J. of Mult.-Valued Logic & Soft Computing, Vol. 0, pp. 1–15
Reprints available directly from the publisher
Photocopying permitted by license only

An Update Method for a Low Power CAM Emulator using an LUT Cascade Based on an EVMDD (*k*)

HIROKI NAKAHARA[1], TSUTOMU SASAO[2], MUNEHIRO MATSUURA[3] AND HISASHI IWAMOTO[4]

[1]*Ehime University, Matsuyama, 790-8577, Japan*
E-mail: nakahara@cs.ehime-u.ac.jp
[2]*Meiji University, Kawasaki, 214-8571, Japan*
E-mail: sasao@cs.meiji.ac.jp
[3]*Kyushu Institute of Technology, Fukuoka, 820-8502, Japan*
E-mail: matsuura@cse.kyutech.ac.jp
[4]*REVSONIC Corp., Yokohama, Japan*
E-mail: hisashi-iwamoto@revsonic.com

Received: May 30, 2014. Accepted: October 31, 2014.

Core routers perform longest prefix matching (LPM) using content addressable memories (CAMs). With the rapid growth of the Internet, LPM has become the bottleneck in network traffic management. In the previous publication, we have proposed an area-efficient and high-performance CAM emulator using an LUT cascade based on an edge-valued multi-valued decision diagram (EVMDD (*k*)). In the internet, registered vectors must be updated frequently. In this paper, we propose an algorithm to update an LUT cascade. We implemented the proposed algorithm on the ARM processor. Its update time is shorter than the peak update time of the BGP protocol. Also, we analyzed the power consumption of the LUT cascade with respect to both the static and the dynamic power. Experimental results show that, as for the lookup speed per area and the power consumption, our architecture outperforms existing CAM realizations on FPGAs.

Keywords: Content addressable memory (CAM), multi-valued decision diagram, longest prefix matching (LPM)

1 INTRODUCTION

1.1 Demands of LPM Architecture

Routers forward packets in IP address lookups using **longest prefix matching (LPM)**. With the rapid growth of the Internet, LPM has become the

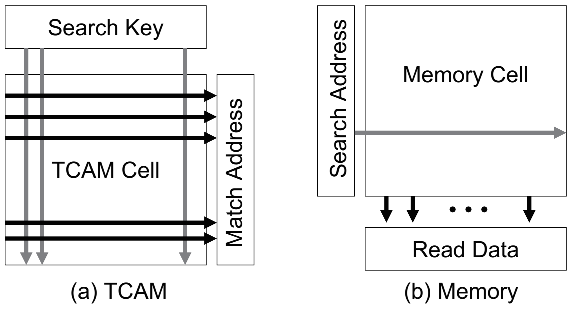


FIGURE 1
Dynamic power consumptions for the TCAM and the memory.

bottleneck in the network traffic management. In this paper, we consider a CAM emulator using an LUT cascade on the FPGA, which has the following features:

High throughput per area: Recently, core routers work at the 100 Gbps link speed for the minimum packet size (40 bytes). A parallel processing is an effective method to increase the system throughput. In this case, the throughput per area is an important measure [4]. A modern FPGA consists of lookup-tables (Slices), on-chip memories (BRAMs), arithmetic circuits (DSP48Es), and so on. Thus, a balanced usage of hardware resources in FPGAs is the key to achieve a high throughput per area.

High-speed updatable: The IP addresses on routers are frequently updated (added and deleted). For a border gateway protocol (BGP), its peak number of updates per second is about 10,000 [1]. The simplest method to update the LPM architecture on an FPGA is direct rewriting of its interconnections using the new configuration data. However, since the time to generate the new configuration is very long, it is infeasible. Thus, the high-speed update on the LPM architecture is essential.

Low-power consumption: The conventional routers use ternary content addressable memories (TCAMs) to realize LPM. With the rapid increase of traffic, core routers dissipate the major part of the total network power [18], since the TCAM performs the LPM by activating all of the TCAM cells (Figure 1 (a)). Thus, we cannot use TCAMs any more, since they dissipate too much dynamic power. Le and Prassana [7] proposed a memory-based IP lookup architecture on field programmable gate arrays (FPGAs), which dissipate lower power than TCAMs, since the memory reads the data by activating only one word corresponding to the address (Figure 1 (a)). In this paper, we consider the memory-based LPM architecture.

