Synthesis and Design of Parameter Extractors for Low-Power Pre-computation-Based Content-Addressable Memory Using Gate-Block Selection Algorithm

Jui-Yuan Hsieh

Department of Electronic Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan, 106
Tel: +886-2-2733-3141 ext 7204
Fax: +886-2-2737-6424
e-mail: m9502104@mail.ntust.edu.tw

Shanq-Jang Ruan

Department of Electronic Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan, 106
Tel: +886-2-2737-6411
Fax: +886-2-2737-6424
e-mail: sjruan@mail.ntust.edu.tw

Abstract— 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 fast search time, it also significantly increases power consumption. In this paper, we propose a gate-block selection algorithm, which can synthesize a proper parameter extractor of the pre-computation-based CAM (PB-CAM) to improve the efficiency for specific applications such as embedded systems. Through experimental results, we found that our approach effectively reduces the number of comparison operations for specific data types (ranging from 19.24% to 27.42%) compared with the 1's count approach. We used Synopsys Nanosim to estimate the power consumption in TSMC 0.35um CMOS process. Compared to the 1's count PB-CAM, our proposed PB-CAM achieves 17.72% to 21.09% in power reduction for specific data types.

I. Introduction

Content-addressable memory (CAM) is a storage device, which provides an efficiently fast data-search function. To achieve an effective function of data searching, the data comparison architecture of CAMs is usually implemented in parallel operation structure. However, because of the parallel processing characteristic for data comparison, power consumption is always an important concern when designing CAM circuitry.

In the past decade, many articles have been devoted to the study of CAMs for low-power, in which power reduction has focused on the circuit and architecture domains [1]. Several techniques on reducing power consumption at circuit level have focused on reducing the match-line (ML) power such as lowering the ML voltage swing [2], [3], reducing the switching capacitance on the MLs by using the NAND ML architecture [4], and reducing the switching activity of the ML [5]. Selective pre-charge is perhaps the most common method used to save power on MLs [6]–[9], since it is simple to implement. Although these techniques effectively save power, the power consumption of CAMs is still high compared with the conventional memories (DRAMs and SRAMs) of similar size. Meanwhile, research in CAM system designs for power efficiency at the architectural level continues to increase. For example, one architectural concept for saving power is to change the encoding of stored data in the CAM cell [10].

Recently, pre-computation technique has received as one of the most effective approaches for low-power designs. Pre-computation-Based CAM (PB-CAM) stores extra information along with data used in the data searching operation to eliminate most of the unnecessary comparison operations, thereby saving power. The pre-computation approach of [11] adopts the 1's count function to extract the extra information from input data. However, the major deficiency of [11] is it suffers from the number of comparison operations in the data comparison process for random input data resulting in the normal distribution characteristic. Unfortunately, this characteristic limits the reduction of comparison operations in PB-CAMs.

In this paper, we present a gate-block selection algorithm to synthesize a proper parameter extractor of the PB-CAM to reduce the number of comparison operations in the data comparison process for specific applications. The proposed approach better reduces the number of comparison operations required than [11]. In addition, our approach reduces not only power consumption but area and delay for the pre-computation circuit compared with [11].

II. Previous Work and Observation

A. PB-CAM Architecture

Fig. 1 shows the memory organization of the PB-CAM that is composed of a data memory, a parameter memory, and a parameter extractor, where $k \ll n$. During the writing phase, the parameter extractor extracts and composes the parameter based on the input data, and then stores it in the parameter memory. During the data searching operation, to reduce the vast number of comparison operations, the operation is divided into two comparison processes. In the first comparison process (parameter comparison), the parameter extractor extracts the parameter from the searching data, and then compares it with all parameters stored in the parameter memory in parallel. If the stored parameters mismatch the parameter of the

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

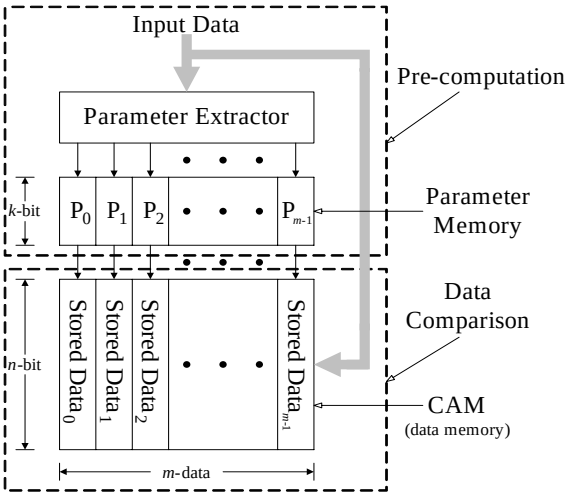

Fig. 1. Memory organization of the PB-CAM.

