Chapter 2. Digital Data Representation

Overview

- Representation of numbers, characters and other forms of information are represented on computers using binary data
- 2.1 Representation of Numbers
- 2.2 Representation of Alphabet and Control Characters
- 2.3 Error Detection and Correction

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - A decimal number has base 10 and digit values of 0, 1, 2, …, 9
    - Example: $123.45_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
  - A binary number has base 2 and digit values of 0 and 1

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Conversion of an unsigned binary number to a decimal number: compute the sum using the base representation of the binary number
    - Example: $101.01_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 4_{10} + 0_{10} + 1_{10} + 0_{10} + 0.25_{10} = 5.25_{10}$
    - A binary integer with *N* binary digits (called bits) can represent an unsigned decimal integer from 0 to 2^N-1

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Conversion of an unsigned decimal number to a binary number
    - Repeated subtraction method
      1. Look for the highest power of 2 that is smaller than the decimal number
      2. Subtract that power of 2 from the decimal number
      3. Take the result of the subtraction in Step 2 as the decimal number
      4. If the decimal number is zero or meet another stopping criterion, go to Step 5; otherwise, go back to Step 1
      5. Construct the binary number by filling 1 in each digit position where the power of 2 is used in the subtraction in Step 2 and 0 in each digit position where the power of 2 is not used in the subtraction in Step 2

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers

TABLE 2.2

Decimal Numbers for Some Powers of 2

Power of 2	Decimal Number
2^{-4}	$\frac{1}{2^4} = 0.0625$
2^{-3}	$\frac{1}{2^3} = 0.125$
2^{-2}	$\frac{1}{2^2} = 0.25$
2^{-1}	$\frac{1}{2^1} = 0.5$
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^9	256
2^{10}	512

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.1: Use the repeated subtraction method to convert the decimal number 123.45 into a binary number with four binary digits after the decimal point

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.1

    $123.45 - 2^6 = 59.45$ or $123.45 = 1 \times 2^6 + 59.45$

    $59.45 - 2^5 = 27.45$ or $123.45 = 1 \times 2^6 + 1 \times 2^5 + 27.45$

    $27.45 - 2^4 = 11.45$ or $123.45 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 11.45$

    $11.45 - 2^3 = 3.45$ or $123.45 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 3.45$

    $3.45 - 2^1 = 1.45$ or $123.45 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1.45$

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.1

    $1.45 - 2^0 = 0.45$ or

    123.45

    $= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0.45$

    $0.45 - 2^{-2} = 0.20$ or$123.45 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0.20$

    $0.20 - 2^{-3} = 0.075$ or

    123.45

    $= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0.075$

    $0.075 - 2^{-4} = 0.0125$ or

    123.45

    $= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 0.0125$

    The binary number: 1111011.0111

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Conversion of an unsigned decimal number to a binary number
    - The division remainder method: for the integer part of the decimal number
      1. Divide the decimal number by 2, obtain the remainder of the division, and take the quotient of the division as the next decimal number
      2. If the decimal number is zero, go to Step 3; otherwise, go back to Step 1
      3. Construct the binary number by reading the division remainders from the last step to the first step of Step 1 performed

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Conversion of an unsigned decimal number to a binary number
    - The division remainder method: for the fractional part of the decimal number
      1. Multiply the decimal number by 2, obtain the unit digit of the multiplication product, and take the fractional part of the multiplication product as the next decimal number
      2. If the decimal number is zero or a required number of digits after the decimal point is obtained, go to Step 3; otherwise, go back to Step 1
      3. Construct the binary number by reading the unit digit of the multiplication product from the first step to the last step of Step 1 performed

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.2: Use the division remainder method to convert the decimal number 123.45 into a binary number with five binary digits after the decimal point

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.2: for the integer part

    $123 \div 2 = 61$ with the remainder of $\mathbf{1}$ or $123 = 61 \times 2^1 + \mathbf{1} \times 2^0$

    $61 \div 2 = 30$ with the remainder of $\mathbf{1}$ or

    $$123 = (30 \times 2^1 + \mathbf{1}) \times 2^1 + 1 \times 2^0$$
    $$= 30 \times 2^2 + \mathbf{1} \times 2^1 + 1 \times 2^0$$

