Error Correcting Codes

Lecture 3. Cyclic Codes

Sang-Hyo Kim

Sungkyunkwan University

Outline

- Definition of Cyclic Codes
- Polynomial Representation of Cyclic Codes
  - Generator polynomial
  - Check polynomial
- Systematic Encoding
- Decoding of Cyclic Codes
- Examples of Cyclic Codes
  - Linear codes
  - Cyclic Redundancy Check (CRC)

Definition of Cyclic Codes

Definition of Cyclic Codes

- For quick understanding, we consider binary codes first (unless otherwise noted).
  - In $GF(2) = \{0,1\}$, note $1 + 1 = 0,\ -1 = 1$.

- Linear $[n, k, d]$ codes can be classified into
  - Cyclic codes
  - Non-cyclic codes

- Definition of cyclic codes
  - An $[n, k]$ code C is said to be 'cyclic' if $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) \in C$ then it cyclic shift $(c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C$.

Advantage of Cyclic Codes

- All below are codewrods
  - Definition (T) Right (cyclic) shift operator

$$\boldsymbol{c} = (c_0, c_1, \dots, c_{n-1})$$
$$T\boldsymbol{c} = (c_{n-1}, c_0, c_1, \dots, c_{n-3}, c_{n-2})$$
$$T^i\boldsymbol{c} = (c_{n-i}, c_{n-i+1}, \dots, c_{n-1}, c_0, \dots, c_{n-i-1})$$

- Advantages
  - Easy implementation of encoder and syndrome decoder
  - Easy to develop implementable decoding algorithms
  - Robust against burst errors

Polynomial Representation

Polynomial representation

Polynomial codes

Generator polynomial of cyclic codes

Check polynomial

Polynomial Representation

- Polynomial representation of code
  - Convenient

$$\boldsymbol{c} = (c_0, c_1, \dots, c_{n-1}) \Leftrightarrow c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

  - Cyclic shift of $\boldsymbol{c}$ (by i symbols)

$$T^i \boldsymbol{c} = (c_{n-i}, c_{n-i+1}, \dots, c_{n-1}, c_0, \dots, c_{n-i-1})$$

$$\leftrightarrow c_{n-i} + c_{n-i+1}\, x + \cdots + c_{n-i-1} x^{n-1}$$

  - Rewritten as

$$\begin{aligned} & c_0 x^i + c_1\, x^{i+1} + \cdots + c_{n-i} x^n + \cdots + c_{n-1} x^{n+i-1} \\ & = x^i (c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}) \\ & = x^i c(x) \end{aligned}$$

    - Multiplication of x is equivalent to the shift operator T.

Polynomial Codes and Cyclic Codes

- Polynomial codes
  - Code elements are represented by polynomials
  - $\boldsymbol{C} = \{c(x)\}$
    - $c(x) = c_0 + \cdots + c_{n-1}x^{n-1}$
  - Each codeword $c(x)$ is divided by a polynomial $g(x)$ of degree $m < n$.
  - $g(x)$ is called the 'generator polynomial.'

- Cyclic codes
  - A polynomial code is 'cyclic' iff $g(x)$ divides $x^n - 1$ $(g(x)|x^n - 1)$
    - Note that in $F_3 = \{0,1,2\}, -1 = 2$ since $1 + 2 = 3 = 0$.
  - For binary codes, the condition is rewritten as $g(x)|x^n + 1$

Generator Polynomial

- Generator Polynomial for a Cyclic Code C
  - If an $[n, k]$ code C is cyclic, there is a polynomial $g(x)$ called **"generator polynomial"** such that

$$\{g(x), xg(x), \dots, x^{k-1}g(x)\}$$

    form a basis of C.

  - The generator polynomial is given by

$$g(x) = g_0 + g_1 x + \cdots + g_{n-k} x^{n-k}$$

    - $g(x)|x^n + 1$, $g_{n-k} \neq 0, g_0 \neq 0$. The degree of $g(x)$ is $n - k$.

Generator Polynomial

- Codeword polynomial (encoding)

  $c(x) = m_0 g(x) + m_1 x g(x) + m_2 x^2 g(x) + \cdots m_{k-1} x^{k-1} g(x)$

  $= \left(m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}\right) g(x)$

  - Thus $c(x) = m(x) g(x)$

- Notes
  - Total number of message polynomials : $2^k = |C|$
  - The i th coefficient in $c(x)$ is given by

$$c_i = \sum_{j=0}^{i} m_j g_{i-j} = m_i * g_i$$

Note: Polynomial multiplication is equivalent to the convolution operation.

Generator Polynomial

- Example : [7,4] Hamming codes (cyclic)
  - $(x^7+1)=(x+1)\left(x^3+x+1\right)\left(x^3+x^2+1\right),\ n-k=3$
  - $g(x)=x^3+x+1\ \left(or\ x^3+x^2+1 \text{ for another Hamming code}\right)$

	polynomial representation	vector representation
$0\cdot g(x)$	0	$(0\,0\,0\,0\,0\,0\,0)$
$1\cdot g(x)$	$1+x+x^3$	$(1\,1\,0\,1\,0\,0\,0)$
$x\cdot g(x)$	$x+x^2+x^4$	$(0\,1\,1\,0\,1\,0\,0)$
$(1+x)g(x)$		
$x^2g(x)$		
$(1+x^2)g(x)$		
$(x+x^2)g(x)$		
$(1+x+x^2)g(x)$		
$x^3g(x)$		
$(1+x^3)g(x)$		
$(x+x^3)g(x)$		
$(1+x+x^3)g(x)$		
$(x^2+x^3)g(x)$		
$(1+x^2+x^3)g(x)$		
$(x+x^2+x^3)g(x)$		
$(1+x+x^2+x^3)g(x)$		

Generator Polynomial of Cyclic Code of Length 7

- Generator polynomials of cyclic code of length 7

$$x^7 + 1 = (x+1)\left(x^3 + x + 1\right)\left(x^3 + x^2 + 1\right)$$

generator polynomial	parameters	name of codes
1	$[7,7]$	entire space
$x+1$	$[7,6]$	even parity check code
x^3+x+1	$[7,4]$	Hamming code
$(x+1)(x^3+x+1)$	$[7,3]$	expurgated Hamming code
x^3+x^2+1	$[7,4]$	Hamming code
$(x+1)(x^3+x^2+1)$	$[7,3]$	expurgated Hamming code
$(x^3+x+1)(x^3+x^2+1)$	$[7,1]$	repetition code
$(x+1)(x^3+x+1)(x^3+x^2+1)$	$[7,0]$	zero code $=\{0\}$

