A Reconfigurable CAM Architecture for Network Search Engines

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

ABSTRACT

A novel reconfigurable content addressable memory, called RCAM, is proposed that supports on-the-fly reconfiguration between CAM and TCAM. The area overhead of the proposed RCAM cell is only 5.6% when compared to conventional TCAM. This overhead is compensated by area saving due to removal of the priority encoder. Other features of our architecture include reconfigurability, and better overall performance and power. To achieve these we incorporated two novel techniques: (i) a hybrid CAM/TCAM architecture that allows user to pre-define CAM/TCAM cell behavior in each bit or word position and ultimately curtail the overall power consumptions of memory unit; and (ii) a wired-AND technique by which we can completely eliminate the sorting requirement and thus significantly reduce the update time. A 4*Kb RCAM architecture was implemented using* 0.18*μm CMOS technology. The simulations indicate a search time of* 6.15*ns, i.e. capability of handling about five OC-192 at wire speed.*

I. INTRODUCTION

Packet forwarding in network search engines is the process by which the next hop address of every incoming packet is determined based on the final destination address of the packet. Packet forwarding for Classless Inter Domain Routing (CIDR) environment requires finding the longest prefix that match the destination address. This is due to the fact that CIDR advocates variable prefix length.

A. Prior work

Several software and hardware schemes have been proposed earlier for longest matching prefix (LMP) like trie based lookup, Patricia tree, prefix expansion and various Content Addressable Memory (CAM) based packet forwarding [1] [2] [3]. All the above schemes mainly target on improving the overall search time which is often achieved at a cost of increasing the forwarding table update time.

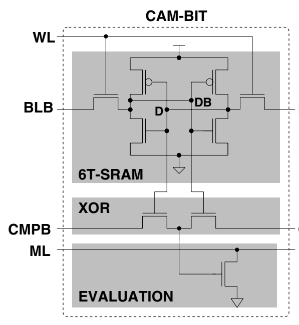
Nowadays, more hardware architects than ever look into CAM for high performance table lookup tasks [4] [5] [6]. A specifically interesting type of CAM, called Ternary CAM (TCAM) can store *don't-care* values in addition to 0's and 1's. Using this capability, the TCAM entries can include wild-cards. Because of 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. While simplicity and high performance are the main reasons for designers to choose TCAM for hardware-based search applications, high power dissipation, low storage density and sorting requirement remain to be the major concerns with this technology, which makes it a hot topic of ongoing research in both industry and academia. A few recent works are presented in [7], [8], [9] and [10].

In TCAM based implementation the replacement of any prefix in the forwarding table will take $O(n)$ shifts (for a table of n entries) to place the new entry in its correct location. This is because the forwarding table entries should be sorted in order wherein the longest prefix has the highest priority. This may lead to a very high update time and stalls the packet forwarding process. One technique to alleviate this problems is to leave predetermined empty spaces between the table entries so that any new entry to the forwarding table can directly be placed in those empty spaces. The drawback of such a technique is waste of memory space. Moreover, the main problem does not go away. It is simply reduced from global sorting to local sorting which makes the performance of system dependent on the traffic. To have a consistent throughput, regardless of traffic, the update time of the forwarding table should be guaranteed to be as small as possible [11].

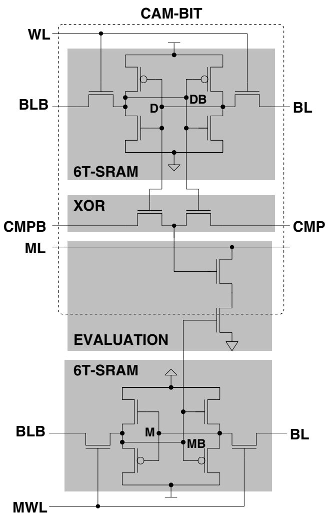
Kobiyashi et al. proposed a Vertical Logical operation with Mask-encoded Prefix (VLMP) based search engine for wire-speed packet processing of an OC-192 link [12]. This architecture expands upon a CAM architecture but is not made up of actual CAM cells. It uses registers and combinational logic to achieve CAM-like functionality. This architecture also allows random storage of forwarding table entries, thus reducing the updation time. The main disadvantage of this architecture is that it cannot get the same prefix search time as that of the conventional TCAM based architecture and also its area overhead compared to conventional TCAM architecture is very high. Another technique which targets update time reduction is PLO_OPT algorithm [13]. This algorithm places all unused entries in the center of the TCAM such that the first half of longest prefixes are always above the free space and next half are always below the free space. Addition or deletion of any new prefix would have to swap at most $n/2$ memory entries, where n is the number entries stored. The drawback of such an implementation is the time taken during swap of the entries and the unused TCAM space.

