International Journal of Engineering Research and General Science Volume 2, Issue 3, April-May 2014
ISSN 2091-2730

Low Power Design of Pre Computation-Based Content-Addressable Memory

SK.Khamuruddeen[1], S.V.Devika[1], V Rajath[2], Vidhan Vikram Varma[2]

[1]Associate professor, Department of ECE, HITAM,Hyderabad, India

[2] Research Scholar (B.Tech), Department of ECE, HITAM,Hyderabad, India

ABSTRACT - Content-addressable memory (CAM) is a special type of computer Memory used in certain very high speed searching applications. It is also known as associative memory, associative storage, or associative array. Content-addressable memory (CAM) is frequently used in applications, such as lookup tables, databases, associative computing, and networking, that require high-speed searches due to its ability to improve application performance by using parallel comparison to reduce search time. Although the use of parallel comparison results in reduced search time, it also significantly increases power consumption. In this paper, we propose a Block-XOR approach to improve the efficiency of low power pre computation- based CAM (PB-CAM). Compared with the ones-count PB-CAM system, the experimental results show that our proposed approach can achieve on average 30% in power reduction and 32% in power performance reduction. The major contribution of this paper is that it presents practical proofs to verify that our proposed Block-XOR PB-CAM system can achieve greater power reduction without the need for a special CAM cell design. This implies that our approach is more flexible and adaptive for general designs.

Keyword's— Content-addressable memory, Block-XOR, pre computation- based CAM

I.INTRODUCTION

1.1 Existing System:

A CAM is a functional memory with a large amount of stored data that compares the input search data with the stored data. Once matching data are found, their addresses are returned as output. The vast number of comparison operations required by CAMs consumes a large amount of power.

1.2 Proposed System:

This proposed system approach can reduce comparison operations by a minimum of 909 and a maximum of 2339. We propose a new parameter extractor called Block-XOR, which achieve the requirement.

II. CAM OVERVIEW

Content addressable memory (CAM) compares input search data against a table of stored data, and returns the address of the matching data [1]–[5]. CAMs have a single clock cycle throughput making them faster than other hardware- and software-based search systems. CAMs can be used in a wide variety of applications requiring high search speeds. A CAM is a good choice for implementing this lookup operation due to its fast search capability.

However, the speed of a CAM comes at the cost of increased silicon area and power consumption, two design parameters that designers strive to reduce. As CAM applications grow, demanding larger CAM sizes, the power problem is further exacerbated. Reducing power consumption, without sacrificing speed or area, is the main thread of recent research in large-capacity. CAMs. Development in the cam area is surveyed at two levels: circuits and architectures levels. We can compare CAM to the inverse of RAM. When read, RAM produces the data for a given address. Conversely, CAM produces an address for a given data word. When searching for data within a RAM block, the search is performed serially. Thus, finding a particular data word can take many cycles. CAM searches all addresses in parallel and produces the address storing a particular word. CAM supports writing "don't care" bits into words of the memory. The don't care bit can be used as a mask for CAM comparisons; any bit set to don't care has no effect on matches.

The output of the CAM can be encoded or un encoded. The encoded output is better suited for designs that ensure duplicate data is not written into the CAM. If duplicate data is written into two locations, the CAM's output will not be correct. If the CAM contains duplicate data, the un encoded output is a better solution; CAM with un encoded outputs can distinguish multiple data

locations. We can pre-load CAM with data during configuration, or you can write into CAM during system operation. In most cases, two clock cycles are required to write each word into CAM. When you use don't care bits, a third clock cycle is required.

2.1 Operation of CAM:

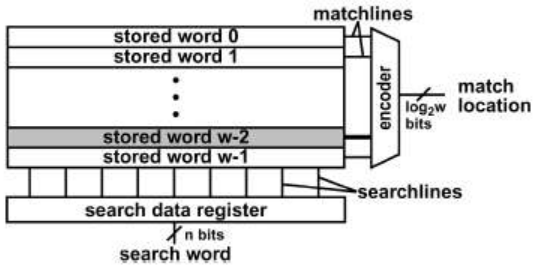


FIG.1 Conceptual view of a content-addressable memory containing w words

Fig.1 shows a simplified block diagram of a CAM. The input to the system is the search word that is broadcast onto the search lines to the table of stored data. The number of bits in a CAM word is usually large, with existing implementations ranging from 36 to 144 bits. A typical CAM employs a table size ranging between a few hundred entries to 32K entries, corresponding to an address space ranging from 7 bits to 15bits.

