Organisasi dan Arsitektur Komputer CII2A3

Cara Kerja Memori Utama

Tim Dosen COA

Fakultas Informatika
Universitas Telkom

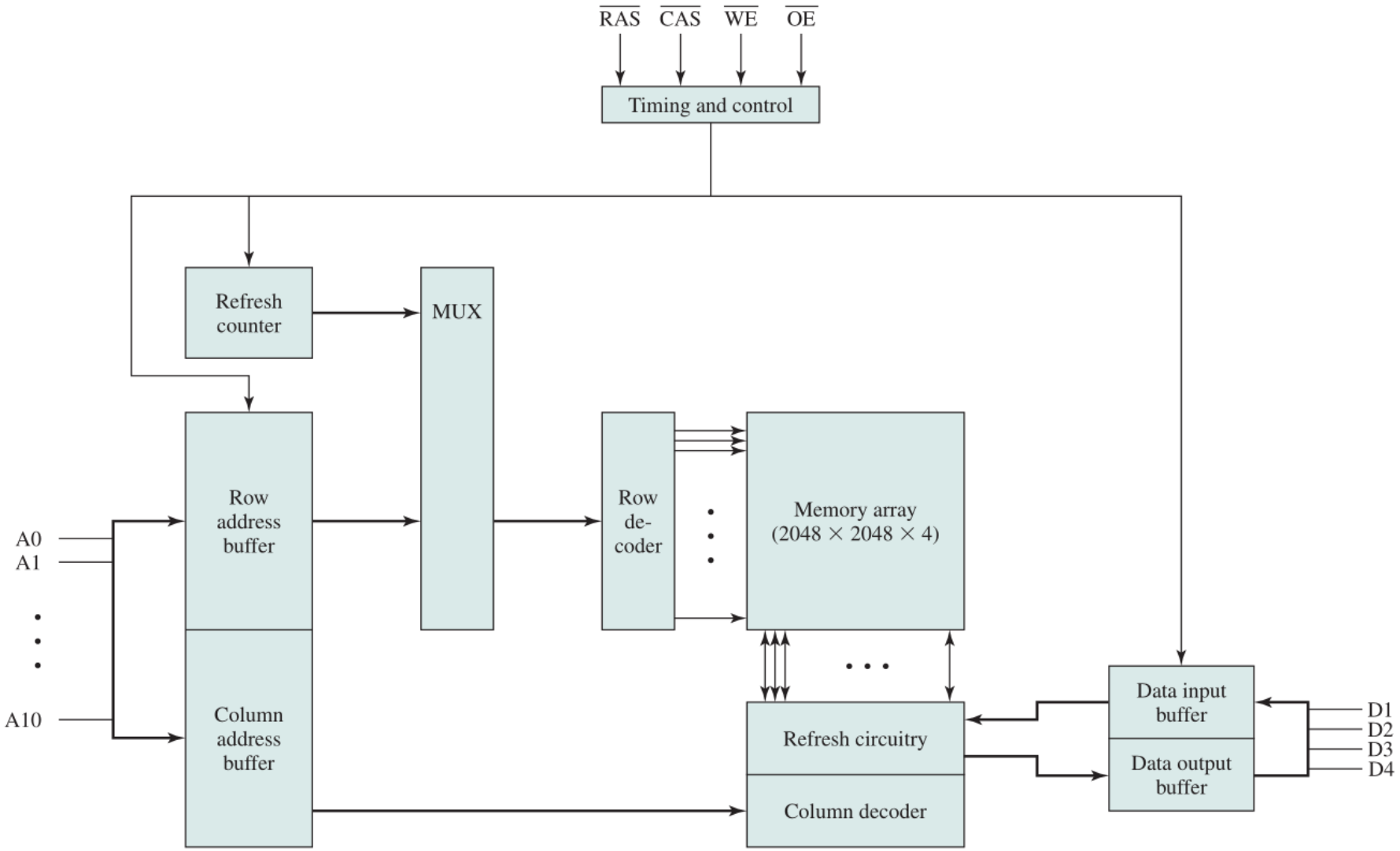
Rencana Studi

Fakultas Informatika
School of Computing
Telkom University

Rencana Studi		Rincian Nilai Kuis																Rincian Nilai Tugas										Rincian Nilai Hasil Proyek										Bobot Tiap CLO (%)			
Pertemuan Ke-	Materi	CLO 1			CLO 2					CLO 3				CLO 4				CLO 1			CLO 2						CLO	CLO 3						CLO 4									
		Kuis (Kognitif) (20 %)																Tugas Partisipatif (35%)										Hasil Proyek (45%)													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1a	1b	1c	2a	2b	2c	2d	2e	3a	3b	1a	1b	2a	1c	2b	2c	3a	3b	4a	3c	4b	3d	4c	1	2	3	4
1	Sistem komputer	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
2	Input/Output	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	14	--	--	--
3	Sistem Bus	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
4	Organisasi memori	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
5	Cara kerja memori utama (RAM)	-	-	-		1	-	-	-	-	-	-	-	-	-	-		-	-	-		4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
6	Memori sekunder	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	--	26	--	--
7	Cara kerja *cache memory* (bag-1)	-	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
8	Cara kerja cache memory (bag-2)	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
9	Arsitektur SAP-1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-				
10	Arsitektur SAP-2	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	2	1	-	-	-	-	-	-	-	-	-	--	--	30	--
11	Arsitektur SAP-3	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	-	-	-	7	2	-	-	-	-	-	-	-				
12	Instruksi *Extended* dan *Indirect*	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-				
13	Arsitektur MIPS	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-				
14	Instruksi MIPS	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	1	-	-	-	-	--	--	--	30
15	Assembly MIPS (bag-1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	2	-	-				
16	Assembly MIPS (bag-2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	7				

