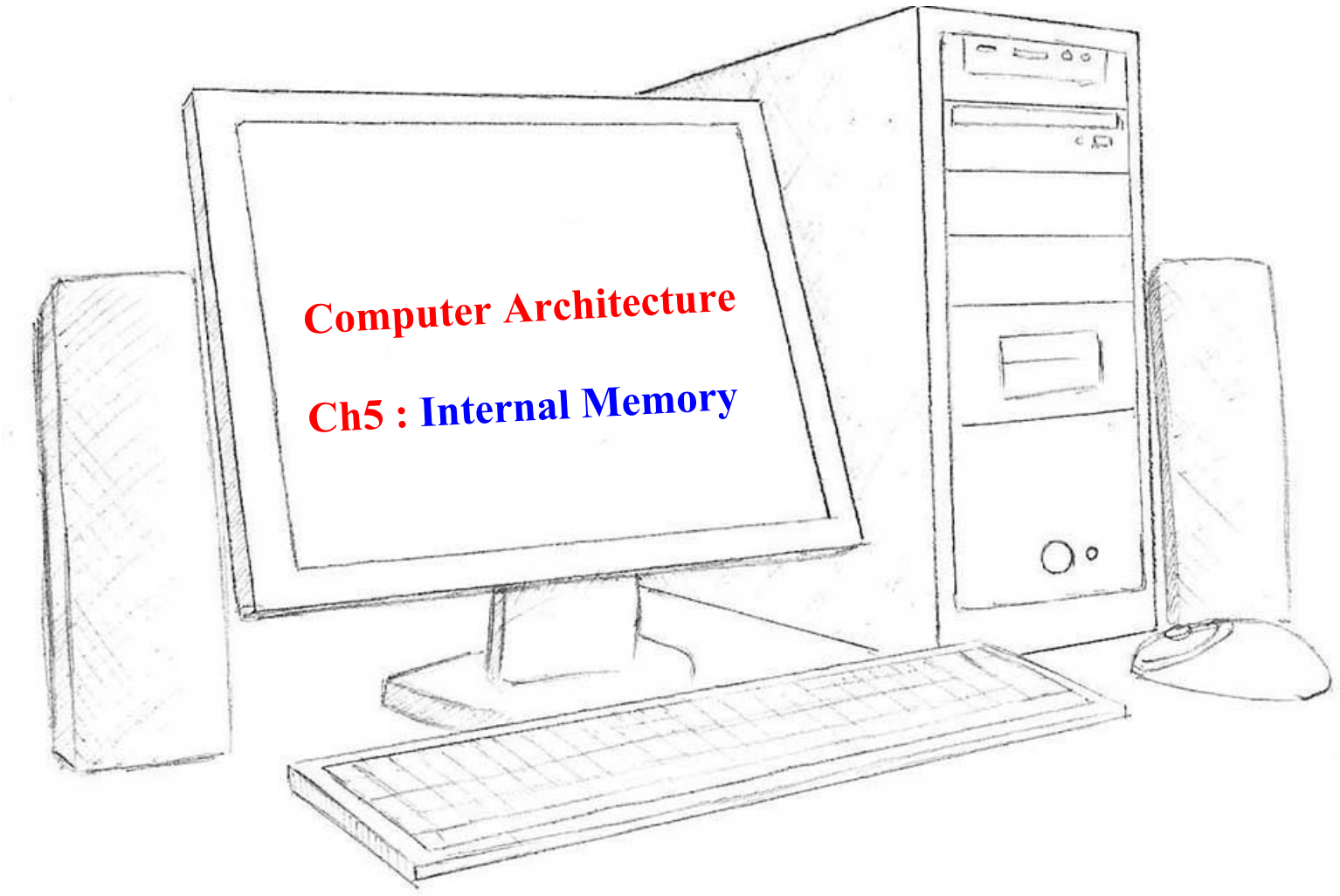


ดร. พิสิทธิ วิสุทธิเมธีกร : pisit.kmutnb@gmail.com

Part II : The Computer System

Ch 5: Internal Memory

5.1 Semiconductor Main Memory

5.2 Error Correction

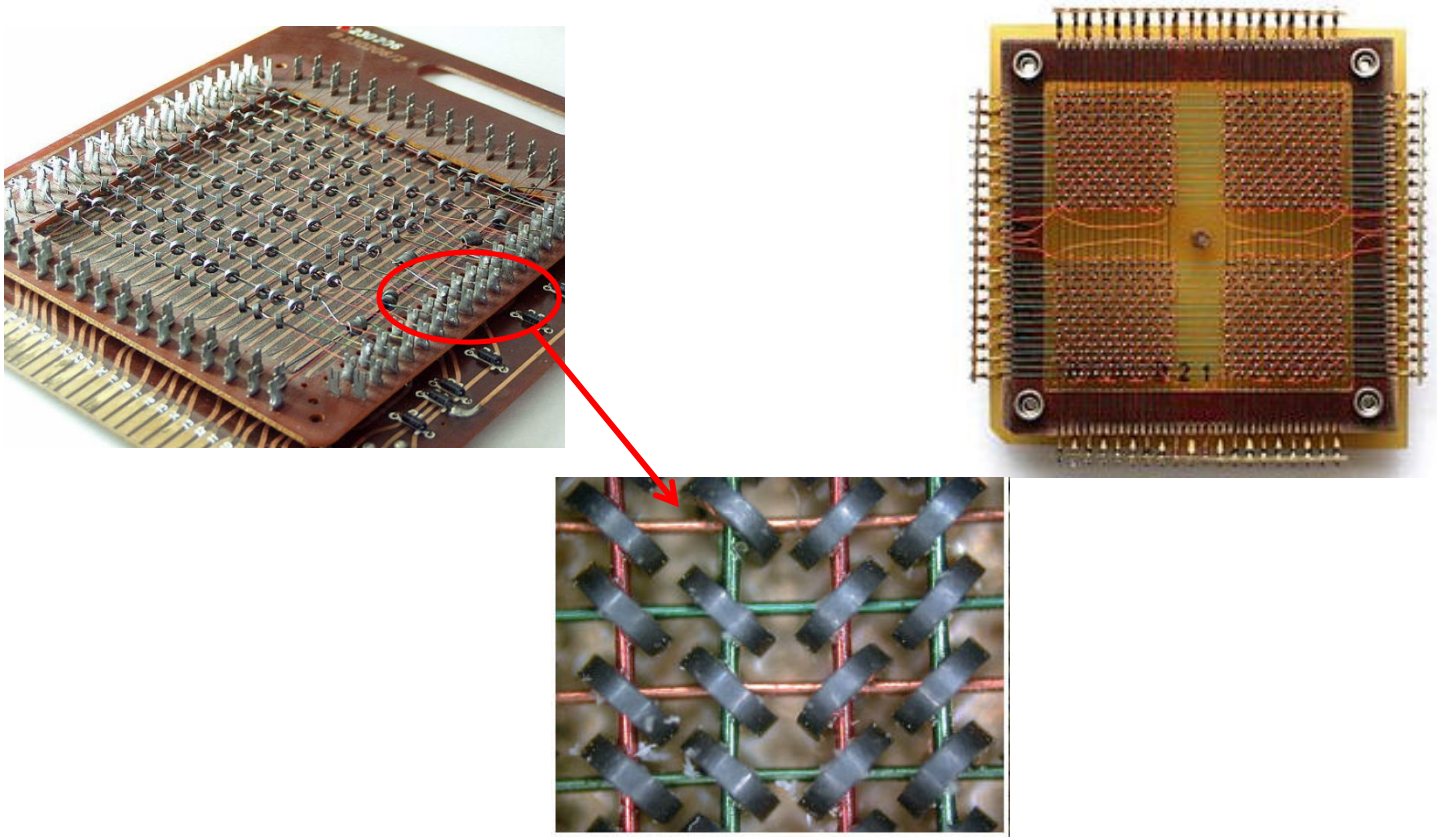
5.3 Advance DRAM Organization

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Present an overview of the principle types of semiconductor main memory.
- Understand the operation of a basic code that can detect and correct single-bit errors in 8-bit words.
- Summarize the properties of contemporary advanced DRAM organizations.

