On the Number of Products to Represent Interval Functions by SOPs with Four-Valued Variables

Tsutomu Sasao
Department of Computer Science and Electronics,
Kyushu Institute of Technology,
Iizuka 820-8502, Japan

***Abstract*—Let A and B be integers such that $A \leq B$. An n-variable interval function is a mapping $IN[n : A, B] : \{0,1\}^n \rightarrow \{0,1\}$, where $IN[n : A, B](X) = 1$ iff $A \leq X \leq B$. Such function is useful for packet classification in the internet, network intrusion detection system, etc. This paper considers the number of products to represent interval functions by sum-of-products expressions with two-valued and four-valued variables. It shows that to represent any interval function of n variables, an SOP with two-valued variables requires up to $2(n-2)$ products, while an SOP with four-valued variables requires at most $n-1$ products. These bounds are useful to estimate the size of a content addressable memory (CAM).**

I. Introduction

A classification function [13] is used for packet classification in the internet [4], where internet service providers (ISPs) provide differentiated services to various users. Classification functions are also used for network intrusion detection system (IDS). Since high-speed processing is necessary, various hardware implementations have been proposed [2], [7], [10], [15], [17].

In this paper, we derive upper bounds on the number of products in sum-of-products expressions (SOPs) with 2-valued and 4-valued variables to represent interval functions. With these bound, we can estimate the size of a circuit for the packet classification. As for hardware to implement classification functions, a Ternary Content Addressable Memory (TCAM) [10] and a CAM with 2-bit encoding [5], [16] are used. Each product in an SOP corresponds to a word in a CAM. A classification function is defined by a set of rules, where each rule is a conjunction of interval functions. For example, consider the classification function

$$f : \{0,1,2,3\} \times \{0,1,2,3\} \rightarrow \{0,1,2\}$$

consisting of two rules:

$$R_1 = (0 \leq X_1 \leq 2) \cdot (1 \leq X_2 \leq 3),$$

and

$$R_2 = (1 \leq X_1 \leq 3) \cdot (0 \leq X_2 \leq 2).$$

In this case, $(0 \leq X_i \leq 2)$ and $(1 \leq X_i \leq 3)$ are interval functions. $f = 1$ if R_1 holds, $f = 2$ if R_1 does not hold, but R_2 holds, and $f = 0$ if neither R_1 nor R_2 holds.

In this example, we use an SOP with 2-valued variables to represent an interval function. The rules are stored in the TCAM array in the order of decreasing priority. Let $X_1 = 2x_1 + x_2$ and $X_2 = 2x_3 + x_4$, where $+$ denotes an integer addition. Then, the interval function $(0 \leq X_1 \leq 2)$ is represented by the SOP: $\bar{x}_1 \vee \bar{x}_2$. In a similar way, the interval function $(1 \leq X_1 \leq 3)$ is represented by the SOP: $x_1 \vee x_2$. In this example, rules have two *fields*: X_1 and X_2, and each field of a rule is represented by an SOP with two products. For example, R_1 can be represented by $R_1 = (\bar{x}_1 \vee \bar{x}_2)(x_3 \vee x_4) = \bar{x}_1 x_3 \vee \bar{x}_1 x_4 \vee \bar{x}_2 x_3 \vee \bar{x}_2 x_4$. Thus, the number of products to represent R_1 by an SOP with 2-valued variables is $2 \times 2 = 4$. Fig. 1.1shows a realization of the example function by TCAM and RAM. The TCAM generates the indexes from 1 to 8, that correspond to the first word to the 8-th word. On the other hand, the RAM converts the indexes into the values of classification function, that are $\{0,1,2\}$. In this example, the upper four words correspond to the product terms for $R1$, and the lower four words correspond to the product terms for $R2$. When neither of $R1$ and $R2$ do not match, then the TCAM and the RAM generate 0.

$R1 = \bar{x}_1 x_3 \vee \bar{x}_1 x_4 \vee \bar{x}_2 x_3 \vee \bar{x}_2 x_4$

$R2 = x_1 \bar{x}_3 \vee x_1 \bar{x}_4 \vee x_2 \bar{x}_3 \vee x_2 \bar{x}_4$

TCAM					RAM	
					Input	Output
x1	x2	x3	x4	Index	0	0
0	*	1	*	1	1	1
0	*	*	1	2	2	1
*	0	1	*	3	3	1
*	0	*	1	4	4	1
1	*	0	*	5	5	2
1	*	*	0	6	6	2
*	1	0	*	7	7	2
*	1	*	0	8	8	2

Fig. 1.1. Realization of a classification function by TCAM and RAM.

In the real packet classification, the numbers of values of the variable are either 2^{32}, 2^{16} or 2^8, and the numbers of variables are 5 to 8. So, the size of the SOP can be very large. This is the reason why we are interested in the number of products in SOPs for interval functions and classification functions. Simplification of a set of rules of a classification function can be done by minimizing SOPs [8].

