PAPER

On the Numbers of Products in Prefix SOPs for Interval Functions

Infall SYAFALNI[†a)], *Student Member and* Tsutomu SASAO[†b)], *Member*

SUMMARY First, this paper derives the prefix sum-of-products expression (PreSOP) and the number of products in a PreSOP for an interval function. Second, it derives $\Psi(n, \tau_p)$, the number of n-variable interval functions that can be represented with τ_p products. Finally, it shows that more than 99.9% of the n-variable interval functions can be represented with $\lceil \frac{3}{2}n - 1 \rceil$ products, when n is sufficiently large. These results are useful for a fast PreSOP generator and for estimating the size of ternary content addressable memories (TCAMs) for packet classification.

key words: *prefix sum-of-products, number of products by PreSOP, distribution of interval functions, estimating the size of TCAM*

1. Introduction

Packet classification [3], [8], [9] is a core function in computer network components, such as routers, firewalls, network address translators, and access control lists (ACL). A ternary content addressable memory (TCAM) [14], [21] implements packet classification functions. Although a TCAM is fast, it is expensive and dissipates high power [1].

Table 1 shows an example of a packet classifier consisting of five fields: source IP, destination IP, source port, destination port, and protocol. In this example, both the source IP and the destination IP are represented by 32 bits, and they are specified by prefixes. Both the source port and the destination port are represented by 16 bits, and they are specified by intervals. The protocol is represented by a value of 8 bits or $*$ (*don't care*). The action has two values: *permit* and *deny*. However, it can have more values such as *deny and log* or *permit and log*. When each of the port fields is specified by either $*$ (*don't care*) or a single value, each rule corresponds to one word in a TCAM. However, when a port field is specified by an open interval such as $(0, 7)$, then **rule expansion** occurs, *i.e.*, each rule corresponds to many words in a TCAM [7]. If the packet header matches all the fields of a rule, then the rule is considered to be matched. If several rules match at the same time, the rule with the smallest number is applied.

To represent a port field, we use an interval function. An interval function can be represented as a sum of prefixes. The expression represented by a sum of prefixes is called a prefix sum-of-products expression (**PreSOP**). Note that each prefix corresponds to a word in a TCAM. To represent any interval function by a PreSOP, in the worst case, $2n - 2$ products are necessary [24].

A sum-of-product expression (**SOP**) requires no more products than a PreSOP to represent the same function, and in many cases, the SOP requires fewer products than the PreSOP. For example to represent the interval $(0, 2^n - 1)$, a PreSOP requires $2(n - 1)$ products, while an SOP requires only n products [17]. Although we can obtain an exact minimum SOP, it is often time consuming. In the design of integrated circuits for mass production, SOPs are routinely used [5], [20]. Their minimization cost can be amortized by many integrated circuits. However, in the case of packet classification, PreSOPs are used instead of SOPs.

To reduce the number of words in a TCAM for ACL, minimization algorithms for SOPs, that are suitable for embedded microprocessors, have been developed [2], [10]. It uses a ternary trie as a basic data structure. Such an algorithm is fast but produces solutions that require many more products than the exact minimum for some classes of functions [17].

Table 2 lists various methods to represent classification functions and their TCAM sizes. In [18], [22], the number of products needed to represent an interval function of n variables by a standard encoding in SOPs is analyzed. They show that any interval function can be represented with at most $2n - 4$ products in an SOP. Especially, the paper [18] shows that only two interval functions require $2n - 4$ products in SOPs. In the paper [19], a 4-valued TCAM is proposed, where inputs are encoded with a 1-out-of-4 code. In this method, any interval function can be represented with at most $n - 1$ products by a 4-valued SOP. To use this method in packet classification, a special CAM and a 4-valued logic minimizer are necessary. In the paper [23], a special TCAM that performs interval matching directly by hardware is proposed. In this case, each rule corresponds to just one word in a TCAM, but we need a special TCAM which would be expensive. In the paper [4], Gray encoding is used to reduce the number of products. In the Gray encoding, any interval $(A - 1, A + 2)$, where A is a non-negative integer, can be represented with a single product. The paper [4] showed that any interval function can be represented with at most $2n - 4$ products in a PreSOP.

In a TCAM, a priority encoder is included to produce a unique address among the matched data [14]. By changing the order of rules stored in a TCAM, we can often reduce the number of products needed to represent the function [7],

Manuscript received September 10, 2012.
Manuscript revised January 15, 2013.

[†]The authors are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.
a) E-mail: infall@aries01.cse.kyutech.ac.jp
b) E-mail: sasao@ieee.org

DOI: 10.1587/transinf.E96.D.1086

Table 1 Example of rules in packet classifier.

Rule	Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
1	66.219.40.$*$	176.31.166.$*$	$(-1, 65536)$	6790	TCP	Permit
2	$*$	15.238.61.128	$*$	$(1023, 65536)$	$*$	Permit
3	$*$	$*$	$*$	$*$	$*$	Deny

Table 2 Various methods to represent classification functions.

Method [Ref.]	Representation	Bound
Binary Tree [24]	n-variable binary prefixes	$2n-2$
2-valued MSOP [18]	n-variable binary non-prefixes	$2n-4$
Gray encoding [4]	n-variable binary prefixes	$2n-4$
Output encoding [16]	$(n+1)$-variable binary prefixes	n
4-valued MSOP [19]	$(\frac{n}{2})$-variable 4-valued non-prefixes	$n-1$
TCAM with comparator circuit [23]	A direct interval	1

[13]. In [12], the authors presented a method to minimize the number of TCAM words. Although this method can find an exact minimum solution, it requires computation time that is impractically large. In [16], an output encoding is used to reduce the number of products. To use this method in a real packet classification, a TCAM and an external memory are necessary.

In this paper, we show that 1) the minimum PreSOP (MPreSOP) and the number of products in a MPreSOP to represent an open interval (A, B) can be easily generated and calculated, and 2) the average number of products in MPreSOPs for interval functions is $\mu(n) \approx n-2$, and the variance is $\sigma^2(n) \approx \frac{n}{2}+1$ [†]. Also, by numerical calculation, we show that 99.9% of the functions can be represented by PreSOPs with at most $\lceil \frac{3}{2}n-1 \rceil$ products when $n > 12$.

