Note for lecture 03:

Definition: a "parity check code (or linear code)" is a linear transformation from the string of data bits to the string of data bits and parity checks.

Example: single parity check code

$$\underbrace{1011000}_{\text{data bits}} \xrightarrow{\text{transform}} \underbrace{1011000}_{\text{data bits}}\ \underbrace{1}_{\text{parity check}}$$

In general, there can be K data bits and L parity checks.

Definition: a "code word" is the result of the transform of a certain parity check code.

→ A code word is the data bits plus parity checks for
- ① single parity check
- ② horizontal & vertical parity check
- ③ CRC

a form of a code word: $\overbrace{\boxed{10011\cdots0}}^{K}\overbrace{\boxed{10\cdots1}}^{L}$

The sender of data sends the code word to the receiver, who ~~then~~ will decode the code word to ① see if there is any bit error in the code word; ② retreive the original data bits.

The receiver will not be able to detect bit error if the error has changed the code word to another code word, say, C_A, in which case, ~~for~~ from the receiver's viewpoint, say, C_B, it is entirely possible that code word C_B was obtained by ~~a certain~~ some other string of data bits and was not because of the error.

Therefore, a useful criterion to measure the effectiveness of a parity check code is to look at the smallest number of bit changes that can convert one code word into another, which we say to be the minimum distance of a code. A longer minimum distance is better, because it would take more bit errors to make a data receiver unable to detect an error; in other words, such a parity check code is more resilient to bit errors!

Exercise: show that the minimum distance of a code using a single parity check is 2.

Answer: We can first show that no two code words in this case can be differed by one.

prove by contradiction

Suppose code words X_1 and X_2 are differed by only one bit, then

→ X_1 and X_2 cannot both have even number of 1s.

→ either X_1 or X_2 must have odd number of 1s, which cannot be a code word.

→ a contradiction.

Next, we give a witness, i.e., there exists two code words such that the distance in between is 2.

$S_1(D) = D^2 + D \longrightarrow C_1(D) = 0$

$S_2(D) = D^2 + 1 \longrightarrow C_2(D) = 0$

and the distance between

$S_1(D) \cdot D + C_1(D)$

and

$S_2(D) \cdot D + C_2(D)$ is 2 ※

Reasoning CRC:

represent a code word by $X(D)$

$$X(D) = \underbrace{S(D)}_{\text{its coefficients are data bits}} \cdot D^L + \underbrace{C(D)}_{\text{its coefficients are parity checks}}$$

by definition of a code word ⇒

By choosing a generator polynomial, $g(D)$ and compute $\frac{S(D) \cdot D^L}{g(D)}$,

we have $S(D) \cdot D^L = g(D) \cdot z(D) + C'(D)$

use ~~let~~ the remainder polynomial $C'(D)$ ~~be~~ as parity checks. ⇒ $C(D) = C'(D)$

$$\Rightarrow X(D) = S(D) \cdot D^L + C(D)$$

$$= g(D) \cdot z(D) + C'(D) + C(D)$$

$$= g(D) \cdot z(D) \quad \text{(according to modular 2 computation)}$$

which means $g(D)$ divides $X(D)$.

Then, Send $X(D)$ $= g(D) \cdot z(D)$ which equals $S(D) \cdot D^L + C(D)$ to the receiver.

Now, use $e(D)$ to represent errors introduced along the sending path.

Then the receiver gets $y(D) = X(D) + e(D)$

compute $\frac{y(D)}{g(D)}$, and we see if $e(D) = 0$

then $g(D)$ must divide $y(D)$ and remainder $= 0$.

Otherwise, we say there's error.

An example of using CRC:

Suppose that 1100 are data bits to send, and suppose that 101 are coefficients of the generator polynomial; i.e., $g(D) = D^2 + 1$

Therefore $L = 2$, which is equal to the degree of $g(D)$.

$\Rightarrow X(D) = S(D) \cdot D^2$ ~~whig~~ which gives $\boxed{110000}$

to be replaced by $c(D)$

Then the long division (modulo 2):

```
        1111     <- z(D)
  101 ) 110000   <- X(D)
  g(D)  101
        ---
         110
         101
         ---
          110
          101
          ---
           110
           101
           ---
            11   <- c(D)
```

I forgot to append this two bits when giving lecture in class on 9/28.

The code word is $\boxed{110011}$

we then send the code word to the other end of the channel.

The receiver of the code word can check out if there's any errors by performing another long division (again, modulo 2) by $g(D)$:

↳ see next page


```
        1111
  101 ) 110011
        101
        ---
         110
         101
         ---
          111
          101
          ---
           101
           101
           ---
             0   <- c(D) = 0
```

implies that there's no error!

Suppose there's an error and the receiver got 110111 instead:

```
        1110
  101 ) 110111
        101
        ---
         111
         101
         ---
          101
          101
          ---
           01
```

$c(D) \neq 0$ implies that there's error!

The error polynomial in this case is $e(D) = D^2$, or 100, and we see that $g(D)$ does not divide $e(D)$

```
        1
  101 ) 100
        101
        ---
         01
```

this is in accordance with our reasoning on P4.

Finally, notice that CRC cannot detect error if $e(D)$ happened to be equal to $g(D)$:

```
        1
  101 ) 101
        101
        ---
          0
```

```
        1110
  101 ) 110110   ≡  110011  +  101
        101         the code   e(D)
        ---         word
         111
         101
         ---
          101
          101
          ---
           00   c(D) = 0 !
```