# Minimization of ACL Storage by Adding Minimal Hardware of Range Matching and Logical Gates to TCAM

Haesung Hwang*, Koji Yamamoto†, Shingo Ata‡, Kazunari Inoue† and Masayuki Murata*
* Graduate School of Information Science and Technology, Osaka University, Japan
{h-hwang, murata}@ist.osaka-u.ac.jp
† Renesas Technology Corporation, Japan
{yamamoto.koji4, inoue.kazunari}@renesas.com
‡Graduate School of Engineering, Osaka City University, Japan
ata@info.eng.osaka-cu.ac.jp

## Abstract

*Ternary Content Addressable Memory (TCAM) is a special type of memory used in routers in order to achieve high speed packet classification. The classification is performed using the five fields in an Access Control List (ACL), port numbers being one of them. Since port numbers that are expressed in ranges require multiple entries in storage, this results in an increased cost of hardware. In this paper we propose a method to reduce the number of entries when expressing ranges in TCAM. We use Range Matching Devices integrated within the TCAM's control logic and optimized prefix expansion that utilizes logical AND and NOT gates in the TCAM array itself. In addition, we use real data of an existing network to show that the proposed architecture can store the ACL in an efficient way.*

## 1 Introduction

TCAM is a widely used type of memory in current IP routers typically when searching routing tables or ACLs while processing a particular packet [6]. However, there have been endless disputes as to whether or not to use TCAM because of its high power consumption, large size in chip area, and expensive manufacturing cost. Since the network administrator has to consider the fact that hardware memory is a limited resource, implementing several ACLs requires a large sized TCAM capacity and an effective management scheme. Additionally, problems of a sudden degradation in performance arise when writing ACLs after the TCAM has been exhausted, which makes it impossible to achieve a wire speed search.

One of such occasions when ACL consumes a lot of TCAM space is caused by writing port numbers which are expressed in ranges. ACL entries normally consist of five fields: source and destination IP address, source and destination port number and protocol type. According to the corresponding rule, it classifies the packet and performs a necessary action (permit/deny) only when it matches every field. Table 1 shows an example of an ACL. Port number fields in an ACL may also be given in ranges instead of exact matches, which is in fact a common procedure nowadays. When the range is expressed using powers of 2, (e.g. `1024-2047`) it can be written as a single entry using the characteristic of TCAM as (`000001**********`). However in most of the cases, the range has to be divided into several subranges to be stored in the memory which is the major reason for the increased number in required TCAM entries. The process of dividing ranges is called *Prefix Expansion (PE)*. In most cases the configuration of ACL is performed manually and, naturally, ranges based on decimal values (10, 100) are commonly used. This might be a convenient setting for a human to understand, but results in superfluous entries in TCAM.

Several methods for reducing the number of entries have been suggested, but their application to conventional TCAM is limited. Furthermore, those algorithms consist only of software solutions or the implementation of new devices outside of the TCAM. Adding an extension to the TCAM itself and thoroughly considering how the number of entries can be reduced has been to our knowledge unprecedented work, making this paper very meaningful.

In this paper, we propose a modified TCAM chip which contains *Range Matching Devices (RMD)* in order to restrain the growth of TCAM entry consumption through pre-

978-1-4244-1982-1/08/$25.00 ©2008 IEEE

fix expansion. By implementing RMD in TCAM, it is possible to maintain the conventional TCAM from the user's point of view and permits simultaneously storing arbitrary ranges. As a result, we show that hardware cost reduction can be accomplished. In addition, even in a situation when the limited number of RMDs is exhausted, we consider the effect of adding logical NOT and AND gates to the TCAM in order to achieve a flexible combination of range expressions. Using this method, we show that it is possible to better manage ACLs compared to when only using logical OR operation.

In Section 2, we introduce the problem of prefix expansion and related issues. In Section 3, we describe the proposed RMD logic structure, as well as the TCAM architecture using AND/NOT gates. In Section 4, the policy of how and what to write in RMDs will be proposed with an explanation of the prefix expansion algorithm using AND/NOT/OR gates. Section 5 provides an evaluation of the proposed method using a database of an existing network. Finally, in Section 6, we conclude the paper and give an outlook on future work.