2. Definitions and Basic Properties

2.1 Interval Functions

Definition 2.1: Let A and B be integers such that $A < B$. An **open interval** (A, B) denotes the set of integers X such that $A < X < B$. Note that endpoints are not included. The **size** of an open interval (A, B) is $C = B - A - 1$.

In this paper, only open intervals are considered. Thus from here, an open interval is simply denoted by an interval.

Definition 2.2: An n-input **interval function** is:

$$IN_0(n : A, B) = \begin{cases} 1, & \text{if } A < X < B \\ 0, & \text{otherwise,} \end{cases}$$

where $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, A and B are integers.

An interval function can be represented by a product of a *Greater-than* (GT) function and a *Less-than* (LT) function.

Definition 2.3: An n-input GT function is:

$$GT(n : A) = \begin{cases} 1, & \text{if } X > A \\ 0, & \text{otherwise.} \end{cases}$$

An n-input LT function is:

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

where $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$, A and B are integers.

Lemma 2.1: Let $-1 \le A < B \le 2^n$. The number of distinct open interval functions in (A, B), is $N(n) = 2^{n-1}(2^n + 1)$.

Proof: Let the size of an interval (A, B) be $C = B-A-1$. For $C = 1, C = 2, \ldots, C = 2^n$, the numbers of distinct interval functions are $2^n, 2^n-1, 2^n-2, \ldots, 1$, respectively. Thus, we have $N(n) = 2^n + (2^n-1) + (2^n-2) + \ldots + 1 = 2^{n-1}(2^n+1)$. □

2.2 Prefix SOPs

Definition 2.4: A **binary literal** has a form x^a, where x is a binary variable and $a \in \{0, 1\}$, and

$$x^a = \begin{cases} 1, & \text{if } x = a \\ 0, & \text{if } x \ne a. \end{cases}$$

Definition 2.5: $x = x^1$ and $\bar{x} = x^0$ are **literals** of a variable x. An AND of literals is a **product**. An OR of products is a **sum-of-products expression** (SOP).

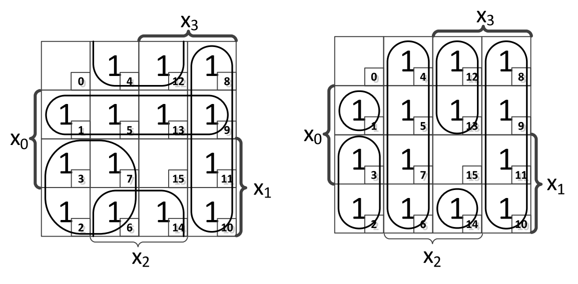
Definition 2.6: Let $\mathcal{F}$ be an SOP. $\tau(\mathcal{F})$ denotes the number of products in $\mathcal{F}$.

Definition 2.7: A prefix SOP (PreSOP) is an SOP consisting of products having the form $x_{n-1}^* x_{n-2}^* \ldots x_{m+1}^* x_m^*$, where x_i^* is x_i or $\bar{x}_i$ and $n-1 \ge m$.

Definition 2.8: An SOP representing a given function f with the fewest products is a minimum sum-of-product expression (MSOP). A PreSOP representing a given function f with the fewest products is a minimum PreSOP (MPreSOP). An MSOP and an MPreSOP for f are denoted by MSOP(f) and MPreSOP(f), respectively.

Lemma 2.2: Let $\tau_m(f) = \tau(\text{MSOP}(f))$ and $\tau_p(f) = \tau(\text{MPreSOP}(f))$. Since an MPreSOP is a restricted case of an MSOP, we have $\tau_m(f) \le \tau_p(f)$.

[†] $A(n) \approx B(n)$ iff $|A(n) - B(n)| \to 0$ as $n \to \infty$.



(a) MSOP (b) MPreSOP

Fig. 1 Maps for $f = IN_0(4:0,15)$.

Example 2.1: Figure 1 (a) shows an MSOP: $\bar{x}_0x_2 \vee x_0\bar{x}_1 \vee x_1\bar{x}_3 \vee \bar{x}_2x_3$ for the interval function $f = IN_0(4:0,15)$. Figure 1 (b) shows the MPreSOP: $\bar{x}_3\bar{x}_2\bar{x}_1x_0 \vee \bar{x}_3\bar{x}_2x_1 \vee \bar{x}_3x_2 \vee x_3\bar{x}_2 \vee x_3x_2\bar{x}_1 \vee x_3x_2x_1\bar{x}_0$. ■

Lemma 2.3: Any product in a PreSOP can be represented by an interval function:

$$x_{n-1}^{a_{n-1}}x_{n-2}^{a_{n-2}}\cdots x_m^{a_m} = IN_0(n : k2^m - 1, (k+1)2^m),$$

where $k = \sum_{i=0}^{n-m-1} a_{m+i}\cdot 2^i$, and m denotes the number of missing variables. Note that $k = 0,1,\ldots,2^{n-m}-1$ and $m = 0,1,2,\ldots,n$.

Thus, the products in the PreSOP for $IN_0(4:0,15)$ are represented as follows:

$\bar{x}_3\bar{x}_2\bar{x}_1x_0 : (0,2)$	$(k=1, m=0)$
$\bar{x}_3\bar{x}_2x_1 : (1,4)$	$(k=1, m=1)$
$\bar{x}_3x_2 : (3,8)$	$(k=1, m=2)$
$x_3\bar{x}_2 : (7,12)$	$(k=2, m=2)$
$x_3x_2\bar{x}_1 : (11,14)$	$(k=6, m=1)$
$x_3x_2x_1\bar{x}_0 : (13,15)$	$(k=14, m=0)$

The numbers in the smaller boxes in Fig. 1 show the values $X = 8x_3 + 4x_2 + 2x_1 + x_0$.

3. Number of Products in an MPreSOP

In this section, we show that, given an interval (A,B), the PreSOP and the number of products in the PreSOP can be generated and calculated easily. Previously, PreSOPs are generated by using binary trees [24]. However, the PreSOP of an interval (A,B) can be generated directly from its binary representations of the endpoints (A and B). Thus, the PreSOP for a given interval can be generated quickly.

