

國立臺灣大學電機資訊學院電機電信電子產業研發碩士專班

碩士論文

Industrial Technology R&D Master Program in Electrical, Communication
and Electronics Engineering

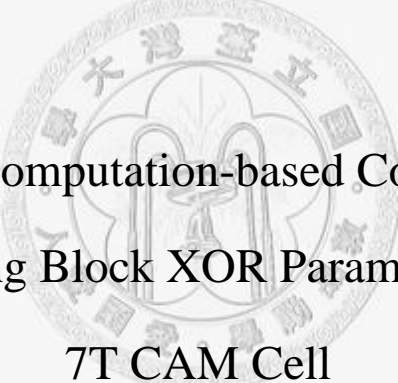
College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以預先計算為基礎並結合互斥或參數分離器及七個
電晶體記憶細胞元的低功率內容可定址記憶體

A Low Power Precomputation-based Content-Addressable
Memory Combining Block XOR Parameter Extractor with
7T CAM Cell



陳 豪

Hao Chen

指導教授：賴飛熊 博士

中華民國 九十七 年 六 月

June 2008

博碩士論文授權書

本授權書所授權之論文為授權人在國立台灣大學取得博碩士學位之論文

論文題目： 以預先計算為基礎並結合互斥或參數分離器及七個電晶體
記憶細胞元的低功率內容可定址記憶體

指導教授： 賴 飛 羆 博士

授權人於自由意志下茲同意將上列論文所擁有著作權部份之全文（含摘要、圖表、數據和內容之全部），非專屬、無償授權于上列本人之論文指導教授，不限地域、時間與次數，以印刷或數位方式將上列論文之一部或全部重製，指導教授得在其著作引用或複製本論文之文字或內容。授權人亦於自由意志下同意指導教授為本論文研究成果之共同創作人及著作人。

授權人： 陳 豪

簽名：

中華民國 97 年 07 月 18 日

國立臺灣大學碩士學位論文
口試委員會審定書

以預先計算為基礎並結合互斥或參數分離器及七個
電晶體記憶細胞元的低功率內容可定址記憶體

A Low Power Precomputation-based Content-Addressable
Memory Combining Block XOR Parameter Extractor with
7T CAM Cell

本論文係陳豪君 (J95921005) 在國立臺灣大學電機電信電子產業
研發碩士專班完成之碩士學位論文，於民國九十七年六月十四日承下
列考試委員審查通過及口試及格，特此證明

口試委員：

賴市雄

(簽名)

(指導教授)

張延臣

李昶瑋

蔡坤霖

張孟洲

系主任：

胡振國

(簽名)

致謝

隨著碩士論文的完成，學生生涯終於要進入尾聲。古有孟母三遷，母親大人二十多年來含辛茹苦，全力支持兄長與我，讓我們在無後顧之憂的環境下完成學業，您的恩情比起孟母只有過之而無不及啊！身為您的兒子我感到無比驕傲，謝謝您！

兩年前，我卸下迷彩服，抱著戒慎恐懼的心情回到台灣大學，卻在開學前後碰上人生的第一個低潮，感謝我的恩師 賴飛羆教授在這個時候拉了我一把，並且在往後的兩年之中不斷地教導我、指引我走向正確的研究方向，謝謝您，老師！

坤霖學長，感謝您時常回實驗室關心我們的研究狀況，並且不時給予我們指導，尤其是對我的論文提出的建議，這本論文的完成您無疑是一個重要的推手。謝謝國軒學長，您認真研究的態度讓我十分欽佩。在口試前的這段期間感謝您的經驗傳承，讓我們順利通過口試。同時也感謝您在這兩年來的照顧。治弘學長，感謝您在研究上的協助，指導我們如何做正確的研究，讓我們寫論文事半功倍。355 實驗室的精神領袖群哥，謝謝您給我们在各方面的指導以及生活上的協助。

歐陽、晏彰、振輝和來自德國的 Sebastian Ang 在這兩年間，無論在課業上或者是研究方面都給了我許多協助的與鼓勵，沒有你們我沒有辦法完成這兩年的學業，更遑論碩士論文了，謝謝你們！。也感謝其他 low power 小組的同學及學弟。

品琇，這兩年有妳的支持、妳的鼓勵，在我徬徨無助的時候聽我發牢騷，在我高興的時候聽我自吹自擂，讓我在研究的路上從來沒有失去過自信。妳的獨立自主也讓我在必要的時候能心無旁騖地專注於學業及研究，謝謝妳！

摘要

內容可定址記憶體常見於需要高速搜尋比對的應用當中，如今最主要的商業應用就是網路路由器了。它平行比對的機制雖具有高速的特性，卻造成了相當高的功率消耗。

過去曾有兩種以預先計算為基礎的內容可定址記憶體，一種是以區塊互斥或來取得參數，而另一種是以數一來取得參數。兩種架構功率消耗的降低方面都有很好的表現。數一的方法雖然有對應於同一參數的資料數量分佈不均勻的問題，但它獨有的7個電晶體的儲存元不但省電還能節省晶片面積。反之，區塊互斥或的架構雖然解決了參數分佈不均勻的問題，卻無法使用7個電晶體的儲存元。

本論文提出了一個新的方法，透過一個額外的編碼器，結合了以上兩者的優點。我們使用 Synopsys 公司的 HSIPCE 搭配 0.18 μm 的製程檔進行模擬。

根據模擬結果，相較於區塊互斥或的方法，本論文提出的方法增加了12.5%的搜尋延遲，但減低了平均功率消耗達23.6%，並且降低了功率-延遲乘積達14%，而在面積的使用上並無增加。

關鍵字：低功率、內容可定址記憶體、預先計算、編碼

A Low Power Precomputation-based Content-Addressable
Memory Combining Block XOR Parameter Extractor with
7T CAM Cell

By

Hao Chen

Master Thesis

Industrial Technology R&D Master Program in Electrical, Communication
and Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Advisor

Prof. Feipei Lai

June 2008

Abstract

Content-addressable memory is widely used in the application requiring high search speed. The primary commercial application of CAMs today is Internet routers. Although it has the high speed characteristic due to the parallel comparison, it results in quite high power consumption.

There have been two precomputation-based CAMs proposed previously. One is to extract parameters by means of counting the number of 1's within input data. Another is to extract parameters by the so called Block XOR method. Both of the two architectures have excellent performance in reducing power consumption. Though the 1's count method has a problem that the number of data related to a single parameter is not uniform distributed, its unique 7T CAM cells not only reduce power consumption but also save chip area. Whereas the Block XOR method solved the problem of data distribution, it can not apply the 7T CAM cells.

In this thesis, we proposed a novel method. Through an extra input encoder, we combined the advantage of both architectures mentioned above. We further implemented and simulated the whole architecture by Synopsys Hspice using 0.18 μm technology.

Simulation results show that comparing to the Block XOR method, our method has 12.5% longer latency, but the reduction of average power and power-delay product reached 23.6% and 14% respectively. As for the area cost, it remains almost equal.

Keywords: Low power, Content-Addressable Memory, Precomputation, Encode

Contents

Abstract

Contents.....I

List of Figures.....IV

List of Tables.....VI

Chapter 1 Introduction 1

1.1 Low Power Requirement..... 1

1.2 Power Dissipation in CMOS VLSI Circuit 2

1.2.2 Short-circuit Power..... 3

1.2.3 Leakage Power 4

1.2.4 Static Power..... 5

1.3 Content-Addressable Memory..... 6

1.3.1 CAM Overview 6

1.3.2 CAM Applications..... 7

1.4 Thesis Organization..... 9

Chapter 2 CAM Basics and Related Works 10

2.1 CAM Cells..... 10

2.1.1 WRITE Operation of a CAM Cell..... 12

2.1.2. READ Operation of a CAM Cell 12

2.2 Search Mechanism..... 13

2.3 Matchlines 15

| | |
|---|-----------|
| 2.3.1 NOR Matchline | 16 |
| 2.3.2 NAND Matchline | 17 |
| 2.3.3 Comparison..... | 18 |
| 2.4 CAM Relative Research | 18 |
| Chapter 3 Encoded PB-CAM | 21 |
| 3.1 Preliminary Work on PB-CAM..... | 21 |
| 3.1.1 1's Count | 22 |
| 3.1.2 Block XOR | 27 |
| 3.2 Encoded PB-CAM..... | 31 |
| 3.2.1 XOR Parameter Extractor..... | 31 |
| 3.2.2 Matchline Selection..... | 32 |
| 3.2.3 Encoding Scheme..... | 34 |
| 3.2.4 An Example of Proposed Encoded PB-CAM Operation | 35 |
| 3.3 Encoded PB-CAM Benefits and Constraints | 37 |
| 3.3.1 Encoded PB-CAM Benefits | 37 |
| 3.3.2 Encoded PB-CAM Constraints..... | 38 |
| Chapter 4 SPICE Implementation | 39 |
| 4.1 Specifications | 40 |
| 4.2 Encoded PB-CAM Implementation | 40 |
| 4.2.1 Parameter Extractor | 41 |
| 4.2.2 Valid Bit Column | 41 |
| 4.2.3 Parameter Storage Block | 42 |
| 4.2.4 Bitline Buffer Implementation | 43 |
| 4.2.5 Data Storage Block..... | 44 |
| 4.2.6 Matchline Buffer..... | 44 |
| 4.2.7 Input Encoder | 44 |

Chapter 5 Simulation Results 46

 5.1 Power Consumption 46

 5.2 Latency 48

 5.3 Area 49

Chapter 6 Conclusion..... 50

Bibliography..... 52



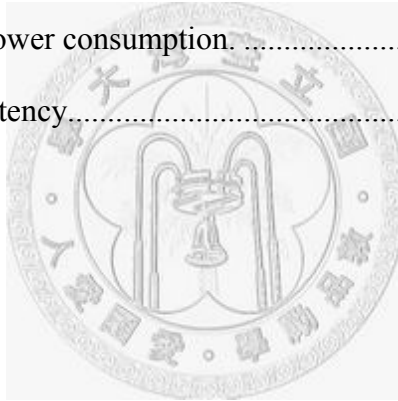
List of Figures

| | |
|---|----|
| Figure 1.1: CMOS inverter..... | 3 |
| Figure 1.2: A simplified CAM architecture..... | 6 |
| Figure 1.3: CAM-based example of the routing table of Table 1.1 | 8 |
| Figure 2.1: A 9T CAM cell | 11 |
| Figure 2.2: WRITE operation of a 9T CAM cell | 11 |
| Figure 2.3: READ operation of a 9T CAM cell | 13 |
| Figure 2.4: Schematic of a CAM that stores four words, each consisting of 4 bits | 14 |
| Figure 2.5: (a) XOR type CAM cell (b) XNOR type CAM cell | 15 |
| Figure 2.6: Schematic of the NOR matchline structure [3]..... | 16 |
| Figure 2.7: Schematic of the NAND matchline structure [3]..... | 17 |
| Figure 2.8: A concept view of precomputation | 19 |
| Figure 3.1: Conceptual view of the 1's count precomputation method | 22 |
| Figure 3.2: Distribution of input data among all parameters for the 1's count of a 14 bit word..... | 24 |
| Figure 3.3: 14-bit 1's count parameter extraction circuit of [4] | 24 |
| Figure 3.4: A 7T CAM cell | 25 |
| Figure 3.5: Conceptual view of word structure composed of 7T cells..... | 26 |
| Figure 3.6: N bits Block XOR circuitry of [5] | 28 |
| Figure 3.7: The error condition occurs when applying 7T CAM cells in Block XOR method | 30 |
| Figure 3.8: Proposed Encoded PB-CAM | 31 |

| | |
|---|----|
| Figure 3.9: Static parameter comparison circuit in [4]..... | 32 |
| Figure 3.10: Matchline structure with parameter comparison circuit | 33 |
| Figure 3.11: An example of parameter extraction and data encoding..... | 36 |
| Figure 3.12: A conceptual view of suggestion when applying PB-CAM to a larger system | 38 |
| Figure 4.1: A simplified block diagram of proposed CAM architecture..... | 40 |
| Figure 4.2: Conceptual view of Parameter Extractor | 41 |
| Figure 4.3: An example XOR gates inside the Parameter Extractor | 41 |
| Figure 4.4: Configuration of valid bit of a data word..... | 42 |
| Figure 4.5 (a) and (b) Bitline buffer for both architectures | 43 |
| Figure 4.6: Input encoder | 44 |
| Figure 4.7: Input encoder synthesized by Synopsys Design Compiler | 45 |
| Figure 5.1: Power-performance comparison of Hspice 0.18 μm technology simulation | 47 |
| Figure 5.2: Comparison of delay and power-delay product..... | 48 |
| Figure 5.3: Number of MOSFETs used in both models..... | 49 |

List of Tables

| | |
|--|----|
| Table 1.1: An example of a routing table | 7 |
| Table 3.1: Average probability and number of data related to the 1's count parameters of a 14 bit word..... | 23 |
| Table 3.1: Truth table of 9T cell and 7T cell..... | 26 |
| Table 3.2: All 16 4-bit combinations and their resulting Block XOR parameter..... | 29 |
| Table 3.2: Truth table of proposed encoding scheme..... | 34 |
| Table 5.1: Comparison of power consumption..... | 47 |
| Table 5.2: Comparison of latency..... | 48 |



Chapter 1

Introduction

Low power VLSI design has been a topic of interest in the last decade. The subject has been very much left isolated since the invention of integrated circuits. With the Moore's Law showing no signs of deceleration, the power consumption of VLSI chip is expected to rise beyond the current status, so power consumption of a chip greatly emerges as a popular topic in recent years. Furthermore, power consumption is also one of the major issues in the System-on-Chip (SoC) system design.

In this thesis, we proposed a low power content-addressable memory (CAM) combining two previously proposed architectures [4] [5], which applied precomputation method. We built SPICE models of the proposed architecture as well as reference models of so called Block XOR method and 1's count method respectively, and figured out power consumption, latency and area of the three models by Synopsys Hspice.

1.1 Low Power Requirement

In recent years, as the technology scales down to the nanometer dimensions, low power

technique has become a critical design constraint for many electronic computing and communications systems. The portable consumer electronics (e.g. PDA, cellular phones and laptop computers) market is constantly demanding more powerful capabilities, smaller and lighter products, and longer battery life. For this kind of devices, low power is a key topic of the design requirement. Even in high-end computer systems, low power is still a critical design consideration, since the expensive cooling and packaging costs and lower reliability associated with high on-chip power dissipation are significant.

