# A Nonredundant Ternary CAM Circuit for Network Search Engines

Mohammad J. Akhbarizadeh, *Student Member, IEEE*, Mehrdad Nourani, *Senior Member, IEEE*, Deepak S. Vijayasarathi, and Poras T. Balsara, *Senior Member, IEEE*

***Abstract*—An optimized Ternary CAM concept is introduced for the hardware search engines in high-speed Internet routers. Our design employs $w + 1$ RAM bits to store a word of size $w$, whereas a conventional TCAM needs $2w$ RAM bits for the same word size. Based on this concept an 8-bit cluster is designed out of 9 SRAM bits, used as the basic building block of our Prefix-CAM (PCAM) structure. Four such clusters merge to store a 32-bit IPv4 prefix, thus, configuring a PCAM suitable for Internet packet forwarding. This PCAM module employs 48% less SRAM cells and a total of 22% less transistors plus 50% less address decode interconnects compared to a conventional TCAM, for equal storage size and equal functionality. We show that PCAM can be employed for multifield packet classification. Other factors, such as lookup speed and power dissipation, are not adversely affected.**

***Index Terms*—Forwarding engine, packet classification, prefix-CAM, route lookup table, ternary content addressable memory.**

## I. Introduction

### A. Motivation

NOWADAYS, more hardware architects than ever look into content addressable memory (CAM) for high-performance table lookup tasks [1]–[3]. Various types of CAM intellectual property cores are available in the market for rapid integration. CAD tools, design automation environments, and field programmable gate array (FPGA) based systems have become more CAM-aware than ever. A specifically interesting type of CAM, called Ternary CAM (TCAM), can store *don't-care* values in addition to 0's and 1's. Using this capability, the TCAM entries can include wild-cards. Due to the wild-cards, a search key may match multiple entries. In this case, a TCAM with properly structured content can produce the highest priority (or the most specific) result. TCAM completes each lookup task in just one clock cycle.

For most networking applications, all the wild-cards are located at the right side of the primitive data element, called *prefix*. In case of basic Internet Protocol (IP) packet forwarding, each data element is a single prefix that represents an Internet route. The search keys are the IP destination addresses. The priority is determined based on the prefix length. Therefore, when a destination address matches multiple route prefixes the longest, which defines the most specific route, will be chosen [4], [5].

For purposes such as quality of service and security, a router has to classify incoming IP packets at maximum speed. In case of packet classification, each table entry, called a rule, can be comprised of multiple data fields. Each field is usually a complete binary value or a prefix. Rarely, a field can contain wild-cards in arbitrary locations [6]. A packet classification task requires a multidimensional search. The search key is composed of IP, transfer control protocol (TCP), and occasionally datalink or even application header fields. The search time and/or storage complexity of software-based algorithmic approaches are usually very high. The great advantage of TCAM is that once all the data fields in a rule are attached, the classification lookup can take place in only one clock cycle [7].

One of the first works to describe ternary CAM was [8]. Later, they introduced a hardware search engine based on that circuit [9]. While simplicity and high performance are the main reasons for designers to choose TCAM for hardware-based search applications, high-power dissipation and low-storage density remain the two major concerns with this technology, which makes it a hot topic of ongoing research in both industry and academia (a few recent examples are [10]–[13]). Our design, called Prefix-CAM (PCAM), primarily addresses the storage density problem [28].

### B. Key Novelty

One of the problems of TCAM, in addition to high power dissipation, is its low-storage density due to the high number of transistors per cell. Each TCAM cell requires 16 transistors [15] (12 in some literature [16]), as opposed to 6 for SRAM or 2 for DRAM [19]. This has inspired some researchers to offer heuristics that optimize TCAM usage [20], [21]. In the state-of-the-art TCAM technology, bits at any *arbitrary position* in a word can be masked independently. This flexibility comes at a cost. Each cell includes two SRAM (DRAM) bits to store each of the three possible states of the cell, namely 0, 1, and *don't-care*. However, most of the networking applications of TCAM do not require such flexibility. The major application of high-density, fast, low-power TCAM products is in the classification and forwarding engines of broadband Internet routers. The majority of such applications need to store and search prefixes [5], [22]. All of the masked bits in a prefix are adjacent and are gathered at the right side. For example, 1100110010* is a 10–stabit IP prefix shown in binary. The 22 lower bits of this prefix are masked. On the other hand, 11x01* (x denotes one *don't-care* bit here) is not a valid prefix because it has a masked bit surrounded by valid bits.

Manuscript received January 12, 2005; revised August 27, 2005. This work was supported in part by the Academic Research and Technology Initiative (ARTI) Award from Cisco Systems Inc..

M. J. Akhbarizadeh is with Cisco Systems Inc., San Jose, CA 95134 USA (e-mail: makhbari@cisco.edu).

M. Nourani and P. T. Balsara are with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX 75083 USA (e-mail: nourani@utdallas.edu; dxv033000@utdallas.edu; poras@utdallas.edu).

D. S. Vijayasarathi is with Intel Corporation, Folsom, CA 95630 USA.

Digital Object Identifier 10.1109/TVLSI.2006.871760

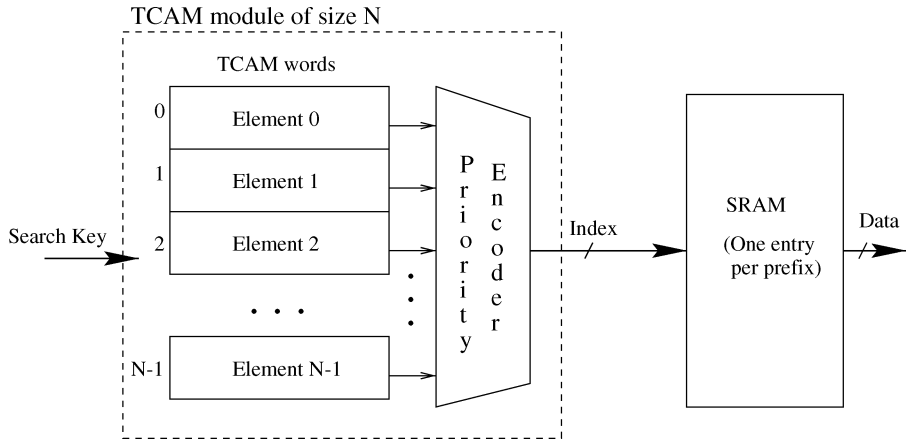



Fig. 1. Basic structure of a TCAM module.

Currently, in a TCAM module with a width of $w$ (i.e., each word stores a prefix with a maximum length of $w$), $2w$ SRAM bits are needed per word (see Fig. 2). On the other hand, all the valid combinations of $w$-bit IP prefixes accumulate to $2^{w+1}-1$ values (null prefix plus the number of all 1-bit prefixes plus all 2-bit prefixes, all the way to the $w$-bit prefixes, or $\sum_{i=0}^{w} 2^i = 2^{w+1} - 1$). This can be represented with no more than $w+1$ RAM bits, instead of $2w$. For $w = 32$, that means 48.4% reduction in memory usage. This is an initiative to introduce a modified TCAM design that effectively saves silicon area, while providing a performance similar to state-of-the-art TCAM.

Our design will alter the structure of a TCAM word and, thus, the way prefixes are stored and the masked comparison operation is performed. However, the changes remain transparent to users and the general TCAM architecture (shown in Fig. 1) is maintained. Therefore, all behavioral schemes of state-of-the-art TCAM (i.e., prefix sorting, updating algorithms, etc.) are perfectly applicable to our PCAM.