Definition 3.1: A **vector literal** has the form $X^{\vec{a}}$, where $X = (x_{n-1}, x_{n-2},\ldots,x_1,x_0)$ and $\vec{a} = (a_{n-1},a_{n-2},\ldots,a_1,a_0)$.

$$X^{\vec{a}} = \begin{cases} 1, & \text{if } X = \vec{a} \\ 0, & \text{if } X \neq \vec{a}. \end{cases}$$

It is equivalent to $x_{n-1}^{a_{n-1}}x_{n-2}^{a_{n-2}}\cdots x_0^{a_0}$.

Lemma 3.1: $x^a = \bar{x}\cdot\bar{a} \vee x\cdot a$.

Lemma 3.2: A GT function has the following MPreSOP:

$$GT(n:A) = (x_{n-1}\bar{a}_{n-1}) \vee \bigvee_{i=n-2}^{0}\left(\bigwedge_{j=n-1}^{i+1} x_j^{a_j}\right)x_i\bar{a}_i,$$

where $\vec{a} = (a_{n-1},a_{n-2},\ldots,a_1,a_0)$ is the binary representation of A. $\tau_p(GT(n:A)) = \sum_{i=0}^{n-1}\bar{a}_i$.

Proof: Appendix. □

Similarly to Lemma 3.2, we have:

Lemma 3.3: An LT function has the following MPreSOP:

$$LT(n:B) = (\bar{x}_{n-1}b_{n-1}) \vee \bigvee_{i=n-2}^{0}\left(\bigwedge_{j=n-1}^{i+1} x_j^{b_j}\right)\bar{x}_ib_i,$$

where $\vec{b} = (b_{n-1},b_{n-2},\ldots,b_1,b_0)$ is the binary representation of B. $\tau_p(LT(n:B)) = \sum_{i=0}^{n-1} b_i$.

Similarly to $GT(n:A)$ and $LT(n:B)$, an interval function $IN_0(n:A,B)$ is represented as an MPreSOP.

Theorem 3.1: Let $\vec{a} = (a_{n-1},a_{n-2},\ldots,a_1,a_0)$ and $\vec{b} = (b_{n-1},b_{n-2},\ldots,b_1,b_0)$ be the binary representations of A and B, respectively. Let s be the largest index such that $a_s \neq b_s$, then $IN_0(n:A,B)$ can be represented by:

$$\bigvee_{i=s-1}^{0}\left[\left(\bigwedge_{j=n-1}^{i+1} x_j^{a_j}\right)x_i\bar{a}_i \vee \left(\bigwedge_{j=n-1}^{i+1} x_j^{b_j}\right)\bar{x}_ib_i\right]. \qquad (1)$$

Proof: Since both $GT(n:A)$ and $LT(n:B)$ have at most n products, the AND operation between GT and LT produces at most n^2 products.

Let a_i and b_i be the components of $\vec{a}$ and $\vec{b}$, respectively, where $s+1 \leq i \leq (n-1)$. For every pair of products in GT and LT functions, there exist a pair of components $(x_i\bar{a}_i)$ and $(\bar{x}_ib_i)$. These components cancel each other when $s+1 \leq i \leq (n-1)$. The cancellation occurs for three cases: the first case occurs when $(x_i\bar{a}_i)\cdot(\bar{x}_ib_i)$; the second case occurs when $(x_i\bar{a}_i)\cdot x_i^{b_i}$, and the last case occurs when $(\bar{x}_ib_i)\cdot x_i^{a_i}$. In the first case, $x_i\cdot\bar{x}_i$ yields 0. In the second and the third cases, by Lemma 3.1, we have $x_i\bar{a}_i(\bar{x}_i\bar{b}_i \vee x_ib_i) = x_i\bar{a}_ib_i = 0$ and $\bar{x}_ib_i(\bar{x}_i\bar{a}_i \vee x_ia_i) = x_i\bar{a}_ib_i = 0$, respectively. Thus, IN_0 can be simplified to:

$$\begin{aligned} IN_0(n:A,B) = &\left(x_{n-1}^{a_{n-1}}x_{n-2}^{a_{n-2}}\cdots x_{s+1}^{a_{s+1}}x_s\bar{a}_s \vee \cdots\right. \\ &\left.\vee\, x_{n-1}^{a_{n-1}}x_{n-2}^{a_{n-1}}\cdots x_1^{a_1}x_0\bar{a}_0\right)\cdot\left(x_{n-1}^{b_{n-1}}x_{n-2}^{b_{n-2}}\cdots\right. \\ &\left.x_{s+1}^{b_{s+1}}\bar{x}_sb_s \vee \cdots \vee x_{n-1}^{b_{n-1}}x_{n-2}^{b_{n-2}}\cdots x_1^{b_1}\bar{x}_0b_0\right). \end{aligned}$$

Thus, the interval function IN_0 has at most $(s+1)^2$ products. Since $a_s \neq b_s$, there are only two cases that produce non-zero products: the first case occurs for the AND operation between a GT product $x_s\bar{a}_s$, and an LT product without a $\bar{x}_s$ literal. The second case occurs for the AND operation between an LT product $\bar{x}_sb_s$, and a GT product without a x_s literal.

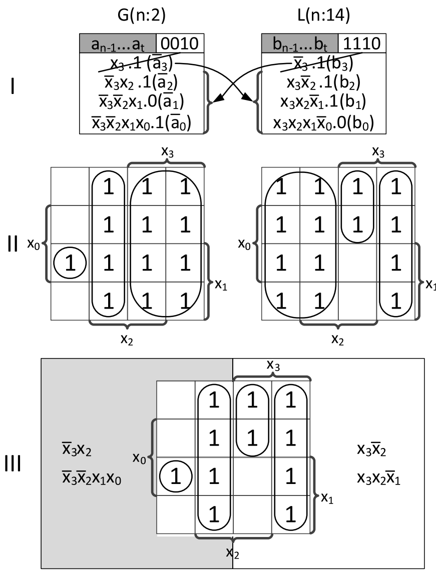


Fig. 2 Derivation of MPreSOP for $IN_0(6:2,14)$.