1.2 Power Dissipation in CMOS VLSI Circuit

In CMOS VLSI, source of power consumption can be summarized by the following equation:

$$P_{total} = P_{dynamic} + P_{short-circuit} + P_{leakage} + P_{static} \quad (1.1)$$

Where:

$P_{dynamic}$ means dynamic power dissipation.

$P_{short-circuit}$ means short-circuit power.

$P_{leakage}$ means the power consumption of leakage current.

P_{static} means the static power consumption.

Each of terms is explained in the following sections.

1.2.1 Dynamic Power

Dynamic power dissipation, also called switching power, is caused by charging and discharging parasitic capacitance of transistors and wires in the circuit. This is the main

source of power dissipation in CMOS VLSI circuit, accounting for about 75% of total power consumption. It can be described by the following equation:

$$P_{dynamic} = \alpha \times f \times C_{load} \times V_{dd}^2 \times p \quad (1.2)$$

where :

α = Constant (from 0 to 1)

f = Switching frequency of the circuit.

C_{load} = Sum of all parasitic capacitance. For example, in 0.18 μm technology, the typical value of C_{load} is about 0.005pF to 1pF.

V_{dd} = Supply voltage of the system.

p = Switching probability of signal.

1.2.2 Short-circuit Power

Short-circuit power ($P_{short-circuit}$) is caused by direct paths from V_{dd} to ground that is generated by transition activities of input signal values. Consider the CMOS inverter shown in Figure 1.1, when the input changes from 1 to 0, there is a tiny period when both NMOS and PMOS transistors are conducted, resulting in a short-circuit current drawn from the V_{dd} to ground.

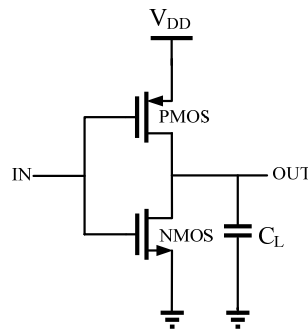


Figure 1.1: CMOS inverter.

Assuming symmetric rise/fall delay and threshold voltage for NMOS and PMOS, the short-circuit power dissipation of a CMOS inverter can be approximated by the following equation [1].

$$P_{short-circuit} = k \times (V_{dd} - 2V_T)^3 \times \tau \times N \times f_{clk} \quad (1.3)$$

where

k = a constant that depends on the transistor sizes and the technology.

V_T = magnitude of the threshold voltage of the NMOS and PMOS transistors.

τ = input rise/fall time.

N = average number of transitions at the output of inverter.

f_{clk} = clock frequency.

Short-circuit power dissipation can be controlled to a small portion of the total power by appropriately sizing transistors and reducing the input rise/fall times to all the gates in the circuit. Short-circuit power dissipation is also reduced by scaling the supply voltage or reducing the switching activity at the gate outputs.

1.2.3 Leakage Power

Leakage power consumption ($P_{leakage}$) can be further divided into two components shown in the following equation [2].

$$P_{leakage} = (I_{subthreshold} + I_{diode}) \times V_{dd} \quad (1.4)$$

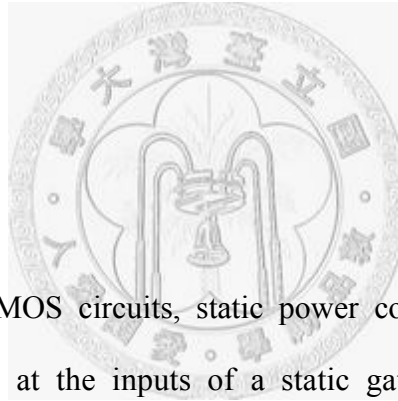
In the above equation, $I_{subthreshold}$ refers to the current arise due to the fact that transistors which are “off” still conduct some weak current. The expression of subthreshold current

in the NMOS transistor of a CMOS inverter with input voltage V_{in} varying between 0 and its threshold voltage V_{th} is shown as follows [2].

$$I_{subthreshold} = K \times W_{eff} \times e^{\frac{V_{in}-V_{th}}{s}} \quad (1.5)$$

where K and S are constants that depend on the technology and W_{eff} is the effective transistor channel width.

The term I_{diode} refers to the currents flowing through the reverse-biased diodes that are formed between the diffusion regions and the substrate. These currents are small for former process technologies, but they are no longer neglectable due to the popularity of submicron technology.



1.2.4 Static Power

In fully complementary CMOS circuits, static power consumption may result from degenerated voltage levels at the inputs of a static gate, or selector-conflicts and bus-conflicts where multiple drivers attempt to drive a signal to different logic values. Such situations are undesirable, and are typically avoided through proper circuit design techniques. Static power consumption is also important for logic families such as pseudo-NMOS, where a gate consists of a single pull-up PMOS transistor and an NMOS network, in which there is a constantly conducting supply-to-ground path.

1.3 Content-Addressable Memory

1.3.1 CAM Overview

Content-addressable memory (CAM) is a memory architecture that is often used to implement IP lookup table functions in a single clock cycle. Instead of returning data stored within the memory, a CAM compares input search data with its stored data in parallel. If the search is successful, which means that a stored word matches the search word, the CAM outputs the address of the matching data. Figure 1.2 shows the conceptual view of a content-addressable memory. Input to the CAM block is the search word. It is broadcast to all the data words within the table through the searchlines. The depicted CAM contains m words each consisting of n bits.

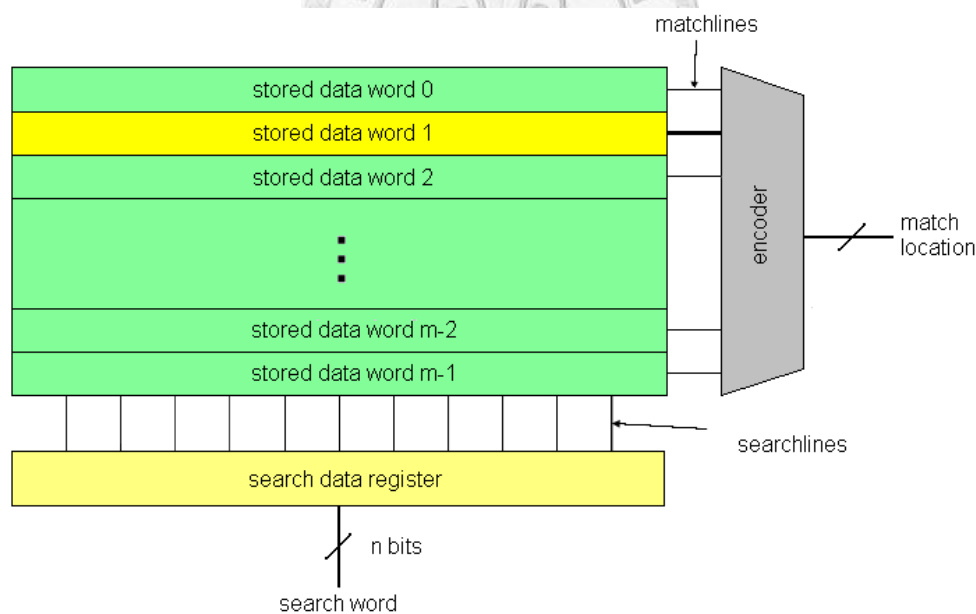


Figure 1.2: A simplified CAM architecture.

Each stored data word has its own matchline connected to an encoder. In this example the stored data word at memory position 2 matches the search data. In the case of match, the matchline of the corresponding address is in a high state whereas all the matchlines

of the mismatching data words are pulled to low. If the CAM is configured to allow multiple matches for a single search input, the encoder is replaced by a priority encoder, which outputs the address of the higher priority match. Data words at a lower address have a higher priority than the data words at higher addresses. As stated above, the CAM architecture is able to compare all its stored words with the search word and to output a match address in case of a match within a single clock cycle. With this single clock cycle throughput CAM is faster than other hardware- or software-based search architectures. This high speed table lookup is achieved by using dedicated comparison circuitry that is active in parallel and thus consume high power.

1.3.2 CAM Applications

CAM is widely used in a variety of applications that need high search speeds. However, the primary commercial application of CAM is the usage within network routers. Here we describe the application of CAM to packet forwarding in network routers. We briefly summarize packet forwarding and then show how a CAM implements the packet forwarding operations.

Table 1.1: An example of a routing table.

| Input address | Output port |
|---------------|-------------|
| 010010 | A |
| 100100 | B |
| 011011 | C |
| 101010 | D |

Using an address-lookup function, network routers forward data packets from an incoming port to an outgoing port. The address-lookup function examines the destination address of the packet and selects the output port according to that address. The router maintains a list, called the routing table, which contains destination addresses and their corresponding output ports. An example of a simplified routing table is shown in Table 1.1. All four entries in the table are 6-bit. The router searches this table for the destination address of each incoming packet, and selects the appropriate output port. For example, if the router receives a packet with the destination address 010010, the packet is forwarded to port A.

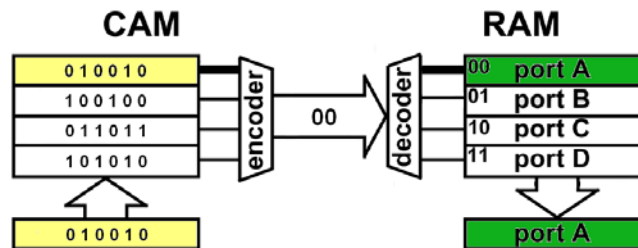


Figure 1.3: CAM-based example of the routing table of Table 1.1.

Figure 1.3 illustrates how a CAM accomplishes address lookup by implementing the routing table shown in Table 1.1. In the left half of figure, the packet destination address of 010010 is the input to the CAM. As in the table, the encoder chooses the upper entry and generates the match location 00, which corresponds to the most-direct route. This match location is the input address to a RAM that contains a list of output ports, as depicted in Figure 1.3. A read operation of a RAM outputs the port designation, port A, to which the incoming packet is forwarded. We can view the match location output of the CAM as a pointer that retrieves the associated word from the RAM. In the particular case of packet forwarding the associated word is the designation of the output port. This CAM/RAM system is a complete implementation of an address-lookup engine for packet forwarding [3].

1.4 Thesis Organization

The remainder of this work is organized as follows. Chapter 2 introduces the CAM basics to gain a better understanding of the CAM architecture. Chapter 3 firstly introduces two relative researches about the proposed CAM architecture and then describes how our proposed Encoded PB-CAM architecture operates in detail. Chapter 4 shows how we implemented the two reference CAM models and our proposed CAM model by SPICE code. In chapter 5, the power consumption, latency and area of the Encode PB-CAM architecture are thoroughly evaluated by Hspice simulation. In chapter 6 we summarize the result of this thesis.



Chapter 2

CAM Basics and Related Works

In chapter 2, we introduce the main building blocks that every content-addressable memory is composed of, and explain how the high-speed parallel comparison is implemented. After the building blocks have been described, a brief overview of different levels of design approaches of related research work will be presented. Because our proposed Encoded PB-CAM is a variation of [4] and [5], later we will put more emphasis on both of the two architectural methods in chapter 3.

2.1 CAM Cells

A schematic of a conventional 9T CAM cell is shown in Figure 2.1. The bit storage unit is a standard 6T SRAM cell and hence the READ as well as WRITE operation is identical to a 6T SRAM cell. The connection of drain and source of control transistor M_{ctl} is dependent on the matchline connection type, which decides whether the relative matchline discharge or not, so the explanation of search operation is left in later sections. In this section we merely describe READ and WRITE operations.

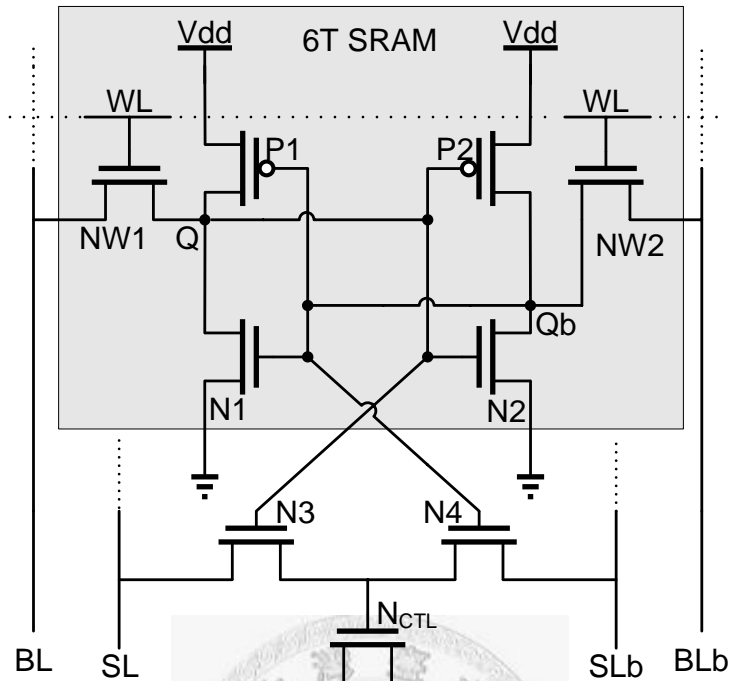


Figure 2.1: A 9T CAM cell.

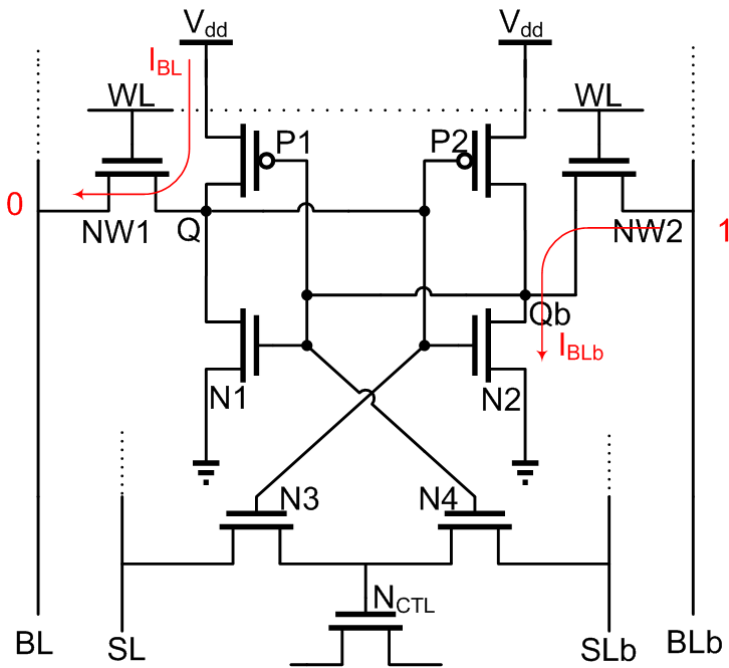


Figure 2.2: WRITE operation of a 9T CAM cell.

