A Low Power CAM Design using Block-XOR Precomputation Approach

Shanq-Jang Ruan
Department of Electrical Engineering
National Taiwan University of Science and Technology
sjruan@et.ntust.edu.tw

Chi-Yu Wu
Department of Electronic Engineering
National Taiwan University of Science and Technology
M9302124@mail.ntust.edu.tw

Chun-Chih Chen
Department of Electrical Engineering
National Taiwan University of Science and Technology
M9302125@mail.ntust.edu.tw

Abstract: In modern process design, the on-chip cache is often used to boost the system performance. However, the on-chip cache usually consumes a significant amount of power in processor. In this paper, we propose a new PB-CAM structure for saving cache access power. Our approach is based on the precomputation-based content addressable memory (PB-CAM) [1]. Although the PB-CAM can eliminate the comparison operations to reduce power consumption by precomputation, it suffers from that the ones count approach limits the reduction amount of comparison operations. Therefore, we devise a Block-XOR approach to improve the efficiency of PB-CAM. In the experiment, we estimate the power by Synopsys **PrimePower**. Compared to [1], the experimental results show that our approach achieves 89% reduction of the power-delay product in parameter extractor. In addition, it results in 21% reduction in total cache power consumption.

Key–Words: Cache, Content addressable memory, Low-Power, Embedded System.

1 Introduction

Content addressable memory (CAM) is a major device in asynchronous transfer mode (ATM), communication networks, and lookup tables due to its high-speed data searching capability [2]-[3]. In order to achieve high speed searching, the CAM contains a large amount of stored data for simultaneous comparison with the input character, and an address from among those matches of comparison is sent to the output.

In order to speedup the comparison operation, CAM requires an enormous number of simultaneous comparison operations. The operations consumes a large amount of power. Some researches reduced match-line power by directly reducing the voltage swing on the match-lines [4, 5], or by using current-based techniques to indirectly reduce the match-line voltage swing [6, 7]. The selective precharge technique reduced match-line power consumption with breaking the search into two segments, and observing that the second segment is rarely activated [8]. Although many works have been done on this research area, the power consumption in CAM is still high in comparison with RAM of similar size. Therefore, reducing the number of comparisons is very significant for optimizing cache power. Some researches in CAM design for reducing comparison at the architecture level continue to increase. For example, Lin et al. presented a precomputation-based content addressable memory (PB-CAM) to reduce the number of comparison operation[1]. Their architecture is composed of the data memory, the parameter memory, and the parameter extractor as shown in Figure 1. The parameter extractor is very important because it determines the number of comparison operations. The PB-CAM used ones count approach to reduce the comparison operations, thereby saving power. However, the parameter extractor implemented by ones count approach decreases the performance of PB-CAM when the input data is random. In this paper, we devise a Block-XOR approach to replace the ones count approach to improve the performance of PB-CAM. By mathematical analysis, compared to [1] our Block-XOR approach can reduce 50% comparison operations.

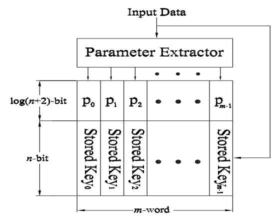

Figure 1: The memory organization of the PB-CAM architecture.

The remainder of this paper is organized as follows. In Section 2, we briefly describe the PB-CAM architecture. Our new architecture is described in Section 3, where the design of the novel parameter extractor, block-Xor is provided. In Section 4, we provide mathematical analysis to prove the effectiveness of our new architecture. Finally, we estimate the power dissipation of our and the original approaches in the Section 5. Finally, we draw the conclusion in Section 6.

2 PREVIOUS WORK and OBSERVATION

In order to understand our approach more clearly, we briefly describe the architecture of the PB-CAM in this section [1]. Figure 1 shows the memory organization of the PB-CAM architecture. Throughout this paper, we assume that the input data for data searching function is random. The architecture consists of the data memory, the parameter memory, and the parameter extractor. In order to reduce massive comparison operation for the data searching function, the operation is divided into two parts. In the first part, the parameter extractor extracts the parameter of the input data, then many comparison circuits compare the parameter of the input data with the parameters stored in parameter memory in parallel. If no match occurs in the first part, it means that the input data mismatch the data related to this stored parameter. Otherwise, the data related to this stored parameter has to be identified in the second part. With the results of the first part, the CAM word circuits can only compare the input data with those unidentified data to identify any match for the second part.