Generator Polynomial

- Codeword polynomial

$$c(x) = m(x)g(x) = \sum_{i=0}^{k-1} m_i x^i g(x)$$

$$= (m_0, m_1, \dots, m_{k-1}) \left.\begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}\right\} \text{basis}$$

- The generator matrix of the corresponding linear block code

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \cdots & g_0 & g_1 & \cdots & \cdots & \cdots & g_{n-k} \end{bmatrix}$$

Example

- [7,4] Hamming code defined by $g(x) = x^3 + x + 1$
  - The generator matrix

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

  - By elementary row operations, it can be transformed into a systematic code generator

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Check Polynomial of Cyclic Codes

- Find $h(x)$ such that $h(x)g(x) = x^n + 1$

$$h(x) = h_0 + h_1 x + \cdots + h_{k-1}x^{k-1} + h_k x^k$$

where $\deg h(x) = k, h_0 \neq 0, h_k = 1$, then

$$c(x)h(x) = m(x)g(x) \cdot h(x)$$

$$= m(x)(x^n + 1) = 0, (\because x^n = 1)$$

- Parity check matrix
  - $\boldsymbol{h}_i \boldsymbol{g}_i^T$ forms a convolution of two sequences

$$\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_1 & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_2 & h_1 & h_0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdots & h_k & \cdots & \cdots & \cdots & \cdots & h_1 & h_0 \end{bmatrix}$$

- [7,4] cyclic code with $g(x) = x^3 + x + 1$

$$h(x) = (x+1)(x^3 + x^2 + 1) = x^4 + x^2 + x + 1$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \Longrightarrow \mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & : & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & : & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & : & 0 & 0 & 1 \end{bmatrix}$$

Systematic code

- Note : Dual code
  - An $[n, k]$ cyclic code generated by $g(x)$ has the $(n, n-k)$dual code, which is generated by

$$h^*(x) = h_k + h_{k-1}x + \cdots + h_1 x^{k-1} + h_0 x^k \text{ (reciprocal of } h(x))$$

Systematic Encoding

Systematic Encoding

Encoder Implementation

Systematic Encoding of Cyclic Codes

- Encoding of systematic cyclic code [1]
  - Division by generator polynomial where $q(x)$ is the quotient and $p(x)$ is the remainder $(x^n + 1 = 0)$

$$x^{n-k}m(x) = q(x)g(x) + p(x)$$

  - Codeword : $c(x) = x^{n-k}m(x) + p(x) = q(x)g(x)$
    - Codeword vector

MSB here in this representation

$$\boldsymbol{c} = (p_0, p_1, \ldots p_{n-k-1}, m_0, m_1, \ldots, m_{k-1})$$

    - We have bijective (1to1 onto) mapping from $m(x)$ to $c(x)$

Systematic Encoding of Cyclic Codes

Note: MSB on the left.

- Example

```
11010011101100 000 <--- input left shifted by 3 bits
1011 <--- divisor
01100011101100 000 <--- result
 1011 <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1 -----------------
00000000000000 100 <---remainder (3 bits)
```

New Basis for Systematic Cyclic Code

- Generator polynomial :

$$g(x) = g_0 + g_1 x + \cdots + g_{n-k} x^{n-k} \quad where\ g_0 = g_{n-k} = 1$$

- Message polynomial :

$$m(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$$

- Consider monomials $(i = 0,1, \dots, k-1)$ and division with $g(x)$

$$x^{n-k+i} = q_i(x) g(x) + r_i(x)$$
$$\Rightarrow x^{n-k+i} + r_i(x) = q_i(x) g(x)$$ 'a codeword'

- New basis

$$\{x^{n-k+i} + r_i(x) | i = 0, \dots, k-1\}$$

  - C is spanned by the basis
  - Example: codeword $x^{n-k} + r_0(x)$ is corresponding to

$$(r_{00}, r_{01}, \dots r_{0,n-k-1} : 1,0, \dots, 0)$$

Generator Matrix of Systematic Cyclic Codes

- Generator matrix of an (n,k) systematic code is given by

$$
\begin{aligned}
\mathbf{G} &= \begin{bmatrix} r_{00} & r_{01} & r_{02} & \cdots & r_{0,n-k-1} & : & 1 & 0 & \cdots & 0 \\ r_{10} & r_{11} & r_{12} & \cdots & r_{1,n-k-1} & : & 0 & 1 & \cdots & 0 \\ \vdots & & & \vdots & & : & & & \vdots & \\ r_{k-1,0} & r_{k-1,1} & r_{k-1,2} & \cdots & r_{k-1,n-k-1} & : & 0 & 0 & \cdots & 1 \end{bmatrix} \\
&= [\mathbf{P} : \mathbf{I}_k]
\end{aligned}
$$

- Corresponding parity check matrix is $[I_{n-k}\colon P^T]$
- Example : [7,4] cyclic codes generated by $g(x) = x^3 + x + 1$

$$
\left.\begin{aligned}
x^3 &= 1 \cdot g(x) + x + 1 \\
x^4 &= xg(x) + x^2 + x \\
x^5 &= (x^2+1)g(x) + x^2 + x \\
x^6 &= (x^3+x+1)g(x) + x^2 + 1
\end{aligned}\right\} \longrightarrow \left\{\begin{aligned}
&1 + x + x^3 \\
&x + x^2 + x^4 \\
&1 + x + x^2 + x^5 \\
&1 + x^2 + x^6
\end{aligned}\right.
$$

Generator Matrix of Systematic Cyclic Codes

- Generator matrix in the systematic form
  - Note: Left and right , up and down are reversed from the previous representation.

