ECE454/544: Fault-Tolerant Computing & Reliability Engineering

Lecture #4 –

Information Redundancy Techniques (I)

Instructor: Dr. Liudong Xing

Administrative Issues (9/19, Monday)

- Homework#1
  - Please download the problems from the course website: https://xingteaching.sites.umassd.edu/
  - Due **Sept. 21, Wednesday**
- Project Proposal
  - Due **Oct. 5, Wednesday**
  - Refer to Proposal Guideline on the course website

Project Teams

No.	Teams	Topic
1	Karen & Zakaria	
2	Devaj & VikramGnanaraj	
3	Connor & Marjan	
4	Bikram & Sushovan	Towards Fault tolerant Edge computing for smart energy and water/ IoT based smart city application
5	Ebenezer & MiniKusum	
6	Cedric & Marcel	

Review of Lecture #3

Passive redundancy uses fault masking to hide the occurrence of faults and prevent the faults from resulting in errors and failures

- **TMR** is the most common form of passive hardware redundancy, triplicated TMR can overcome the effects of the single-point of failure (voter)
- Hardware and software voting have their pros and cons, the decision must be made based on several factors
- Mid-value select technique and voting on part of data can be used to alleviate the problem of disagreeing results in an NMR system

Review of Lecture #3 (Cont'd)

- **Active** redundancy uses detection, location, and recovery techniques (reconfiguration)
  - **Duplication with comparison** can only detect faults, not tolerate them
  - **Hot standby sparing** can minimize the disruption in performance but consume more power than **cold standby sparing**
- **Hybrid** redundancy employs both fault masking and reconfiguration
  - **NMR with spare** technique can accomplish the same results using fewer hardware modules than passive approaches, but with fault detection/location/recovery schemes
  - **Self-purging redundancy** technique uses the system output to remove modules whose output disagrees with the system output

Information Redundancy

- Addition of redundant information to data to allow fault detection, fault masking, or fault tolerance
- Many errors in computer systems are committed at the bit or byte level
- **Error detecting / correcting codes**
  - Addition of redundant information to data words
  - Mapping of data words into new representation containing redundant information

Topics

- Basic concepts
- Example codes
- Code selection issue

Basic Concepts (1)

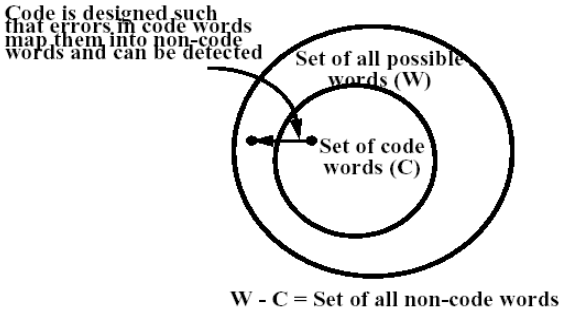
- **Code**: a means of representing information using a well-defined set of rules
- **Code word**: a collection of symbols used to represent a particular piece of information based on a specific code
  - A code word is said to be ***valid*** if it adheres to all the rules that define the code
- **Binary code**: a code in which all symbols forming each code word are either 0 or 1

Basic Concepts (2)

- **Error detecting code**: A code which is capable of detecting errors
- **Error correcting code**: A code which is capable of correcting errors

  The code word is structured such that it is possible to determine the correct code word from the corrupted code word.
  - Single-error correcting code: a code that can correct single-bit errors
  - Double-error correcting code: a code that can correct 2-bit errors

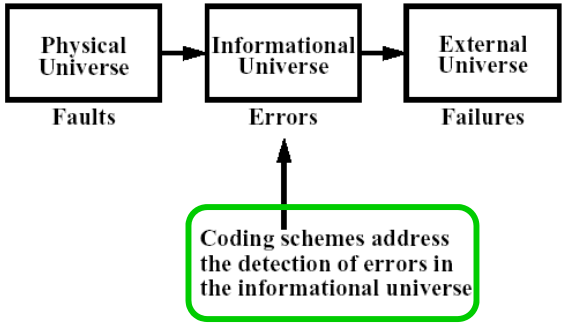
Basic Concepts (3)