As we stated above, the parameter extractor is very significant for comparison power, since this circuit determines the remainder of unidentified data after the first comparison process. Therefore, the design target for the parameter extractor is to filter out the unmatched data as more as possible for reducing the number of second comparison. In [1], Lin adopted the ones count function to perform the parameter extractor because the function is efficient in filtering comparison operations. When input data is n-bit length, the number of ones count is n+1, plus 1 is used to express an additional state of ones count to indicate the availability of stored data. Hence, based on the parameter extractor function, the minimal bit length of parameter is equal to $\lceil \log(n+2) \rceil$. In the PB-CAM architecture, the parameter extractor is implemented by many full two bits adders. Figure 2 illustrates the design of the parameter extractor for ones count approach. The circle element is defined as follows. With n bits input data, the circle element uses $\lceil n/3 \rceil$ adders to generate partial addition of the n bits input data in parallel. Next it outputs $\lceil n/3 \rceil$ carry signals to the left link and $\lceil n/3 \rceil$ sum signals to the right link at the same time. According to the definition, the circuit design of parameter extractor is built. The design is shown in Figure 3. Throughout this paper, we will use a 14 bits example to build the PB-CAM and our proposed circuits for fair comparison.

The parameter extractor filters a large amount of unmatched data to reduce the number of comparison operations for low power. For a 14 bits length input data, all the input data contain 2^{14} numbers, and the number of input data related to the same parameter is C_n^{14} , where n is a kind of ones count (from 0 ones count to 14 ones count). Then we can compute the average probability that the parameter occurs. The average probability is defined as

$$Average\ Probability = (C_n^{14})/(2^{14}). \quad (1)$$

Table 1 lists the average probability for the 14 bits length input data. If a match occurs in the first part by the parameter is 2, the number of comparison operations is $C_n^{14} = 91$ at most for the second part. Compared to the traditional CAM, the comparison circuit must compare all stored data. Obviously, the PB-CAM can filter a large amount of unmatched data to reduce the number of comparison operations for low power in some cases. However, the average probability of some parameters such as 0, 1, 2, 12, 13 and 14 are less than 1%. That would be waste power to the first part comparison, because the first part circuit can not effectively filter out the unnecessary comparison for second part.

In Table 1, the number of comparison operations is over 2000 for the parameter value ranges from 5 to 9. As mentioned above, the average probabilities for those parameters (1, 2, 12, 13 and 14) are almost about 82%. Although the number of comparison operations is less than that of the traditional CAM, the PB-CAM fail to reduce the number of comparison operations in the second part when the range of parameter value is from 5 to 9, thereby consume a large amount of power. We show the distribution of the input data among the all parameters in Figure 4 for observation. We can see that the curve in Figure 4 performs normal distribution characteristic. The normal curve is called Gaussian distribution. Note that the Gaussian distribution will limit the further reducing of comparison operations in PB-CAM. Thus, we propose a new parameter extractor to replace the ones count approach. Our approach presents the same number of comparison operations for all parameters, and less comparison operations than that of the ones count approach. This is especially suitable for real-time embedded system.

Table 1: Average probability and the number of data related to the parameter.

Parameter		Number of data related to the 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	

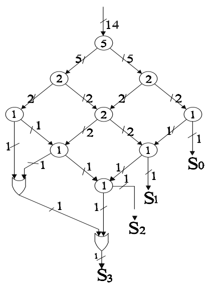

Figure 2: The 14 bits block diagram of parameter extractor.

