PCAM: A Ternary CAM Optimized for Longest Prefix Matching Tasks

Mohammad J. Akhbarizadeh, Mehrdad Nourani, Deepak S. Vijayasarathi, and Poras T. Balsara
Center for Integrated Circuits & Systems
The University of Texas at Dallas
Richardson, TX 75083
{eazadeh, nourani, dxv033000, poras}@utdallas.edu

***Abstract*— An optimized Ternary CAM concept is introduced for application in the longest prefix matching tasks of the Internet search engines. It employs $w+1$ RAM bits for a word of size w. A conventional TCAM needs $2w$ RAM bits for the same word size. Based on this concept an 8 bit Prefix-CAM cluster is designed out of 9 SRAM bits, four of which merge to store a 32-bit IPv4 prefix. A complete Prefix-CAM module employs 22% less transistors than a conventional TCAM, for equal storage size and equal functionality. We confirm the 22% area saving by implementing the layouts for Prefix-CAM and TCAM words. Our design also reduces interconnect area by reducing address decode lines.**

I. Introduction

Nowadays, more hardware architects than ever look into Content Addressable Memory (CAM) for high performance table lookup tasks [1], [2], [3]. Various types of CAM intellectual property cores are available in the market for rapid integration. CAD tools, design automation environments and FPGA based systems have become CAM-aware more 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. This gives TCAM the ability to store variable size data (called prefixes). TCAM can look up a given key in its entire contents to find all matching prefixes and is capable of finding the longest (i.e the most specific) match among those, all in one clock cycle [4]. This property has special application in high speed Internet routers where classification and forwarding engines need to find a longest matching prefix with wire-speed rates, in huge and ever growing lookup tables. While simplicity and high performance are the main reasons for designers to choose TCAM for such applications, high power dissipation and low storage density remain to be the two major concerns with this technology. This design primarily addresses the latter problem.

The rest of this paper is organized as follows. Section II starts with clarifying necessary assumptions and definitions that will be used throughout the paper. Then, conventional TCAM design is explained to be used as a reference model for comparison purposes. The motivation behind this work is also explained in the same section. Section III explains the main contribution of this paper. we first derive the optimized logic equations then describes the circuitry for our novel Prefix-CAM. Finally, two additional functional units necessary for this design are introduced. Section IV summarizes our experimental results and practical observation. We will conclude the paper in Section V.

II. Background

A. Assumptions and Definitions

In this paper, a prefix is assumed to be at most 32-bits long, which is the maximum required size for an IP version 4 (IPv4) routing table. However, this will not restrict our design to one application. Packet classification or IPv6 can also benefit from the optimized circuit introduced in this work. Also, we discuss the static TCAM where memory cells are SRAM, though most of the discussions and results are also valid for typical dynamic TCAMs.

An arbitrary prefix can be shown in binary radix as a string of 1s and 0s, followed by an * that marks all other bits at the right side of prefix as *don't-care* (to be masked). 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=(98A00000H, 001FFFFFH). In the current example, $p_{30} = (0,0)$, $p_{31} = (1,0)$, and $p_0 = (0,1)$.

B. The Reference Model

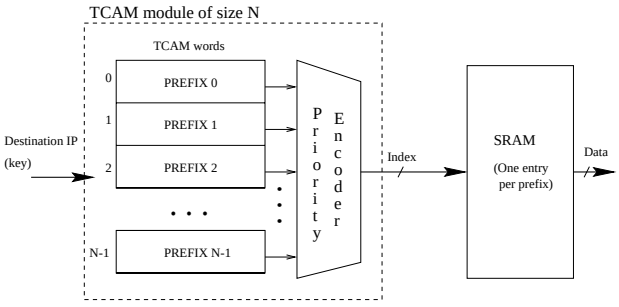


Fig. 1. Basic Structure of a TCAM module.

Figure 1 shows the basic structure of a TCAM used in many of the state of the art Internet forwarding and classification engines. In this structure, the entries are sorted based on the prefix length. Longer prefixes are placed in memory locations with higher addresses. A given destination address to be searched, 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 longest prefix 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 a SRAM module that has N entries, one entry for each TCAM prefix.