1.2 Proposed Method

In the previous publications, we proposed CAM emulators based on the edge-valued multi-valued decision diagrams (EVMDD (k)s) [11] for the IP address matching [13] and the packet classification [12]. They are more efficient than other FPGA implementations. However, they did not consider the update method.

Previous work showed that the LUT cascade based on the EVMDD (k) is smaller than one based on the MTMDD (k). The addition and deletion can be done in time that is proportional to the number of cells in the LUT cascade based on the multi-terminal MDD (MTMDD (k)) [13]. We applied this method to the LUT cascade based on the EVMDD (k) [8]. Thus, the proposed LUT cascade based on the EVMDD (k) satisfies above conditions.

The power consumption consists of the dynamic power consumption and the static power consumption. Since the LUT cascade is the memory-based, its dynamic power is lower than that of the TCAM-based one. Also, since the LUT cascade based on the EVMDD (k) is smaller than that based on the MTMDD (k), the proposed EVMDD (k) based one dissipates lower static power than that of the MTMDD (k) based one. We will analyze the static power and the dynamic power.

The paper is the enhanced version of [8].

1.3 Organization of the Paper

The rest of the paper is organized as follows: Chapter 2 defines an LPM function; Chapter 3 introduces the LUT cascade based on an EVMDD (k); Chapter 4 shows the update method for the LUT cascade based on an EVMDD (k); Chapter 5 shows experimental results; and Chapter 6 concludes the paper.

2 DEFINITION OF A LONGEST PREFIX MATCHING (LPM) FUNCTION

Definition 1. *The* **LPM table** *stores ternary vectors of the form* $VEC_1 \cdot VEC_2$*, where* VEC_1 *consists of* $0's$ *and* $1's$*, and* VEC_2 *consists of* $*'s$ *(don't cares). The* **length** *of prefix is the number of bits in* VEC_1*. To assure that the longest prefix address is produced, entries are stored in the descending prefix length. Let* $B \in \{0, 1\}$*. The* **LPM function** *[15] is the logic function* $\vec{f} : B^n \to B^m$*, where* $\vec{f}(x)$ *is the minimum address of* VEC_1 *corresponding to* $\vec{x}$*. If there is no such vector,* $\vec{f}(\vec{x}) = 0^m$*.*

We can assign an arbitrary monotone increasing index to the LPM table. In this paper, we use an M_1**-monotone increasing function** [9] to reduce the amount of memory.

x_0	x_1	x_2	x_3	**Rule**
0	0	0	0	1
0	0	0	1	2
1	0	0	0	3
1	0	0	1	4
0	0	1	*	5
1	0	0	*	6
0	1	*	*	7
		otherwise		0

TABLE 1
Example of LPM function.

Definition 2. *[9] Let Z be the set of integers, and I be a set of integers including* 0. *An integer function $f(X) : I \to Z$ such that $0 \leq f(X+1) - f(X) \leq 1$ and $f(0) = 0$ is* ***an M_1-monotone increasing function*** *on I. That is, for an M_1-monotone increasing function $f(X)$, $f(0) = 0$, and the increment of X by one increases the value of $f(X)$ by at most one.*

Example 1. *Table 1 shows an LPM function that is also an M_1-monotone increasing function.*

3 CAM EMULATOR USING AN LUT CASCADE BASED ON AN EVMDD (K)

3.1 LUT Cascade Based on an MTMDD (k)

Definition 3. *A* ***binary decision diagram (BDD)*** *[2] is obtained by applying* ***Shannon expansions*** *repeatedly to a logic function f. Each non-terminal node labeled with a variable x_i has two outgoing edges which indicate nodes representing cofactors of f with respect to x_i.*

Definition 4. ***A multi-terminal BDD (MTBDD)*** *[3] is an extension of a BDD and represents an integer-valued function. In the MTBDD, the terminal nodes are labeled by integers.*

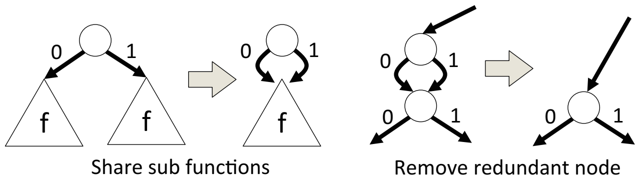


FIGURE 2
Conversion of a binary tree node into an MTBDD node.