2.1.1 WRITE Operation of a CAM Cell

The WRITE operation is achieved by feeding complementary data to the bitline and bitlinebar (denoted as BL and BLb in the figure) and enabling the wordline (denoted as WL), which turns on the access transistors (NW1 and NW2). Since NW1 and NW2 are turned on, the internal nodes Q and Qb of cross-coupled inverters are written by the data on BL and BLb.

Figure 2.2 in previous page shows an example of writing logic “0” into a cell storing logic “1”. In the beginning, $V_Q = “V_{dd}”$ and $V_{Qb} = “GND”$, resulting in P2 and N1 turned off, P1 and N2 turned on. Once the WL enabled (WL rises to V_{dd}), access transistors (NW1 and NW2) conduct and result in currents I_{BL} and I_{BLb} , which form voltage dividers (P1-NW1 and NW2-N2). If these transient currents could flip one of the nodes (V_Q and V_{Qb}) to the threshold voltage of inverter, the other node is also soon flipped by the cross-coupled feedback path, so a careful sizing of transistors P1-P2 and N1-N4 is necessary to guarantee the correctness of WRITE operation.

2.1.2. READ Operation of a CAM Cell

The READ operation is achieved by firstly precharging both BL and BLb to V_{dd} and then enabling WL. Figure 2.3 in next page shows an example of READ operation where logic “0” is stored within the cell (i.e. $V_Q = “GND”$, $V_{Qb} = “V_{dd}”$). Since the BL drivers are turned off during the READ operation, current I_{READ} discharges BL through NW1 and N1. BLb remains at V_{dd} because voltage of node Q_b also equals V_{dd} . Therefore, there is a small voltage drop between BL and BLb, which is amplified to a rail-to-rail voltage by bitline sense amplifier (BSA). Because the BL are shared among all the cells in a column, the parasitic capacitance on BL are considerable.

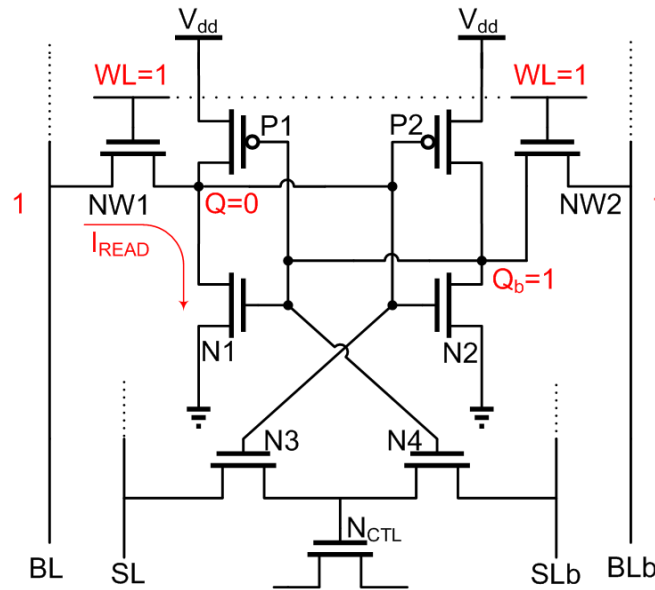


Figure 2.3: READ operation of a 9T CAM cell.

The small voltage swing in the BLs reduces power consumption and the access time during the READ operation. As shown in Figure 2.3, the current I_{READ} raises the voltage of V_Q . Thus, the driver transistors N1 and N2 are sized such that V_Q remains lower than threshold voltage of inverter, and hence the data within cell does not flip during READ operation. Typically, the driver transistors N1 and N2 are sized 1.5 to 2 times wider than the access transistors NW1 and NW2.

2.2 Search Mechanism

In this section, we introduce the internal search mechanism of a CAM cell. In order to simplify the schematic, we omit the NMOS access transistors and bitlines which are only used in READ and WRITE operation.

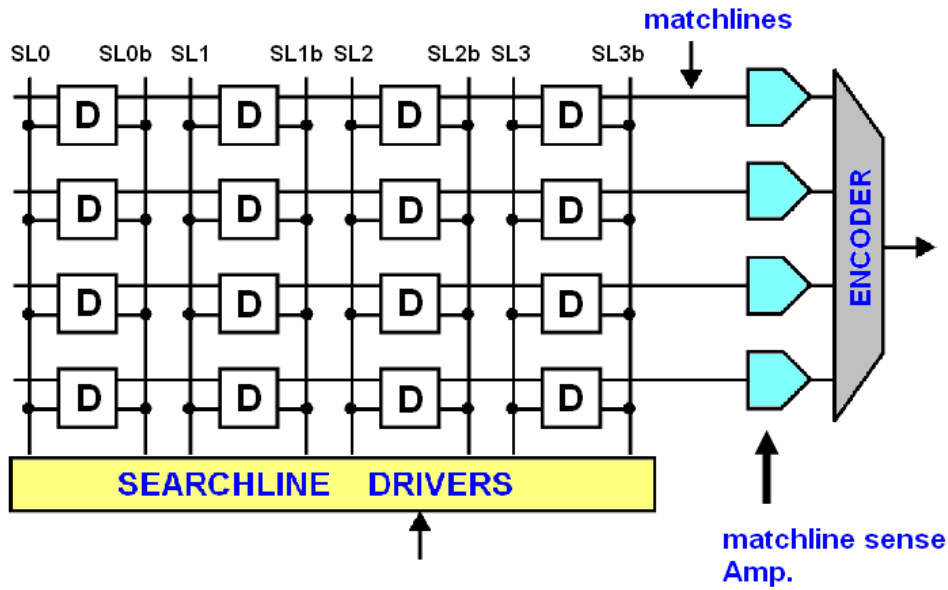


Figure 2.4: Schematic of a CAM that stores four words, each consisting of four bits

Figure 2.4 shows a schematic of a CAM that stores four data words. Each data word consists of four bits that are arranged horizontally and stored within CAM cells. CAM cells are the main building block of every CAM architecture and responsible for storing the data bits as well as comparing the stored bits with the search word. The four CAM cells that represent a data word are connected to the corresponding unique matchline. Between the matchline and decoders are sense amplifiers. Each bit of the search word is connected to a complementary pair of searchlines connecting to the CAM cells of the corresponding bit position.

Figure 2.5 shows both XOR type and XNOR type CAM cells consist of store and compare units. As mentioned in previous section, the store unit is usually implemented as the traditional 6T SRAM cell, and the compare unit needs two NMOS transistors to perform the comparison between the stored and search data. Besides the store and compare units, a pull-down transistor M_{CTL} , which is controlled by the comparison unit, is necessary to connect or disconnect the matchline (ML) to the ground. Consider the

XOR cell shown in Figure 2.5(a), when logic “1” is stored in Q (logic “0” stored in Qb) and a logic “1” is fed to SL (logic “0” is fed to SLb), a logic “0” will be transferred from SLb to the gate of pull-down transistor M_{CTL} , cutting off M_{CTL} and keeping ML floating. Whereas in Figure 2.5(b), when logic “1” is stored in Q (logic “0” stored in Qb) ,and a logic “1” is fed to SL (logic “0” is fed to SLb), a logic “1” will be transferred from SLb to the gate of pull-down transistor M_{CTL} , turning on M_{CTL} and connecting ML_n to ML_{n+1} . Consequently, in a XOR cell, if searched data on SL is identical to the data stored in Q, then ML will be kept HIGH; in a XNOR cell, if searched data on SL is identical to the data stored in Q, then ML_n connected to ML_{n+1} .

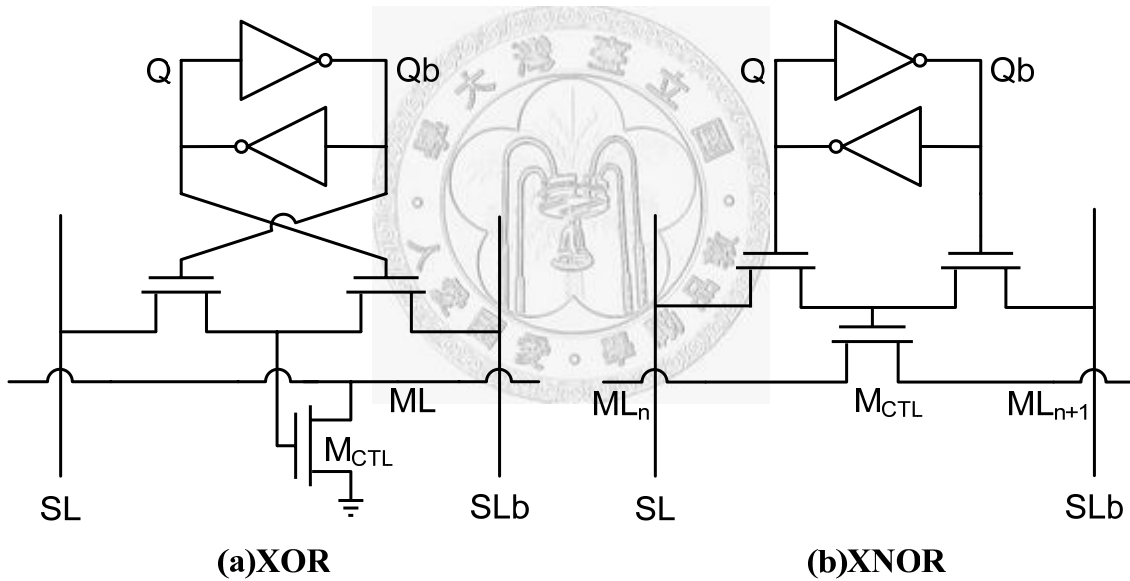


Figure 2.5: (a) XOR type CAM cell. (b) XNOR type CAM cell.

2.3 Matchlines

The matchline is the central structure of a CAM. Not only does it comprise the CAM cells, it also evaluates the voltage with the matchline sense amplifiers (MLSA) and

generates a corresponding output result. There are two different kinds of matchline structures which are the NOR matchline and the NAND matchline. The obvious difference is that the NOR matchline uses XOR type CAM cells as opposed to the NAND matchline that uses XNOR type CAM cells. Apart from that, there are slight differences in design and performance that will be discussed in the following sections.

2.3.1 NOR Matchline

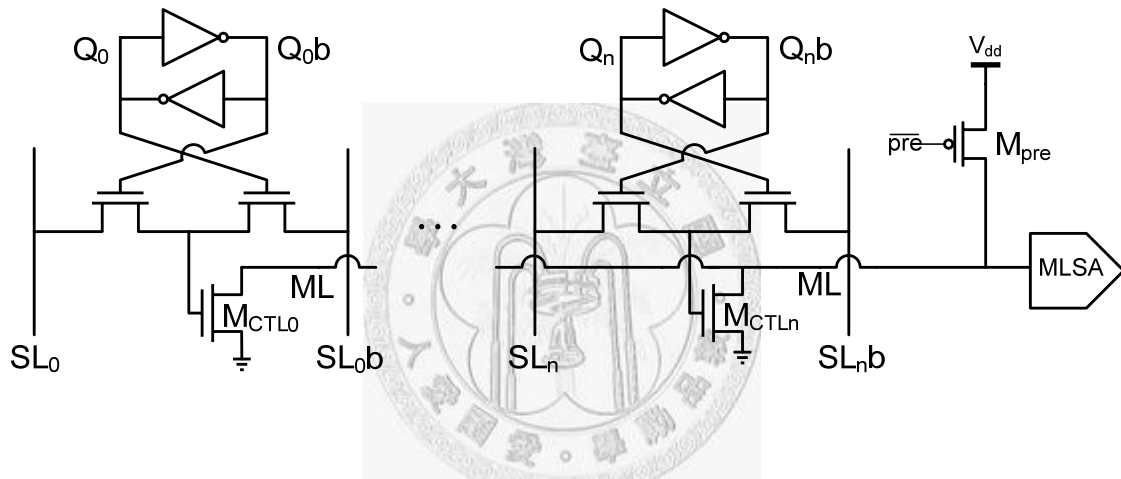


Figure 2.6: Schematic of the NOR matchline structure [3].

A NOR matchline consists of XOR type CAM cells that are connected in parallel as shown in Figure 2.6. The typical search process consists of three phases. These three phases are searchline precharge, matchline precharge and match evaluation. In the searchline precharge phase all searchlines are precharged low. Like this, all the pulldown paths are disabled to prevent the incorrect report of a mismatch although there has not been a comparison yet. In the matchline precharge phase the matchline is precharged high by the M_{pre} transistor. The matchline is now in a temporary match state which is a prerequisite for the match evaluation phase. In this last phase of the search cycle the search word is broadcast onto the searchlines and the match evaluation takes place in each of the NOR CAM cells. A single CAM cell mismatch will connect the

matchline to ground and thus discharge it to a low state. If all CAM cells report a match, the matchline will keep its precharged high state. The matchline sense amplifier now evaluates the voltage on the matchline and if it senses a low state, it reports a miss; otherwise the match condition is fulfilled. Due to its parallel nature, the NOR matchline evaluates a hit or a miss at a very high speed which is the main feature of the NOR matchline.

2.3.2 NAND Matchline

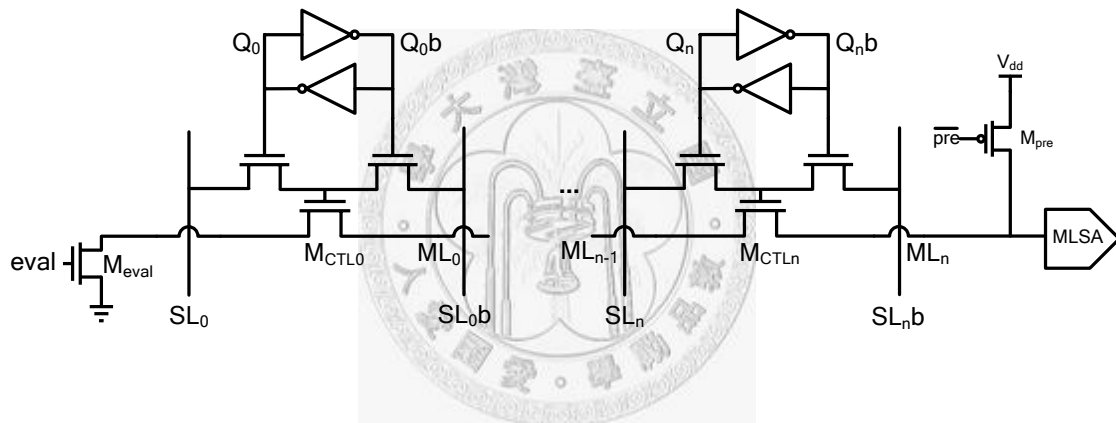


Figure 2.7: Schematic of the NAND matchline structure [3].