The block diagram of a single TCAM bit is shown in Figure 2. It is a straight-forward SRAM based combination of a Value Bit (CAM) and a Mask Bit. CAM bit itself is comprised of a SRAM cell and an XOR gate. 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 a SRAM cell. When Mask Bit is at logic 1, the CAM Bit is valid. When Mask Bit is at logic 0, the CAM Bit is *don't-care*. A TCAM word of width w (one line in Figure 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 Figure 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 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:

$$\overline{ML} = \sum_{i=0}^{w-1} x_i \cdot \overline{m_i} \qquad (1)$$

In this equation $x_i = d_i \oplus cmp_i$ 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 Equation 1 on element of which is shown in Figure 2. Despite some variations that try to improve power dissipation, search speed, flexibility, or density, the same design concept is used by most of the respective works, as observed by the authors. Examples can be found in [1], [4], [5], and [6].

C. Motivation and 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 each cell. Each TCAM cell requires 16 transistors [5] (14 in some literature [6]), as opposed to 6 for SRAM or 2 for DRAM [7]. This has inspired some researchers to offer heuristics that optimize TCAM usage [8]. In the state of the art Ternary CAM technology, any bit in a word can be masked independently. This flexibility comes at a cost. Each cell includes two SRAM (DRAM) bits to be able 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 for high density, fast, low power TCAM products is in the classification and forwarding engines of broadband Internet routers . The absolute

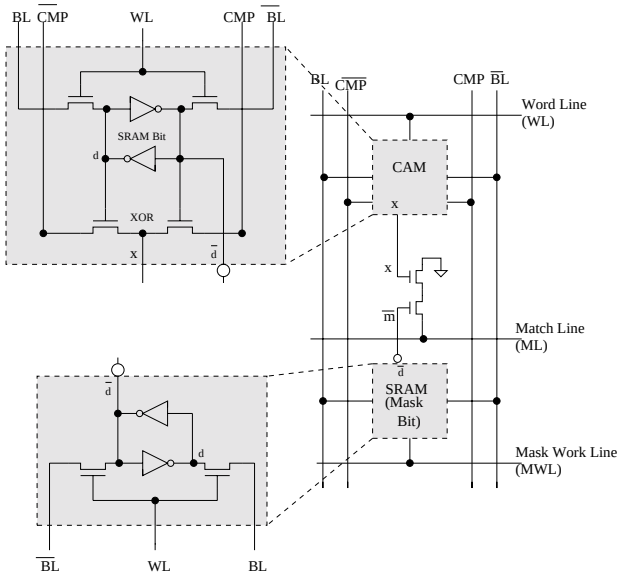


Fig. 2. Conventional TCAM cell.

majority of such applications need to store and search for prefixes, mostly IP (Internet Protocol) prefixes [9], [10]. All of the masked bits in a prefix are adjacent and are gathered at the right side. For example, 1100110010* is a 10 bit IP prefix shown in binary. The 22 lower bits of this prefix are masked. On the other hand, 11x01* is not a valid prefix because it has a masked bit surrounded by valid bits.

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 each word (see Figure 2). On the other hand, all the valid combinations of w bit IP prefixes accumulate to $2^{w+1}-1$ values (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, i.e. the way prefixes are stored and masked comparison is performed (shown in Figure 2). However, the general TCAM architecture (shown in Figure 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 Prefix-CAM.

III. The Prefix CAM

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 design. The design is called Prefix-CAM, or PCAM for short.

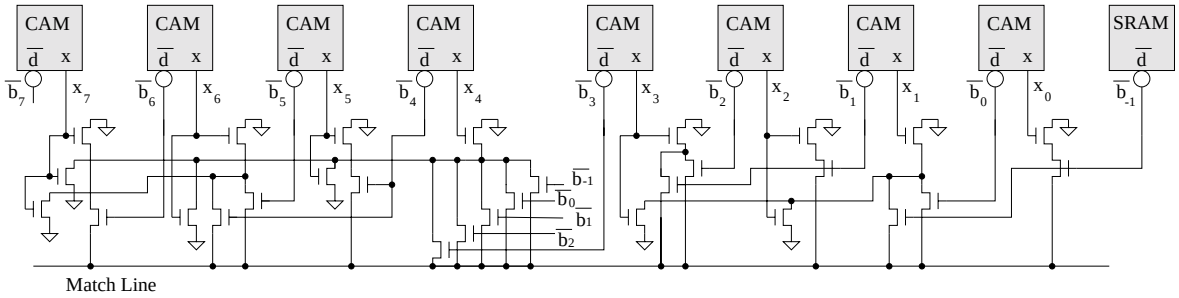


Fig. 3. PCAM Cluster-9 that stores a 8-bit prefix. For clarity of the figure, word line (WL), bit lines ($BL[7:-1], \overline{BL}[7:-1]$), and comparand lines ($CMP[7:0], \overline{CMP}[7:0]$) are hidden.