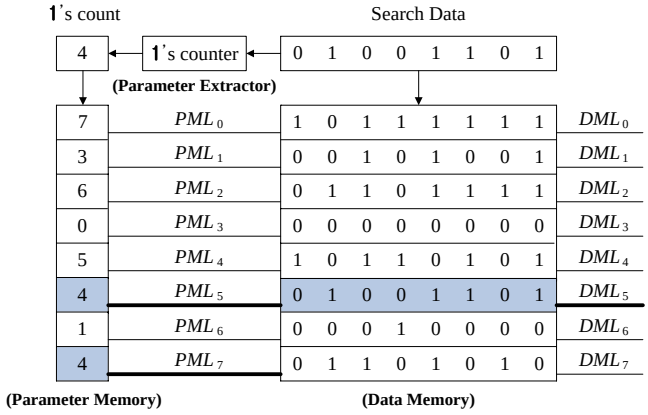

Fig. 2. Conceptual view of the 1's count PB-CAM.

searching data, it means the searching data does not exist in the data memory. Otherwise, the data related to the matching parameters must be compared in the second comparison process (data comparison). Although the first comparison process must compare the entire parameter memory, the size of the parameter memory is far smaller than that of the CAM (data memory). Moreover, since the first comparison process already filters out the unmatched data, the second comparison process only needs to compare the data from among those matches in the first comparison process. The PB-CAM adopts this concept to reduce the number of comparison operations, thereby saving power.

Obviously, the parameter extractor is critical in the PB-CAM, because it determines the number of comparison operations in the second comparison process. Therefore, the design goal of the parameter extractor is to filter out as many unmatched data as possible to reduce the required number of comparison operations in the second comparison process.

B. 1's count PB-CAM

Fig. 2 illustrates an example of the searching operation in the 1's count PB-CAM. Suppose that the searching data is 01001101_2, the parameter extractor (1's counter) counts the number of ones (which is four in this case), and then the number four is compared with all parameter (i.e. number of ones) stored in the parameter memory. As can be seen, only PML_5 and PML_7 match, since only they have a 1's count of four. As a result, only two comparison operations are performed and consumed power in the data memory. Finally, only DML_5 results in a match. Although the 1's count PB-CAM can reduce the number of comparison operations in some cases, it fails to reduce the number of comparison operations for simple random input data.

Assume that the inputs are independent and uniformly distributed, the number of input data related to the same parameter for n-bit input data can be determined by

$$N_{s.p.} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \quad , \tag{1}$$

where k is a number of ones for n-bit input data (which is from 0 to n). Therefore, the average probability of every parameter occurring can be derived by

$$P_{avg} = \frac{N_{s.p.}}{2^n} \quad . \tag{2}$$

The 1's count columns in Table I list the number of data related to the same parameter and their average probabilities for random input data that is 15-bit in length. Obviously, the 1's count approach demonstrates the normal distribution characteristic. Although the number of comparison operations required for the 1's count PB-CAM is fewer than that of conventional CAMs, the 1's count PB-CAM fails to reduce the number of comparison operations in the second comparison process when the parameters ranging between 5 and 10. Notice that when the bit length of input data increases, the failed range of the parameters will increase as well. Simply stated, the normal distribution limits further the reduction of comparison operations in PB-CAMs so that the effect of reducing power consumption is also limited. In addition, for the 1's count approach, the parameter extractor is implemented with many full adders, which not only wastes area but increases delay.

C. Block-XOR PB-CAM

To improve the deficiencies of the 1's count approach, the concept behind the block-xor approach is to eliminate the normal distribution characteristic for random input data resulting from the 1's count approach and to reduce the delay and area of the parameter extractor [12]. They use several XOR gates instead of the 1's counter as a parameter extractor to generate the parameter as shown in Fig. 3. For fair comparison, the same bit length of random input data and parameter are set to 15 and 4, respectively. Certainly, the parameter extractor of the block-xor PB-CAM is better than that of the 1's count PB-CAM in terms of the area, power, and speed. Moreover, according to the operation characteristic of XOR gate, the Block-XOR columns in Table I list the number of data related to the same parameter and their average probabilities for random input data that is 15-bit in length. Obviously, the block-xor approach shows the uniform distribution characteristic for the generated parameter by the block-xor parameter extractor. As can be seen from Table I, although the parameter extractor of the block-xor approach is better than that of the 1's count approach only for