Each stored word has a match line that indicates whether the search word and stored word are identical (the match case) or are different (a mismatch case, or miss). The match lines are fed to an encoder that generates a binary match location corresponding to the match line that is in the match state. An encoder is used in systems where only a single match is expected.

In addition, there is often a hit signal (not shown in the figure) that flags the case in which there is no matching location in the CAM. The overall function of a CAM is to take a search word and return the matching memory location. One can think of this operation as a fully programmable arbitrary mapping of the large space of the input search word to the smaller space of the output match location. The operation of a CAM is like that of the tag portion of a fully associative cache. The tag portion of a cache compares its input, which is an address, to all addresses stored in the tag memory. In the case of match, a single match line goes high, indicating the location of a match. Many circuits are common to both CAMs and caches; however, we focus on large capacity CAM s rather than on fully associative caches, which target smaller capacity and higher speed.

Today's largest commercially available single-chip CAMs are 18 M bit implementations, although the largest CAMs reported in the literature are 9 M bit in size. As a rule of thumb, the largest available CAM chip is usually about Half the size of the largest available SRAM chip. This rule of thumb comes from the fact that a typical CAM cell consists of two SRAM cells.

2.2 Simple CAM architecture:

Content Addressable Memories (CAMs) are fully associative storage devices. Fixed-length binary words can be stored in any location in the device. The memory can be queried to determine if a particular word, or key, is stored, and if so, the address at which it is stored. This search operation is performed in a single clock cycle by a parallel bitwise comparison of the key against all stored words.

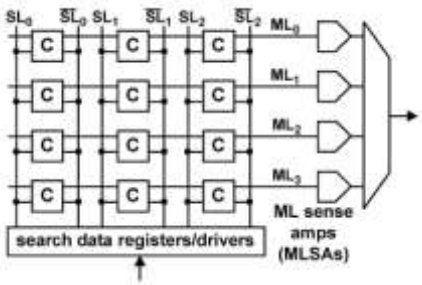


Fig 2. Simple schematic of a model CAM with 4 words having 3 bits each.

We now take a more detailed look at CAM architecture. A small model is shown in Fig. 2. The figure shows a CAM consisting of 4 words, with each word containing 3 bits arranged horizontally (corresponding to 3 CAM cells). There is a match line corresponding to each word (ML0, ML1, etc.) feeding into match line sense amplifiers (MLSAs), and there is a differential search line pair corresponding to each bit of the search word (SL_0, $\overline{SL_0}$, SL_1, $\overline{SL_1}$ etc.). A CAM search operation begins with loading the search-data word into the search-data registers followed by precharging all match lines high, putting them all temporarily in the match state.

Next, the search line drivers broadcast the search word onto the differential search lines, and each CAM core cell compares its stored bit against the bit on its corresponding search lines. Match lines on which all bits match remain in the pre charged-high state. Matchlines that have at least one bit that misses, discharge to ground. The MLSA then detects whether its match line has a matching condition or miss condition. Finally, the encoder maps the match line of the matching location to its encoded address.

2.3 LOW POWER PB-CAM

Since content addressable memory (CAM) is frequently used in applications, that require high-speed searches, and because of its ability to improve application performance by using parallel comparison, it results in reduced search time. But it also significantly increases power consumption. So the main CAM-design challenge is to reduce power consumption associated with the large amount of parallel active circuitry, without sacrificing speed or memory density.

2.3.1 Power saving CAM architecture:

Architectural technique for saving power, which applies to binary CAM, is pre-computation. Pre-computation stores some extra information along with each word that is used in the search operation to save power. These extra bits are derived from the stored word, and used in an initial search before searching the main word. If this initial search fails, then the CAM aborts the subsequent search, thus saving power.

2.4 PB-CAM Architecture:

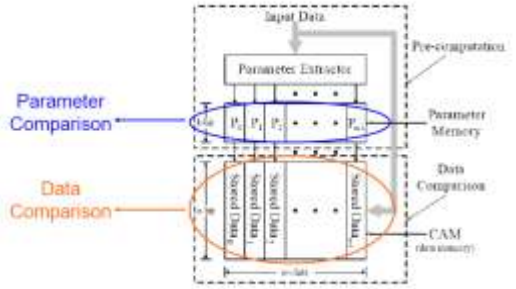


Fig.3 Memory organization of PB-CAM architecture

Fig. 3 shows the memory organization of the PB-CAM architecture which consists of data memory, parameter memory, and parameter extractor, where $k << n$. To reduce massive comparison operations for data searches, the operation is divided into two parts. In the first part, the parameter extractor extracts a parameter from the input data, which is then compared to parameters stored in parallel in the parameter memory. If no match is returned in the first part, it means that the input data mismatch the data related to the stored parameter. Otherwise, the data related to those stored parameters have to be compared in the second part. It should be noted that although the first part must access the entire parameter memory, the parameter memory is far smaller than that of the CAM (data memory). Moreover, since comparisons made in the first part have already filtered out the unmatched data, the second part only needs to compare the data that match from the first part.