Figure 2.7 shows the NAND matchline structure. In this structure the matchline is formed by XNOR type CAM cells. The search cycle of the NAND matchline differs from the NOR matchline due to the different architecture of the cell type used. In the first phase the precharge transistor M_{pre} sets the initial matchline voltage to V_{dd} . Then the search word is broadcast onto the searchlines. To start the evaluation phase, the evaluation transistor M_{eval} is turned on. When turned on, the evaluation transistor connects the matchline to GND. So in the match case where all the pass transistors within the XNOR type CAM cell will be turned on, the precharged voltage will discharge to a low state. If at least one CAM cell has a mismatch, the connection to

GND will be cut off and the matchline maintains its precharged high state.

2.3.3 Comparison

As opposed to the NOR matchline structure, the matchline sense amplifier of the NAND matchline interprets a high matchline state as a miss and a voltage drop as a match. The NAND matchline is slower than the NOR matchline structure. In the worst mismatch case just the last CAM cell in the XNOR cell chain reports a mismatch. So the final evaluation result will not be ready until each of the CAM cells was traversed in a serial manner. The critical evaluation path leads through all the pass transistors of the CAM cells that form a word as opposed to just two transistors of one of the pulldown paths in the parallel NOR matchline structure in case of a one bit miss.

2.4 CAM Relative Research

This section gives a brief overview of Precomputation-Based CAM that has been proposed to reduce the high power consumption of content-addressable memories. This architectural technique can be implemented in at least two different methods that will be discussed more detail later in the beginning of next chapter since they served as a foundation for the proposed Encoded PB-CAM.

The precomputation-based approach has a character that the whole matchline is just evaluated under some particular condition. The precomputation method works with different stages of evaluation. The basic idea of this approach is to extract some extra information, the so called parameter, from every data word and store it alongside the

data within the CAM. When a search operation takes place, the same parameter extraction algorithm is applied to the search word. Data words and their respective matchlines are considered possible matches when their precomputed parameter matches the precomputed parameter of the search word. So before the main comparison is done, the precomputed parameter of the search word is compared with the precomputed parameter of the data words stored within the CAM and just those matchlines are driven high and therefore considered as a possible match whose parameter matches the parameter of the search word. This method saves energy by just driving high matchlines that can be considered as a possible match in spite of driving high all the matchlines as conventional CAM.

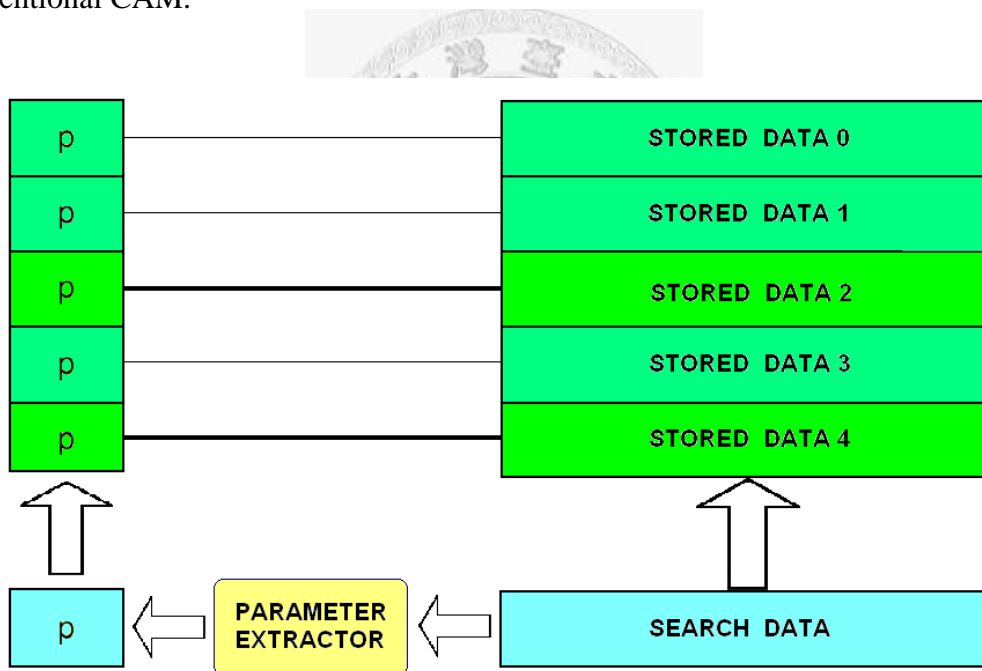
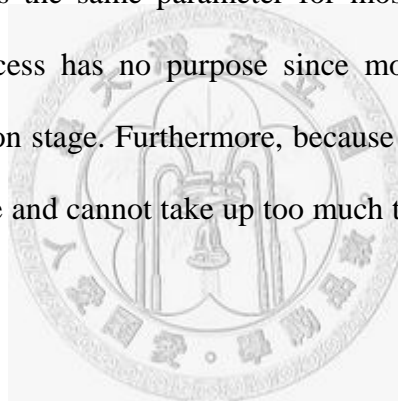


Figure 2.8: A concept view of precomputation.

Figure 2.8 is a concept view of precomputation. Instead of comparing the search word with all the data stored within the CAM, the parameter extractor extracts the search word's unique parameter. In the first comparison stage this search word parameter is compared with all the stored parameters of the data words in the CAM. Just the data

words who share exactly the same parameter are considered a possible match because the same parameter extractor was used for both the search word and the data words. In the depicted example the parameter of data word 0 and data word 2 match the parameter of the search word as indicated by the same color. Just the matchlines of these two data words will be precharged high and thus evaluated. When implementing the precomputation method in the CAM architecture, it is necessary to employ an extra precomputation function that equally distributes the outcome of the computation among all possible extraction results. This is comparable to hash functions that strive to produce hash sums with a minimum likelihood of collision with other hash sums. If a parameter extractor extracts the same parameter for most of its different inputs, the whole precomputation process has no purpose since most of the parameters would match in the first comparison stage. Furthermore, because the whole extraction process should be as fast as possible and cannot take up too much time, the parameter should be easy to compute.



Chapter 3

Encoded PB-CAM

In this chapter, we firstly introduce the precomputation-based technique called 1's count parameter extraction method proposed in [4], as well as the 7-tansistor (7T) CAM cells within it. Then we explain the drawback of 1's count method and how it is overcome by the so called Block-XOR precomputation-based method, which is proposed in [5]. Although the number of data words related to a single parameter is equal among all the possible parameters by using Block-XOR method, however, the power-saving 7T CAM cell is no more adoptable, only conventional 9-transistor (9T) CAM cells are suitable for it. Finally, in our proposed Encoded PB-CAM, we introduce one extra bit for every 4-bit block in Block-XOR method, such that the 7T CAM cell can also be applied to it, thus saving power.

3.1 Preliminary Work on PB-CAM

In this section, we introduced two PB-CAM that our proposed Encoded PB CAM is based on, which have been proposed by Lin in 2003 [4] and Ruan in 2008 [5]

respectively. These two methods are excellent, but there are still some limitations that we have to overcome.

3.1.1 1's Count

The precomputation approach was first applied to content-addressable memories by Lin [4]. In [4], the parameter circuitry that extracts the parameter from the data words as well as from the search word implements a 1's count. In WRITE operation, the number of "1's" within a data word is counted firstly and then stored alongside it within the CAM.

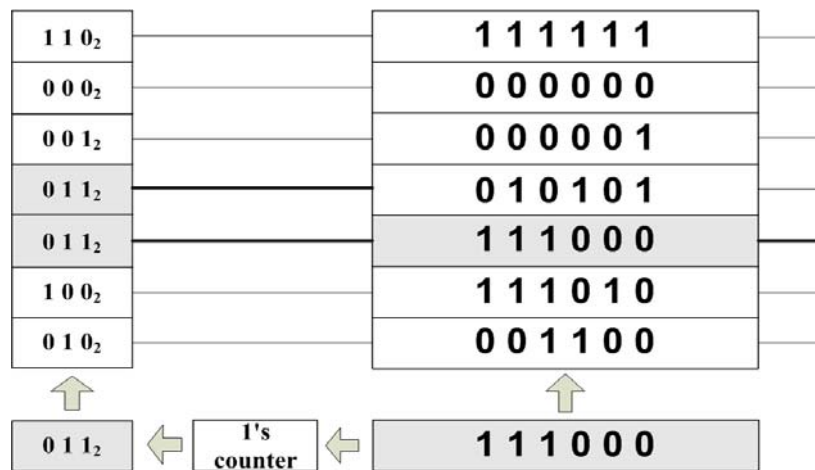


Figure 3.1: Conceptual view of the 1's count precomputation method.

In the SEARCH operation, the same parameter extraction circuitry also computes the number of 1's in the search word. In the first comparison stage, the number of 1's of the search word will be compared to the parameters within the CAM and just those matchlines with a matching parameter will be precharged as depicted in Figure 3.1. Just the data words with a precharged matchline will be compared with the search word in the final comparison stage which results in power savings.

Table 3.1: Average probability and number of data related to the 1's count parameters of a 14 bit word.

| 1's count | Related input data | Average probability |
|-----------|--------------------|---------------------|
| 0 | 1 | 0.01% |
| 1 | 14 | 0.09% |
| 2 | 91 | 0.56% |
| 3 | 364 | 2.22% |
| 4 | 1001 | 6.11% |
| 5 | 2002 | 12.22% |
| 6 | 3003 | 18.33% |
| 7 | 3432 | 20.95% |
| 8 | 3003 | 18.33% |
| 9 | 2002 | 12.22% |
| 10 | 1001 | 6.11% |
| 11 | 364 | 2.22% |
| 12 | 91 | 0.56% |
| 13 | 14 | 0.09% |
| 14 | 1 | 0.01% |
| Total | 16384 (2^{14}) | 100% |

This precomputation method however proved to be not very efficient since the number of data words that are related to the same parameter is not equally distributed among the different 1's count results. In [5], the disadvantages of using the 1's count as a precomputation method were illustrated. Table 3.1 and Figure 3.2 show the average probability and the number of data related to the same parameter of a 14-bit word, when the 1's count function is used to extract the parameter. As depicted, there are many more input words that share a 1's count of seven than for example a 1's count of zero, which is not desirable for the parameter. A desirable parameter extractor would equally distribute the number of data related to the same parameter among all the different possible parameters.

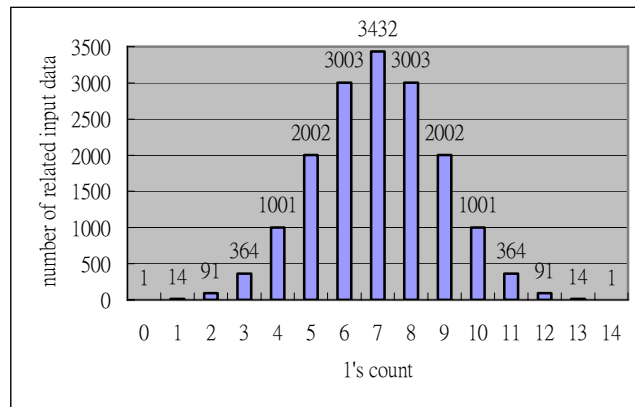


Figure 3.2: Distribution of input data among all parameters for the 1's count of a 14 bit word.

Another disadvantage of the 1's count is the complexity of the parameter extraction circuitry. The 1's count function is implemented with interconnected full adders that need several consecutive levels before the result is computed (shown in Figure. 3.3). For a 14-bit word the circuit complexity to implement the 1's count function might still be tolerable but with a growing number of data bits the area needed and the delay caused by the circuitry increases to a strong level which will affect the overall performance of the CAM.

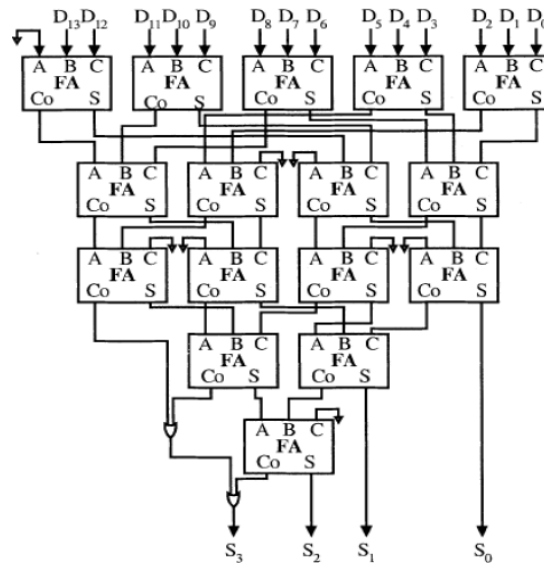


Figure 3.3: 14-bit 1's count parameter extraction circuit of [4].

In 1's count method, it is no doubt the conventional 9T CAM cell is suitable for storing data in those data words. However, a smart power-saving 7T CAM cell is also proposed in [4], which deserves to be mentioned here because it is a part the main building block of our Encoded PB-CAM.

A 7T CAM cell is shown in Figure 3.4. This cell is composed of a standard five-transistor D-latch device to store a data bit and a NAND-type comparison circuit containing two NMOS to form a discharging path for the matchline. In the 7T CAM cell, the access performance is worse than in traditional 9T CAM cells because of the single-ended writing structure. However, the poor writing performance dose not matter because the writing operation seldom occurs in CAMs.

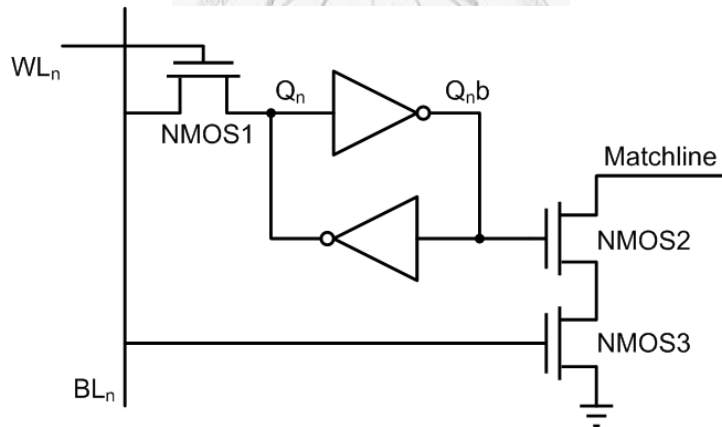


Figure 3.4: A 7T CAM cell.