# 2 Problems in Expressing Port Ranges and Related Work

Unlike RAM (Random Access Memory) which uses a memory address as a search key and returns the content of the memory as the result, TCAM is a memory device which searches using the content of the memory and returns the memory address where the supplied data was found. Each bit in the TCAM can either consist of `0`, `1` or $\star$, representing an arbitrary value (don't care bit). The basic explanation on TCAM can be found in [1, 2, 7, 8, 10, 11]. When a search key is given as an input, only the TCAM entry which exactly matches the key is returned. Using this fact, information of the packet header that needs to be access controlled is written in TCAM which are referenced and processed accordingly when the corresponding packets arrive.

Two kinds of CAMs exist: Binary CAM (BCAM), which only returns the entry that exactly matches the input data, and TCAM, which can also perform partial matches of the entries excluding the $\star$ part. Storing the IP addresses of ACLs and routing tables uses this don't care bit to represent a network level address. However, in order to write port numbers that are expressed in a range, it is impossible to represent the range in the form of IP addresses except to write every corresponding number in the memory. For example, when writing the range `1024-65535` in a TCAM entry, the simplest form is to write every single number to exactly match the entire entry. However, it is obvious that 64,512 entries are required to write a single range which ends up consuming an huge number of entries in TCAM. We refer to this method as *Full Expansion* in this paper.

In order to solve this problem, prefix expansion and the implementation of new hardware has been proposed in other papers [2–4, 7]. In the following section, we introduce the existing methods and describe their algorithms.

## 2.1 Prefix Expansion

In order to express several port numbers in a single entry, there is a method of writing least significant bits as $\star$ using TCAM's characteristics, which makes it possible to represent the subranges in units of $2^i$. The range `1024-65535` would then be represented by the following 6 lines. We refer to this as prefix expansion.

```
1***************  32768-65535
01**************  16384-32767
001*************  8192-16383
0001************  4096-8191
00001***********  2048-4095
000001**********  1024-2047
```

Compared to the full expansion, the prefix expansion method can significantly reduce the number of TCAM entries needed, but it still requires several entries for an ACL. Therefore, past research has suggested algorithms for further reducing the required lines.

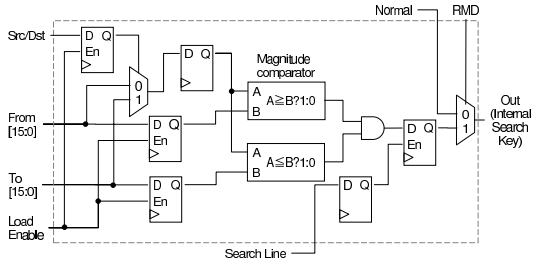
In [3], the *Dynamic Range Encoding Scheme* is suggested which encodes the subset of the ranges in ACL, and maps it to unused bits in each entry of TCAM. Each encoded range affects other ranges which further reduces entries; the entry expansion ratio of 1.23 can be achieved whereas average full expansion ratio is approximately 6.20. However, the proposed algorithm uses TCAM resources when encoding the range and ends up accessing TCAM more frequently when there are more ranges to encode. Also, it needs an additional memory for the mapping purpose which can result in an increased search time.

In [4], the subranges in the ACL are processed to be in powers of 2 by under the premise that the range ends up being semantically the same as before. The total numbers of required entries decreases by 50%. However, it is unknown whether the algorithm is applicable to real world databases which do not normally have interdepending ranges. Also, it has a potential problem upon ACL updates.

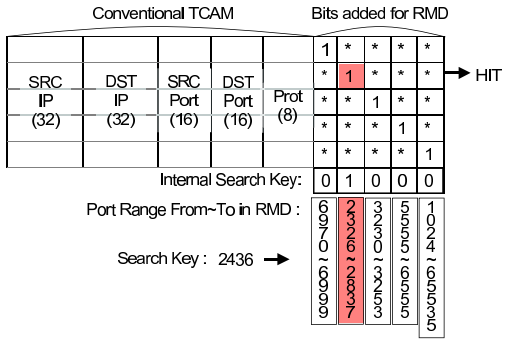
In [5], the ranges are encoded using ternary values, but the major difference from the two papers mentioned above is that it places don't care bits at arbitrary places instead of in a prefix form. The result also uses unused bits in each entry. Using 32 bits for this encoding, the suggested algorithm can reduce the necessary entries up to 50%. However, it is also mentioned in the paper that the optimal encoding method depends on the ACL's content which cannot be known a priori.