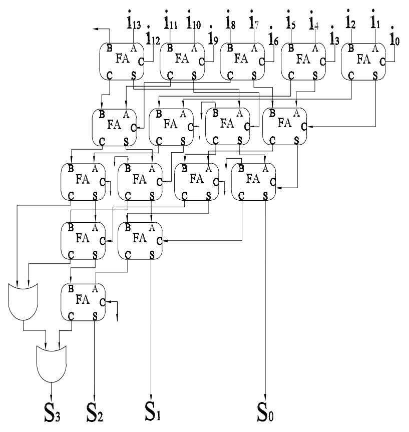

Figure 3: The 14 bits ones count of parameter extractor.

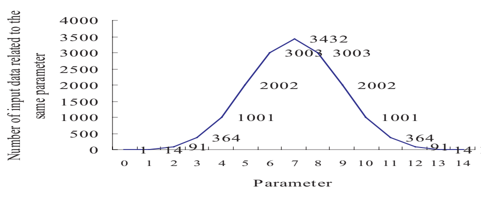

Figure 4: The distribution of the input data among the all parameters.

3 PROPOSED APPROACH

The key idea of our method is to reduce the comparison operations and eliminate the Gaussian distribution. In order to eliminate the Gaussian distribution, we distribute the input data among parameter uniformly. For example, the number of input data related to the same parameter is $\frac{2^{14}}{15}$ for a 14 bits length input data so that if we assume the first access results in a parameter-hit, the number of comparison operations will be 1093 at most in the second comparison process. Compared with the ones count approach, our approach can reduce about 1000 comparison operations at least for 82% cases (the range of parameter value is from 5 to 9). According to the observation, we propose a new parameter extractor to achieve above requirement. We name this approach as *Block-XOR* method.

Under our method, for obtaining the new parameter, the bits length of input data is partitioned into several blocks, and then computes an output bit for each block by XOR logic operation. Note that the parameter consists of these output bits. Figure 5 shows the structure of our Block-XOR approach. In order to compare with the ones counts approach, we set the bit length of parameter to $\lceil \log(n+2) \rceil$ where n is the length of input data and assume the 14 bits length input data in Block-XOR approach. Therefore, the number of block is $\lceil n/\log(n+2) \rceil$. For our example, the length of parameter is $\lceil \log(14+2) \rceil = 4$ bits, hence the number of block is $\lceil 14/\log(14+2) \rceil = 4$. Accordingly, all the blocks will contain 4 bits except the last one, which only contains 2 bits.

Based on the above discussion, we can obtain a new parameter bit from each block. First we list all possible states in 4-bit block, as shown in Table 2. In summary, the number of the parameter to be ”0” or ”1” will be equal. For the 2-bit block, Table 3 shows the same result. Afterward, we can calculate the number of input data related to the new parameter and the average probability that the parameter occurs by rule of product. Table 4 lists the average probability and the number of input data related to the same parameter for all 14 bits length input data. For example, the parameter ”0110” consists of the two bits signals ”1” and two bits ”0”, we can obtain the ”0” and ”1” signals from Table 2. each signal has eight kinds in four bits data. As can be seen in Table 3,

Table 2: The parameter of 4-bit block.

Block bits	A output bit of parameter	Block bits	A output bit of parameter
0000	0	1000	1
0001	1	1001	0
0010	1	1010	0
0011	0	1011	1
0100	1	1100	0
0101	0	1101	1
0110	0	1110	1
0111	1	1111	0

the block bits of ”00” and ”11” corresponds to ”0”, and ”01” and ”10” correspond to ”1” respectively. By rule of product, the number of input data related to the same parameter is $8 \times 8 \times 8 \times 2 = 1024$. Consequently, the average probability can be determined as $1024/(1024 \times 16) \times 100\% = 6.25\%$.