    $30 \div 2 = 15$ with the remainder of $\mathbf{0}$ or

    $$123 = 15 \times 2^3 + \mathbf{0} \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

    $15 \div 2 = 7$ with the remainder of $\mathbf{1}$ or

    $$123 = 7 \times 2^4 + \mathbf{1} \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.2: for the integer part

    $7 \div 2 = 3$ with the remainder of $\mathbf{1}$ or

    $123 = 3 \times 2^5 + \mathbf{1} \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

    $3 \div 2 = 1$ with the remainder of $\mathbf{1}$ or

    123

    $= 1 \times 2^6 + \mathbf{1} \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

    $1 \div 2 = 0$ with the remainder of $\mathbf{1}$ or

    123

    $= (0 \times 2 + \mathbf{1}) \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

    $= \mathbf{1} \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

    the binary integer is 1111011

2.1 Representation of Numbers

- 2.1.1 Conversion between Unsigned Binary and Decimal Numbers
  - Example 2.2: for the factional part

    $0.45 \times 2 = \mathbf{0}.90$ or $0.45 = (\mathbf{0} + 0.90) \times 2^{-1} = \mathbf{0} \times 2^{-1} + 0.90 \times 2^{-1}$ (note that the unit digit is highlighted in bold)

    $0.90 \times 2 = \mathbf{1}.80$ or

    $0.45 = 0 \times 2^{-1} + 0.90 \times 2^{-1} = 0 \times 2^{-1} + (\mathbf{1} + 0.80) \times 2^{-1} \times 2^{-1}$

    $= 0 \times 2^{-1} + \mathbf{1} \times 2^{-2} + 0.80 \times 2^{-2}$

    $0.80 \times 2 = \mathbf{1}.60$ or

    $0.45 = 0 \times 2^{-1} + 1 \times 2^{-2} + 0.80 \times 2^{-2}$

    $= 0 \times 2^{-1} + 1 \times 2^{-2} + (\mathbf{1} + 0.60) \times 2^{-1} \times 2^{-2}$

    $= 0 \times 2^{-1} + 1 \times 2^{-2} + \mathbf{1} \times 2^{-3} + 0.60 \times 2^{-3}$

    $0.60 \times 2 = \mathbf{1}.20$ or

    $0.45 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0.60 \times 2^{-2}$

    $= 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + (\mathbf{1} + 0.20) \times 2^{-1} \times 2^{-3}$

    $= 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + \mathbf{1} \times 2^{-4} + 0.20 \times 2^{-4}$

    the binary number is 0.0111

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.1 Signed Magnitude Method
    - Use the left-most bit of a binary integer to represent the sign: 0 for the positive sign and 1 for the negative sign
    - A *N*-bit binary integer can represent any value in the range $[-(2^{N-1}-1), (2^{N-1}-1)]$
    - To perform the addition of two signed integers using the signed magnitude method, we need to process the sign bit separately from the number bits
    - Example: perform the addition of one positive integer and one negative integer
      - Determine which integer has the smaller magnitude (the smaller integer in the number bits)
      - Perform the subtraction of the smaller integer from the larger integer using the number bits
      - Assign the sign of the larger integer as the sign of the subtraction result

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.1 Signed Magnitude Method

TABLE 2.3

Examples of Representing Signed Numbers Using the Signed Magnitude Method

Signed Decimal Numbers	8-bit Binary Numbers
-123	11111011
123	01111011
0	00000000
-127	11111111
127	01111111

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - Diminished complement: given an integer *a* with base *b* and *N* digits, the diminished complement of *a* is $(b^N - 1) - a$
    - Example: given a decimal integer 43 with base 10 and 3 digits, the diminished complement of 43 is $(10^3 - 1) - 43 = 999 - 43 = 956$