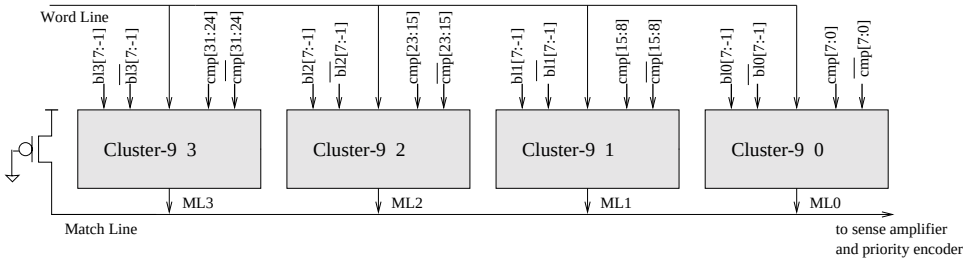


Fig. 4. 32-bit PCAM word.

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:

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

Since in a prefix, all the bits to the right of a masked bit are also masked, the above definition means a bit is masked when 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). On the other hand, if the 9-bit encoded prefix $B = 010110001$ is given then the above definition tells us that our decoded 8-bit prefix would be $P = (01011000, 00000001) = 0101100*$ (1 bit masked). A similar encoding method is used by [11] in a route lookup scheme without TCAM.

B. Circuit Design for the Prefix-CAM

We start our steps towards the new design from the reference TCAM circuit (see Figure 2). In a TCAM word made by replicating that circuit w times the match-line (ML) logic is given by Equation 1. Now notice that the encoded mask logic as explained in the previous subsection is[1] :

$$m_i = \prod_{j=0}^{i} b_{j-1},\ i \in \{0,\cdots,w-1\} \quad (4)$$

After replacing all m_i in Equation 1 with the right side of Equation 4 and applying DeMorgan's Law we get:

$$\overline{ML} = \sum_{i=0}^{w-1} x_i \cdot \sum_{j=0}^{i} \overline{b}_{j-1} \quad (5)$$

For $w = 8$, the required number of SRAM cells to implement Equation 5 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 8 the area gain starts falling to a point that it even becomes negative. On the other hand, still more transistors can be saved by carefully factorizing the expression in Equation 5, which gives us Equation 6:

$$\begin{aligned} \overline{ML} = \quad & x_0 \cdot \overline{b}_{-1} + (x_1 + x_2 + x_3) \cdot (\overline{b}_{-1} + \overline{b}_0) \\ & + x_2 \cdot \overline{b}_1 + x_3 \cdot (\overline{b}_1 + \overline{b}_2) + (x_4 + x_5 \\ & + x_6 + x_7) \cdot (\overline{b}_{-1} + \overline{b}_0 + \overline{b}_1 + \overline{b}_2 + \overline{b}_3) \\ & + x_5 \cdot b_4 + (x_6 + x_7) \cdot (\overline{b}_4 + \overline{b}_5) + x_7 \cdot \overline{b}_6 \end{aligned} \quad (6)$$

Even after factorizing, 8 is the optimum value for w, as our calculations and experimentations showed. Figure 3 demonstrates the net-list resulted from Equation 6. Word line (WL), 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 Figure 2, only x and $\overline{d}$ in that figure are here advertised as x_i and b_i,

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

respectively. Let us call this circuit *Cluster-9*. When implemented, Equation 6 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, 4 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 Figure 4. For example, prefix 129.110.64/14=(816E4000H,00003FFFH) would be encoded to four slices to be stored in four Cluster-9s, i.e. $B^0 = 011111111$, $B^1 = 010111111$, $B^2 = 011011100$, and $B^3 = 100000010$.