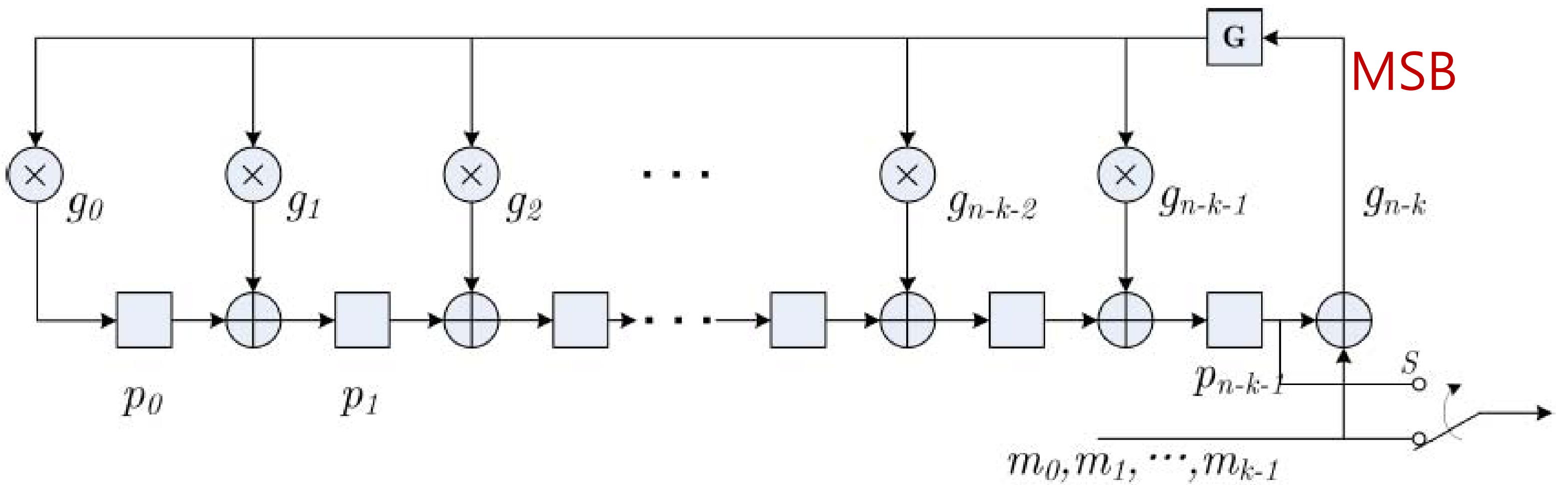
$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & : & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & : & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & : & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & : & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Parity check matrix in the systematic form

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & : & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & : & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & : & 0 & 1 & 1 & 1 \end{bmatrix}$$

Encoder Implementation

- Systematic encoder using division circuit
  - Polynomial division can easily be implemented by a simple shift register circuit



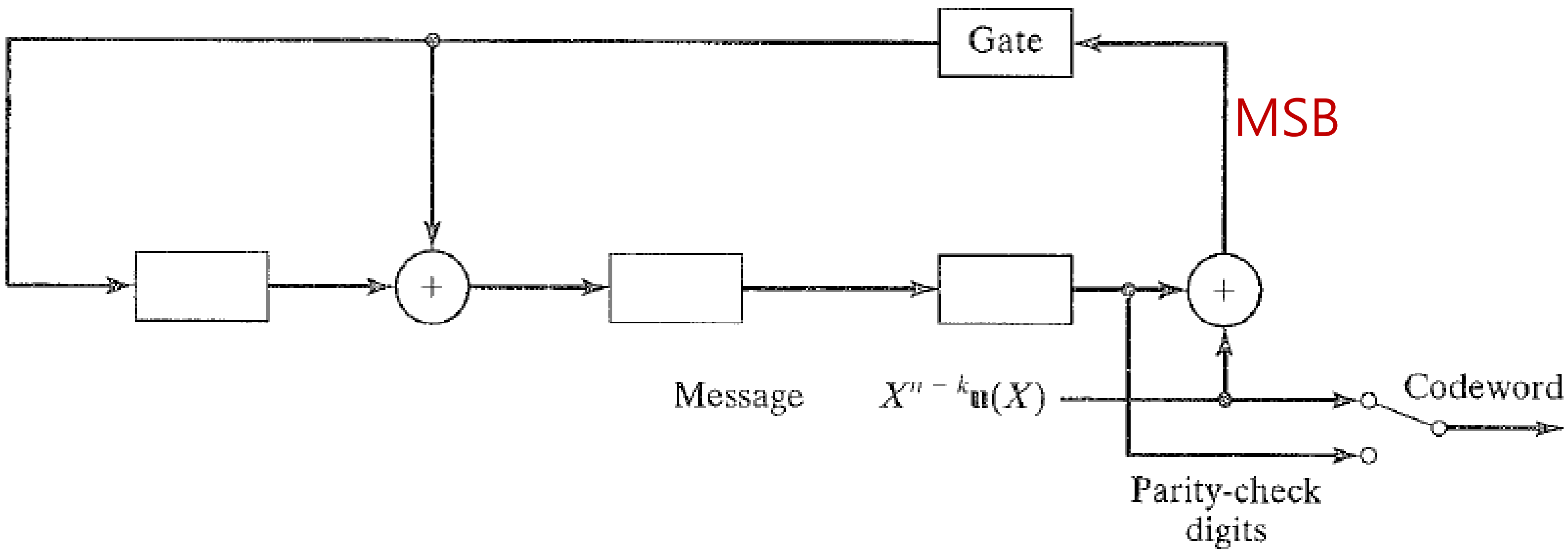
1. First k clocks : G is closed. S is switched down. m_i's come in.
2. Next $n-k$ clocks : G is open. S is switched up. Parity bits comes out.

Encoder Implementation

- Remarks
  - Non-systematic codes are constructed by (generator matrix) multiplication.
  - Systematic code is encoded by division.
  - Encoding is simply to compute the remainder polynomial.
  - Division operations can be conducted by simple symbol operations. (modulo 2 additions for binary codes)
  - Practical encoding is implemented with the systematic encoding with polynomial division.

Encoder of [7,4] Hamming Codes

- (Systematic) Encoder of [7,4] Hamming code with generator polynomial $x^3 + x + 1$



Decoding of Cyclic Codes

Syndrome of a Cyclic Code

- Received polynomial : $y(x) = c(x) + e(x)$
  - Codeword

$$c(x) = (\underbrace{c_0, c_1, \ldots, c_{n-k-1}}_{\text{parity check}} : \underbrace{c_{n-k} \ldots c_{n-1}}_{\text{Information}}) \quad \text{MSB}$$

  - Parity part : $p(x)$ (of max. degree $n-k-1$)
  - Information part : $c_{n-k}x^{n-k} + \cdots + c_{n-1}x^{n-1} = x^{n-k}m(x)$
  - Received

$$\begin{aligned} y(x) &= \underbrace{m(x)x^{n-k} + p(x)}_{c(x)} + \underbrace{e_m(x)x^{n-k}}_{\text{error in information part}} + \underbrace{e_p(x)}_{\text{errors in parity}} \\ &= \underbrace{[m(x) + e_m(x)]x^{n-k}}_{\text{received information}} + \underbrace{[p(x) + e_p(x)]}_{\text{received parity}} \end{aligned}$$

Syndrome of a Cyclic Code

- Syndrome polynomial

$$
\begin{aligned}
s(x) &= r(x) \mod g(x) \\
&= \underbrace{([m(x) + e_m(x)]x^{n-k} \mod g(x))}_{\text{parity from the received information}} + \underbrace{p(x) + e_p(x)}_{\text{received parity}}
\end{aligned}
$$

Example of Syndrome Calculation

- Example : Cyclic code generated by $g(x) = x^3 + x + 1$
  - Received sequence : $\boldsymbol{r} = (1010110)$