**Table 1. Example of Access Control List**

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
access-list 102 deny tcp any range 137 139 any
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 111 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 191 permit udp any any range 16384 16483
```




(a) Example of Range Matching Device




(b) Bits for Range Matching Device in TCAM

**Figure 1. Example of Range Matching Device with TCAM**

## 2.2 Range Specification using Logical Operation Unit

Cisco's IP routers use a device called L4Op (Layer-4 Operation Unit) to write port numbers in ranges in order to make use of the limited TCAM resources [9]. L4Op is a hardware device which resides outside of the TCAM that makes it possible for the ranges to be logically compared using the operators *gt (greater than)*, *lt (less than)*, *range*, and *neq (not equal)* and determines if the input port number matches the specified condition. Whether it satisfies the given condition or not, is determined first by matching every other field except for the port ranges. Only after a positive result, does it begin to search if logical terms of port ranges also match. By this method, port fields in ranges are only managed under L4Op which is independent of the TCAM itself. As a result, the possibility of expressing a rule in a single TCAM entry has the effect of writing more rules to TCAM in the end.

However, the biggest problem of this technique is that L4Op is implemented outside of the TCAM circuit. Because the input data format for the L4Op is entirely different from that of TCAM's, the I/O (Input/Output) band consumption ends up being twice as much, trying to represent the same rate when there is a search from the outside to TCAM. In other words, the double amount of I/O pin numbers is needed under the same frequency. When the physical size of the device is the same, increasing the I/O pin numbers for external I/O causes the relative pin space to be narrow and ends up requiring a higher precision of wired connections. Also, the signal integrity and the SSO (Simultaneous Signal Output) noise become important factors in the recent high speed hardware devices which creates restrictions in designing boards in order to reduce the interference noise between the pins. From the above, it is desirable to have a smaller number of pins if the hardware chip performance should remain the same.

# 3 TCAM Hardware Implementing Range Specifications

As previously mentioned in Section 2, reducing the number of entries by prefix expansion and L4Op is difficult to optimize and hard to update. Numerous studies so far use an existing TCAM device and implement additional software or external devices to minimize the number of required entries. In this paper, we aim to achieve the task of supporting port numbers in ranges by extending the TCAM device.

As the extension to the TCAM, we analyze the effect of adding a range determining circuit and logical operator between the entries which is one of the main contributions in this paper. The following contains a detailed description of each method.

## 3.1 Range Matching Device

Figure 1(a) shows the logic circuit diagram of the proposed RMD. It contains *Normal*, *RMD*, *Src/Dst*, *From*, *To*, *Load Enable* and *Search Line* as input and *Out* as an output. *Normal* and *RMD* are used when the user decides whether to use this device in TCAM as a normal memory or as a storage for a range. In order to make a RMD contain a range,

there has to be a positive signal to *Load Enable* and *Src/Dst* which determines if this is set for the source or destination port number. At the same time, each *From* and *To* line gets input of 16 bits which represents the range.

