LUTRAM-Based TCAM Emulation on Xilinx FPGAs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 12, pp. 2726-2730, DEC. 2020
28. Muhammad Irfan, Hasan Erdem Yantır, Zahid Ullah, and Ray C. C. Cheung, "Comp-TCAM: An adaptable composite Ternary content-addressable Memory on FPGAs," IEEE Embedded Systems Letters, Nov. 2021.
29. Irfan, Z. Ullah, M. H. Chowdhury, and R. C. C. Cheung, "RPETCAM: Reconfigurable power-efficient ternary content-addressable memory on FPGAs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 28, no. 8, pp. 1925–1929, Aug. 2020.
30. K. Locke, Parameterizable Content-Addressable Memory, Xilinx, San Jose, CA, USA, 2011.
31. TCAM Source Code. Accessed: Aug. 23, 2023. [Online]. Available: https://github.com/sridharraj240/TCAM

Received: 08. 06. 2022
Accepted: 03. 09. 2022