Table 4 indicates the uniform property of our Block-XOR method. We can see that the number of comparison operations is 1024 at most for each parameters. Obviously, our approach reduces the number of comparisons and equalizes the comparison time, hence low power and real time. However, compared to Table 4, our approach (Table 4) does not provide valid bit for checking whether the data is valid or not. For this reason we provide a valid bit. Furthermore, in order to guarantee the uniformly distribution property of Block-XOR approach, we modified the architecture of our Block-XOR approach as shown in Figure 6. First, we add a multiplexer to select the output parameter. The select line is defined as

$$S = A_3A_2A_1A_0 \tag{2}$$

According to the above function, if the parameter is ”0000–1110”(S = 0), the multiplexer will transmit the i_0 to the output, In other words, the parameter does not change. Otherwise, ($A_3A_2A_1A_0$ = ”1111”, S = 1) the first block of input data will become the new parameter. Then the ”1111” now can be used as valid bit. We do not consider the situation of the first block is ”1111” because the ”1111” block bits will result in ”0” output bit. Table 5 presents the number of input data related to the parameter for the modified Block-XOR approach. When the parameter is ”1111”, the new parameter is provided by the first block which the output bit is ”1”, hence the number of input data related to those parameter is 1024+(1024/8) = 1152 and the average probability is (1152/(1024×7+1152×8))×100%=7.03%. If the parameter is not ”1111”, the number of input data related to the same parameter is 1024 and the average probability is (1024/(1024×7+1152×8))×100%=6.25%.

Table 3: The parameter of 2-bit block.

Block bits	A output bit of parameter
00	0
01	1
10	1
11	0

Table 4: The average probability and the number of input data related to the same parameter for the 14 bits length input data.

Parameter		Number of data related to the parameter	Average probability
0000	0	8×8×8×2=1024	6.25%
0001	1	8×8×8×2=1024	6.25%
0010	2	8×8×8×2=1024	6.25%
0011	3	8×8×8×2=1024	6.25%
0100	4	8×8×8×2=1024	6.25%
0101	5	8×8×8×2=1024	6.25%
0110	6	8×8×8×2=1024	6.25%
0111	7	8×8×8×2=1024	6.25%
1000	8	8×8×8×2=1024	6.25%
1001	9	8×8×8×2=1024	6.25%
1010	10	8×8×8×2=1024	6.25%
1011	11	8×8×8×2=1024	6.25%
1100	12	8×8×8×2=1024	6.25%
1101	13	8×8×8×2=1024	6.25%
1110	14	8×8×8×2=1024	6.25%
1111	15	8×8×8×2=1024	6.25%

4 MATHEMATICAL ANALYSIS

To eliminate Gaussian distribution, we uniformly distributes the parameter among the input data. However, when the parameter is 0, 1, 2, 3, 4, 10, 11, 12, 13 and 14, we can observe that the comparison operations of ones count approach is less than that of the Block-XOR approach from Table 1 and Table 5. Although our approach is better than ones count approach only for the parameter from 5 to 9, we must draw attention to the probability of the parameter occurring is 82%. For example, when the parameter is

Table 5: The average probability and the number of input data related to the same parameter for the modified Block-XOR.

Parameter		Number of data related to the parameter	Average probability
0000	0	1024	6.25%
0001	1	1024+(1024/8)	7.03%
0010	2	1024+(1024/8)	7.03%
0011	3	1024	6.25%
0100	4	1024+(1024/8)	7.03%
0101	5	1024	6.25%
0110	6	1024	6.25%
0111	7	1024+(1024/8)	7.03%
1000	8	1024+(1024/8)	7.03%
1001	9	1024	6.25%
1010	10	1024	6.25%
1011	11	1024+(1024/8)	7.03%
1100	12	1024	6.25%
1101	13	1024+(1024/8)	7.03%
1110	14	1024+(1024/8)	7.03%
1111	15	Valid bit	6.25%

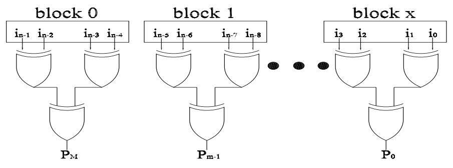
block 0
block 1
block x
i_{n-1} i_{n-2} i_{n-3} i_{n-4}
i_{n-5} i_{n-6} i_{n-7} i_{n-8}
i_3 i_2 i_1 i_0
P_M
P_{m-1}
P_0

Figure 5: The n bits Block-XOR approach block diagram.