By performing all AND operations for those cases, the interval function IN_0 is represented as a disjunction of the products for GT and LT functions without $x_{n-1}^{a_{n-1}}x_{n-2}^{a_{n-2}}\cdots x_{s+1}^{a_{s+1}}x_s\bar{a}_s$ and $x_{n-1}^{b_{n-1}}x_{n-2}^{b_{n-2}}\cdots x_{s+1}^{b_{s+1}}\bar{x}_s b_s$. □

Lemma 3.4: Let $f(x_{n-1},x_{n-2},\ldots,x_1,x_0) = \bar{x}_{n-1}f_0 \vee x_{n-1}f_0$, where $f_0 = f(0,x_{n-2},\ldots,x_1,x_0)$ and $f_1 = f(1,x_{n-2},\ldots,x_1,x_0)$. Then, MPreSOP($f$) has the form

$$\bar{x}_{n-1}\text{MPreSOP}(f_0) \vee x_{n-1}\text{MPreSOP}(f_1).$$

Proof: Appendix. □

Theorem 3.2: Let $\vec{a} = (a_{n-1},a_{n-2},\ldots,a_1,a_0)$ and $\vec{b} = (b_{n-1},b_{n-2},\ldots,b_1,b_0)$ be the binary representations of A and B, respectively. Let s be the largest index such that $a_s \neq b_s$. Then, the expression Eq. (1) in Theorem 3.1 is the MPreSOP, and

$$\tau_p(IN_0(n:A,B)) = \sum_{i=0}^{s-1}(\bar{a}_i + b_i).$$

Proof: It is clear from Eq. (1). □

Example 3.1: Derive $\tau_p(IN_0(6:2,14))$ and the MPreSOP for $IN_0(6:2,14)$. The binary representations of $A = 2$ and $B = 14$ are $\vec{a} = (0,0,0,0,1,0)$ and $\vec{b} = (0,0,1,1,1,0)$, respectively. Since $(a_5,a_4) = (b_5,b_4) = (0,0)$ and $a_3 \neq b_3$, we have $s = 3$. Thus,

$$\tau_p(IN_0(6:2,14)) = \sum_{i=0}^{2}(\bar{a}_i + b_i) = 4.$$

In Fig. 2, the top row shows products of $GT(n:2)$ and $LT(n:14)$. The middle row shows their maps. And the bottom row shows the MPreSOP for $IN_0(n:2,14)$. Note that the largest products in $GT(n:2)$ and $LT(n:14)$ are removed, where the grey side indicates the $GT(n:2)$ part and the white side indicates the $LT(n:14)$ part. Thus, the MPreSOP is $IN_0(6:2,14) = \bar{x}_5\bar{x}_4\bar{x}_3\bar{x}_2x_1x_0 \vee \bar{x}_5\bar{x}_4\bar{x}_3x_2 \vee \bar{x}_5\bar{x}_4x_3\bar{x}_2 \vee \bar{x}_5\bar{x}_4x_3x_2\bar{x}_1$. ■

Theorem 3.1 can be used to generate MPreSOPs from binary representations of endpoints (A,B).

Lemma 3.2, Lemma 3.3, and Theorem 3.1 do not cover the cases when endpoints are $A = -1$ and $B \leq 2^n - 1$, or, $A \geq 0$ and $B = 2^n$, or, $A = -1$ and $B = 2^n$, because they require $(n+1)$ bits to represent the interval functions. These points are called **extremal endpoints**.

Lemma 3.5: In the extremal endpoints, we have $GT(n:-1) = LT(n:2^n) = IN_0(n:-1,2^n) = 1$, $IN_0(n:-1,B) = LT(n:B)$, and $IN_0(n:A,2^n) = GT(n:A)$.

4. Number of Interval Functions Requiring τ_p Products

In this section, we derive the formula for a number of interval functions requiring τ_p products in their MPreSOPs. With this, we can estimate the size of TCAM for packet classification [9].

Definition 4.1: Let $\Psi(n,\tau_p)$ be the number of n-variable interval functions that require τ_p products in their MPreSOPs.

To derive $\Psi(n,\tau_p)$, we have to consider the *extremal endpoints* that appeared before Lemma 3.5. Thus, the integers must be represented by $(n+1)$ bits for enumeration.

Definition 4.2: Let A and B be integers such that $-1 \leq A < B \leq 2^n$. Let $s(A,B)$ be the largest integer such that $a_s \neq b_s$. $s(A,B)$ is called **separation index**, where $0 \leq s(A,B) \leq n$. The binary representations of A and B are $\vec{a} = (a_n,a_{n-1},\ldots,a_1,a_0)$ and $\vec{b} = (b_n,b_{n-1},\ldots,b_1,b_0)$, respectively. $A = -1$ and $B = 2^n$ correspond to *extremal endpoints* that require $(n+1)$ bits.

In deriving $\Psi(n,\tau_p)$, the separation index s is used as a parameter to enumerate the number of interval functions. Table 3 shows all $IN_0(4:A,B)$ functions of 4 variables that require $\tau_p = 3$ products, where the upper integer denotes A, while the lower integer denotes B. The integers at the left side of the bits are the decimal representations of A and B. Also, A and B are represented by $n+1 = 5$bit numbers, the least significant s bits are separated to show the patterns of the combinations. For example, when $s = 2$, two cases satisfy the requirements. In the first case, the lower endpoint (A) contributes $\binom{2}{1} = 2$ products, while the upper endpoint (B) contributes $\binom{2}{2} = 1$ product. In the second case, A contributes $\binom{2}{2} = 1$ product, while B contributes $\binom{2}{1} = 2$ products. Thus, we have $\binom{2}{2}\binom{2}{1} + \binom{2}{1}\binom{2}{2} = 4$.

The top four pairs in Table 3 in the column headed by $s = 2$ denote all possible combinations of the least significant s bits that produce $\tau_p = 3$ products. The bottom four pairs correspond to the repetition of the top four pairs. Note

Table 3 List of interval functions for $n = 4$ and $\tau_p = 3$ for different s.