A direct method to represent an interval $[A, B]$ is to store the pair of integers. However, this method requires a comparator of values in each field [15], and conventional memory cannot

be used. Another method is to use a Look-Up-Table (LUT) for each field [7]. However, this can be expensive when the number of bits in a field is large. In many cases, we have to update rules of the classification functions frequently. Due to this, CAMs are often used in the network applications.

This paper gives tight upper bounds on the numbers of products in SOPs with 2-valued variables and 4-valued variables for interval functions. Note that a CAM word corresponds to a product in an SOP.

This paper is organized as follows: Section 2 defines interval functions, and shows their properties. Section 3 considers the number of products in SOPs with 2-valued variables to represent an interval function. Section 4 considers the number of products in SOPs with 4-valued variables to represent an interval function. Section 5 reviews TCAM and CAM with 2-bit encoding. Section 6 shows experimental results. And, finally Section 7 concludes the paper.

II. Interval Functions

An interval function is a generalization of a comparator function. To define an interval function, we use a *Greater-than-or-Equal-to function* and a *Less-than-or-Equal-to function* [13].

Definition 2.1: An n-input **GE function** (Greater-than-or-Equal-to function) is

$$GE(n:A) = \begin{cases} 1 & \text{if } X \geq A \\ 0 & \text{otherwise} \end{cases}$$

where $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, $\vec{x} = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ is a binary input vector, X is an integer represented by $\vec{x}$, and A is an integer such that $0 \leq A \leq 2^n - 1$.

Definition 2.2: An n-input **LE function** (Less-than-or-Equal-to function) is

$$LE(n:B) = \begin{cases} 1 & \text{if } X \leq B \\ 0 & \text{otherwise} \end{cases}$$

where X is an integer represented by $\vec{x}$, and B is an integer such that $0 \leq B \leq 2^n - 1$.

Definition 2.3: An n-input **interval function** [1] is

$$IN(n:A,B) = \begin{cases} 1 & \text{if } A \leq X \leq B \\ 0 & \text{otherwise} \end{cases}$$

where X is an integer represented by $\vec{x}$, and A and B are integers such that $0 \leq A \leq B \leq 2^n - 1$.

Example 2.1: Consider the case of $n = 4$, $A = 1$, and $B = 14$. Figs. 2.1 and 2.2 show maps for $GE(4:1)$, $LE(4:14)$, and $IN(4:1,14)$. In these maps, integers X that satisfy the relations are shown, where $X = 8x_3 + 4x_2 + 2x_1 + x_0$. ■

From the definitions, we have the following:

Lemma 2.1:

$$IN(n:A,B) = GE(n:A) \cdot LE(n:B).$$

Lemma 2.2: Let $N(n)$ be the number of distinct interval functions $IN(n:A,B)$, where $1 \leq A < B \leq 2^n - 1$. Then,

$$N(n) = (2^n - 1) \cdot (2^{n-1} - 1).$$

[1] An interval function was called a range function in [13].

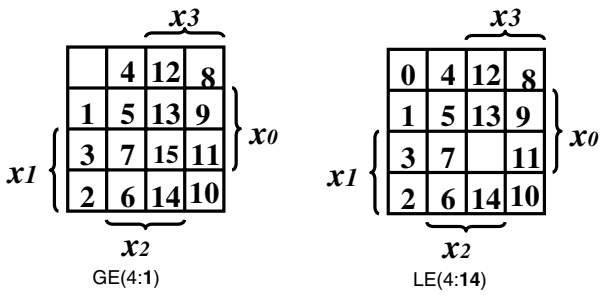

Fig. 2.1. Maps for GE(4:1) and LE(4:14).

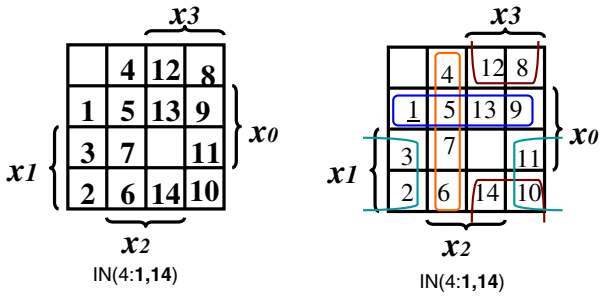

Fig. 2.2. Maps for IN(4:1,14).

Table 2.1 shows the numbers of interval functions of n variables for $4 \leq n \leq 12$.

III. Sizes of SOPs with 2-Valued Variables

In this part, we consider sum-of-products expressions with 2-valued variables to represent interval functions [13].