Part 1: Bagaimana cara kerja model memori dengan lebar bus alamat minimal?

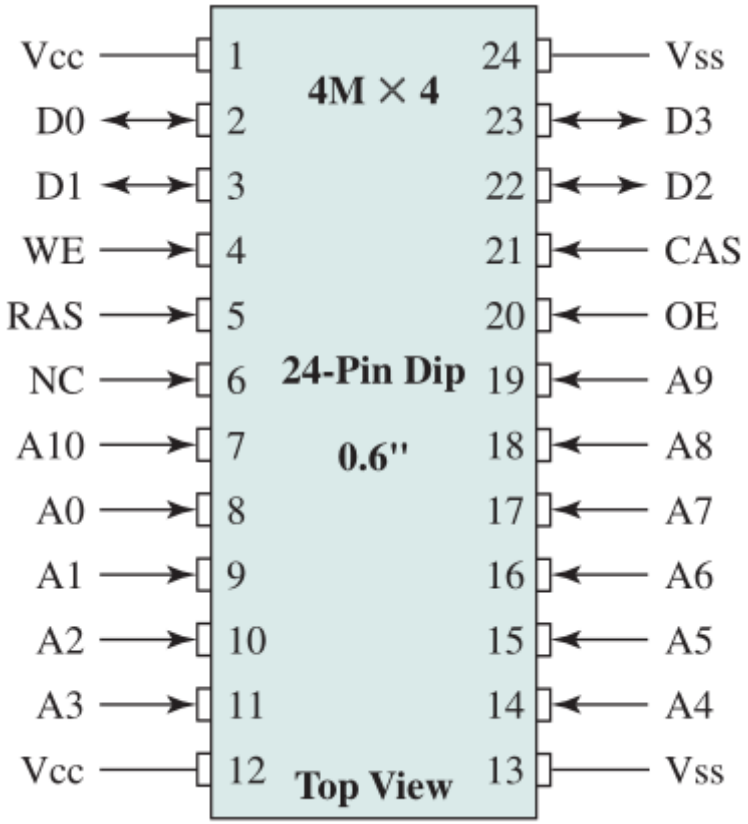
Chip Logic DRAM 16 Mbit (4Mx4) (1)



Chip Logic DRAM 16 Mbit (4Mx4) (2)

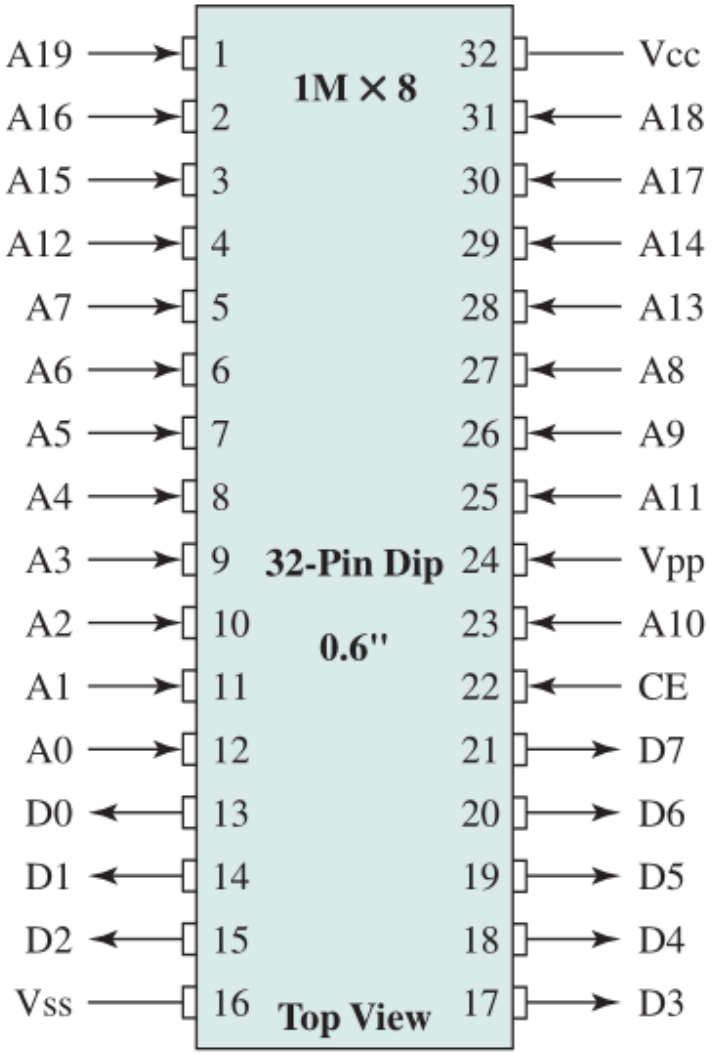
- Dalam satu saat dapat dibaca/ditulis data 4 bit (1 bit tiap memori *plane*)
- Terdiri dari 4 buah *array* berukuran 2048x2048
- Alamat tiap sel ditentukan oleh alamat baris dan kolom
- Pin alamat baris paralel dengan alamat kolom → hemat jumlah pin → digunakan $\overline{\text{RAS}}$ dan $\overline{\text{CAS}}$
- Terdapat 4 pin untuk data *in (write)* dan *out (read)* secara paralel → digunakan $\overline{\text{WE}}$ dan $\overline{\text{OE}}$
- *Refresh* dilakukan dengan membaca data dan menuliskannya kembali