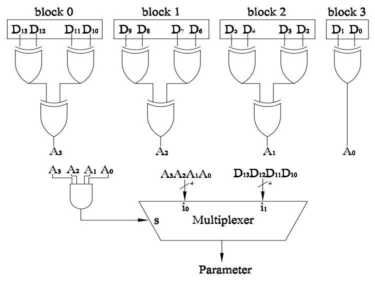

Figure 6: The n bits Block-XOR approach block diagram.

7 in the 14 bits input data, we have 20.94% chance to reduce the comparison operations for more than 2280 in contrast to the ones count approach. By summation, the probability of the parameter from 5 to 9 is 82%. Therefore, compared to ones count approach, we can reduce the number of comparison operations for more than 1000 at most cases. In other words, for only 18%, the ones count approach is better than our approach. Further, when the bit length of input data is 30 bits, the Block-XOR approach has 90% chance to reduce the number of comparison operations for more than 2.107 millions in comparison with the ones count approach. In conclusion, the Block-XOR can reduce almost half of comparison operations of ones count approach in comparison circuit for 14-bit length input data. Moreover, due to its uniformly distributed characteristic, it is even more suitable for real-time system.

5 Experimental Result

Accordingly the Block-XOR approach can reduces much more comparison operations than that of the ones count approach. In this section, we demonstrate that the power consumption between ones count and our Block-XOR approaches for verification. To compare two approaches, we adopt the same memory access power dissipation from [1]. The parameter extractors for the ones count and Block-XOR approaches have been synthesize by Synopsys **Design Compiler** with TSMC $0.25\mu m$ technology at $2.5V$. After that, we estimate the power by Synopsys **Prime-Power**. In the experiment, the input patterns are random and 14 bits length. The power dissipation of the parameter extractor for the ones count and Block-XOR approaches are tabulated in Table 6. From Table 6, we can easily observe that the delay and average power in Block-XOR are much smaller than that of ones count; therefore the power-delay product is far less than that of ones count. For clearly, we list the reduction of power-delay product in Table 7. The reduction of power-delay product is up to 89%.

Table 6: Experimental result in the parameter extractor for 14-bits input data.

	Ones count approach	Block-XOR approach
Delay time (ps)	4.70	1.28
Average power (mW)	4.69	1.86
Power-Delay product (fJ)	22.04	2.39

From [1], the power consumption of access memory is 86 **fJ** per bit for each search operation, and search access time is 10 ns. We assume that the data memory size is 214 words, and each word is 14 bits. As a result, the total power can be computed as:

$$\begin{aligned}&Total\ power =\\&Power_{Parameter\ extractor\ circuit}\\&+No.\ of\ first\ comparison_{Parameter}\\&\times Power_{Access\ parameter\ memory\ for\ 4bits\ data}\\&+No.\ of second\ comparison_{Data}\\&\times Power_{Access\ data\ memory\ for\ 14bits\ data}\end{aligned} \tag{3}$$

The total power for search operation in PB-CAM is shown in Table 7. For the **second** part comparison in the ones count approach, the average probability of active comparator operation is about 82% for the parameters value ranges from 5 to 9, therefore the number of comparison operation can be calculated as (2002×2+3003×3+3432)/5=2688. Compared with our approach, we set the number of the second part comparison operation to the worst case, i.e. 1152. Although the assumption is disadvantage to our Block-XOR approach, the power reduction rate is also up to 57% in the second part comparison. It implies that our Block-XOR parameter extractor can effectively filler out for more unnecessary comparison operation than ones count does. The reduction is computed as:

$$\begin{aligned}&Power\ Reduction =\\&\frac{Power_{Ones\ count}-Power_{Block-XOR}}{Power_{Ones\ count}}\\&\\&Comparisons\ Reduction =\\&\frac{No.\ of\ comparisons_{Ones\ count}-No.\ of\ comparisons_{Block-XOR}}{(No.\ of\ comparisons_{Ones\ count}}\end{aligned} \tag{4}$$

Discusstion:

Table 7: Comparison of two approaches.