- Only part of combinations are considered to be valid
- The occurrence of one of invalid combinations signals the existence of an error

Basic Concepts (4)

- **Fault:** a physical defect or imperfection
- **Error**: the manifestation of a fault; the corruption of information



Basic Concepts (5)

- **Encoding process**: the process of determining the corresponding code word for a particular data item.
  - E.g., the BCD code word of decimal digit 7 is 0111
- **Decoding process**: the process of recovering the original data from the code word.
  - E.g., the decoding process transforms the BCD code word 1001 into decimal digit 9

Error Models Used in Coding Theory

- **Bit Error**: A single bit of information is corrupted from a 1 to a 0 or from a 0 to a 1
- **Symmetric Errors**: 0 to 1 and 1 to 0 errors are equally likely to occur
- **Asymmetric Errors**: A given word has only 0 to 1 errors or 1 to 0 errors, and it is known *a priori* which type exists
- **Unidirectional Errors**: A given word has only 0 to 1 errors or 1 to 0 errors, but it is not known *a priori* which type exists
- **Byte Errors**: Errors will affect bytes independently, but it is not known how many bits are affected or the type of effect

Fundamental Types of Codes (1)

- **Block Code:**

  A sequence of information digits is broken into sections (or blocks) which each contain k digits.

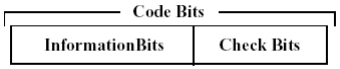
  The blocks are operated upon independently using the rules of the code to form code words with n digits each

- **Tree Code:**

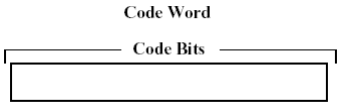
  The complete sequence of information is operated on without breaking it up into independent blocks

Fundamental Types of Codes (2)

- **Separable codes**:
  A code is separable if the code words are formed by appending a collection of check bits to the original information bits



- **Non-separable codes**:
  A code is non-separable if the code words are not partitioned directly into information bits and check bits.



Distance Concepts

- **Hamming Distance** (HD) between two binary words is the number of bit positions in which the two words differ
- Examples:
  - HD(0000, 0101) = ?
  - HD(0000, 1000) = ?
- Calculation:
  - $A=(a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0)$, $B=(b_{n-1}\ b_{n-2}\ \ldots\ b_1\ b_0)$,
  - Modulo-2 addition: 0+0=0, 0+1=1+0=1, 1+1=0
  - |A|=weight of A=number of 1s in A
  - *HD*(A,B)=|A+B|
- **Code Distance** (CD) for a given code is the minimum HD between any two valid code words

Error Detection/Correction Capabilities

- **Theorem 1**: a code can detect up to ***d*** bits errors *iff* the *CD >= d+1*
- **Theorem 2**: a code can correct up to ***c*** bits errors *iff* the *CD >= 2c+1*
- **Theorem 3**: a code can correct up to ***c*** bits errors and detect an additional ***d*** bits errors *iff* the *CD >= 2c+d+1*

Agenda

- ✓ Basic concepts
- Example codes
  - **Parity codes**
  - m-of-n
  - Berger
  - Checksums
  - Cyclic
  - Arithmetic
- Code selection issue

Parity Codes (1)

- Through the addition of some extra bits to a binary data word such that the resulting code word has either an odd or even number of 1s
  - Odd Parity: the total number of 1s in the code word is odd
    - 0010 → ?
      → 0010 0
  - Even Parity: the total number of 1s in the code word is even
    - 0010 → ?
      → 0010 1

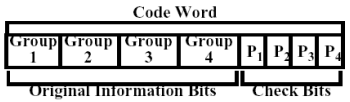
Parity Codes (2)

- Single-bit parity

Code Word Using Single-Bit Parity

Original Information	P

- Multiple-bit parity
  - The original information is partitioned into two or more groups with one parity bit being assigned to each group.