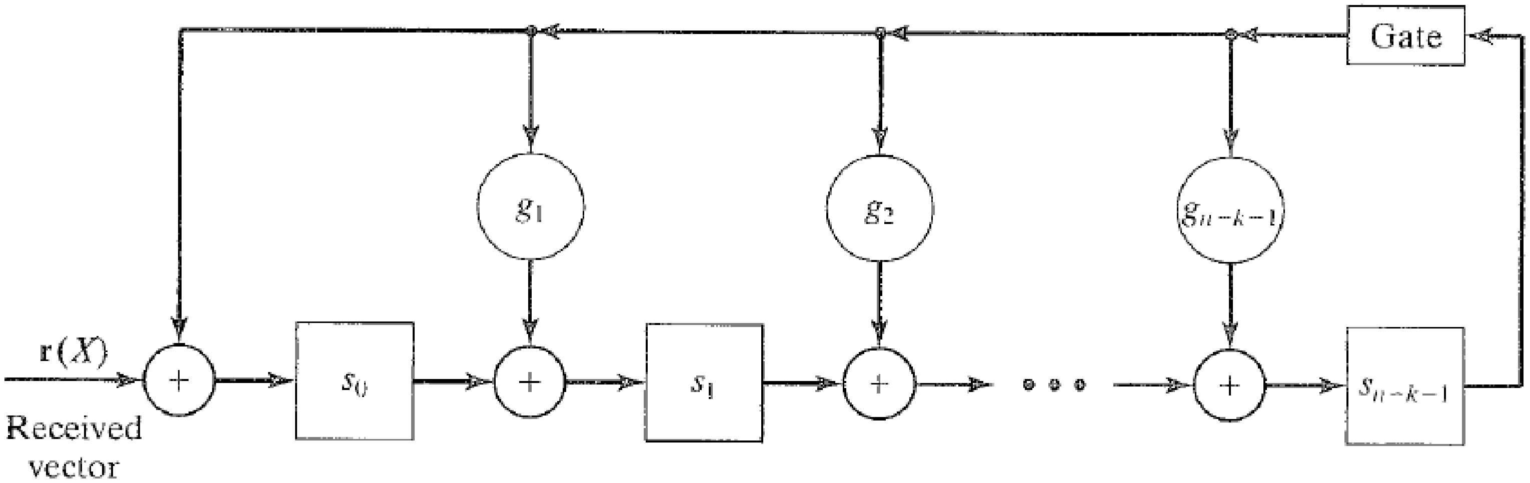
$$r(x) = 1 + x^2 + x^4 + x^5$$

  - Syndrome : $\boldsymbol{s} = Hr^T \leftrightarrow s(x) = r(x) \bmod g(x)$
  - Above two operations are equivalent
    - $Hr^T = (0\ 0\ 1)^T$
    - $r(x) \bmod g(x) = x^2$

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & : & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & : & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & : & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & : & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & : & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & : & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & : & 0 & 1 & 1 & 1 \end{bmatrix}$$

Syndrome Computing Circuit

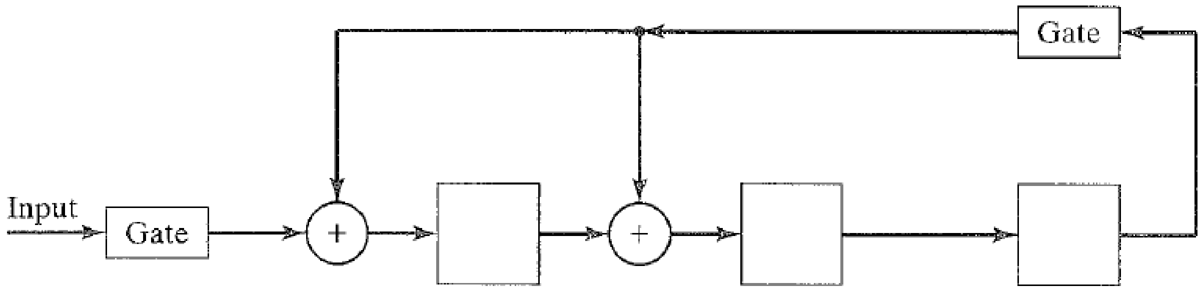
- Syndrome: $s(x) = r(x) \bmod g(x)$



  - A similar division circuit as the encoder circuit
  - Difference is that $r(x)$ comes in from the left side.
  - Memory units (registers) are initially set to zero.
  - $s(x)$'s coefficients are the values stored in the memories after the last symbol comes into s_0.