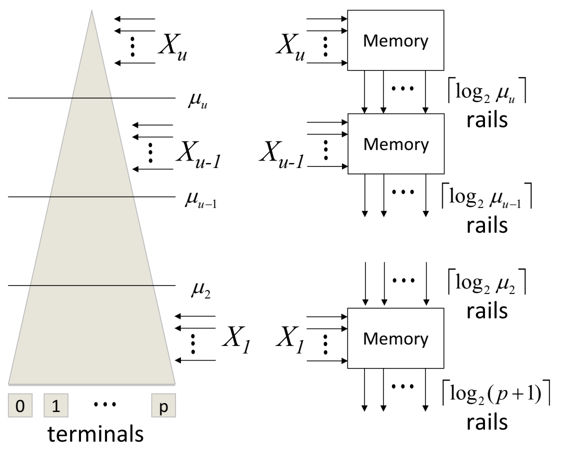

X_u
μ_u
$X_{u\text{-}1}$
μ_{u-1}
μ_2
X_1
0
1
p
terminals
Memory
$\lceil \log_2 \mu_u \rceil$ rails
$\lceil \log_2 \mu_{u-1} \rceil$ rails
$\lceil \log_2 \mu_2 \rceil$ rails
$\lceil \log_2 (p+1) \rceil$ rails

FIGURE 3
An LUT cascade based on an MTMDD (k).

Definition 5. *Let $X = (X_1, X_2, \ldots, X_u)$ be a partition of the input variables, and $|X_i|$ be the number of input variables in X_i. X_i is called* ***a super variable****. When the Shannon expansions are performed with respect to super variables X_i, where $|X_i| = k$, all the non-terminal nodes have 2^k edges. In this case, we have a* ***multi-valued multi-terminal decision diagram (MTMDD(k))*** *[5]. Note that, an MTMDD(1) means an MTBDD.*

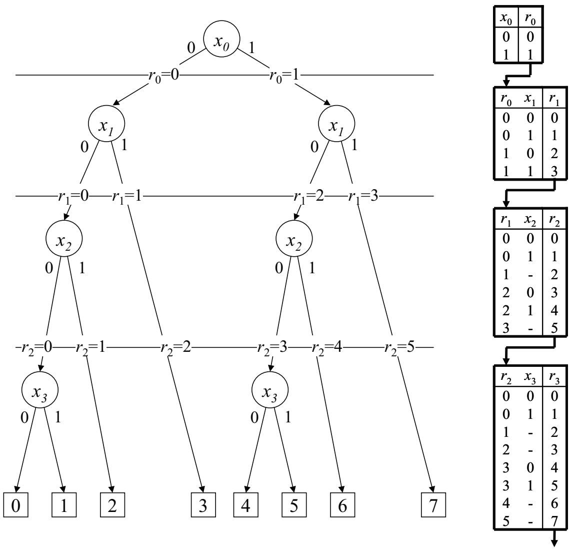
Definition 6. ***The width of the MDD (k) at the height*** *k is the number of edges crossing the section of the MDD (k) between super variables X_{i+1} and X_i, and denoted by μ_i where the edges incident to the same node are counted as one.*

Let p be the number of rules, and $|X| = n$. An M_1-monotone increasing function can be realized by **an LUT cascade** [16] shown in Figure 3. Connections between LUT_i and LUT_{i-1} requires $r_i = \lceil log_2\ \mu_i \rceil$ rails. Since a modern FPGA has BRAMs and distributed RAMs (realized by Slices), LUT cascades are easy to implement. The amount of memory for LUT_i based on an MTMDD (k) is $r_i \cdot 2^{(k+r_{i+1})}$. Thus, the total amount of memory for an LUT cascade is $M = \sum_{i=0}^{u} r_i \cdot 2^{(k+r_{i+1})}$.

