August 2014 IJMIE Volume 4, Issue 8 ISSN: 2249-0558

REVIEW OF CTAM, CTAM-BASED AM, BSP, PCTAM AND PCTAM-BASED AM

Firoz Shaikh,
Graduate Student, Computer Science and Engineering Department,
Jadavpur University, Kolkata-700032
Email: firoz983656@gmail.com

Swapan K. Ray, SMIEE,
Professor, Computer Science and Engineering Department,
Jadavpur University, Kolkata-700032
Email: skray@ieee.org

INTRODUCTION:

Packet Classification requires a very high-speed search of a large rule database. Since the various software-based approaches suggested for Packet Classification are often either incapable of meeting this high-speed search or require an excessive amount of memory, hardware –based approaches have been increasingly sought in the recent years. The traditionally popular hardware device for high-speed search, namely, Content Addressable Memory (CAM) (see Section 2.2) and, specifically, its augmented version Ternary CAM (TCAM) is being employed for high-speed packet classification very often. As a matter of fact, TCAM is presently viewed as the de-facto standard hardware device for high-speed packet classification [1]. It should be observed that there are really three overwhelming popularities of TCAM for use in high-speed packet classification. First is its high-speed searching capability which is due to its parallel comparison architecture. Second is the relative simplicity of its operation as well as relative simplicity of designing systems based on it. Third is the fact that directs implementation of prefix-based searches is possible on a TCAM. However, this third advantage is balance out by the great disadvantage carrying out range-based search because each range on TCAM needs to be converted into not one but multiple prefixes. Obviously, this range to prefix conversion causes the wastage of much of the CAM's capacity (see section 2.2). However, TCAM has important disadvantages like high cost, large power consumption and limited capacity [1], [2]. The reason that lies behind all the three disadvantages' factors of a TCAM is that while the static RAM needs only 4-6 transistors per bit of storage, the TCAM requires 11-15 transistors per bit. As a result, use of TCAMs in building very large packet classifiers having hundreds of thousands of rules is not generally envisaged. As an alternative to the TCAM which has a parallel architecture, a Content- To- Address Memory (CTAM) was recently designed [3]. Since, barring a few simple digital circuits, CTAM is built almost entirely with RAM array; it can overcome all the aforesaid disadvantages of TCAMs.

Though the idea of an Associative Memory (AM) is fairly old, a very clear concept about what an AM really is and how it works is not yet available in the literature. However, notionally it is accepted that an AM is a new kind of memory, different from the traditional Random-Access Memory (RAM) that can accelerate any application which requires fast searches of a database. This acceleration can be achieved by utilizing any inherent association that might exist between the different entities in the concerned database. The most important feature that distinguishes an AM from the RAM is that it should be possible in an AM to access any record or part (say, a field) of it by specifying the content of any part of the record as a query, rather than by specifying any memory address as is done in a

RAM. That is, in an AM, the memory is accessed by a (search) content as a query, rather than by an address.

Two different approaches towards building a large and fast AM are presently known. The older, well known and widely used approach is the CAM [4], [5]. The second and the newer approach is the AM based on the concept of Content- To- Address Memory (CTAM) [3]. Between the two, CAM chips have been commercially available for around 25-30 years and are widely used in many applications including packet classification. In contrast, the CTAM –based AM being a new concept, is still in the design stage. Regarding the principle of operation while the CAM is based on the concept of parallel comparison using considerably augmented SRAM cells, the CTAM-based AM is based on the concept of pipelined binary search [3]. However, since the CTAM-based AM employs simple SRAM cells, it is expected that it will have less cost and less power consumption than the CAM. Use of this CTAM-based AM has been suggested, in the recent years, toward solution of several high-speed search problems on large databases [6], [7], [8], [9]. Concept and scheme of implementation of a CTAM will now be described in subsection 4.2. Most of the materials presented in this section and in the next section are adopted from [A PARALLEL –PIPELINED ARCHITECTURE FOR HIGH –SPEED IP-LOOKUP USING ASSOCIATIVE SEARCH MEMORY]. In subsequent subsections 4.3 through 4.6, will be described the concept of CTAM-based AM, Binary Search Processor (BSP), Pipelined CTAM (PCTAM) and Pipelined AM (PAM), respectively.