B. Main Contribution

We propose reconfigurable CAM (RCAM) cell that has the ability to selectively switch its functionality between a binary CAM (BCAM or simply CAM) and ternary CAM (TCAM) for any bit position. The configurability feature makes the contribution of our architecture twofold. First, it allows a user to configure it as a full CAM (e.g. of size $2n \times w$ bits) CAM, a full TCAM (of size $n \times w$ bits) or a hybrid module (of size $k \times w$ bits, where $n \leq k \leq 2n$). In the hybrid



(a) CAM cell.



(b) TCAM cell.

Figure 1. Conventional binary and ternary CAM structures.

mode, the behavior (CAM or TCAM) can be defined for each bit or word position which is an attractive option for power minimization of TCAM-based search engines. Second, the *masking* operation in conventional TCAM is replaced by a novel wired-AND operation. Effectively, our wired-AND technique replaces a *longest matching prefix* operation with an *exact matching mask* operation. Applying the wired-AND technique completely eliminates the need for a priority encoder and the sorting requirement during updation. Elimination of these two means significant reduction in power, cost and increasing the overall throughput. All of these are achieved with negligible overhead, i.e. 5.6% more than the conventional 9 transistor (9T) TCAM unit of the same size and performance.

II. Background

A. Binary CAM Cell

Figure 1(a) shows the basic cell structure of a conventional binary CAM. It consists of three parts - a 6T SRAM cell, 2T XOR logic and 1T Evaluation logic. The data (D and DB) is read/written into the SRAM bit of the CAM cell using the bit lines (BL and BLB). The search key is given through the comparand lines (CMP and $CMPB$) and will be fed to two NMOS transistors that behave like an XOR gate. The word line (WL) is activated whenever we want to read/write any data into the SRAM bit. The match line (ML) is precharged for every search operation. If there is a match, i.e if the incoming comparand bit is same as that of the stored data bit, then the charge of the match line (ML) is retained. Otherwise, ML discharges through the path created by the mismatch. Overall, the boolean function of ML can be expressed as:

$$ML = \overline{(D \oplus CMP)} \tag{1}$$

where, D and CMP are the data and comparand bits, respectively.

B. Ternary CAM Cell

Figure 1(b) shows the basic TCAM cell structure. The main difference between binary CAM and ternary CAM is the use of additional "don't-care" bits. Such additional state enables TCAM to perform partial match of the word as opposed to the binary CAM which exercises only the exact match between the comparand and data bits. Typically, a ternary CAM is a 16T structure and consists of three parts - a 9T CAM bit, a 6T SRAM bit and 1T evaluation logic. The bit lines (BL and BLB), comparand lines (CMP and $CMPB$), word line (WL) and match line (ML) have the same functionality as that of binary CAM. The mask word line (MWL) is activated whenever we want to read/write the mask bit. Throughout this paper we assume the mask bit $M = 1$ indicates a *don't-care* bit. Thus, the corresponding bit is *masked* as it should not affect the result of comparison. The boolean equation for the match line in ternary CAM can be written as:

$$ML = M + \overline{(D \oplus CMP)} = \overline{(D \oplus CMP) \cdot MB} \tag{2}$$

where, D is the data bit, CMP is the comparand bit and M and MB are the mask bit and its complement, respectively. Equation 2 also shows that the two transistors in the evaluation block (shown also as **E** in some figures in this paper) form a NAND gate whose two inputs are $D \oplus CMP$ and MB.

III. Reconfigurable CAM Cell

Reconfigurable Content Addressable memory (RCAM) is an 18T TCAM cell targeting reconfiguration and reusability. Figure 2 shows the basic structure of a RCAM cell. It is made up of two 9T CAM cells along with one extra reconfiguration transistor whose gate is controlled by R. In general, the reconfiguration and precharge transistors are shared among bits and thus are drawn outside the RCAM cell area. The main difference between RCAM and the conventional TCAM is that both data and mask SRAM bit of the RCAM cell are embedded with the 2T XOR logic and 1T evaluation logic. Consequently, there will be two individual CAM bits as opposed to the conventional TCAM cell which has a CAM data bit and a SRAM mask bit.