TABLE I
THE NUMBER OF DATA RELATED TO THE SAME PARAMETER AND AVERAGE PROBABILITY FOR THE 1'S COUNT AND THE BLOCK-XOR APPROACHES

Parameter		Number of data related to the parameter		Average probability	
		1's count	Block-XOR	1's count	Block-XOR
0000	0	1	2048	0.003%	6.25%
0001	1	15	2048	0.046%	6.25%
0010	2	105	2048	0.320%	6.25%
0011	3	455	2048	1.389%	6.25%
0100	4	1365	2048	4.166%	6.25%
0101	5	3003	2048	9.164%	6.25%
0110	6	5005	2048	15.274%	6.25%
0111	7	6435	2048	19.638%	6.25%
1000	8	6435	2048	19.638%	6.25%
1001	9	5005	2048	15.274%	6.25%
1010	10	3003	2048	9.164%	6.25%
1011	11	1365	2048	4.166%	6.25%
1100	12	455	2048	1.389%	6.25%
1101	13	105	2048	0.320%	6.25%
1110	14	15	2048	0.046%	6.25%
1111	15	1	2048	0.003%	6.25%

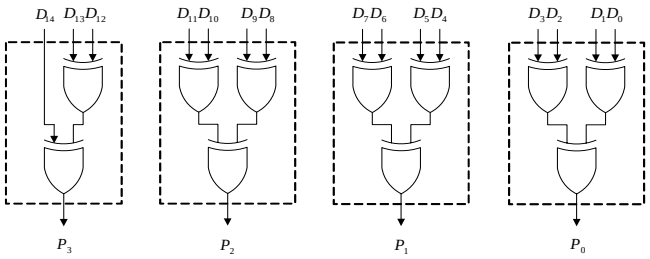

Fig. 3. 15-bit parameter extractor of the block-xor PB-CAM.

parameters ranging between 5 and 10, we must draw attention to the fact that the total probability of these parameters occurs is 88%. The probability viewpoint prove that the parameter extractor of the block-xor approach reduce the number of comparison operations in 88% of the cases for 15-bit random input data compared with that of the 1's count approach. Notice that when the bit length of random input data is increased, the total amount of probability is increased as well, this is because the 1's count approach results in the normal distribution characteristic.

However, for most of applications, their data distribution characteristic are specific rather than random.

III. PROPOSED APPROACH

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, 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 $\lceil n/l \rceil$, where n is the bit length of the input data, and then the levels in each partition block equal $\lceil log_2 l \rceil$. 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

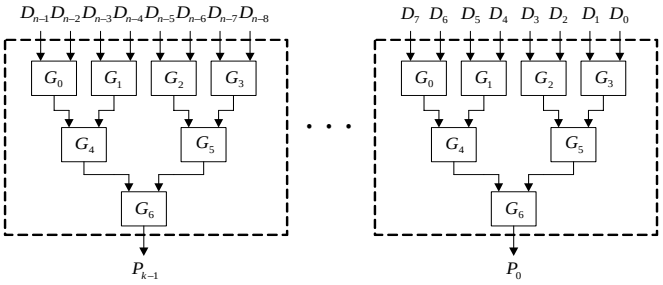

Fig. 4. n-bit block diagram of the proposed parameter extractor architecture.

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 in Section II-A, when the PB-CAM performs data searching operation, it must compare the entire parameter memory. To avoid wasting the large mount of power in the parameter memory, we set the input of each partition block to 8 bits. Fig. 4 shows the proposed parameter extractor architecture. We first partition the input data bit into several blocks, $G_0 \sim G_6$ 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. 4 so that the parameter $(P_{k-1}, P_{k-2}, \cdots, P_0)$ can reduce the number of data comparison operations as many as possible.

In our proposed parameter extractor, the bit length of the parameter is set into $\lceil n/8 \rceil$, and then the levels in each partition block equal $\lceil log_2 8 \rceil$ (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 $(6^7)^{\lceil n/8 \rceil}$ different logic combinations based on the proposed parameter extractor. Obviously, the optimal combination of the parameter extractor can not be found in polynomial time.

A. Gate-block Selection Algorithm

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.