When the RMD is operated as a search device, either source or destination port is selected based on the input information above and is compared with the stored range. Only when the input is equal to or above *From*, and also equal to or below than *To* does it returns the result of *Out* as 1 (otherwise 0) which becomes the internal search key in Figure 1(b). In TCAM, separate bits for the RMD have to be assigned in addition to the other data bits such as source/destination IP address, source/destination port, and the protocol. A bit is assigned for each RMD. The bit, corresponding to the RMD which contains the desired range, is set to 1 and rest of the bits for other RMD are set to $\star$. Figure 1(b) shows an example of a simple TCAM structure. Let us assume that we have five kinds of ranges that are already stored in the RMD and also assume the second entry uses the range of `2326-2837`. According to the above RMD bit assigning rule, the RMD part within the TCAM becomes $\star 1 \star\star\star$. When there is a search key port number 2436, it returns the output of `01000` (Internal Search Key) and one can easily see that the second entry's result is 1, meaning a match. This packet will be either denied or permitted, depending on the specified action in the ACL rule.

### 3.2 Adding Logical Operational Gates between the Entries

Since TCAM is considered to have performed a successful search when it returns at least one entry that satisfies every field, we can consider a situation where each entries' result performs a logical OR action with the other entries' result. In this paper, in addition to the conventional PE-OR method, we analyze the performance of extending the TCAM feature by adding NOT and AND gates. Figure 2 shows a modified TCAM circuit which has NOT/AND gates. NOT gates are first implemented in the left most part which is followed by AND gates. Since it is impossible to perform a logical AND operation in the hardware only when it is needed, we need sets of ANDs (2, 4, 8 in this case), to be already embedded in the hardware from the beginning. Depending on how many entries we need to represent each subrange that use AND sets, we can pick the smallest AND set needed and write any rules exceeding 8 lines in a regular PE-OR form.

## 4 Management of Range Matching Device

In this section we propose a policy for storing ranges in RMD and clarify the three PE algorithm used in this paper.

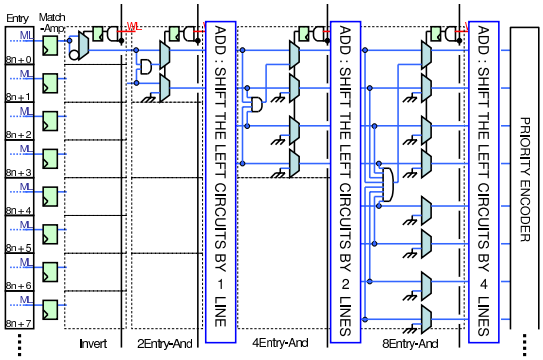



**Figure 2. Additional NOT/AND Operation Gates in TCAM Array**

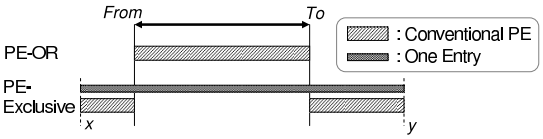



**Figure 3. Prefix Expansion Algorithms**

### 4.1 RMD Policy

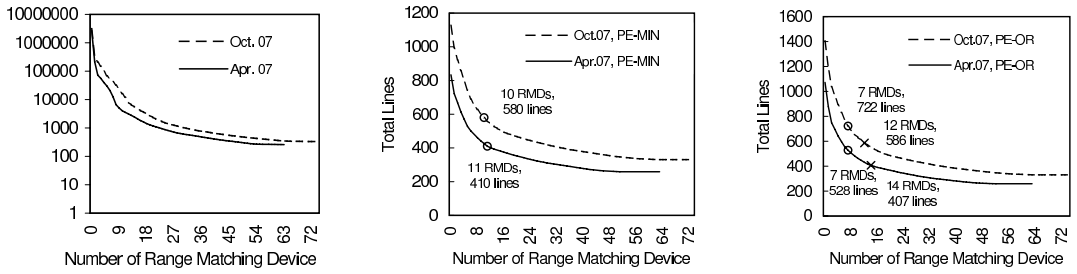
RMDs are limited resources and should be used valuably since they can determine the TCAM performance specification. Therefore, in order to minimize the TCAM entries, it is desirable to give a higher priority to the range which has the highest entry reduction ratio, when being written to the RMD. In this paper, the weight of each range which determines the rank to be written in the RMD is calculated as *(Lines after PE - 1)* multiplied by the *(Number of ACLs referring this range)*.

In other words, a range has a higher tendency to be written in the RMD when it expands to more lines, or has a higher portion in the ACL data base, compared to the other ranges.

### 4.2 Optimization of the Prefix Expansion

In this section, we present an optimized prefix expansion scheme using NOT/AND/OR operation between entries in the TCAM with the proposed hardware structure in Section 3.2. Here we briefly explain three prefix expansion algorithms: PE-OR, PE-Exclusive, and PE-MIN. Figure 3 shows a simple representation of the first two algorithms.

PE-OR is the conventional prefix expansion method which represents the range in units of powers of 2. Especially it shows an outstanding performance when the range is from $2^i$ to $2^{i+1} - 1$ which only consumes a single entry.