CTAM

During a memory read operation, the traditional memory RAM performs the forward function that is it outputs the content of the inputted address. Thus, a RAM can also be called as Address-To-Content Memory or ATCM. In a RAM, the user supplies the address, and gets back the data. Thus, in the ATCM, every time the address of every piece of stored data element must be kept track of to retrieve any data. In order to support the bidirectional association, the two-memory access or interface registers MAR (Memory Address Register) and MDR (Memory Data Register) also function as bidirectional registers, switching direction with the operational mode of the memory. Thus, the MAR inputs the address in the ATCM while MDR outputs the data (content) in the ATCM. On the other hand, the CTAM performs the inverse function i.e., given a desired content as a query, during a memory read operation, the CTAM outputs the address of the location where this query content is present. If the input data is not unique, i.e., if there are multiple locations which store the same input data, then the CTAM will output all the addresses where content is stored. So, the CTAM will suffer from duplicate content problem since the same content may be stored at more than one address. The MDR inputs the content in the CTAM while the MAR outputs the address in the CTAM. During writing, both MAR and MDR performs as in the input mode. So a CTAM basically performs an inverse function of what a RAM. Fig 4.1 illustrates the difference between an ATCM and a CTAM from the functional perspective in the context of querying mode on a RAM.

Now we need to present the novel concept of a Content-To-Address Memory (CTAM) in the general context of the functioning of a memory unit. A $N(=2^n)\times W$ memory unit may be viewed as a bidirectional one-to-one association between a set of addresses $\{A_i\}$; $A_i€ \{0,1, 2,..,N-1\}$ and the set of their respective stored contents $\{X_i\}$, $X_i €\{0,1,2,\ldots\ldots,2^W-1\}$, where the A_is and the X_is are all represented in binary. This bidirectional association can be viewed as a pair of complementary mapping functions shown in equation 1. **Fig. 4.2.** Illustrates

Concept of memory read operation as a bi-directional association between an address or location and its content or data. (a) ATCM Read mode (b) CTAM Read mode

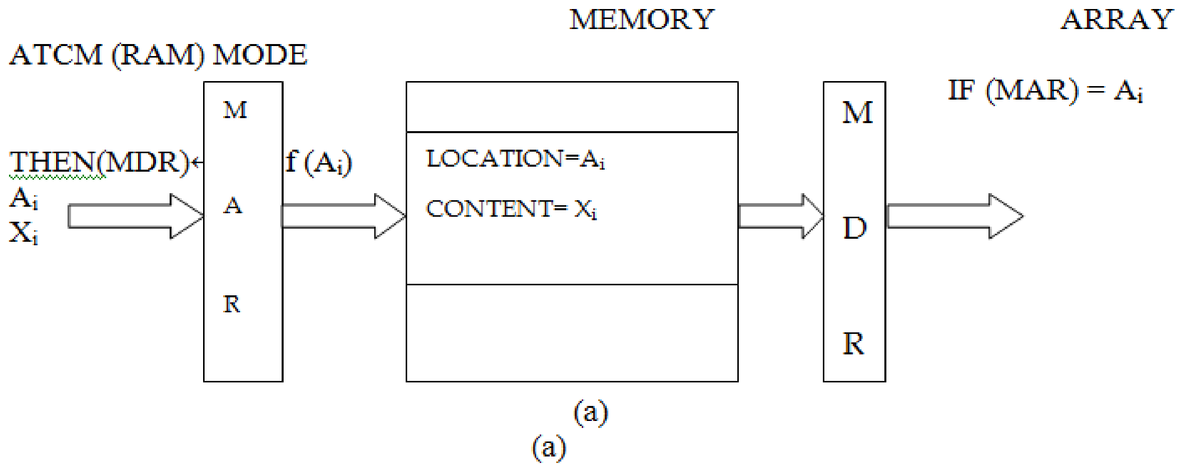
$X_i = f(A_i)$……...........(1a) $A_i = f^{-1}(X_i)$…………......(1b)

Memory Address	Data Element
7	ROOM
6	CHAIR
5	TABLE
4	BAG
3	BOOK
2	PEN
1	WINDOW
0	DOOR