Definition 3.1: (Two-valued case) Let x be a variable that may take one of two values in $\{0, 1\}$. Then, $\bar{x}$ is a complement of the variable. x and $\bar{x}$ are **literals** of a variable x. The AND of literals is a **product**. A **minterm** is a logical product of n literals where each variable occurs as exactly one literal. The OR of products is a **sum-of-products expression** (**SOP**). Let two functions be f and g. f **implies** g if every $\vec{x}$ satisfying $f(\vec{x}) = 1$ also satisfies $g(\vec{x}) = 1$. A minterm that implies f is a **minterm of** f. A **prime implicant** (**PI**) of a function f is a product that implies f, such that the deletion of any literal from the product results in a new product that does not imply f. An **irredundant sum-of-products expression** (**ISOP**) is an SOP, where each product is a PI, and no PI can be deleted without changing the function represented by the

TABLE 2.1
Numbers of Interval Functions.

n	Number
4	105
6	1,953
8	32,385
10	522,753
12	8,382,465

expression. The **size** of an SOP is the number of PIs in the SOP. Among the ISOPs for f, the ISOP with the minimum size is a **minimum SOP** (**MSOP**). The size of a MSOP with 2-valued variables for function f is denoted as $\tau_2(f)$.

For a given n, we are interested in the most complicated interval function. That is, the function with the largest $\tau_2(IN(n : A, B))$.

Definition 3.2: $\mu_2(n)$ denotes the number of products in an MSOP with 2-valued variables for the n-variable interval function with the largest number of products.

By exhaustive examination, we have the following:

Theorem 3.1: $\mu_2(n) = n,\ (n = 1, 2, 3, 4)$.

Example 3.1: In SOPs with 2-valued variables, the most complicated interval functions up to $n = 4$ include:

$$\begin{aligned} IN(1:1,1) &= x_0, \\ IN(2:1,2) &= x_1\bar{x}_0 \vee \bar{x}_1 x_0, \\ IN(3:1,7) &= x_2 \vee x_1 \vee x_0, \\ IN(3:1,6) &= \bar{x}_2 x_1 \vee \bar{x}_1 x_0 \vee \bar{x}_0 x_2, \\ IN(4:1,15) &= x_3 \vee x_2 \vee x_1 \vee x_0, \\ IN(4:5,10) &= \bar{x}_3 x_2 x_0 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_0 \vee x_3 \bar{x}_2 \bar{x}_1. \end{aligned}$$

■

One might conjecture that Theorem 3.1 is true for larger n, but it is not. As will be shown, $\mu_2(n) = 2(n-2)$ for $n \geq 5$ (Theorem 3.2).

Lemma 3.1: $GE(n : A)$ can be represented by an SOP with 2-valued variables having at most $1 + \sum_{i=1}^{n-1} \bar{a}_i$ disjoint products, where $\vec{a} = (a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$ is the binary representation of A.

Lemma 3.2: $LE(n : B)$ can be represented by an SOP with 2-valued variables having at most $1 + \sum_{i=1}^{n-1} b_i$ disjoint products, where $\vec{b} = (b_{n-1}, b_{n-2}, \ldots, b_1, b_0)$ is the binary representation of B.

Lemma 3.3: For $n \geq 4$, $IN(n : A, B)$ can be represented by an SOP with 2-valued variables having at most $2(n-2)$ products.

Up to here, we have shown that any interval function can be represented by $2(n-2)$ products, when $n \geq 5$. Similar results were obtained in [14], independently. From here, we are going to show that there exist interval functions that require $2(n-2)$ products, when $n \geq 5$. To show the lower bound, we use an idea of independent sets of minterms [12].

Definition 3.3: A set of minterms MI for f is **independent** if no implicant of f contains a pair of minterms in MI.

Lemma 3.4: Let MI be an independent set of minterms for f. Then, any SOP for f requires at least $|MI|$ products.

Lemma 3.5: $\tau_2(IN(n : 2^{n-3}+1, 7\cdot 2^{n-3}-2)) = 2(n-2)$.

Lemma 3.6: $\tau_2(IN(n : 2^{n-2}+1, 3\cdot 2^{n-2}-2)) = 2(n-2)$.

From Lemmas 3.3, 3.5 and 3.6, we have:

Theorem 3.2: $\mu_2(n) = 2(n-2)$, where $n \geq 5$.

Furthermore, we have the following:

Conjecture 3.1: For $n \geq 5$, among the interval functions, only

$$IN(n : 2^{n-3}+1, 7\cdot 2^{n-3}-2) \ and$$
$$IN(n : 2^{n-2}+1, 3\cdot 2^{n-2}-2)$$

require $2(n-2)$ products, and other interval functions require fewer products.

