S.72-3320 Advanced Digital Communication (4 cr)

Cyclic Codes

Targets today

- Taxonomy of coding
- How cyclic codes are defined?
- Systematic and nonsystematic codes
- Why cyclic codes are used?
- How their performance is defined?
- How practical encoding and decoding circuits are realized?
- How to construct cyclic codes?

Taxonomy of Coding

Error Correction Coding

= FEC
- no feedback channel
- quality paid by redundant bits

Error Detection Coding

- used in ARQ as in TCP/IP
- feedback channel
- retransmissions
- quality paid by delay

Cryptography (Ciphering)

- Secrecy/ Security
- Encryption (DES)

Source Coding

- Makes bits equal probable

Error Control Coding

- Strives to utilize channel capacity by adding extra bits

Line Coding

- for baseband communications
- RX synchronization
- Spectral shaping for BW requirements
- error detection

Compression Coding

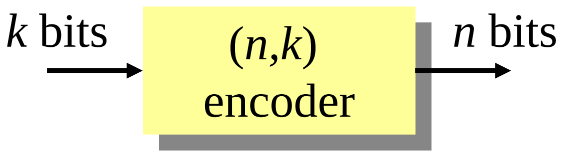
- Redundancy removal:
  - Destructive (jpeg, mpeg)
  - Non-destructive (zip)

FEC: Forward Error Correction
ARQ: Automatic Repeat Request
DES: Data Encryption Standard

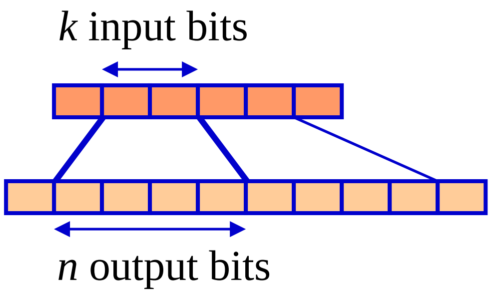
Background

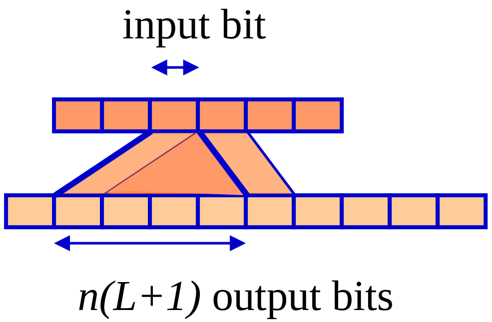
- Coding is used for
  - error detection and/or error correction (channel coding)
  - ciphering (security) and compression (source coding)
- In coding extra bits are added or removed in data transmission
- Channel coding can be realized by two approaches
  - FEC (forward error coding)
    - block coding, often realized by cyclic coding
    - convolutional coding
  - ARQ (automatic repeat request)
    - stop-and-wait
    - go-back-N
    - selective repeat … etc.
- Note: ARQ applies FEC for error detection

Block and convolutional coding



- Block coding: mapping of source bits of length *k* into (binary) channel input sequences *n (>k) - realized by cyclic codes!*
- Binary coding produces 2^k code words of length *n*. Extra bits in the code words are used for error detection/correction
- (1) block, and (2) convolutional codes:
  - (n,k) block codes: Encoder output of *n* bits depends only on the *k* input bits
  - (n,k,L) convolutional codes:
    - *each source bit* influences *n(L+1)* encoder output bits
      - *n(L+1)* is the constraint length
      - *L* is the memory depth
- Essential difference of block and conv. coding is in <u>simplicity</u> of design of encoding and decoding circuits





Why cyclic codes?