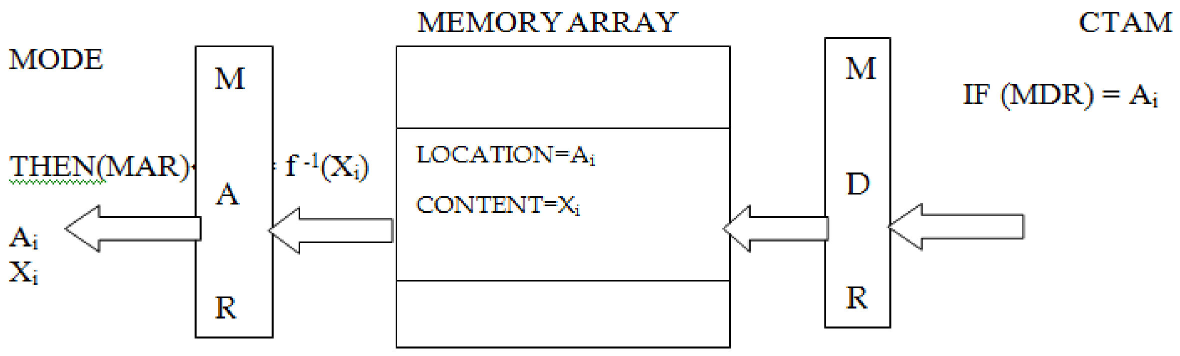
ATCM Query:
Q. What is the data element of address 4?
A. BAG

CTAM Query:
Q. What is the address of the location whose data Element is BAG?
A. 4

Fig. 4.1. Functional difference between ATCM and CTAM in the context of querying mode in CTAM database.



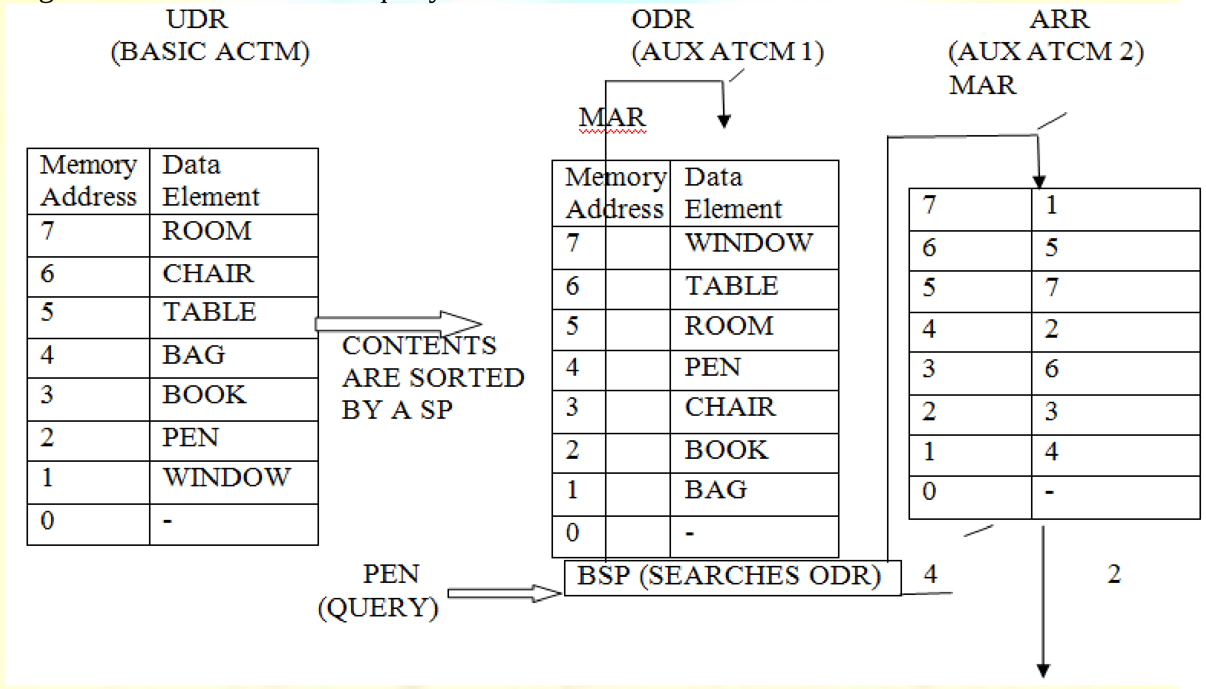
(a)



(b)

Fig. 4.2. Concept of memory read operation as a bi-directional association between an address or location and its content or data. (a) ATCM Read mode (b) CTAM Read mode