### C. Paper Organization

The rest of this paper is organized as follows. This section will continue by clarifying necessary assumptions and definitions that will be used throughout the paper. Then, the conventional TCAM design is explained in Section II-B, to be used as a reference model for comparison purposes. The motivation behind this work is explained in Section I-B. Section III explains the main contribution of this paper. We first derive the optimized logic equations, then describe the circuitry for our novel PCAM. Then, two additional functional units (*decoder* and *encoder*) necessary for this design are introduced. Section IV extends the application of PCAM for the general case of multidimensional packet classification. Section V summarizes our implementation results and practical observations. Paper summary and conclusions appear in Section VI.

## II. Background

### A. Assumptions and Definitions

In this paper, we discuss static TCAM circuit where memory cells are SRAM. The majority of high-end commercial TCAM chips are static. Many of the discussions and examples in this paper are presented for IPv4. However, it can be easily demonstrated that the results are directly applicable to IPv6 as search engines apply similar procedures (with different search-key lengths) to both. In fact, IPv6 is more storage-hungry and thus our solution can even be more beneficial to this protocol than IPv4.

Any prefix can be shown in binary radix as a string of 1's and 0's, followed by a wild-card symbol * that marks all other bits at the right side of prefix as *don't-care*. For instance, $P = 10011000101*$ is an 11-bit prefix. We will also show prefix $P$ as a value $(V)$ and mask $(M)$ pair, i.e., $P[w-1:0] = (V[w-1:0], M[w-1:0])$, where $w$ is the maximum width of prefix. Also, $P[w-1:0] = p_{w-1}p_{w-2}\ldots p_0$, $V[w-1:0] = v_{w-1}v_{w-2}\ldots v_0$, and $M[w-1:0] = m_{w-1}m_{w-2}\ldots m_0$. Therefore, bit $i$ of prefix is denoted by $p_i = (v_i, m_i)$. A mask bit $m_i$ is 1 wherever the prefix value is *don't-care* and 0 where the value is valid. So, if $P[31:0] = 10011000101*$ then, $P[31:0] = (V[31:0], M[31:0]) = (98A00000H,\ 001FFFFFH)$. In this example, $p_{31} = (1,0)$, $p_{30} = (0,0)$, ..., and $p_0 = (0,1)$. Traditionally, an IPv4 prefix such as $P$ might also be represented as 152.160/11, which indicates that the meaningful part of the prefix value at the left side of the $'/'$ sign, and the prefix length at the right side of it.

### B. The Reference Model

Fig. 1 shows the basic structure of a TCAM used in many of the state-of-the-art Internet forwarding and classification engines [14]. In this structure, the entries are sorted based on their priorities. Higher priority entries are placed in memory locations with higher addresses. A given search key is distributed among all memory locations at the same time. Then, the results of the masked comparisons, performed by all TCAM words simultaneously, are taken into a huge priority encoder that gives priority to the higher memory locations. Therefore, the index of the highest priority element will be chosen in case of multiple matches. This index may then be used to extract corresponding classification or forwarding information such as egress number from an SRAM module that has $N$ entries, one for each TCAM entry.

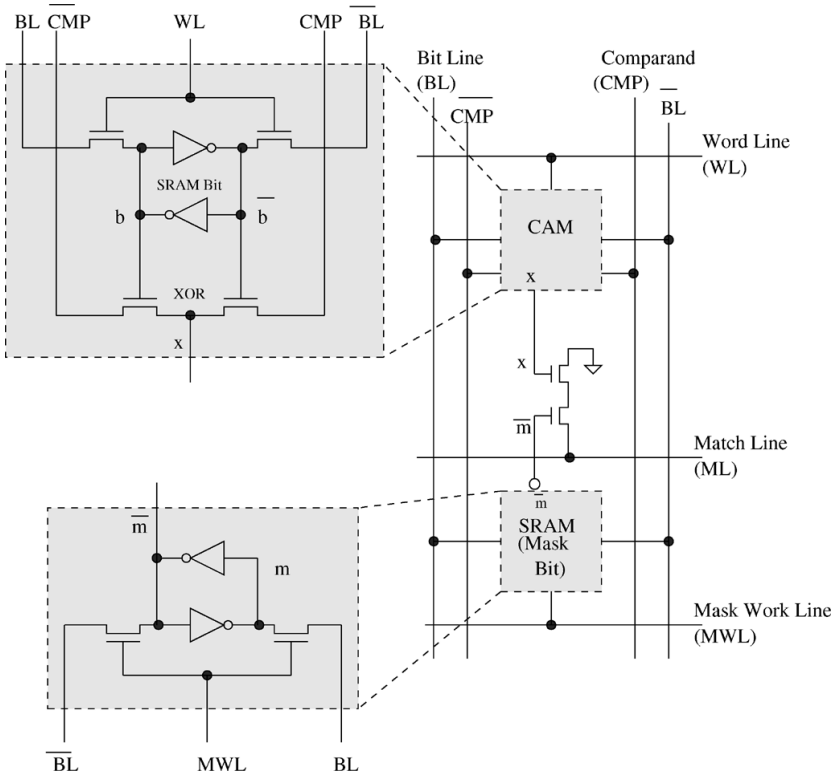



Fig. 2. Conventional TCAM cell.

The block diagram of a single TCAM cell is shown in Fig. 2. It is a straightforward SRAM based combination of a value bit (CAM) and a mask bit. A CAM bit is comprised of an SRAM cell and an XOR gate [15]. The XOR gate compares the content of the SRAM cell with the input comparand (given as the differential signal $CMP$ and $\overline{CMP}$). Mask bit is also an SRAM cell. When mask bit is at logic 0, the CAM bit is valid and participates in the comparison. When mask bit is at logic 1, the CAM bit is *don't-care* and the comparison operation generates a match regardless of the value of the CAM bit. A TCAM word of width $w$ (one line in Fig. 1) replicates such cell $w$ times, connected together through WL, MWL, and ML. ML is the single output of each TCAM word that is taken to the priority encoder to determine the match index (see Fig. 1).

At the beginning of every search operation, the Match Line is precharged to the VDD level. For each TCAM cell in a word, the input data bit to be searched (called *key* or *comparand*) is given as a differential value on $CMP$ and $\overline{CMP}$ lines. If every TCAM cell in a word either matches the comparand bit or is masked then match line stays charged, which indicates a word match. If for any cell in the TCAM word the mask bit is $0(\overline{m} = 1)$ and the CAM bit does not match the comparand, then the comparison circuit connects the match line to ground. That would discharge the match line that indicates a word mismatch. The above explanation can be symbolized as the following equation:

$$\overline{ML} = \sum_{i=0}^{w-1} x_i \cdot \overline{m_i}. \tag{1}$$

In this equation $x_i = d_i \oplus \mathrm{cmp_i}$ is the result of comparison between CAM cell content and comparand bit. The comparison logic of a TCAM word is a pseudo-NMOS implementation of (1) on element of which is shown in Fig. 2.

### C. Implementation Alternatives

CAM bit (cell) optimization for speed, power, and area is a very well-studied subject in the literature. The reference model discussed in Section II-B is based on CAM (TCAM) circuits presented in [1] ([15]). We chose this implementation of CAM cell due to its simplicity. However, as $x_i = d_i \oplus \mathrm{cmp_i}$ is the universal behavior of a CAM cell, any other implementation can also be employed in the PCAM architecture. The novelty of PCAM architecture is not in its CAM cells. Rather, it is in the way that the IP prefixes are encoded and architectured within a word. We will elaborate on this in Section III.

The CAM/TCAM cell design and optimization is not the focal point of our work. Yet, to be more informative, here we comment briefly on other alternatives and their pros and cons.