	Ones count approach	Block-XOR approach	Reduction
$Power_{Parameter\ extractor}$ (fJ)	22.04	2.39	89%
$No.\ of\ first\ comparison_{Parameter}$	2^{14}	2^{14}	0
$Power_{Access\ parameter\ memory\ for\ 4bits\ data}$ (fJ)	344	344	0
$Power_{First\ comparison}$(nJ)	5.63	5.63	0
$No.\ of\ second\ comparison_{Data}$	2688	1152	57%
$Power_{Access\ data\ memory\ for\ 14bits\ data}$ (fJ)	1204	1204	0
$Power_{Second\ comparison}$ (nJ)	3.24	1.39	57%
Total Power (nJ)	8.87	7.02	21%

From Table 7, we can see that 21% of power consumption is saved by reducing the number of comparison operations. It should be noted that when the parameter bit length is the same, the power of the first part comparison is the same for two approaches. In other words, if the parameter length is constant, the power consumption of the first comparison for any approach is the same for the PB-CAM structure. Due to the bit length of the parameter is fixed in this paper, we only have to focus on the power consumption in the second comparison. As shown in Table 7, our approach reduces 57% power dissipation for second part comparison. In summary, the Block-XOR approach not only reduces the comparison operations in memory but also decreases the power consumption of parameter extractor building block. It is intuitively that the circuit structure of Block-XOR is much simpler than that of one count.

6 Conclusion

In this paper, a Block-XOR parameter extractor for low power PB-CAM is proposed. The mathematical analysis and experimental results confirmed that our approach can effectively save the power not only for comparison circuits but also for memory. In order to eliminate the Gaussian distribution in original design, we used the Block-XOR approach to devise the parameter extractor. By our mathematical analysis, the number of comparison operation is far less than that of the ones count approach so that the Block-XOR approach reduced the power dissipation significantly. Moreover we implemented the parameter extractor to prove that the power consumption in our approach is less than that of ones count. Compared to [1], the experimental results shows that our approach reduces power-delay up to 89%. In addition, it obtains 21% reduction in total cache power consumption.

References:

[1] C. S. Lin, J. C. Chang, and B. D. Liu, A Low-Power Pre-computation-Based Fully Parallel Content-Addressable Memory, *IEEE Journal of Solid-State Circuits,* Vol. 38, April 2003 pp. 654-622.

[2] T. Ogura, M. Nakanishi, and T. Nakabayshi, and R. Kasai, A 336-kb content-addressable memory for highly parallel image processing, in*Proc. IEEE Custom Integrated Circuit Conf.,* 1996, pp. 13.4.1-13.4.4.

[3] K. Nogami, T. Sakurai, K. Sakaue, Y. Miyazawa, S. Tanaka, Y. Hiruta, K. Katoh, T. Takayanagi, T. Hirotori, Y. Itoh, and M. Uchida, A 9-ns hit-delay 32-kB cache macro for high-speed RISC, *IEEE J. Solid-state Circuits,* vol. 25, pp. 100-108, Feb. 1990.

[4] H. Miyatake, M. Tanaka, and Y. Mori, A design for high-speed low power CMOS fully parallel content-addressable memory macros, *IEEE J. Solid-State Circuits,* vol. 36, pp. 956-968, June 2001.

[5] I. Arsovski, T. Chandler, and A. Sheikholeslami, A ternary content-addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme,*IEEE J. Solid-State Circuits,* vol. 38, pp.155-158, Jan. 2003.

[6] I. Arsovski and A. Sheikholeslami, A mismatch-dependent power allocation technique for match-line sensing in content-addressable memories, *IEEE J. Solid-State Circuits,* vol. 38, pp. 1958-1966, Nov. 2003.

[7] C. A. Zukowski and S. Y. Use of selective precharge for lowpower content-addressable memories, in*Proc. IEEE Int. Symp. Circuits and Systems,* vol. 3, 1997, pp. 1788-1791.

[8] S. Hall, G. Hall, and J. McCal, High-Speed Digital System Design,*Power-Driven Micro-architecture Workshop In Conjunction With ISCA98 in Bar-celona,*June 1998.