Though no physical devices for building a CTAM exist, The CTAM can be implemented by augmenting a basic ATCM (RAM) which stores the input unordered DFs, with two more auxiliary ATCMs, a Sort Processor (SP) and a Binary Search Processor (BSP), as is illustrated in Fig.4.3. The basic ATCM to be called the “Unordered Data RAM” (UDR) has $2^n – 1$ (n=3) DFs stored, in an unordered manner, in location 1 through $2^n – 1$. No data is stored in location 0 as it is not searched by a BSP. The content of the UDR is first sorted by the external Sort Processor in an ascending order to build the Ordered Data RAM (ODR) which can be searched by the BSP against the given query. In Fig. 4.3, since the search content or query given to the BSP is PEN, the BSP returns the address 4 at its output which is not actual desired address of PEN in the UDR .Since sorting had altered the original order of data that had existed in the basic RAM or the UDR, an address reordering is need following the BSP search using an Address Reordering RAM (ARR) (i.e., AUX. ATCM2). The ARR maps all the BSP generated ODR output addresses back to their original UDR addresses. Thus, in the example in Fig 4.3, at its address 4, the ARR store 2 which is the original UDR address of the query PEN.



Memory Address	Data Element
7	ROOM
6	CHAIR
5	TABLE
4	BAG
3	BOOK
2	PEN
1	WINDOW
0	-

Memory Address	Data Element
7	WINDOW
6	TABLE
5	ROOM
4	PEN
3	CHAIR
2	BOOK
1	BAG
0	-

7	1
6	5
5	7
4	2
3	6
2	3
1	4
0	-

Fig. 4.3 Scheme of implementation of 7-word CTAM by augmenting the basic ATCM with two AUXILIARY ATCMs and a BSP

Hence the BSP finally reads the ARR at address 4 to yield the actual CTAM output address of 2 as its response to the query “PEN” in the basic ATCM.

CTAM-BASED AM

In order to explain the concept and implementation of a CTAM-based AM, we consider ATCM stores multicolumn or multi-field records instead of single-column or single field records. Alternatively, a multi-field record may be stored by being distributed within multiple single column RAMs of the same word capacity with each data value in the record being stored in the same memory address. In both ATCM and CTAM, the concept of content –to-address association exists. The difference in the nature of this association is that in case of an ATCM, the association is forward (Address →Content) whereas in case of the CTAM the association is backward (Address ←Content). But the Associative Memory (AM) utilizes two different kinds of association in a multi-field database. The first association "content – to- address association" is between each field value within a record and the common address of all fields in the record, i.e., the address of the record. The second associations, all the different field values in the entire record are "mutually associated" with one another via their common RAM address in the database. It should be noted that in general, no association exists between the different field values within a column. An AM seeks to utilize both the association for performing high-speed associative search and retrieval of any record or part of it, based on any given field value as a query input.

In contrast to an ATCM or CTAM, the AM deal with a multiple association involving a collection of multi-field records, stored in a RAM array as in a relational database. Now we consider a 7-record database, with each record having three fields, as shown in Fig. 4.4. An example associative query on this database is like this: "What is the Field 3 value in the record whose Field 1 has the value F1?". We assume that all the FVs in any particular column are unique. Now, if an ATCM is used to store the database and search is done by writing a suitable program, then first read the content of Field 1 column memory at all successive address from 1 to 6, read out the respective contents and compare each of them to F1 to identify the record 6 with which F1 is associated. Then the query answer can be obtained just by reading out the field 3 value of the record 6. Obviously, the search is linear and the search time will increase linearly with the number of records in the ATCM.

Field 1	Field 2	Field 3	Memory Address
G1	G2	G3	7
F1	F2	F3	6
E1	E2	E3	5
D1	D2	D3	4
C1	C2	C3	3
B1	B2	B3	2
A1	A2	A3	1
x	x	x	0

Fig. 4.4. 7 3-field records stored in a simple wide-memory array

However, instead of doing that, we would like to utilize the associative search employing the CTAM-based AM to speed up the search process. So, we store the 3 FV columns in 3 separate ATCMs, namely, ATCM1, ATCM2 and ATCM3 as shown in Fig. 4.5. Additionally, we augment the Field 1 memory column ATCM1 into a CTAM in the manner described earlier (see Fig.4.1). Then utilizing the "content – to- address association", a search into the CTAM against the query FV "F1", yields the memory address of the concerned FV as 6. Now utilizing the second association “mutual association” between all the three FVs in the record 6, via their common record number 6, the AM just reads the ATCM3 at the same memory address 6 to obtained the desired answer “F3” of the CTAM based AM.