Contoh *Chip Packaging* (1)



- DRAM 16 Mbit:
  - Alamat = 11 pin = A0-A10
  - Digunakan oleh baris dan kolom secara bergantian → setara dengan 2x11 pin = 22 pin = 2^{22} alamat = 4 M alamat
  - 1 alamat = 4 bit → kapasitas = 4 M x 4 bit = 16 bit
  - CAS untuk mengaktifkan kolom
  - RAS untuk mengaktifkan kolom
  - D0-D3 = data keluar (4 jalur)
  - OE = *Output Enable* (memungkinkan data dibaca)
  - WE = *Write Enable* = memungkinkan data disimpan ke memori

Contoh *Chip Packaging* (2)

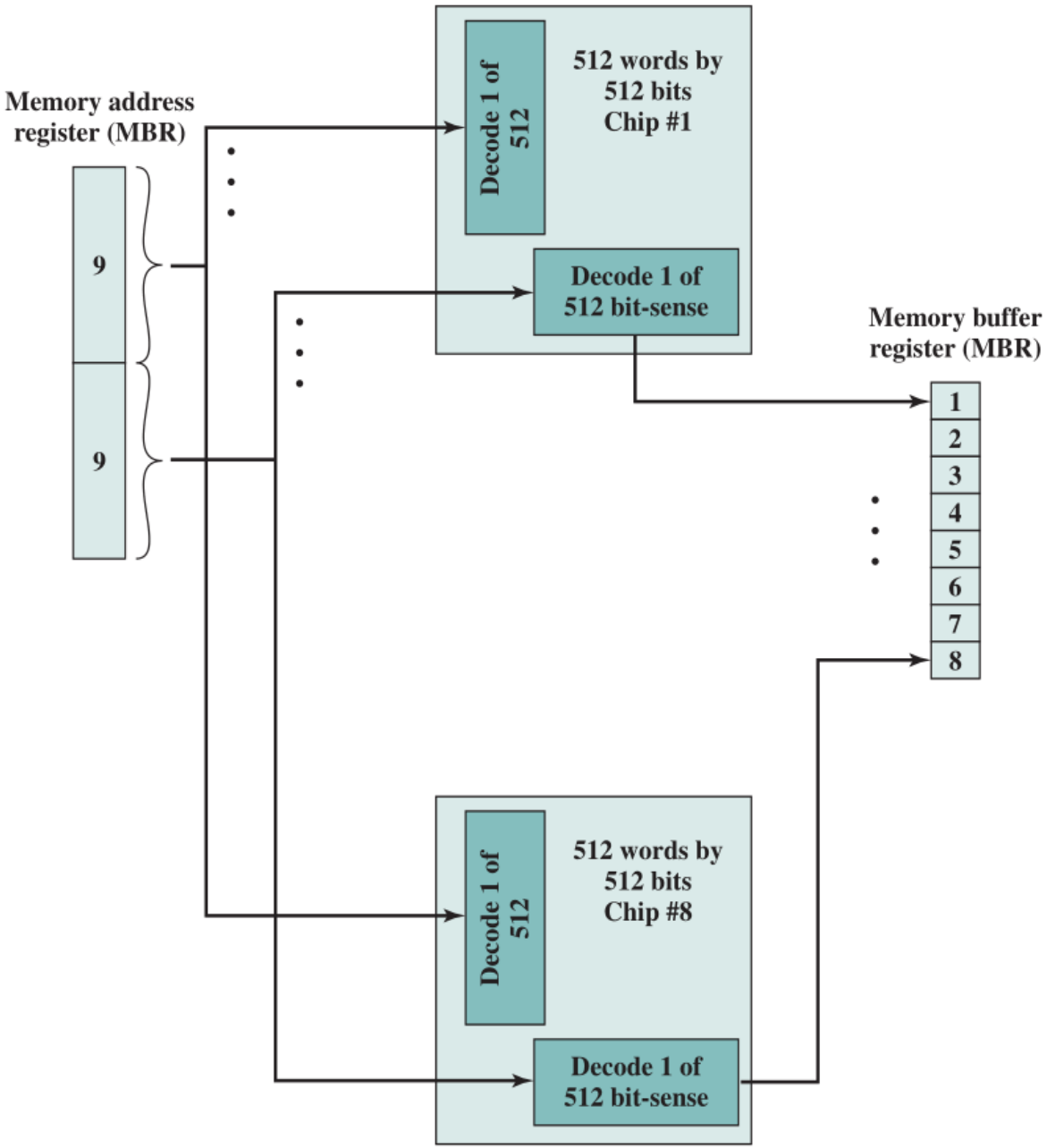


- EPROM 8 Mbit:
  - Alamat = 20 bit (pin) = A0-A19 = 1 M alamat
  - 1 alamat = 8 bit → kapasitas = 8 x 1 Mbit = 8 Mbit
  - *Chip Enable (CE)* untuk memilih *chip* yang aktif bila digunakan lebih dari satu chip
  - D0-D7 = data keluar (8 jalur)
  - Vss = *ground*
  - Vcc = tegangan positif

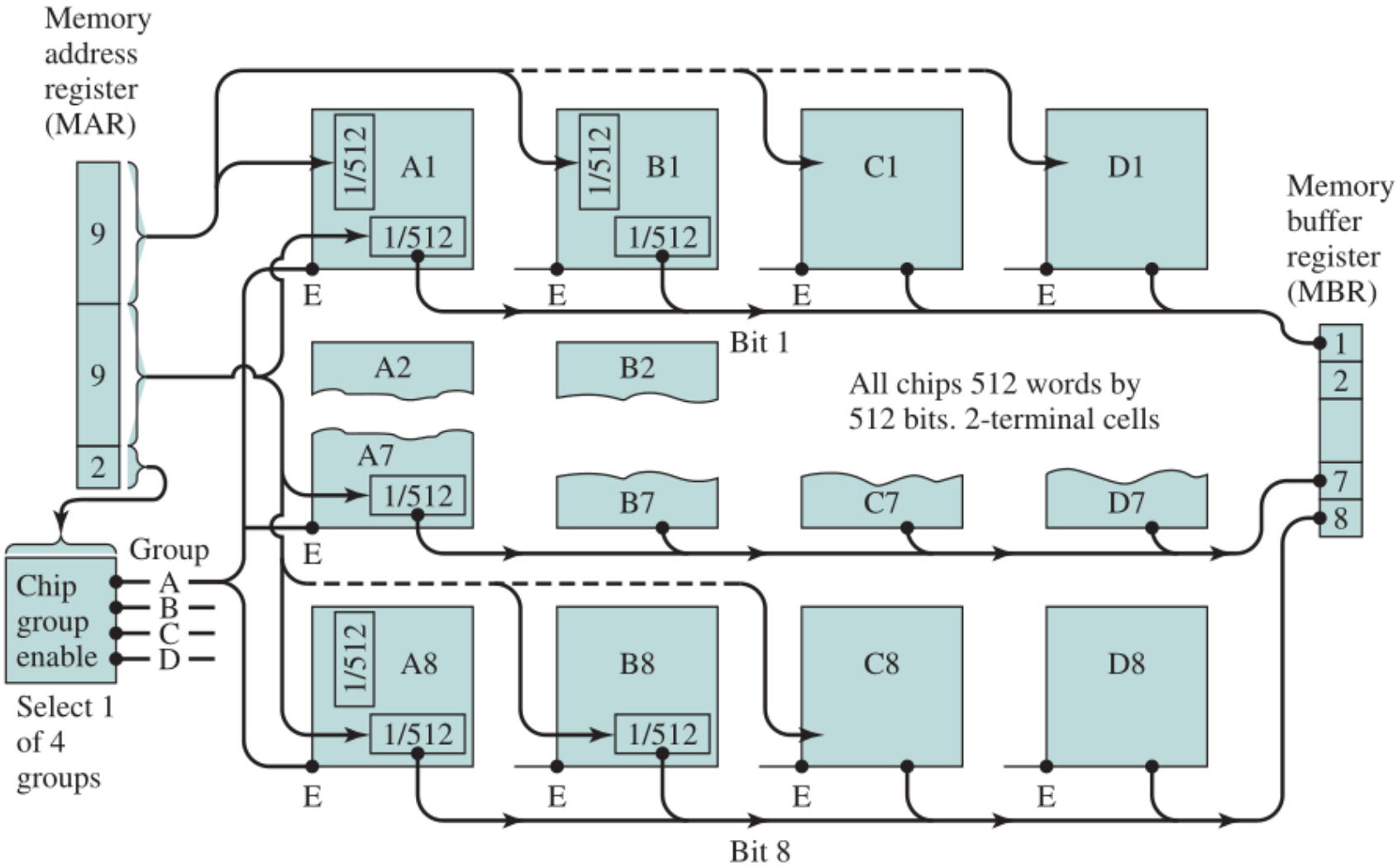
Part 2: Bagaimana cara memperbesar kapasitas memori?