A. Operation

The RCAM has two separate comparand lines, i.e. $CMP0/CMPB0$ in $CAM0$ for the even-numbered words and $CMP1/CMPB1$ in $CAM1$ for the odd-numbered words. The source of the evaluation transistors of both CAM bits are tied to the drain of the reconfiguration transistor. The drain of the evaluation transistors are connected to the match lines $ML0$ and $ML1$, respectively. The controlling input R of the reconfiguration transistor basically reconfigures the RCAM cell into either two independent CAM cells ($R = 1$) or into one TCAM cell ($R = 0$). Depending on the flexibility desired, we can have one R for the entire RCAM module or one R for each row or column. For simplicity, we assume there is only one control R per memory unit. Assume that the data and mask bit values are already written into the SRAM bits using the word lines ($WL0$ and $WL1$) and the bit lines (BL and BLB). The operation of RCAM cell can be explained as follows.

1) **CAM Mode** ($R = 1$): Both match lines $ML0$ and $ML1$ in RCAM should be precharged before evaluation in

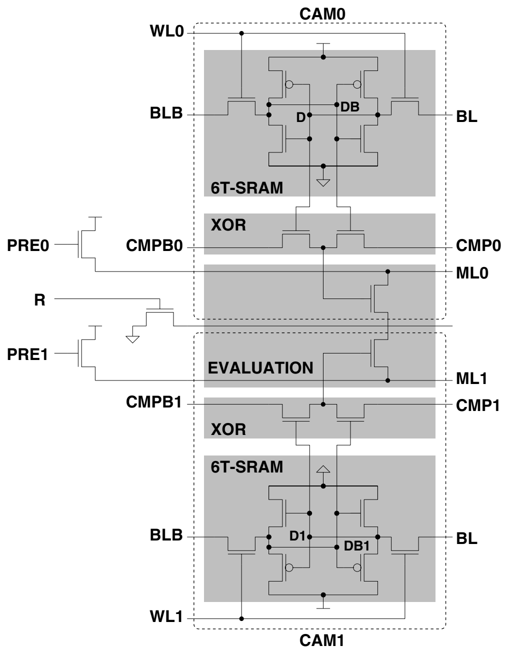


Figure 2. RCAM cell structure.

TABLE I
BEHAVIOR OF A BASIC RCAM CELL

R	CAM1 (Odd Cells)	CAM0 (Even Cells)	Outputs	Operation
1	CAM Bit	CAM Bit	ML1,ML0	Two CAM bits
0	Mask Bit	CAM Bit	ML0	One TCAM bit

CAM mode. The comparand lines $CMP0/CMPB0$ and $CMP1/CMPB1$ provide the key to be searched. $ML0$ and $ML1$ generate the comparison results.

2) **TCAM Mode** ($R=0$): Match line $ML0$ is pre-charged to V_{dd} and the match line $ML1$ is tied to ground during every search operation in TCAM mode. The comparand lines $CMP1$ and $CMPB1$ are connected to V_{dd} and ground, respectively. By this arrangement $CAM1$ behaves exactly like a mask bit in Figure 1(b). Therefore, $CAM0$ and $CAM1$ together form a TCAM cell.

The boolean equation for the main match line ($ML0$) in RCAM can be written as:

$$\begin{cases} ML0 = \overline{R}\cdot\overline{(D1\oplus CMP1)}+\overline{(D0\oplus CMP0)} \\ ML1 = \overline{R}\cdot\overline{(D0\oplus CMP0)}+\overline{(D1\oplus CMP1)} \end{cases} \tag{3}$$

where all signals are shown in Figure 2. When $R=1$, we have $ML0=\overline{(D0\oplus CMP0)}$ and $ML1=\overline{(D1\oplus CMP1)}$, that are identical to Equation 1. When $R=0$ and $CMP1=1$, we have $ML0=ML1=D1+\overline{(D0\oplus CMP0)}$ which is identical to Equation 2 when $D1$ is considered as the mask bit. For consistency, we consider $ML0$ as the final output of the cell when it operates in TCAM mode. For more clarity, the working of RCAM cell in both CAM and TCAM mode is summarized in Table I.

B. Applications

In what follows we briefly discuss three applications that benefit from a hybrid binary and ternary CAM cells in terms of cost, power and update time.