$s = 2$	$s = 3$		$s = 4$
0: 000 | 00	0: 00 | 000	3: 00 | 011	1: 0 | 0001
5: 001 | 01	8: 01 | 000	11: 01 | 011	16: 1 | 0000
0: 000 | 00	1: 00 | 001	3: 00 | 011	2: 0 | 0010
6: 001 | 10	9: 01 | 001	13: 01 | 101	16: 1 | 0000
1: 000 | 01	1: 00 | 001	3: 00 | 011	4: 0 | 0100
7: 001 | 11	10: 01 | 010	14: 01 | 110	16: 1 | 0000
2: 000 | 10	1: 00 | 001	5: 00 | 101	8: 0 | 1000
7: 001 | 11	12: 01 | 100	11: 01 | 011	16: 1 | 0000
8: 010 | 00	2: 00 | 010	5: 00 | 101	-1: 1 | 1111
13: 011 | 01	9: 01 | 001	13: 01 | 101	7: 0 | 0111
8: 010 | 00	2: 00 | 010	5: 00 | 101	-1: 1 | 1111
14: 011 | 10	10: 01 | 010	14: 01 | 110	11: 0 | 1011
9: 010 | 01	2: 00 | 010	6: 00 | 110	-1: 1 | 1111
15: 011 | 11	12: 01 | 100	11: 01 | 011	13: 0 | 1101
10: 010 | 10	4: 00 | 100	6: 00 | 110	-1: 1 | 1111
15: 011 | 11	9: 01 | 001	13: 01 | 101	14: 0 | 1110
	4: 00 | 100	6: 00 | 110	
	10: 01 | 010	14: 01 | 110	
	4: 00 | 100	7: 00 | 111	
	12: 01 | 100	15: 01 | 111	

that, in the top four pairs, the 4th bits in A (*i.e.*, a_3) and B (*i.e.*, b_3) are 0s, while, in the bottom four pairs, the 4th bits are 1s.

By Theorem 3.2, we enumerate τ_p with respect to the least significant s bits in the binary representations of the endpoints. Thus, we define:

Definition 4.3: Let $\eta(s, \tau_p)$ be the number of interval functions with the separation index s that require τ_p products in their PreSOPs.

Lemma 4.1: $\eta(s, \tau_p) = \binom{2s}{\tau_p}$.

Proof: By Theorem 3.2, $\eta(s, \tau_p)$ is equal to the number of combinations such that

$$\sum_{j=0}^{s-1} (\bar{a}_j + b_j) = \tau_p.$$

This number is equal to the number of ways to distribute τ_p elements to $2s$ bins. Thus, we have $\binom{2s}{\tau_p}$. □

Lemma 4.2: Let $r(n, s)$ be the number of repetitions that multiplies $\eta(s, \tau_p)$. Then, we have $r(n, s) = 2^{n-s-1}$.

From here, we will consider the *extremal endpoints* that require $(n + 1)$ bits to represent the numbers. These cases occur when one of endpoints is -1 or 2^n. Note that the binary representation of -1 is $\vec{a} = (a_n, a_{n-1}, \ldots, a_1, a_0) = (1, 1, \cdots, 1, 1)$ and the binary representation of 2^n is $\vec{b} = (b_n, b_{n-1}, \ldots, b_1, b_0) = (1, 0, \cdots, 0, 0)$.

Lemma 4.3: For the interval functions with $s(A, B) = n$ and $\tau_p \leq n$, we have $\eta(n, \tau_p) = 2\binom{n}{\tau_p}$.

Proof: When $s(A, B) = n$, $a_n \neq b_n$. This occurs when $A = -1$ and $B \leq 2^n - 1$, or, $A \geq 0$ and $B = 2^n$. In these cases, one of endpoints *i.e.*, $A = -1$ or $B = 2^n$, contributes no product. Because of that, the other endpoints must contribute τ_p products. The number of ways to produce τ_p products is $\binom{n}{\tau_p}$. Thus, we have $\eta(n, \tau_p) = \binom{n}{\tau_p}\binom{n}{0} + \binom{n}{0}\binom{n}{\tau_p} = 2\binom{n}{\tau_p}$. □

Example 4.1: In Table 3, when $s = n = 4$ and $\tau_p = 3$, the *extremal endpoints* occur, and we have $\eta(4, 3) = 2\binom{4}{3} = 8$. ■

Lemma 4.4: For $s(A, B) \geq n$ and $\tau_p > n$, $\eta(s, \tau_p) = 0$.

Proof: Note that no interval function satisfies the condition. For $s = n$ and $\tau_p > n$, the only allowable endpoints are $2^n + 1$ and 2^n. These endpoints contribute no products, and the remaining space from another endpoint can be represented with only n products. □

Theorem 4.1:

$$\Psi(n, \tau_p) = \left(\sum_{s=1}^{n-1} 2^{n-s-1} \binom{2s}{\tau_p} \right) + 2\binom{n}{\tau_p}.$$

Proof: $\Psi(n, \tau_p)$ is given as the sum of the number of open interval functions $\eta(s, \tau_p)$ times the repetition factor $r(n, s)$ for every possible s:

$$\Psi(n, \tau_p) = \left(\sum_{s=\lceil \frac{1}{2}\tau_p \rceil}^{n-1} r(n, s)\eta(s, \tau_p) \right) + \eta(n, \tau_p). \qquad (2)$$

For the separation index $s(A, B)$, we have $\lceil \frac{1}{2}\tau_p \rceil \leq s(A, B) \leq n$. $s(A, B)$ takes its maximum and minimum values in the extremal endpoints. Thus, the sum operation in Eq. (2) is bounded from $\lceil \frac{1}{2}\tau_p \rceil$ to $n - 1$. Moreover, since $\binom{2s}{\tau_p} = 0$ when $2s < \tau_p$, the lower bound can be simply 1. From Lemmas 4.1 to 4.4 and Eq. (2), we have the theorem. □

5. Statistical Properties

In this section, we show some statistical properties of the number of products in PreSOPs. We assume that each function is equally likely. Therefore, the probability distribution function of the number of interval functions with τ_p products in PreSOPs is given by $\frac{\Psi(n,\tau_p)}{N(n)}$.

5.1 Average Number of Products

The average number of products in PreSOPs (**mean**) for n-variable interval functions is given by

$$\mu(n) = \frac{1}{N(n)} \sum_{\tau_p=1}^{2(n-1)} \tau_p \cdot \Psi(n, \tau_p),$$

where $N(n)$ is the total number of distinct interval functions of n variables, and $\Psi(n, \tau_p)$ is the number of interval functions that require τ_p products.