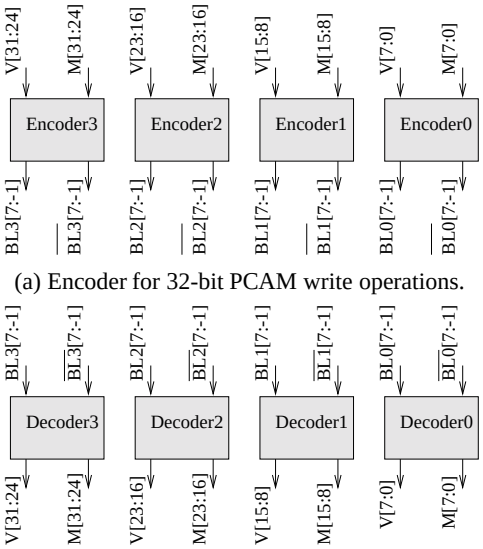
C. Encoder and Decoder Units

A TCAM part constructed around the Prefix-CAM 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 Equations 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. Figure 5(a) shows such 4-block configuration. This configuration works directly in conjunction with the 32-bit PCAM word of Figure 4. To revisit the example given at the end of Section III-B, *Encoder0* generates B^0 (to be stored in *Cluster0* of Figure 4), *Encoder1* generates B^1 (to be stored in *Cluster1* of Figure 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, at times it may come handy to be able to read a particular location within TCAM given the address of that location. In such case, to convert the read prefix to its classic (V,M) form for output, a decoder block is necessary.
   The decoder implements Equation 4 for the prefix mask (M). The prefix value (V) is then obtained from the equation:

$$v_i = b_i \cdot \overline{m_i},\ i \in \{0,\cdots,w-1\}. \qquad (7)$$

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



(a) Encoder for 32-bit PCAM write operations.

(b) Decoder for 32-bit PCAM read operations.

Fig. 5. Encoder and Decoder blocks for a 32-bit PCAM module

   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 (Figure 1) thus complying with all the updating algorithms that pertain to this architecture. For examples of such updating algorithms see [12].

IV. Experimental Results

A. Layout Implementation

Layouts for a PCAM word as well as an equivalent TCAM word were implemented using Cadence [13]. The design rule was the *Taiwan Semiconductor Manufacturing Company* (TSMC) 0.18 micron library provided by Mosis [14] with $VDD = 1.8v$. Figures 7(a) and 7(b) show the layouts for an 32-bit conventional TCAM word and a 32-bit PCAM word composed of four Cluster-9 units, respectively. The layouts are shown in the same scale so that 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 8 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. Both layouts use 2 layers of metal in most parts. Metal 3 is used only in one occasion in each layout. The PCAM layout in Figure 7(b) is an effort to meet the same ratio of area saving as estimated by counting the number of transistors. A close look at this layout shows that given more effort, it would be possible to get a tighter layout that saves even more area than the initial estimation. 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 Figure 2). The NMOS comparison logic transistors in PCAM (the transistors shown in Figure 3) have the same size as the comparison logic transistors in TCAM (the transistors that connect ML to ground in Figure 2). The TCAM layout size is $A_{TCAM} = 65.70 \times 65.50 = 4303.35nm^2$ and the PCAM layout size is $A_{PCAM} = 65.15 \times 51.38 = 3335.68nm^2$. Thus, the percentage of reduced area is $\frac{A_{TCAM} - A_{PCAM}}{A_{TCAM}} \times 100 = 22.48\%$. This confirms the estimation obtained by counting the transistors.

TABLE I

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

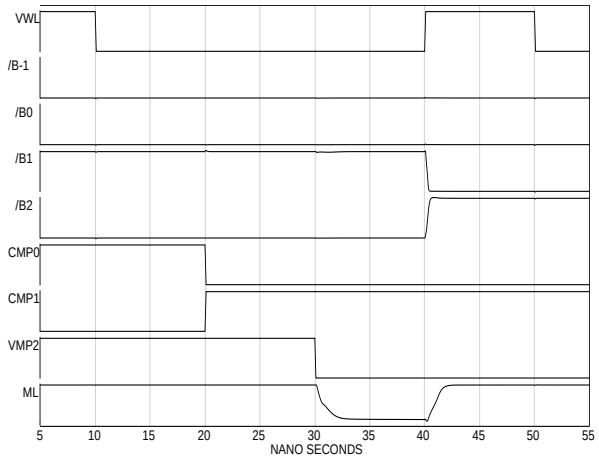


Fig. 6. Simulation waveforms for four test samples.