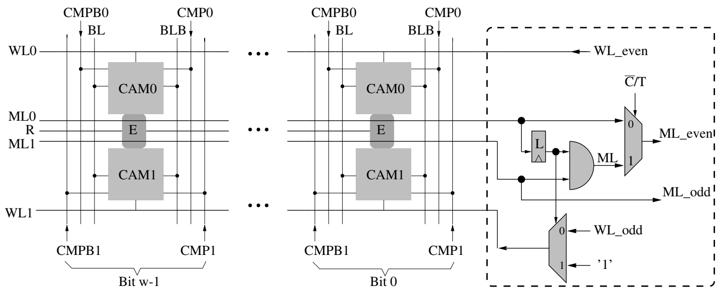


Figure 3. RCAM word architecture.

1. Classifier Engines: Packet classification in general refers to finding the best matching rule containing multiple fields among the rule set for a given search key. The standard five tuple fields mainly include the source address, destination address, protocol, source port and destination port. Rule fields are combination of prefixes, wild cards and exact values. Hence, a hybrid CAM/TCAMs that have the ability to store apply *exact*, *range-based* and *maximal* matchings will be quite efficient [15].

2. Low-Power Forwarding Engines: By comparing only the first 4-8 bits of incoming packet's destination address we can identify up to 80% mismatches in the forwarding table. Using this technique, average power saving of up to 76% is reported [16]. Application of such scheme requires mix of CAM (e.g. for the first 8 bits) and TCAM (for the rest) for which RCAM is a preferred choice.

3. Engines with no Priority Logic: The RCAM architecture has the ability to switch between TCAM and CAM functionality. This feature is exploited here by generating the maximal match through *exact matching* within CAM bits that hold the masking information. Our architecture, therefore, completely eliminates the use of prioritizer block, priority encoder and the need to sort entries. This would be possible by having cells that can be configured as a CAM or TCAM in each cycle. The detailed working of the RCAM architecture will be explained in Section IV.

IV. RCAM ARCHITECTURE

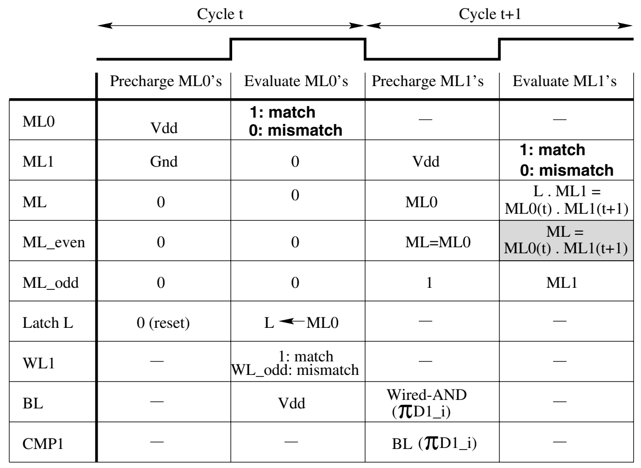
A. Architectural Details

The RCAM architecture uses the RCAM cell as its basic building block. Figure 3 shows how the word structure is formed from the basic RCAM cell. The match lines $ML0$ and $ML1$ along with the word lines $WL0$ and $WL1$ in the RCAM cell are connected horizontally with the match and word lines of neighboring RCAM. Together, they form two final word match lines (ML_{even} and ML_{odd}) and two final word lines (WL_{even} and WL_{odd}) for each word of RCAM. The multiplexer with select line $\overline{C}/T$ is used to choose between CAM and TCAM modes. A brief explanation of each mode follow.

1) **CAM Mode** ($\overline{C}/T=0$): The input of reconfigurable controlling transistor (R) is kept at 1 to essentially separate the two CAM cells (see Figure 2). Each RCAM cell will be equivalent to two independent CAM cells. The resulting outputs are ML_{even} and ML_{odd} which in turn are fed to the corresponding n-input encoder.

TABLE II
BEHAVIOR OF RCAM ARCHITECTURE

Mode	Cycle	$\overline{C}/T$	R	Operation
CAM	1	0	1	$2n$ word exact match search
TCAM (i)	1	1	0	a priority encoder is needed; n word partial match search
TCAM (ii)	1	1	0	n word partial match search using both even and odd CAM cells.
	2	1	1	Exact match search of longest mask using wired-AND technique.