ในอดีตเครื่องคอมพิวเตอร์ในยุคแรกๆจะใช้หน่วยความจำแบบที่เรียกว่า Magnetic Core สำหรับเก็บข้อมูลหรือคำสั่ง นั่นคือใช้เป็นหน่วยความจำหลักโดยมีลักษณะดังในรูปที่ 5.1

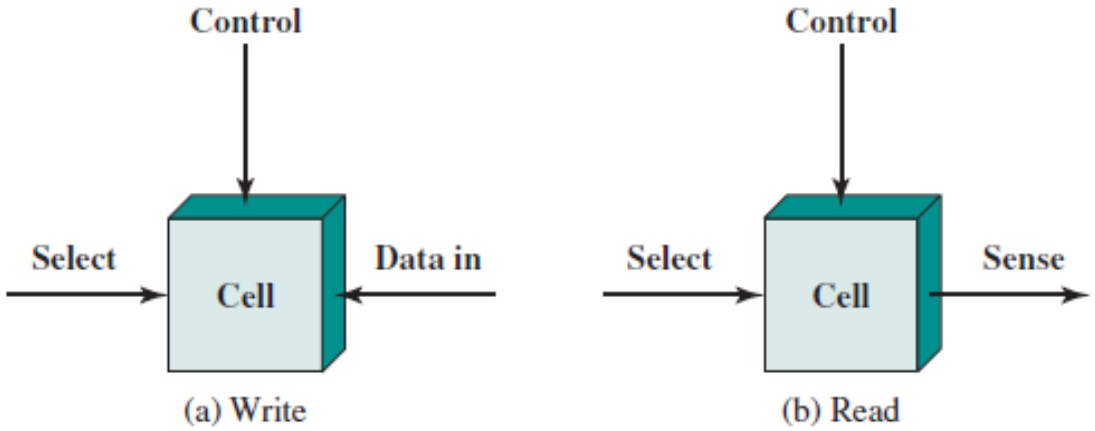


รูปที่ 5.1 Magnetic Core Memory

และเมื่อมีการพัฒนาเทคโนโลยีสารกึ่งตัวนำขึ้นมาก็ทำให้เกิดหน่วยความจำในรูปแบบของสารกึ่งตัวนำอย่างแพร่หลายและมีรูปแบบที่ต่างๆกันตามความต้องการในการใช้งานแต่อย่างไรก็ดีคุณสมบัติของหน่วยความจำที่พัฒนาจากสารกึ่งตัวนำจะมีคุณลักษณะ คือ

- การเก็บข้อมูลจะมีสองสถานะคือ Logic 1 หรือ 0
- สามารถเขียนข้อมูลไปยังหน่วยความจำได้ (การ write) อย่างน้อยก็หนึ่งครั้ง
- สามารถอ่านข้อมูลจากหน่วยความจำได้

ในรูปที่ 5.2 เป็นการทำงานพื้นฐานกับหน่วยความจำนั่นก็คือ การเขียนและอ่านข้อมูล



รูปที่ 5.2 Memory Cell Operation

ตารางที่ 5.1 Semiconductor Memory

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

DRAM and SRAM

ในตารางที่ 5.1 จะเป็นหน่วยความจำที่พัฒนามาจากสารกึ่งตัวนำทั้งสิ้น โดยที่ใช้กันมากก็คือ RAM ซึ่งมาจากคำว่า Random Access Memory อย่างไรก็ดีอาจจะเป็นการใช้งานคำที่ผิดความหมายไปเพราะแท้จริงแล้วหน่วยความจำที่เป็นแบบสารกึ่งตัวนำนั้นเป็นหน่วยความจำแบบ Random Access Memory ทั้งสิ้น

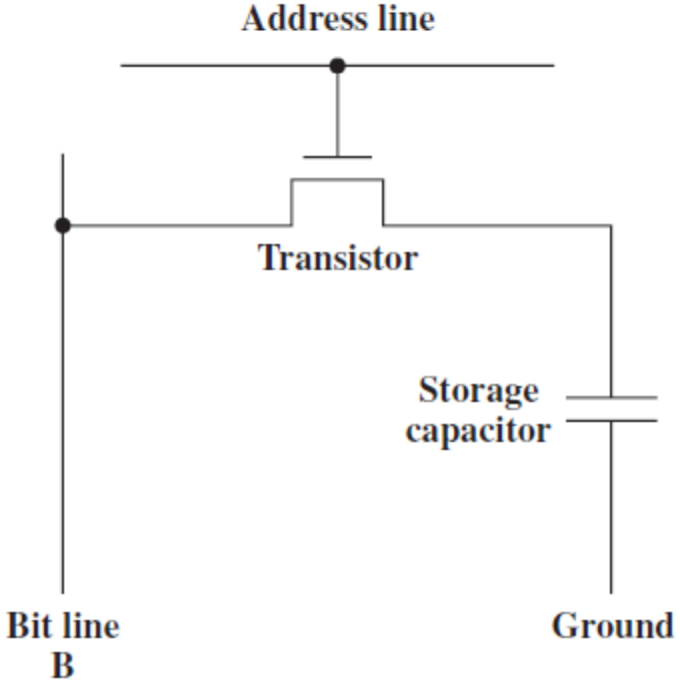
คุณสมบัติที่สำคัญของ RAM ได้แก่

- Volatile
- Read / Write using electrical signal

นอกจากนั้น RAM แบ่งออกเป็น 2 ชนิด คือ

- Dynamic RAM (DRAM)
- Static RAM (SRAM)

Dynamic RAM (DRAM) จะใช้วิธีการเก็บข้อมูลเป็นค่าของประจุที่ตัวเก็บประจุหรือที่เรียกว่า คาปาซิเตอร์ ดังในรูปที่ 5.3 โดยมีการทำงานคือเมื่อต้องการเขียนข้อมูลเก็บใน DRAM ก็จะส่งข้อมูลมาที่ Bit line พร้อมด้วยสัญญาณ address โดยสัญญาณ address จะทำให้ ทรานซิสเตอร์ ON ระดับแรงดันที่ขา Bit line จะถูกชาร์จมาที่ Storage capacitor จากนั้นให้สัญญาณแอดเดรสหายไปก็เป็นการจบขั้นตอนการเขียนข้อมูล

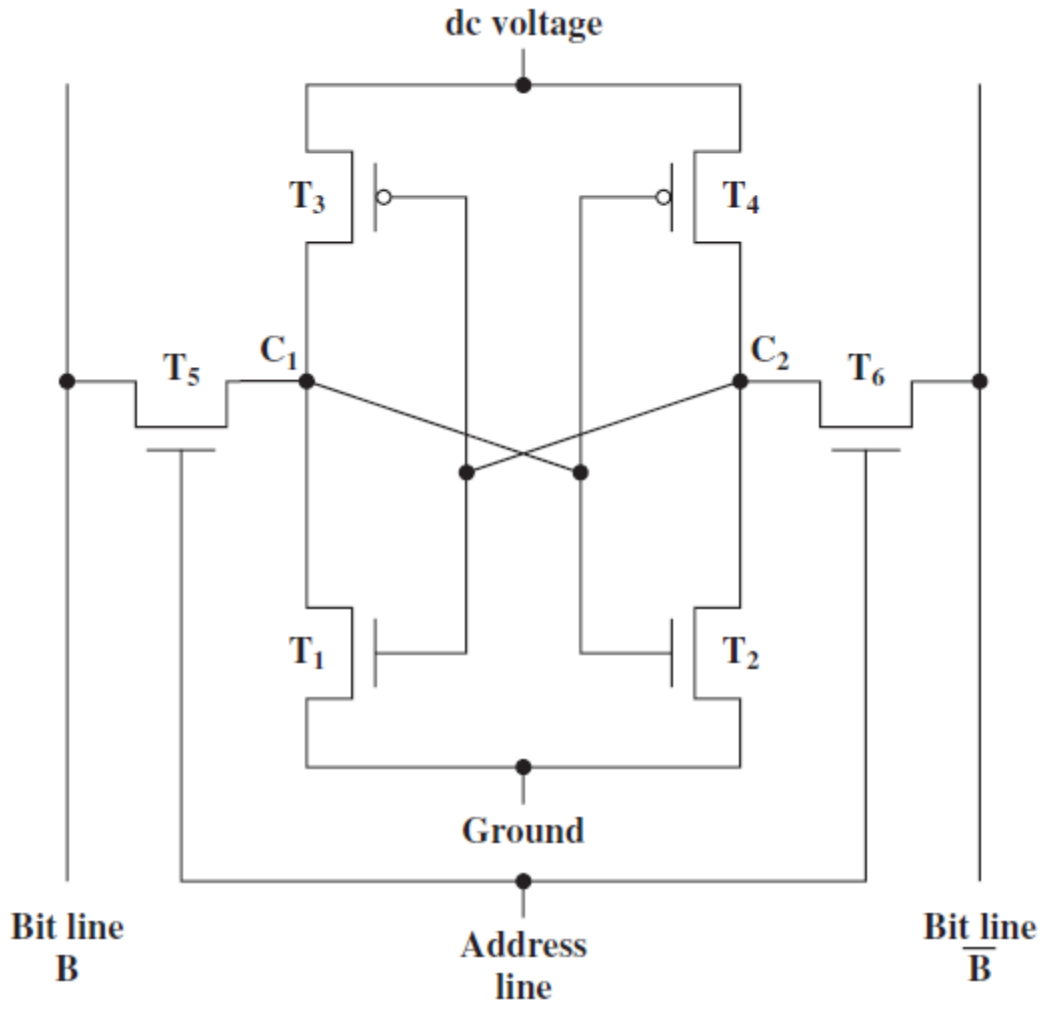


รูปที่ 5.3 Dynamic RAM cell

สำหรับการอ่านข้อมูลก็เพียงให้สัญญาณแอดเดรสมีค่าเท่ากับ 1 แล้วทรานซิสเตอร์ On ก็จะทำให้แรงดันที่ Capacitor ผ่านทรานซิสเตอร์แล้วไปปรากฏที่ขา Bit line และส่งต่อไปยัง Sense Amplifier เพื่อเปลี่ยนสถานะของแรงดันให้เป็นระดับแรงดันตามลอจิกที่ตรวจจับได้ซึ่งการ่านข้อมูลจะทำให้ประจุที่ capacitor เกิดการ discharge