Organisasi Modul (1)

- Organisasi memori 256 kB
  - Terdiri dari 8 memori *plane* berukuran 512x512 bit
  - Alamat terdiri dari 18 bit (9 baris, 9 kolom)
  - Data terdiri dari 8 bit (1 byte)



Organisasi Modul (2)



Organisasi Modul (3)

- Organisasi memori 1 Mbyte
  - Terdiri dari 4 buah bank memori masing-masing berukuran 256 kB (4 kolom A, B, C, D)
  - Alamat terdiri dari 20 bit (2 bit MSB digunakan untuk memilih group chip A, B, C, atau D)

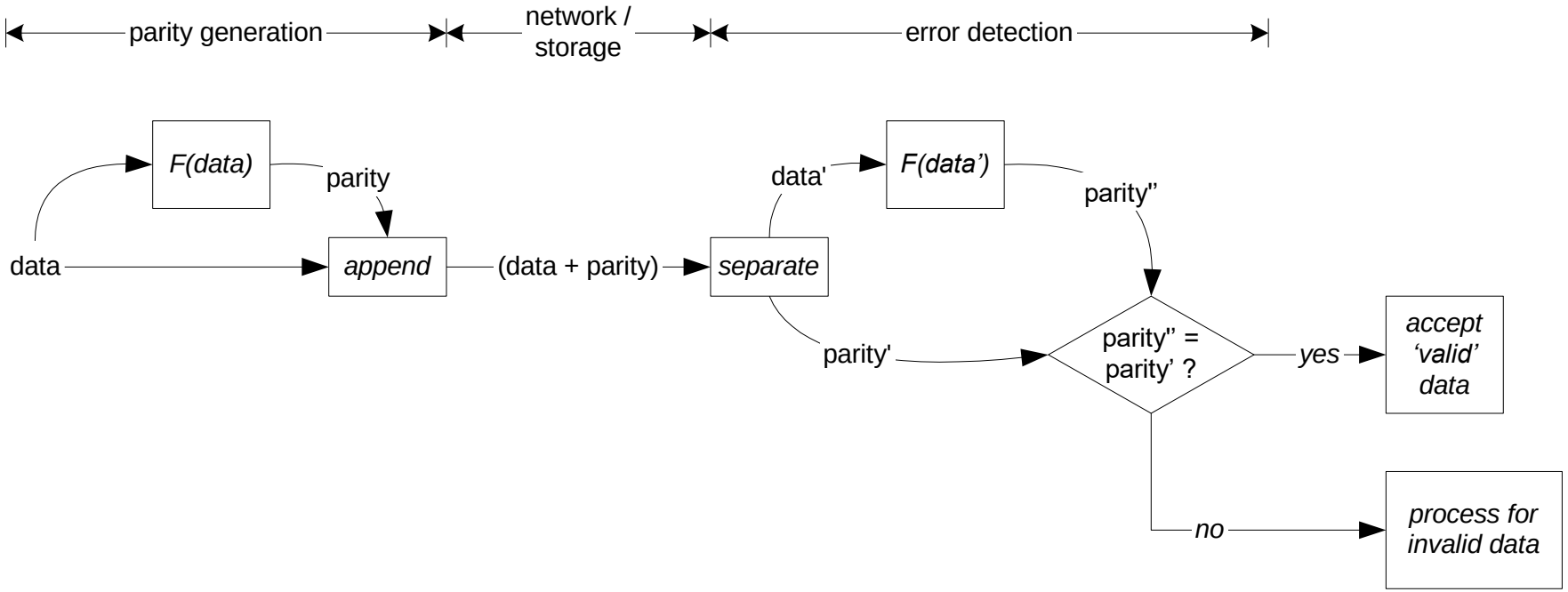
Part 3: Bagaimana cara kerja Hamming code?

Penanganan Kesalahan Dengan Paritas (1)

- Mendeteksi kesalahan data pada level bit
- Bit paritas:
  - bit ekstra yang ditambahkan pada suatu unit data terkecil
  - digunakan dalam proses pengecekan kebenaran data ketika data akan disimpan atau dikirim
  - dihasilkan oleh fungsi generator paritas
- Jenis paritas:
  - Paritas genap *(even)*:
    - Menambahkan sebuah bit sehingga total bit '1' suatu word berjumlah genap
  - Paritas ganjil *(odd)*:
    - Menambahkan sebuah bit sehingga total bit '1' suatu word berjumlah ganjil
- Dapat mendeteksi kesalahan bit berjumlah ganjil, lokasi bit tidak bisa diketahui

Penanganan Kesalahan Dengan Paritas (2)

Fakultas Informatika
School of Computing
Telkom University



Contoh:

Data = **1001001,** jenis paritas = paritas genap

Jika bit yang dibaca **10010011** → data dianggap valid

Jika bit yang dibaca **10110011** → data dianggap tidak valid

Jika bit yang dibaca **10110010** → data dianggap valid !!