- For practical applications rather large n and k must be used. This is because in order to correct up to t errors it should be that

$$2^{n-k}-1 \approx \underbrace{\binom{n}{1}}_{=n} + \binom{n}{2} + \ldots + \binom{n}{t} = \sum_{i=1}^{t} \binom{n}{i}$$

number of syndromes (or check-bit error patterns)

number of error patters in encoded word

$$\Rightarrow 1 - R_C \approx \frac{1}{n} \log_2 \left[\sum_{i=1}^{t} \binom{n}{i} \right] \quad \text{note: } q = n - k = n(1 - R_C)$$

- Hence for $R_C = k/n \approx 1$, large n and k must be used (next slide)

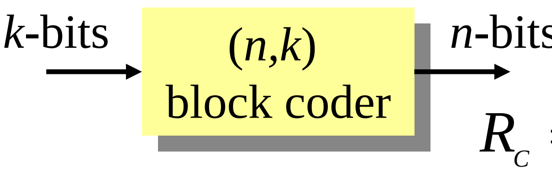
k-bits → (n,k) block coder → n-bits

- Cyclic codes are
  - **linear**: sum of any two code words is a code word
  - **cyclic**: any cyclic shift of a code word produces another code word
- Advantages: Encoding, decoding and syndrome computation easy by <u>shift registers</u>

Example

- Consider a relatively high SNR channel such that only 1 or 2 bit errors are likely to happen. Consider the ration

$$1 - R_c \approx \frac{1}{n}\log_2\left[\sum_{i=1}^{t}\binom{n}{i}\right]$$



$$R_C = k/n$$

$$\varepsilon(n,k) \triangleq \frac{n-k}{\log_2\left[\binom{n}{1}+\binom{n}{2}\right]} = \frac{\text{Number of check-bits}}{\text{Number of 2-bit error patterns}}$$

- Take a constant code rate of $R_c = k/n = 0.8$ and consider ε with some values of larger n and k :

$$\varepsilon(10,8) = 0.35, \varepsilon(32,24) = 0.89, \varepsilon(50,40) = 0.97$$

- This demonstrates that long codes are more advantages when a high code rate and high error correction capability is required

Some block codes that can be realized by cyclic codes

- (n,1) **Repetition codes**. High coding gain (minimum distance always *n*-1), but very low rate: 1/*n*
- (n,k) **Hamming codes**. Minimum distance always 3. Thus can detect 2 errors and correct one error. $n=2^m-1$, $k = n - m$, $m \geq 3$
- **Maximum-length codes**. For every integer $k \geq 3$ there exists a maximum length code (*n*,*k*) with $n = 2^k - 1, d_{min} = 2^{k-1}$.
- **BCH-codes**. For every integer $m \geq 3$ there exist a code with $n = 2^m$-1, $k \geq n - mt$ and $d_{\min} \geq 2t+1$ where *t* is the error correction capability
- (*n*,*k*) **Reed-Solomon (RS) codes**. Works with *k* **symbols** that consists of *m* bits that are encoded to yield code words of *n* **symbols**. For these codes $n = 2^m - 1$, number of check symbols $n - k = 2t$ and $d_{\min} = 2t+1$
- Nowadays BCH and RS are very popular due to large d_{min}, large number of codes, and easy generation
- Code selection criteria: **number of codes, correlation properties, code gain, code rate, error correction/detection properties**

1: Task: find out from literature what is meant by dual codes!

Defining cyclic codes: code polynomial and generator polynomial

- An (n,k) linear code $\mathbf{X}$ is called a cyclic code when every <u>cyclic shift</u> of a code $\mathbf{X}$, as for instance $\mathbf{X}'$, is also a code, e.g.

$$\mathbf{X} = (x_{n-1} x_{n-2} \cdots x_1\, x_0) \Rightarrow \quad \mathbf{X}' = (x_{n-2} x_{n-3} \cdots x_0\, x_{n-1})$$

- Each (n,k) cyclic code has the associated code vector with the n-bit code polynomial

$$\mathbf{X}(p) = x_{n-1} p^{n-1} + x_{n-2} p^{n-2} + \cdots + x_1 p + x_0$$

$$\mathbf{X}'(p) = x_{n-2} p^{n-1} + x_{n-3} p^{n-2} + \cdots + x_0 p + x_{n-1}$$