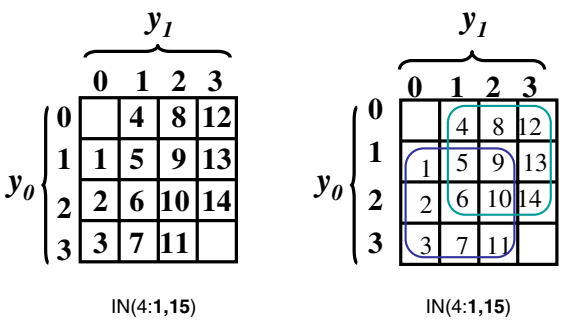

Fig. 4.1. Map for IN(4:1,14):4-valued variables

IV. Sizes of SOPs with 4-Valued Variables

Let $X = \sum_{i=0}^{n-1} x_i 2^i$. If n is an even number, i.e. n is written as $n = 2r$, where r is an integer, then X can be represented by the base-4 instead of the base-2. That is, X is represented by $\vec{y} = (y_{r-1}, y_{r-1}, \ldots, y_1, y_0)$, where $X = \Sigma_{i=0}^{r-1} y_i 4^i$ and $y_i \in \{0, 1, 2, 3\}$. In this case, SOPs with 4-valued variables can be used to represent interval functions as follows:

Definition 4.1: (Four-valued case) Let y_i may take one of 4 values in $\{0, 1, 2, 3\}$. $y_i^{S_i}$ is a **literal** of a variable y_i, where $S_i \subseteq \{0, 1, 2, 3\}$. $y_i^{S_i} = 1$ if $y_i \in S_i$ and $y_i^{S_i} = 0$ otherwise. The AND of literals is a **product**. A **minterm** is logical product of n literals where $|S_i| = 1$ for all i. The **size** of an SOP is the number of PIs in the SOP. The size of a minimum SOP with 4-valued variables for function f is denoted as $\tau_4(f)$.

The following lemma is well known [11].

Lemma 4.1: For any $n = 2r$ variable logic function f,

$$\tau_4(f) \leq \tau_2(f).$$

Example 4.1: Consider $IN(4 : 1, 14)$. The map in the left of Fig. 4.1 shows the interval function using 4-valued variables. In this map, integers X that satisfy the relation are shown, where $X = 4y_1 + y_0$. In this case, only two products are necessary to represent the function, as shown in the map in the right. Let $y_1 = 2x_3 + x_2$ and $y_0 = 2x_1 + x_0$. Then, the SOP with 4-valued variables is

$$IN(4:1,14) = y_1^{\{0,1,2\}} \cdot y_0^{\{1,2,3\}} \vee y_1^{\{1,2,3\}} \cdot y_0^{\{0,1,2\}}.$$

Note that an SOP with two-valued variables requires four products to represent the same function as shown in Fig. 2.2.

■

From here, we are going to derive the value of $\tau_4(f)$ for an interval function f.

Lemma 4.2: $GE(n:A)$ can be represented by an SOP with 4-valued variables having at most $1+\sum_{i=1}^{r-1}(\alpha_i \neq 3)$ disjoint products, where $\vec{\alpha} = (\alpha_{r-1}, \alpha_{r-2}, \ldots, \alpha_1, \alpha_0)$ is the base-4 representation of A, and $n = 2r$.

(Proof) $GE(n:A)$ can be represented as:

$GE(n:A) = (y_{r-1} > \alpha_{r-1}) \vee (y_{r-1} \equiv \alpha_{r-1})(y_{r-2} > \alpha_{r-2}) \vee (y_{r-1} \equiv \alpha_{r-1})(y_{r-2} \equiv \alpha_{r-2})(y_{r-3} > \alpha_{r-3}) \vee (y_{r-1} \equiv \alpha_{r-1})(y_{r-2} \equiv \alpha_{r-2})(y_{r-3} \equiv \alpha_{n-3})(y_{n-4} > \alpha_{n-4}) \vee \cdots \vee (y_{r-1} \equiv \alpha_{r-1})(y_{n-2} \equiv \alpha_{r-2})(y_{r-3} \equiv \alpha_{r-3}) \cdots (y_1 \equiv \alpha_1)(y_0 \geq \alpha_0)$, where $A = \sum_{i=0}^{r-1} \alpha_i 4^i$. Here, the symbol $\equiv$ denotes the equivalence operator. Note that the number of products in the SOP is at most $1+\sum_{i=1}^{r-1}(\alpha_i \neq 3)$. □

Lemma 4.3: $LE(n:B)$ can be represented by an SOP with 4-valued variables having at most $1+\sum_{i=1}^{r-1}(\beta_i \neq 0)$ disjoint products, where $\vec{\beta} = (\beta_{r-1}, \beta_{r-2}, \ldots, \beta_1, \beta_0)$ is the base-4 representation of B, and $n = 2r$.