Example 2. *Figure 4 shows an example of an LUT cascade based on an MTMDD (k).*

As for an M_1-monotone increasing function, the upper bound on the number of rails in the LUT cascade has been analyzed [14].*

* In [14], the M_1-monotone increasing function is called segment index encoder function.



x_0	r_0
0	0
1	1

r_0	x_1	r_1
0	0	0
0	1	1
1	0	2
1	1	3

r_1	x_2	r_2
0	0	0
0	1	1
1	-	2
2	0	3
2	1	4
3	-	5

r_2	x_3	r_3
0	0	0
0	1	1
1	-	2
2	-	3
3	0	4
3	1	5
4	-	6
5	-	7

FIGURE 4
Example of an LUT cascade based on an MTMDD (k).

Theorem 1. *Let p be the number of unique indices for the M_1-monotone increasing function. The upper bound on the number of rails in the LUT cascade is $r = \lceil log_2(p+1) \rceil$.*

3.2 CAM Emulator Using an LUT Cascade Based on an EVMDD (*k*)

To reduce the amount of memory for an LUT cascade, we introduce an LUT cascade based on **an edge-valued multi-valued decision diagram (EVMDD (*k*))**, which is an extension of an EVBDD [6]. An EVBDD consists of one terminal node representing zero and non-terminal nodes with a weighted 1-edge, where the weight has an integer value α. An EVBDD is obtained by recursively applying the conversion shown in Figure 5 to each

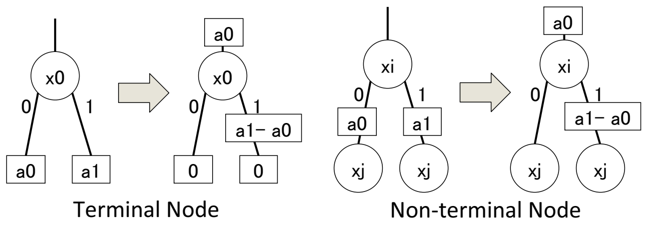


FIGURE 5
Conversion of an MTBDD node into an EVBDD node.

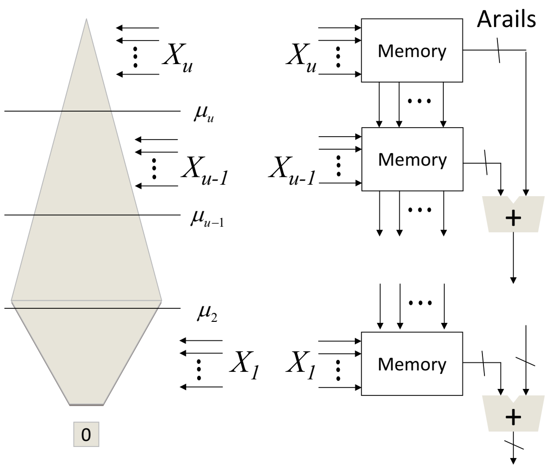


FIGURE 6
An LUT cascade based on an EVMDD (k).

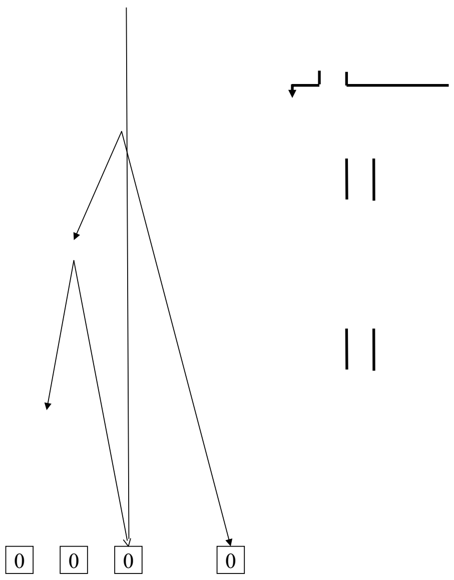
non-terminal node in an MTBDD. Note that, in the EVBDD, 0-edges have zero weights.