Table I 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 Figure 6, 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 1 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 $t = 10ns$ on the waveforms of Figure 6. To show that masked bits do not affect the search operation, step 2 toggles the first 2 bits of the key (CMP_0 and CMP_1). As expected, 01010110 also generates a match ($t = 20ns$ on the same figure). Step 3 changes the third bit of the key (CMP_2). The result will be a mismatch ($t = 30ns$ on the waveform) because this location is not masked. In step 4 the prefix is changed to 010100111. Now 3 bits are masked instead of 2, which will cause the same key in step 3 to generate a match ($t = 40ns$ 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 II, for both PCAM and the reference model. This table shows the

TABLE II

COMPARING PCAM AND THE CONVENTIONAL TCAM.

-	-25^oC		25^oC		75^oC	
	PCAM	TCAM	PCAM	TCAM	PCAM	TCAM
$t_r[ns]$	1.23	1.12	1.56	1.41	2.01	1.89
$t_f[ns]$	2.25	2.06	2.60	2.34	3.11	2.65
$t_{plh}[ns]$	0.57	0.61	0.71	0.78	0.94	1.07
$t_{phl}[ns]$	1.00	0.93	1.24	1.15	1.54	1.42
$P_d[uw]$	119.3	111.5	102.4	94.5	91.2	85.1

TABLE III

SIMULATION RESULTS FOR ENCODER AND DECODER BLOCKS.

Unit	t_p [ns]	Size [MFET]
Encoder	0.84	310
Decoder	2.35	211

major delay factors: rise time (t_r), fall time (t_f), and propagation delays (t_{plh} and t_{phl}) as well as the worst case dynamic power dissipation (P_d) for three different temperatures. ML always drives a double standard inverter load. $VOL_{max} \approx 0.24$ and $VOH_{min} \approx 1.78$ for both TCAM and PCAM. Delays are all reported in nano-seconds and power dissipation is reported in micro-Watts. P_d is obtained by toggling all the comparand lines every 4 nanoseconds in a way that the match line altered in every period.

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. In deep submicron technologies, the static power consumption is the dominant factor in the overall power and increases exponentially with the reduction of transistor size. Therefore, PCAM has the important advantage of less static power consumption in deep sub-micron technologies.

B. The Encoder and Decoder Blocks

The encoder block of Figure 5(a) was modeled in VHDL. The model was simulated and synthesized using Synopsys Design Analyzer tool [15] and 0.18 micron TSMC library from Artisan Components, Inc. So was the decoder block of Figure 5(b). 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 nanosecond) and sizes (in number of MFET transistors) of both blocks are summarized in Table III. The sizes of both blocks are negligible compared to the total size of a regular TCAM (see Section IV-C for more details).

C. Estimation of Conserved Area

- **Area preserved due to reduced number of devices:** Most of the area on a TCAM layout is occupied by the TCAM words. Other units such as the sense amplifiers, priority encoder, and IO buffers occupy less than 5% of the die area. The area occupied by the two additional PCAM units Encoder and Decoder is only a miniscule portion of