(Proof) $LE(n:B)$ can be represented as

$LE(n,B) = (y_{r-1} < \beta_{r-1}) \vee (y_{r-1} \equiv \beta_{r-1})(y_{r-2} < \beta_{n-2}) \vee (y_{r-1} \equiv \beta_{r-1})(y_{n-2} \equiv \beta_{n-2})(y_{n-3} < \beta_{n-3}) \vee (y_{r-1} \equiv \beta_{r-1})(y_{n-2} \equiv \beta_{n-2})(y_{n-3} \equiv \beta_{n-3})(y_{n-4} < \beta_{n-4}) \vee \cdots \vee (y_{r-1} \equiv \beta_{r-1})(y_{n-2} \equiv \beta_{r-2})(y_{r-3} \equiv \beta_{r-3}) \cdots (y_1 \equiv \beta_1)(y_0 \geq \beta_0)$, where $B = \sum_{i=r-1}^{0} \beta_i 4^i$. Note that the number of products in the SOP is at most $1+\sum_{i=1}^{r-1}(\beta_i \neq 0)$. □

Lemma 4.4: $IN(4:A,B)$ can be represented by an SOP with 4-valued variables having at most 3 products.

(Proof) This can be done by exhaustive minimization SOPs for all the 105 interval functions of 4-variables. □

Theorem 4.1: For $n \geq 4$, $IN(n:A,B)$ can be represented by an SOP with 4-valued variables having at most $n-1$ products, where $n = 2r$.

(Proof) We use mathematical induction to prove the theorem. When $n = 4$, by Lemma 4.4 any interval function can be represented with at most three products, and the theorem holds. Assume that the theorem holds for the SOPs with 4-valued variables representing $k(=2t)$-variable interval functions. That is, any interval function with k variables can be represented by an SOP with 4-valued variables having $k-1$ products. Next, consider the case of $k+2$ variables. Let $(\alpha_t, \alpha_{t-1}, \ldots, \alpha_0)$ and $(\beta_t, \beta_{t-1}, \ldots, \beta_0)$ be base-4 representations of A and B, respectively. Since $A \leq B$, we need only to consider the following two cases:

1) When $\alpha_t = \beta_t$.
   $IN(k+2:A,B) = (y_t \equiv \alpha_t)GE(t:A')LE(t:B') = (y_t \equiv \alpha_t)IN(t:A',B')$,
   where $A' = A - \alpha_t 4^t$ and $B' = B - \beta_t 4^t$. In this case, the number of products is at most $k-1$, by the hypothesis of the induction.
2) When $\alpha_t < \beta_t$.
   $IN(k+2:A,B) = [(y_t > \alpha_t) \vee (y_t \equiv \alpha_t)GE(t:A')] \cdot [(y_t < \beta_t) \vee (y_t \equiv \beta_t)LE(t:B')] = (\alpha_t < y_t < \beta_t) \vee (y_t \equiv \alpha_t)GE(t:A') \vee (y_t \equiv \beta_t)LE(t:B')$,
   where $A' = A - \alpha_t 4^t$ and $B' = B - \beta_t 4^t$.
   In this case, the number of products is at most $1+2+\sum_{i=1}^{t-1}((\alpha \neq 3)+(\beta \neq 0)) \leq 1+2t = (k+2)-1$.

TABLE 4.1
INDEPENDENT SET OF VECTORS.

	Vector							
Num	y_{t-1}	y_{t-2}	y_{t-3}	y_{t-4}	y_3	y_2	y_1	y_0
1	0	1	1	1	1	1	1	2
2	0	1	1	1	1	1	2	0
3	0	1	1	1	1	2	0	0
4	0	1	1	1	2	0	0	0
5	0	1	1	2	0	0	0	0
6	0	1	2	0	0	0	0	0
7	0	2	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1
9	2	0	1	1	1	1	1	1
10	2	1	0	1	1	1	1	1
11	2	1	1	0	1	1	1	1
12	2	1	1	1	0	1	1	1
13	2	1	1	1	1	0	1	1
14	2	1	1	1	1	1	0	1
15	2	1	1	1	1	1	1	0

So, the theorem holds for $k+2 = 2(t+1)$-variable interval functions. Thus, by mathematical induction, the theorem holds for all X. Hence, the number of the products in an SOP with 4-valued variables representing an n-variable interval function is at most $n-1$. □