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - One's complement method uses the diminished complement to represent a negative integer: 156 – 43

$$156 - 43 = x$$

$$156 - 43 + 999 = x + 999$$

$$156 + 956 \text{ (the diminished complement of } 43) = x + 999$$

$$1112 = x + 999$$

$$1112 - 1000 + 1 \text{ (end carry around: the carry - out beyond 3 digits is moved as the unit digit and added)} = x + 999 - 1000 + 1$$

$$112 + 1 = x$$

$$113 = x$$

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - Given a binary integer 0101 with base 2 and 4 digits, the diminished complement of 0101 is $(2^4 - 1) - 0101 = 1111 - 0101 = 1010$
    - The diminished complement of a binary integer is easy to implement in digital circuits by simply flipping each bit

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - A binary integer with N bits can represent a signed integer in the range of $[-(2^{N-1}-1), (2^{N-1}-1)]$ with the left-most bit indicating the sign (the left-most bit of 0 for a positive integer and the left-most bit of 1 for a negative integer)
    - both 0000 and 1111 represent zero in one's complement method

2.1 Representation of Numbers

TABLE 2.4

Representation of Signed Binary Integers with 4 Bits Using One's Complement Method

Signed Integer	Binary Integer
-7	1000
-6	1001
-5	1010
-4	1011
-3	1100
-2	1101
-1	1110
0	0000 or 1111
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - The addition of two binary integers is carried out in the following steps:
      1. Represent the binary integers using one's complement methods
      2. Perform the addition of the binary integers represented by one's complement method
      3. Perform the end carry around to obtain the result

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - Example 2.3: Perform the addition of one positive 4-bit binary integer and one negative 4-bit binary integer, 0100 – 0011 (4 – 3), using one's complement method

      1. Represent – 0011 using its diminished complement 1100
      2. 0100

         + 1100 (the diminished complement of 0011)

         = 10000
      3. End carry around

         0000

         + 1

         = 0001

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.2 One's Complement Method
    - Example 2.4: Perform the addition of one positive 4-bit binary integer and one negative 4-bit binary integer, 0011 – 0100 (3 – 4), using one's complement method
      - Represent -0100 using its diminished complement 1011
      - 0011
      + 1011 (the diminished complement of 0100)
      = 1110
      - 1110
        + 0
      = 1110

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.3 Two's Complement Method
    - The most common method used on computers
    - Use the complement which is defined by $b^N - a$ for an integer *a* with base *b* and *N* digits
    - Two's complement is one's complement plus one
    - Example: the complement of 0101 is $2^4 - 0101 = 10000 - 0101 = 1011$
    - Obtain the complement of a binary integer in digital circuits by flipping each bit and then adding 1

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.3 Two's Complement Method
    - Using *N* bits, two's complement method can represent signed integers in the range $[-2^{N-1}, 2^{N-1} - 1]$

2.1 Representation of Numbers

TABLE 2.5

Representation of Signed Binary Integers with 4 Bits Using Two's Complement Method

Signed Integer	Binary Integer
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.3 Two's Complement Method
    - The addition of two signed integers with *N* bits represented by two's complement method is performed in the following steps:
      1. Represent the binary integers using two's complement methods
      2. Perform the addition of the binary integers represented by two's complement method
      3. Discard the carry-out beyond *N* bits

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.3 Two's Complement Method
    - Example 2.5: Perform the addition of one positive 4-bit integer and one negative 4-bit integer, 0100 – 0011 (4 – 3), using two's complement method.
      - Represent – 0011 by its complement 1101. In Step 2, we perform the addition:
      - 0100
      - + 1101 (the complement of 0011)
      - = 10001
      - Drop the carry-out bit 1 beyond 4 bits and obtain 0001 as the result

2.1 Representation of Numbers

- 2.1.2 Representation of Signed Integers
  - 2.1.2.3 Two's Complement Method
    - Example 2.6: Perform the addition of one positive 4-bit integer and one negative 4-bit integer, 0011 – 0100 (3 – 4), using two's complement method.
      - Represent -0100 by its complement 1100. In Step 2, we perform the addition:
      - 0011
      + 1100 (the complement of 0100)
      = 1111
      - There is no carry-out, and the result is 1111 which is -1

2.1 Representation of Numbers

- 2.1.3 Representation of Signed Floating Point Numbers
  - A floating point number is represented using the sign bit (0 for the positive sign and 1 for the negative sign), bits for the exponent, and bits for the fractional part of the floating point number which is called significand