- Note that the (n,k) code vector has the <u>polynomial of degree of n-1 or less.</u> Mapping between code vector and code polynomial is one-to-one, e.g. they specify each other uniquely
- Manipulation of the associated polynomial is done in a *Galois field* (for instance GF(2)) having elements {0,1}, where operations are performed mod-2. <u>Thus results are always {0,1}</u> -> binary logic circuits applicable
- For each cyclic code, there exists only one <u>generator polynomial</u> whose degree equals the number of check bits $q=n-k$ in the encoded word

Example: Generating of (7,4) cyclic code, by generator polynomial G$(p)=p^3+p+1$

$$\mathbf{M}=(1101)=p^3+p^2+1 \quad \text{<- message}$$

$$\mathbf{G}=(1011)=p^3+p+1 \quad \text{<- generator}$$

$$\mathbf{X}=\mathbf{MG}=p^3(p^3+p^2+1)+p(p^3+p^2+1)+p^3+p^2+1$$

$$=p^6+p^5+\cancel{p^3}+p^4+\cancel{p^3}+p+p^3+p^2+1$$

$$=p^6+p^5+p^4+p^3+p^2+p+1=(1111111) \quad \text{<- encoded word}$$

The same result obtained by Maple:

```
> expand((x^3+x^2+1)*(x^3+x+1)) mod 2;
```

$$x^6+x^4+x^3+x^5+x^2+x+1$$

```
>
```

Rotation of cyclic code yields another cyclic code

- Theorem: A single cyclic shift of **X** is obtained by multiplication of $p\mathbf{X}$ where after division by the factor p^n+1 yields a cyclic code at the remainder:

$$\mathbf{X}'(p) = p\mathbf{X}(p)\,\mathrm{mod}(p^n+1)$$

and by induction, any cyclic shift *i* is obtained by

$$\boxed{\mathbf{X}^{(i)}(p) = p^{(i)}\mathbf{X}(p)\,\mathrm{mod}(p^n+1)}$$

$$\begin{array}{r|l} & 1 \\ \hline p^3+1 & p^3+p \\ & \underline{p^3+1} \\ & p+1 \leftarrow \end{array}$$

- Example: $101 \rightarrow \mathbf{X}(p) \rightarrow p^2+1$

Shift left by 1 bit: $p\mathbf{X}(p) \rightarrow p^3+p$ $\leftarrow$ not a three-bit code (1010), divide by the common factor

$$\frac{p\mathbf{X}(p)}{p^3+1} = 1 + \frac{p+1}{p^3+1} \rightarrow 011$$ *n*-1 bit *rotated* code word

- Important point of implementation is is that the division by p^n+1 can be realized by a tapped shift register.

```
> factor((1+(p+1)/(p^3+1))*(p^3+1)) mod 2;
                    (p + 1) (p^2 + p)
> expand(%) mod 2;
                    p^3 + p
>
```

- Prove that

$$\mathbf{X}'(p) = p\mathbf{X}(p)\,\mathrm{mod}(p^n+1) \qquad (1)$$

Note first that

$$\mathbf{X}(p) = x_{n-1}p^{n-1} + x_{n-2}p^{n-2} + \cdots + x_1 p + x_0$$

$$p\mathbf{X}(p) = x_{n-1}p^{n} + x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p \qquad (2)$$

then, by using (1) and (2)

$$\begin{array}{r|l} & x_{n-1} \\ \hline p^n+1 & x_{n-1}p^n + x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p \\ & x_{n-1}p^n + x_{n-1} \\ \hline & x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p + x_{n-1} \leftarrow \mathbf{X}'(p) \end{array}$$