ดังนั้นในการใช้งาน DRAM จะต้องมีการรีเฟรชข้อมูลที่เก็บไว้ในคาปาซิเตอร์ทุกๆ คาบเวลาหนึ่งเพื่อให้ข้อมูลที่เก็บอยู่ไม่ศูนย์หายไปนั่นเอง ซึ่งอาจกล่าวได้ว่าการรีเฟรชนี้เป็นความยุ่งยากประการหนึ่งในการใช้งาน DRAM ด้วยหลักการของ DRAM อาจกล่าวได้ว่า DRAM เป็นอุปกรณ์แบบ Analog แม้สิ่งที่เก็บจะเป็นบิตข้อมูลที่เป็นดิจิตอลก็ตาม

Static RAM (SRAM) ถือได้ว่าเป็นอุปกรณ์แบบดิจิตอลที่ใช้อุปกรณ์แบบลอจิกเช่นเดียวกับ Processor ในการเก็บข้อมูล โดยบิตข้อมูลใน SRAM นี้จะเก็บในอุปกรณ์ที่เรียกว่า Flip – Flop มีโครงสร้างดังในรูปที่ 5.4 ซึ่งพบว่ามี Transistor หลายตัวกว่าเมื่อเทียบกับกรณีของ DRAM แต่ข้อดีคือไม่ต้องรีเฟรชเหมือน DRAM และทำงานได้เร็วกว่า ส่วนข้อเสียก็คือเมื่อวงจรซับซ้อนกว่าราคาก็จะแพงขึ้น ขนาดต่อบิตก็จะใหญ่กว่า ดังนั้นมักจะถูกนำไปใช้เป็น Cache



รูปที่ 5.4 Static RAM cell

สรุปข้อเปรียบเทียบ SRAM & DRAM

Dynamic cell

- Simpler to build, smaller
- More dense
- Less expensive
- Needs refresh
- Larger memory units

Static

- Faster
- Cache

ROM (Read Only Memory) คุณสมบัติสำคัญคือ เก็บข้อมูลแบบถาวรและเป็น Nonvolatile นั่นคือไฟเลี้ยงหายไปแต่ข้อมูลก็ยังอยู่ โดยมักมีการประยุกต์ใช้ ROM ในงานต่างๆ เช่น

- Microprogramming
- Library subroutines
- Systems programs (BIOS)
- Function tables

สำหรับการสร้าง ROM นั้นก็เหมือนกับ IC และข้อมูลที่บรรจุใน ROM จะบรรจุตั้งแต่ในขั้นตอนการผลิตซึ่งการเกิดปัญหาขึ้นสองลักษณะคือ

- ขั้นตอนการนำข้อมูลบันทึกใน ROM ทำให้ต้นทุนสูงขึ้นถ้าผลิตน้อยชิ้น
- ข้อมูลที่บันทึกตอนผลิตนั้นต้องห้ามผิดไม่เช่นนั้นจะใช้งานไม่ได้เลย

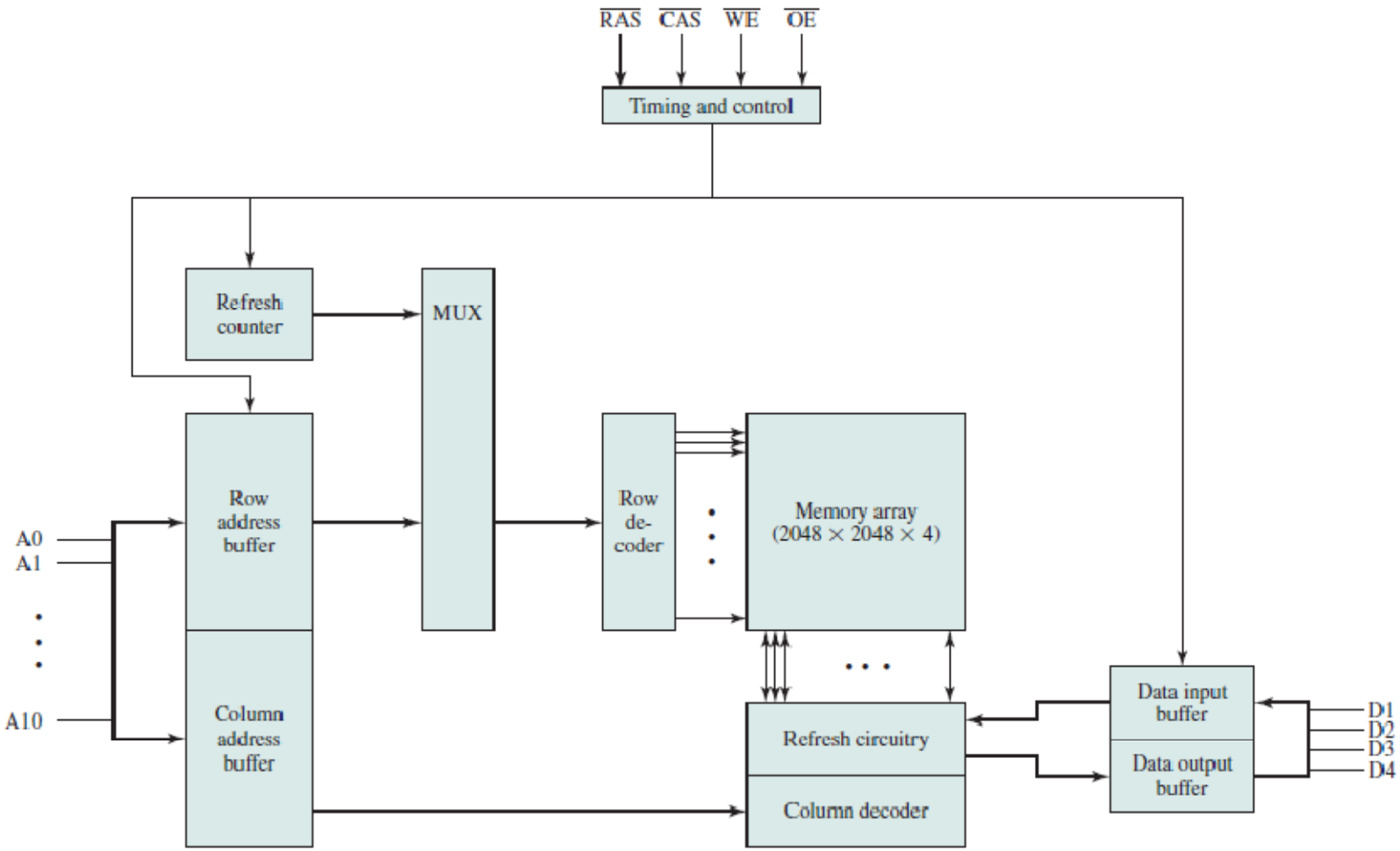
ชนิดของ ROM

- Programmable (once)
  - PROM
    - ต้องใช้เครื่องมือพิเศษในการโปรแกรมข้อมูล
- Read “mostly”
  - Erasable Programmable (EPROM)
    - ลบข้อมูลที่บันทึกด้วยแสดง UV
  - Electrically Erasable (EEPROM)
    - สามารถบันทึกและอ่านข้อมูลโดยใช้สัญญาณไฟฟ้า แต่การเขียนข้อมูลจะใช้เวลานานกว่าการอ่าน
  - Flash memory
    - สามารถลบข้อมูลทั้งหมดโดยใช้สัญญาณไฟฟ้า

Chip Logic

หน่วยความจำก็เป็นวงจรรวมประเภทหนึ่ง (Integrated Circuit) แต่จุดสำคัญประการหนึ่งในการออกแบบก็คือการกำหนดว่าจำนวนบิตข้อมูลที่ทำการอ่าน/เขียนในหนึ่งครั้งนั้นมีค่าเท่าใด ซึ่งจำนวนบิตข้อมูลที่เขียนหรืออ่านในแต่ละครั้งนั้นจะเป็นสิ่งที่กำหนด Organization ของตัว chip ซึ่งปกติแล้ว memory chip จะจัด memory cell ในแบบของ Array โดยจะระบุว่าเก็บข้อมูลได้กี่ WORD และแต่ละ WORD มีกี่ บิต เช่น Memory chip เบอร์หนึ่ง 16 M bit ถูกจัดโครงสร้างเป็น 1M WORDs และแต่ละ WORD คือ 16 บิต หรือ chip บางเบอร์ก็เป็นแบบ 1 bit-per-chip organization คือ อ่าน/เขียน ข้อมูลครั้งละ 1 บิต

และสิ่งที่พบใน DRAM ก็คือ มีการลดจำนวนขาสัญญาณ Address ด้วยการ Multiplex row address และ column address