- **NOR- versus NAND-Type Match-Line Circuit:** The CAM cell shown in Fig. 2 belongs to the first category that uses a NOR-type circuit to generate the match-line. References [1], [21], and [29] are other examples of employing NOR-type circuit. Another alternative is to use a NAND-type match-line. Some examples, reported in literature, are [30] and [18]. NAND-type circuits are in general slower because the match-line drivers (transistors) belonging to a word are connected in series. But the search current in the match circuit and, thus, the overall power consumption is reduced. In spite of higher speed, the NOR-type match circuit may experience lower noise margins (e.g., $V_{\mathrm{dd}} - V_{\mathrm{th}}$ instead of $V_{\mathrm{dd}}$ when NMOS match-line drivers are used).

- **PMOS versus NMOS Drivers in CAM Cell:** In some implementations (e.g., [1], [18]), researchers used PMOS transistors for the comparison logic instead of NMOS. This design saves some dynamic power by limiting the voltage swing on ML between $V_{\rm dd}$ and $V_t$. It also saves some silicon area by minimizing the gap between $N+$ and $P+$ areas. But in order to achieve the same mobility as that of the NMOS transistor, the width of the PMOS transistor should be at least doubled due to the fact that PMOS has almost 50% less mobility than that of an NMOS counterpart. If the PMOS are sized the same as that of the NMOS comparison transistor, there will be an increase in the search time. We preferred to stay with the more standard NMOS comparison logic for this design.
- **NAND- versus XOR-Based TCAM Cell:** Our reference model of Fig. 2 used an XOR-based TCAM cell. Another widely known TCAM cell is the NAND-based design [16]. Like the XOR-based design, there are two SRAM cells plus four comparison transistors here. Only, the comparison logic is in the form of NAND logic and data is stored differently. If standard six-transistor (6T) SRAM cells are used, then the resulting TCAM cell has 16 transistors, just like the previous design. Authors in [16] employ 4T SRAM cells [17] for this cell to get 12T TCAM cells. Note that it is conceptually possible to use 4T SRAM cells for the XOR based design as well. The only clear functional difference between the two cells is that the NAND-based cell produces a mismatch regardless of the comparand value, when the cell contains an 11-bit combination. But it is more efficient to implement such functionality using a valid bit. The rest of this paper develops on the XOR-based TCAM cell design, as deriving our final equations from the XOR-based TCAM equations is more straightforward.

## III. PCAM

The design challenge ahead of us was that any $w$-bit prefix has to be encoded prior to being stored in a TCAM word that has only $w+1$ SRAM bits. The employed encoding method must need minimal logic for masked comparisons. The comparison logic is the overhead and must be small enough to preserve a major portion of the area released by eliminating the mask bits. The rest of this section is a step by step development of such a design. The design is called PCAM.

### A. The Prefix Encoding Method

Let $B[w-1:-1]=b_{w-1}\cdots b_0,b_{-1}$ be the array of SRAM cells storing a $w+1$ bits encoded bit-string for which the generic $w$-bit prefix is $P[w-1:0]=(V[w-1:0],M[w-1:0])$. The encoding method is defined by the following equation:[1]

$$b_i=\begin{cases} m_0, & \text{when } i=-1\\ 1, & \text{when } m_{i+1}m_i=11,\ i\in\{0,\cdots,w-2\}\\ 0, & \text{when } m_{i+1}m_i=01,\ i\in\{0,\cdots,w-2\}\\ v_i, & \text{when } m_i=0,\ i\in\{0,\cdots,w-1\}.\end{cases} \quad (2)$$

Since all the bits to the right of a masked bit in a prefix are also masked, the above definition conveys that a bit is masked if and only if all its right-hand bits are set to 1. Also, in the definition consider that if $p_i$ is masked and $p_{i+1}$ is not, then $b_i=0$ (i.e., the most significant masked bit is equal to zero). Considering this point, the above definition can be simplified for implementation as follows:

$$b_i=\begin{cases} m_0, & \text{when } i=-1\\ m_{i+1}+\overline{m}_i\cdot v_i, & \text{when } i\in\{0,\cdots,w-2\}\\ v_{w-1}, & \text{when } i=w-1.\end{cases} \quad (3)$$

For example, if the 8-bit prefix $P=10010*=(10010000,00000111)$ then $B=100100111$ (3 bits masked). A similar encoding method is used by [23] in a route lookup scheme without TCAM.

On the other hand, considering the above discussion, to convert an encoded prefix back to its conventional (value, mask) form, these equations must be used:

$$v_i=b_i\cdot\overline{m_i},\ i\in\{0,\cdots,w-1\} \quad (4)$$

$$m_i=\prod_{j=0}^{i}b_{j-1},\ i\in\{0,\cdots,w-1\}. \quad (5)$$

For instance, given a 9-bit encoded prefix $B=010110001$ the original decoded 8-bit prefix would be $P=(01011000,\ 00000001)=0101100*$ (1 bit masked).

### B. Circuit Design for the PCAM

Our steps toward a new design start from the reference TCAM circuit (see Fig. 2). In a TCAM word made by replicating that circuit $w$ times the match-line (ML) logic is given by (1).

After replacing all $m_i$ in (1) with the right side of (5) and applying DeMorgan's Law, we get

$$\overline{ML}=\sum_{i=0}^{w-1}x_i\cdot\sum_{j=0}^{i}\overline{b}_{j-1}. \quad (6)$$

Equation (6) describes a $w$-bit TCAM word that requires $w+1$ SRAM cells, instead of $2w$ in the reference model. For $w=8$, the required number of SRAM cells to implement (6), is 9 versus 16 in conventional TCAM (43.8% reduction). Also, this implementation translates to a total of 17.2% reduction in transistor usage. If $w$ grows beyond eight, the area gain starts falling to a point that it could even become negative. On the other hand, still more transistors can be saved by carefully factorizing the expression in (6), which gives us (7)

$$\begin{aligned}\overline{ML}=&x_0\cdot\overline{b}_{-1}+(x_1+x_2+x_3)\cdot(\overline{b}_{-1}+\overline{b}_0)\\&+x_2\cdot\overline{b}_1+x_3\cdot(\overline{b}_1+\overline{b}_2)+(x_4+x_5+x_6+x_7)\\&\cdot(\overline{b}_{-1}+\overline{b}_0+\overline{b}_1+\overline{b}_2+\overline{b}_3)+x_5\cdot b_4\\&+(x_6+x_7)\cdot(\overline{b}_4+\overline{b}_5)+x_7\cdot\overline{b}_6.\end{aligned} \quad (7)$$

Even after factorizing, eight is the optimum value for $w$, as our calculations and experimentations showed. Fig. 3 demonstrates the net-list resulted from (7). Word line ($WL$),

[1]The symbols $+$, $\cdot$, $\prod$, and $\sum$ are used as Boolean notations.

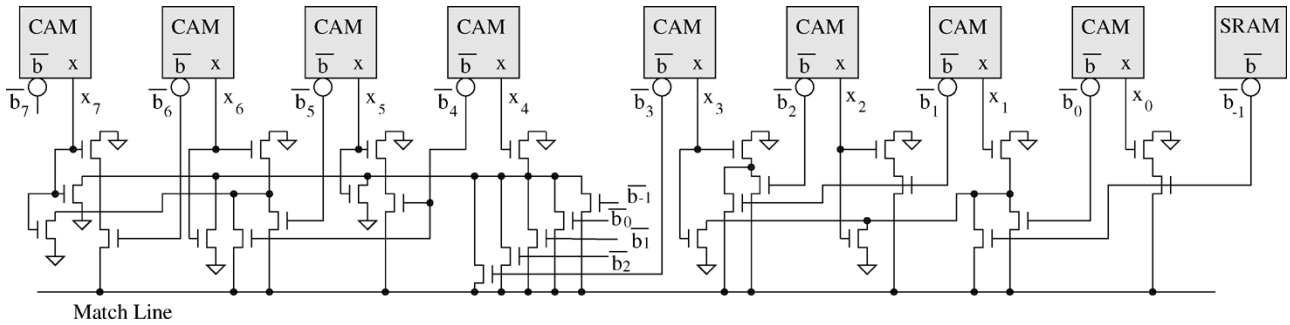



Fig. 3. PCAM *Cluster-9* that stores an 8-bit prefix. For clarity of the figure, word line $(W, L)$, bit lines $(BL[7 : -1], \overline{BL}[7 : -1])$, and comparand lines $(CMP[7 : 0], \overline{CMP}[7 : 0])$ are hidden.