Definition 7. An edge-valued MDD (k) (EVMDD (k)) *[10] is an extension of the MDD (k), and represents a multi-valued input M_1-monotone increasing function. It consists of one terminal node representing zero and non-terminal nodes with edges having integer weights, and 0-edges always have zero weights.*

Let p be the number of rules, and $|X| = n$. An M_1-monotone increasing function is efficiently realized by an LUT cascade with adders [11] shown in Figure 6. In this case, the rails represent sub-functions in the EVMDD (k). Each LUT_i has an additional rail representing the weight of the edge. We call such an output **Arail** which consists of a_i rails. Since the width of the EVMDD (k) for M_1-monotone increasing function is often smaller than that of the MTMDD (s), we can reduce the amount of memory for the LUT cascade by using an EVMDD (k). Since adders are realized by DSP blocks (DSP48Es), FPGA resources are efficiently used.

Example 3. *Figure 7 shows an example of an LUT cascade based on an EVMDD (k).*

The amount of memory for LUT_i is $(r_i + a_i) \cdot 2^{k+r_{i+1}}$. Let $|X| = n$ be the number of inputs, and $k = |X_i|$. The LUT cascade requires $u = \lceil \frac{n}{k} \rceil$


0
0
0
0

non-zero, while the deletion is archived by rewriting the index corresponding to zero. Thus, the update requires both an addition and a deletion.

4.2 Update of the LUT Cascade Based on the EVMDD (k)

To update the LUT cascade based on the EVMDD (k), first, we update the EVMDD (k) corresponding to the update vector. We show an algorithm to update the EVMDD (k) as follows:

Algorithm 1.

(1) Traverse the EVMDD (k) from the root node to the terminal node corresponding to the update vector by converting the EVMDD node into the MTMDD node.
(2) When it reaches the terminal node, then rewrite the terminal value.
(3) Return to the root node by converting the MTMDD node to the EVMDD node shown in Figure 5.
(4) Terminate.

Then, we modify the memory of the LUT cascade according to the modified part of the EVMDD (k). Modification of the LUT cascade can be done as follows:

Algorithm 2.

(1) Apply the Algorithm 1.
(2) Traverse the modified EVMDD (k) corresponding to the update vector. Then, modify the memory of the LUT cascade corresponding to the modified node on the EVMDD (k).
(3) Terminate.

4.3 Analysis of the Memory Size of the LUT Cascade

We analyze the upper bound on the memory size with respect to the number of update vector p'.

Theorem 2. *Let p be the width of the EVMDD (k) representing M_1 monotone increasing function. When p' vectors are updated, the width of the EVMDD (k) is at most $p + p' + 1$.*

Proof. As for the EVMDD (k), by shifting down all the edge values to the terminal node, we have the MTMDD (k). From Theorem 1, the width of the MTMDD (k) increases at most p'. Thus, the width of the EVMDD (k) is at most $p + p' + 1$ after the update of p' vectors.

By Theorem 1, we have an upper bound of the number of rails on the LUT cascade from the upper bound of the width of the EVMDD (k)

Theorem 3. *Let p be the width of the EVMDD (k) representing M_1 monotone increasing function. After p' vectors are updated, the number of rails on the LUT cascade based on the EVMDD (k) is at most $r = \lceil log_2(p + p' + 1)\rceil$.*

Proof. From Theorem 2, the width of the EVMDD (k) is at most $p + p' + 1$. Obviously, the number of rails is at most $r = \lceil log_2(p + p' + 1)\rceil$.

Theorem 3 introduces the upper bound of the number of rails. In the LPM function, the length of vector n is fixed. For example, that for the IPv4 address is 32, while that for the IPv6 address is 128. Therefore, Expr. (1) shows the upper bound of the memory size of the LUT cascade based on the EVMDD (k).

Corollary 1. *Assume that p vectors are stored on an LUT cascade based on the EVMDD (k). When p' vectors are updated, then, its memory size becomes at most $\frac{n}{k}2^{n/k+1}\lceil log_2(p + p' + 1)\rceil$, where n is the length of the vector.*

Proof. The upper bounds of both the adder rail and the rail are the same, and $p + p'1 + 1$. Thus, the number of outputs for each LUT is at most $2\lceil log_2(p + p' + 1)\rceil$. The number of words for each memory is $2^{n/k}$, and the number of memories on the LUT cascade is $\frac{n}{k}$. Thus, we have $\frac{n}{k}2^{n/k+1}\lceil log_2(p + p' + 1)\rceil$.