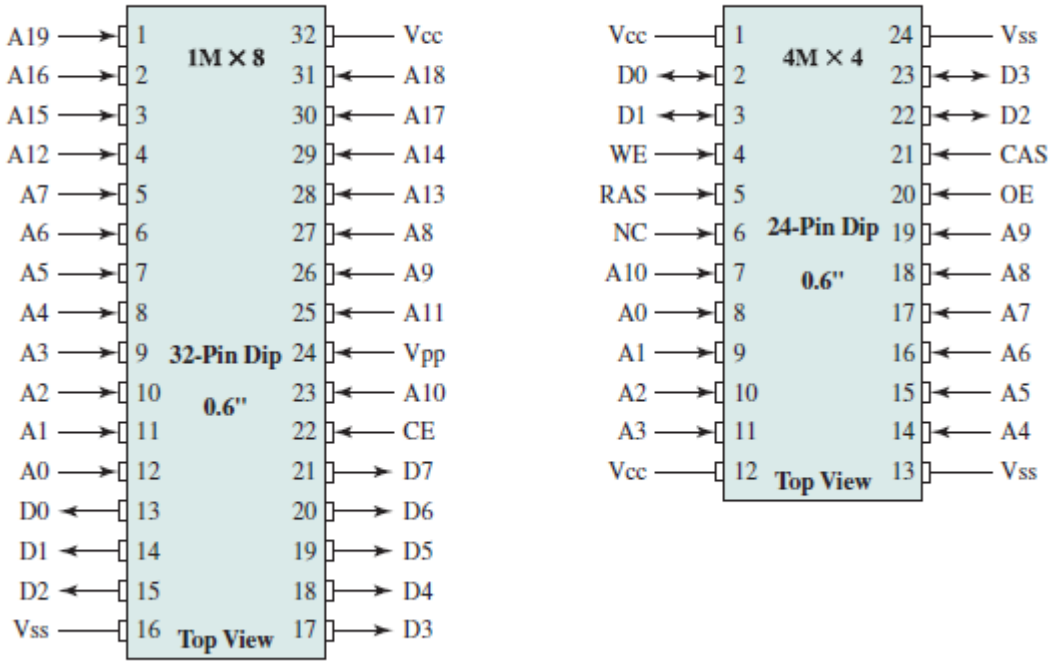


รูปที่ 5.5 16 M bit DRAM (4 M x 4)

Memory ในรูปที่ 5.5 นั้นเป็น array ขนาด 2048 x 2048 โดยแต่ละ element เก็บข้อมูลได้ 4 bit และสัญญาณแอดเดรสมี 11 สายสัญญาณนั่นคืออ้างแอดเดรสได้ 2048 ตำแหน่งสำหรับ 2048 column และ 2048 row ส่วนขณะใดแอดเดรส A0-A10 จะเป็น ROW หรือ Column นั้นขึ้นกับขาสัญญาณ RAS (Row Address Select) หรือ CAS (Column Address Select) ส่วนจะเป็นกระบวนการเขียนหรืออ่านกับ Memory นั้นก็ขึ้นกับสัญญาณ WE (Write Enable) และ OE (Output Enable) นอกจากนั้นในรูปที่ 5.5 นั้นยังมีส่วนของ Refreshing Circuit โดยการ Refresh จะต้องมีการ Disable RAM ออกจากระบบขณะจะทำกระบวนการ Refresh โดย Refresh Counter จะสร้าง Address ของ ROW ที่จะทำการ refresh จนครบทุก ROW ของหน่วยความจำ โดยจะมีการอ่านข้อมูลที่บันทึกไว้ออกมาและเขียนกลับเข้าไปใหม่ในตำแหน่งเดิม

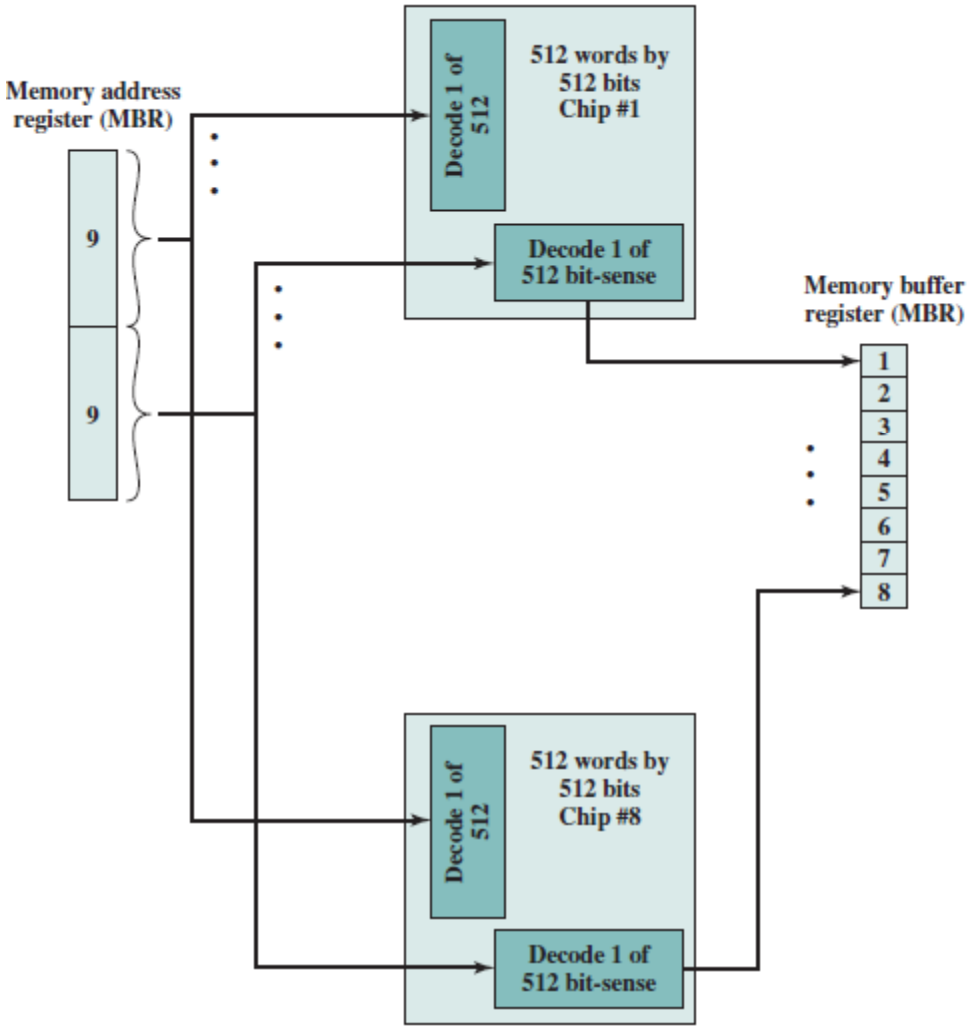
Chip Packaging รูปที่ 5.6 เป็นตัวอย่าง EPROM และ DRAM

- EPROM 1M x 8 bit คืออ้างแอดเดรสได้ 2^{20}ตำแหน่ง และ 1 word เก็บข้อมูลได้ 8 บิต
- DRAM 4M x 4 คืออ้างแอดเดรสได้ 4 x 2^{20} ตำแหน่ง และ 1 word เท่ากับ 4 บิต