Penanganan Kesalahan Dengan *Hamming Code Single Bit*

- Biasa digunakan dalam konteks *Error Control* (deteksi maupun koreksi)
- *Codeword* = bit-bit data + bit paritas/kontrol
- *Hamming distance*: jumlah perbedaan bit dari dua buah *codeword*
  - *Hamming distance* n + 1 → Dapat mendeteksi n *error*
  - *Hamming distance* 2n + 1 → Dapat me-*recover* n *error*

- Contoh *codeword* dengan 7 bit informasi dan 1 bit paritas genap:

0000000	0
0000001	1
0000010	1
0000011	0

} Antar *codeword* terdapat 2 bit berbeda

Hamming distance-nya = 2 → hanya dapat mendeteksi 1 bit *error*

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (1)

- Blok data sebanyak **M** digit dikodekan menjadi **N** digit (N>M)
  - Ditulis dengan notasi (N,M)
    - Misal c*odeword* terdiri dari 7 bit data + 4 bit *check* → (11,7)
  - Jumlah bit *codeword* harus memenuhi persamaan:

    $2^K - 1 \geq M+K$

    K = jumlah bit kontrol; M = jumlah bit data
  - Posisi (letak) bit-bit K ditentukan dengan rumus 2^x; x = 0, 1, 2, 3, ...
  - Posisi bit dimulai dari posisi ke-1, **bukan ke-0 !!!**
  - M/N = *code rate* atau *code efficiency*
  - 1 - M/N = *redundancy*
- Dapat mendeteksi kesalahan sebanyak 2 bit
- Dapat menentukan posisi bit yang *error* jika terjadi kesalahan sebanyak 1 bit
- *Recovery* dilakukan dengan meng-*inverse* bit pada posisi yang salah

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (2)