5 EXPERIMENTAL RESULTS

5.1 Comparison of Update Time

We implemented Algorithm 2 using the ARM Cortex-A9 MPCore (666 MHz, L1 cache 32KB I/D, L2 cache 512KB) on the Avanet Corp. Zedboard which has a 512 MB DDR3 SDRAM. The operating system (OS) was Ubuntu 12.04 LTS. We wrote Algorithm 2 by C-language. Then, we generated the execution code by gcc compiler with an optimize option -O3. The size of the execution code was 96.3 KB. Thus, the proposed program and the work area (stack and heap) fit in the available memory. Figure 8 compares the update time of LUT cascades with respect to the number of updates. Although the update time for the EVMDD (k) based one is longer than that for the MTMDD (k) based one, it is about a half of the required time for the BGP protocol which requires 10,000 updates per second. Thus, its update time is acceptable.

5.2 Comparison of Area-Performance Efficiency

We assumed that the length of the vector is 32. We implemented the Xilinx Inc. CAM IPs [19] on the Xilinx Inc. FPGA (Virtex 4: XC4VLX25). Figure 9

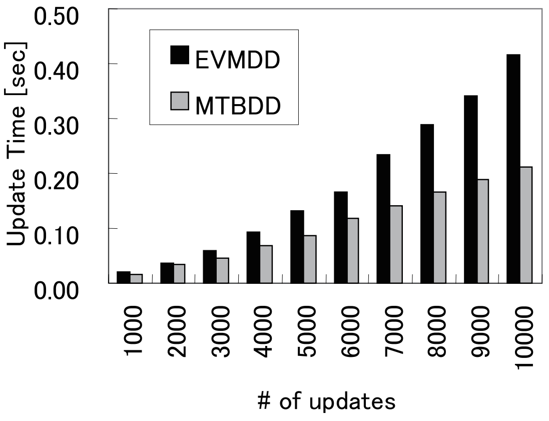


FIGURE 8
Comparison of Update Time.

shows a 4-input LUT realization of the CAM. Each 4-input LUT realizes a 4-bit registered vector. The slices of the Xilinx FPGA consists of the LUT and the multiplexer. The CAM IP uses cascaded multiplexers to realize the AND functions. Thus, an arbitrary length of the registered vector can be realized by cascading LUTs. In Figure 9, the encoder generates the binary number corresponding to the matched vector. Figure 10 shows a 4-input LUT and a BRAM realization of the CAM. In the BRAM, the registered vectors represented by

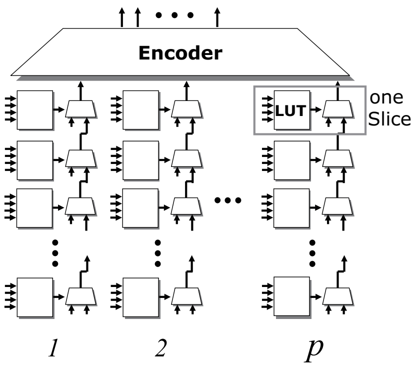


FIGURE 9
CAM IP using 4-input LUTs.

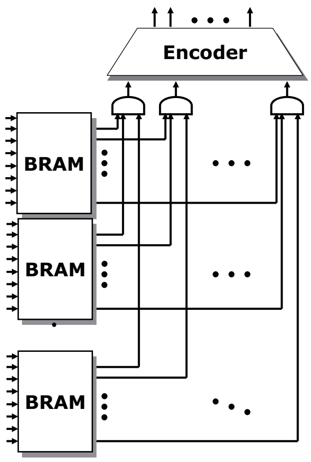


FIGURE 10
CAM IP using 4-input LUTs and BRAMs.

Realization	**4-LUT**	**BRAM +LUT**	**Cascade (MT)**	**Cascade (EV)**
# of 4LUTs	3156	1271	—	140
# of Block RAMs	—	32	18	16
Equivalent # of 4LUTs	3156	7415	3456	3072
Max. Freq. (MHz)	55.1	46.5	188.7	181.1
Efficiency (KHz/LUT)	17.4	6.2	54.6	**58.9**

TABLE 2
Comparison with other realizations (p=255).