Cycle t
Cycle t+1

	Precharge ML0's	Evaluate ML0's	Precharge ML1's	Evaluate ML1's
ML0	Vdd	**1: match 0: mismatch**	—	—
ML1	Gnd	0	Vdd	**1: match 0: mismatch**
ML	0	0	ML0	L . ML1 = ML0(t) . ML1(t+1)
ML_even	0	0	ML=ML0	ML = ML0(t) . ML1(t+1)
ML_odd	0	0	1	ML1
Latch L	0 (reset)	L ← ML0	—	—
WL1	—	1: match WL_odd: mismatch	—	—
BL	—	Vdd	Wired-AND (πD1_i)	—
CMP1	—	—	BL (πD1_i)	—

Figure 4. Control signals and operations of a bit-slice RCAM.

Together, two encoders (for odd and even cells) choose one out of $2n$ words in the RCAM block.

2) **TCAM Mode** ($\overline{C}/T = 1$): There are two possible schemes to make RCAM cell behave as a TCAM.

   (i) $R = 0$ and use a conventional n-input priority encoder. This scheme has been already mentioned in Table I and while possible is not of our interest as it offers no advantage over conventional TCAM.

   (ii) R will be 0 and 1 in the first and second cycles, respectively. The RCAM architecture in this mode takes two search cycles to compute the longest prefix match. In this case, $ML1$ and thus ML_{odd} are tied to ground. In this scheme, we use the wired-AND strategy and a n-input regular encoder. This scheme offers a significant reduction of update time (to be discussed in Section IV-C).

The behavior of the RCAM architecture is summarized in Table II. Note that in this paper we focus on TCAM(ii) that employs the wired-AND technique.

More details about the control signals (shown in Figure 3) in TCAM mode are shown in Figure 4. In this figure, Vdd and Gnd denote precharging toward Vdd and tying to ground, respectively. The shaded boolean function at the end of second cycle indicates how the final signal is obtained. The details of blocks and control signals can be found in [18].

B. Wired-AND Technique

The principle difference between RCAM and the conventional TCAM is the way by which the longest prefix is determined. Traditionally, TCAM-based packet forwarding engines determines the longest prefix entry by sorting the forwarding table and determining the longest prefix from multiple matches using a priority encoder (PE). The wired-AND technique used in our architecture completely eliminates the need for any sorting or priority encoding. The wired-AND technique in RCAM is the concept by which selected bits in the same column of different rows are read simultaneously on the same BL/BLB wire by activating their corresponding word lines.

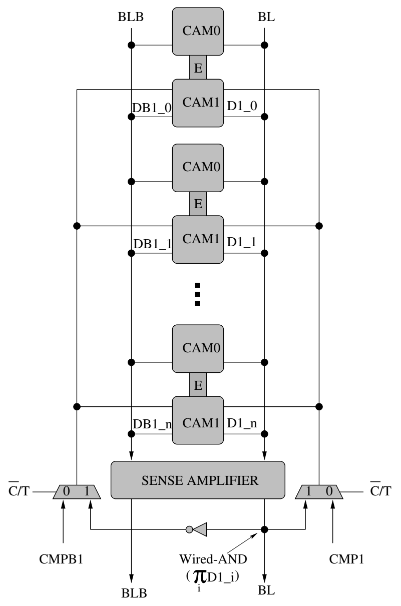


Figure 5. A bit-slice of RCAM architecture

In conventional TCAM if we read the data bits simultaneously using BL/BLB lines the output is going to be either a zero or one based on the strength of the equivalent pull up or pull down logic formed by the inputs being read. To get the wired-AND logic when multiple rows are read on the same line we size the related transistors in RCAM cells in such a way that it can withstand the strength of r parallel pull-up transistors. Therefore, the pull-down transistor present in the SRAM of the mask bit is sized r times the original value to counter this rare worst case. From the implementation point of view resizing cells for $2 \leq r \leq 16$ is quite straightforward. From practical point of view, in networking applications researchers found out that the maximum number of multiple matching prefix that can occur in a forwarding table is quite small. Based on empirical data, the authors in [19] reported this number to be 6. Similarly, the authors in [20] found that the highest number of multi match prefix to be eight across 112 ACLs in a router database with a total of 215K rules. The simulation results show that the distribution of multi match prefix per search was mainly concentrated between 3 or 4 matches. To be on the safe side we chose $r = 8$. In applications that may require larger r the pull-down transistors can be easily tuned. Figure 5 shows the bit-slice of RCAM architecture which incorporates this wired-AND technique.