Syndrome Computing Circuit

- Example
  - Syndrome circuit of [7,4] Hamming code

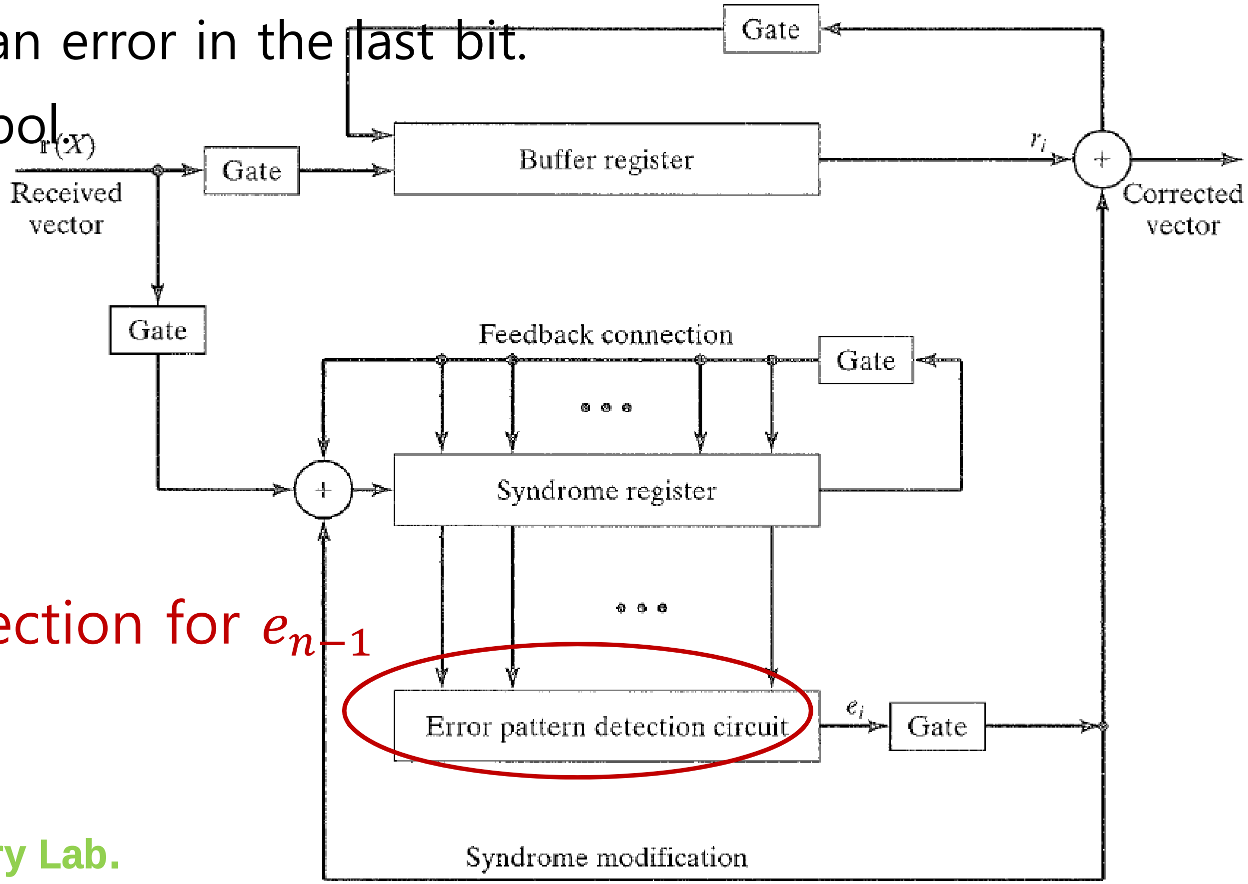


  - $\boldsymbol{r} = (0\ 1\ 1\ 0\ 1\ 0\ 0)$ (MSB on left)

Shift	Input	Register contents
		0 0 0 (initial state)
1	0	0 0 0
2	1	1 0 0
3	1	1 1 0
4	0	0 1 1
5	1	0 1 1
6	0	1 1 1
7	0	1 0 1 (syndrome s)
8	—	1 0 0 (syndrome $s^{(1)}$)
9	—	0 1 0 (syndrome $s^{(2)}$)

General Cyclic Code Decoder (Meggit Dec.)

- Detect error patterns, using error detection circuits one at a time by cyclically shifting $\boldsymbol{r}$.
- Use the fact if $s(x) = r(x) \bmod g(x)$ then $xr(x) \bmod g(x) = xs(x) \bmod g(x)$. So a cyclic class of error patterns can be detected by one error pattern detector.
- Error patterns detected has an error in the last bit.
- If detected, flip the last symbol.
  One symbol is corrected.
- Decoder proceeds
  with the remained errors

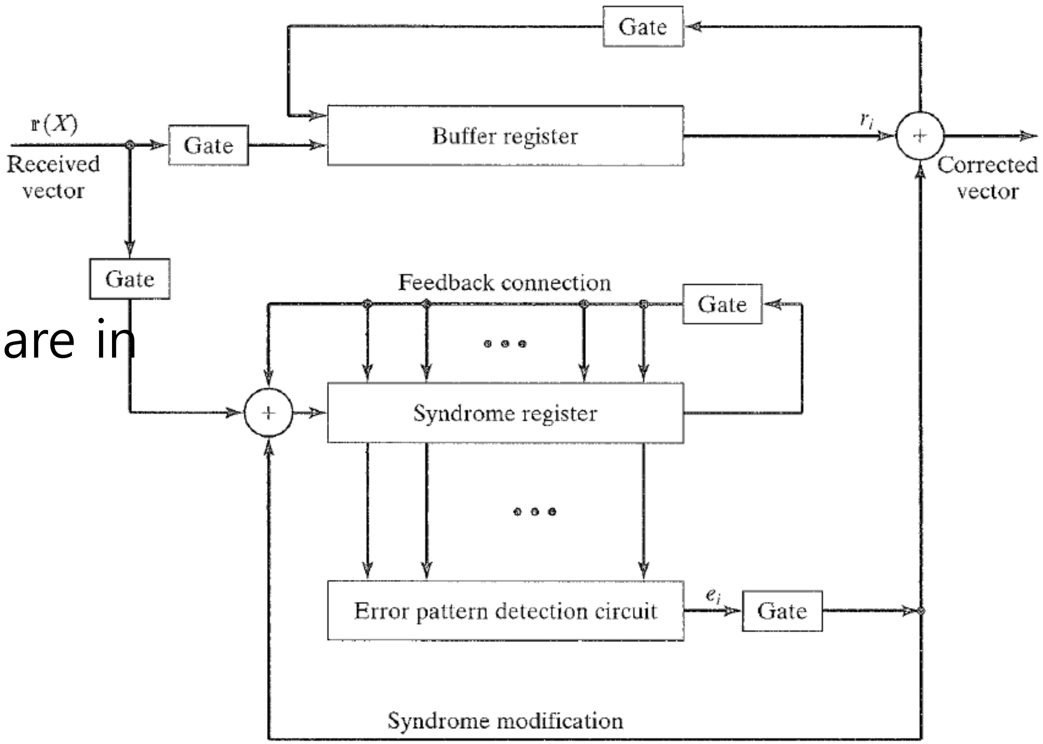


Error detection for e_{n-1}

General Cyclic Code Decoder (Meggit Dec.)