Realization	**4-LUT**	**BRAM +LUT**	**Cascade (MT)**	**Cascade (EV)**
# of 4LUTs	6383	2806	—	154
# of Block RAMs	—	64	21	18
Equivalent # of 4LUTs	6383	15094	4032	3456
Max. Freq. (MHz)	52.6	42.1	172.9	168.8
Efficiency (KHz/LUT)	8.2	2.7	42.8	**48.8**

TABLE 3
Comparison with other realizations (p=511).

Realization	**4-LUT**	**BRAM +LUT**	**Cascade (MT)**	**Cascade (EV)**
# of 4LUTs	13294	6133	—	182
# of Block RAMs	—	128	32	22
Equivalent # of 4LUTs	13294	30709	6144	4224
Max. Freq. (MHz)	50.1	38.4	165.7	152.3
Efficiency (KHz/LUT)	3.7	1.2	26.9	**36.0**

TABLE 4
Comparison with other realizations (p=1023).

1-hot codes are written to the array by columns. For a given search vector, when the vector is registered, the BRAM produces a non-zero output. The encoder generates the binary number corresponding to the matched vector.

In the experiment, the synthesis tool was Xilinx Inc. ISE Web Pack 9.2i. As for the number of vectors p, Tables 2, 3 and 4 compare the EVMDD (k) based one with the 4-input LUT based CAM IP (4-LUT), and the block RAM and 4-input LUT based CAM IP (BRAM+LUT). Since the different realization uses different resources, to do fair comparison, we assume that one 4-input LUT corresponds to 96 bits of a BRAM [17]. We used **the equivalent**

# of Vectors p	4-LUT	BRAM +LUT	Cascade (MT)	Cascade (EV)
255	52.5	71.0	6.3	6.3
511	106.0	157.5	7.0	6.8
1023	217.5	244.0	10.6	8.2

TABLE 5
Comparison of Power Consumption (mW).

number of 4-input LUTs as follows:

$$\text{Equivalent \# of 4LUTs} \quad = \quad \text{\# of 4-input LUTs} + \text{ \# of BRAMs} \times 192.$$

Since the LPM architecture on the router requires high throughput per area, we used **efficiency [kHz/LUT]**, which shows the clock frequency per a 4-input LUT. Tables 2, 3 and 4 show that the LUT cascade based on the EVMDD (k) has the highest efficiency.

5.3 Comparison of Power Consumption

We used the HuMANDATA Inc. Virtex 4 FPGA board (XCM-201-LX25). We set the system clock frequency to 48 MHz, since the FPGA borad had the off-chip 48MHz oscillator. To make the comparison fair, we tried to make the temperature same. Table 5 compares power consumption of the EVMDD (k) based one with that of the 4-input LUT based CAM IP (4-LUT), and that of the block RAM and 4-input LUT based CAM IP (BRAM+LUT). Table 5 shows that the LUT cascade based on the EVMDD (k) dissipates the lowest power.

We analyzed the detail of the power consumption. Table 6 shows the static and the dynamic power consumption. The 4-input LUT based CAM IP (4-LUT) dissipated the highest dynamic power. Since the block RAM and 4-input LUT based CAM IP (BRAM+LUT) consumed much hardware, it dissipated the highest static power. We obtained the power consumption for a single 4-input LUT and a BRAM. The static power for the 4-input LUT was

	4-LUT		BRAM+LUT		Cascade (MT)		Cascade (EV)	
p	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic
255	15.0	37.5	45.0	26.0	2.2	4.1	2.3	4.1
511	30.5	75.5	102.5	55.0	2.5	4.5	2.5	4.3
1023	52.5	165.0	160.0	84.0	3.6	7.0	2.8	5.4

TABLE 6
Detail of Power Consumption (mW).

0.0023 mW, while that for the BRAM was 0.1222 mW. The dynamic power [†] for the 4-input LUT was 0.0059 mW, while that for the BRAM was 0.2277 mW. This means that the total power consumption for one BRAM is equal to that for 39-53 4-input LUTs. Although the LUT cascade based on the EVMDD (k) requires additional 4-input LUTs for the adder, its power consumption is equal to that of 3-4 BRAMs. Thus, as for $p = 255$ and $p = 511$, the power consumption of the EVMDD (k) based one was nearly equal to that of the MTMDD (k) based one. As for $p = 1023$, since the MTMDD (k) consumed 10 more BRAMs than the EVMDD (k) based one, the power consumption for the BRAM was dominant. Therefore, the EVMDD (k) based architecture dissipated the lowest power. Recently, since Internet traffic tends to be increased, the number of entries p will be increased. Thus, the EVMDD (k) based architecture is suitable for low power applications.