C. Update Time

The behavior of the RCAM architecture is summarized in Table II. The longest prefix match operation in RCAM, when operating as a TCAM, takes two search cycles. The main advantage in this scheme compared to the conventional

TCAM architecture is the drastic reduction of total update time. The low update time in the RCAM forwarding table is a direct consequence of allowing the prefixes to be stored in any order and thus eliminating time consumed for sorting. To improve the table update time some of TCAM-based forwarding techniques partition the forwarding table according to their prefix lengths leaving empty spaces at the end of each partition for insertion of new entries. This technique does not require frequent updation but if empty spaces in any one of the prefix lengths gets filled the TCAM forwarding table has to be sorted. In general, any forwarding table architecture that uses priority encoder to find longest prefix match has to sort its table one way or another. Our RCAM does not face this problem since it allows prefixes to be stored in any order and does not need any priority logic.

In order to perform the longest prefix matching the prefix entries, which is a combination of the data and mask word, are stored in the even and odd words of the RCAM block, respectively. During the first search operation, RCAM is configured to work in TCAM mode and finds all possible matches for the key presented. In the second search, RCAM identifies the longest prefix match indirectly through the corresponding mask word. In other words, we first AND all mask words corresponding to the matches obtained in the previous search and then search the odd cells (home of masks) for an exact match (CAM mode).

To be clear about this, let us assume that in the first search operation there were three matches corresponding to the mask words $D1_1 = 001111$, $D1_2 = 000111$ and $D1_3 = 000001$. In this example, obviously, $D1_3$ corresponds to the largest prefix. The logical AND of these three mask words generates $\prod_{i=1}^{3} D1_i = 000001$. In the second round, we search the *mask data* words that exactly match 000001 which indirectly gives us the longest prefix match. In our implementation, the wired-AND operation will generate $\prod_{i=1}^{3} D1_i$. The bit-slice of RCAM architecture incorporating the wired-AND scheme is shown in Figure 5. The multiplexers in the bit-slice architecture are used to multiplex the comparand line inputs (CMP1 and CMPB1) according to the mode of operation. In CAM mode ($\overline{C}/T = 0$) a normal comparand (key) is presented and in TCAM mode ($\overline{C}/T = 1$) the wired-AND of the selected mask words is fed into the comparand lines.

$$CMP_1 = \prod_{i \in Matched} D1_i \tag{4}$$

Note that in TCAM mode, odd CAM cells (i.e. $D1_i$) hold the mask data. Also note that, since the wired-AND of $DB1$ lines makes $\prod_i DB1_i$ (which is different from $\overline{\prod_i D1_i}$) we used an inverter to directly make $CMPB1$.

V. Time Analysis

A. Search Time

Although the RCAM architecture takes two cycles to find the longest prefix, the overall search time is not doubled when compared to conventional TCAM architecture. The reason is the removal of the prioritizer circuit from critical path of the RCAM architecture. Analytically, the overall search time of RCAM architecture is given by:

$$t_{rcam\_search} \simeq 2t_{rcam\_block} + t_{encoder} \tag{5}$$

where, t_{rcam_block} is the delay of the RCAM block ($n \times w$ cells) and $t_{encoder}$ is the delay of a regular n-input encoder logic.

The analytical expression for the overall search time of a conventional TCAM architecture is given by:

$$t_{tcam\_search} \simeq t_{tcam\_block} + t_{priority\_encoder} \tag{6}$$

where, t_{tcam_block} is the delay of the TCAM block ($n \times w$ cells) and $t_{priority\_encoder}$ is the delay of a n-input priority encoder logic.

A straightforward VLSI implementation of these components indicate that [17]:

$$t_{rcam\_block} \simeq t_{tcam\_block} \simeq t_{priority\_encoder} >> t_{encoder} \tag{7}$$

Therefore, by combining Equations 5, 6 and 7, we can simplify the difference in search speed between the RCAM and TCAM architectures as follows:

$$\frac{t_{rcam\_search}}{t_{tcam\_search}} \simeq 1 + \frac{t_{encoder}}{2t_{priority\_encoder}} \tag{8}$$

According to Equation 8 the difference in search speed between the RCAM and TCAM architecture depends on the delay (size) of regular and priority encoders. For up to a few thousand words, $t_{priority\_encoder} \simeq 2t_{encoder}$ is a good approximation. Therefore, Equation 8 predicts that $\frac{t_{rcam\_search}}{t_{tcam\_search}} \simeq 1.25$ which means the search time of RCAM will be approximately 25% slower than conventional TCAM. In spite of longer search time in the next subsection we will show the overall performance of the system will significantly improve.