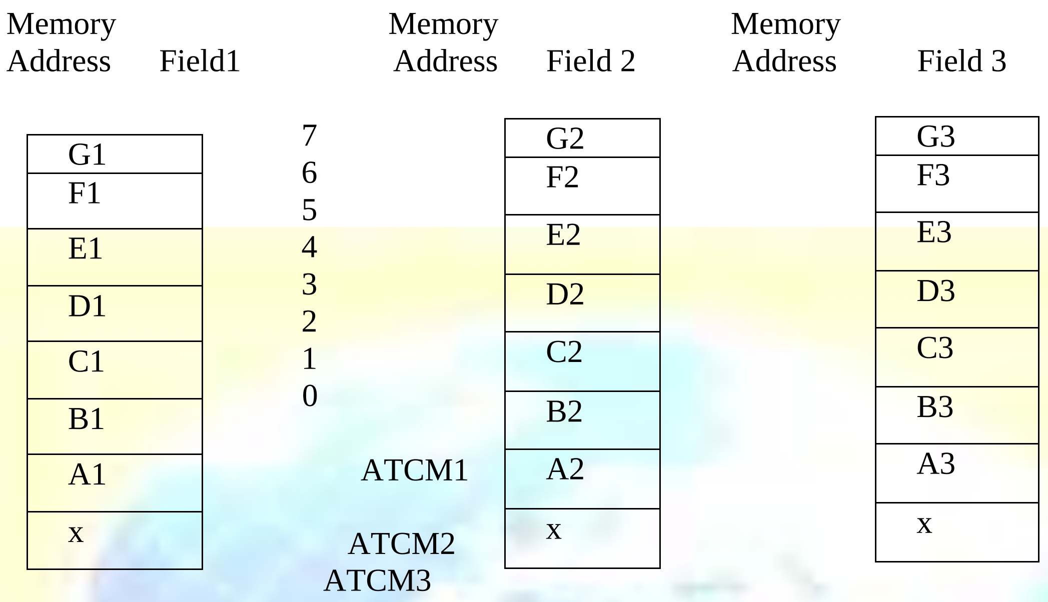


Fig. 4.5. 7 3-field records stored in 3 separate simple narrow-memory array

4.4 MODULAR BINARY SEARCH PROCESSOR ARCHITECTURE

The Binary Search Processor (BSP), described in more details in [3], is used as the main component to build CTAM-based AM which was described in the previous section. An inspection of the scheme of implementation of a CTAM as shown in Fig. 4.1 suggests that a pipelined structure of the CTAM can be obtained by just pipelining the BSP. Now, in order to discuss the design of the BSP and the method of pipelining the BSP, BSP runs the well-known Binary Search Algorithm (BSA) [10], [11] to perform a search operation on its ordered data set. For N= (2^n -1) number of data elements in the ordered data set, the BSP can search a data residing in its Key Storage RAM (KSR) in n(=$\log_2$N) or a smaller number of trials or clock cycles. This ordered data set is sometimes referred as KSR. BSA covers the N elements search space in ($\log_2$N) successive trials, with each trial actually halving the search space. If the number of bits in the address is n, i.e., the address is of the form $A_{n-1}A_{n-2}$……. A_0, then the BSA actually finds out the first address bit in the first trial, the second address bit in the second trial, and so on up to the n-th trial for finding out the address bit A_0. So it can be generalized as follows; in the k-th trial the BSA finds out the address bit A_{n-k} of an address A ($A_{n-1}A_{n-2}$…….A_0) where the Search Key Value (SKV) resides for k=1,2 3,…..n. Hence the address of an SKV can be found out after the nth trial.

In each of these trials, the BSA actually performs a simple set of identical operations. In each trial, BSA performs four identical operations; (1) computes the TA (2) read the value at that trial address, i.e, (TA) (3) compare the content of that TA with the SKV and (4) finally outputs the value of this comparison, such that next TA can be formed. The output of the comparison is indicated by producing three-bit Comparison Result (CR=G-E-L). For the k-th trial the value of CR would be CR_{n-k} =G_{n-k},E_{n-k},L_{n-k}. If after the kth trial, the SKV is greater, equal or less than the content of the TA, then it would be indicated by setting the corresponding value of the CR to 1, i.e , for SKV>(TA), G_{n-k} would be set to 1, for SKV=(TA), E_{n-k} would be set to 1, for SKV<(TA), L_{n-k} would be set to 1. So depending on the CR's value, the next trial address would be formed as explained below.