Compared with the traditional CAM cell circuit, the proposed PB-CAM cell circuit has an incorrect condition accrued in the SEARCH operation. Notice that when the stored data Q_n is logic “1” and the search data on BL_n is logic “0”. This is a case of mismatch (BL_n is unequal to stored data Q_n) and the matchline should be pulled down to GND, but an undesirable condition happens here: BL_n and Q_nb turn off NMOS3 and NMOS2, respectively. As a result, the input data mismatches the stored data, but the match line is kept in a precharged voltage (V_{dd}).

Table 3.1: Truth table of 9T cell and 7T cell.

| status | Search data on BL_n | Stored data in Q_n | 9T | 7T |
|----------|-----------------------|----------------------|----------|----------|
| Match | 0 | 0 | V_{dd} | V_{dd} |
| | 1 | 1 | V_{dd} | V_{dd} |
| mismatch | 1 | 0 | 0 | 0 |
| | 0 | 1 | 0 | V_{dd} |

The truth table of both the conventional 9T CAM cell and the 7T CAM cell is shown in Table I. Although 7T cell has an incorrect condition occurring in the data searching operation, it can be ignored based on the explanation of the conceptual view of word structure shown in Fig 3.5. The 1's count PB-CAM word circuit has three cases for the data searching operation, each of them are explained as below.

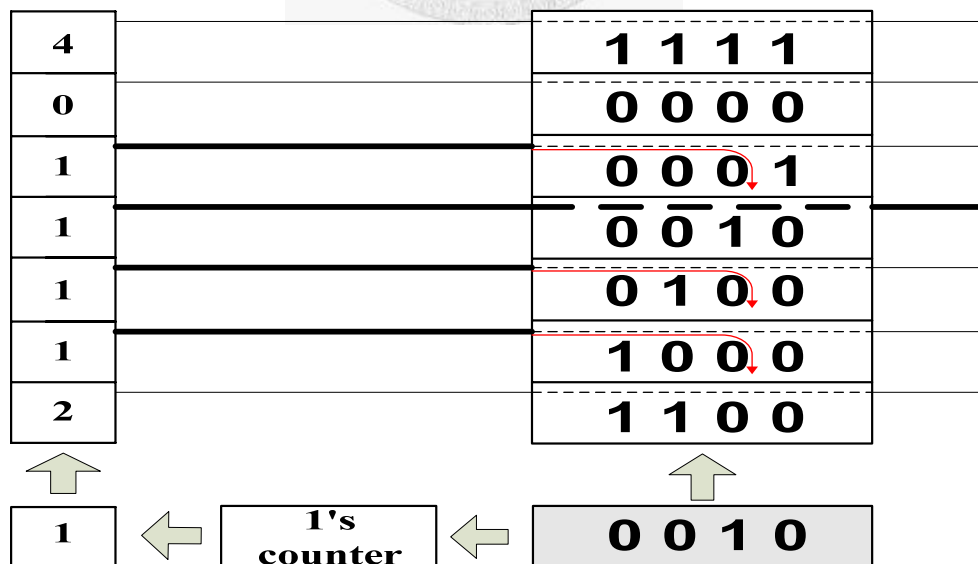


Figure 3.5: Conceptual view of word structure composed of 7T cells.

Case I : search data on BLs equal the stored data word. In this case, since the parameter also equals to that of the search data word, the matchline is precharged to V_{dd} . The precharged matchline has no discharge path to GND, so logic “1” is left as the output.

Case II : parameter and search data on BLs are both not equal to the stored data word. In this case, the parameter also mismatch that of the search data word, the matchline is not precharged and kept in logic “0” as output.

Case III : parameter of the search data equal to that of the stored data word, but the data on BLs mismatch the stored data word. In this case, the matchline is precharged to V_{dd} , but there is at least one pull-down path to GND, so logic “0” is the output.

In the example shown in Figure 3.5, a data “0010”, which has the 1’s count parameter of 1, is being searched. It is fed into 1’s counter and the data word block simultaneously, as a result, all data word relative to 1’s count of 1 is precharged. Observe that among those precharged data word, “0001”, “0100” and “1000” are unequal to searched data “0010”, they have the pulled down path respectively because there is at least a stored “1” evaluated to a “0” on the bitlines. Eventually, the 7T cell works correctly.

3.1.2 Block XOR

To overcome the previously described problems, Ruan introduced a new parameter extraction method called Block-XOR approach [5]. Instead of counting the ones of a given data word, every four bits of the data word form a block and from every 4-bit block one single parameter bit is extracted by using three XOR gates as shown in Figure 3.6.

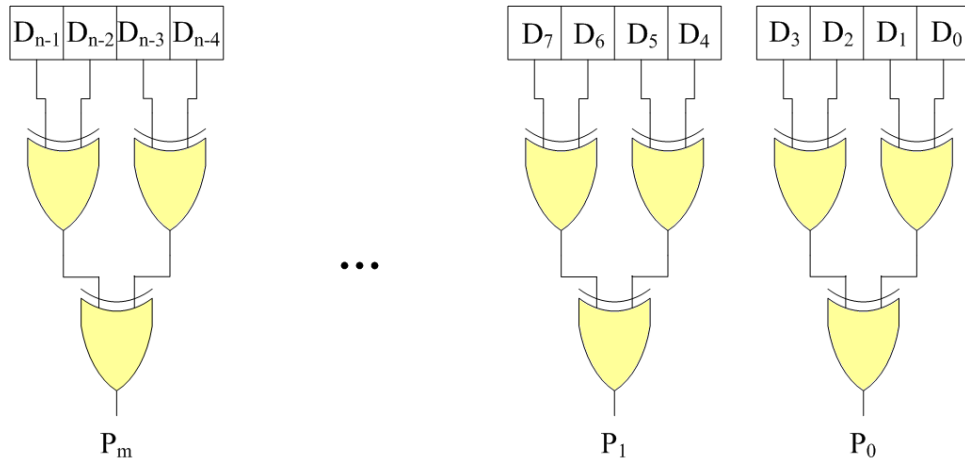


Figure 3.6: N bits Block XOR circuitry of [5].

There are two main advantages of the Block XOR approach as compared to the 1's count method. Most importantly, the number of data words related to a single parameter is equal among all the possible parameters. Table 3.2 shows all the 16 different 4-bit combinations and their resulting parameter when extracted with the Block XOR method. As can be seen from the table, a '0' as well as a '1' within the parameter bits is both related to eight different 4-bit combinations. For further illustration, we have a look at the number of related data to a 4-bit parameter that was extracted from a 16-bit word using the Block XOR method. Each single '0' or '1' of the parameter is related to eight different 4-bit combinations of the data word, and since the 4-bit blocks are not interdependent the multiplication rule can be applied to show that each parameter has $8 \times 8 \times 8 \times 8 = 4096$ input words related to it. A 16-bit word has $2^{16} = 65536$ different data words, so the 16 parameters are equally distributed among all the input words.

Table 3.2: All 16 4-bit combinations and their resulting Block XOR parameter.

| 4 bit block | Block XOR Parameter |
|-------------|---------------------|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 1 |
| 0011 | 0 |
| 0100 | 1 |
| 0101 | 0 |
| 0110 | 0 |
| 0111 | 1 |
| 1000 | 1 |
| 1001 | 0 |
| 1010 | 0 |
| 1011 | 1 |
| 1100 | 0 |
| 1101 | 1 |
| 1110 | 1 |
| 1111 | 0 |

The other advantage is the easy implementation of the Block XOR method. Even for larger input words, it is simple to implement the XOR parameter computation. Each parameter bit is extracted by just three XOR gates which makes this method not only easy to implement but also very fast.

Although the Block XOR method solves inequality of distribution of data related to the same parameter among all the different possible parameters, the power-saving 7T CAM cell is not feasible for storing data word. Figure 3.7 shows the reason why we can not make use of 7T CAM cell.

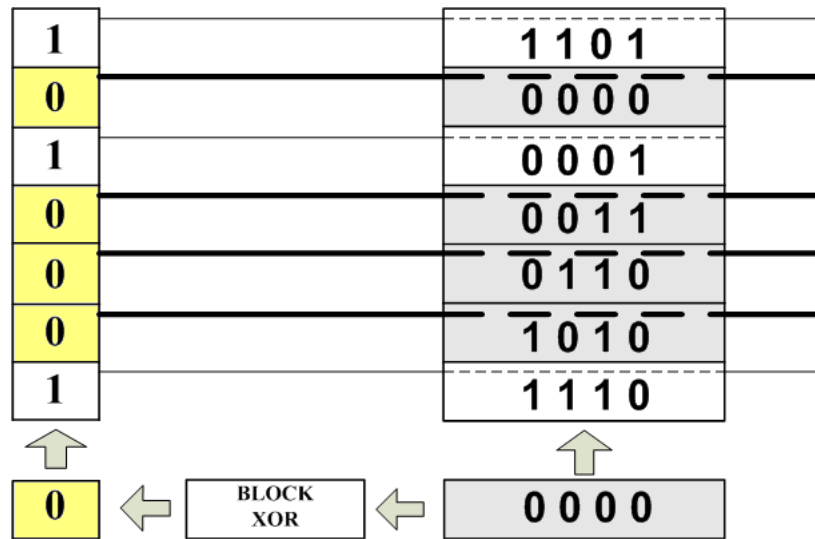


Figure 3.7: The error condition occurs when using 7T CAM cell in Block XOR method.

The search operation can be divided into three stages. Assume a group of data “0000” is being searched in a Block XOR PB-CAM. Firstly, through the parameter extractor composed of three XOR gates, a parameter of “0” is obtained. Secondly, the matchlines of data words which have the same parameter as “0” are all precharged to V_{dd} . The error condition occurs in the final stage, that is, the data comparison stage. Refer to Table 3.1, when “0000” is fed into the bitlines, because we use the 7T CAM cell, if the data stored in the 7T cell is logic “1”, then the discharge path to GND would be disconnected, as a result, the matchlines of data words “0011”, “0110”, “1100”, “0101”, “1010”, “0101” and “1111” would be kept in V_{dd} as output, which is a fatal error. This error condition happens not only when “0000” is being searched, the same problem is also taken place when any data word listed above is being searched.

3.2 Encoded PB-CAM

Up to now, two excellent PB-CAM have been introduced in detail. In the former section, the 1's count method adopts power-saving 7T CAM cell, but it suffers the problem of number of data relative to a single parameter. Whereas the Block XOR method solved the problem that 1's count method suffers, but the significant drawback is that it can not make use of the 7T cell. In this section, we propose an Encoded PB-CAM which strives to combine the 7T CAM cells in 1's count method with the Block XOR parameter extractor in the Block XOR method.

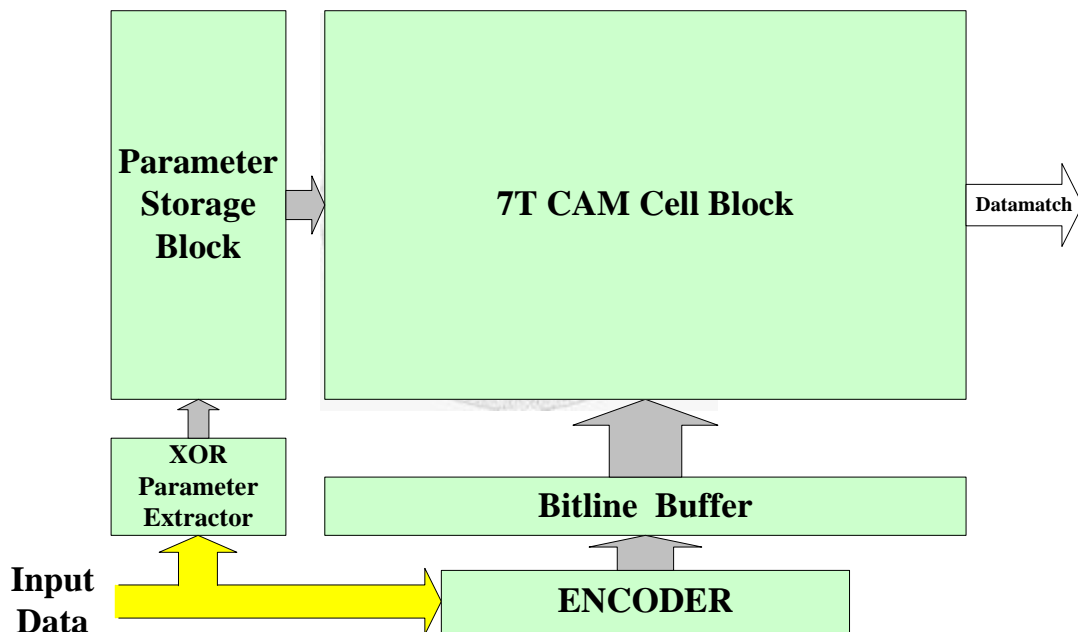


Figure 3.8: Proposed Encoded PB-CAM.

3.2.1 XOR Parameter Extractor

Figure 3.8 shows the main building blocks of the Encoded PB-CAM example architecture. The Encoded PB-CAM architecture is still a precomputation-based

architecture, which means that a parameter will be computed for each data word and stored alongside it within the memory banks. In the search process the same parameter extraction circuitry that extracted the parameter from the stored data words extracts the parameter from the search word. The parameter is then compared to all parameters within the active bank and just those matchlines with a matching parameter are driven high in order to activate them for the final comparison process. In our proposed Encoded PB-CAM, we use the same Block XOR parameter extractor that has been proposed by Raun in [5]. The operation of Block XOR parameter extractor has been introduced in section 3.1.2, so now we do not place emphasis on it.