The PB-CAM exploits this characteristic to reduce the comparison operations, thereby saving power. Therefore, the parameter extractor of the PB-CAM is critical, because it determines the number of comparison operations in the second part. So, the parameter extractor plays a significant role since this circuit determines the number of comparison operations required in the second part. Therefore, the design goal of the parameter extractor is to filter out as many unmatched data as possible to minimize the required number of comparison operations in the second part. Two parameter extractors are discussed, namely One's count parameter extractor and Block-XOR parameter extractor.

2.5 One's count approach:

For ones count approach, with an n-bit data length, there are n+1 types of one's count (from 0 ones to n ones count). Further, it is necessary to add an extra type of one's count to indicate the availability of stored data. Therefore, the minimal bit length of the parameter is equal to log (n+ 2). The below fig 5 shows the conceptual view of one's count approach. The extra information holds the number of ones in the stored word. For example, in fig.10, when searching for the data word, 01001101, the pre-computation circuit the number of one's (which is four in this case). The number four is compared on the left-hand side to the stored one's count. Only match lines PML_5 and PML7 match, since only they have a one's count of four. In the data-memory stage in fig.3.2, only two comparisons actively consume power and only match line PML5 results in a match. The 14-bit ones-count parameter extractor is implemented with full adders as shown in Fig. 4.

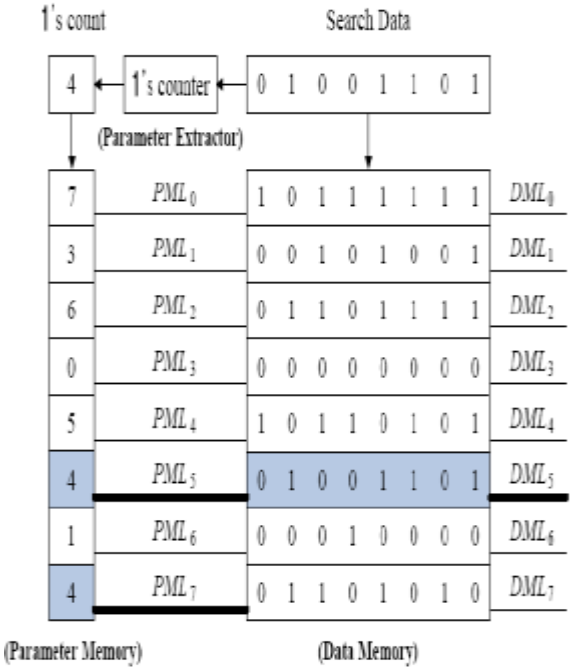


Fig.4 conceptual view of one's count approach

2.6 Mathematical Analysis:

For a 14-bit length input data, all the input data contain 2^{14} numbers, and the number of input data related to the same parameter for ones count approach is $\binom{14}{n}$, where n is a type of ones-count (from 0 to 14 ones-counts). Then we can compute the average probability that the parameter occurs. The average probability can be determined by