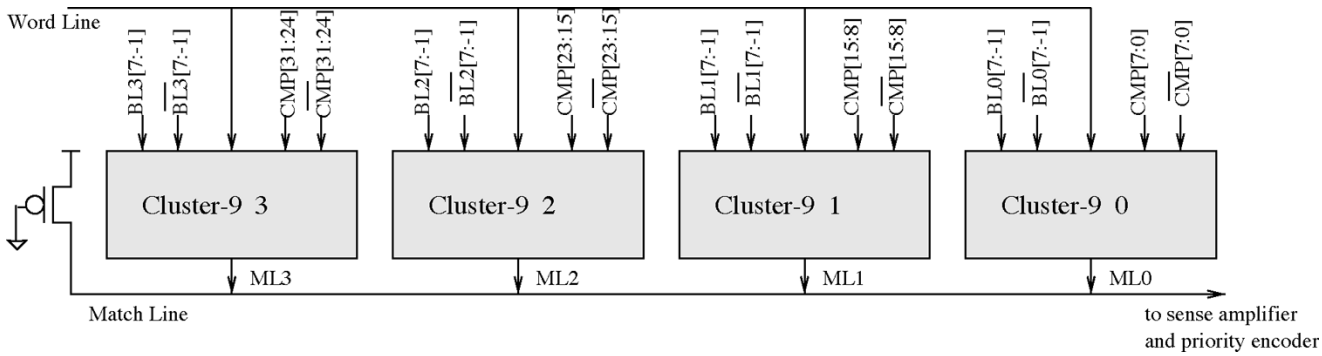



Fig. 4. Thirty-two-bit PCAM word.

bit lines $(BL[7 : -1], \overline{BL}[7 : -1])$, and comparand lines $(CMP[7 : 0], \overline{CMP}[7 : 0])$ are hidden in this figure to make it less crowded and easy to comprehend. The CAM and SRAM cells in this figure are the same as the CAM and SRAM cells in Fig. 2, only $x$ and $\overline{b}$ in that figure are here advertised as $x_i$ and $b_i$, respectively. Let us call this circuit *Cluster-9*. When implemented, (7) requires only 21 NMOS transistors. That results in a total of 99 transistors for a *Cluster-9*, which is 22.7% less than a conventional TCAM word of size 8. In a PCAM module, four of such clusters should be put together, connected only through match line and word line, to make room for a 32-bit IPv4 prefix. This is visualized in Fig. 4. For example, prefix $129.110.64/14 = (816\text{E}4000\text{H}, 00003\text{FFF}\text{H})$ would be encoded to four slices to be stored in four *Cluster-9*s, i.e., $B^0 = 011111111$, $B^1 = 010111111$, $B^2 = 011011100$, and $B^3 = 100000010$. IPv6 requires 16 *Cluster-9* units put together to make room for a single 128-bit prefix. Of course, both IPv4 and IPv6 forwarding engines can have elements other than address prefix in each entry, such as *Virtual Private Network Id* (VPN). These are typically exact (neither range nor prefix) fields. Additional *Cluster-9* units can be appended to each line to allocate such fields.

If the 6T SRAM cells in this design are replaced by 4T SRAM cells [17], the number of transistors in *Cluster-9* will be further reduced to 81. However, 4T SRAM technology has complex process and poor stability in low voltages [17], and a 6T SRAM cell is more standard in designs reported by both academia and industry. Therefore, in this paper we use *Cluster-9* with the original 6T SRAM cells.

### C. Encoder and Decoder Units

A TCAM part constructed around the PCAM idea, has to offer a conventional interface to the outside world. The prefixes written to and read from this TCAM should have the classic *(Value,Mask)* representation. Therefore, encoding and decoding are needed to convert the classic prefixes to encoded ones for writing and transforming the encoded words to $(V, M)$ pairs for reading.

1) *Encoding unit for write operations:* The encoder implements the set of (3). This unit should not be mistaken with the priority encoder unit that exists at the output stage of all TCAM modules to resolve multiple matches. When $w = 8$, four such blocks are required to implement a 32-bit prefix encoder. Fig. 5(a) shows such 4-block configuration. This configuration works directly in conjunction with the 32-bit PCAM word of Fig. 4. To revisit the example given at the end of Section III-B, *Encoder0* generates $B^0$ (to be stored in *Cluster9-0* of Fig. 4), *Encoder1* generates $B^1$ (to be stored in *Cluster9-1* of Fig. 4), and so on.

   Since a small combinational logic block combines prefix value and prefix mask into a single encoded word, the write operation can be done in one cycle. The same operation would require two cycles to fulfill, through the shared bit-line interconnects, in a conventional TCAM, one cycle to write the value and a second cycle to write the mask.

2) *Decoding unit for read operations:* Addressed read operation is not a primary feature expected from a TCAM device. TCAMs normally output their search result only. Yet,

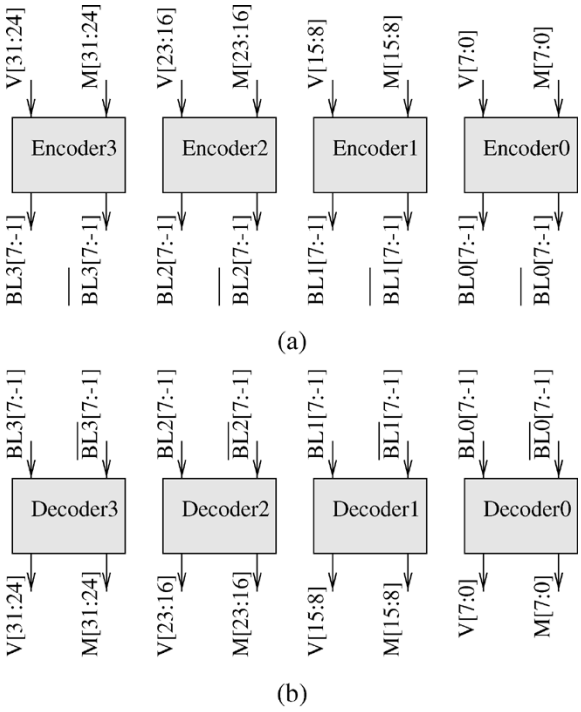



Fig. 5. Encoder and decoder blocks for a 32-bit PCAM module. (a) Encoder for 32-bit PCAM write operations. (b) Decoder for 32-bit PCAM read operations.

at times it may become handy to be able to read a particular location within a TCAM given the address of that location. In such cases, to convert the read prefix to its classic $(V, M)$ form for output, a decoder block is necessary.

The decoder implements (4) and (5) for the prefix value $(V)$ and mask $(M)$, respectively.

For a PCAM of width 32, when $w = 8$, four such decoder blocks are necessary to fully decode one word. Fig. 5(b) shows such preparation. All *Cluster-9* blocks of order 0 (as shown in Fig. 4) connect their bit lines to *Decoder0* which will produce $P[7:0] = (V[7:0], M[7:0])$, and so forth.