TABLE 2.6

Representation of a Floating Point Number

Sign	Exponent	Significand
0	00011	1000100000

2.1 Representation of Numbers

- 2.1.3 Representation of Signed Floating Point Numbers
  - The number of bits used for the exponent determines the range of numbers that can be represented
  - The number of bits used for the significand determines the precision of the number that is represented

2.1 Representation of Numbers

- 2.1.3 Representation of Signed Floating Point Numbers
  - Biased exponent representation
    - Example: 5 bits for an exponent can represent a value in the range [0, 31], and value near the middle of the range, e.g., 16, is used as the bias value for a 5-bit exponent
    - A biased exponent larger than the bias value represents a positive exponent, and a biased exponent smaller than the bias value represents a negative exponent
    - The exponent is obtained by subtracting the bias value from the biased exponent

2.1 Representation of Numbers

- 2.1.3 Representation of Signed Floating Point Numbers
  - Biased exponent representation
    - The exponent value of 3 is represented by the biased exponent which is 16 +3 = 19 or 10000 + 11 = 10011
    - The exponent value of -1 is represented by the biased exponent which is 16 – 1 = 15 or 10000 – 1 = 1111

2.1 Representation of Numbers

- 2.1.3 Representation of Signed Floating Point Numbers
  - Normalization
    - 0.10001×2^3, 0.010001×2^4, 0.0010001×2^5 represents 100.01
    - Normalization requires that the left-most bit of the significand must be 1
    - With normalization, 100.01 can be represented by 0.10001×2^3

2.2 Representation of Alphabet and Control Characters

- Text and control information is represented on computers by using codes for alphabet, number, control and other characters
- ASCII (American Standard Code for Information Interchange) character codes that use 8 bits
- A code itself uses 7 bits, and the left-most bit (the 8^{th} bit) is used as the parity bit
- Example: the character code for letter A is 65 or 1000001. If the parity bit is 0, the complete code for letter A is 01000001

2.2 Representation of Alphabet and Control Characters

TABLE 2.7

ASCII Character Codes

Character	Code	Character	Code	Character	Code	Character	Code
Null	0	Space	32	@	64	`	96
Start of heading	1	!	33	A	65	a	97
Start of text	2	"	34	B	66	b	98
End of text	3	#	35	C	67	c	99
End of transmission	4	$	36	D	68	d	100
Enquiry	5	%	37	E	69	e	101
Acknowledge	6	&	38	F	70	f	102
Bell (beep)	7	'	39	G	71	g	103
Backspace	8	(	40	H	72	h	104
Horizontal tab	9	)	41	I	73	i	105
Line feed, new line	10	*	42	J	74	j	106
Vertical tab	11	+	43	K	75	k	107
Form feed, new page	12	,	44	L	76	l	108
Carriage return	13	-	45	M	77	m	109
Shift out	14	.	46	N	78	n	110
Shift in	15	/	47	O	79	o	111
Data link escape	16	0	48	P	80	p	112
Device control 1	17	1	49	Q	81	q	113
Device control 2	18	2	50	R	82	r	114
Device control 3	19	3	51	S	83	s	115
Device control 4	20	4	52	T	84	t	116
Negative acknowledge	21	5	53	U	85	u	117
Synchronous idle	22	6	54	V	86	v	118
End of transmission block	23	7	55	W	87	w	119
Cancel	24	8	56	X	88	x	120

2.3 Error Detection and Correction

- Detecting and correcting errors are required for reliable data storage and transmission
- The parity bit method can detect an odd number of errors, but cannot detect an even number of errors or identify where errors occur to correct errors

2.3 Error Detection and Correction

- The Hamming distance of two codes is the number of bits in which two codes are different
- Example
  - 1000001 (65) along with its parity bit 0 produces the ASCII code 01000001 for letter A
  - 1000011 (67) along with its parity 1 produces the ASCII code 11000011 for letter C
  - The two codes:

    01000001

    11000011

    differ in two bit positions and have the Hamming distance of 2

2.3 Error Detection and Correction

- For a code system such as ASCII, the minimum Hamming distance, D_{min}, is the smallest distance among all pairs of codes in the code system
- If the number of bit errors that occur to a code is no more than D_{min}, the errors can be detected
  - D_{min} is the smallest Hamming distance among all pairs of valid codes
  - The code with no more than D_{min} bits of errors is different from any valid codes in the system
  - Detect the presence of error(s) in a given code if the given code is different from any valid code in the system