$$\text{Average probability} = \frac{\binom{14}{n}}{2^{14}}. \quad (1)$$

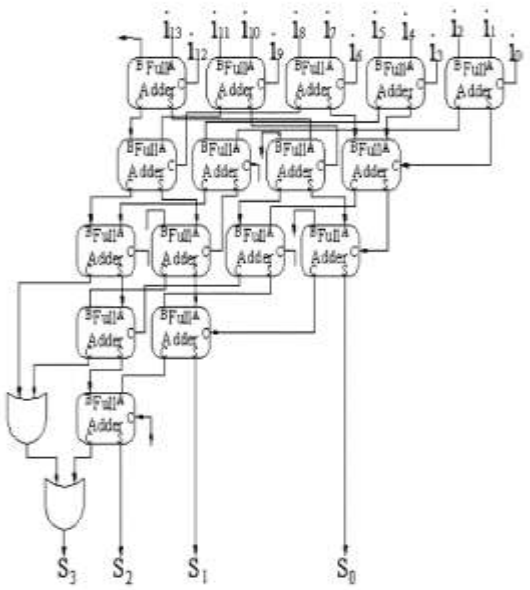


Fig. 5 14-bit ones-count parameter extractor

TABLE I

NUMBER OF DATA RELATED TO THE SAME PARAMETERS AND AVERAGE PROBABILITIES FOR THE ONES COUNT APPROACH

Parameter		Number of data related to the same parameter	Average probability
0000	0	1	0.01%
0001	1	14	0.09%
0010	2	91	0.56%
0011	3	364	2.22%
0100	4	1001	6.11%
0101	5	2002	12.22%
0110	6	3003	18.33%
0111	7	3432	20.95%
1000	8	3003	18.33%
1001	9	2002	12.22%
1010	10	1001	6.11%
1011	11	364	2.22%
1100	12	91	0.56%
1101	13	14	0.09%
1110	14	1	0.01%
1111	15	valid bit	

Table I lists the number of data related to the same parameter and their average probabilities for the input data that is 14-bit in length. For example, if a match occurs in the first part of the comparison with the parameter 2, the maximum number of required comparison operations for the second part is $\binom{14}{2} = 91$. With conventional CAMs, the comparison circuit must compare all stored data, whereas with the ones-count PB-CAMs, a large amount of unmatched data can be initially filtered out, reducing comparison operations for minimum power consumption in some cases. However, the average probabilities of some parameters, such as 0, 1, 2, 12, 13, and 14 are less than 1%.

In Table I, parameters with over 2000 comparison operations range between 5 and 9. However, the summation of the average probabilities for these parameters is close to 82%. Although the number of comparison operations required for ones-count PB-

CAMs is fewer than that of conventional CAMs, ones-count PB-CAMs fail to reduce the number of comparison operations in the second part when the parameter value is between 5 and 9, thereby consuming a large amount of power. From the Table I we can see that random input patterns for the ones-count approach demonstrate the Gaussian distribution characteristic. The Gaussian distribution will limit any further reduction of the comparison operations in PB-CAMs.

2.7 Block –XOR approach:

The key idea behind this method is to reduce the number of comparison operations by eliminating the Gaussian distribution. For a 14-bit input data, if we can distribute the input data uniformly over the parameters, then the number of input data related to each parameter would be $2^{14}/15 = 1093$, and the maximum number of required comparison operations would be $2^{14}/15 = 1093$ for each case in the second part of the comparison process. Compared with the ones-count approach, this approach can reduce comparison operations by a minimum of 909 and a maximum of 2339 (i.e., for parameter value is from 5 to 9) for 82% of the cases. Based on these observations, a new parameter extractor called Block-XOR, which is shown in Fig.3.4, is used to achieve the previous requirement.

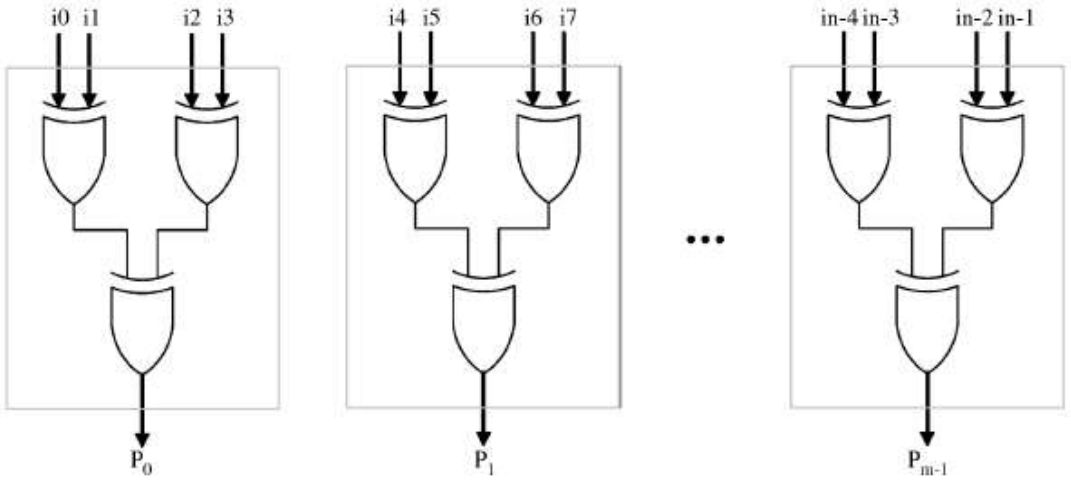


Fig. 6 concept of n-bit Block-XOR block diagram.