Example 4.2: Consider $GE(6:6)$. In this case, $(\alpha_2, \alpha_1, \alpha_0) = (0,1,2)$.
With arithmetic operators, it is represented as
$GE(6:6) = (y_2 > 0) \vee (y_2 \equiv 0) \cdot (y_1 > 1) \vee (y_2 \equiv 0) \cdot (y_1 \equiv 1) \cdot (y_0 \geq 2)$.
With logical operators, it is represented as
$GE(6:6) = y_2^{\{1,2,3\}} \vee y_2^{\{0\}} y_1^{\{2,3\}} \vee y_2^{\{0\}} y_1^{\{1\}} y_0^{\{2,3\}}$.
Consider $LE(6:36)$. In this case, $(\beta_2, \beta_1, \beta_0) = (2,1,0)$.
With arithmetic operators, it is represented as
$LE(6:6) = (y_2 < 2) \vee (y_2 \equiv 2) \cdot (y_1 < 1) \vee (y_2 \equiv 2) \cdot (y_1 \equiv 1) \cdot (y_0 \leq 0)$.
With logical operators, it is represented as
$LE(6:6) = y_2^{\{3\}} \vee y_2^{\{2\}} y_1^{\{0\}} \vee y_2^{\{2\}} y_1^{\{1\}} y_0^{\{0\}}$.
Consider $IN(6:6,36)$.
With arithmetic operators, it is represented as
$IN(6:6,36) = [(y_2 > 0) \vee (y_2 \equiv 0) \cdot (y_1 > 1) \vee (y_2 \equiv 0) \cdot (y_1 \equiv 1) \cdot (y_0 \geq 2)] \cdot [(y_2 < 2) \vee (y_2 \equiv 2) \cdot (y_1 < 1) \vee (y_2 \equiv 2) \cdot (y_1 \equiv 1) \cdot (y_0 \leq 0)] = (y \equiv 1) \vee (y_2 \equiv 0) \cdot (y_1 > 1) \vee (y_2 \equiv 0) \cdot (y_1 \equiv 1) \cdot (y_0 \geq 2) \vee (y_2 \equiv 2) \cdot (y_1 < 1) \vee (y_2 \equiv 2) \cdot (y_1 \equiv 1) \cdot (y_0 \leq 0)$.
With logical operators, it is represented as
$IN(6:6,36) = y_2^{\{1\}} \vee y_2^{\{0\}} y_1^{\{2,3\}} \vee y_2^{\{0\}} y_1^{\{1\}} y_0^{\{2,3\}} \vee y_2^{\{2\}} y_1^{\{0\}} \vee y_2^{\{2\}} y_1^{\{1\}} y_0^{\{0\}}$. ■

Lemma 4.5: Let $A = \frac{1}{3}(4^{r-1}+2)$ and $B = \frac{4}{3}(7 \cdot 4^{r-2}-1)$. Then, $IN(n:A,B)$ requires at least $n-1$ products in the SOP with 4-valued variables, where $n = 2r$.

(Proof) We show an independent set of $n-1$ vectors. Note that the base-4 representations of A, and B are $\vec{\alpha} = (0,1,1,\ldots,1,2)$ and $\vec{\beta} = (2,1,1,\ldots,1,0)$, respectively.
It is clear that $A < B$. Consider the set of $n-1$ vectors shown in Table 4.1. We can verify that no pair of vectors form an

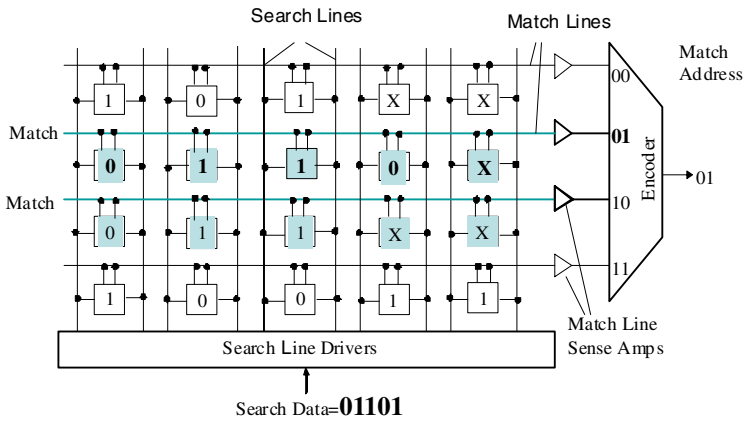

Fig. 5.1. TCAM architecture.

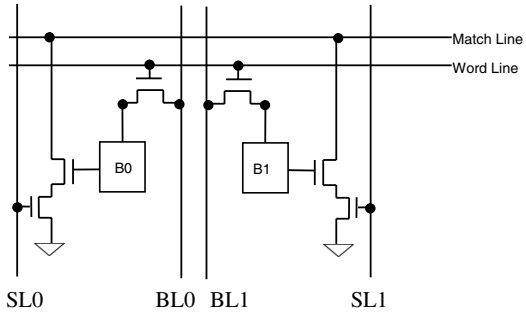

Fig. 5.2. TCAM cell.

implicant of the function. Thus, the function requires at least $n-1$ products in an SOP. □

By Theorem 4.1 and Lemma 4.5, we have the following:

Theorem 4.2: Let $\mu_4(n)$ be the number of products in an MSOP with 4-valued variables for the n-variable interval function with the largest number of products. Then, $\mu_4(n) = n-1$, where $n = 2r$.

