

Lecture 07

Error Correction

Tongwei Ren

Software Institute, Nanjing University

Sep. 29, 2018



Review

- Memory management
- Exchange and overlap technique
 - Partitioning and paging
- Virtual memory
 - Page based VM, segment based VM, segment and page based VM



Error

- A semiconductor memory system is subject to errors
- Type
 - Hard failure
 - A permanent physical defect so that the memory cell or cells become stuck at 0 or 1 or switch erratically between 0 and 1
 - Caused by harsh environmental abuse and manufacturing defects
 - Soft failure
 - A random, nondestructive event that alters the contents of one or more memory cells without damaging the memory
 - Caused by power supply problems or alpha particles



Error Correction

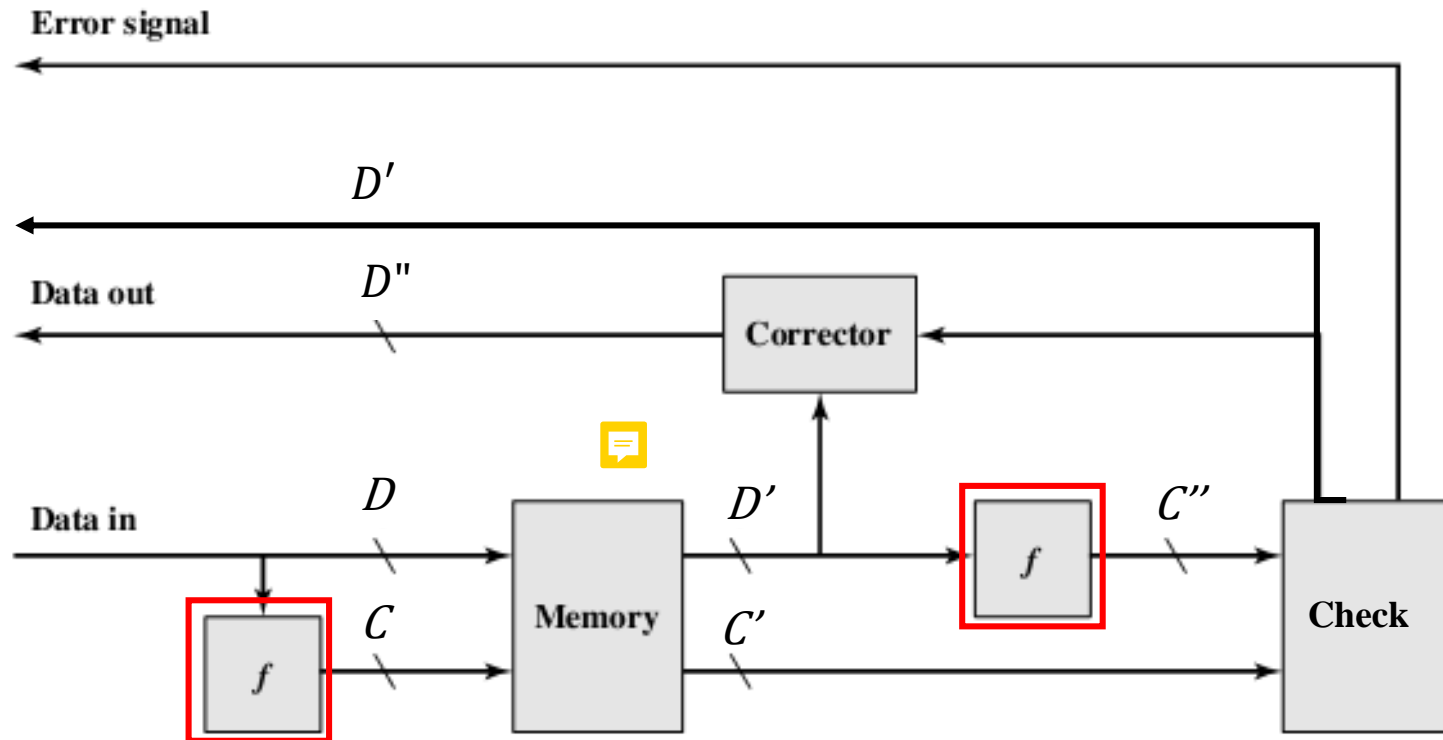


- Basic idea
 - Add some bits to store additional information for correction
- Process
 - Data in: produce a K bits code C on the M bits data D with a function f
 - Data out: produce a new K bits code C'' on the data D' with function f , and compare with the obtained K bits code C'
 - No error detected: send D'
 - An error detected which can be corrected: correct it and send D''
 - An error detected which cannot be corrected: report




Error Correction (cont.)

- Process (cont.)



Parity Checking

- Basic idea
 - Add one bit to denote whether the number of 1 in data is odd or even
- Procedure
 - Assume the data is $D = D_M \dots D_2 D_1$
 - Data in 
 - Odd parity: $C = D_M \oplus \dots \oplus D_2 \oplus D_1 \oplus 1$
 - Even parity: $C = D_M \oplus \dots \oplus D_2 \oplus D_1$
 - Data out
 - Odd parity: $C'' = D'_M \oplus \dots \oplus D'_2 \oplus D'_1 \oplus 1$
 - Even parity: $C'' = D'_M \oplus \dots \oplus D'_2 \oplus D'_1$

Parity Checking (cont.)