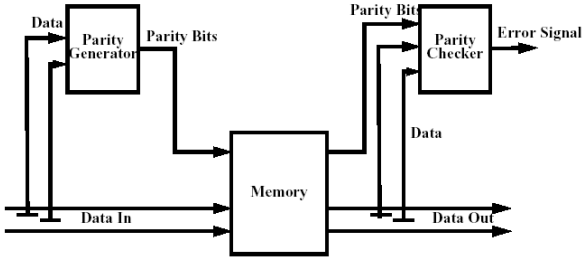


  - One parity group may use odd parity, while another uses even parity.

The parity code is NOT NECESSARILY a separable code!

Single-Bit Parity Codes

- Features
  - Simple and inexpensive to implement
  - Detects all single-bit errors since it is distance-2 code
  - Detects all errors which involve an odd number of bits
- Example use of single-bit parity codes in a memory of a computer system

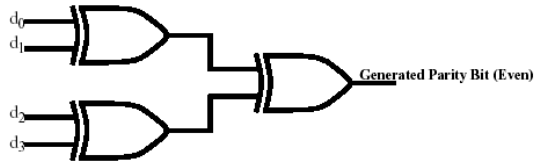


Information redundancy often requires hardware redundancy!

Example Single-Bit Parity Generation Circuit

Exercise: design a single-bit Even parity code generation circuit for two-bit data words (d_0, d_1)

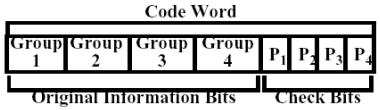
- Truth table
- Karnaugh map
- Logic function
- Implementation: circuit



Odd?

Multiple-Bit Parity Codes

- The original information is partitioned into two or more groups with one parity bit being assigned to each group



- Each bit may appear in more than one parity group – overlapping parity

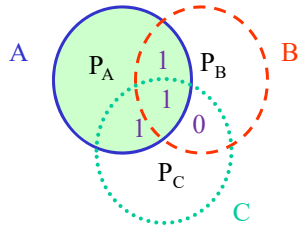
- **Example:**
  - Hamming Single Error-Correcting (SEC) Code (devised by Richard Hamming at Bell Labs)
  - Horizontal and Vertical Parity code

Hamming SEC Code

- References
  - W. Stallings, "Computer Organization and Architecture: Designing for performance", NJ: Prentice Hall (Chapter 5)
  - B. W. Johnson, "Design and Analysis of Fault Tolerant Digital Systems", Addison-Wesley, 1989 (Chapter 3.5.9)
  - Martin L. Shooman, "Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design", John & Sons Wiley, January 2002 (Chapter 2.4.3)

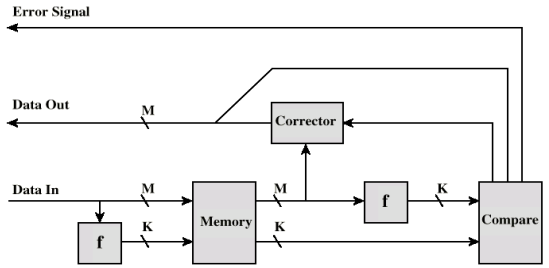
Venn Diagram Illustration of Hamming SEC Code

- The use of hamming code on 4-bit data word



- Assign 4 data bits to 4 inner compartments, parity bits to 3 left compartments
- Use even parity for each group (circle) → P_A=1, P_B=0, P_C=0
- Any single bit error can be easily detected by checking the discrepancies in parity bits

Hamming SEC Codes General Logic



- When a data word (*M* bits) is to be written into memory, a calculation *f* is performed to produce the check bits (*K* bits); both data and check bits are stored
- When data are read out, a new set of *K* check bits is generated and compared with the fetched check bits

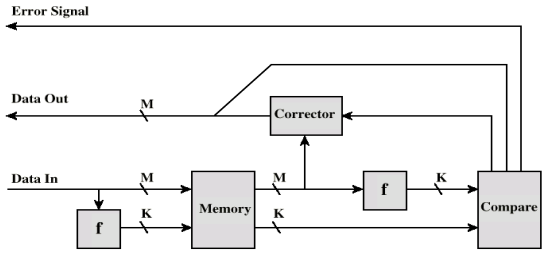
General Logic (Cont'd)

- Outcomes:
  - Fetched data are sent out when no errors are detected
  - Data and check bits are fed into a corrector producing a corrected data word to be sent out when an error is detected and is possible to correct error
  - An error signal is sent out when an error is detected and is not possible to correct error

Hamming SEC Codes

- How many check bits?
- How to arrange data bits and check bits?
- How to generate check bits?