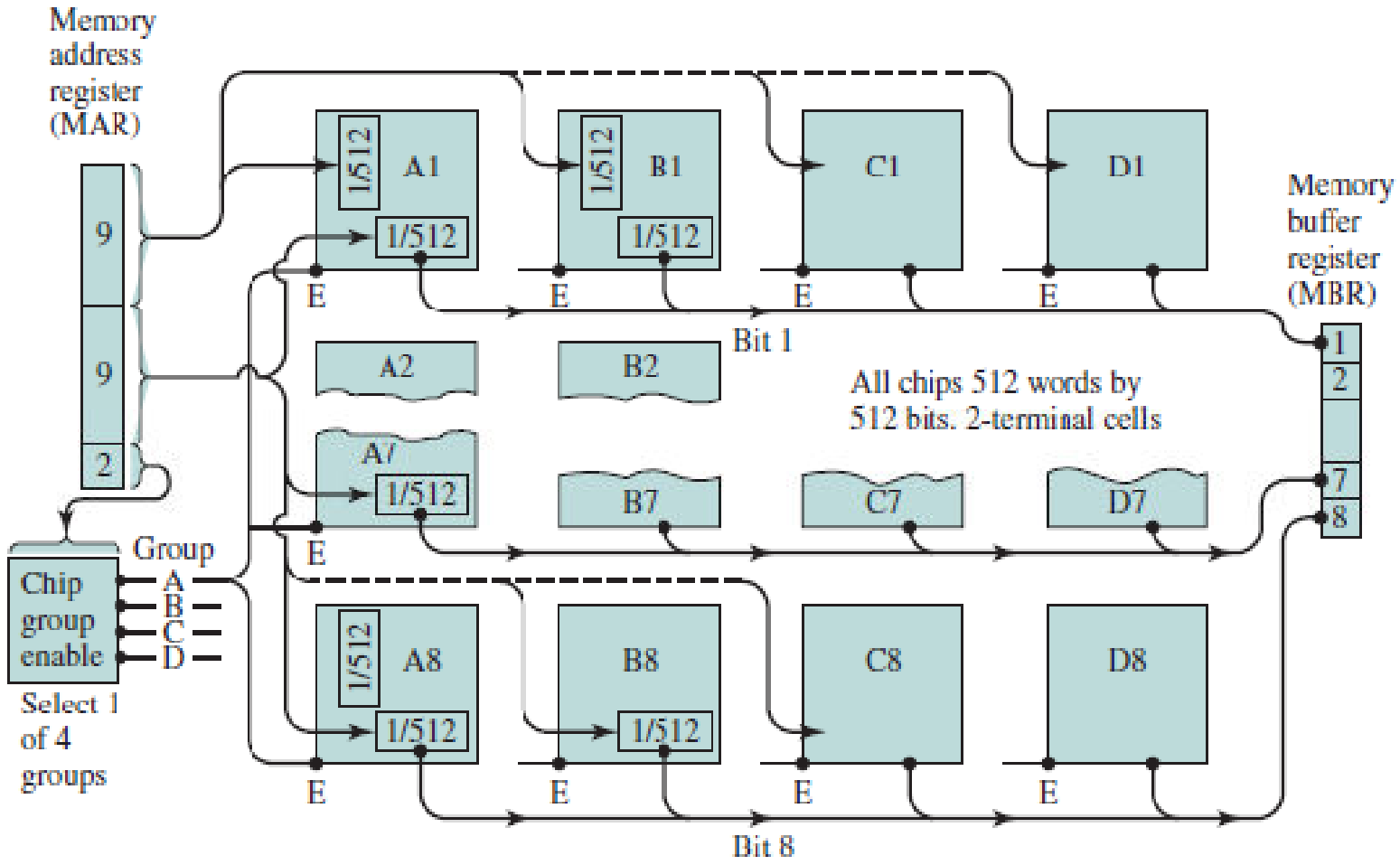


รูปที่ 5.6 Memory Package Pins and Signals

Module Organization



รูปที่ 5.7 256 KByte Memory organization

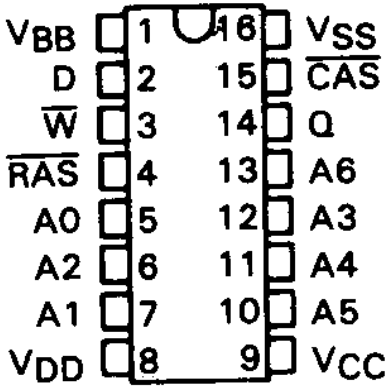


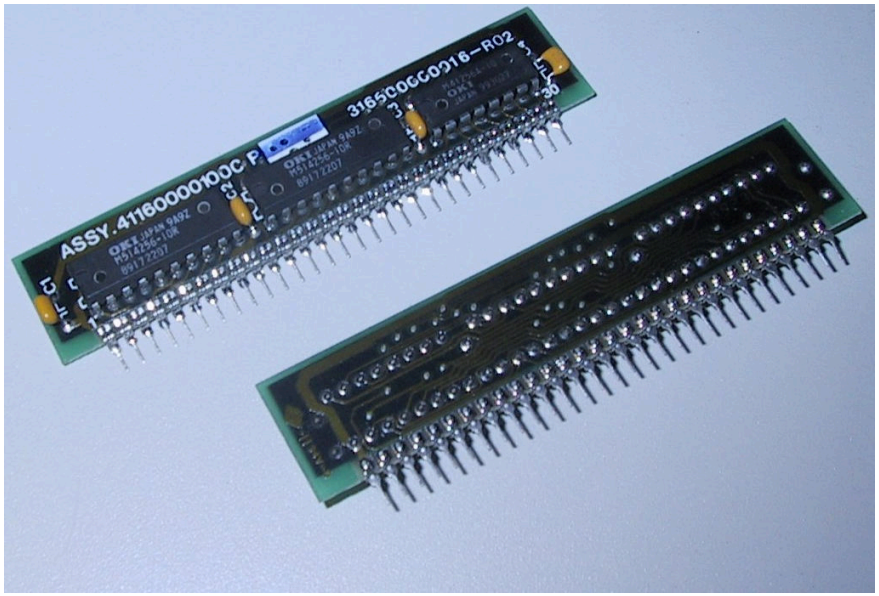
รูปที่ 5.8 1 MByte Memory organization

TMS 4116 : 16384 bit DRAM

- **16,384 X 1 Organization**
- **10% Tolerance on All Supplies**
- **All Inputs Including Clocks TTL Compatible**
- **Unlatched Three-State Fully TTL-Compatible Output**

DIP Package

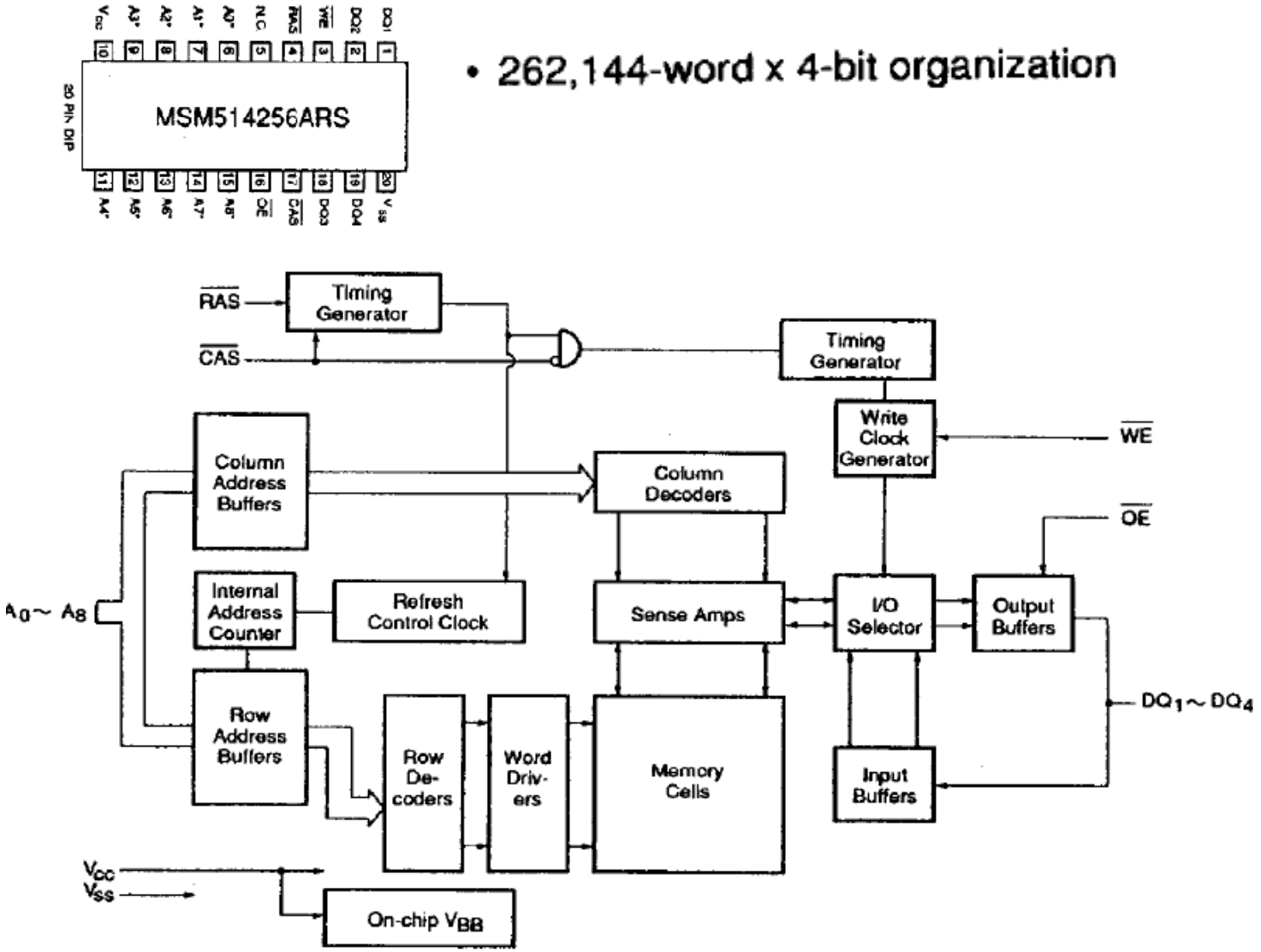




SIPP (Single Inline Pin Package)

ในตัวอย่าง DRAM นี้ก็สร้างจาก DRAM ที่เป็นแบบ DIP มาประกอบกัน ซึ่งหน่วยความจำแบบนี้เดิมทีใช้ใน main board คอมพิวเตอร์ที่มี CPU เป็น 386SX และไม่มีการใช้ในปัจจุบันแล้ว