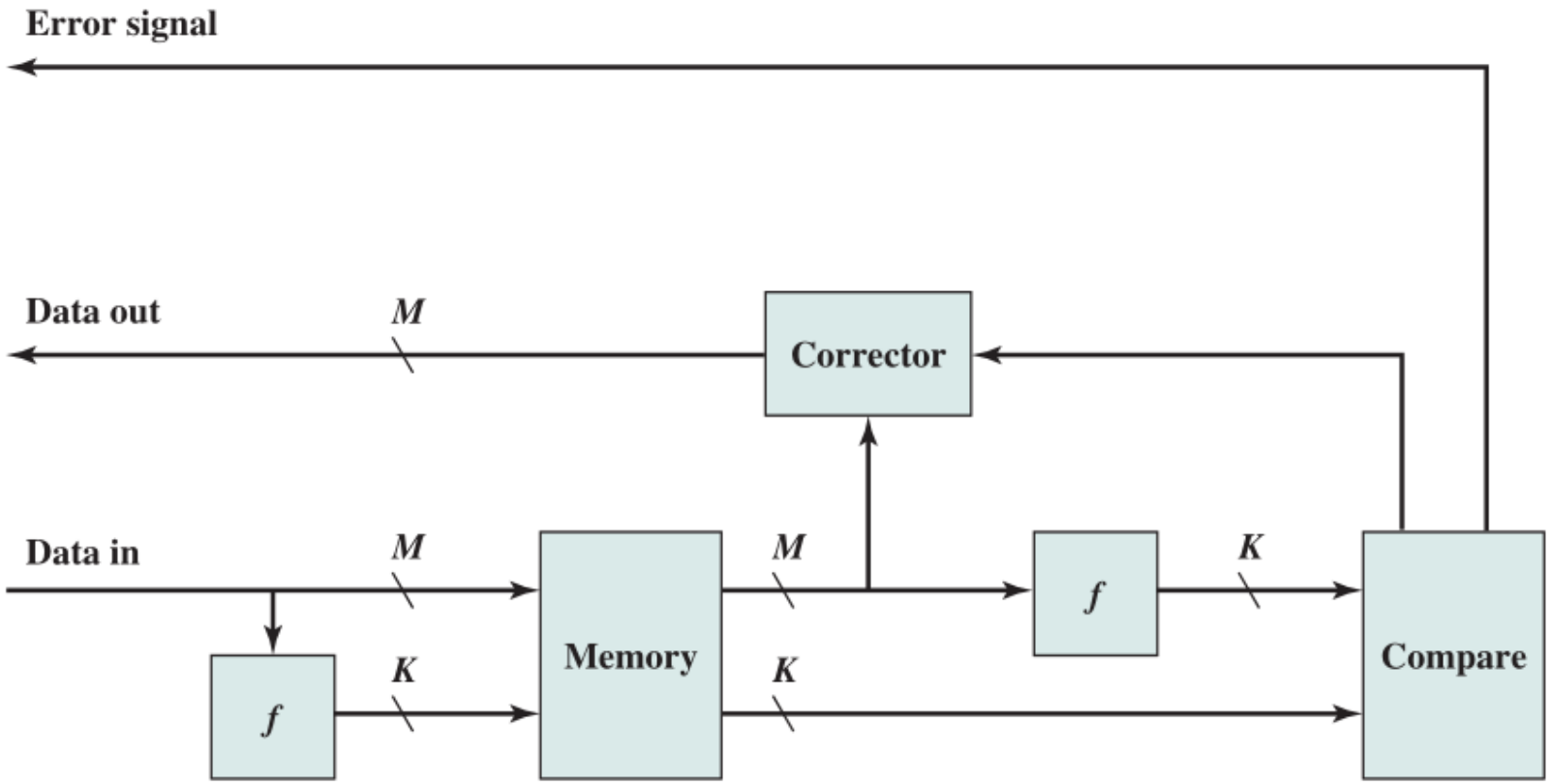


Figure 5.7 Error-Correcting Code Function

- Cara menentukan bit-bit C:
  - **Cara 1:**

    C1 = hasil XOR **semua data** yang terletak pada nomor posisi yang bit ke-1 dari kanan nilainya 1 (misal posisi 3→001**1** (D1), posisi 5→010**1** (D2), dst

    C2 = hasil XOR **semua data** yang terletak pada nomor posisi yang bit ke-2 dari kanan nilainya 1 (misal posisi 3→00**1**1 (D1), posisi 6→01**1**0 (D3), dst

    C4 = hasil XOR **semua data** yang terletak pada nomor posisi yang bit ke-3 dari kanan nilainya 1 (misal posisi 5→0**1**01 (D2), posisi 6→0**1**10 (D3), dst

    C8 = hasil XOR **semua data** yang terletak pada nomor posisi yang bit ke-4 dari kanan nilainya 1 (misal posisi 9→**1**001 (D5), posisi 10→**1**010 (D6), dst

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (4)

- Contoh bit-bit data: 00111001 (8 bit)

  Jumlah bit *codeword*: $2^K - 1 \geq M+K$

  Jika K = 3 → $2^3 - 1 \geq 8 + 3$ (salah)

  Jika K = 4 → $2^4 - 1 \geq 8 + 4$ (ok)

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	0	1	1		1	0	0		1		

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

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

Figure 5.9 Layout of Data Bits and Check Bits

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (5)

Fakultas Informatika
School of Computing
Telkom University

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	0	1	1		1	0	0		1		

(3) (5) (7) (9) (11)

C1 = 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 = **1**

(3) (6) (7) (10) (11)

C2 = 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 = **1**

(5) (6) (7) (12)

C4 = 1 ⊕ 0 ⊕ 1 ⊕ 1 = **1**

(9) (10) (11) (12)

C8 = 1 ⊕ 1 ⊕ 0 ⊕ 0 = **0**

Hasil = 001101001111

Misal ada error pada bit 3 (posisi ke-6) dari 0 berubah menjadi 1:

(3) (5) (7) (9) (11)

C1 = 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 = **1**

(3) (6) (7) (10) (11)

C2 = 1 ⊕ **1** ⊕ 1 ⊕ 1 ⊕ 0 = **0**

(5) (6) (7) (12)

C4 = 1 ⊕ **1** ⊕ 1 ⊕ 1 = **0**

(9) (10) (11) (12)

C8 = 1 ⊕ 1 ⊕ 0 ⊕ 0 = **0**

Hasil = 001101000101

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (6)

Ketika dilakukan pemeriksaan:

Hasilnya bukan 0000 yang berarti ada error.

Bit hasil pemeriksaan menunjukkan posisi bit yang salah.

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

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
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

error

Figure 5.10 Check Bit Calculation

Hamming Code Multi Bit (7)

Ketika dilakukan pemeriksaan:

Hasilnya bukan 0000 yang berarti ada error.

Bit hasil pemeriksaan menunjukkan posisi bit yang salah.

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

Table 5.2 Increase in 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

- **Cara 2:**
  - Lakukan penjumlahan modulo 2 (biner) **semua nomor posisi bit data yang bernilai 1 (tidak termasuk bit kontrol)**

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	0	1	1	**0**	1	0	0	**1**	1	**1**	**1**

```
 (3)  0011
 (7)  0111
 (9)  1001
(10)  1010 +
      ----
      0111
```

- Hasil:

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	0	1	1	**0**	1	0	0	**1**	1	**1**	**1**

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (9)

Fakultas Informatika
School of Computing
Telkom University

- Pendeteksian kesalahan *(decoder):*
  - Jumlahkan (modulo 2) semua posisi bit yang bernilai 1 (**termasuk bit *check***)
    - Jika hasilnya 0 → tidak terjadi kesalahan
    - Jika hasilnya ≠ 0 → hasil penjumlahan merupakan posisi bit yang salah

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	0	1	1	**0**	1	0	0	**1**	1	**1**	**1**

```
(1)  0001
(2)  0010
(3)  0011
(4)  0100
(7)  0111
(9)  1001
(10) 1010 +
     0000
```
→ *tidak error*

- Contoh jika terjadi *error* sebanyak **1 bit** (bit ke-11):

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	**1**	1	1	**0**	1	0	0	**1**	1	**1**	**1**

```
(1)  0001
(2)  0010
(3)  0011
(4)  0100
(7)  0111
(9)  1001
(10) 1010
(11) 1011 +
     1011
```
→ *terjadi error pada bit ke-11 (1011)*