V. CAM Hardware

This section reviews operations of CAMs. Each word in a TCAM corresponds to a products in an SOP with two-valued variables. On the other hand, each word in a CAM with 2-bit encoding corresponds to a products in an SOP with four-valued variables.

A. TCAM [10], [1]

A Ternary Content Addressable Memory (TCAM) can treat three states 0,1, and *don't care*. In spite of the term *ternary*, a TCAM performs purely *binary* operations. Fig. 5.1 illustrates an architecture of a TCAM. It operates as follows:

1) During a search operation, all the bits in the TCAM cells are compared with the data in the search lines.
2) After a search operation, matched lines are set to high, while mismatched lines are set to low.
3) The priority encoder produces the address of the first matched line.

Fig. 5.2 shows the detail of TCAM cell. A logical zero is stored in the TCAM cell as $(B0, B1) = (0, 1)$. A logical one is stored in the TCAM cell as $(B0, B1) = (1, 0)$. A logical don't care is stored in the TCAM cell as $(B0, B1) = (0, 0)$. The cells $B0$ and $B1$ can be either static TCAM cells (4 CMOS transistors) or dynamic TCAM cells (1 CMOS transistor). It operates as follows:

1) Before the search operation, all the search lines (SL0 and SL1) are set low to make the match lines are separated from the ground. The match lines are then precharged to be high, but then disconnected from power supply so that the values of the lines are stored in the capacitance of the match lines.
2) After precharging the match lines, the search lines are driven to perform a search operation. To search for a logical zero, SL0 (search line) is set to high, while SL1 is set to low; to search a logical one, SL1 is set to high, while SL0 is set to low; and to perform the don't care search, both SL0 and SL1 are set to low.
3) If there is a mismatch, then the search line is high, and the stored bit is high to make a path from the match line to ground. Thus, the charge stored in the match line will remain intact only if there are no mismatch.

In the TCAM, a pair of cells $(B0, B1)$ stores only three distinct states (0,1), (1,0), and (0,0).

B. CAM with 2-bit encoding [5], [16]

Next, consider a **CAM with 2-bit encoding**, where two bits are represented by a 1-out-of-4 code shown in Table 5.1. Fig. 5.3 show the cells for a CAM with 2-bit encoding. This circuit corresponds to two bits of a TCAM. In the cells, B0,B1, B2, and B3, complemented data are stored. To search 00, only SL0 is set to high, to search 01, only SL1 is set to high, to search 10, only SL2 is set to high, and to search 11, only SL3 is set to high. Since the complemented data is stored in the cells B_i (i=0,1,2,3), when the searched data mismatches the data stored in the cells, the mismatched lines are discharged to low. For example, to store the value 00, CAM cells are sat to $(B0, B1, B2, B3) = (0, 1, 1, 1)$. Thus, when only $SL0$ is set to high, the match line remains high. However, if $SL1$ is set to high, the match line becomes low, which shows a mismatch.

In the CAM with 2-bit encoding, the CAM cells $(B0, B1, B2, B3)$ can take 15 different states: $(0, 0, 0, 0), (0, 0, 0, 1), \ldots, (1, 1, 1, 1)$. Note that the states $(1, 1, 1, 1)$ is not used. This is the reason why a CAM with 2-bit encoding is more efficient than the TCAM. Note that in a TCAM, cells for a pair of variable can take at most $3 \times 3 = 9$ states.

VI. Experimental Results

Lemma 4.1, shows that a CAM with 2-bit encoding never requires more products than a TCAM to represent interval functions. To compare the average numbers of products, SOPs

TABLE 5.1
1-OUT-OF-4 CODE.

Binary	1-out-of-4
00	0001
01	0010
10	0100
11	1000

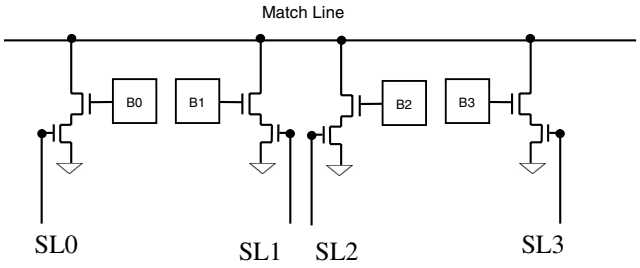

Fig. 5.3. CAM with 2-bit encoding.

with 2-valued and 4-valued variables for interval functions were minimized. We generated all the interval functions of for $n = 4, 6, 8$, and 10 variables, and minimized the SOPs for them, and obtained average numbers of products. Minimization program for multi-valued logic [11] was used to obtain exact minimum solutions. Table 6.1 compares the size of SOPs of two-valued variables and 4-valued variables. On the average, SOPs with 4-valued variables require about 30% fewer products than SOPs with 2-valued variables. One of the most interesting interval functions of $n = 10$ variables is $IN(10 : 257, 766)$, which requires 16 products in an SOP with two-valued variables, while only 5 products in an SOP with four-valued variables.