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

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

of comparison operations in each data search operation can be formulated as

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

where N_0 is the number of $zero$ entries, and N_1 is the number of one entries for the generated parameters. To illustrate clearly, we use Table II as an example. 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:

$$C_{avg} = \frac{5^2 + 1^2}{5+1} = 4.33 \tag{5}$$

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 II 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. 5 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.

Algorithm to select proper logic gates for specific data :

Input data= $(D_0, D_1, \cdots, D_{n-1})$

n: bit length of the input data,
l: number of input bits for each partition block.

Step 1 : *Record*
NAND_parameter(k)=$\overline{D_{2i} \cdot D_{2i+1}}$
NOR_parameter(k)=$\overline{D_{2i} + D_{2i+1}}$
XOR_parameter(k)=$D_{2i} \oplus D_{2i+1}$
for i, k=0, 1, $\cdots$, $(n/2)$-1, $\forall$ *input patterns*
Step 2 : *Compute*
NAND_*C_{avg}(k)*
NOR_*C_{avg}(k)*
XOR_*C_{avg}(k)*
using Equ. 4, $\forall$ *k*
Step 3 : *Select a logic gate with the minimal* C_{avg}*(k),* $\forall$ *k*
Step 4 : ***If*** *generated parameter bits* $> \lceil n/l \rceil$,
repeat Step 1 to Step 3,
and use previous generated parameter as input data.
else
finish.

Fig. 5. Gate-Block Selection Algorithm.

B. An Example

To better understand our proposed approach, we give a simple example as illustrated in Fig. 6. 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 $\lceil log_2 4 \rceil$ (i.e. two levels). First, we use different logic gates (NAND, NOR, and XOR) to generate the parameter for D_1D_0, respectively, and then records their generated parameter for each pattern as shown in Fig. 6 (a). After that, according to Equ. 4, we calculate the average number of comparison operations C_{avg} for each logic gate. Obviously, using NAND gate is the best selection for D_1D_0, hence NAND gate is selected as a part of the parameter extractor. Similarly, NOR gate is selected to generate the parameter for D_3D_2 (see Fig. 6 (b)). Now, the parameter bits is 2, which is greater than $\lceil 4/4 \rceil$ (expected parameter bits). According to the proposed algorithm, we repeat **Step 1** to **Step 3** to determine the parameter bits for the next level. For Y_1Y_0, as shown in Fig.6 (c), XOR gate is the best parameter extractor for Y_1Y_0. Through the algorithm, the generated parameter is only 1 bit, which is no longer greater than $\lceil 4/4 \rceil$, hence the procedure of synthesizing parameter extractor is done. Finally, Fig. 6 (d) shows the synthesized parameter extractor for the input data.

IV. EXPERIMENTAL RESULTS

To prove the proposed pre-computation approach, we simulated some benchmarks (Mibench [13]) for the 1's count, block-xor, and our proposed pre-computation approaches. For fair comparison, the same bit length of the parameter of the block-xor approach is set as our proposed approach (which is 4, i.e. the input of each partition is 8 bits). In our experiment, the memory capacity is 128 words by 32-bit. Table III presents the number of comparison operations required for different pre-computation approaches. Obviously, our proposed pre-computation approach effectively reduces the number of