- Repeating the same division with higher degrees of p yields then

$$\mathbf{X}^{(i)}(p) = p^{(i)}\mathbf{X}(p)\,\mathrm{mod}(p^n+1)$$

Cyclic codes and the common factor p^n+1

- Theorem: Cyclic code polynomial **X** can be generated by multiplying the message polynomial **M** of degree *k*-1 by the generator polynomial **G** of degree $q=n-k$ where **G** is an *q*-th order factor of $p^n + 1$.
- Proof: assume message polynomial:

$$\mathbf{M}(p)=m_{k-1}p^{k-1}+m_{k-2}p^{k-2}+\cdots+m_1p+x_0$$

and the *n*-1 degree code is

$$\mathbf{X}(p)=x_{n-1}p^{n-1}+x_{n-2}p^{n-2}+\cdots+x_1p+x_0$$

or in terms of **G**

$$\mathbf{X}(p)=\mathbf{MG}=\mathbf{G}(p)m_{k-1}p^{k-1}+\mathbf{G}(p)m_{k-2}p^{k-2}+\cdots+\mathbf{G}(p)m_1p+\mathbf{G}(p)x_0$$

Consider then a shifted code version…

$$
\begin{aligned}
p\mathbf{X}(p) &= x_{n-1}p^n + x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p \\
&= x_{n-1}(p^n+1) + (x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p + x_{n-1}) \\
&= x_{n-1}(p^n+1) + \mathbf{X}'(p) = p\mathbf{MG}
\end{aligned}
$$

G is a factor of p^n+1

term has the factor p^n+1

must be a multiple of **G**

- Now, if $p\mathbf{X}(p) = p\mathbf{MG}$ and assume **G** is a factor of p^n+1 (not **M**), then **X**'(p) must be a multiple of **G** that we actually already proved:

$$X'(p) = p\mathbf{MG} \bmod (p^n+1)$$

- Therefore, X' can be expressed by $\mathbf{M_1G}$ for some other data vector $\mathbf{M_1}$ and **X**' is must be a code polynomial.
- Continuing this way for $p^{(i)}\mathbf{X}(p)$, i = 2,3… we can see that **X**'', **X**''' etc are all code polynomial generated by the multiplication **MG** of the respective, different message polynomials
- Therefore, the (n,k) linear code **X**, generated by **MG** is indeed cyclic when **G** is selected to be a factor of p^n+1

Cyclic Codes & Common Factor

$$p\mathbf{X}(p) = x_{n-1}p^n + x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p$$
$$= x_{n-1}(p^n + 1) + (x_{n-2}p^{n-1} + \cdots + x_1 p^2 + x_0 p + x_{n-1})$$
$$= x_{n-1}(p^n + 1) + \underbrace{\mathbf{X}'(p)}_{\mathbf{M_1 G}} = p\mathbf{MG}$$

$$21x + 7y = 4 \cdot 21,\ \ 3 \cdot 7 = 21 = 1 + p^2, \mathbf{M} = 7$$
$$x = 1 \Rightarrow y = 3 \cdot 3$$
$$x = 2 \Rightarrow y = 3 \cdot 2$$
$$x = 3 \Rightarrow y = 3 \cdot 1$$

Factoring cyclic code generator polynomial

- **Any factor of p^n+1 with the degree of q=n-k generates an (n,k) cyclic code**
- Example: Consider the polynomial p^7+1. This can be factored as

$$p^7+1=(p+1)(p^3+p+1)(p^3+p^2+1)$$

- Both the factors p^3+p+1 or p^3+p^2+1 can be used to generate an unique cyclic code. For a message polynomial p^2 +1 the following encoded word is generated:

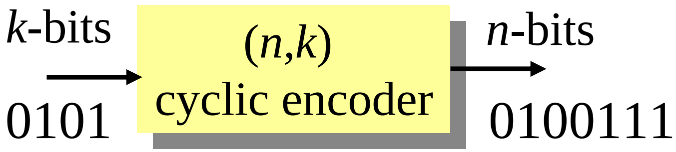
$$(p^2+1)(p^3+p+1)=p^5+p^2+p+1$$

and the respective code vector (of degree n-1 or smaller) is

0100111

- Hence, in this example

$$\begin{cases} q=3=n-k \\ n=7 \Rightarrow k=4 \end{cases}$$



Example of Calculus of GF(2) in Maple

```
> Factor(x^7+1) mod 2;
```

$$(x^3+x+1)(x^3+x^2+1)(1+x)$$

```
> Factor(x^9+1) mod 2;
```

$$(x^6+x^3+1)(x^2+x+1)(1+x)$$

```
> expand(%) mod 2;
```

$$x^9+1$$

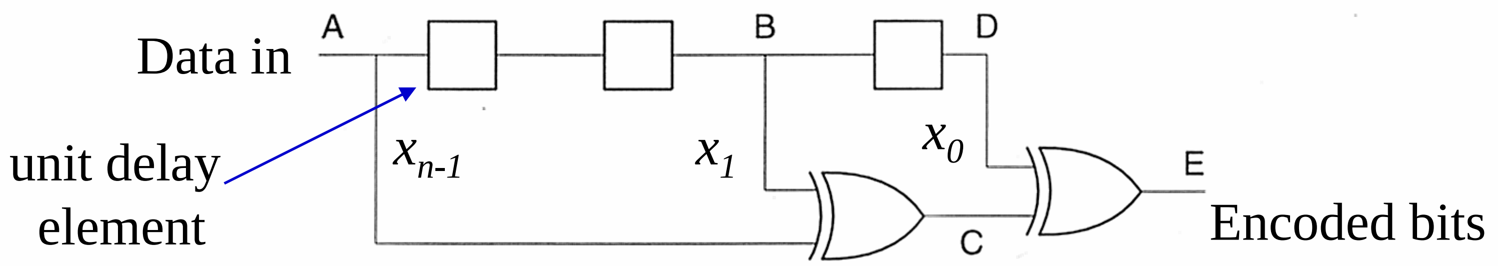
```
> expand((x^3+x+1)*(x^2+1)) mod 2;
```

$$x^5+x+x^2+1$$

```
>
```

Encoder applies shift registers for multiplication of data by the generator polynomial

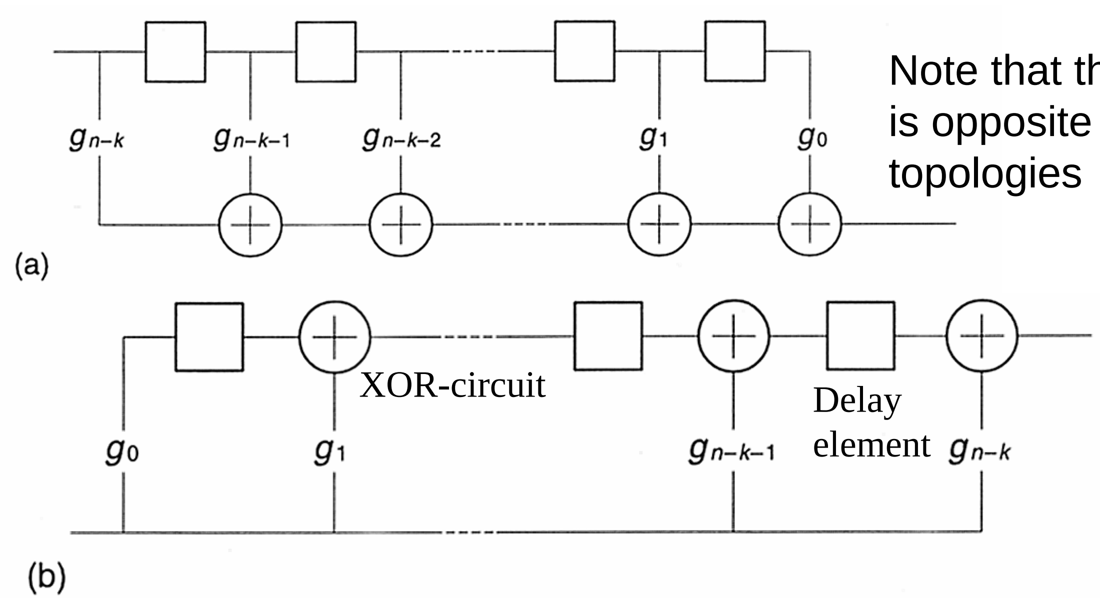
- Figure shows a shift register to realize multiplication by $p^3 + p + 1$