Hamming SEC Codes
How many check bits?



- The comparator receives 2 *K*-bit values as inputs and generates the *K*-bit **syndrome word** by performing a bit-by-bit comparison
- The value 0 of the syndrome word indicates no error, leaving 2^K-1 values to indicate an error occurring on any of *M* data bits or *K* check bits →

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

Examples

- Example: M=4, K?
- Example: M=8, K?

Hamming SEC Codes

How to arrange & generate check bits?

- Number all bit positions from 1 to (M+K)
- Arrangement of bit positions (e.g., M=8, K=4)
  - Check bits: bit positions whose position numbers are power of 2
  - Data bits: other positions

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

- Check-bit generating rule:
  - Each check bit is generated by performing *exclusive-or* operation on every data bit whose position number contains a 1 in the same bit as the position number of that check bit

$$C_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$
$$C_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$
$$C_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$$
$$C_8 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$$

Hamming SEC Generation (Summary)

- Step1: Find # of check bits K based on

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

- Step 2: Arrange all bit positions
- Step 3: Generate check bits (exclusive-OR)
- **Step 4: Generate the SEC code word**

Hands-On Problem

- Suppose a 4-bit data word stored in memory is $D_4D_3D_2D_1$=1001. Generate the Hamming SEC code for this word.
- Assume an error occurs on D_4: 1→0, what happens?
- Assume an error occurs on a check bit, what happens?

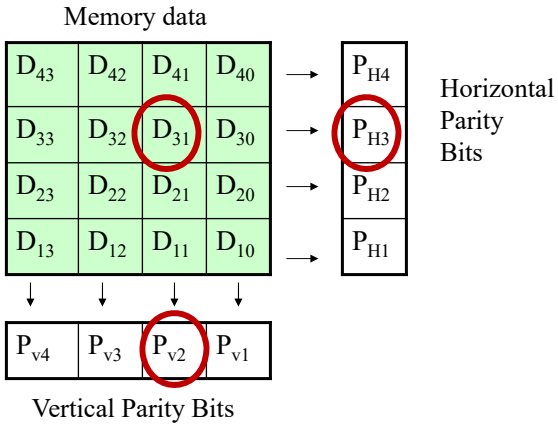
Hamming SEC Codes Characteristics

Syndrome Word = Original Checkbits ⊕ Regenerated Checkbits

- No error detected if syndrome contains all 0s
- An error has occurred in check bits if syndrome contains **one and only one bit set to 1**; no correction needed.
- An error has occurred in data bits if syndrome contains **more than one bit set to 1**; the numerical value indicates the position of the data bit in error. The data bit is inverted for correction.

Horizontal and Vertical Parity

- Useful for correcting errors in groups of data words which are transmitted from one point to another
- Uses a parity bit for each row and each column



- Any single-bit error can be detected and located because the error affects the parity in both a column and a row

Agenda

- ✓ Basic concepts
- Example codes
  - √ Parity codes: Hamming SEC code, Horizontal and Vertical Parity code
  - – m-of-n
  - – Berger
  - – Checksums
  - – Cyclic
  - – Arithmetic
- Code selection issue

Next Lecture!

Summary of Lecture #4

- Basic definitions
  - Code, code word, binary code, error detecting /correcting code, encoding / decoding process
- Error models for code development
  - Bit error, symmetric errors, asymmetric errors, unidirectional errors, and byte errors
- Coding theory concepts
  - Hamming distance, code distance, error detection/correction capabilities (3 theorems)
- Parity codes
  - Single-bit and multiple-bit parity codes
  - Hamming single error correcting codes
    - Calculate number of check bits
    - Arrange bit positions
    - Generate the check bits
    - Correct the erroneous bit according to the syndrome word
  - Horizontal and Vertical parity code: can correct any single-bit errors in groups of data words

Things to DO

- Homework
- ECE544 Project Proposal
  - Due **Wednesday, Oct. 5**