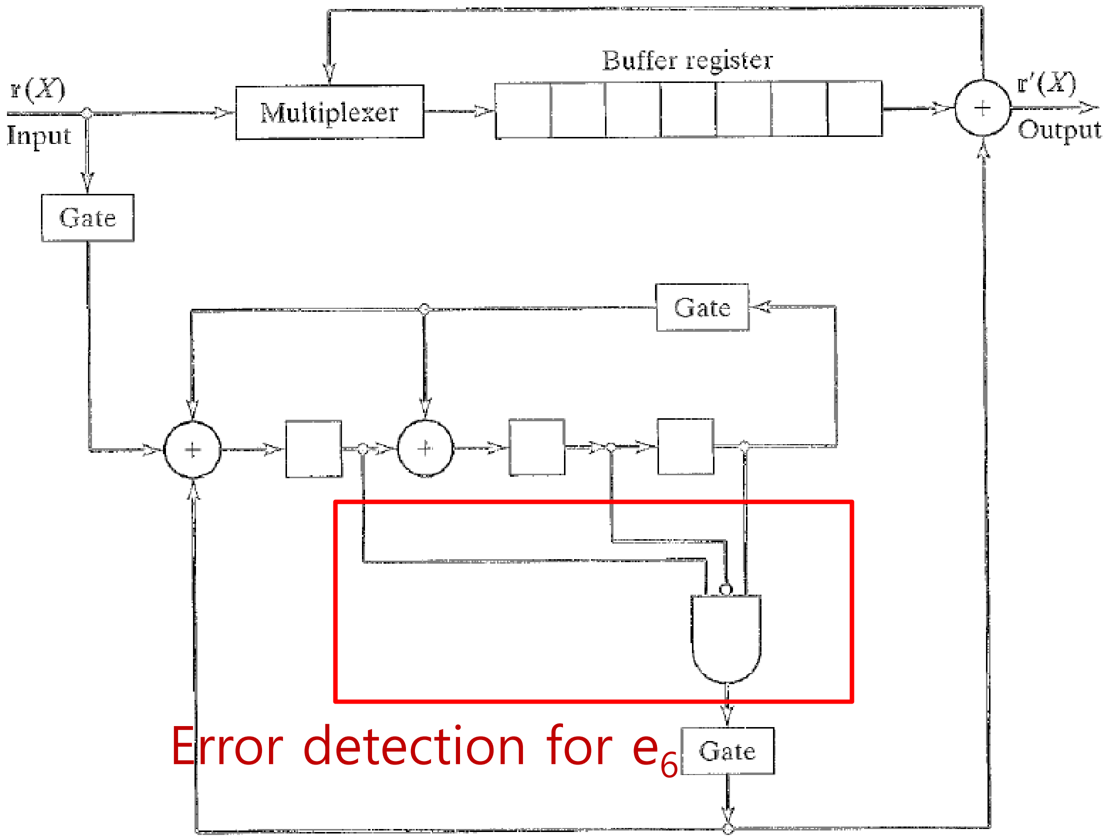
- Procedure
  - Put the received vector from the left.
  - After $n-1$ clocks we check e_{n-1} using the detection circuit.
  - For following $n-1$ clocks e_i, for $i < n-1$, are checked.
  - During the time, e_i are added (subtracted) to r_i for correction.

- Note
  - The errors are corrected within the error correction capability.
  - E.g.: if $t = 1$ and e_{n-1} and e_{n-2} are in error then only another e_i is detected. (becomes 1)