Theorem 5.1:

$$\mu(n) \approx n - 2.$$

Proof: Appendix. □

5.2 Variance of the Numbers of Products

The **variance** [6] of the numbers of products in PreSOPs for interval functions is given by

$$\sigma^2(n) = \sum_{\tau_p=1}^{2(n-1)} \tau_p^2 \frac{\Psi(n,\tau_p)}{N(n)} - \mu(n)^2.$$

Theorem 5.2:

$$\sigma^2(n) \approx \frac{n}{2} + 1.$$

Proof: Appendix. □

Lemma 5.1: (Chebyshev's inequality [6]) Let X be a random variable with a finite expected value μ and a finite nonzero variance σ^2. Then, for any real number $k > 0$, we have

$$Pr(|X - \mu|) \geq k\sigma) \leq \frac{1}{k^2}.$$

For $k = 2$, we have,

$$Pr(|X - \mu|) \geq 2\sigma) \leq \frac{1}{4}.$$

Corollary 5.1: For large n, at least 75% of n-variable interval functions can be represented by PreSOPs with $(n - 2) + \sqrt{2(n+2)}$ products.

6. Experimental Results

Although we have the formula for the number of products in MPreSOPs, we do not have one in MSOPs. To compare the numbers of products of PreSOPs with MSOPs [5], we generated all the interval functions for $n = 1$ to $n = 12$. To obtain exact minimum SOPs, we used ESPRESSO-EXACT [5]. In MSOPs, the maximums occur for $IN_0(n : 2^{n-3}, 7 \cdot 2^{n-3} - 1)$ and $IN_0(n : 2^{n-2}, 3 \cdot 2^{n-2} - 1)$ [18]. On the other hand, as shown in Sect. 3, in PreSOPs, the maximum occurs for $IN_0(n : 0, 2^n - 1)$.

Figure 3 compares average numbers of products in MSOPs and PreSOPs for different sizes of intervals. The curves for MSOPs tangent to the envelope $\tau = \log_2(B - A)$ and τ is the number of products in an MSOP. In Fig. 3 the envelope is shown by the black dotted curve. The peaks of the curves show that PreSOPs require more products (including the maximum, $\tau_p = 2(n-1)$) when the size of interval approaches to $2^n - 2$, and the curves drop sharply when the sizes of intervals are $2^n - 1$ ($\tau_p = n$) and 2^n ($\tau_p = 1$). Table 4 shows the average numbers of products needed to represent interval functions. The ratio of the numbers of products in PreSOPs to that of MSOPs for $n > 5$ is about 1.05.

We calculated the probabilities of the interval functions that can be represented with at most τ_p products for $n = 8$ to $n = 16$. Table 5 shows that, for $n > 12$, more than 98.75% of the interval functions can be represented with $\lceil n + \sqrt{2(n+2)} - 2\rceil$ products, and more than 99.9% of the interval functions can be represented with $\lceil \frac{3}{2}n - 1\rceil$ products.

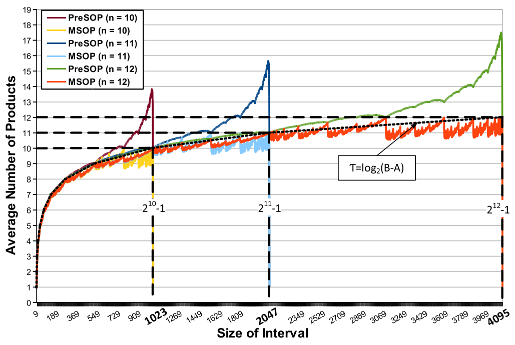


Fig. 3 Average numbers of products in MSOPs and PreSOPs for different sizes of intervals.

Table 4 Average numbers of products needed to represent interval functions.

n	$N(n)$	MSOP	PreSOP	Difference	Ratio
4	136	2.39706	2.47794	0.08088	1.03374
5	528	3.12879	3.27462	0.14583	1.04661
6	2080	3.94327	4.15433	0.21106	1.05352
7	8256	4.81541	5.08539	0.26999	1.05607
8	32896	5.72671	6.04672	0.32001	1.05588
9	131328	6.66450	7.02535	0.36084	1.05414
10	524800	7.62032	8.01366	0.39334	1.05162
11	2098176	8.58858	9.00732	0.41874	1.04876
12	8390656	9.56540	10.00391	0.43850	1.04584

Table 5 Probabilities of interval functions that can be represented with at most τ_p products.

n	$\tau_p = \lceil\mu(n) + c\rceil$		
	$c = \sigma(n)$	$c = 2\sigma(n)$	$c = \sigma^2(n)$
8	95.03587%	99.67169%	99.67169%
9	94.00204%	99.44642%	99.89416%
10	93.01715%	99.17569%	99.80526%
11	92.08265%	99.68811%	99.93380%
12	91.19742%	99.54448%	99.88636%
13	90.35921%	99.37685%	99.95988%
14	89.56528%	99.18796%	99.93412%
15	88.81274%	98.98061%	99.97608%
16	88.09875%	98.75744%	99.96191%

7. Conclusions and Comments

As key contributions of this work, we

1. showed that the MPreSOP and the number of products in the MPreSOP (τ_p) to represent an interval (A, B) can be directly obtained.
2. derived the formula $\Psi(n, \tau_p)$ for the number of interval functions that require τ_p products in PreSOPs.
3. showed by experiments that the average number of products in MSOPs is approximated by $\tau = \log_2(B - A)$.
4. showed that the average number of products $\mu(n)$ needed to represent interval functions of an n-bit field by PreSOPs is about $n - 2$, with an approximate variance $\sigma^2(n)$ of $\frac{n}{2} + 1$.
5. showed by numerical computations that more than 99.9% of the interval functions of an n-bit field can be

represented by PreSOPs with $\lceil \frac{3}{2}n - 1 \rceil$ products, when $n > 12$.