In this approach, we first partition the input data bit into several blocks, from which an output bit is computed using XOR logic operation for each of these blocks. The output bits are then combined to become the input parameter for the second part of the comparison process. To compare with the ones-count approach, we set the bit length of the parameter to [log (n+ 2)]. Where n is the bit length of the input data. Therefore, the number of blocks is [n/ log (n+2)] in this approach. Taking the 14-bit input length as an example, the bit length of the parameter is log (14+2) = 4-bit, and the number of blocks is [14/ log(14+2)] = 4 . Accordingly, all the blocks contain 4 bits except the last one, which contains the remainder 2 bits as shown in the upper part of Fig. 6.

However, the concept of Block-XOR approach does not provide a, valid bit for checking whether the data is valid; hence it cannot be applied to the PB-CAM directly. For this reason, modified architecture is used as shown in the lower part of Fig. 6 to provide a valid bit and to guarantee the uniform distribution property of the Block-XOR approach. We added a multiplexer to select the correct parameter.

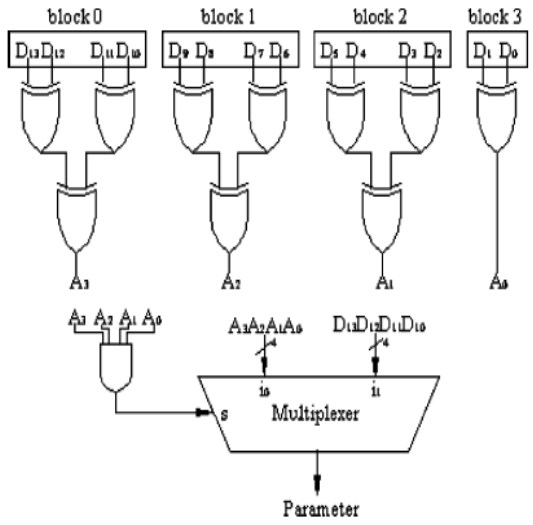


Fig. 7: Structure of Block-XOR approach with valid bit.

The selected signal is defined as

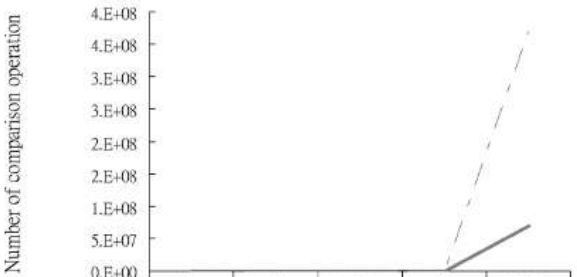
S=A3A2A1A0. (2)

According to (2), if the parameter is "0000 to 1110" (S = "0"), the multiplexer will transmit the i0 data as the output. In other words, the parameter does not change. Otherwise, (A3A2A1A0 =''1111", S =''1"), the first block of the input data becomes the new parameter, and "1111" can then be used as the valid bit. The case where the first block is "1111" was not considered, because the "1111" block bits will result in "0" for one of the four parameter bits.

2.8 Comparison between Two Approaches:

To eliminate the Gaussian distribution, we uniformly distribute the parameter over the input data. However, as can be seen from Tables III and IV, when the parameter is 0, 1, 2, 3, 4, 10, 11, 12, 13, or 14, the number of comparison operations required for the ones-count approach is fewer than that for the Block-XOR PB-CAM. Although the Block-XOR PB-CAM is better than the ones-count PB-CAM only for parameters between 5 and 9, we must draw attention to the fact that the probability that these parameters occurs is 82%. For example, when the parameter is 7, there is a 20.95% chance that the Block-XOR PB-CAM can result in more than 2280 fewer comparison operations as compared to the ones-count approach. Compared with the ones-count approach, we can reduce the number of comparison operations for more than 1000 in most cases. In other words, the ones-count approach is better than Block-XOR approach in only 18% of the cases.

The number of comparison operations required for different input bit length 4, 8, 14, 16, and 32 bits is shown in Fig.8. As can be seen, from the fig 3.6 Block-XOR PB-CAM becomes more effective in reducing the number of comparison operations as the input bit length increases. This implies that the longer the input bit length is, the fewer the number of comparison operations required (i.e., power reduction). Therefore, the Block-XOR PB-CAM is more suitable for wide-input CAM applications. In addition, the Block-XOR parameter extractor can compute parameter bits in parallel with three XOR gate delays for any input bit length, hence short constant delay. On the contrary, as the input bit length increases, the delay of the ones-count parameter extractor will increase significantly.



	4 bits	8 bits	14 bits	16 bits	32 bits
Ones count	5	48	2689	5375	370888989
Block-XOR	2	20	1152	2304	69206016

Fig.8.Comparision operations for different input bit length.

