The most simple error control code - Parity Bit

September 29, 2013

Introduction

When sending digital data , there may be error between transmitted code and the received code

There are lots of sources of error such as quantization noise, EM noise, thermal noise

Therefore it is necessary to implement some method to check whether the received codes have error or not

But does the receiver how to whether the recived code has error or not ? It is impossible for the receiver to *know* the code *before* it received the code .

For example, the transmitter transmit the code 00100110 , after passing through a noisy channel, the receiver get 10100110 , receiver will not know that he has receive a code with error in the first bit.
It is impossible for the receiver to *know* the message has error in the first bit, because that implies the receiver already knows the message the from the transmitter before transmission !

Then how the receiver check the error ? This problem can be solved using *Error Control Coding*

The core idea of error control coding is to add extra bit in the information to be send for error detection and error correction.

There are lots of error control coding, the most simple one is **parity bit**

Parity Bit Method

Suppose we have a 7-bit data, now we add one more extra parity bit, to form a 8-bit data code.

There are two method : even parity and odd parity.

Consider the even parity method (opposite for the odd parity)

The even parity method is that the last bit will be the bit such that the number of 1 in the code is even

For example

Information Bit	Number of '1'	Parity Bit
0010010	2	0
1110110	5	1

Therefore
For data 0010010 (7-bit, with even number of 1), the transmitted code will be 0010010 (8-bit, even number of 1)
For data 1110110 (7-bit, with odd number of 1), the transmitted code will be 1110111 (8-bit, even number of 1)

After the receiver get the bit, it will check the number of '1' in the received code, if the number of '1' is even, that means there is no error, if the number is odd, that means error has occurred.
If there is error, the receiver will acknowledge the transmitter to $re-send$ the code again.

There is one pitfall in this method, if 0010010 code becomes 0011110 (2 bit error occurred , although the chance is relatively low) , then this method can not detect error has occurred.

For odd parity method, the ideas is the similar.

Modular 2 Sum

Counting the number of '1' actually can be implemented using modular 2 addition.

In modular 2 addition :

$$0+0=0 \quad 0+1=1 \quad 1+0=1 \quad 1+1=0$$

Therefore , for a received code in the receiver $c_0c_1c_2...c_{n-1}c_n$, where $c_0...c_{n-1}$ are the information bits, and c_n is the parity bit

The receiver just need to do the following addition

$$S = c_0 \oplus c_1 \oplus c_2... \oplus c_{n-1} \oplus c_n = \begin{cases} 0 & \text{if there is even number of '1'} \\ 1 & \text{if there is odd number of '1'} \end{cases}$$

Parity Bit as an application of GaloisField(2)

The Galois Field (2) is the smallest finite field.

The operation + is corresponds to the logical XOR

+	0	1
0	0	1
1	1	0

The operation − is corresponds to the logical AND

×	0	1
0	0	0
1	0	1

Other error control code

Notice that, this method can only know "error has occurred" , but do not know "where" the error has occured. For error-control code that can also know "where" the error has occured, please refer to *linear block code*.

$-END-$