รูปที่ 5.9 ตัวอย่าง DRAM : DIP และ SIPP package



รูปที่ 5.10 ตัวอย่าง DRAM 262,144 word x 4 bit organization

ความผิดพลาดที่เกิดขึ้นกับหน่วยความจำแบบ Semiconductor แบ่งออกเป็น 2 ลักษณะคือ

- **Hard Failure**

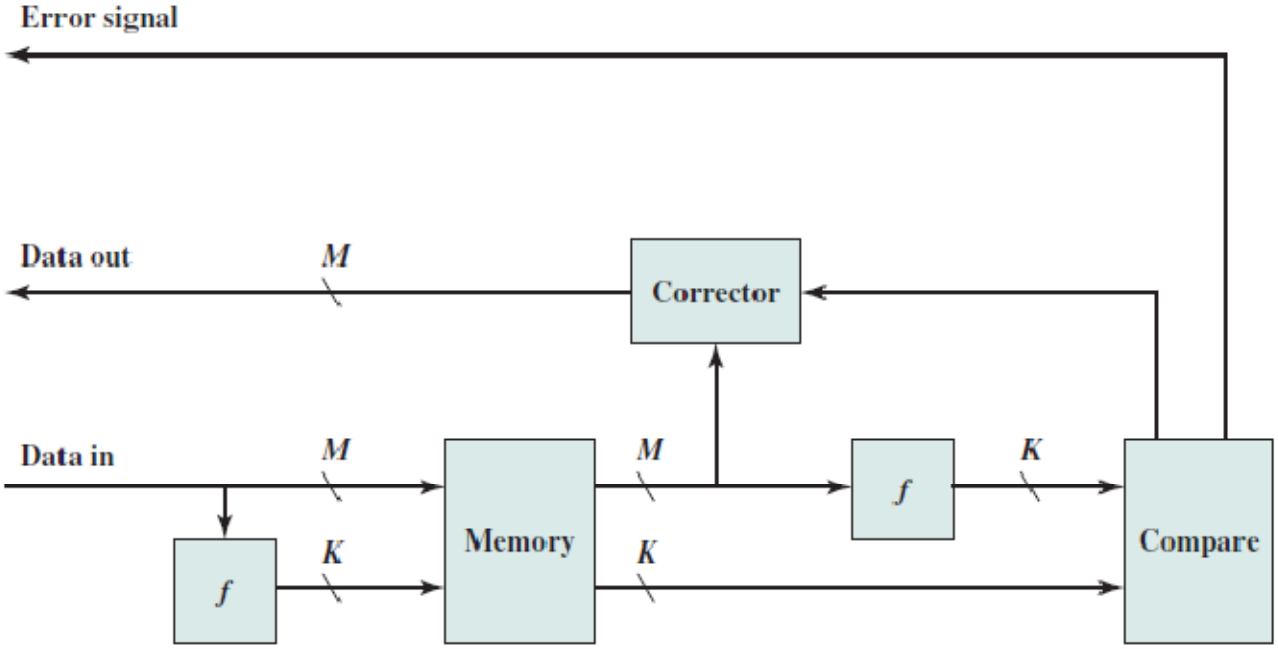
เป็นความเสียหายทางกายภาพกับ memory cell ที่ทำให้ไม่สามารถเก็บข้อมูลได้โดยข้อมูลอาจค้างสภาวะเป็นลอจิก 0 หรือ 1 หรืออาจเป็นค่าระหว่าง 0 หรือ 1 โดยสาเหตุของ Hard Failure อาจเกิดได้จากสภาพแวดล้อมในการใช้งานหรือเสียหายระหว่างการผลิตก็เป็นไปได้

- **Soft Error**

เป็นเหตุการณ์ที่เกิดขึ้นและทำให้ข้อมูลที่เก็บไว้ใน memory เกิดการเปลี่ยนสถานะโดยไม่ได้ทำให้หน่วยความจำเสียหายอย่างถาวร ซึ่งอาจเกิดจากปัญหาของแหล่งจ่ายไฟเลี้ยงและรังสีที่เรียกว่า อนุภาคแอลฟา ที่ส่งผลต่อข้อมูลที่เก็บใน memory cell

นอกจากนั้นในหน่วยความจำสมัยใหม่จะมีวงจรลอจิกทำการตรวจจับและแก้ไขข้อผิดพลาดของข้อมูลที่เกิดขึ้น (Error Detection and Correction)

ในรูปที่ 5.11 แสดงระบบหน่วยความจำที่มีการตรวจจับและแก้ไขความผิดพลาดของข้อมูลที่เกิดขึ้น โดยจากรูปข้อมูลที่ต้องการบันทึกมีขนาด M บิต ซึ่งมีการนำข้อมูลนี้มาผ่านฟังก์ชัน f เพื่อสร้างรหัส K บิต ที่ใช้ในการตรวจจับข้อผิดพลาด จากนั้นจึงเก็บทั้งข้อมูล M บิตและรหัส K บิตลงในหน่วยความจำ เมื่อมีการอ่านข้อมูลจากหน่วยความจำ รหัส K บิตนั้นจะถูกใช้ในการตรวจจับความผิดพลาดของข้อมูลและอาจจะใช้ในการแก้ไขข้อมูลให้ถูกต้องได้ทั้งนี้ขึ้นกับลักษณะของรหัส f บิตที่สร้างขึ้นจากข้อมูล M บิตนี้



รูปที่ 5.11 Error Correcting Code Function

จากรูปที่ 5.11 เมื่ออ่านข้อมูลจากหน่วยความจำ จะนำข้อมูล *M* บิตมาสร้างรหัส *K* บิตใหม่ด้วยฟังก์ชัน *f* จากนั้นจะนำรหัส *K* บิตที่เก็บไว้ในหน่วยความจำและรหัสที่สร้างใหม่มาเปรียบเทียบกัน หากค่าไม่เท่ากันแสดงว่าข้อมูล *M* บิตมีความผิดพลาด โดยผลของการเปรียบเทียบหรือตรวจจับความผิดพลาดอาจเกิดขึ้น 3 ลักษณะคือ

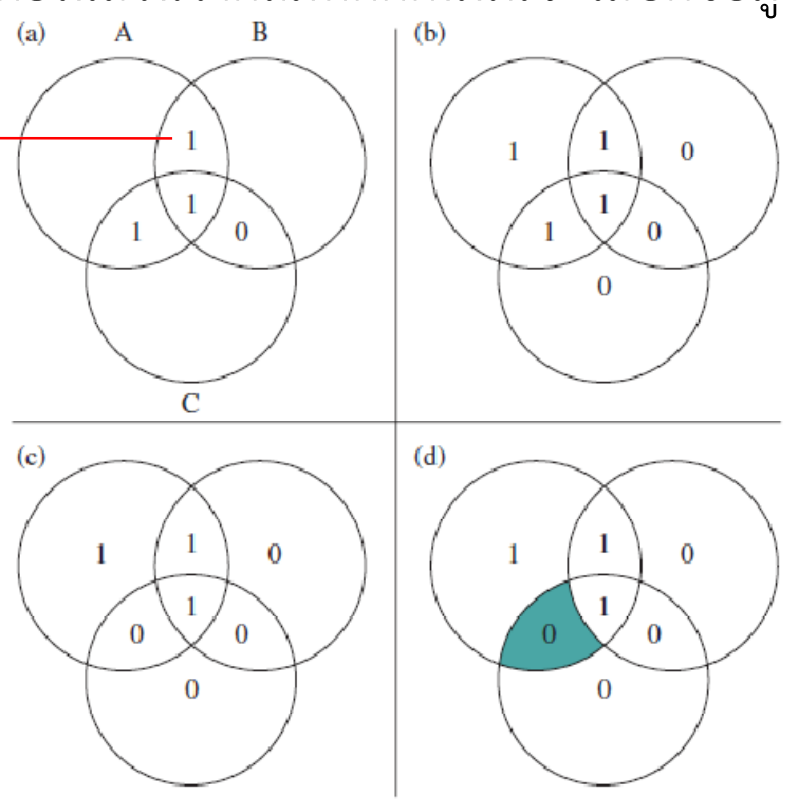
- ไม่เกิดข้อผิดพลาด บิตข้อมูลก็ถูกส่งออกไป
- ตรวจจับข้อผิดพลาดได้และมีการแก้ไขบิตข้อมูลที่ผิดพลาดก่อนที่จะส่งออก
- ตรวจจับข้อผิดพลาดได้แต่แก้ไขข้อมูลให้ถูกต้องไม่ได้ จากนั้นส่งสัญญาณแจ้ง Error

สำหรับรหัส *K* บิตที่สร้างขึ้นโดยนำข้อมูล *M* บิตมาผ่านฟังก์ชัน *f* นั้นจะเรียกว่า Error Correcting Code (รหัสในการตรวจแก้ไขความผิดพลาด) ซึ่งมีหลายแบบขึ้นกับจำนวนบิตข้อมูลที่ผิดพลาดในข้อมูล 1 word ที่สามารถตรวจจับและแก้ไขข้อผิดพลาดได้

Hamming Code

Hamming Code เป็นวิธีการตรวจจับและแก้ไขข้อผิดพลาดข้อมูลวิธีหนึ่งพัฒนาโดย Richard Hamming ซึ่งทำงานที่ Bell Laboratories จากรูปที่ 5.12 เป็นการใช้ Venn diagram สำหรับกรณีข้อมูล 4 บิต โดยในส่วนวงกลมที่ตัดกันนั้นจะใส่บิตข้อมูล