- *Recovery*: *invert* bit ke-11

Penanganan Kesalahan Dengan *Hamming Code Multi Bit* (10)

Fakultas Informatika
School of Computing
Telkom University

- Contoh jika terjadi *error* sebanyak **2 bit** (bit ke-11 dan bit ke-1):

Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	**1**	1	1	**0**	1	0	0	**1**	1	**1**	**0**

```
 (2)  0010
 (3)  0011
 (4)  0100
 (7)  0111
 (9)  1001
(10)  1010
(11)  1011  +
      1010  →
```

kesalahan terdeteksi, posisi bit yang salah tidak diketahui

- Contoh jika terjadi *error* sebanyak **3 bit** (bit ke-1, bit ke-10, dan bit ke-11):

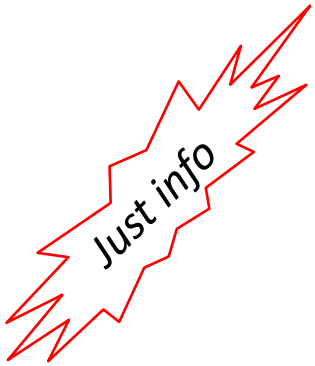
Posisi	12	11	10	9	8	7	6	5	4	3	2	1
Data	0	**1**	**0**	1	**0**	1	0	0	**1**	1	**1**	**0**

```
 (2)  0010
 (3)  0011
 (4)  0100
 (7)  0111
 (9)  1001
(11)  1011  +
      0000  →
```

*kesalahan **tidak** terdeteksi, dianggap valid*

Penerapan Pendeteksi Kesalahan Bit

- Digunakan pada aplikasi yang punya batasan *single-bit error*, misalnya: *error correcting semiconductor memory system*
- Jarang digunakan pada komunikasi data atau jaringan komputer, dimana probabilitas terbesar *error* yang terjadi adalah *burst error*



Part 4: Bagaimana teknologi perkembangan ROM?

"Read" Only Memory (ROM)

- *Tidak termasuk memori utama*
- *Teknologi:*
  - *→ROM (Read Only Memory)*
    - *→ PROM (Programmable ROM)*
      - *→ EPROM (Erasable PROM)*
        - *→ EEPROM (Electrically EPROM)*
          - *→ Flash Memory*

Read Only Memory (ROM)

- Data bersifat permanen (*non-volatile*)
- Nilai data dihasilkan dari kombinasi rangkaian logika di dalamnya
- Penulisan data dilakukan pada saat pembuatan chip
- Data tidak dapat dihapus
- Aplikasi:
  - *Library subroutine* untuk fungsi-fungsi yang sering digunakan
  - Program sistem
  - Tabel fungsi, dll
  - *BIOS (Basic Input Output System)*
- Kapan data bisa hilang???

Programmable ROM (PROM)

- Data ditulis sesudah *chip* dibuat
- Digunakan bila jumlahnya tidak terlalu besar (bukan *mass product*) dan memori memuat data khusus
- *Non-volatile*
- Hanya dapat ditulisi satu kali
- Penulisan data dilakukan secara elektrik dengan PROM programmer
  - Sel memori "dibakar" (*burning-in*) dengan arus listrik yang cukup besar, lokasi bit akan terbakar dan menunjukkan sebuah nilai (0 atau 1)
- Data tidak dapat dihapus
- Lebih fleksibel daripada ROM
- Contoh aplikasi: BIOS

Erasable PROM (EPROM)

- Data ditulis sesudah chip dibuat
- *Non-volatile*
- Proses baca dan tulis secara elektrik
- Data dapat dihapus dan ditulisi berulang-ulang dengan radiasi *ultraviolet*
  - Sinar tersebut melewati celah di kumpulan *chip*
- Data dihapus dalam satuan chip (semua data dihapus)
- Lebih mahal daripada PROM
- Contoh aplikasi:
  - BIOS

Electrically EPROM (EEPROM)

- Data ditulis sesudah chip dibuat
- *Non-volatile*
- Data dapat ditulis dan dihapus kapan saja secara elektrik, data tidak perlu dihapus terlebih dahulu
- Data dihapus dalam satuan byte
- Lebih mahal daripada EPROM
- Kurang rapat dibanding EPROM
- Contoh aplikasi:
  - BIOS

Flash Memory

- *Non-volatile*
- Biaya dan fungsionalitas berada diantara EPROM dan EEPROM
- Data dihapus dalam satuan blok memori
- Lebih rapat dibanding EEPROM
- Contoh aplikasi:
  - BIOS → isinya lebih mudah diubah, termasuk oleh virus

Jenis Memori Semikonduktor

Table 5.1 Semiconductor Memory Types

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		

Referensi

- [SCH85] Schneider, Michael G. 1985. "*The Principle of Computer Organization".* 1st edition. John Wiley & Sons. Canada.
- [STA19] Stalling, William. 2019. "*Computer Organization and Architecture: Designing for Performance*". 11th edition. Prentice Hall.
- https://en.wikipedia.org/wiki/Magnetic-core_memory#/media/File:Magnetic-core_memory,_at_angle.jpg