III Gate-Block Selection Algorithm:

To make the parameter extractor of the block-xor PB-BAM more useful for specific data types, we take into account the different characteristic of logic gates to synthesize the parameter extractors for different data types. As can be seen in Fig. 3.5, if the input bits of each partition block is set into l, the bit length of the parameter (i.e. the number of blocks) will be [n/l], where n is the bit length of the input data, and then the levels in each partition block equal [$\log_2$ l]. We observe that when the input bits of each partition block decreases, the mismatch rate and the number of comparison operations in each data comparison process will decrease (this is because that the combinations of the parameter increase). Although the increasing parameter bit length can decrease the mismatch rate and the number of comparison operations in each data comparison process, the parameter memory size must be increased. In other words, it increases the power consumption of the parameter memory as well. As we stated, when the PBCAM performs data searching operation, it must compare the entire parameter memory. To avoid wasting the large amount of power in the parameter memory, we set the input of each partition block to 8 bits. Fig. 3.7 shows the proposed parameter extractor architecture. We first partition the input data bit into several blocks, G0~G6 in each block stand for different logic gates, from which an output bit is computed using synthesized logic operation for each of these blocks. The output bits are then combined to become the parameter for data comparison process.

The objective of our work is to select the proper logic gates in Fig. 9 so that the parameter (Pk−1, Pk−2, · · ·, P0) can reduce the number of data comparison operations as many as possible.

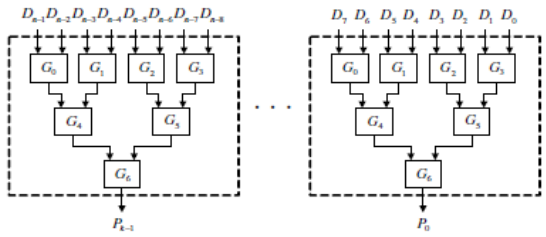


Fig. 9: n-bit block diagram of the proposed parameter extractor architecture.

In our proposed parameter extractor, the bit length of the parameter is set into [n/8], and then the levels in each partition block equal [$\log_2$ 8] (which is 3). Suppose that we use basic logic gates (AND, OR, XOR, NAND, NOR, and NXOR) to synthesize a parameter extractor for a specific data type, which has $(6^7)^{[n/8]}$ different logic combinations based on the proposed parameter extractor. Obviously, the optimal combination of the parameter extractor cannot be found in polynomial time.

To synthesize a proper parameter extractor in polynomial time for a specific data type, we propose a gate-block selection algorithm to find an approximately optimal combination. We illustrate how to select proper logic gates to synthesize a parameter extractor for specific data type from mathematical analysis below.

3.1 Mathematical Analysis:

For a 2-input logic gate, let p be the probability of the output signal Y that is one state. The probability mass function of the the output signal Y is given by

$$P_Y(y) = \begin{cases} 1-p & y=0, \\ p & y=1, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Assume that the inputs are independent, if we use any 2-input logic gate as a parameter extractor to generate the parameter for 2-bit data, then the PB-CAM requires the average number of comparison operations in each data search operation can be formulated as

$$\begin{aligned} C_{avg} &= N_0(1-p) + N_1 \cdot p \\ &= N_0\left(\frac{N_0}{N_0+N_1}\right) + N_1\left(\frac{N_1}{N_0+N_1}\right) \\ &= \frac{{N_0}^2 + {N_1}^2}{N_0+N_1} \end{aligned} \quad (4)$$

Where N0 is the number of zero entries, and N1 is the number of one entries for the generated parameters. To illustrate clearly, we use Table V as an example.

TABLE II

TRUTH TABLE AND AVERAGE NUMBER OF COMPARISON OPERATIONS OF BASIC LOGIC GATES FOR A 2-BIT SKEW DATA

A	B	AND	OR	XOR	NAND	NOR	NXOR
0	0	0	0	0	1	1	1
1	0	0	1	1	1	0	0
1	0	0	1	1	1	0	0
0	0	0	0	0	1	1	1
1	1	1	1	0	0	0	1
0	0	0	0	0	1	1	1
C_{avg}		4.33	3	3.33	4.33	3	3.33