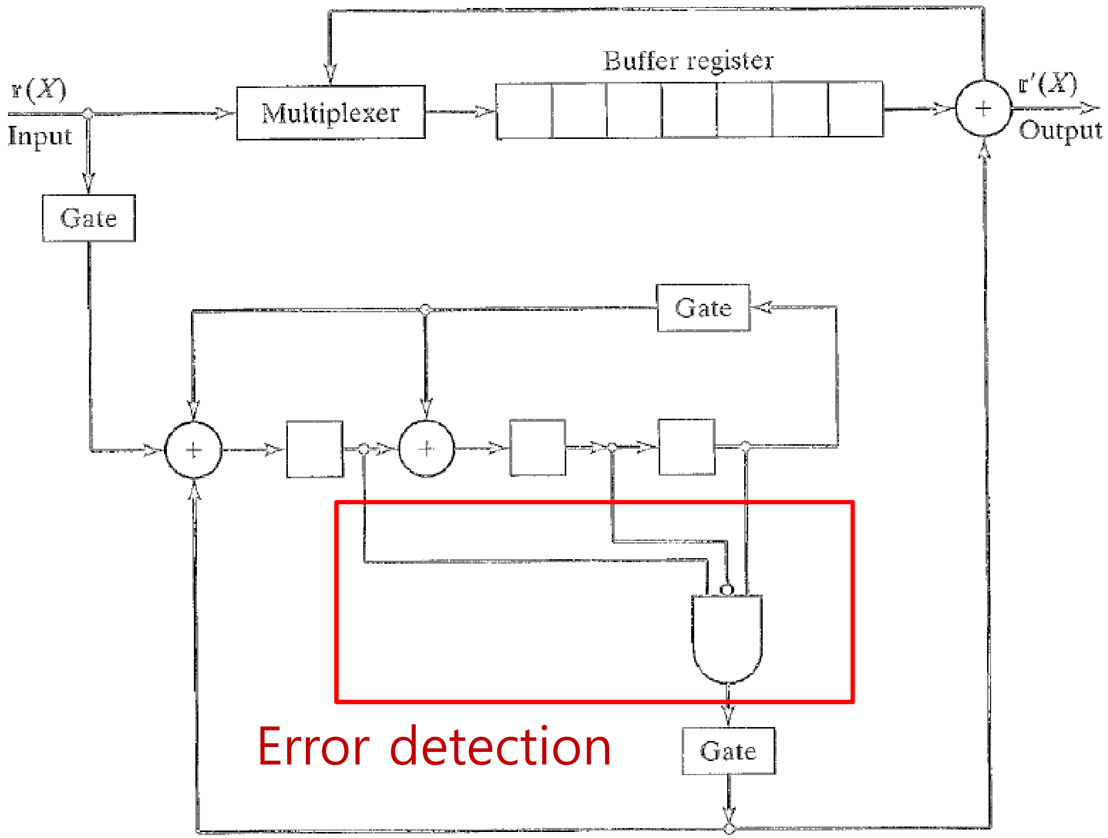
Meggit Decoder of [7,4] Hamming Code

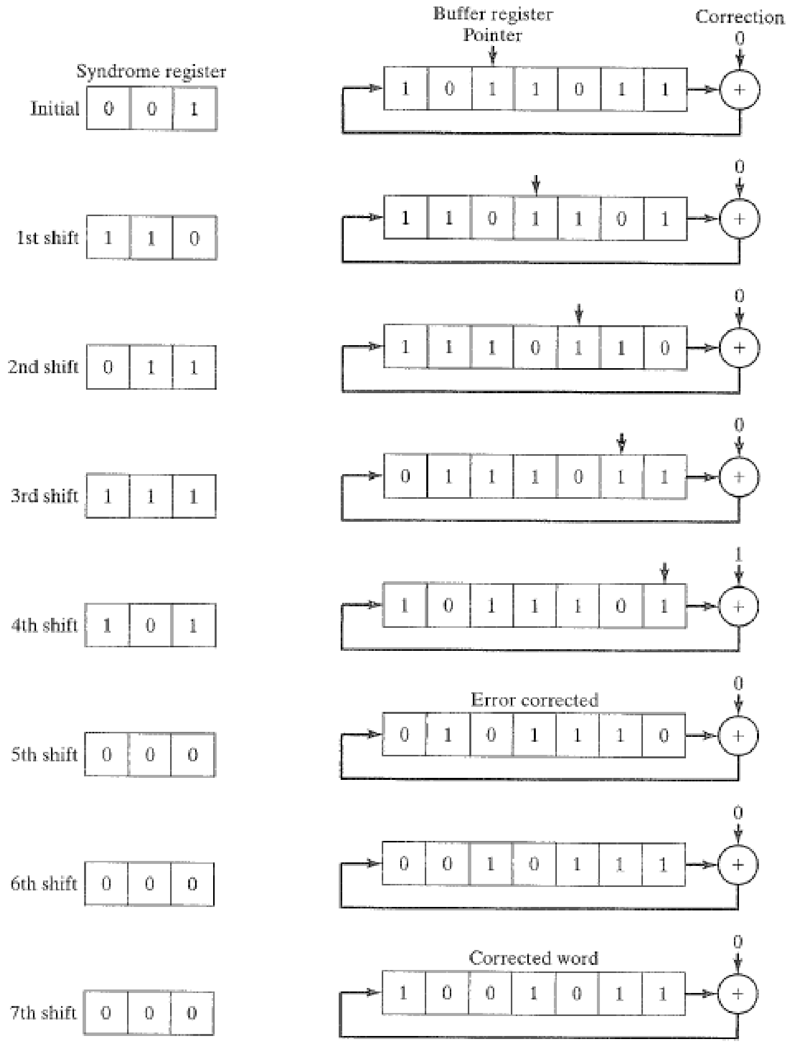
Error pattern $e(X)$	Syndrome $s(X)$	Syndrome vector (s_0, s_1, s_2)
$e_6(X) = X^6$	$s(X) = 1 + X^2$	(1 0 1)
$e_5(X) = X^5$	$s(X) = 1 + X + X^2$	(1 1 1)
$e_4(X) = X^4$	$s(X) = X + X^2$	(0 1 1)
$e_3(X) = X^3$	$s(X) = 1 + X$	(1 1 0)
$e_2(X) = X^2$	$s(X) = X^2$	(0 0 1)
$e_1(X) = X^1$	$s(X) = X$	(0 1 0)
$e_0(X) = X^0$	$s(X) = 1$	(1 0 0)



Meggit Decoder of [7,4] Hamming Code

- Decoding example (MSB left)
- r (=1101101)= c (=1101001)+ e(=0000100)





Examples of Cyclic Codes

Linear Codes

Cyclic Redundancy Check

Hamming Codes

- Hamming codes

$$
\begin{aligned}
n &= 2^m - 1,\ k = n - m\,,\ d = 3 \\
g(x) &= \text{primitive polynomial of degree } m \\
h(x) &= (x^n + 1)/g(x),\ \text{check polynomial} \\
& \deg h(x) = 2^m - 1 - m
\end{aligned}
$$

- Example : [7,4] Hamming code

$$
\begin{aligned}
g(x) &= x^3 + x + 1, \\
h(x) &= x^4 + x^2 + x + 1
\end{aligned}
$$

Simplex Codes

- Simplex codes

$$
\begin{aligned}
n &= 2^m - 1,\ k^{\perp} = m,\ d = 2^{m-1} \\
g^{\perp}(x) &= h^*(x) = h_k + h_{k-1}x + \cdots + h_1 x^{k-1} + h_0 x^k \\
h^{\perp}(x) &= g^*(x) = (x^n + 1)/h^*(x)
\end{aligned}
$$

- Example : [7,3] simplex code

$$
\begin{aligned}
g^{\perp}(x) &= x^4 + x^3 + x^2 + x + 1 = h^*(x) \\
h^{\perp}(x) &= x^3 + x^2 + 1 = g^*(x)
\end{aligned}
$$

$$
\mathbf{G}^{\perp} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}
$$

  - Codewords

$$
(0000000), (1011100), (0101110), (0010111)
$$

$$
(1001011), (1100101), (1110010), (0111001)
$$

Golay Codes

- [23,12,7] Golay codes example

$$
\begin{aligned}
t &= 3\ \text{"triple error correcting codes"} \\
x^{23}+1 &= (x+1)g_1(x)g_2(x) \\
g_1(x) &= 1+x^2+x^4+x^5+x^6+x^{10}+x^{11} \\
g_2(x) &= 1+x+x^5+x^6+x^7+x^9+x^{11} = g_1^*(x)
\end{aligned}
$$

  - $g_1(x)$ *or* $g_2(x)$ may be used as a generator polynomial of Golay code
- Extended Golay code (Rate=1/2)
  - With an even parity bit

$$
\mathbf{H}_E = \begin{bmatrix} & & & \vdots & 0 \\ & & & \vdots & 0 \\ & \mathbf{H} & & \vdots & \vdots \\ & & & \vdots & 0 \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ 1\,1\,1 & \cdots & 1 & \vdots & 1 \end{bmatrix}
$$

Link: Golay's Original Paper

※ Note: The paper on Golay code was referred to as 'the best single published page in coding theory' by Erwin Berlekamp

Other Examples

- Shortened cyclic codes

$$[n,k] \rightarrow [n-s,k-s]$$

  - not a cyclic code after shortening

- Expurgated cyclic codes
  - Example
    - a new code has the generator polynomial $g_1(x) = (1+x)g(x)$
      $\Rightarrow$ Every Codeword has even weight
    - All odd weight codewords are thrown out
  - General expurgation
    - $g_1(x) = f(x)g(x)$ for $f(x)$ $(\gcd(f(x), g(x)) = 1)$

Cyclic Redundancy Check

- Cyclic redundancy check codes
  - Error detection codes used in digital communication networks and storage devices to detect changes in raw data.
    - In general, $1 - 2^{-m}$ portion of errors are detected. Good for burst errors.
  - Very popular because they are easy to implement and analyze.
  - Can also be used for error detection or a hash function.
  - Invented by William W. Peterson in 1961.
- Technical aspects
  - Shortened cyclic codes
  - Codeword length varies according to the length of the message
  - The number of parities is kept the same.
  - Non-cyclic codes since it is shortened usually.
  - Has a generator polynomial $g(x)$ degree $m = n - k$

Wesley Peterson (wiki)

CRC as a Cyclic Code: Even Parity Check Code

- Data $\boldsymbol{D} = (d_0, d_1, \dots, d_{k-2}, d_{k-1})$ where k is the data length
- Parity bit p is added

$$D = (p, d_0, d_1, \dots, d_{k-2}, d_{k-1}) *$$

such that $d_0 + d_1 + \cdots + p = 0$

- Received data
  - Even parity ⇒ no error or undetectable
  - Odd parity ⇒ errors