To represent an interval of an n-bit field, a PreSOP requires up to $2(n-1)$ products. When both the source and the destination ports have $n = 16$ bits fields, the number of products needed to represent one rule by a PreSOP can be up to $2(n-1) \times 2(n-1) = 30 \times 30 = 900$. Theorem 5.1 shows that the average number of products needed to represent an interval is about $n - 2$. Thus, when the rule have two interval fields, the average numbers of products needed to represent one rule by PreSOP would be $(n-2)^2$, which is $14^2 = 196$ when $n = 16$. In the real applications, the rule expansion is not so large. For example, the paper [24] reported that the average number of TCAM entries for 12 real packet classifiers is 6.2 times of their number of rules. This is because, in the real applications, the distributions of the interval functions are not uniform.

Acknowledgments

This research is supported in part by the Grants in Aid for Scientific Research of JSPS. The reviewer's comments were useful to derive Theorems 5.1 and 5.2. Prof. J.T. Butler's comments were also useful.

References

[1] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," IEEE ISPASS, pp.120–129, March 2006.

[2] S. Ahmad and R. Mahapatra, "An efficient approach to on-chip logic minimization," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.15, no.9, pp.1040–1050, Sept. 2007.

[3] F. Baboescu and G. Varghese, "Scalable packet classification," IEEE/ACM Trans. Netw., vol.13, no.1, pp.2–14, Feb. 2005.

[4] A. Bremler-Barr and D. Hendler, "Space-efficient TCAM-based classification using gray coding," IEEE INFOCOM, pp.1388–1396, May 2007.

[5] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, MA, 1984.

[6] A. DasGupta, Fundamentals of Probability: A First Course, Springer, New York, 2010.

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

[8] P. Gupta and N. McKeown, "Packet classification on multiple fields," ACM SIGCOMM, pp.147–160, Sept. 1999.

[9] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary CAMs," ACM SIGCOMM, pp.193–204, Aug. 2005.

[10] R.L. Lysecky and F. Vahid, "On-chip logic minimization," ACM DAC, pp.334–337, 2003.

[11] L. Lovasz, J. Pelikan, and K. Vesztergombi, Binomial Coefficients and Pascal's Triangle, Springer, New York, 2003.

[12] R. McGeer and P. Yalagandula, "Minimizing rulesets for TCAM implementation," IEEE INFOCOM, pp.1314–1322, April 2009.

[13] C.R. Meiners, A.X. Liu, and E. Torng, "Topological transformation approaches to optimizing TCAM-based packet classification systems," ACM SIGMETRICS, pp.73–84, June 2009.

[14] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE J. Solid-State Circuits, vol.41, no.3, pp.712–727, March 2006.

[15] B. Reusch, "Generation of prime implicant from subfunctions and a unifying approach to the covering problem," IEEE Trans. Comput., vol.C-24, no.9, pp.924–930, Sept. 1975.

[16] O. Rottenstreich and I. Keslassy, "On the code length of TCAM coding schemes," IEEE ISIT, pp.1908–1912, July 2010.

[17] T. Sasao and J.T. Butler, "Worst and best irredundant sum-of-products expressions," IEEE Trans. Comput., vol.50, no.9, pp.935–948, Sept. 2001.

[18] T. Sasao, "On the complexity of classification functions," IEEE ISMVL, pp.57–63, May 2008.

[19] T. Sasao, "On the number of products to represent interval functions by SOPs with four-valued variables," IEEE ISMVL, pp.282–287, May 2010.

[20] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, Boston, MA, 1999.

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

[22] B. Schieber, D. Geist, and A. Zaks, "Computing the minimum DNF representation of boolean functions defined by intervals," Discrete Appl. Math., vol.149, issue 1-3, pp.154–173, Aug. 2005.

[23] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," IEEE ICNP, pp.120–131, Nov. 2003.

[24] D.E. Taylor, "Survey and taxonomy of packet classification techniques," ACM Comput. Surv., vol.37, no.3, pp.238–275, 2005.

Appendix

Proof of Lemma 3.2

Since we consider PreSOPs, we have the following:
When $a_{n-1} = 0$:

$$GT(n : A) = x_{n-1} \vee \bar{x}_{n-1}\mathrm{MPreSOP}(GT(n-1 : \hat{A})),$$

where $\hat{A}$ is the integer represented by $(a_{n-2}, a_{n-3}, \ldots, a_1, a_0)$.
When $a_{n-1} = 1$:

$$GT(n : A) = x_{n-1}\mathrm{MPreSOP}(GT(n-1 : \hat{A})).$$

For $n = 1$ and $n = 2$, it is clear that the lemma holds. By mathematical induction, we have the lemma. □

Proof of Lemma 3.4

In the case of SOPs, an MSOP can be found by the following approach [20]:
1) Generate the set of all the prime implicants (PIs).
2) Among the set of PIs, select a minimum set of PIs that covers the minterms.

Let PI(f) be the set of all the PIs for f. Then, we have the following relation [15]

$$\mathrm{PI}(f) = \bar{x}_{n-1}\mathrm{PI}(f_0) \cup x_{n-1}\mathrm{PI}(f_1) \cup \mathrm{PI}(f_0 \cdot f_1).$$

In the case of PreSOPs, an MPreSOP can be found by a similar approach, but the first step should be modified as follows:
1') Generate PrePI(f), the set of all the PIs for f having the form $x^*_{n-1}x^*_{n-2}\cdots x^*_m$. Note that PrePI($f$) can be written as

$$\mathrm{PrePI}(f) = \bar{x}_{n-1}\mathrm{PrePI}(f_0) \cup x_{n-1}\mathrm{PrePI}(f_1).$$

Since functions are represented by PreSOPs, all the products have the form $x^*_{n-1}x^*_{n-2}\cdots x^*_m$. Thus, we do not have to generate PrePI($f_0 \cdot f_1$). This implies that to obtain the MPreSOP(f), we can minimize the subfunctions independently as follows: MPreSOP(f) = $\bar{x}_{n-1}$MPreSOP(f_0) $\vee$ x_{n-1}MPreSOP(f_1). □