The encoded prefixes within PCAM can be read in one cycle. Furthermore, the decoder block implements simple combinational logic that can be done within the same cycle. Therefore, prefix reading can be performed in one cycle. The same operation would need two clock cycles in conventional TCAM with shared bit line, one cycle to read the value and a second cycle to read the mask.

Note that neither Encoder nor Decoder has any negative impact on common TCAM updating algorithms. PCAM fits in the basic TCAM architecture (Fig. 1), thus, complying with all the updating algorithms that pertain to this architecture. For examples of such updating algorithms see [24].

## IV. Hybrid Structure for Packet Classification

A PCAM module based on *Cluster-9* unit introduced in the previous section will have a granularity of eight. It means every data field must be truncated into 8-bit prefix chunks. Fortunately, this works for nearly all the header fields used commonly in packet classification applications. Table I lists the layer two, three, and four fields conventionally used for packet classification, and the way they are normally represented in the *rules table* [6]. As the table shows, all but one field value is commonly expressed as either complete value (exact match lookup), prefix, or range. A range is often converted to prefixes before being stored in any TCAM [6]. All the header fields listed in this table have a length which is a multiple of eight and all but one of them are stored as prefixes or exact values in TCAM. The only exception is the 8-bit *L4-Prtcl* (layer-4 protocol number) field in the IP header. Wild-cards can appear arbitrarily in any position of this field in the rules table. However, the number of occasions when such arbitrary wild-card is required is relatively small. The protocol field only appears in a 5-tuple rules table where the width of each rule is 104 bits, 96 bits of which is allocated by L3-DA (32 bits prefix), L3-SA (32 bits prefix), L4-DP (16 bits range), and L4-SP (16 bits range). Only 8 bits out of 104, or 7.7% of the rule bits, need to be arbitrarily maskable. Therefore, enough flexibility can be added to the architecture by adding a few TCAM bits to each PCAM line. In such hybrid PCAM module, each line is comprised of few TCAM bits attached to a long line of *Cluster-9* units. Arbitrary data fields are allocated on the TCAM portion and the rest of the rule elements use *Cluster-9* units for storage.

TABLE I
Some Header Fields That may be Used for IPv4 Packet Classification

| Field Name | Bit Width | Protocol Layer | lookup method |
|---|---|---|---|
| L2-SA | 48 | 2 (Ethernet) | exact |
| L2-DA | 48 | 2 (Ethernet) | exact |
| L3-Prtcl | 8 | 2 (Ethernet) | exact |
| Dst Addr | 32 | 3 (IP) | prefix |
| Src Addr | 32 | 3 (IP) | prefix |
| L4-Prtcl | 8 | 3 (IP) | arbitrary |
| Dst Port | 16 | 4 (TCP) | range |
| Src Port | 16 | 4 (TCP) | range |

A viable design option is 64 bits as the basic line width in which 8 bits are TCAM and 56 bits are PCAM (7 *Cluster-9* units). Adding TCAM bits to a PCAM line is straightforward. It is simply done by connecting them through the WL and ML lines. Of course, the TCAM bits would also need separate MWL interconnects. Fig. 6 illustrates the idea. Consequently, 12.5% of the hybrid PCAM chip would be occupied by conventional TCAM circuitry. Since 87.5% of the storage is PCAM, which is over 20% more dense than conventional TCAM, more than a 17% increase of device density is still gained. The resulting chip is capable of employing multidimensional packet classification applications.

## V. Experimental Results

### A. Word Implementation

To examine the merits of our approach in practice, layouts for a PCAM module and a TCAM module were implemented in Cadence [25]. The design rule was the Taiwan Semiconductor Manufacturing Company (TSMC) 0.18-$\mu$m library provided by Mosis [26] with $VDD = 1.8$ V. A 32-bit conventional TCAM word and a 32-bit PCAM word are demonstrated in Fig. 7(a) and (b), respectively. The layouts are shown in the same scale so that

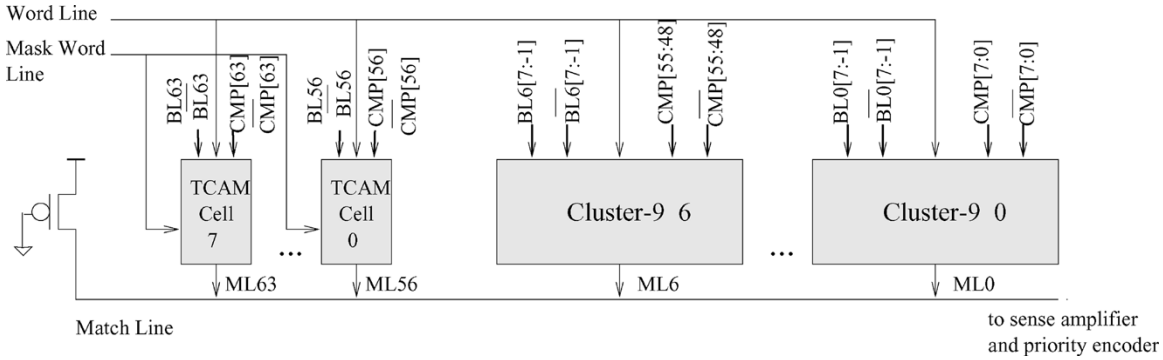



Fig. 6. Sixty-four-bit hybrid configuration of 8 TCAM bits and 56 PCAM bits.

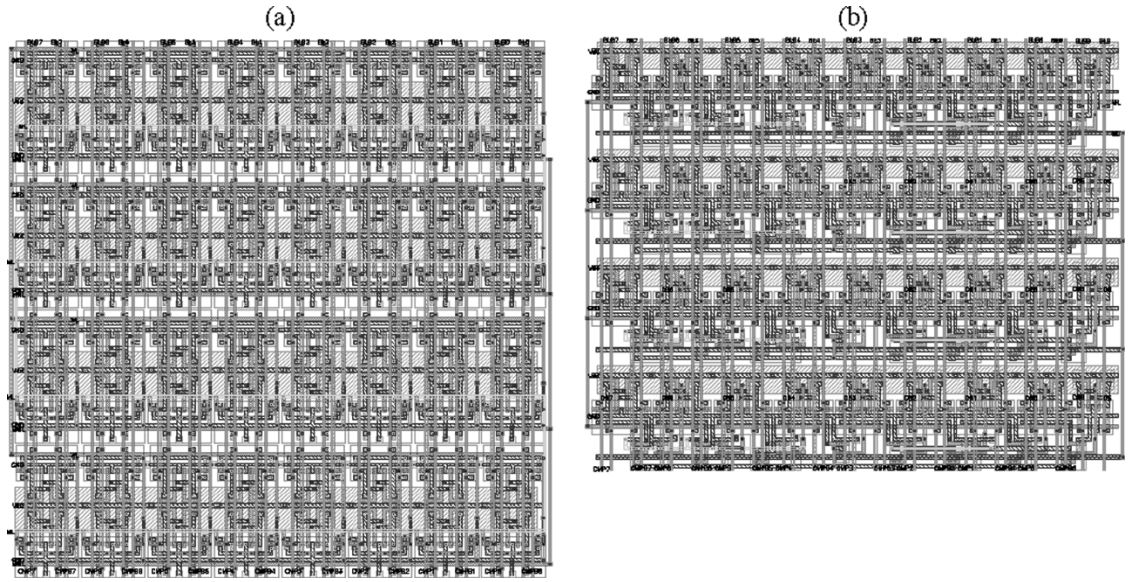



Fig. 7. Layouts for 32-bit TCAM and PCAM words: (a) 32-bit conventional TCAM word and (b) 32-bit PCAM word.

their size correspondence is preserved. To make it easy for the reader to compare them visually, the layouts are composed in a square shape formed of four similar rows. For the TCAM word, each row has eight cells. For the PCAM word, each row consists of a *Cluster-9* unit. The cell stability and noise tolerance issues are already incorporated in the layout design.