3.2.2 Matchline Selection

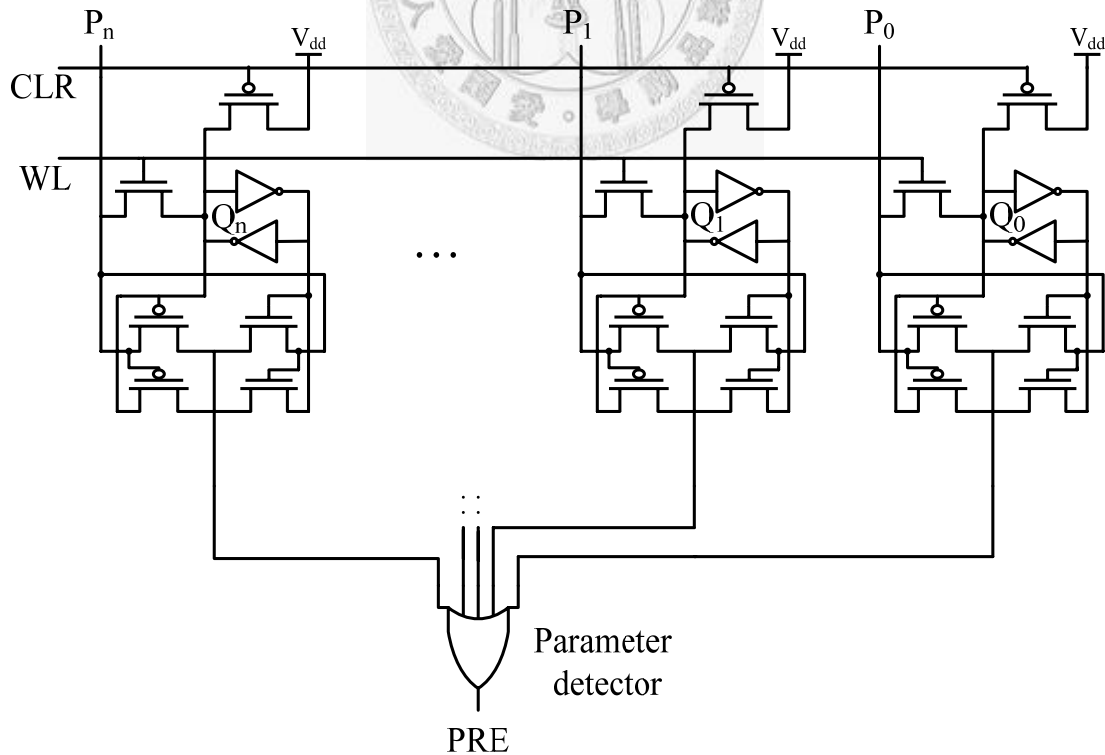


Figure 3.9: Static parameter comparison circuit in [4].

Figure 3.9 shows the parameter comparison function which is realized by the static CMOS structure. The operation of this circuit is similar to the conventional CAM word circuit, which compares the parameter of the input data with the parameter of stored data and then generates the PRE signal to decide whether precharge the related matchline or not. It is composed of the ten-transistor CAM cell with the clear signal and the comparison detector which is implemented by an OR gate is utilized to identify the comparison result. The operation of this circuit is introduced as follows. In the parameter writing operation, the WL signal rises to V_{dd} to allow the parameter of input data to be written into the parameter storage unit. In the parameter comparing operation, if the parameter of input data mismatch the parameter of data word that have been previously stored in the storage unit, then the output signal PRE of the comparison detector becomes logic “1” to disable the PMOS MP_{PRE} shown in Figure 3.10. Otherwise, the output of parameter detector equals zero to enable M_{PRE} and form a current source of related matchline.

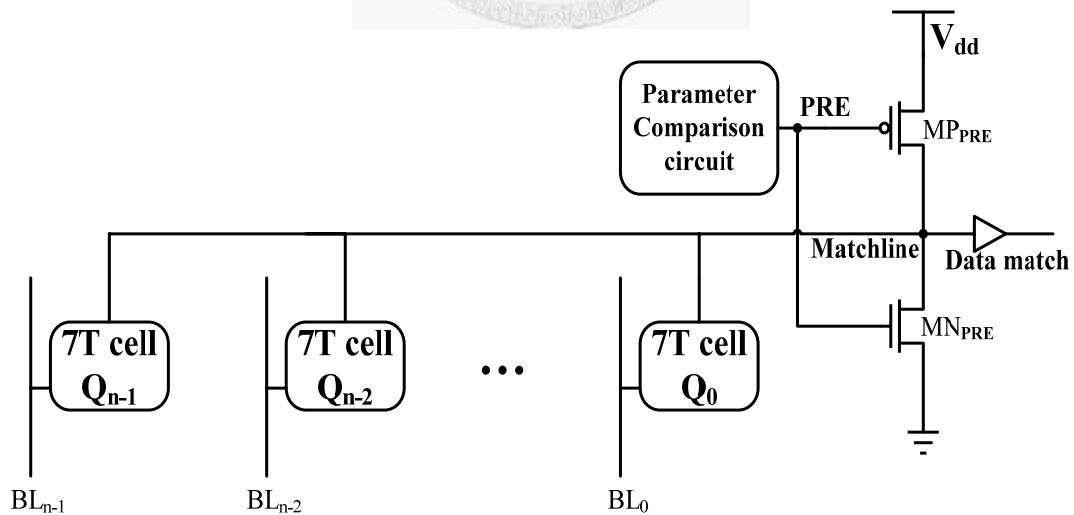


Figure 3.10: Matchline structure with parameter comparison circuit.

3.2.3 Encoding Scheme

As mentioned in the end of section 3.1, the Block XOR architecture can not make use of power-saving 7T CAM cell because the error conditions that occur when searching some specific data word. Through our proposed encoding skill, we overcame this problem and combined the 7T cell with Block XOR parameter extractor.

Table 3.2: Truth table of proposed encoding scheme.

Table 3.2 (a)

| parameter | Input Data | Reduced Data | Encoded Data |
|-----------|------------|--------------|--------------|
| 0 | 0000 | 000 | 11000 |
| | 1001 | 001 | 10001 |
| | 1010 | 010 | 10010 |
| | 0011 | 011 | 01011 |
| | 1100 | 100 | 10100 |
| | 0101 | 101 | 01101 |
| | 0110 | 110 | 01110 |
| | 1111 | 111 | 00111 |

Table 3.2 (b)

| parameter | Input Data | Reduced Data | Encoded Data |
|-----------|------------|--------------|--------------|
| 1 | 1000 | 000 | 11000 |
| | 0001 | 001 | 10001 |
| | 0010 | 010 | 10010 |
| | 1011 | 011 | 01011 |
| | 0100 | 100 | 10100 |
| | 1101 | 101 | 01101 |
| | 1110 | 110 | 01110 |
| | 0111 | 111 | 00111 |

Observe that the 16 kinds of different possible input have already been silently divided into the two halves at first stage of parameter comparing, a half is those which have an XOR parameter as “0”, other half are those which have an XOR parameter as “1”. The truth table of our proposed encoding scheme is shown in Table 3.2(a) and 3.2 (b). No matter the parameter is 0 or 1, there are only eight different input associate to each of the parameter, so we leave D0 to D2 and abandon D3. That is to say, in fact, after the Block XOR parameter extraction, we only have to store three bits for the second stage of data comparison rather than the whole four bits. But even so, the 7T cell is still not adoptable.

Take input data “000” and “111” for example, when we search “000” toward “111”, the matchline has no discharge path and results in a mistake. So we must “create” a discharge path for the matchline. If we store “000” as “11000”, and “111” as “00111”, then when search “11000” toward “00111”, a discharge path of matchline appears because the search operation includes a “0” toward “1”, which creates a discharge path for the matchline. And this purpose needs an extra encoder to accomplish. Table 3.2 shown in previous page contains the encoded data of each input data.

3.2.4 An Example of Proposed Encoded PB-CAM Operation

Figure 3.11 shows an example of parameter extraction and data encoding where the input data is “10101110”. Through our example, we can obtain parameter “01” and encoded data “0101010110”.

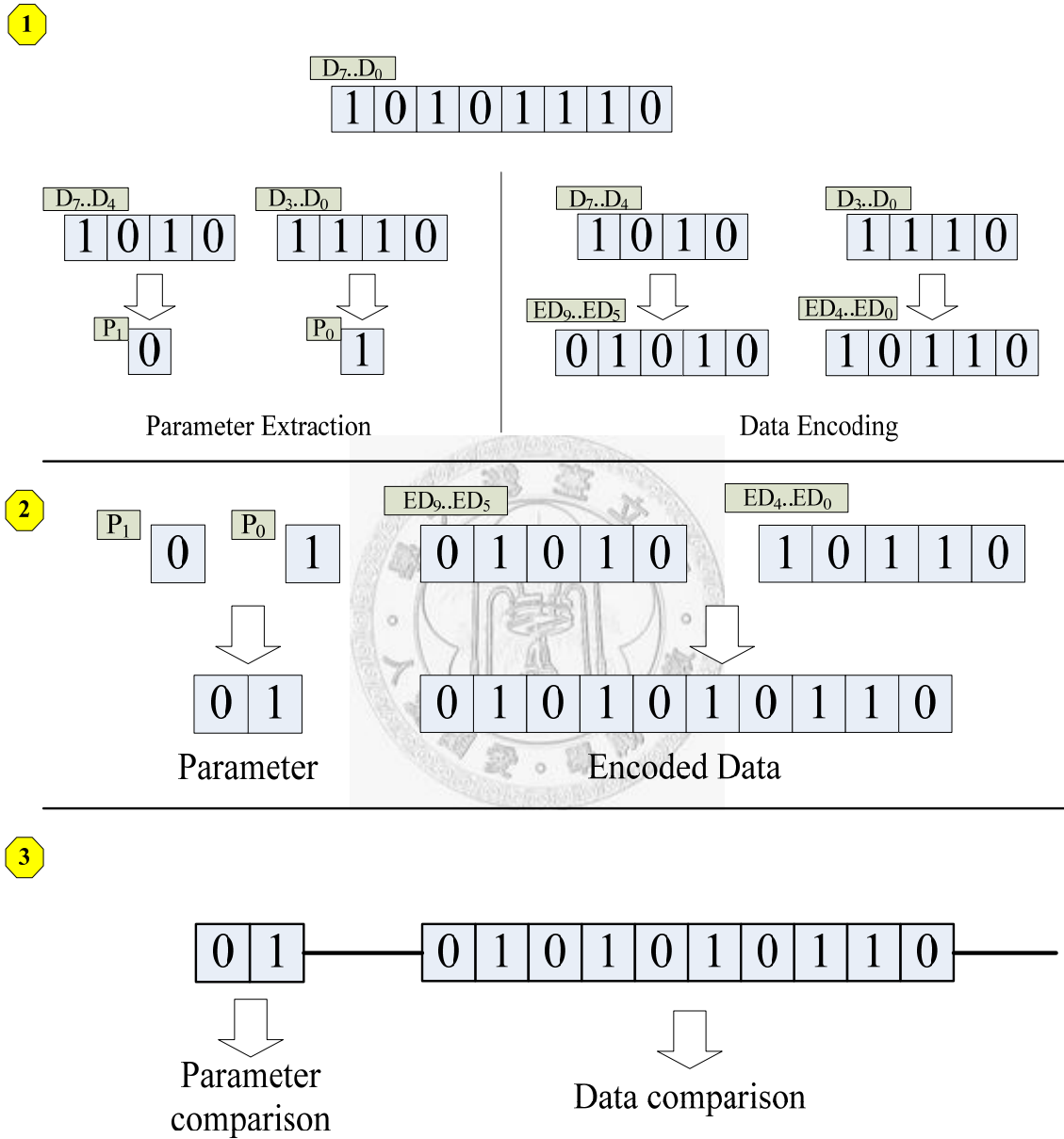


Figure 3.11: An example of parameter extraction and data encoding.

3.3 Encoded PB-CAM Benefits and Constraints

3.3.1 Encoded PB-CAM Benefits

We know bitlines are quite a high power consumption part inside the architecture of the memory device because it has parasitic capacity of a lot of MOSFETs. As to the 9T CAM cell, every cell needs two complementary bitlines to make comparisons with the data that have been stored in cell. And by our proposed encoding skill, though we transform four bits into five bits, all we paid is an extra encoder and one more CAM cell for every four bit blocks. But actually we turned four 9T CAM cell into five 7T CAM cell, furthermore, we reduced the number of high power consumption bitlines. Regardless of power consumed by the extra encoder, only consider the total number of MOSFETs that compose of those CAM cells of a four bit data word, it has reduced. For example, a 32-bit data word which is composed of 9T CAM cell, it requires $9 \times 32 = 288$ MOSFETs with 64 bitlines, and we only need $7 \times 40 = 280$ MOSFETs with 40 bitlines after encoding. The extra encoder is only made up of several logic gates, it does not cause too much power consumption and cost almost no area. So compare to Raun's original Block XOR method, power consumption thus reduced.

3.3.2 Encoded PB-CAM Constraints

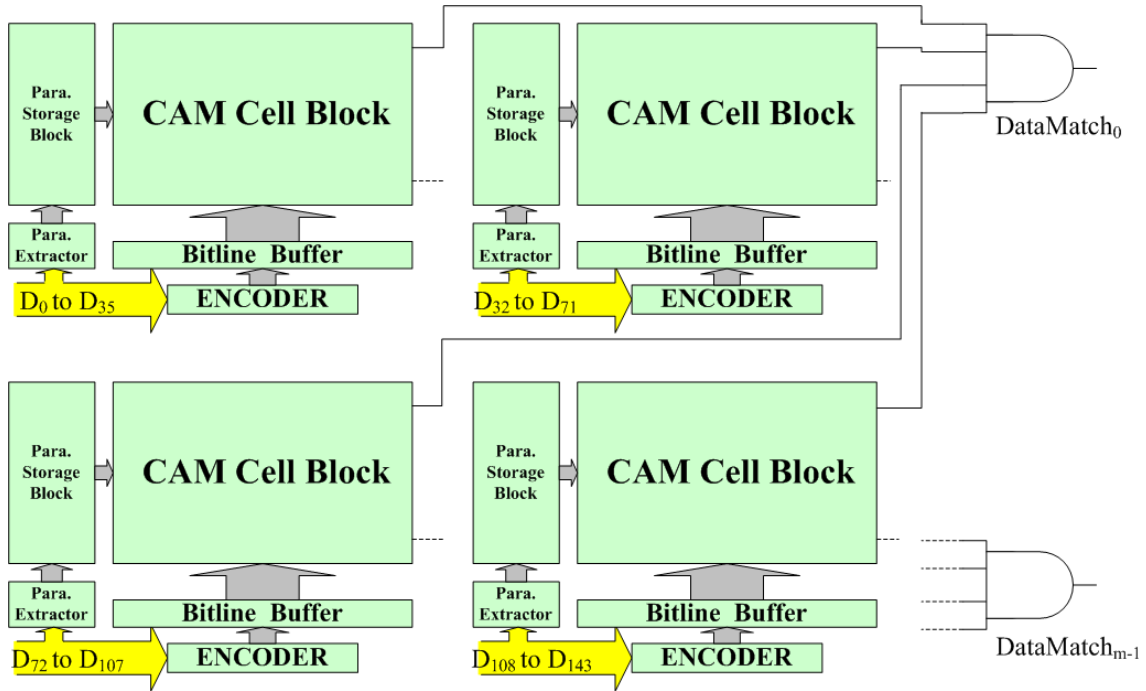


Figure 3.12: A conceptual view of suggestion when applying PB-CAM to a larger system.