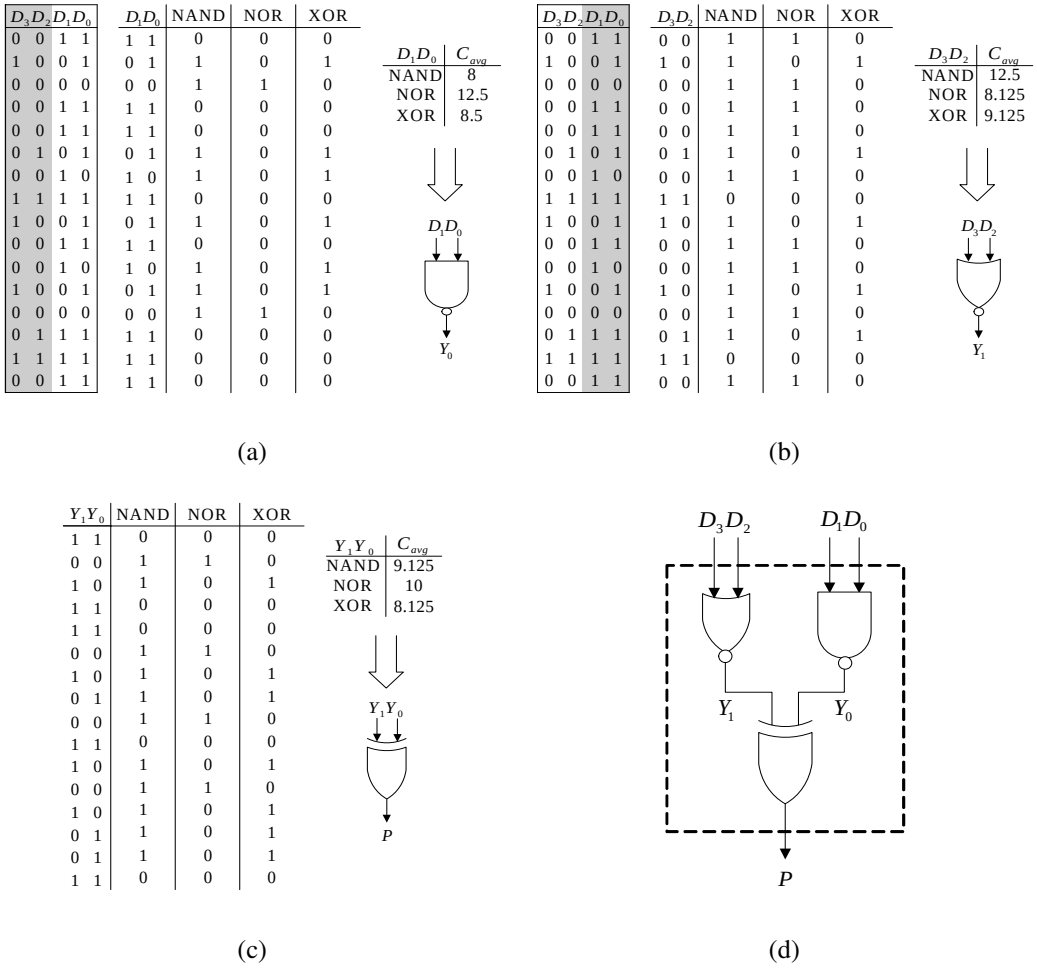

Fig. 6. An example for synthesis of the parameter extractor.

TABLE III
NUMBER OF COMPARISON OPERATIONS FOR DIFFERENT PRE-COMPUTATION APPROACHES

Mibench (32 bits)	No. of Comparison Operations			Reduction Rate (%)	
	1's Count	Block-XOR	**Proposed**	Block-XOR	**Proposed**
bitcount	422380	346132	341096	18.05	19.24
blowfish	435004	360773	337670	17.06	22.38
crc	405983	331605	308350	18.32	24.05
dijstra	426475	344112	319306	19.31	25.13
fft	468361	362362	343951	22.63	26.56
ispell	775538	689230	613811	11.13	20.85
patricia	433285	357407	314480	17.51	27.42
quicksort	416439	325130	306307	21.93	26.45
rijndael	448336	374075	359401	16.56	19.84
susan	639221	515755	488750	19.32	23.54
Average Reduction Rate				18.18	23.55

comparison operations (i.e. filter out most unmatched data). In detail, compared to the 1's count approach, our proposed approach reduce 19.24% to 27.42% of comparison operations for Mibench that is better than the block-xor approach (ranging from 11.13% to 22.63%).

To further verify the low-power performance of the proposed PB-CAM, we implemented, simulated, and compared the power consumption of the 1's count, block-xor, and our proposed PB-CAMs. To make the comparison fair and accurate, all PB-CAMs were described in Spice in TSMC 0.35μm double-poly quadruple-metal CMOS process without using any special transistors such as low-V_{th} transistors. Table IV lists the implemented configuration of three PB-CAMs. We used Synopsys Nanosim to simulate the power consumption for these PB-CAMs. The simulation results are presented in Table V. The results show that our proposed approach achieving 17.72% to 21.09% of power reduction for Mibench compared with the 1's count PB-CAM, that is also batter than the block-xor PB-CAM (14.05% to 18.27%). As we expected, our proposed PB-CAM effectively reduces power consumption by reducing the number of comparison operations in the data comparison process. The experimental results also confirmed that our proposed approach is more suitable for specific applications compared with the 1's count and block-xor approaches.