If r is taken as another index equal to (n-k) , i.e, r=n-k, then the kth trial can be represented as (n-k)th trial for k=1,2,3,…….n. Now the first trial can be represented as (n-1) th trial, the 2nd trial as (n-2) th trial, so now trials are (n-1)th , (n-2)th ,……0th trial number or the trials are for r=(n-1), (n-2),…0. Now the next TA formation procedure can be represented by a simple mathematical operation as given in equation 1.

$$TA_r = TA_{r+1} + G_{r+1} * 2^r \;\; + E_{r+1} * 0 - L_{r+1} * 2^r \quad \ldots\ldots\ldots\ldots(1)$$

So, from equation (1), it is clear that the next TA can be calculated depending on the value of CR (=G-E-L) and the current TA. So in each trial, the BSA performs the same set of operations and can find out the data or the SKV, in an ordered set of N (2^n-1) number of elements, in n (= $\log_2 N$) or less number of trials. In an intermediate stage r, a match for the SKV may be found and that would be indicated by the E_r bit value of the CR being 1. Here the number of trials can be made constant, i.e, equal to n, by setting the TA for all the successive trials as current TA as can be seen from the equation 1 that for all the successive trials after a match is found, the TA would be same as current TA and all the successive trials would find a match in that same TA with the SKI. Hence after the nth trial, the address corresponding to the SKI would be found from the last stage.

At each of these n numbers of stages, all the operations can be performed by a simple Binary Search Processing Element (BSPE). For n number of stages, there should be n BSPEs. At the rth stage, BSPEr works on the data and finds out the address bit Ar of an address $A(A_{n-1}A_{n-2}\ldots\ldots.A_0)$. It can be generalized that $BSPE_{n-1}$, $BSPE_{n-2}$,…….$BSPE_0$ finds out the address bit A_{n-1} , A_{n-2} , ……., A_0 respectively, corresponding to the address $A(A_{n-1}A_{n-2}\ldots\ldots A_0)$. Logic diagram of the r-th BSPE in the modular architecture of a BSP is shown in Fig. 4.6.

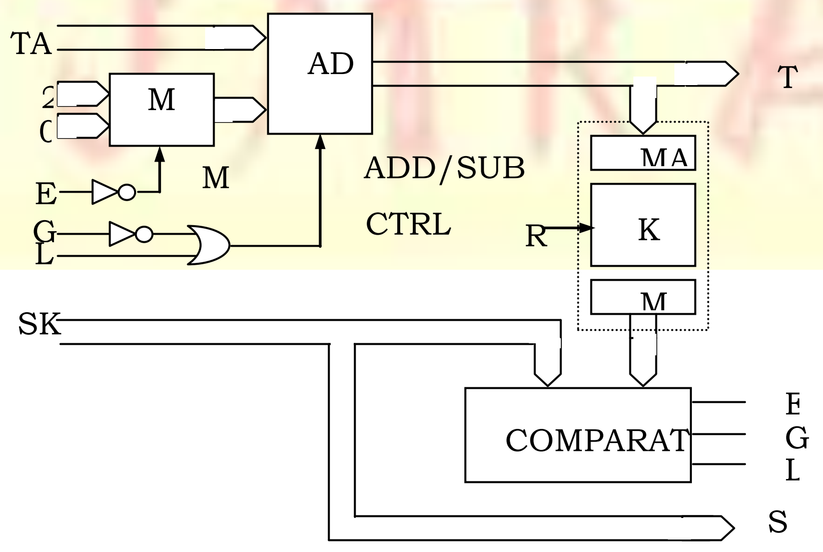


Fig. 4.6. Logic diagram of a BSPE in the modular BSP

Here some points should be noted regarding the BSPEs in the modular BSP.

(1) The MSB BSPE i.e, $BSPE_{n-1}$ should receive, externally, the input parameter values TA_n =0000…0, G_n=1, E_n=0, L_n=0, besides the SKV.

(2) The Least significant BSPE, i.e, $BSPE_0$ should provide as external outputs, the generated values of the parameters TA_0, G_0, E_0, L_0 and the SKV.

(3) The success of a search will be indicated at the end of the nth trial by providing E_0=1 and TA_0 will provide the address of the matching word. In the case of unsuccessful search, it will be indicated by E_0=0 and TA_0 provides the address of the key value which is either immediately small (indicated by G_0=1) or immediately greater than the SKV (indicated by L_0=1).

(4) At each stage, a stage specific constant value is required. This stage specific constant value depends on the stage. For r-th stage in the BSPL this would be 2^r .