When applying these PB-CAM architectures to a large system, we notice the number of bits of extracted parameter equals the number of bits of input divided by four. Suppose we have a system of 144-bits input, then there would be 36-bits parameter. And the comparison result of this 36-bits parameter would be fed into the parameter detector which is implemented by an OR gate. Even after optimized, such an OR gate still cause a long delay time, So we suggest that when applying this architecture to a large system, it would be better to divide the whole system into several smaller ones, and feed those matchlines in the same address into an AND gates, then take the output of the AND gates as match signals. The conceptual view is shown in Figure 3.12.

Chapter 4

SPICE Implementation

In order to be able to thoroughly evaluate the proposed Encoded PB-CAM architecture in terms of power consumption and latency, SPICE models of the two reference architectures as well as the proposed Encoded PB-CAM architecture were designed and implemented. In our simulation, we use HSPICE with 0.18 μm technology file to evaluate power consumption and latency. Besides the encoder that is unique for Encoded PB-CAM, all the building blocks are identical in three architectures.

The two SPICE models shall be described in this chapter for the following purposes:

- Illustrate how an implementation of the Encoded PB-CAM architecture could look like and clarify how the implemented architecture exactly operates.
- Highlight the differences of the new approach as compared to the standard approach.
- Identify possible implementation problems or parts of the architecture that might have to be improved for real applications.

In the following sections the implementation of the three SPICE models will be described in detail.

4.1 Specifications

In order to be able to compare the simulation results of the references and the Encoded PB-CAM model, the three implementations have the same data word capacity and data word length. The models designed for this evaluation can both store up to 128 data words within their memory and the input data words are 32 bits long.

In our architecture, the address decoder and address encoder are identical to that used in the conventional CAM architecture, so these devices are not included in our evaluation of power consumption and latency.

4.2 Encoded PB-CAM Implementation

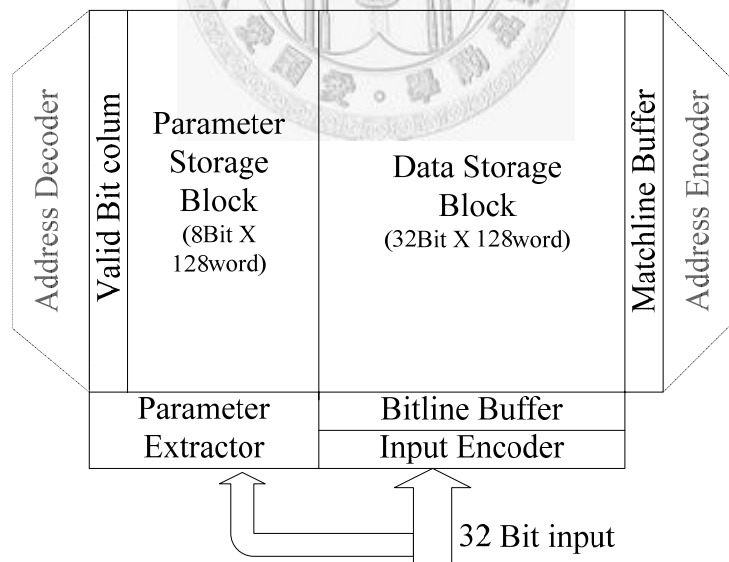


Figure 4.1: A simplified block diagram of proposed CAM architecture.

As mentioned above, we do not include address encoder and address encoder in our evaluation. We only take the power consumption of parameter extractor, valid bit

column, parameter storage block, data storage block, bitline buffer and matchline buffer into account.

4.2.1 Parameter Extractor

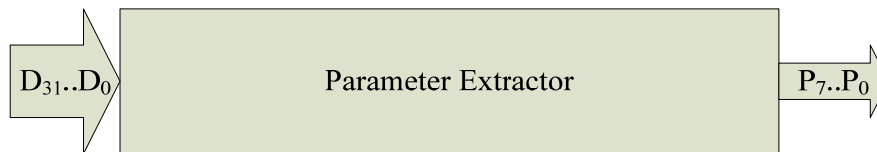


Figure 4.2: Conceptual view of Parameter Extractor.

Figure 4.2 shows a conceptual view of parameter extractor. Input of the parameter extractor is a 32 Bit data D_{31} to D_0 and the P_7 to P_0 as the output. Inside the parameter are eight combinational logic blocks which are composed of three XOR gates, as shown in Figure 4.3.

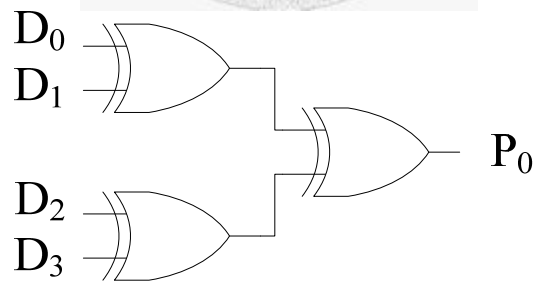


Figure 4.3: An example of XOR gates inside the Parameter Extractor.

4.2.2 Valid Bit Column

A valid bit is used to decide whether a row of data is valid or invalid. In our design, if

the valid bit is set as “0”, then the relative data word is valid. Otherwise, the relative data word is invalid. When a data word is set as invalid, even if the stored data match the search data, the matchline is not precharged. In other words, it is not included in the lookup table represented by this CAM.

To achieve this goal, we use an extra D-latch to store the valid information, and feed the output of this D-latch directly to the input of parameter detector, as shown in the figure below.

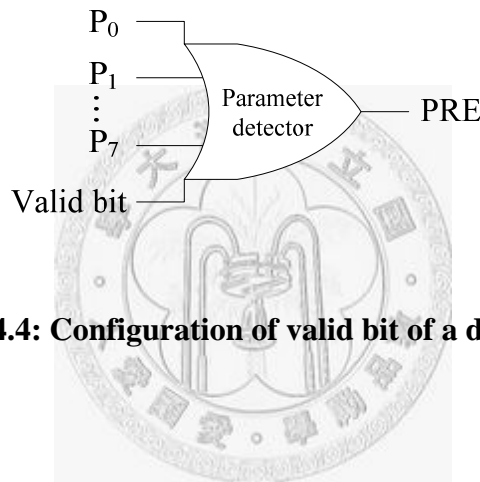


Figure 4.4: Configuration of valid bit of a data word.

4.2.3 Parameter Storage Block

The parameter storage unit is used to store the parameter bit that is extracted from the input data when the lookup table function is built up. Because in our design, we have 32 bit input, through the parameter extractor, we obtain 8 bit parameter word. In the search operation, parameter extractor extracts the parameter from input data and feed these parameters into the storage block to compare with that stored inside the block. If a row of parameter all identical to the search data and the row is set as valid, then the input P_0 to P_7 and valid bit of detector are all “0”, which results the PRE signal become “0” and turns on the PMOS to precharge the matchline of relative data word. Otherwise, one of input of parameter extractor would be “1”, which turns off the PMOS current source.

4.2.4 Bitline Buffer Implementation

The bitline buffer is one of the most obvious differences between XOR PB-CAM and Encoded PB-CAM. In our Encoded PB-CAM, we do not need a complementary signal for the 7T CAM cell, however we encoded each 4-bit block into 5-bit. So totally, we need 40 bitline buffers. Whereas in the Block XOR PB-CAM, although there are only 4 bit in a block, it requires complementary signal to ensure a correct search operation, so totally it needs 64 bitline buffers. The conceptual views are shown in Figure 4.5 (a) and (b), respectively. Observe that in Figure 4.5, the input is encoded data ED_0 to ED_{39} and output is BL_0 to BL_{39} , where the complementary signal is not contained.

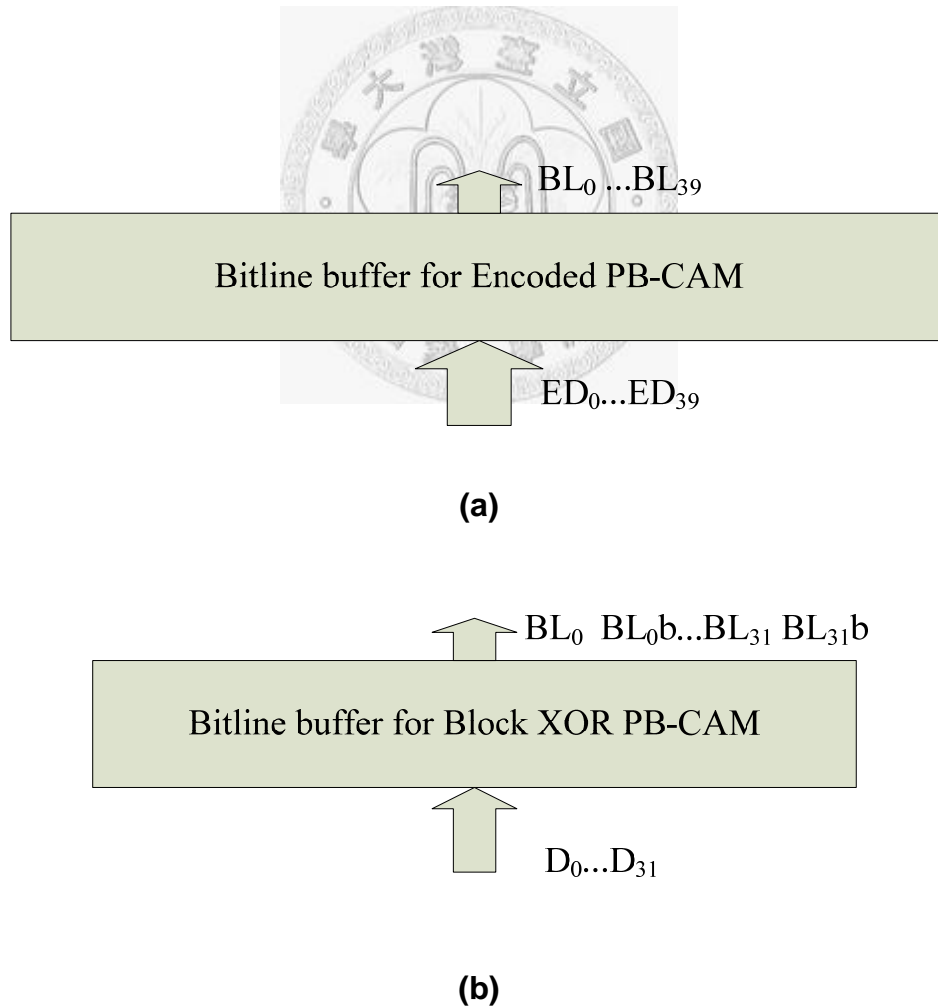


Figure 4.5 (a) and (b): Bitline buffer for both architectures.

4.2.5 Data Storage Block

The CAM cell block is the main building block used to store the data words, here we use 7T cell in our proposed architecture and 9T cell in Block XOR PB-CAM architecture.

4.2.6 Matchline Buffer

As in [4] and [5], we merely use two sequential CMOS inverters as the matchline buffer, so in our simulation result, the logic High performs the MATCH signal.

4.2.7 Input Encoder

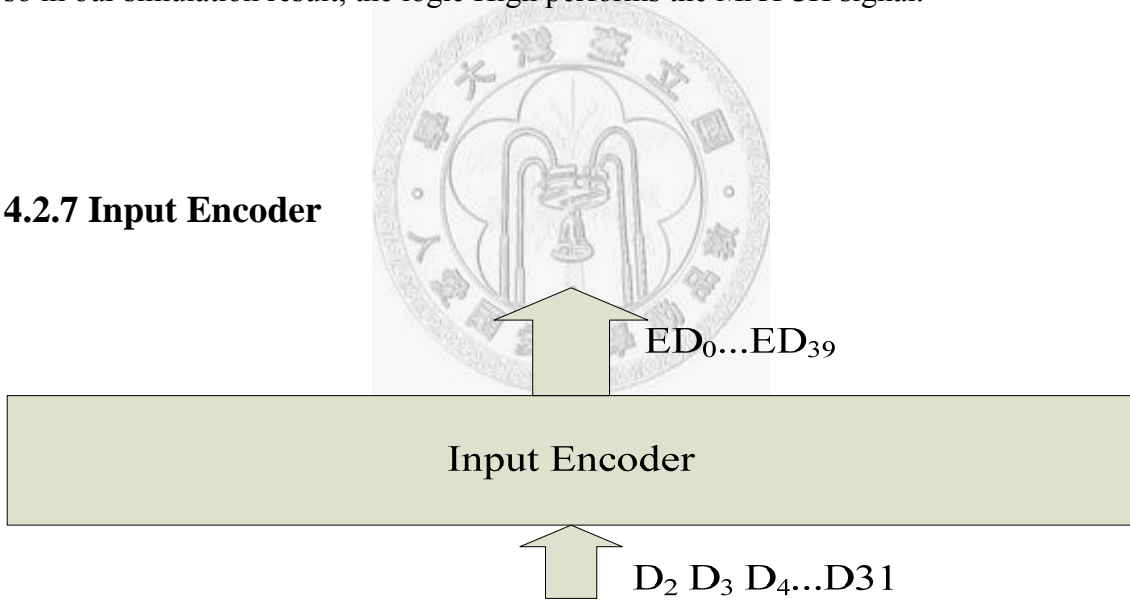


Figure 4.6: Input encoder.

The input encoder is the unique building block that appears in our proposed method. Its input is the search data word D_2 to D_4 , D_6 to D_8 and so on, and the output is encoded data ED_0 to ED_{39} . Its function is to encode a block of 3-bit input data into 5-bit output data which is suitable for the matchlines consist of 7T CAM cells. It is quite hard to realize the relationship between input data and the output data, so we implemented the

desired function by means of the aid of EDA tool. Figure 4.7 shows the synthesized circuitry by Synopsys Design Compiler. As shown in the figure, despite of the inverter, we need only six extra basic logic gates to implement our desired function.