CRC as a Cyclic Code: Even Parity Check Code

- Polynomial representation

$$\begin{aligned} g(x) &= x+1 \\ D(x) &= d_0 + d_1 x + d_2 x^2 + \cdots + d_{k-1} x^{k-1} \\ xD(x) &= q(x)g(x) + p(x) \end{aligned}$$

$$\deg p(x) = 0 \Rightarrow p(x) = p$$

  - Therefore, $xD(x) + p(x) = q(x)g(x)$

$$p + d_0 x + d_1 x^2 + \cdots + d_{k-1} x^k = q(x)g(x)$$

$$g(x) | (xD(x) + p(x))$$

Since $(xD(x) + p(x))|_{x=1} = 0$, single error is detectable

CRC as a Cyclic Code: General Form

- Generator polynomial

$$g(x) = g_0 + g_1 x + \cdots + g_{m-1} x^{m-1} + g_m x^m$$

- Data polynomial

$$\begin{aligned} D(x) &= d_0 + d_1 x + d_2 x^2 + \cdots + d_{k-1} x^{k-1} \\ x^m D(x) &= q(x) g(x) + p(x) \end{aligned}$$

where $q(x)$ is the quotient polynomial and $r(x)$ is the remainder $(\deg p(x) \leq m-1)$

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{m-1} x^{m-1}$$

$$x^m D(x) + p(x) = q(x) g(x)$$

$$p_0 + \cdots + p_{m-1} x^{m-1} + d_0 x^m + d_1 x^{m+1} + \cdots + d_{k-1} x^{m+k-1} = q(x) g(x)$$

  - Codeword $\boldsymbol{c} = (p_0, \ldots, p_{m-1}, d_0, d_1, \ldots, d_{k-1}, d_k)^*$
  - The number of errors we can detect : m

Period of $g(x)$ and Error Polynomial

- Period of polynomial $g(x)$
  - The least positive integer e such that $x^e + 1$ is divisible by $g(x)$
- Received polynomial
  - $r(x) = c(x) + e(x)$ $(r_0, r_1, \cdots, r_{k+m-1})$
- Record length of bit errors
  - Length of the duration from the first to the last bit error

Basic Properties

- Theorem
  - All single bit errors are detected by CRC with gen. $x^c + 1$, $c > 0$.
  - Example: $\boldsymbol{e} = (0\ 0\ 1\ 0\ 0\ 0)$
- Theorem
  - All causes of an odd number of bits in error are detected by a code with generator $x^c + 1$, (e.g. $x + 1$), $c > 0$.
  - Example: $\boldsymbol{e} = (0\ 1\ 1\ 0\ 1\ 0)$
- Theorem
  - A code detect all double error patterns if the record length is not greater than the period of the generator polynomial.
  - Example: $\boldsymbol{e} = (0\ \underbrace{1\ 0\ 0\ 0\ 0\ 1}_{\text{Record length: 6}}\ 0\ 0\ 0\ 0\ 0\ 0\ )$ if the period is 7.

Basic Properties

- Theorem
  - A code with generator of degree m detects all single burst errors of a length not greater than m.
- Theorem
  - A code detects all single-, double-, and triple errors iff the generator polynomial is of the form $(x^c + 1)a(x)$ and the record length is not greater than the period of $g(x)$.
- Theorem
  - A code with $g(x) = (x^c + 1)a(x)$ has a guaranteed double burst error capability provided the record length is not greater than the period of the generator polynomial.
  - The code will detect any combination of double bursts when the length of shorter burst is not greater than the degree of $a(x)$ and the sum of the burst lengths is not greater than $c + 1$.

Examples of CRC

- Example: Generator polynomial of CRC-CCITT code

$$\begin{aligned} g(x) &= x^{16} + x^{12} + x^5 + 1 \\ &= (x+1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1) \end{aligned}$$

  - Period of $g(x)$ is 32767
  - Error detection probability
    - All odd number of errors
    - All single, double, and triple errors if record length is ≤ 32767
    - All single burst errors of 16 bit, or less
    - Detect 99.99695% of all possible burst of length 17, and 99.99847% of all possible longer burst.

※ Note: If an error $e(x)$ is not divisible by $g(x)$, then $e(x)$ can be detected.

Examples of CRC

- Popularly used CRC codes
  - CRC-7 : $g(x) = x^7 + x^6 + x^4 + 1$
  - CRC-8 : $g(x) = (x^5 + x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x + 1)$
  - CRC-12: $g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - CRC-ANSI : $g(x) = x^{16} + x^{12} + x^2 + 1$
  - CRC-CCITT: $g(x) = x^{16} + x^{12} + x^5 + 1$

- Note
  - Primitive polynomial is not the best CRC generator, although this has the maximum period.
  - At the price of reduction of period, the CRC can cover more error patterns.
  - Many works had been done in mid 70's

CRC implementation: CRCencoder()


```
void crcEncoder(int *in, int *out, int N) {
  int n, m;
  int mem[16], fb;
  int nCRC;

  nCRC = 16; // for CRC CCITT
  //crc initialized
  for(n=0;n<16;n++) mem[n] = 0;
  for(n=0;n<N;n++) { // CCITT g(x) = X^16 + X^12 + X^5 + 1
  fb = (mem[15]+in[n])%2;
  mem[15] = mem[14];
  mem[14] = mem[13];
  mem[13] = mem[12];
  mem[12] = (fb+mem[11])%2;
  mem[11] = mem[10];
```

CRC implementation: CRCencoder()


```
  mem[10] = mem[9];
  mem[9] = mem[8];
  mem[8] = mem[7];
  mem[7] = mem[6];
  mem[6] = mem[5];
  mem[5] = (fb+mem[4])%2;
  mem[4] = mem[3];
  mem[3] = mem[2];
  mem[2] = mem[1];
  mem[1] = mem[0];
  mem[0] = fb;
}
for(n=0;n<N;n++) out[n] = in[n];// data part
for(n=0;n<nCRC;n++) out[N+n]=mem[nCRC-n-1];// parity part
```

CRCdecoder()


```
int crcDecoder(int *in, int N) {
  int n, m;  int mem[16], fb;
  int nCRC;

  nCRC = 16; // for CRC CCITT
  //crc initialized
  for(n=0;n<16;n++) mem[n] = 0;
  for(n=0;n<N;n++) { // CCITT g(x) = X^16 + X^12 + X^5 + 1
    fb = (mem[15]+in[n])%2;
    mem[15] = mem[14];
    mem[14] = mem[13];
    mem[13] = mem[12];
    mem[12] = (fb+mem[11])%2;
    mem[11] = mem[10];
    mem[10] = mem[9];
    mem[9] = mem[8];
```

CRCdecoder()


```
    mem[8] = mem[7];
    mem[7] = mem[6];
    mem[6] = mem[5];
    mem[5] = (fb+mem[4])%2;
    mem[4] = mem[3];
    mem[3] = mem[2];
    mem[2] = mem[1];
    mem[1] = mem[0];
    mem[0] = fb;
}
for(n=0;n<nCRC;n++)
if (in[N+n]!=mem[nCRC-n-1])
  return 1;// CRC bad
return 0; //CRC good

}
```