4.5 PCTAM

The pipelined version of the BSP is referred as Pipeline BSP or PBSP. This PBSP will function as a Pipelined—CTAM (PCTAM) [3]. Though BSP, as described in the previous section, has a modular architecture, it executes the BSA only sequentially, i.e , all the trials are executed sequentially. The design of the basic PCTAM is based on a Binary Search Processor (BSP) which performs the search operation on its adjunct Key Storage RAM (KSR) by applying popular binary search algorithm (BSA).A pipelined version of the BSP called the Pipelined Binary Search Processor (PBSP) acts as the heart of the PCTAM. An n-stage PBSP, capable of performing pipelined binary search on a 2^n-1 word RAM in which data are ordered, contains n identical and extremely simple BSP Elements (BSPE), each BSPE dedicated to carry out a particular trial in the execution of the BSA.

So if these n successive trials can be pipelined, then the search operation can be speed up to n times. The BSA actually searches a data in n (=$\log_2 N$) trials or clock cycles—yielding a maximum throughput of 1/n searches per clock cycle. By pipelining the operation of the BSP, the search operation speed can be increased n times and can achieve a throughput of one search per clock cycle—but with the latency of n clock cycles for the each individual search.

By pipelining the operation of a modular BSP having n identical simple processing stages (processing elements PEs) and letting each PE search its dedicated local copy of the ordered RAM, a synchronous pipelined BSP (PBSP) or pipelined CTAM (PCTAM) has been designed. Figure 4.7 (a) shows the block diagram of an individual stage in the PCTAM and Figure 4.7 (b) shows the block diagram representation of the n-stage PCTAM itself. The r-th stage in the PCTAM**,** r=n-1, n-2,…..,0 receives from its predecessor the trial address (TA_r+1) and the result of comparison of SKI with (TA) in the form of "Greater than "(G),"Equal to"(E) and "Less than" (L) bits and generates the corresponding output values for its successor stage. A few basic points relevant to the design and performance of the PCTAM are as follows:

1. Each local copy of the ordered-RAM is of the size (2^n×w), where N<=2^n-1. In case N<2^n-1, then the highest possible value of the SKI, i.e. 2^W-1, is stored in all the higher addresses beyond N. Location 0 is not used to store a data element since it is not accessed in the binary search algorithm.

2. Although, in the PCTAM , the ordered-data RAM needs to be replicated n times, the fast dwindling cost of RAM together with the extreme simplicity of the n PEs make the

PCTAM an attractive hardware element for building large and high- throughput search processors.

3. Addition of two more stages (these are only marginally different from the other n-stages) to the PCTAM can take care of the duplicate content problem [SK RAY].

4. The PCTAM can achieve a throughout rate higher than **$1/2t_a$**, where **t_a** is the RAM access time in seconds.

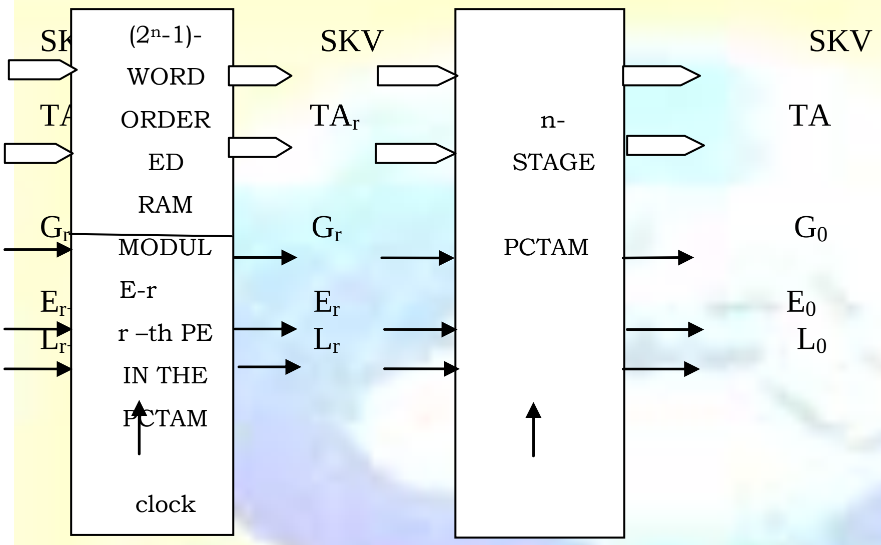


Fig. 4.7: Block diagram representation of (a) the r-th stage of a PCTAM and (b) a n-stage PCTAM

4.6 PAM

This section will be devoted to explain how PCAM has been realized from the PCTAM with a little augmentation. A CAM or AM is a memory array where a word is accessed by specifying a part of its content, rather than its address. Functionally, each W-bit word in an N×W CAM array has two broad or major fields, namely a W_s (<W)-bit "search key" field (SKF) and a W_d (<=W)-bit data field (DF). We assume that, for each of the N words in the CAM, the SKF is stored in an N×W SKF Storage RAM (SKFSR) and the DF is stored in a separate N×W_d DF Storage RAM (DFSR). The following two steps can now build a PCAM

1. Order the contents of the SKFSR, ascendingly or descendingly and build a n-stage PCTAM for searching the ordered content of the SKFSR.