(a) Full Expansion with RMD (b) PE-MIN with RMD (c) PE-OR with RMD

**Figure 4. Comparisons of Total Required Entries (w/ or w/o PE combined with RMD)**

However, in the case of the range `16385-65534`, it ends up being 29 lines which is the worst possible case.

PE-Exclusive first finds the minimum range of $[2^i, 2^{i+1} - 1]$ ($x$ and $y$ in Figure 3 respectively) which entirely covers the given range, and expresses the unnecessary parts in combinations of powers of 2. This method is convenient in expressing ranges which are not suitable for PE-OR. For example, the above range of `16385-65534` can be expressed by only 3 lines: `16384-65535 AND (NOT 16384) AND (NOT 65535)`.

PE-MIN is the most optimized entry reduction algorithm by finding the appropriate ranges in a brute force method. Below is the description of the algorithm.

1. As a first step, select the largest $\alpha$ in $k2^{\alpha}(1 \leq \alpha \leq 15, k = constant)$ which resides within the range $[From, To]$ and define that as a *Reference Point* $b$. Furthermore, divide the range in to two parts $[From, b]$ and $[b, To]$

2. Let $D = To - b$. For the $[b, To]$, find the largest value of $i(i < log_2 y)$ which satisfies $2^i < D \leq 2^{(i+1)}$

3. For the two ranges of $[2^i, D]$ and $[D, 2^{(i+1)}]$, recursively determine how many entries are needed for each range, $x_l, x_r$, respectively

4. If $x_l < x_r$, add a prefix of $[b, 2^i]$, or else add $[b, 2^{(i+1)}]$

5. Repeat steps 2–4 for the range $[From, b]$

# 5 Evaluation

In this paper, we analyze how many entries are required in writing the port numbers expressed in ranges and discuss its result, using a real-life database. Table 2 shows the information about the two campus level ACL sets used in this analysis which were obtained in April and October 2007.

**Table 2. Two ACL Sets for Evaluation**

| Date Captured | Apr. 07 | | Oct. 07 | |
|---|---|---|---|---|
| # of unique ACL entries | 6,440 | | 7,202 | |
| | Src | Dest | Src | Dest |
| # of ranges | 3 | 256 | 6 | 325 |
| # of unique ranges | 63 | | 74 | |

## 5.1 Entry Reduction using Range Matching Devices

Figure 4 shows how the required entries can be reduced when RMDs are used. Figure 4(a) shows how writing every single number (Full Expansion) that are represented in ranges can result in consuming an enormous amount of TCAM resources. From this figure, we can easily see that we need an order of a million when there are no RMDs. The reason for this phenomenon is that the range `1024-65535` which are registered/dynamic ports can be seen quite frequently which expands to 64,512 lines. By only using one RMD, this range gets written in the memory with the highest priority which results in one tenth of the total required number of lines.

Next, Figure 4(b) shows the effect of entry reduction when RMDs are used with prefix expansion. We calculate the weight proposed in the Section 4.1 and write ranges with larger weight first to the RMD. As a result, we can see that it is possible to reduce a large number of entries by putting them in to RMDs. Especially, the reduction rate is high when a small number of RMDs are implemented and in the case of using PE-MIN, when there are ten and eleven RMDs in the Oct. 07 and Apr. 07 data, respectively, the total required ranges to write can be reduced as much as 50% from that of not using any RMDs at all. In addition, seven RMDs can achieve the same effect in the PE-OR case (see Figure 4(c)).

Furthermore, when we compare the PE-MIN and PE-OR schemes, the required total entries decrease about 25% by

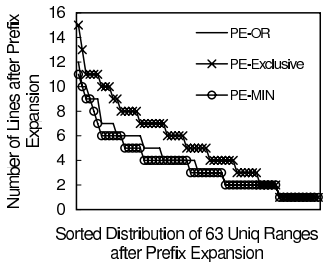



**Figure 5. Comparisons of Prefix Expansion Algorithms**

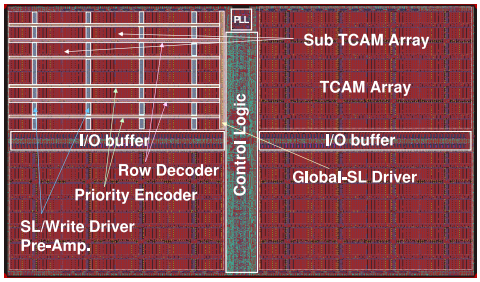
adding AND/NOT logical gates to the TCAM. When PE-OR tries to accomplish the same level as that of PE-MIN, two to three further RMDs are needed.