2.3 Error Detection and Correction

- Correct error(s) in a given code to the valid code that is closest the given code with error(s) in the Hamming distance
- D_{min} of a code system must be $2p + 1$ to correct p bits of errors

2.3 Error Detection and Correction

- Determine how many check bits are required to detect and correct errors
  - Detect and correct 1 bit of error

$$(n+1)2^m \leq 2^n$$
$$(m+k+1)2^m \leq 2^{m+k}$$
$$m+k+1 \leq 2^k$$

2.3 Error Detection and Correction

- Determine how many check bits are required to detect and correct errors
  - Example 2.7: Determine the number of check bits needed for a code with 4 data bits in order to detect and correct 1 bit of error

    $m = 4$

    $$4 + k + 1 \leq 2^k$$
    $$5 + k \leq 2^k$$

    Comparing the values of $5 + k$ and 2^k, we obtain $k \geq 3$

    Let $k = 3$

2.3 Error Detection and Correction

- Hamming algorithm to construct Hamming codes
  1. Number *n* bit positions from right to left, starting with 1
  2. Let each bit position which is a power of 2 be a check bit, let other bits as data bits, and put in the values of data bits
  3. Write each bit position as the sum of the numbers that are powers of 2, using the highest power of 2 first, and then determine which bit positions each check bit contributes to the sum of the numbers for these bit positions
  4. Use the parity bit to determine the value of each check bit by considering the values at bit positions where the check bit contributes to the sum of numbers

2.3 Error Detection and Correction

- Example 2.8: Construct the Hamming code with data bits 0100, using the Hamming algorithm and the even parity bit
  - From Example 2.7, we need 3 check bits for 4 data bits and totally 7 bits for the code
  - Step 1: number 7 bit positions

Bit position	7	6	5	4	3	2	1
Value							

2.3 Error Detection and Correction

- Example 2.8
  - Step 2: let bit positions 1, 2 and 4 as the check bits, and put in the values of data bits 0100

Bit position	7	6	5	4	3	2	1
Value	0	1	0		0		

2.3 Error Detection and Correction

- Example 2.8
  - Step 3: write each bit position as the sum of the numbers that are powers of 2

    1 = 1

    2 = 2

    3 = 1 + 2

    4 = 4

    5 = 1 + 4

    6 = 2 + 4

    7 = 1 + 2 + 4

2.3 Error Detection and Correction

- Example 2.8
  - Step 4: compute the value of each check bit using the even parity

Bit position	7	5	3	1
Value	0	0	0	?

the even parity bit at bit position 1 is 0

2.3 Error Detection and Correction

- Example 2.8
  - Step 4: compute the value of each check bit using the even parity

Bit position	7	6	3	2
Value	0	1	0	?

the even parity bit at bit position 2 is 1

2.3 Error Detection and Correction

- Example 2.8
  - Step 4: compute the value of each check bit using the even parity

Bit position	7	6	5	4
Value	0	1	0	?

the even parity bit at bit position 4 is 1

the Hamming code is 0101010

Bit position	7	6	5	4	3	2	1
Value	0	1	0	1	0	1	0

2.3 Error Detection and Correction

- Example 2.9: Detect and correct a 1-bit error in a code 0101110. Note that the code 0101110 is produced by introducing a 1-bit error to the Hamming code 0101010 from Example 2.8 at bit position 3

2.3 Error Detection and Correction

- Example 2.9
  - A 1-bit error at bit position 1, 3, 5 or 7, using the check bit at position 1
  - A 1-bit error at bit position 2, 3, 6 or 7, using the check bit at position 2
  - No 1-bit error at bit positions 4, 5, 6 and 7, using the check bit at position 4,
  - A 1-bit error at position 3 is detected

2.3 Error Detection and Correction

- Example 2.9
  - Use the sum of the bit positions of the check bits that indicate an error
    - Check bits 1 and 2 indicates an error, 1 + 2 = 3
    - The 1-bit error occurs at bit position 3