- In practice, multiplication can be realized by two equivalent topologies:

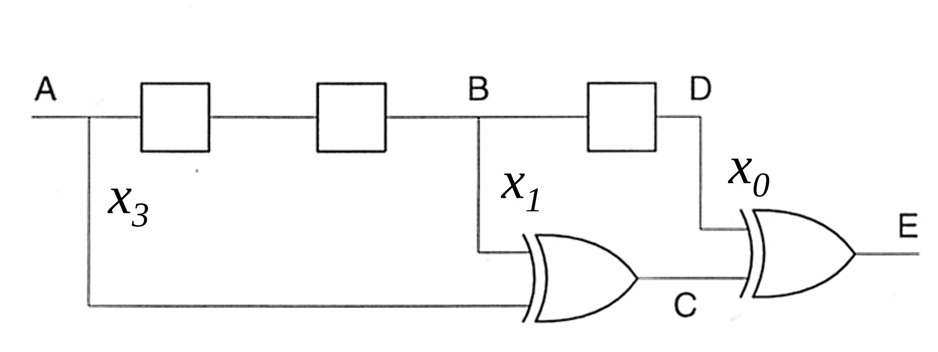
Fibonacci-form

Galois-form



Note that the tap order is opposite in these topologies

Example: Multiplication of data by a shift register



										out
1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	1	1	0	0	0	0	1
0	0	0	0	0	1	1	0	0	0	1
0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	1	1	0

generator polynomial determines connection of taps

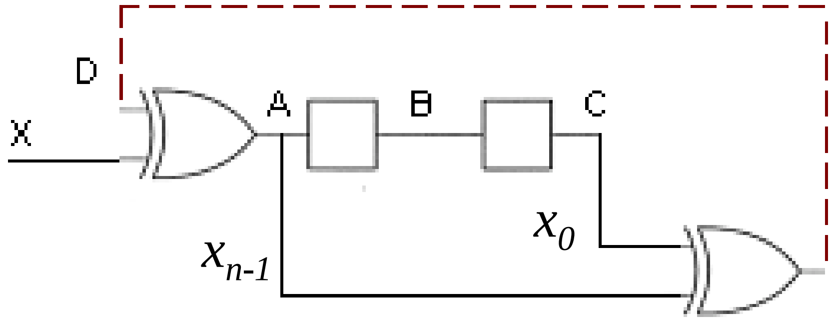
word to be encoded

$$(p+1)(p^3+p+1)$$
$$= p^4 + p^2 + \cancel{p} + p^3 + \cancel{p} + 1$$
$$= p^4 + p^3 + p^2 + 1 \rightarrow 11101$$

Encoded word

$$\mathbf{X}(p) = x_{n-1}p^{n-1} + x_{n-2}p^{n-2} + \cdots + x_1 p + x_0$$

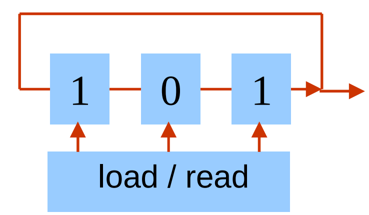
Calculating the remainder (word rotation) by a shift register



			X	A	B	C	D
0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	1
0	0	1	0	1	1	0	1
0	0	0	1	0	1	1	1
0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1

Adding the dashed-line (feedback) enables division by p^n+1

Remainder is left to the shift register



Alternate way to realize rotation

Word to be rotated (divided by the common factor)

$$101 \rightarrow \mathbf{X}(p) = p^2 + 1$$

$$p\mathbf{X}(p) = p^3 + p$$

Determines tap connections

$$\frac{p\mathbf{X}(p)}{p^3+1} = 1 + \frac{p+1}{p^3+1} \rightarrow 011$$

Remainder

Maple script:

```
> rem((x^3+x),(x^3+1),x) mod 2;
                                        1+x
>
```