6 CONCLUSION

This paper showed an update method for a CAM emulator using an LUT cascade based on an EVMDD (k). Since the EVMDD (k) represents the M_1-monotone increasing function, it is suitable to implement the LPM function, which is used for the router. The experimental result showed that the proposed update method is acceptable for the BGP protocol which requires 100,000 updates per second. Compared with other CAM realizations, the LUT cascade based on the EVMDD (k) has a higher throughput per area and a lower power consumption.

ACKNOWLEDGEMENTS

This research is supported in part by the Grants in Aid for Scientistic Research of JSPS, and the Adaptable and Seamless Technology Transfer Program through target-driven R&D of JST.

REFERENCES

[1] The BGP Instability Report: http://bgpupdates.potaroo.net/instability/bgpupd.html

[2] R. E. Bryant, “Graph-based algorithms for Boolean function manipulation,” *IEEE Trans. on Compt.*, Vol. C-35, No. 8, 1986, pp. 677–691.

[3] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang, “Spectral transforms for large Boolean functions with applications to technology mapping,” *DAC1993*, 1993, pp. 54–60.

[†] The clock frequency was set to 48 MHz.

[4] W. Jiang and V. K. Prasanna, “Scalable packet classification on FPGA,” *IEEE Trans. on VLSI*, Vol. 20, No. 9, 2012, pp. 1668–1680.

[5] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, “Multi-valued decision diagrams: Theory and applications,” *Multiple-Valued Logic: An International Journal*, Vol. 4, No. 1–2, 1998, pp. 9–62.

[6] Y-T. Lai and S. Sastry, “Edge-valued binary decision diagrams for multi-level hierarchical verification,” *DAC1992*, 1992, pp. 608–613.

[7] H. Le and V. K. Prasanna, “Scalable high throughput and power efficient IP-lookup on FPGA,” *FCCM2009*, April, 2009.

[8] H. Nakahara, T. Sasao, and M. Matsuura, “An update method for a CAM emulator using an LUT cascade based on an EVMDD (k),” *The 44th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2014)*, 2014, pp. 1–6.

[9] S. Nagayama and T. Sasao, “Complexities of graph-based representations for elementary functions” *IEEE Trans. on Comput.*, Vol. 58. No. 1, Jan. 2009, pp. 106–119.

[10] S. Nagayama and T. Sasao, “Representations of elementary functions using edge-valued MDDs,” *ISMVL2007*, 2007.

[11] S. Nagayama, T. Sasao, and J. T. Butler, “Design method for numerical function generators using recursive segmentation and EVBDDs,” *IEICE Trans. on Fund.*, Vol. E90-A, No. 12, 2007, pp. 2752–2761.

[12] H. Nakahara, T. Sasao, and M. Matsuura, “A packet classifier using LUT cascades based on EVMDDs (k),” *FPL 2013*, 2013, pp. 1–6.

[13] H. Nakahara, T. Sasao and M. Matsuura, “A CAM emulator using look-up table cascades,” *RAW2007*, CD-ROM RAW-9-paper-2.

[14] T. Sasao, *Memory-Based Logic Synthesis, Springer.*, 2011.

[15] T. Sasao and J. T. Butler, “Implementation of multiple-valued CAM functions by LUT cascades,” *ISMVL2006*, 2006.

[16] T. Sasao, M. Matsuura, and Y. Iguchi, “A cascade realization of multiple-output function for reconfigurable hardware,” *IWLS2001*, 2001, pp. 225–230.

[17] T. Sproull, G. Brebner, and C. Neely, ”Mutable codesign for embedded protocol processing,” *FPL2005*, Aug. 24–26, 2005, pp. 51–56.

[18] R. Tucker, “Optical packet-switched WDM networks: a cost and energy perspective,” *OFC/NFOEC2008*, 2008.

[19] Xilinx Inc., “Content-Addressable Memory,” *Datasheet 253*, pp. 1–13.