Then, the question arises: which is a better choice? From only the number of required TCAM entries point of view, PE-MIN is a definite choice since it can minimize the vertical lines. However, Adding NOT/AND gates to fully support PE-MIN might end up increasing the horizontal bits (bits added for RMD, $x$ in Figure 7(a)) in the TCAM which are not a highly desirable choice since determining the optimum AND set kinds might be different for each ACLs. Choosing PE-OR might seem like an appealing choice since it does not require any additional logical operation gates. The problem of using only PE-OR though, is that this method is weak for some malicious worst case tests, like setting worst case ranges in each source and destination port numbers to create $29 \times 29 = 841$ lines for a single rule. Unless this is the case, using PE-OR might be suitable enough.

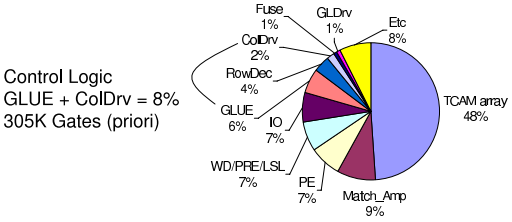
One of the biggest advantages of adding logical NOT/AND gates in addition to minimizing the total entries required in storing the ACL, is that it makes it possible for the "except" rules of the ACL to be stored directly in the TCAM, whereas the conventional methods such as using Cisco IOS or Juniper Junos chose to use the software.

## 5.2 Comparison of the Three Prefix Expansion Method

In order to find out the effect of the different prefix expansion algorithms, we analyzed the distribution of expanded lines in each method. Figure 5 shows how each algorithm expands the range. The total lines can be reduced to approximately 25% using the PE-MIN compared to PE-OR. Comparing the entry reduction itself, PE-MIN uses the least amount of TCAM resource but it requires a built-in logical NOT and AND gates in the TCAM itself.




(a) TCAM VLSI in 90nm Technology




580 Gates x 20 RMDs = 11.6 K Gates
Current TCAM : TCAM with RMD = 100 : 100.3

(b) Cost compared to the existing chip

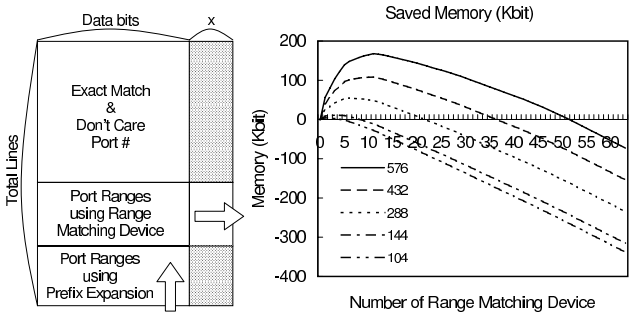
**Figure 6. Existing TCAM and Cost Comparison with Proposed Hardware**

## 5.3 Overhead Cost Estimation of Adding Range Matching Devices

Finally, we analyze the effect of increase in the hardware cost caused by adding RMDs. Figure 6(a) shows the TCAM hardware structure using 90 nm technology. The ratio of the components in this chip are shown in Figure 6(b). Among all of the components, 8% is *Control Logic* which is known to be approximately 305 K gates. Meanwhile, gate calculation of a single RMD is 580 gates. When inserting 20 RMDs in TCAM, this results in around 11.6 K gates. This corresponds in 0.3% of the total TCAM gates. Therefore, with the 0.3% increase in manufacturing cost, it is possible to significantly reduce the required number of TCAM lines to write port numbers which results in more ACL storage space for the limited TCAM resource. We also point out that the component ratio would stay about roughly the same even if the chip technology advances to 65 nm.

Also as we showed in Figure 7(a), when one more RMD is added, consequently one more TCAM bit is needed to represent the result of the RMD part. RMD gives us the effect of reducing the vertical lines in TCAM, but results in increasing the horizontal lines. Therefore, it is necessary to consider the trade-off of both factors in order to decided the optimal number of RMDs for implementation in a TCAM.

For example, we have data which has a bit length of $x_0$ to write in the TCAM when there are no RMDs, and let $y_0$ be




(a) Reduced and Increased Memory Space

(b) Saved Memory Space

**Figure 7. Wasted and Saved Memory when Range Matching Devices are used**

the required number of entries to write prefix expansion-ed ACL. In this case, the total required bits $C_0$ in TCAM can be represented as $C_0 = x_0 y_0$. Let us say that when 1 RMD is added in a situation where $i$ RMDs are being used, $d_{i+1}$ lines can be reduced. The total necessary bit after adding the RMD in TCAM can be now expressed as the Equation 1.

