* Note: I will go over this in class again if I see necessary.

Cyclic Redundancy Check (CRC)

CRC can detect both single and burst errors

Commonly used CRCs CRC-1 (parity), CRC-3-GSM, CRC-4-ITU, CRC-5-EPC, CRC-5-USB, CRC-7-MVB (Train communication), CRC-8 (satellite DVB-S2), CRC-8/16-CCITT (wireless, Bluetooth), CRC-40 (GSM control channel-slower/reliable).

You can check hardware implementation of CRC and parity check here.

In CRC, you should know about the input message, the generator and the codeword.

When you divide message+padding by generator, you should get a remainder. Add that remainder to your message to get the codeword.

Lets go through the CRC process then,

Transmit Process

1. Your input message bit pattern
2. Your generator
3. Add padding (0s) after the message. The size of the padding will be number_of_bits(generator) - 1
4. Divide the message+padding with the generator
5. Store the remainder of size = number_of_bits(generator) - 1
6. Add that to your message and send.

Receive Process

1. Divide the received message by the generator
2. Did you get a remainder of 0's?
3. If not, there is/are error(s)
4. If yes, your recieved bits are okay.

Let's go through an example,

Impelementation (simple long division)- Transmitter side

```
      ---------- TRANSMISSION PROCESS (longer division)
------

      Your polynomial (input message) = 1101011011 (10
bits)
      Your generator = 10011 (5 bits)
      number of padding bits = number_of_bits(10011) - 1
                              = 4
      Your input message after padding = 11010110110000
(14 bits)
      The division process is a XOR at each step you see
below.
      Multiply generator with 1 when MSB is 1.

      _1100001010______
10011| 11010110110000
       10011|||||||||
       -----|||||||||
        10011||||||||
        10011||||||||
       ------||||||||
         00001|||||||
         00000|||||||
       -------|||||||
          00010||||||
          00000||||||
       --------||||||
           00101|||||
           00000|||||
       ---------|||||
            01011||||
            00000||||
       ----------||||
             10110|||
             10011|||
       -----------|||
              01010||
              00000||
       ------------||
               10100|
               10011|
       -------------|
                01110
                00000
       --------------
                 1110

So your tranismitted message is 11010110111110 where
1110 is the remainder

  ---------- shorter division ------------------

this division took a long while to solve. How about
making it shorter (optional read)


      _1100001010______
10011| 11010110110000
       10011|||||||||
       -----|||||||||
        10011||||||||
        10011||||||||
       ------||||||||
         000010110         ; we need 5 bits to make MSB
1
             10011         ; we add 4 zeros in the
quotient
       -----------|||
             0010100       ; we need 2 bits to make MSB
1
               10011       ; we add 1 zero in the
quotient
       -------------|
               001110      ; we are still adding bits
after the division is complete, add a zero to in the
quotient.

 ------------ arithmetic example -----------------

    Check an arithmetic exmple below,
       _20040_____________
    32|641281
       64
       ---
       128     ; adding 3 numbers to make it greater
than 32
       128     ; added 2 zeros in the quotient
       ---
       0001  ; after all that it still has 1 left in
dividend
             ; add zero in quotient
```

Impelementation (simple long division)- Receiver side (no error)

```
Receive side should receive -
message + crc_calculated(remainder) = 11010110111110
1101011011(1110)
Now we go through same process of division with
generator --
10011
But this time with dividend,

      _1100001010______
10011| 11010110111110
       10011|||||||||
       -----|||||||||
        10011||||||||
        10011||||||||
       ------||||||||
         00001|||||||
         00000|||||||
       -------|||||||
          00010||||||
          00000||||||
       --------||||||
           00101|||||
           00000|||||
       ---------|||||
            01011||||
            00000||||
       ----------||||
             10111|||
             10011|||
       -----------|||
              01001||
              00000||
       ------------||
               10011|
               10011|
       -------------|
                00000
                00000
       --------------
                 0000
```

Implementation- Receiver Side (With error)

```
Receive side should receive -
message + crc_calculated(remainder) = 11010110111110
But unfortunately received = 11010110011110
Now we go through same process of division with
generator --
10011
But this time with dividend,

      _1100001000______
10011| 11010110011110
       10011x|x|x|x|x
       -----x|x|x|x|x
        10011|x|x|x|x
        10011|x|x|x|x
       ------|x|x|x|x
         00001x|x|x|x
         00000x|x|x|x
       -------x|x|x|x
          00010|x|x|x
          00000|x|x|x
```

```
     --------|x|x|x
         00100x|x|x
         00000x|x|x
     ---------x|x|x
          01001|x|x
          00000|x|x
     ----------|x|x
           10011x|x
           10011x|x
     -----------x|x
            00001|x
            00000|x
     ------------|x
             00011x
             00000x
     --------------
              00110
              00000
     --------------
               0110

remainder is non-zero
so there is a receive error
```