B. Update Time

The overall speedup of RCAM architecture compared to TCAM depends on both search and update time. In [18] we have analytically shown that the overall speedup of RCAM with respect to TCAM is given by:

$$Speedup = \frac{T_{TCAM}}{T_{RCAM}} = \frac{t_{tcam\_search}}{t_{rcam\_search}} * \left[1 + \frac{S_{avg}}{\alpha + 1}\right] \tag{9}$$

where, T_{TCAM} (T_{RCAM}) is the sum of total search time taken for N_S searches and N_U updates for TCAM (RCAM), S_{avg} is the average number of the shifts during the update time and $\alpha = \frac{N_S}{N_U}$. Note that no sorting (shifting) is needed in RCAM. For large tables (e.g. $10^3 \leq N_U \leq 10^4$ and $100 \leq \alpha \leq 1000$) the RCAM structure will outperform TCAM by 1 to 2 order of magnitude (speedup of 10 to 100).

VI. Experimental Results

The RCAM cell along with the conventional CAM and TCAM cells were implemented in 0.18μm digital CMOS technology using Cadence tools [21]. The RCAM cell was simulated using Spice [14] for all the possible cases in both the CAM and TCAM configurations and the results were extensively reported in [18].

We have summarized some power, performance and area metrics in Table III. According to Spice simulation [14], the

TABLE III

MAIN CHARACTERISTICS OF DIFFERENT 1-BIT CELLS

Metric	Conventional		RCAM	
	CAM	TCAM	CAM Mode	TCAM Mode
Delay (*ps*)	153.86	166.94	140.77	163.06
Area (μm^2)	24.97	47.17	26.36	52.72
Power (μW)	6.07	7.63	15.01	15.88

TABLE IV

COMPARING $4Kb$ PCAM UNIT WITH PREVIOUS WORK FOR KEY METRICS.

Comparing Metric	*RCAM Architecture*	
	CAM Mode ($C/T = 0$)	*TCAM Mode ($C/T = 1$)*
Technology	0.18μm, 6 metals	0.18μm, 6 metals
Chip Configuration	4Kb	2Kb
Match Line Arch.	NOR type	NOR type+Wired-AND
V_{dd}	1.8V	1.8V
Maximum Speed	284.3Mhz @1.8V	162.7Mhz @1.8V
Average Power	8.5mW @ 100Mhz	8.62mW @ 100Mhz
Power-Performance	20.75fJ/bit/search	42.08fJ/bit/search
Core Area	0.254mm^2	0.254mm^2

delay of a RCAM cell is slightly better than CAM and TCAM because the addition of the reconfigurable transistor forms new parallel paths that decrease parasitic resistances. However, this negligible gain in the search speed of one cell alone will not have a significant effect on the overall performance. This is because in RCAM/TCAM word architecture the overall performance is mainly dictated by both the time taken to charge/discharge the highly capacitive match line of the entire word and by the update time. The area per data bit, reported in Table III, indicates that RCAM cell consumes 5.6% and 11.8% more silicon compared to TCAM and CAM, respectively. The power (per data bit) consumed by RCAM working as CAM and TCAM is also higher than power of 1-bit CAM/TCAM due to existence of extra transistors. However, similar to our argument on performance, the increase in power consumption for an individual RCAM cell will be offset by the power saved from the elimination of the priority encoder and the update cycles in RCAM architecture.

Table IV shows the delay, area and power numbers of a $4Kb$ (i.e. 128×32) RCAM architecture in both modes compared to two implementations reported in the literature. Due to differences in technology, size, cell-library and design objectives (e.g. area, power) a direct comparison of the these implementations is not possible. Our main goals of showing Table IV is to illustrate that our implementation have comparable size, performance and power while it provides the reconfiguration capability. In particular, by eliminating the priority encoder, the RCAM unit runs in 162.7MHz, i.e. expectedly almost half of its operational frequency in CAM mode. This is consistent with the analytical discussion in Section V where we argued that after removing the priority encoder, the TCAM mode operation is done in two cycles as opposed to one cycle in CAM mode.