- Procedure (cont.)
 - Check: $S = C'' \oplus C'$
 - $S = 1$: the number of error bits is odd
 - $S = 0$: correct or the number of error bits is even
- Advantage
 - Low cost
- Disadvantage
 - Cannot find the errors when the number of error bits is even
 - Cannot be used for correction
- Suitable to check one byte data



Hamming Code

- Basic idea
 - Divide the bits into several groups, and use parity check code on each group for error correction
- Procedure
 - Divide the M bits into K groups
 - Data in: produce one bit for each group, and get a K bits code
 - Data out: produce one bit for each group, and get a new K bits code
 - Check: produce K bits code for fetched data, take the exclusive-OR of each bit of the produced code and fetched code, and produce K bits **syndrome word**



Hamming Code (cont.)

- Check code length
 - Assume at most one bit error occurs
 - Possible errors
 - One bit error in data: M
 - One bit error in check code: K
 - No error: 1
- Length of check code
$$2^K \geq M + K + 1$$

	Single-Error Correction	
Data Bits	Check Bits	% Increase
8	4	50
16	5	31.25
32	6	18.75
64	7	10.94
128	8	6.25
256	9	3.52

Hamming Code (cont.)

- Syndrome word value
 - Map each value of syndrome word to one possible situation
 - Rule
 - All 0s: no error has been detected
 - One bit is 1: an error has occurred in one of the check bits, and no correction is needed
 - More than one bit is 1: the numerical value of the syndrome indicates the position of the data bit in error, and invert this data bit for correction



Hamming Code (cont.)

- Data bits division
 - Assume the 8 bits data is $D = D_8 \dots D_2 D_1$, the 4 bits check code is $C = C_4 C_3 C_2 C_1$
 - Relationship of data bit / check code and syndrome word

syndrome word	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C4				C3		C2	C1

- Data bits division

$$C_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$

$$C_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$

$$C_3 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$$

$$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

Hamming Code (cont.)

- Bit arrangement
 - Set the bit in the position with the same value of its syndrome word

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					C4				C3		C2	C1

Hamming Code (cont.)

- Example

- Assume the 8 bits data is $D=01101010$, and use even parity in producing hamming code

- The code is

$$C_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$C_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

$$C_3 = D_2 \oplus D_3 \oplus D_4 \oplus D_8 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_8 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

- The 12 bits content when store

011001010011



Hamming Code (cont.)

- Example (cont.)

- The 12 bits content when fetch

- Case 1: 011001010011

$$D' = 01101010 \rightarrow C'' = 0011, C' = 0011$$

$$S = C'' \oplus C' = 0011 \oplus 0011 = 0000$$

- Case 2: 011101010011

$$D' = 01111010 \rightarrow C'' = 1010, C' = 0011$$

$$S = C'' \oplus C' = 1010 \oplus 0011 = 1001$$

- Case 3: 011011010011

$$D' = 01101010 \rightarrow C'' = 0011, C' = 1011$$

$$S = C'' \oplus C' = 0011 \oplus 1011 = 1000$$

[谭琼, 141250122]



Hamming Code (cont.)

- SEC
 - Single-Error-Correcting
 - Can find and correct one bit error
- SEC-DED
 - Single-Error-Correcting, Double-Error-Detecting
 - Can find two bits error and check one bit error
 - Add one additional bit
$$C_5 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_8$$
 - If an error occurs in one bit data, three bits check code will be changed



Hamming Code (cont.)

- SEC-DED (cont.)
 - All 0s: no error has been detected
 - One bit is 1: an error has occurred in one of the 5 check bits, and no correction is needed
 - Two bits are 1: errors have occurred in two of data and check bits, but the positions of errors cannot be found
 - Three bits are 1: an error has occurred in one of the 8 data bits, and the error can be corrected
 - More than three bits are 1: serious situation, examine the hardware

[马昕, 131250093]




Hamming Code (cont.)

- SEC-DED (cont.)
 - An error-correcting code enhances the reliability of the memory at the cost of added complexity
 - The size of main memory is actually larger than is apparent to the user

Data Bits	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
	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

Cyclic Redundancy Check

- Problem of parity checking
 - Additional cost is large
 - Require to divide data into bytes
- CRC 
 - Suitable for storing and transmitting large size data in stream format
 - Generate the relationship between data and check code with mathematic function

Cyclic Redundancy Check (cont.)

M

- Basic idea
 - Assume the data has n bits, left shift the data, and divide it (**mod 2 operation**) with a $k + 1$ bits **generator polynomial**
 - Use the k bits remainder as the check code
 - Put the check code behind the data
- Check
 - If the $n + k$ bits content can be divide by generator polynomial, no error occurs
 - Otherwise, errors occur



Cyclic Redundancy Check (cont.)

- Example
 - Assume the data is 100011, the generation polynomial is 1001
 - The check code is 111

$$\begin{array}{r} 100111 \\ 1001 \overline{) 100011000} \\ \underline{1001} \\ 0011 \\ \underline{0000} \\ 0111 \\ \underline{0000} \\ 1110 \\ \underline{1001} \\ 1110 \\ \underline{1001} \\ 1110 \\ \underline{1001} \\ 111 \end{array}$$



Summary

- Error correction
- Parity checking
- Hamming code
- Cyclic redundancy check

Thank You