The NMOS and PMOS devices of the CAM cells in both layouts are sized similarly. The mask SRAM bits in TCAM are sized similar to the SRAM bits in CAM cells (see Fig. 2). In spite of differences in comparison logic, both TCAM (Fig. 2) and PCAM (Fig. 3) experience the same topology during the discharge of match line (ML). Note carefully that in PCAM (Fig. 3) if there is a mismatch, any path from ML to ground passes through only a pair of series NMOS transistors. In other words, the path during mismatch in both TCAM and PCAM structures will have only two minimum-sized transistors. Therefore, from implementation point of view, transistor sizing in both cases will be the same.

Table II summarizes one of the test scenarios used to verify the functionality and timing behavior of PCAM. An 8-bit prefix is used for simplicity. The experiment is visualized in Fig. 8, showing only the affected signals. Notice that for PCAM content complemented signals are shown, i.e., $\overline{b}_{-1}$, $\overline{b}_0$, $\overline{b}_1$, and $\overline{b}_2$. At the beginning of step one, the encoded prefix 010101011 (encoded form of the 6-bit prefix 010101*) is present in memory. The first two bits of the prefix are masked. At this step, the key 01010101 is placed on the comparand lines which generates a match (last line of the table). This can be observed at $t = 10$ ns on the waveforms of Fig. 8. To show that masked bits do not affect the search operation, step two toggles the first two bits of the key ($CMP_0$ and $CMP_1$). As expected, 01010110 also generates a match ($t = 20$ ns on the same figure). Step three changes the third bit of the key ($CMP_2$). The result will be a mismatch ($t = 30$ ns on the waveform) because this location is not masked. In step four, the prefix is changed to 010100111. Now three bits

TABLE II
DEMONSTRATION OF A PCAM WORD FUNCTIONALITY IN FOUR STEPS

| | Step 1 | Step 2 | Step 3 | Step 4 |
|---|---|---|---|---|
| 8-b prefix | 010101* | 010101* | 010101* | 01010* |
| 9-b encoded | 010101011 | 010101011 | 010101011 | 010100111 |
| key (*CMP*) | 01010101 | 01010110 | 01010010 | 01010010 |
| ML | HI | HI | LOW | HI |

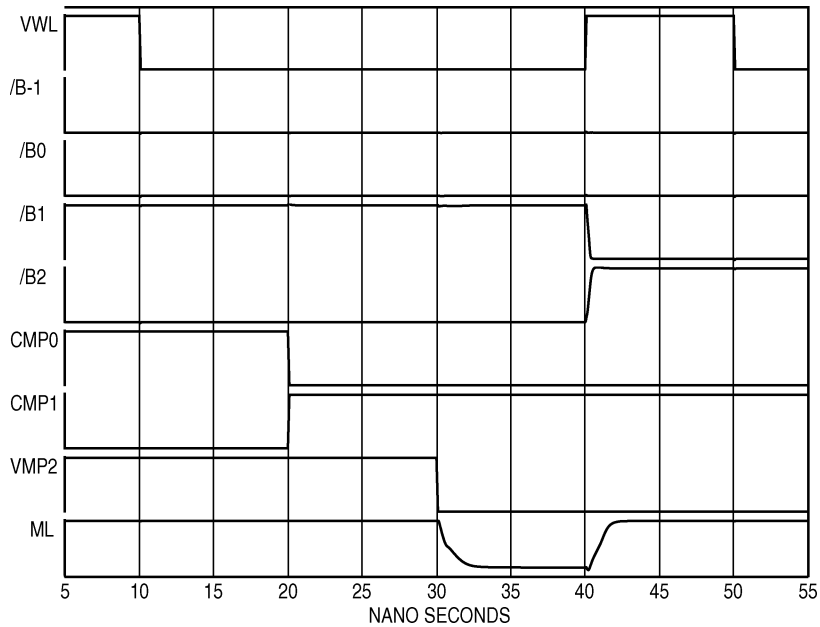



Fig. 8. Simulation waveforms for four test samples.

TABLE III
COMPARING PCAM AND THE CONVENTIONAL TCAM

| - | $-25^oC$ | | $25^oC$ | | $75^oC$ | |
|---|---|---|---|---|---|---|
| | PCAM | TCAM | PCAM | TCAM | PCAM | TCAM |
| $t_r[ns]$ | 0.33 | 0.35 | 0.56 | 0.59 | 0.94 | 1.01 |
| $t_f[ns]$ | 0.79 | 0.75 | 0.1.10 | 1.02 | 1.48 | 1.35 |
| $t_{plh}[ns]$ | 3.59 | 3.55 | 3.82 | 3.63 | 4.14 | 3.87 |
| $t_{phl}[ns]$ | 3.83 | 3.68 | 4.49 | 4.39 | 4.97 | 4.75 |

are masked instead of two, which will cause the same key in step three to generate a match ($t = 40$ ns on the waveform).

Extensive simulation was performed for different mask lengths and various sequences of input to compare the timing and power performance of PCAM against the reference TCAM. The worst case timing results are summarized in Table III for both PCAM and the reference model. This table shows the major delay factors: rise time $(t_r)$, fall time $(t_f)$, and propagation delays ($t_{\rm plh}$ and $t_{\rm phl}$), as well as the worst case dynamic power dissipation $(P_d)$ for three different temperatures. ML always drives a double standard inverter load. $V_{\rm OL\,max} \approx 0.24$ and $V_{\rm OH\,min} \approx 1.78$ for both TCAM and PCAM. Delays are all reported in nanoseconds and power dissipation is reported in milliwatts. $P_d$ is obtained by applying 400 random patterns with a 6-ns rate.

As the table shows, PCAM catches up with the reference model both in terms of speed and dynamic power consumption. Cutting the number of transistors does not have any noticeable negative effect on the performance of PCAM. Moreover, the reduced number of transistor causes a reduction in the average static power consumption due to leakage current, which is important in deep-submicrometer technologies where static power is the dominant factor in the overall power consumption and increases exponentially with the reduction of transistor size. PCAM static power consumption, due to current leakage, is at least 18% smaller than TCAM in all our observations. Although in most cases, the improvement was better than 24%.

TABLE IV
SIMULATION RESULTS FOR 32-BIT ENCODER AND DECODER BLOCKS

| Unit | $t_p$ [ns] | Size [Transistor] |
|---|---|---|
| Encoder | 0.84 | 310 |
| Decoder | 1.35 | 211 |

### *B. Encoder and Decoder Blocks*

The encoder and decoder blocks (Fig. 5) were evaluated separately because they are new units, which need to be justified in terms of size and performance. None of these blocks affects search performance. The encoder affects the critical path of write circuitry while the decoder affects the read critical path. The delays (in nanoseconds) and sizes (in number of MFET transistors) of both blocks are summarized in Table IV. The sizes of both blocks are negligible compared to the total size of a regular TCAM.

### *C. Layout*

The 2-kb modules are configured as 64 lines $\times$ 32 columns. For the purpose of visual comparison, both annotated layouts are shown in Fig. 9. Both layouts use three layers of metal in most parts. Metal 4 is used only in one occasion in each layout.