2. Deploy a N-word "reordering" RAM called "Address Reordering RAM" (ARR) and write in its location j, the word i, for all j and all i, if the word originally occupying location i in the SKFSR has moved to location j after the contents of the SKFSR were ordered..

3. Augment the n-stage PCTAM with two more stages namely, the ARR followed by the DFSR.

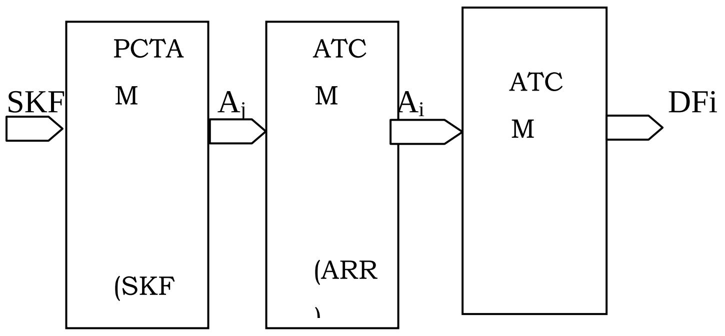


Fig. 4.8. Three block representation of a PCAM

Fig. 4.8 shows block diagram representation of the n+2 stage PCAM, where only the first stage is a PCTAM but the next two stages are simple ATCMs or RAMs. Using static RAM (SRAM), the PCAM is capable of searching an arbitrarily large SKFSR for any large stream of SKFs (i.e, SKVs) and reading out its associated information from the DFSR at a constant high throughput rate on the order of 50-100 million associative searches per second.

REFERENCES

1. J. V. Lunteren and T. Engbersen, “Fast and scalable packet classification,” IEEE J. on Selected Areas in Communiation, Vol.-21, No.4, MAY 2003, PP. 560- 571.

2. P. Gupta and N. Mckeown, “Algorithm for packet classification,“ IEEE Network, March/April 2001, pp. 24-32.

3. S. K. Ray, “ Large - Capacity High – Throughput Low – Cost Pipelined CAM Using Pipelined CTAM, “ IEEE Transactions on Computers, Vol-55, No. 5, May 2006, pp. 575-587.

4. L. Chisvin and R.J. Duckworth, “Content-Addressable and Associative Memory Alternatives to the Ubiquitous RAM,” Computer, pp. 51-64, July 1989.

5. T. Cohonen, Content-Addressable Memories. Berlin: Springer-Verlag, 1980.

6. S. K. Ray, S. Dutta and A. K. Saha, “A low-cost pipelined multi-lingual E-dictionary using a pipelined CTAM, Proc. Int. Conf. on Computing: Theory and Applications, held at ISI, Kolkata during March 5-7, pp. 158-163, Published by IEEE Computer Science Press.

7. S.K. RAY and S. Ghosh, “Low-cost Dictionary Machine using RAM-based CAM” in Proc. Intl. Conf. on Advanced Computing(ICAC 09), held in Tiruchirapalli, India, pp. 559-566, Aug. 6-8, 2009.

8. S. K. Ray and F.Shaikh, “PCAM-based Wire-speed Range Matching in Multidimensional Packet Classification” Proc. COMSNETS 2009 International Conference held in Bangalore, during Jan 5-10, 2009 pp. 518-519 (also available in IEEE Explore)

9. S.K. Ray, D. Roy, and A.R. Shaikh, “DNA Sequence Alignment Engine: An AM-Based Design” presented in the IEEE INDICON 2011 Conf. held in Hyderabad, India during Dec. 16-18, 2011 (also available in IEEE Explore)\

10. D.E. Knuth, The Art of Computer Programming, Vol.3: Sorting and Searching, Reading, Mass: Addision-Wesley, 1973

11. T.H.Cormen, C.E, Leisersonn and R. L. Revest, Introduction to Algorithms, New Delhi: Prentice-Hall of India, 1998