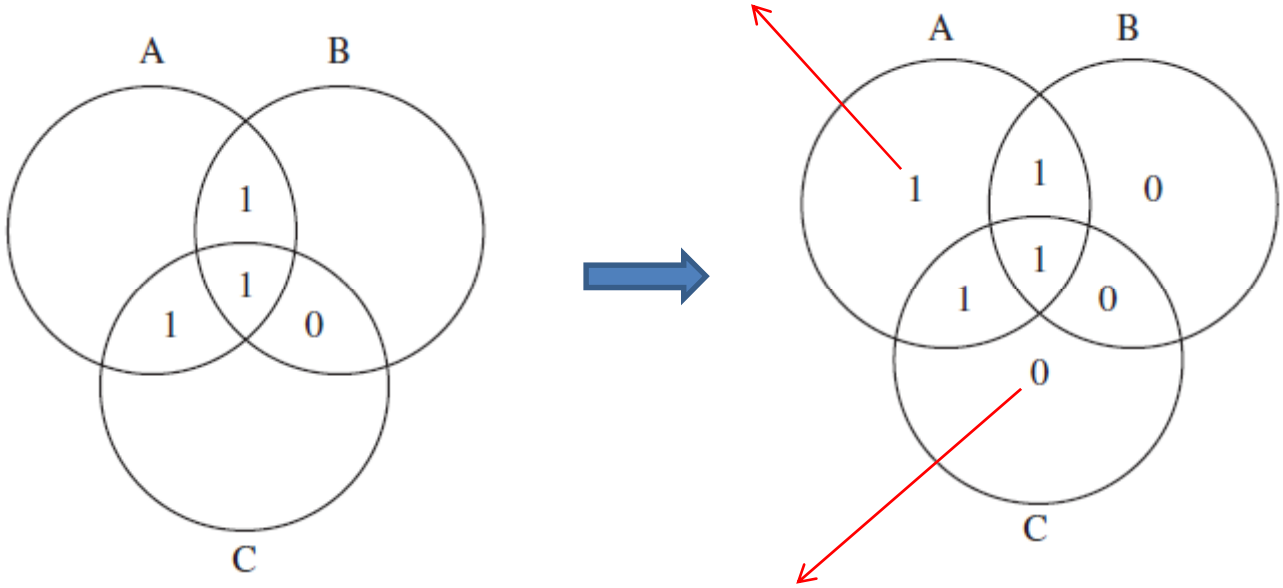
ข้อมูล 4 บิตคือ
1110



และส่วนที่เหลือของวงกกลมที่ไม่ได้ตัดกับวงกลมอื่นจะใส่ตัวเลขที่เรียกว่า Parity bit ซึ่งใช้ในการตรวจจับความผิดพลาดของข้อมูล โดยสถานะของพาริตี้บิตจะเป็นค่าที่เติมลงไปแล้วทำให้จำนวนบิตที่เป็น 1 ในวงกลมหนึ่งๆ เป็นจำนวนเลขคู่

รูปที่ 5.12 Hamming Error Correcting Code

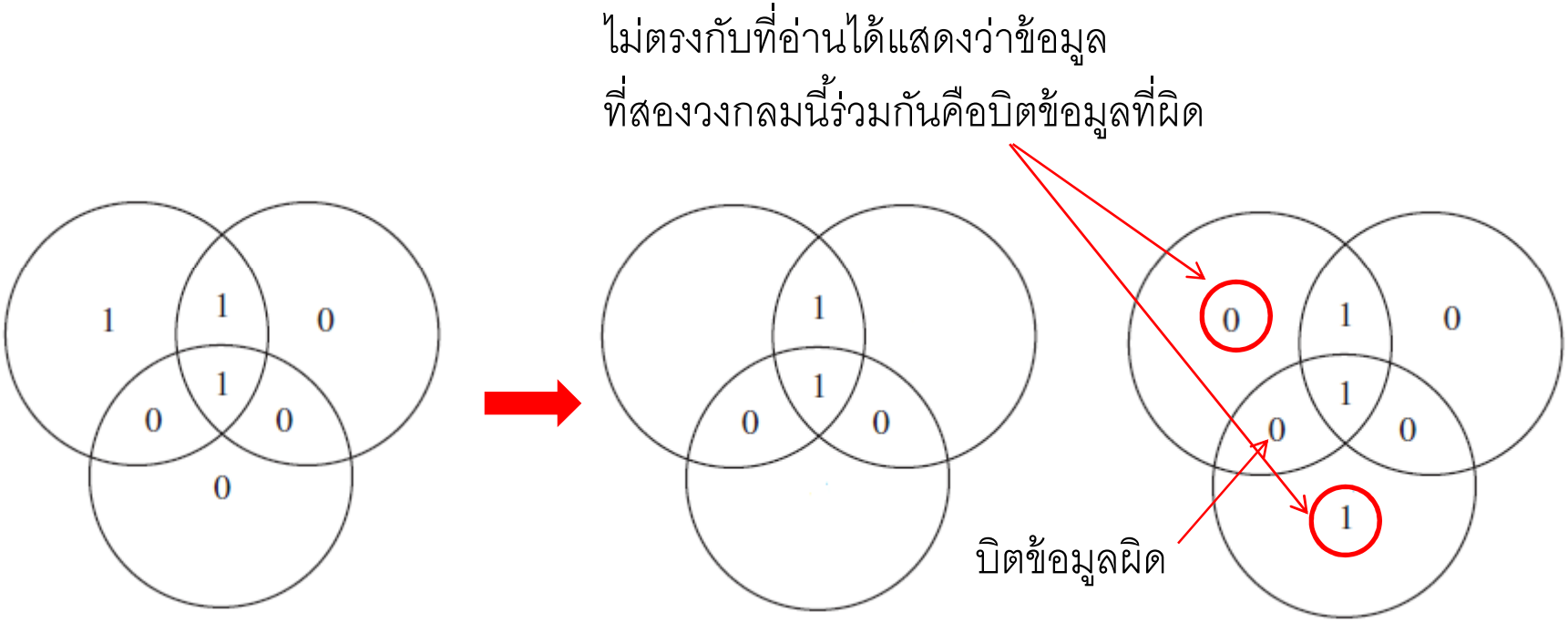
บิตข้อมูลที่เป็น ‘1’ มีจำนวนเป็นเลขคี่จึงให้ parity =1 เพื่อบิตที่เป็น 1 จะได้เป็นจำนวนคู่



Parity = 0 เพราะบิตที่เป็น ‘1’ เป็นจำนวนคู่อยู่แล้ว

รูปที่ 5.13 การเติมสถานะของ parity bit

จากรูปที่ 5.12 C) สมมติว่ามีการบันทึกข้อมูลและพาริตี้บิตดังรูป 5.14 a) เมื่อมีการอ่านข้อมูลจะตรวจสอบว่ามีความผิดพลาดหรือไม่ โดยการนำเฉพาะข้อมูลที่อ่านได้มาสร้างพาริตี้บิตใหม่



ข้อมูลและพาริตี้บิตที่อ่านได้

a)

นำข้อมูลมาสร้างพาริตี้บิตเอง

b)

รูปที่ 5.14 การเติมสถานะของ parity bit และตรวจสอบความผิดพลาด

จากแนวคิดของ Hamming code ในที่นี้จะสร้าง code ในการตรวจจับและแก้ไขข้อผิดพลาดของข้อมูลเมื่อเกิดความผิดพลาด 1 บิตในข้อมูล 8 บิต ซึ่งเริ่มจากหาความยาวของ code นั่นคือจำนวนบิตของ code ที่สร้างขึ้นจากข้อมูลเพื่อใช้ในการตรวจสอบความผิดพลาด จากรูปที่ 5.11 รหัส K บิตถูกสร้างขึ้นจากข้อมูล M บิตและถูกเก็บไว้ในหน่วยความจำ เมื่อมีการอ่านข้อมูลจะต้องมีการนำข้อมูลที่อ่านได้มาสร้างรหัส K บิตใหม่เพื่อเปรียบเทียบกับรหัสที่เก็บไว้ซึ่งเป็นการเปรียบแบบบิตต่อบิต ซึ่งจะใช้ Exclusive OR logic gate แบบสองอินพุตเพื่อตรวจสอบบิต โดยเอาต์พุตของการเปรียบเทียบจะเรียกว่า Syndrome ซึ่งถ้าบิตใดเป็น 0 ก็ไม่เกิดข้อผิดพลาดถ้าบิตใดของการเปรียบเทียบได้ผลเป็น 1 ก็แสดงว่าเกิดข้อผิดพลาด จากรหัส K บิตแสดงว่า Syndrome ที่ได้ก็ต้องมีขนาด K บิตด้วยและมีค่าเป็นเลขฐานสิบอยู่ระหว่าง 0 – 2^K-1 ซึ่งผลที่ได้ หาก syndrome = 0 แสดงว่าข้อมูลถูกต้อง ดังนั้นกรณี ที่เหลือคือเกิดความผิดพลาดและสิ่งที่ต้องการรู้ต่อมาก็คือ ตำแหน่งของบิตข้อมูลที่ผิดพลาด ซึ่งบิตที่ผิดอาจจะเป็น รหัส K บิตหรือข้อมูล M บิต ดังนั้นรูปแบบความผิดพลาด 2^K-1 แบบต้องมีค่ามากกว่าจำนวนบิต $M + K$

$$2^K\text{-}1 \geq M + K$$

ตารางที่ 5.2 Word Length with Error Correction

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

จากอสมการ $2^K - 1 \geq M + K$

สมมติว่าจำนวนบิตข้อมูล M =8 จะใช้รหัสทดสอบกี่บิต K = เท่าใด โดยเริ่มสุ่มจาก 3 และ 4 เพื่อตรวจสอบว่าอสมการข้างต้นจะเป็นจริงหรือไม่

$$2^3 - 1 < 8 + 3$$
$$2^4 - 1 > 8 + 4$$

พบว่า K = 4 จะทำให้อสมการเป็นจริงถ้าข้อมูล 8 บิตจะใช้รหัส 4 บิตส่วนกรณีอื่นแสดงในตาราง 5.2 ในส่วนของ Single-Error-Correction