Table V summarizes the key metrics for our PCAM/TCAM structures, as well as three similar implementations reported in the literature. Due to differences in technology, size, cell-library, and design objectives (e.g., area, power), a direct comparison of these implementations is not possible. Specifically, as we explained in Section II-B, we chose a straightforward structure for the base CAM/TCAM cell (Fig. 2) to continue implementing PCAM architecture. We made no efforts to customize the CAM cell itself. Without such customization, as Table V shows, it is expected that the size of our TCAM/PCAM cells be larger than others (e.g., [21] or [31]).

Our main goals of showing Table V are: 1) to illustrate that our implementation has comparable size, performance, and

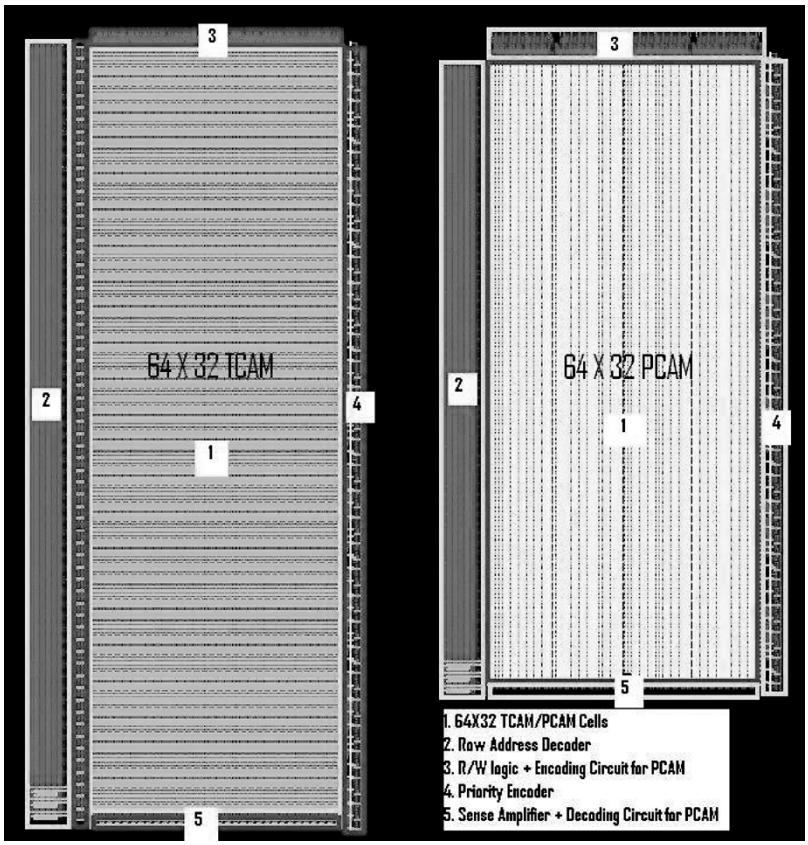



Fig. 9. Layouts for 2-Kb TCAM and PCAM modules.

TABLE V
COMPARING PCAM WITH PREVIOUS WORK FOR KEY METRICS

| *Comparing Metric* | *Our Architecture* | | Reference [29] | Reference [18] | Reference [21] |
|---|---|---|---|---|---|
| | *TCAM* | *PCAM* | *CAM* | *TCAM* | *CAM* |
| Technology | 0.18$\mu m$, 6 metals | 0.18$\mu m$, 6 metals | 0.18$\mu m$, 6 metals | 0.13$\mu m$, 7 metals | 0.35$\mu m$ CMOS |
| Chip Configuration | 2Kb | 2Kb | 36Kb | 9Mb | 3.6Kb |
| Match Line Architecture | NOR type | NOR type | NOR type | NAND type | NOR type |
| $V_{dd}$ | 1.8V | 1.8V | 1.8V | 1.0V | 1.5V-3.3V |
| # of Transistors (8 bits) | 128 | 99 | NA | NA | 56 |
| Cell Size for (8 bits) | 377.40$\mu m^2$ | 328.44$\mu m^2$ | NA | NA | 336$\mu m^2$ |
| Speed | 229.3Mhz @1.8V | 222.8Mhz @1.8V | 142Mhz @1.8V | 166Mhz @1.0V | 100Mhz @3.3V |
| Power | 6.2mW | 6.2mW | 94mW | 5W | 33mW |
| Power Performance | 13.20fJ/bit/search | 13.58fJ/bit/search | 17.95fJ/bit/search | 3.19fJ/bit/search | 86fJ/bit/search |
| Core Area | 0.132$mm^2$ | 0.111$mm^2$ | 4.83$mm^2$ | 148$mm^2$(with i/o) | 0.53$mm^2$ |

power and 2) to compare TCAM and PCAM full architectures (columns two and three) using the same base CAM cell. In particular, the core area statistics show that PCAM implementation consumes at least 16% less area while it achieves higher or comparable speed. Note that in terms of power performance metric, PCAM/TCAM implementation indicates higher values than those of [18]. This is because this reference has a specifically targeted low-power technique for its CAM/TCAM architecture while our TCAM/PCAM implementation is aimed at reducing the area, not the power.

The main area saving in PCAM is due to cell part and other units (address decoder, priority encoder, etc.) do not grow with the same rate as the cell part. Therefore, we expect to get more area saving for larger PCAM. We confirmed this by implementing 64-kb PCAM and TCAM modules. The 64-kb TCAM and PCAM module layout sizes are $A_{\mathrm{TCAM}} = 2.240 \times 1.975 = 4.424$ mm$^2$ and $A_{\mathrm{PCAM}} = 1.850 \times 1.924 = 3.559$ mm$^2$. Thus, the percentage of reduced area is $(A_{\mathrm{TCAM}} - A_{\mathrm{PCAM}})/A_{\mathrm{TCAM}} = 19.5\%$. This is quite close to the estimation obtained from counting the transistors.

## VI. CONCLUSION

This paper introduces PCAM design, which is a TCAM optimized for prefix storage and lookup applications. Such applications include high-speed Internet packet classification and forwarding, where lookup tables are huge and the appetite for higher storage density is always growing. To address this concern, our design removes the explicit mask bits from TCAM cells, hence, reducing the number of RAM cells in PCAM by

43.8% compared to a conventional design. Some logic is then added to compensate for the reduced number of memory cells. Eventually, the number of transistors is reduced by 22%. This approach also removes all mask word lines, which are long interconnects going to each and every TCAM word. PCAM structure can also reduce the static power consumption which is important for deep submicron technologies. Implemented layouts confirmed the increased device density for PCAM module compared to the conventional TCAM. All these improvements are achieved without sacrificing performance or functionality.

## REFERENCES


[1] H. Miyatake, M. Tanaka, and Y. Mori, "A design for high-speed low-power CMOS fully parallel content-addressable memory macros," *IEEE J. Solid-State Circuits*, vol. 36, no. 6, pp. 956–968, Jun. 2001.

[2] T. Pei and C. Zukowski, "Putting routing tables in silicon," *IEEE Network*, vol. 6, no. 1, pp. 42–50, Jan. 1992.

[3] A. McAuley and P. Francis, "Fast routing table lookup using CAMs," in *Proc. 12th Annu. Joint Conf. IEEE INFOCOM'93*, 1993, pp. 1382–1391.

[4] V. Fuller, T. Li, J. Yu, and K. Varadhan, Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy, RFC 1519, 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1519.txt

[5] M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, no. 2, pp. 8–23, Mar. 2001.

[6] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp. 24–32, Mar./Apr. 2001.

[7] J. Lutheren and A. Engbersen, "Multi-field packet classification using ternary CAM," *Electron. Lett.*, vol. 38, no. 1, pp. 21–23, Jan. 2002.

[8] J. Wade and C. Sodini, "Dynamic cross-coupled bitline content addressable memory cell for high density arrays," in *Proc. Int. Electron. Devices Meeting*, 1985, pp. 284–287.

