3.3 Cyclic Codes

- Cyclic codes are special linear block codes with one extra property:
  
  If a code-word is cyclically shifted (rotated), the result is another code-word.
  
  For ex: if code-word = 1011000 and we cyclically left-shift, then another code-word = 0110001.

- Let First-word = \(a_0\) to \(a_6\) and Second-word = \(b_0\) to \(b_6\), we can shift the bits by using the following:
  \[b_1 = a_0, \quad b_2 = a_1, \quad b_3 = a_2, \quad b_4 = a_3, \quad b_5 = a_4, \quad b_6 = a_5, \quad b_0 = a_6\]

### 3.3.1 Cyclic Redundancy Check (CRC)

- CRC is a cyclic code that is used in networks such as LANs and WANs.

#### Table 10.3 A CRC code with \(C7, 4\)

<table>
<thead>
<tr>
<th>Dataword</th>
<th>Codeword</th>
<th>Dataword</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00000000</td>
<td>1000</td>
<td>1000101</td>
</tr>
<tr>
<td>0001</td>
<td>00010111</td>
<td>1001</td>
<td>1001110</td>
</tr>
<tr>
<td>0010</td>
<td>00101110</td>
<td>1010</td>
<td>1010111</td>
</tr>
<tr>
<td>0011</td>
<td>00111011</td>
<td>1011</td>
<td>1011000</td>
</tr>
<tr>
<td>0100</td>
<td>01001111</td>
<td>1100</td>
<td>1100110</td>
</tr>
<tr>
<td>0101</td>
<td>01011100</td>
<td>1101</td>
<td>1101001</td>
</tr>
<tr>
<td>0110</td>
<td>0110001</td>
<td>1110</td>
<td>1110100</td>
</tr>
<tr>
<td>0111</td>
<td>0111010</td>
<td>1111</td>
<td>1111111</td>
</tr>
</tbody>
</table>

- Let Size of data-word = \(k\) bits (here \(k=4\)).
  
  Size of code-word = \(n\) bits (here \(n=7\)).
  
  Size of divisor = \(n-k+1\) bits (here \(n-k+1=4\)). (Augmented \(\rightarrow\) increased)

- Here is how it works (Figure 10.5):

  1) **At Sender**
  
  - \(n-k\) 0s is appended to the data-word to create augmented data-word. (here \(n-k=3\)).
  
  - The augmented data-word is fed into the generator (Figure 10.6).
  
  - The generator divides the augmented data-word by the divisor.
  
  - The remainder is called check-bits \((r_2 r_1 r_0)\).
  
  - The check-bits \((r_2 r_1 r_0)\) are appended to the data-word to create the code-word.

  2) **At Receiver**
  
  - The possibly corrupted code-word is fed into the checker.
  
  - The checker is a replica of the generator.
  
  - The checker divides the code-word by the divisor.
  
  - The remainder is called syndrome bits \((r_2 f_1 f_0)\).
  
  - The syndrome bits are fed to the decision-logic-analyzer.
  
  - The decision-logic-analyzer performs following functions:

    1) **For No Error**
    
    - If all syndrome-bits are 0s, the received code-word is accepted.
    
    - Data-word is extracted from received code-word (Figure 10.7a).

    2) **For Error**
    
    - If all syndrome-bits are not 0s, the received code-word is discarded (Figure 10.7b).
Example 3.3

Given the dataword 10100111 and the divisor 10111, show the generation of the CRC codeword at the sender site (using binary division).
3.3.2 Polynomials
- A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1 (Figure 10.8).
- The power of each term shows the position of the bit; the coefficient shows the value of the bit.

\[ a_6 \ a_5 \ a_4 \ a_3 \ a_2 \ a_1 \ a_0 \]
\[ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \]
\[ x^6 \ + \ a_5 \ + \ a_4 \ + \ a_3 \ + \ a_2 \ + \ x^1 \ + \ x^0 \]

Figure 10.8 A polynomial to represent a binary word

3.3.3 Cyclic Code Encoder Using Polynomials
- Let Data-word = 1001 = x^3+1.
  Divisor = 1011 = x^3+x+1.
- In polynomial representation, the divisor is referred to as generator polynomial t(x) (Figure 10.9).

\[ \text{Dataword} \ x^3 + 1 \]

\[ \text{Divisor} \ x^3 + x + 1 \]
\[ \text{Augmented Dividend} \ x^6 + x^3 \]
\[ \text{Remainder} \ x^2 + x \]

Figure 10.9 CRC division using polynomials

3.3.4 Cyclic Code Analysis
- We define the following, where f(x) is a polynomial with binary coefficients:

<table>
<thead>
<tr>
<th>Dataword: ( d(x) )</th>
<th>Codeword: ( c(x) )</th>
<th>Generator: ( g(x) )</th>
<th>Syndrome: ( s(x) )</th>
<th>Error: ( e(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>In a cyclic code,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. If ( s(x) ) = 0,</td>
<td>one or more bits is</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>corrupted.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. If ( s(x) ) = 0,</td>
<td>either</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a. No bit is corrupted,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Some bits are</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>corrupted, but the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>decoder failed to detect them.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Single Bit Error
- If the generator has more than one term and the coefficient of \( x^0 \) is 1, all single-bit errors can be caught.

Two Isolated Single-Bit Errors
- If a generator cannot divide \( x^{t+1} \) (t between 0 & n-1), then all isolated double errors can be detected (Figure 10.10).

\[ \text{Difference: } j - i \]
\[ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \]
\[ x^{n-1} \ x^i \ x^j \ x^0 \]

Figure 10.10 Representation of two isolated single-bit errors using polynomials
Odd Numbers of Errors

- A generator that contains a factor of $x+1$ can detect all odd-numbered errors.

A good polynomial generator needs to have the following characteristics:

1. It should have at least two terms.
2. The coefficient of the term $x^8$ should be 1.
3. It should not divide $x^t + 1$, for $t$ between 2 and $n - 1$.
4. It should have the factor $x + 1$.

### Standard Polynomials

#### Table 10.4 Standard polynomials

<table>
<thead>
<tr>
<th>Name</th>
<th>Polynomial</th>
<th>Used in</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC-8</td>
<td>$x^8 + x^7 + x + 1$</td>
<td>ATM header</td>
</tr>
<tr>
<td></td>
<td>100000111</td>
<td></td>
</tr>
<tr>
<td>CRC-10</td>
<td>$x^{10} + x^9 + x^3 + x^4 + x^2 + 1$</td>
<td>ATM AAL</td>
</tr>
<tr>
<td></td>
<td>11000110101</td>
<td></td>
</tr>
<tr>
<td>CRC-16</td>
<td>$x^{16} + x^{12} + x^5 + 1$</td>
<td>HDLC</td>
</tr>
<tr>
<td></td>
<td>100010000000100001</td>
<td></td>
</tr>
<tr>
<td>CRC-32</td>
<td>$x^{32} + x^{26} + x^{24} + x^{23} + x^{22} + x^{18} + x^{16} + x^{14} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$</td>
<td>LANs</td>
</tr>
<tr>
<td></td>
<td>1000001001110000100011011011011</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.3.5 Advantages of Cyclic Codes

- The cyclic codes have a very good performance in detecting
  → single-bit errors
  → double errors
  → odd number of errors and
  → burst-errors.
- They can easily be implemented in hardware and software. They are fast when implemented in hardware.