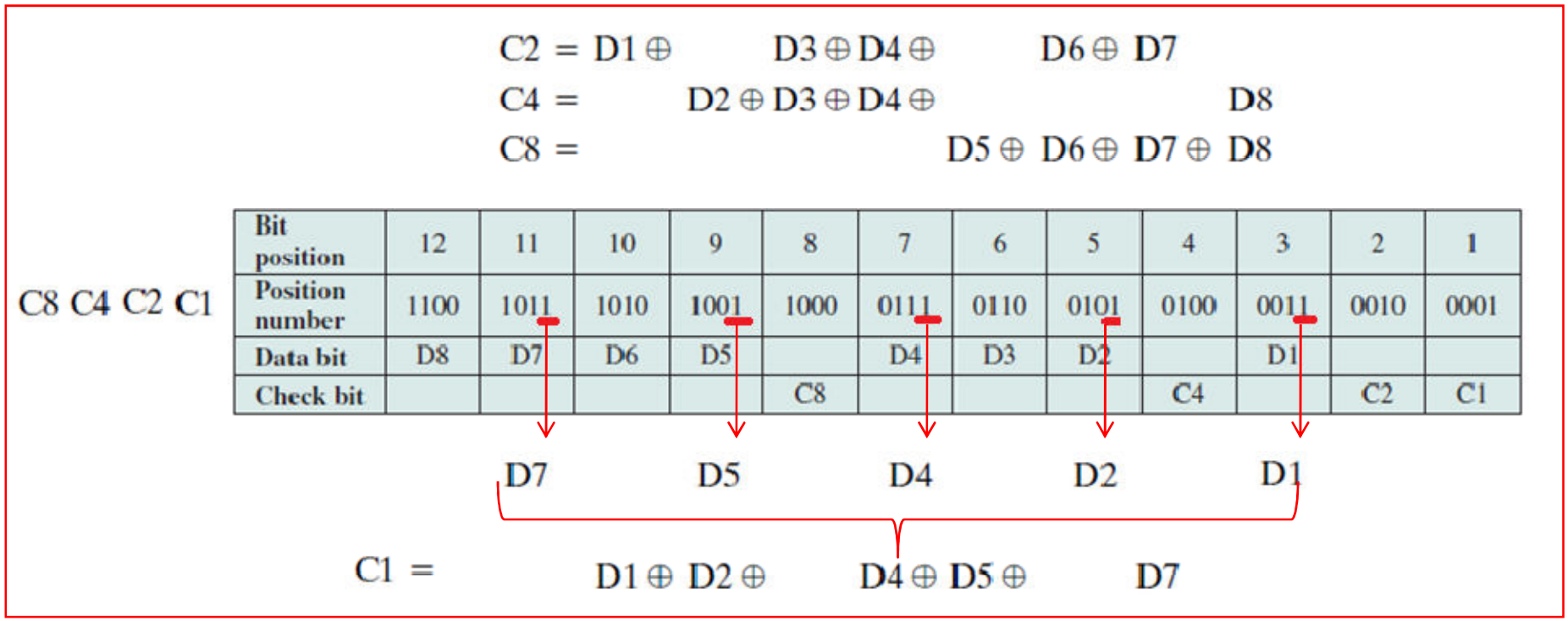
รหัสที่สร้างขึ้นมีคุณสมบัติดังนี้

- ถ้ารหัส M บิตที่สร้างขึ้นเป็น 0 ทุกบิต แสดงว่าไม่มีการตรวจจับข้อผิดพลาดได้
- ถ้ารหัสที่ได้มีบิตที่เป็น 1 เพียงบิตเดียว แสดงว่าจะเกิดความผิดพลาดที่ check bit บิตใดบิตหนึ่งใน 4 บิต
- ถ้ารหัสมีบิตที่เป็น 1 มากกว่า 1 บิต แสดงว่า ค่าของรหัส 4 บิตนั้นจะแสดงตำแหน่งบิตข้อมูลที่ผิดซึ่งมี 1 บิตที่ผิดดังนั้นเพียงกลับสถานะบิตนั้นก็เป็นการแก้ไขข้อมูลให้ถูกต้องแล้วนั่นเอง จากนั้นจัดวางตำแหน่งบิตข้อมูล 8 บิตและรหัสที่สร้างใหม่ดังตารางที่ 5.3 และรหัสที่สร้างมีสมการดังนี้

$$C1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7$$
$$C2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7$$
$$C4 = D2 \oplus D3 \oplus D4 \oplus D8$$
$$C8 = D5 \oplus D6 \oplus D7 \oplus D8$$

ตารางที่ 5.3

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1



รูปที่ 5.15 แสดงที่มาสมการคำนวณหาค่าข้อมูลของรหัส C1

สมการคำนวณ C1 ได้จากดูค่าบิตข้อมูล D1-D8 ว่าค่าบิตข้อมูลใดที่รหัส C1 มีค่าเป็น 1 ให้เอาข้อมูลตำแหน่งนั้นมาเป็นข้อมูลในการคำนวณค่ารหัส C1 ดังตัวอย่างในรูปที่ 5.15 ซึ่งรหัสที่พิจารณาอยู่นี้จะใช้ตรวจจับเฉพาะเกิดบิตผิดเพียง 1 บิตเท่านั้นเรียกว่า **Single Error Correction** (SEC)

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1

จากตารางข้างต้น สมมติ ค่าข้อมูล บิต D8-D1 คือ **00111001** จะคำนวณ check bit ได้ดังนี้

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

สมมติว่า D3 ผิด โดยมีการอ่านค่าได้เท่ากับ 1 แทนที่จะเป็น 0 เหมือนตอนที่บันทึก คำนวณ Check bit ใหม่ดังนี้

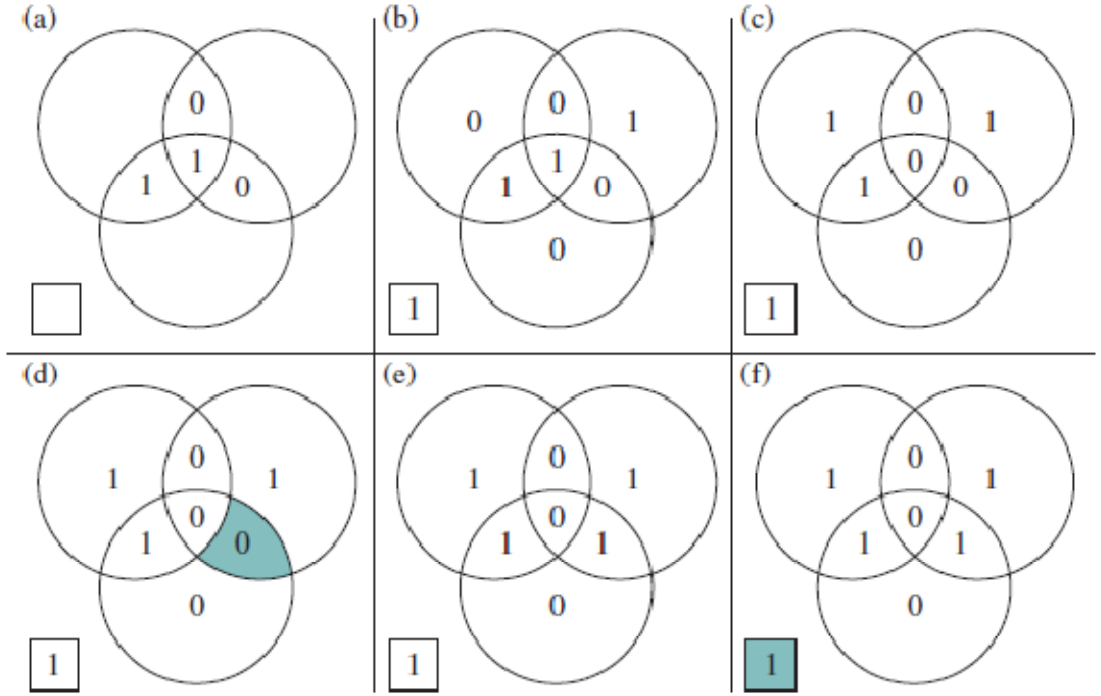
$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

จากนั้นนำ check bit ที่อ่านได้จากหน่วยความจำและ check bit ที่คำนวณใหม่มาเปรียบเทียบกันบิตต่อบิตด้วยการนำมากระทำ Exclusive OR ดังนี้

	C8	C4	C2	C1
	0	1	1	1
⊕	0	0	0	1
	0	1	1	0

ซึ่งได้ค่าเท่ากับ 0110 ซึ่งเป็นการบอกว่าบิตที่มีค่าประจำตำแหน่ง 0110 คือ D3 นั้นคือบิตที่เกิดข้อมูลผิด ซึ่งที่อ่านได้คือ 1 ดังนั้นที่ถูกควรเป็น 0 นั่นเอง

Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check bit					0				0		0	1



รูปที่ 5.16 Hamming SEC-DEC Code

จากวิธีการตรวจจับและแก้ไขความผิดพลาดของข้อมูลที่กล่าวมานั้นจะใช้ได้เฉพาะกรณีมีข้อมูลและบิตตรวจสอบมีความผิดพลาดเกิดขึ้นเพียงบิตเดียวเท่านั้นถ้าจะตรวจจับว่าเกิดผิดพลาดพร้อมกันสองบิตจะต้องเพิ่มบิตตรวจสอบอีก 1 บิต แต่อย่างไรก็ดีจะตรวจจับว่าเกิดความผิดพลาดของบิตที่เก็บไว้ว่าเกิดผิดพลาด 2 บิตได้แต่ไม่สามารถระบุบิตที่ผิดได้ วิธีการนี้คือ Hamming Single Error Correcting Double Error Detecting Code