(a) 32-bit conventional TCAM word (b) 32-bit PCAM word

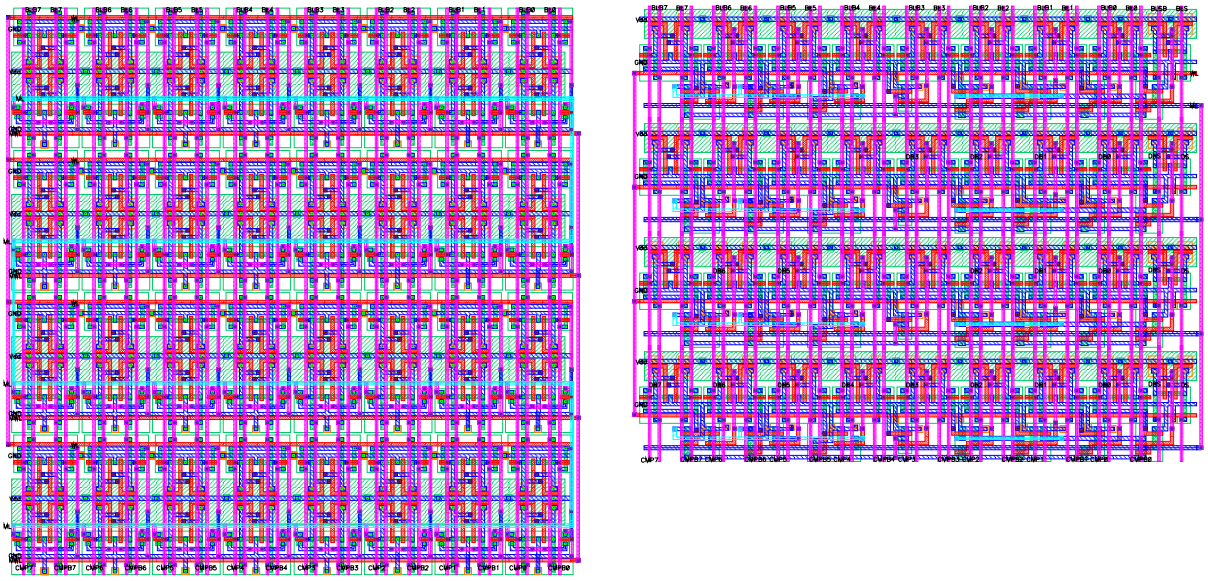


Fig. 7. Layouts for 32-bit TCAM and PCAM words.

this 5%. Therefore, the area saving for one TCAM word can be reflected closely in a large TCAM module fabrication. Our estimations show that a reasonably designed 32K PCAM layout can achieve at least 21% area reduction compared to a TCAM with the same size.

- **Area preserved due to reduced address decode interconnects:** There are two address decode interconnects for each individual TCAM word, one to control data read and writes (often called the word line) and one to control mask read and writes (often called mask word line). Because PCAM eliminates the mask bit, the mask word line is not needed anymore. That accounts for half of the address decode interconnects. Therefore, we expect a VLSI implementation of PCAM to have 50% less address decode interconnects than conventional TCAM.

V. CONCLUSION

This paper introduces Prefix CAM design, which is a ternary CAM 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 and thus the layout area are 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 sub-micron technologies. All these improvements are achieved without compromising performance or functionality, compared to the conventional TCAM.

REFERENCES

[1] H. Miyatake, M. Tanaka, and Y. Mori, "A Design for High-Speed Low-Power CMOS Fully Parallel Content-Addressable Memory Macros," *IEEE Journal of Solid-State Circuits, vol. 36, no. 6*, June 2001.

[2] T. Pei and C. Zukowski, "Putting Routing Tables in Silicon," *IEEE Network Magazine*, January 1992.

[3] A. McAuley and Paul Francis, "Fast Routing Table Lookup Using CAMs," *IEEE INFOCOM'93*, March 1993.

[4] V. Srinivasan, B. Nataraj, and S. Khanna, "Methods For Longest Prefix Matching In a Content Addressable Memory," *US Patent 6,237,061*, January 1999.

[5] R. Kempke and A. McAuley, "Ternary CAM Memory Architecture and Methodology," *U.S. Patent no. 5,841,874*, August 1996.

[6] 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 Journal of Solid-State Circuits, vol. 38, no. 1,*, January 2003.

[7] J. Rabaey, *Digital Integrated Circuits*, Prentice Hall, 1996.

[8] H. Liu, "Routing Table Compaction in Ternary CAM," *IEEE Micro*, January, February 2002.

[9] M. Ruiz-Sanchez, E. Biersack and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network Magazine*, March 2001.

[10] A.P.J. Engbersen, J. van Lunteren, "Prefix-Based Parallel Packet Classification," *IBM Research Report*, Zurich Research Laboratory, March 2000.

[11] H. Mohammadi, N. Yazdani, B. Robatmili, and M. Nourani, "HASIL: Hardware Assisted Software-Based IP Lookup for Large Routing Tables," in *International Conference on Networks (ICON)*, September 2003.

[12] D. Shah and P. Gupta, "Fast Updating Algorithms for TCAMs," *IEEE Micro*, February 2001.

[13] Cadence Design Systems Inc., "Virtuoso Layout Editor Users Guide - Version 4.4.6," June 2000.

[14] The Mosis Service., *http://www.mosis.com.*

[15] Synopsys Inc., "User Manuals for SYNOPSYS Toolset Version 2002.06," 2002.