Suppose that a 2-input AND gate is used to generate the parameter, the average number of comparison operations in each data search operation for the PB-CAM can be derived:

$$C_{avg} = \frac{5^2+1^2}{5+1} = 4.33 \quad (5)$$

In other words, when we use a 2-input AND gate to generate the parameter for this 2-bit data, the average number of comparison operations required for each data search operation in the PB-CAM is 4.33. According to Equ. 4, Table V derives the average number of comparison operations for six basic logic gates. Obviously, using OR and NOR gates are the best selection for this case, because they require the least average number of comparison operations (which is 3). Moreover, when we use the inverse relation of logic gates (AND/NAND, OR/NOR, and XOR/NXOR) to generate the parameter, the average number of comparison operations for each data search operation required in the PB-CAM will be the same. To reduce the complexity of our proposed algorithm and the performance of the parameter extractor, our proposed approach only selects NAND, NOR, and XOR gates to synthesize the parameter extractor for our implementation. This is because that NAND and NOR is better than AND and OR in terms of the area, power, and speed. Based on this mathematical analysis, Fig.3.8 shows our proposed gate block selection algorithm.

Note that when the input is random, the synthesized result will be the same as the block-XOR approach. In order words, the block-XOR approach is a subset of our proposed algorithm. To better understand our proposed approach, we give a simple example as

illustrated in figure. In this example, a 4-bit data is assigned as input data. Because the input data is only 4 bits in this example, we set the number of input bits of each partition block to 4, and then the levels in each partition block equal [log2 4] (i.e. two levels).

IV RESULT:

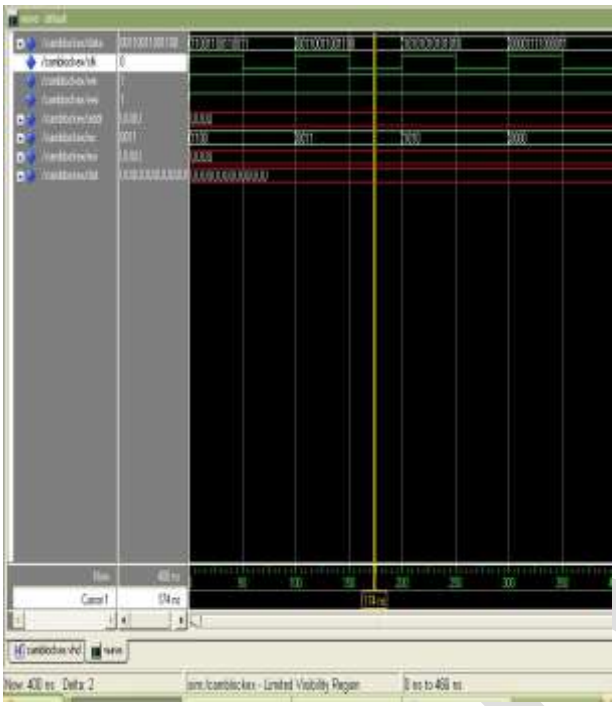


Fig.5.11.VHDL output showing the data write into the CAM

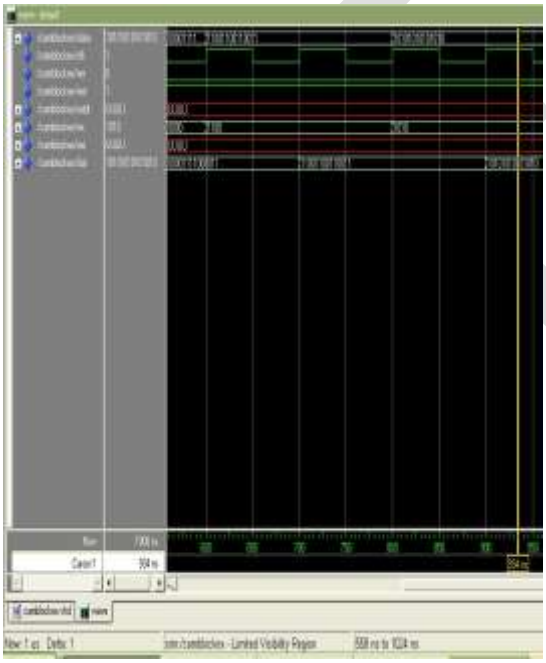


Fig.5.12.VHDL output showing the data read from the CAM

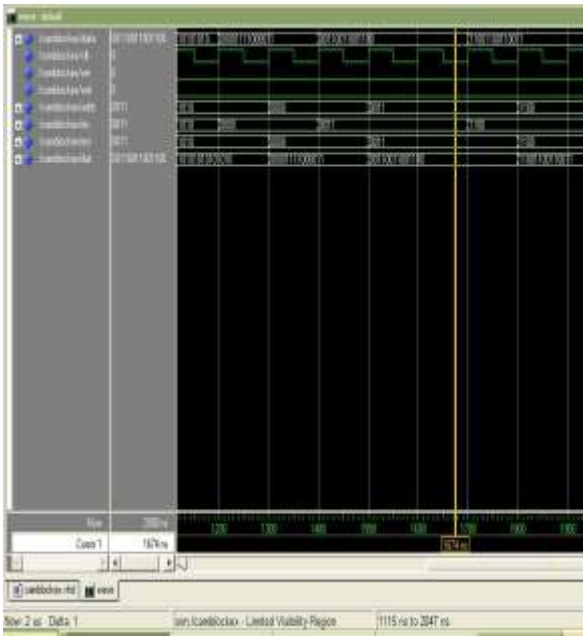


Fig.5.13.VHDL output showing the address read from the CAM

V. CONCLUSION:

In this 14-bit low power pre computation–based content addressable memory (PB-CAM) is simulated in VHDL. Mathematical analysis and simulation results confirmed that the Block-XOR PB-CAM can effectively save power by reducing the number of comparison operations in the second part of the comparison process. In addition, it takes less area as compared with the one's count parameter extractor. This PB-CAM takes data as input and gives the address pointing to the same data as well as different data as an output exactly after one clock cycle. So it is flexible and adaptive for the low power and high speed search applications.

In this synthesis, a gate-block selection algorithm was proposed. The proposed algorithm can synthesize a proper parameter extractor of the PB-CAM for a specific data type. Mathematical analysis and simulation results confirmed that the proposed PB-CAM effectively save power by reducing the number of comparison operations in the data comparison process. In addition, the proposed parameter extractor can compute parameter bits in parallel with only three logic gate delays for any input bit length (i.e. constant delay of search operation).

REFERENCES

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

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

[3] I. Arsovski, T. Chandler, and A. Sheikholeslami, "A ternary contentaddressable memory (TCAM) based on 4T static storage and includinga current-race sensing scheme," IEEE J. Solid-State Circuits, vol. 38,no. 1, pp. 155–158, Jan. 2003.'

[4] I. Arsovski and A. Sheikholeslami, "A mismatch-dependent power allocationtechnique for match-line sensing in content-addressable memories,"IEEE J. Solid-State Circuits, vol. 38, no. 11, pp. 1958–1966,Nov. 2003.

[5] Y. J. Chang, S. J. Ruan, and F. Lai, "Design and analysis of low power cache using two-level filter scheme," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 4, pp. 568–580, Aug. 2003.

[6] K. Vivekanandarajah, T. Srikanthan, and S. Bhattacharyya, "Dynamic filter cache for low power instruction memory hierarchy," in Proc. EuromicroSymp. Digit. Syst. Des., Sep. 2004, pp. 607–610.

[7] R. Min, W. B. Jone, and Y. Hu, "Location cache: A low-powre L2cache system," in Proc. Int. Symp. Low Power Electron. Des., Apr.2004, pp. 120–125.

[8] K. Pagiamtzis and A. Sheikholeslami, "Using cache to reduce power in content-addressable memories (CAMs)," in Proc. IEEE Custom Integr.Circuits Conf., Sep. 2005, pp. 369–372.

[9] C. S. Lin, J. C. Chang, and B. D. Liu, "A low-power precomputationbased fully parallel content-addressable memory," IEEE J. Solid-State Circuits, vol. 38, no. 4, pp. 622–654, Apr. 2003.

[10] K. H. Cheng, C. H.Wei, and S. Y. Jiang, "Static divided word matching line for low-power content addressable memory design," in Proc. IEEEInt. Symp. Circuits Syst., May 2004, vol. 2, pp. 23–26.

[11] S. Hanzawa, T. Sakata, K. Kajigaya, R. Takemura, and T. Kawahara,"A large-scale and low-power CAM architecture featuring a one-hotspot block code for IP-address lookup in a network router," IEEE J. Solid-State Circuits, vol. 40, no. 4, pp. 853–861, Apr. 2005.

[12] Y. Oike, M. Ikeda, and K. Asada, "A high-speed and low-voltage associative co-processor with exact Hamming/Manhattan-distance estimation using word-parallel and hierarchical search architecture," IEEE J. Solid-State Circuits, vol. 39, no. 8, pp. 1383–1387, Aug. 2004.

[13] K. Pagiamtzis and A. Sheikholeslami, "A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme,"IEEE J. Solid-State Circuits, vol. 39, no. 9, pp. 1512–1519, Sep. 2004.

[14] D. K. Bhavsar, "A built-in self-test method for write-only content addressablememories," in Proc. 23rd IEEE VLSI Test Symp., 2005, pp.9–14.