VII. CONCLUSION AND COMMENTS

In this paper, upper bounds on the number of products in SOPs with 2-valued and 4-valued variables to represent interval functions are derived. To represent any interval function of $n = 2r$ variables, an SOP with 2-valued variables requires up to $2(n - 2)$ products, while an SOP with 4-valued variables requires at most $n-1$ products. Since, a TCAM implements an SOP with 2-valued variables, and a CAM with 2-bit encoding implements an SOP with 4-valued variable, CAM with 2-bit encoding is promising for compact implementation of packet classifiers.

In this paper, two bits are grouped to represent a 1-out-of-4 code. However, three bits can be grouped to represent a 1-out-of-8 code. In this case, 8-valued logic is used to represent the function. The number of products often can be further reduces. However, 8 lines are used to represent a three-bit number by a 1-out-of-8 code. So, the number of the search lines increases by $\frac{8}{6} = 1.33$. Thus, the total area increases if the number of products remains same. Note that the use of 1-out-of-4 code never increases the total area, but often reduces the area.

In the practical applications, CAM data must be updated frequently. So, we need a fast logic minimization algorithm. Also, evaluation using practical rule sets should be necessary.

TABLE 6.1
AVERAGE NUMBER OF PRODUCTS TO REPRESENT INTERVAL FUNCTION

n	SOP 2-valued	SOP 4-valued	Ratio
4	2.648	1.857	0.7014
6	4.069	2.899	0.7124
8	5.778	4.146	0.7176
10	7.638	5.512	0.7216

ACKNOWLEDGMENTS

This research is supported in part by the Grants in Aid for Scientific Research of JSPS, and the grant of the MEXT Knowledge Cluster Project.

REFERENCES

[1] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software* (ISPASS-2006).

[2] F. Baboescu and G. Varghese, "Scalable packet classification," *IEEE/ACM Transactions on Networking*(TON), Vol.13, No.1, pp.2-14, Feb. 2005.

[3] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, "Packet classifiers in ternary CAMs can be smaller,"*ACM SIGMETRICS 2006*, pp.311-322.

[4] P. Gupta and N. McKeown, "Packet classification on multiple fields," *Proc. of ACM SIGCOMM*, pp. 147-160, 1999.

[5] S. Hanzawa, T. Sakata, K. Kajigaya, R. Takemura, and T. Kawahara, "A large-scale and low-power CAM architecture featuring a one-hot-spot block code for IP-address lookup in a network router," *IEEE Journal of Solid-State Circuits*,VOL. 40, NO. 4, APRIL 2005, pp.853- 861.

[6] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, August 22-26, 2005, Philadelphia, Pennsylvania, USA.

[7] H. Liu,"Efficient mapping of range classifier into ternary-CAM," *10th Symposium on High Performance Interconnects HOT Interconnects* (HotI'02), pp. 95-100, 2002.

[8] R. McGeer and P. Yalagandula," Minimizing rulesets for TCAM implementation," In *INFOCOM*, April 2009.

[9] C. R. Meiners, A. X. Liu, and E. Torng. "Topological transformation approaches to optimizing TCAM-based packet classification systems," In *Proceedings of the ACM Special Interest Group for the computer systems performance evaluation community* (SIGMETRICS), Seattle, WA, 2009.

[10] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, No. 3, pp. 712-727, March 2006.

[11] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[12] T. Sasao and J. T. Butler,"Worst and best irredundant sum-of-products expressions,"*IEEE Transactions on Computers*, Vol. 50, No. 9,Sept. 2001, pp.935-948.

[13] T. Sasao, " On the complexity of classification functions," *International Symposium on Multiple-valued Logic* (ISMVL-2008), Dallas, TX, May 22-24, 2008.

[14] B. Schieber, D. Geist, and A. Zaks, "Computing the minimum dnf representation of boolean functions defined by intervals," *Discrete Applied Mathematics*, Elsevier, Vol. 149, Issue 1-3, pp.154-173, Aug. 2005.

[15] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," *Proc. 11th IEEE International Conference on Network Protocols*, (ICNP 03), Nov.4-7, 2003, pp. 120-131.

[16] H. Wong and S. Kengeri, "Structure and method of an encoded ternary content addressable memory (CAM) cell for low-power compare operation, " US Pat. 6288922, Filed Aug 11, 2000.

[17] F. Yu, R.H. Katz, and T.V. Lakshman, "Efficient multimatch packet classification and lookup with TCAM," *IEEE Micro*, Vol.25, No.1, pp.50-59, Jan.- Feb. 2005.