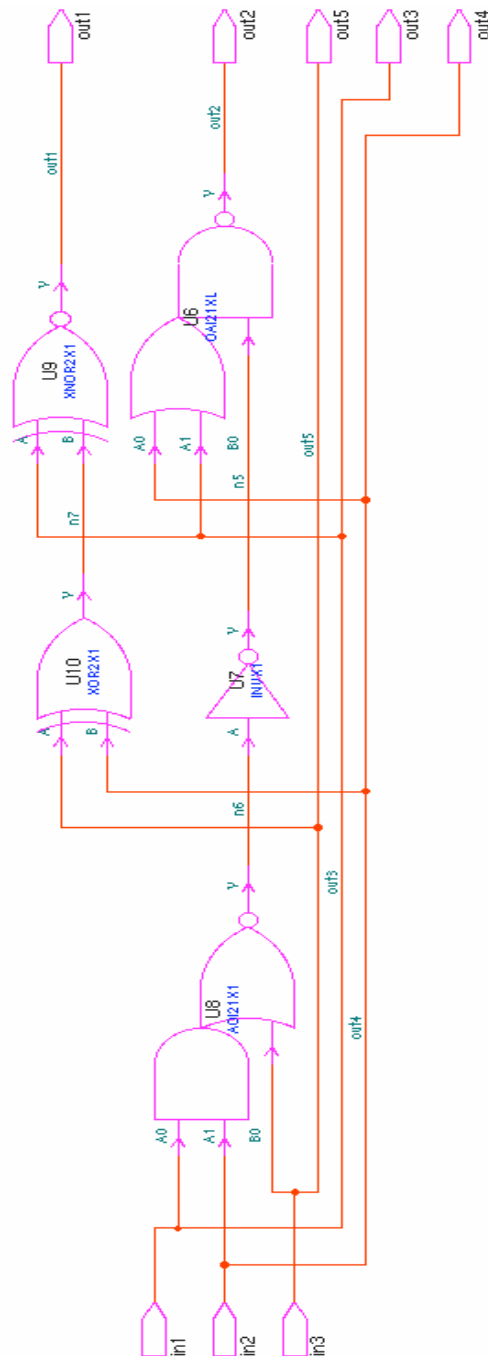


Figure 4.7: Input encoder synthesized by Synopsys Design Compiler.

Chapter 5

Simulation Results

In our design, we adopt Hspice to carry on simulation and verification of the whole circuit. Hspice can simulate the behavior and power consumption accurately so that we can avoid mistake in the design. As mentioned in the previous chapter, the memory capacity of Encode PB-CAM is 128 words by 32 bits.

During our simulation work, we firstly simulate the write operation of the CAM cell. Our purpose is to make sure the sizes of transistors in the CAM cells are all correct so that we can ensure our design is capable of executing write operations. After the write operation has been done, we started up to measure the power consumption and delay time of search operation of the whole architecture.

5.1 Power Consumption

In order to evaluate and compare the power consumption of the reference Block XOR PB-CAM model and the Encoded PB-CAM SPICE model, a set of random input stimuli was generated to simulate the search operation. These stimuli were then fed to the two

models and the average power consumption as well as RMS power were measured and output by .lis file after the simulations were finished. The search data were generated by a random number generator that produces uniquely distributed random data. Table 5.1 shows the Hspice simulation results for the input stimuli when run on the 0.18 μm technology model with $V_{\text{dd}} = 1.8\text{V}$. Figure 5.1 illustrates these results in a bar diagram.

Table 5.1: Comparison of power consumption.

| | Encoded PB-CAM | Block XOR PB-CAM | 1's count PB-CAM |
|--------------------------------------|-------------------|---------------------|---------------------|
| Average power (mW) | 9.21 | 12.05 | 13.41 |
| RMS power (mW) | 16.23 | 21.27 | 17.38 |
| Power-performance (fJ/bit/search) | 11.24 | 14.71 | 16.37 |
| Power-performance comparison (%) | 68.7 | 89.9 | 100 |

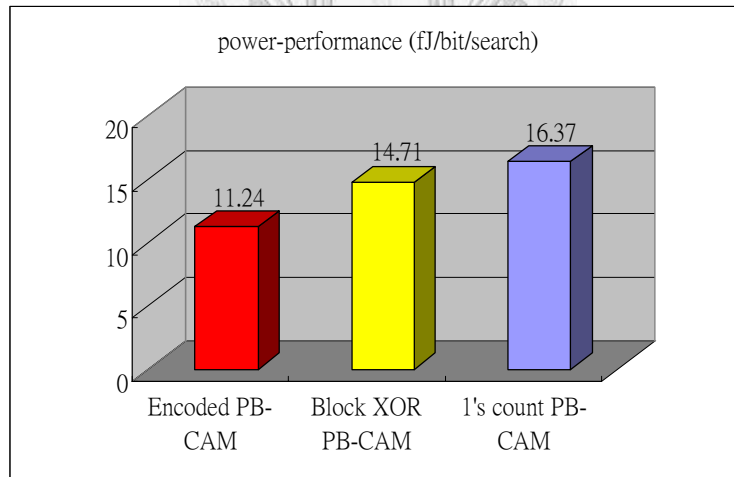


Figure 5.1: Power-performance comparison of Hspice 0.18 μm technology simulation.

The simulation shows significant energy savings of the Encoded PB-CAM compared to the Block XOR PB-CAM and 1's count PB-CAM. It results from the application of the power saving 7T CAM cell and uniformly distributed parameters.

5.2 Latency

In this section, we compare the latency of the three models. The latency is measured from the moment that input rise to half swing of system supply voltage to the moment that one of the matchlines also rises to half swing of system supply voltage.

Table 5.2 shows the delay time and power-delay product of three models. As we can see in the table, our proposed Encoded PB-CAM has a 0.3 ns longer delay than the Block XOR PB-CAM, which is caused by evaluation of the input encoder. However, Encoded PB-CAM has the best power-delay product among the three models. Figure 5.2 in next page depicts the bar diagram of latency and power-delay product.

Table 5.2: Comparison of latency.

| | Encoded PB-CAM | Block XOR PB-CAM | 1's count PB-CAM |
|---------------------------|----------------|------------------|------------------|
| Average delay (ns) | 2.51 | 2.23 | 3.64 |
| Latency comparison (%) | 68.9 | 61.3 | 100 |
| Power-delay product | 28.2 | 32.8 | 59.6 |
| Power-delay reduction (%) | 47.3 | 55.0 | 100 |

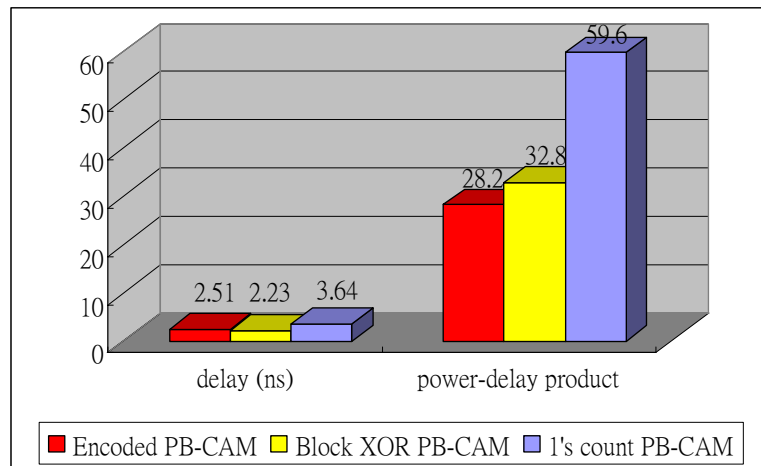


Figure 5.2: Comparison of delay and power-delay product.

5.3 Area

Here in this thesis, the area costs for the two reference model as well as the Encoded BP-CAM model are estimated by the number of transistors used in the implementations. Because the implementations of the three models are very similar, the area comparison in this case is fair. The only two implementation differences are the CAM cell and the input encoder; all the other building blocks are identical in the three models.

Hspice not only outputs the power consumption and the waveforms of the input and output signals, but also the number of transistors used by the SPICE models. Figure 5.3 shows that the implementation of the Block XOR PB-CAM consists of 52,224 transistors, while in the Encoded PB-CAM model just needs 51,455 transistors, and 1's count PB-CAM consists of 41,042 transistors.

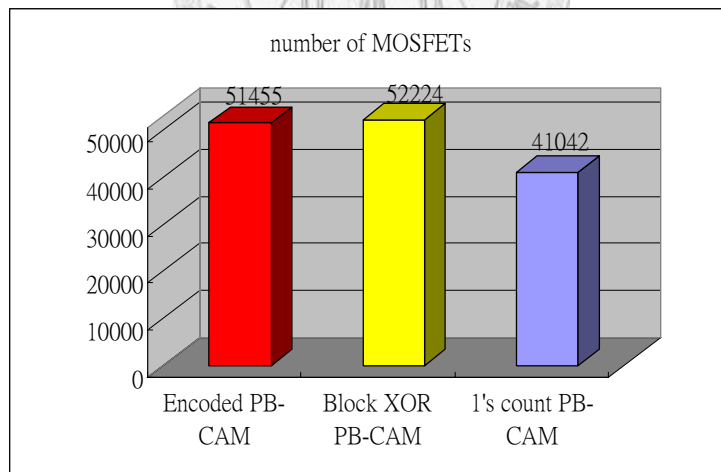


Figure 5.3: Number of MOSFETs used in both models.

As illustrated in Figure 5.3, the area costs of Encoded PB-CAM and Block XOR PB-CAM are almost the same, and the 1's count PB-CAM requires less area. Although our Encoded PB-CAM costs much area, it saves power. So it is a worth trade off.

Chapter 6

Conclusion

The objective of this work was to combine the advantage of two different precomputation based content-addressable memory, then implement it and evaluate reduction on power consumption and latency as well as the silicon area compared to the 1's count and Block XOR precomputation-based CAM architectures [4],[5].

The proposed Encoded-CAM architecture combines the circuit techniques of 7T CAM cell mentioned in 1's count precomputation-based CAM [4] and parameter extractor proposed in [5] by means of an extra input encoder so that the power consumption and power-delay product are significantly reduced. By the extra input encoder, we make use of 7T CAM cell to store data words and hence reduce the number of high power-consumed bitlines from 64 to 40 in a 32-Bits CAM architecture, thus the total power are saved.

The precomputation technique of the Encoded PB-CAM architecture accounts for a magnificent reduction of power consumption. The two-level comparison process excludes all the data words from the final comparison stage that do not have a matching parameter in the first comparison stage. This eliminates unnecessary comparisons and

therefore saves energy.

To evaluate the Encoded PB-CAM architecture in terms of power consumption, area and latency, SPICE models of the proposed architecture as well as reference 1's count and Block XOR PB-CAM architecture were designed and implemented.

The SPICE models were used within the Hspice circuit simulation tool with 0.18 μm technology file. To evaluate the power consumption of the three models, random input stimuli were generated. These input stimuli were used in the simulations to compare the average power consumption of the two models in search mode.

We built a 32-Bits 128 words model to evaluate power consumption, latency and area cost of both models. Hspice simulation shows Encoded CAM reduces 23.6% of average power consumption and 14% power-delay product compared to Block XOR PB-CAM . Furthermore, it reduces 31.3% of average power consumption and 52.7% power-delay product compared to 1's count PB-CAM.

Reducing the leakage power dissipation, without compromising speed performance, is one of the recent major research topics in VLSI design. Our work dose not put emphasis on leakage current, which could be left as future research topic.

Bibliography

- [1] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design – Circuits and Systems*. Kluwer Academic Publishers, Norwell, MA, 1995.
- [2] A. R. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Kluwer Academic Publishers, Norwell, MA, 1995.
- [3] K. Pagiamtzis and A. Sheikholeslami. *Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey*. IEEE Journal of Solid-State Circuits, 41(3):712– 727, March 2006.
- [4] J.-C. Chang C.-S. Lin and B.-D. Liu. *A low-power precomputation-based fully parallel content-addressable memory*. IEEE J. Solid-State Circuits, 38(4):654 662, April 2003.
- [5] Shanq-Jang Ruan, Chi-Yu Wu, and Jui-Yuan Hsieh. *Low Power Design of Precomputation-Based Content-Addressable Memory*. IEEE Trans. VLSI Syst., vol. 16, no. 3, page 331–335, Mar. 2008.
- [6] 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, no. 6, pp. 956-968, June 2001.
- [7] G. Thirugnanam, N. Vijaykrishnan, and M. J. Irwin, *A novel low power CAM design*. in Proc. IEEE ASIC/SOC Conf., Sept. 2001, pp. 198-202.
- [8] H. Choi, M. K. Yim, J. Y. Lee, B. W. Yun, and Y. T. Lee, *Low-power 4-way associative cache for embedded SOC design*. in Proc. IEEE 2000 ASIC/SOC Conf., Sept. 2000, pp. 231-235.
- [9] K. Pagiamtzis, and A. Sheikholeslami, *Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories*. Proceedings of the IEEE Custom Integrated

- Circuits Conference (CICC), pp. 383-386, Sep. 2003.
- [10] A. Roth, D. Foss, R. McKenzie, and D. Perry, *Advanced ternary CAM circuits on 0.13 μ m logic process technology*. Proceedings of the IEEE Custom Integrated Circuits Conference (CICC), pp. 465-468, Oct. 2004.
- [11] A. Efthymiou, and J. D. Garside, *A CAM with mixed serial-parallel comparison for use in low energy caches*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 3, pp. 325-329, Mar. 2004.
- [12] 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 Journal of Solid-state Circuits, pp. 1690-1696, vol. 33, no. 11, Nov. 1998.
- [13] C. A. Zukowski and S.-Y. Wang. *Use of selective precharge for lowpower contentaddressable memories*. In Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), volume 3, pages 1788 – 1791, 1997.
- [14] T. Sakata S. Hanzawa and K. Kajigaya. *A dynamic CAM based on a one-hot-spot block code-for millions-entry lookup*. In Symp. VLSI Circuits Dig. Tech. Papers, pages 382 –385, 2004.
- [15] I. Arsovski and A. Sheikholeslami. *A current-saving match-line sensing scheme for content-addressable memories*. In IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, pages 304 – 305, 2003.
- [16] T. Homma E. Komoto and T. Nakamura. *A high-speed and compactsize JPEG Huffmandecoder using CAM*. In Symp. VLSI Circuits Dig. Tech. Papers, pages 37 – 38, 1993.
- [17] T. Chandler I. Arsovski 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, 38(1):155 – 158, January 2003.
- [18] T. Kohonen. *Content-Addressable Memory*. Springer, New York, 2nd. ed. edition, 1987.
- [19] K. Ishibashi, K. Komiyaji, H. Toyoshima, M. Minami, N. Ohki, H. Ishida, T. Yamanaka, T.

- Nagano, and T. Nishida, *A 300-MHz 4-Mb wave-pipeline CMOS SRAM using a multiphase PLL*. IEEE J. Solid-State Circuits, vol. 30, pp.1189-1195, Nov. 1995.
- [20] 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, no. 11, pp. 1958–1966, Nov. 2003.
- [21] 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.
- [22] 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
- [23] 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 Syst., May 2004, vol. 2, pp. 23–26.