$$C_{i+1} = x_{i+1} y_{i+1} = (x_i + 1)(y_i - d_{i+1}) \quad (1)$$

Therefore, the change $\Delta_{i+1}$ in total bits by adding an RMD can be expressed as,

$$\Delta_{i+1} = x_{i+1} y_{i+1} - x_i y_i = y_i - (x_i + 1) d_{i+1} \quad (2)$$

In order to reduce the total bits after adding the RMD, $\Delta_{i+1}$ has to be negative. As a result we get the following relation: $d_{i+1} > y_i/(x_i + 1)$.

Figure 7(b) shows the total saved memory as the RMDs are added. Five kinds of data bits are considered: 104, 144, 288, 432, and 576 (104 bits for the five tuple and others for a multiplication of 72 which is a common unit of TCAM cells in current technology). From this result, it can be found that memory space is saved in the beginning when the RMDs are first added but after a certain number of RMDs, the overhead increases and it is no longer possible to gain any further benefits. Also, when the data bits are shorter, the wasted memory increases faster as the RMDs are added. It is considered that the ratio of port numbers expressed in ranges in an ACL affects the optimal number of RMDs required in order to make the most out of its advantages.

## 6 Conclusion and Future Work

In this paper, we proposed a new TCAM architecture combining prefix expansion and range matching devices in order to reduce the number of lines that are required in TCAM to represent port numbers in ranges. Also, we proposed a method how to express ranges and proposed an optimal prefix expansion method. By using a real-life ACL database, we have also shown how the proposed architecture can effectively manage the ACL compared to the conventional method.

Future studies will be on the analysis of other ACLs using the proposed method in order to achieve a general-purpose TCAM architecture. Also, it is required to have the new TCAM implemented in the network processor to investigate further performance characteristics such as power consumption.

## Acknowledgment

We would like to thank Associate Professor of Osaka University, Go Hasegawa, for the valuable advice.

## References

[1] B. Agrawal and T. Sherwood. Modeling TCAM Power for Next Generation Network Devices. *IEEE ISPASS*, 2006.

[2] S. Ahmad and R. Mahapatra. An Efficient Approach to On-Chip Logic Minimization. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 15(9):1040–1050, 2007.

[3] H. Che, Z. Wang, K. Zheng, and B. Liu. DRES: Dynamic Range Encoding Scheme for TCAM Coprocessors. *technique report, Univ. of Texas at Arlington, http://crystal. uta. edu/hche/dres. pdf*, 2006.

[4] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla. Packet Classifiers In Ternary CAMs Can Be Smaller. In *ACM SIGMETRICS '06/Performance '06*, pages 311–322, New York, NY, USA, 2006. ACM Press.

[5] K. Lakshminarayanan, Rangarajan, and S. Venkatachary. Algorithms for Advanced Packet Classification with Ternary CAMs. In *ACM SIGCOMM '05*, pages 193–204, New York, NY, USA, 2005. ACM Press.

[6] H. Liu. Reducing Routing Table Size Using Ternary-CAM. *Hot Interconnects*, 9(2001):22–24, 2001.

[7] H. Liu. Efficient Mapping of Range Classifier into Ternary-CAM. In *HOTI '02*, page 95, Washington, DC, USA, 2002. IEEE Computer Society.

[8] A. J. McAuley and P. Francis. Fast Routing Table Lookup Using CAMs. In *IEEE INFOCOM*, pages 1382–1391, 1993.

[9] M. A. Ross, S.-D. Chen, and A. V. Bechtolsheim. Logical Operation Unit for Packet Processing, US Patent 6,658,002, 2003.

[10] S. Sharma and R. Panigrahy. Sorting and Searching Using Ternary CAMs. In *HOTI '02*, page 101, Washington, DC, USA, 2002. IEEE Computer Society.

[11] F. Zane, G. Narlikar, and A. Basu. CoolCAM: Power-Efficient TCAMs for Forwarding Engines. In *IEEE INFOCOM*, 2003.