Corollary A.1: Let $f(x_{n-1}, x_{n-2}, \ldots, x_0) = \bar{x}_{n-1}f_0 \vee x_{n-1}f_1$, where $f_0 = f(0, x_{n-2}, \ldots, x_0)$ and $f_1 = f(1, x_{n-2}, \ldots, x_0)$. Then,

$$\tau_p(f) = \tau_p(f_0) + \tau_p(f_1).$$

Lemma A.1:

$$\sum_{i=1}^{n} i\binom{n}{i} = n2^{n-1}.$$

$$\sum_{i=1}^{n} i^2\binom{n}{i} = n(n+1)2^{n-2}.$$

Lemma A.2:

$$\sum_{i=1}^{n} i2^i = 2^{n+1}(n-1) + 2.$$

$$\sum_{i=1}^{n} i^2 2^i = 2^{n+1}(n^2 - 2n + 3) - 6.$$

Proof of Theorem 5.1

The average number of products in PreSOPs is

$$\begin{aligned}
\mu(n) &= \frac{1}{N(n)} \sum_{k=1}^{2(n-1)} k\Psi(n,k) \\
&= \frac{1}{N(n)} \sum_{k=1}^{2(n-1)} k \sum_{s=1}^{n-1} 2^{n-s-1}\binom{2s}{k} + \frac{2}{N(n)} \sum_{k=1}^{2(n-1)} k\binom{n}{k} \\
&= \frac{1}{N(n)} \sum_{k=1}^{2(n-1)} k \sum_{s=1}^{n-1} 2^{n-s-1}\binom{2s}{k} + \frac{n2^n}{(2^n+1)2^{n-1}}.
\end{aligned}$$

The second term is negligibly smaller than the first term. For the first term, by Lemmas A.1 and A.2, we have

$$\begin{aligned}
&\sum_{k=1}^{2(n-1)} k \sum_{s=1}^{n-1} 2^{n-s-1}\binom{2s}{k} \\
&= \sum_{s=1}^{n-1} 2^{n-s-1} \sum_{k=1}^{2(n-1)} k\binom{2s}{k} = \sum_{s=1}^{n-1} 2^{n-s-1} s 2^{2s} \\
&= \sum_{s=1}^{n-1} s2^{n+s-1} = 2^{n-1}\sum_{s=1}^{n-1} s2^s \\
&= 2^{n-1}(2^n(n-2)+2) = 2^{2n-1}(n-2) + 2^n.
\end{aligned}$$

Thus,

$$\mu(n) \approx \frac{2^{2n-1}(n-2)+2^n}{N(n)} \approx \frac{2^{2n-1}(n-2)}{2^{2n-1}} = n-2.$$

Hence, we have the theorem. □

Proof of Theorem 5.2

The variance of the numbers of products in PreSOPs is

$$\begin{aligned}
\sigma^2(n) &= \frac{1}{N(n)} \sum_{k=1}^{2(n-1)} k^2\Psi(n,k) - \mu^2(n) \\
&= \frac{G}{N(n)} - \mu^2(n).
\end{aligned}$$

Note that by Theorem 4.1, we have

$$\begin{aligned}
G &= \sum_{k=1}^{2(n-1)} k^2\Psi(n,k) \\
&= \sum_{k=1}^{2(n-1)} k^2 \sum_{s=1}^{n-1} 2^{n-s-1}\binom{2s}{k} + 2\sum_{k=1}^{2(n-1)} k^2\binom{n}{k}.
\end{aligned}$$

The first term in G is equal to

$$\begin{aligned}
&\sum_{s=1}^{n-1} 2^{n-s-1} \sum_{k=1}^{2(n-1)} k^2\binom{2s}{k} \\
&= \sum_{s=1}^{n-1} 2^{n-s-1} 2s(2s+1)2^{2s-2} \\
&= \sum_{s=1}^{n-1} 2^{n+s-1}(s^2 + \frac{s}{2}) \\
&= 2^{n-1}\sum_{s=1}^{n-1} s^2 2^s + 2^{n-2}\sum_{s=1}^{n-1} s2^s.
\end{aligned}$$

The second term in G is negligibly smaller than the first term, so we can ignore it. Thus, from Lemma A.2, G is approximated by

$$2^{n-1}(2^n((n-1)^2 - 2(n-1) + 3) - 6) + 2^{n-2}((n-2)2^n)$$

or,

$$2^{2n-1}((n-1)^2 - 2(n-1) + 3) + 2^{2n-2}(n-2).$$

Thus, we have

$$\begin{aligned}
\frac{G}{N(n)} &\approx \frac{2^{2n-1}((n-1)^2 - 2(n-1) + 3)}{2^{2n-1}} + \frac{n-2}{2} \\
&\approx (n-1)^2 - 2(n-1) + 3 + \frac{n-2}{2}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\sigma^2(n) &\approx \frac{G}{N(n)} - (n-2)^2 \\
&\approx (n-1)^2 - 2(n-1) + 3 - (n-2)^2 + \frac{n-2}{2} \\
&\approx \frac{n+2}{2}.
\end{aligned}$$

Thus, we have the theorem. □

Infall Syafalni was born in 1987. He received B.E. in electrical engineering from Bandung Institute of Technology (ITB), Indonesia in 2008, and M.Sc. in electronic engineering from USM, Malaysia in 2011. He is currently a doctoral student in the Department of Computer Science and Electronics at Kyushu Institute of Technology, Japan. His research interests include logic design and VLSI design. He is a student member of the IEEE.

Tsutomu Sasao received the B.E., M.E., and Ph.D. degrees in electronics engineering from Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan, the IBM T.J.Watson Research Center, Yorktown Heights, NY, and the Naval Postgraduate School, Monterey, CA. He has served as the Director of the Center for Microelectronic Systems at the Kyushu Institute of Technology, Iizuka, Japan. Now, he is a Professor of the Department of Computer Science and Electronics. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published more than nine books on logic design, including Logic Synthesis and Optimization, Representation of Discrete Functions, Switching Theory for Logic Synthesis, Logic Synthesis and Verification, and Memory-Based Logic Synthesis, in 1993, 1996, 1999, 2001, and 2011, respectively. He has served as Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the NIWA Memorial Award in 1979, Takeda Techno-Entrepreneurship Award in 2001, and Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVLs in 1986, 1996, 2003 and 2004. He has served as an Associate Editor of the IEEE Transactions on Computers. He is a Fellow of the IEEE.