In terms of area, in spite of area increase in one individual cell the overall area actually improves as the priority encoder is replaced with a regular binary encoder. The power-performance metric was also evaluated for both the modes of RCAM and is found to comparable with other existing architectures [22] [23] [24]. Unfortunately, we cannot directly compare RCAM in TCAM mode with the TCAM units reported in these references because of the difference in implementation technology (0.18 μm versus 0.13 and 0.35 μm) and the size ($2Kb$ versus 4 to 36Mb). In general, RCAM in TCAM mode with $42.08 fJ/bit/search$ shows 6.5% higher power-performance value but achieves maximum speed of $162.7Mhz$ that is 15.1% higher performance than conventional TCAMs.

REFERENCES

[1] S. Nilsson and G. Karlsson, "IP-Address Lookup Using LC-Tries," *IEEE J. Selected Areas in Communications*, vol. 17, no. 6, pp. 1083-1092, 1999.

[2] Paolo Ferragina and Roberto Grossi, "The String B-Tree: A New Data Structure for String Search in External Memory and its Applications," *Journal of the ACM*, vol. 46, no. 2, 1999.

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

[4] 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.

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

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

[7] F. Zane, G. Narlikar, A. Basu, "CoolCAMs: Power-Efficient TCAMs for Forwarding Engines," *IEEE INFOCOM*, March 2003.

[8] E. Spitznagel, D. Taylor, and J. Turner, "Packet Classification Using Extended TCAMs," *IEEE Int. Conf. on Network Protocols (ICNP'03)*, November 2003.

[9] R. Panigrahi and Samar Sharma, "Sorting and Searching Using Ternary CAMs," *IEEE Micro*, Feb. 2003.

[10] V. Ravikumar and R. Mahapatra, "TCAM Architecture for IP Lookup Using Prefix Properties," *IEEE Micro*, April 2004.

[11] Craig Labovitz, Robert Malan, Farnam Jahanian, "Internet routing Instability," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, Oct. 1998.

[12] M. Kobayashi, T. Murase, and A. Kuriyama, "A Longest Prefix Match Search Engine for Multigigabit IP Processing," *Proc. Int'l Conf. on Communications (ICC)*, IEEE Press, Piscataway, N.J., pp. 1360, 2000.

[13] D. Shaw and P. Gupta, "Fast Updating Algorithms for TCAM," IEEE Micro, Jan./Feb. 2001.

[14] Texas Instruments Inc., "TI Spice3 User's and Reference Manual - Version 1.6," 1994.

[15] D. E. Taylor and E. W. Spitznagel, "On Using Content Addressable Memory for Packet Classification," *Technical Report WUCSE-2005-9*, March 2005.

[16] Deepak S Vijayasarathi, Mehrdad Nourani, Mohammad J. Akhbarizadeh, Poras T. Balsara, "Ripple-Precharge TCAM A Low-Power Solution for Network Search Engines," *Proc. of ICCD Conference*, pp. 243-248, 2005.

[17] M. Akhbarizadeh and M. Nourani, "Hardware-Based IP Routing Using Partitioned Lookup Table," in *IEEE Transactions on Networking*, vol. 13, no. 4, pp. 769-781, Aug. 2005.

[18] Deepak S Vijayasarathi, Mehrdad Nourani, "Design and Implementation of Reconfigurable CAM Architecture," *Technical Report*, UTD-EE12-12-2005, 2005.

[19] M. Kounavis, A. Kumar, HM Vin, R. Yavatkar, and A. Campbell, "Directions in Packet Classification for Network Processors," Workshop on Network Processors & Applications – NP2, Feb. 2003.

[20] Karthik Lakshminarayanan, Anand Rangarajan, Srinivasan Venkatachary, "Algorithms for Advanced Packet Classification with Ternary CAM," *SIGCOMM'05*, Aug., 2005.

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

[22] K. Pagiamtzis and A. Sheikholeslami, "Pipelined Match-Lines and Hierarchical Search-Lines for Low-Power Content-Addressable Memories," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 383-386 , Sept. 2003.

[23] Alan Roth, Dick Foss, Robert McKenzie, and Douglas Perry. "Advanced Ternary CAM Circuits on 0.13 um Logic Process Technology," in *Proceedings of Custom Integrated Circuits Conference*, pp. 465-468, Oct. 2004.

[24] C. Lin, J. Chang and B. Liu, "A Low-Power Precomputation-Based Fully Parallel Content-Addressable Memory," *IEEE Journal of Solid-state Circuits,* vol. 38, no. 4, April 2003.