[9] ——, "A ternary content addressable search engine," *IEEE J. Solid-State Circuits*, vol. 24, no. 4, pp. 1003–1013, Aug. 1989.

[10] F. Zane, G. Narlikar, and A. Basu, "CoolCAMs: Power-efficient TCAMs for forwarding engines," in *Proc. IEEE INFOCOM*, 2003, pp. 42–52.

[11] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," in *Proc. 11th IEEE Int. Conf. Netw. Protocols (ICNP'03)*, 2003, pp. 120–131.

[12] R. Panigrahi and S. Sharma, "Sorting and searching using ternary CAMs," *IEEE Micro*, vol. 23, no. 1, pp. 44–53, Feb. 2003.

[13] V. Ravikumar and R. Mahapatra, "TCAM architecture for IP lookup using prefix properties," *IEEE Micro*, vol. 24, no. 2, pp. 60–69, Apr. 2004.

[14] V. Srinivasan, B. Nataraj, and S. Khanna, "Methods for longest prefix matching in a content addressable memory," U.S. Patent 6 237 061, Jan. 5, 1999.

[15] R. Kempke and A. McAuley, "Ternary CAM memory architecture and methodology," U.S. Patent 5 841 874, Aug. 13, 1996.

[16] I. Arsovski, T. Chandler, and A. Sheikholeslami, "A ternary content-addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 155–158, Jan. 2003.

[17] C. Lage, J. Hayden, and C. Subramanian, "Advanced SRAM technology—The race between 4T and 6T cells," in *Proc. IEEE Int. Electron. Devices Meeting*, 1996, pp. 271–274.

[18] A. Roth, D. Foss, R. McKenzie, and D. Perry, "Advanced ternary CAM circuits on 0.13 $\mu$m logic process technology," in *Proc. Custom Integr. Circuits Conf.*, 2004, pp. 465–468.

[19] J. Rabaey, *Digital Integrated Circuits*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[20] H. Liu, "Routing table compaction in ternary CAM," *IEEE Micro*, vol. 22, no. 1, pp. 58–64, Jan./Feb. 2002.

[21] C.-S. Lin, J.-C. Chang, and B.-D. Liu, "A low-power precomputation-based fully parallel content-addressable memory," *IEEE J. Solid-State Circuits*, vol. 38, no. 4, pp. 654–662, Apr. 2003.

[22] A. P. J. Engbersen and J. van Lunteren, "Prefix-based parallel packet classification," Zurich Research Lab., Rueschlikon, Switzerland, Mar. 2000, IBM Res. Rep.

[23] H. Mohammadi, N. Yazdani, B. Robatmili, and M. Nourani, "HASIL: Hardware assisted software-based IP lookup for large routing tables," in *Proc. 11th Int. Conf. Netw. (ICON)*, 2003, pp. 99–104.

[24] D. Shah and P. Gupta, "Fast updating algorithms for TCAM," *IEEE Micro*, vol. 21, no. 1, pp. 36–47, Jan./Feb. 2001.

[25] *"Virtuoso Layout Editor Users Guide—Version 4.4.6,"* Cadence Design Systems Inc., San Jose, CA, 2000.

[26] The Mosis Service. Marina del Rey, CA [Online]. Available: http://www.mosis.com

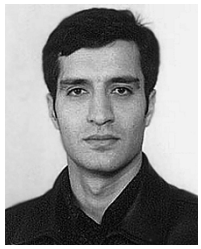
[27] *"User Manuals for SYNOPSYS Toolset Version 2002.06,"* Synopsys Inc., Mountain View, CA, 2002.

[28] M. J. Akhbarizadeh, M. Nourani, D. Vijayasarathi, and P. Balsara, "PCAM: A ternary CAM optimized for longest prefix matching tasks," in *Proc. IEEE Int. Conf. Comp. Des. (ICCD'04)*, 2004, pp. 6–11.

[29] K. Pagiamtzis and A. Sheikholeslami, "Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2003, pp. 383–386.

[30] F. Shafai, K. Schultz, G. Gibson, A. Bluschke, and D. Somppi, "Fully parallel 30-MHz, 2.5-Mb CAM," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1690–1696, Nov. 1998.

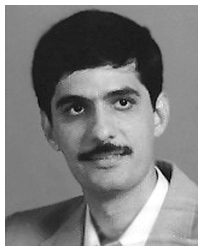
[31] P. F. Lin and J. B. Kuo, "A 0.8-V 128-Kb four-way set-associative two-level CMOS cache memory using two-stage wordline/bitline-oriented tag-compare (WLOTC/BLOTC) scheme," *IEEE J. Solid-State Circuits*, vol. 37, no. 10, pp. 1307–1317, Oct. 2002.

**Mohammad J. Akhbarizadeh** (S'01) received the B.Sc. degree in computer engineering from the University of Tehran, Tehran, Iran, in 1996. He received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Texas at Dallas, Richardson, in 2002 and 2005, respectively, where he focused on algorithms for hardware-based IP route lookup.

He is currently with Cisco Systems Inc., San Jose, CA. He is a member of the Center for Integrated Circuits and Systems at the University of Texas at Dallas. His research interests include all aspects of network processing systems.

Dr. Akhbarizadeh was a recipient of the TexTEC student research award in 2002 and the SIAWE scholarship award in 2004.



**Mehrdad Nourani** (S'91–M'94–SM'05) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 1986, and the Ph.D. degree in computer engineering from Case Western Reserve University, Cleveland, OH, in 1993.

In 1994, he was a Postdoctoral Fellow in the Department of Computer Engineering at Case Western Reserve University. From 1998 to 1999, he served in the Department of Electrical and Computer Engineering at the University of Tehran, Tehran, Iran and in the Department of Electrical Engineering and Computer Science at Case Western Reserve University. In August 1999, he joined the faculty of the University of Texas at Dallas, Richardson, where he is currently an Associate Professor of Electrical Engineering and a member of the Center for Integrated Circuits and Systems (CICS). He has published over 120 papers in numerous journals and refereed several conference proceedings. His current research interests include design for testability, system-on-chip testing, signal integrity modeling and test, application specific processor architectures, packet processing devices, high-level synthesis, and low-power design methodologies.

Dr. Nourani received the Texas Telecommunications Consortium Award (1999), The Clark Foundation Research Initiation Grant (2001), the National Science Foundation Career Award (2002), and the Cisco Systems Incorporated URP Award (2004). He is a member of the IEEE Computer Society and the ACM SIGDA.

**Deepak S. Vijayasarathi** received the B.Sc. degree in electronics and communication from SSN College of Engineering (SSNCE), University of Madras, Madras, India, in 2003, and the M.Sc. degree in electrical engineering from the University of Texas at Dallas, Richardson, in 2005, where he focused on hardware engines for packet processing architectures.

He is currently with Intel Corporation, Folsom, CA. He is a member of the Center for Integrated Circuits and Systems at the University of Texas at Dallas.

**Poras T. Balsara** (M'85–SM'95) received the L.E.E. diploma in electronics from The Victoria Jubilee Technical Institute, Bombay, India, in 1980, and the B.E. (electrical) degree from the University of Bombay, Bombay, India, in 1983. He received the M.S. and Ph.D. degrees from Pennsylvania State University, University Park, in 1985 and 1989, respectively.

He is currently a Professor of Electrical Engineering at the Erik Jonsson School of Engineering and Computer Science at The University of Texas at Dallas, Richardson. His research interests include, VLSI Design, circuits and systems for communications and signal processing, design of energy efficient digital circuits and systems, computer arithmetic, application-specific architecture design, and reconfigurable computing. He has published several journal and conference papers in these areas.