In addition, Table VI compares the critical path, area, power, and generated parameter bits of the 1's count and block-xor PB-CAMs with those of our proposed PB-CAM. The 'FA' and 'LG' in Table VI represent full adder and logic gate (which is NAND, NOR, or XOR), respectively. As shown in Table VI, all features of our proposed PB-CAM are superior to those of the 1's count and block-xor PB-CAMs. Notice that our pro-

TABLE IV
IMPLEMENTATION CONFIGURATION

	All PB-CAMs
Technology	TSMC 0.35μm, 2P4M, 3.3 V
Configuration	128 $\times$ 32
Word structure	Static [11]
Match Line (CAM)	NOR type
CAM cell	Traditional Design

TABLE V
POWER CONSUMPTION OF DIFFERENT PB-CAMS

Mibench	Average Power (mW)			Reduction Rate (%)	
(32 bits)	1's Count	Block-XOR	**Proposed**	Block-XOR	**Proposed**
bitcount	74.88	61.85	60.70	17.40	18.93
blowfish	73.71	61.57	59.56	16.46	19.19
crc	74.33	61.55	59.24	17.19	20.29
dijstra	73.87	60.98	59.14	17.44	19.93
fft	74.58	60.95	59.50	18.27	20.22
ispell	73.77	63.40	60.70	14.05	17.72
patricia	74.54	61.75	59.37	17.16	20.35
quicksort	74.60	61.02	58.87	18.20	21.09
rijndael	73.65	61.38	60.25	16.66	18.19
susan	75.63	62.77	61.41	17.00	18.80
Average Reduction Rate				16.98	19.47

TABLE VI
COMPARISON OF DIFFERENT PARAMETER EXTRACTORS FOR MIBENCH

	1's Count	Block-XOR	**Proposed**
Critical Path	FA$\times$8+OR$\times$1	XOR$\times$3	LG$\times$3
Area	FA$\times$41+OR$\times$1	XOR$\times$28	LG$\times$28
Average Power	6.58 mW	1.02 mW	0.67 mW
Parameter Bits	6	4	4

posed PB-CAM computes the parameter for all input data in only three logic gate delays (i.e. constant delay of search operation). Moreover, compared to the generated parameter bits of the 1's count approach (which is 6), our proposed approach is reduced to 4. In other words, it means our proposed approach saves 33.3% of the size of the parameter memory for 32-bit data compared with the 1's count approach.

V. CONCLUSION

In this paper, 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. The experimental 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). As shown in our experimental results, our proposed PB-CAM is very suitable for specific applications such as embedded systems.

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, pp. 712–727, Mar. 2006.

[2] 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, Jun. 2001.

[3] 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.

[4] F. Shafai, K. J. Schultz, G. F. R. Gibson, A. G. Bluschke, and D. E. Somppi, "Fully parallel 30-mhz, 2.5-mb cam," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1690–1696, Nov. 1998.

[5] I. Y.-L. Hsiao, D.-H. Wang, and C.-W. Jen, "Power modeling and low-power design of content addressable memories," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 4, May 2001, pp. 926–929.

[6] A. Efthymiou and J. D. Garside, "A cam with mixed serial-parallel comparison for use in low energy caches," *IEEE Trans. Very Large Scale Integration Systems*, vol. 12, pp. 325–329, Mar. 2004.

[7] N. Mohan and M. Sachdev, "Low power dual matchline ternary content addressable memory," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, May 2004, pp. 633–636.

[8] K.-H. Cheng, C.-H. Wei, and S.-Y. Jiang, "Static divided word matching line for low-power content addressable memory design," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, May 2004, pp. 629–632.

[9] A. Roth, D. Foss, R. McKenzie, and D. Perry, "Advanced ternary cam circuits on 0.13 /spl mu/m logic process technology," in *Proc. IEEE Custom Integrated Circuits Conf.*, Oct. 2004, pp. 465–468.

[10] S. Hanzawa, T. Sakata, K. Kajigaya, R. Takemura, and T. Kawahara, "A dynamic cam - based on a one-hot-spot block code - for millions-entry lookup," in *Symp. VLSI Circuits Dig. Technical Papers*, Jun. 2004, pp. 382–385.

[11] C.-S. Lin, J.-C. Chang, and B.-D. Liu, "A low-power precomputation-based fully parallel content-addressable memory," *IEEE J. Solid-State Circuits*, vol. 38, pp. 654–662, Apr. 2003.

[12] S.-J. Ruan, C.-Y. Wu, and J.-Y. Hsieh, "Low power design of precomputation-based content addressable memory," *IEEE Trans. Very Large Scale Integration Systems*, (accepted).

[13] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshop on Workload Characterization (WWC)*, Dec. 2001, pp. 3–14.