Examples of cyclic code generator polynomials

- The generator polynomial for an *(n,k)* cyclic code is defined by

$$\mathbf{G}(p) = p^q + p^{q-1}g_{q-1}\cdots + pg_1 + 1, q = n - k$$

and $\mathbf{G}(p)$ is a factor of p^n+1, as noted earlier. Any factor of p^n+1 that has the degree q (the number of check bits) may serve as the generator polynomial. We noticed earlier that a cyclic code is generated by the multiplication

$$\mathbf{X}(p) = \mathbf{M}(p)\mathbf{G}(p)$$

where $\mathbf{M}(p)$ is the k-bit message to be encoded

- Only few of the possible generating polynomials yield high quality codes (in terms of their minimum Hamming distance)

$$\mathbf{G}(p) = p^3 + 0 + p + 1$$

Some cyclic codes:

Type	n	k	R_c	d_{min}	$G(p)$
Hamming codes	7	4	0.57	3	1 011
	15	11	0.73	3	10 011
	31	26	0.84	3	100 101
BCH codes	15	7	0.46	5	111 010 001
	31	21	0.68	5	11 101 101 001
	63	45	0.71	7	1 111 000 001 011 001 111

Systematic cyclic codes

- Define the length $q=n-k$ check vector **C** and the length-k message vector **M** by

$$\mathbf{M}(p) = m_{k-1}p^{k-1} + \cdots + m_1 p + m_0$$

$$\mathbf{C}(p) = c_{q-1}p^{q-1} + \cdots + c_1 p + c_0$$

- Thus the systematic n:th degree codeword polynomial is

$$\begin{aligned}\mathbf{X}(p) &= p^{n-k}(m_{k-1}p^{k-1} + \cdots + m_1 p + m_0) \\ &\quad + c_{q-1}p^{q-1} + \cdots + c_1 p + c_0 \\ &= p^q\mathbf{M}(p) + \mathbf{C}(p)\end{aligned}$$

message bits

check bits

How to determine the check-bits??

Question: Why these denote the message bits still the message bits are **M**(p) ???

k-bits → (n,k) cyclic encoder → n-bits

Determining check-bits

$$\mathbf{X}(p)=\mathbf{M}(p)\mathbf{G}(p)=p^{q}\mathbf{M}(p)+\mathbf{C}(p)$$

$$\frac{7}{5}=1+\frac{7-5\cdot 1}{5}$$

$$\Rightarrow \frac{p^{n-k}\mathbf{M}(p)}{\mathbf{G}(p)}=\mathbf{M}(p)+\frac{\mathbf{C}(p)}{\mathbf{G}(p)}$$

Definition of systematic cyclic code

- Note that the check-vector polynomial $\mathbf{C}(p)$ is the remainder left over after dividing $p^{n-k}\mathbf{M}(p)/\mathbf{G}(p)$ $\boxed{\Rightarrow \mathbf{C}(p)=\mathrm{mod}\left[p^{n-k}\mathbf{M}(p)/\mathbf{G}(p)\right]}$

Example: (7,4) Cyclic code:

$$\mathbf{G}(p)=p^{3}+p^{2}+1$$

$$\mathbf{M}(p)=p^{3}+p$$

$$p^{7-4}\mathbf{M}(p)=p^{6}-p^{4}$$

$$p^{n-k}\mathbf{M}(p)/\mathbf{G}(p)=\underbrace{p^{3}+p^{2}+1}_{\mathbf{Q}(p)}+\underbrace{1}_{\mathbf{C}(p)}$$

$$p^{n-k}\mathbf{M}(p)+\mathbf{C}(p)=p^{3}(p^{3}+p)+1=p^{6}+p^{4}+1$$

1010 -> 1010<u>001</u>

Timo O. Korhonen, HUT Communication Laboratory

Division of the generated code by the generator polynomial leaves no reminder

$$\mathbf{G}(p) = p^3 + p^2 + 1$$

$$\mathbf{M}(p) = p^3 + p$$

$$p^{7-4}\mathbf{M}(p) = p^6 - p^4$$

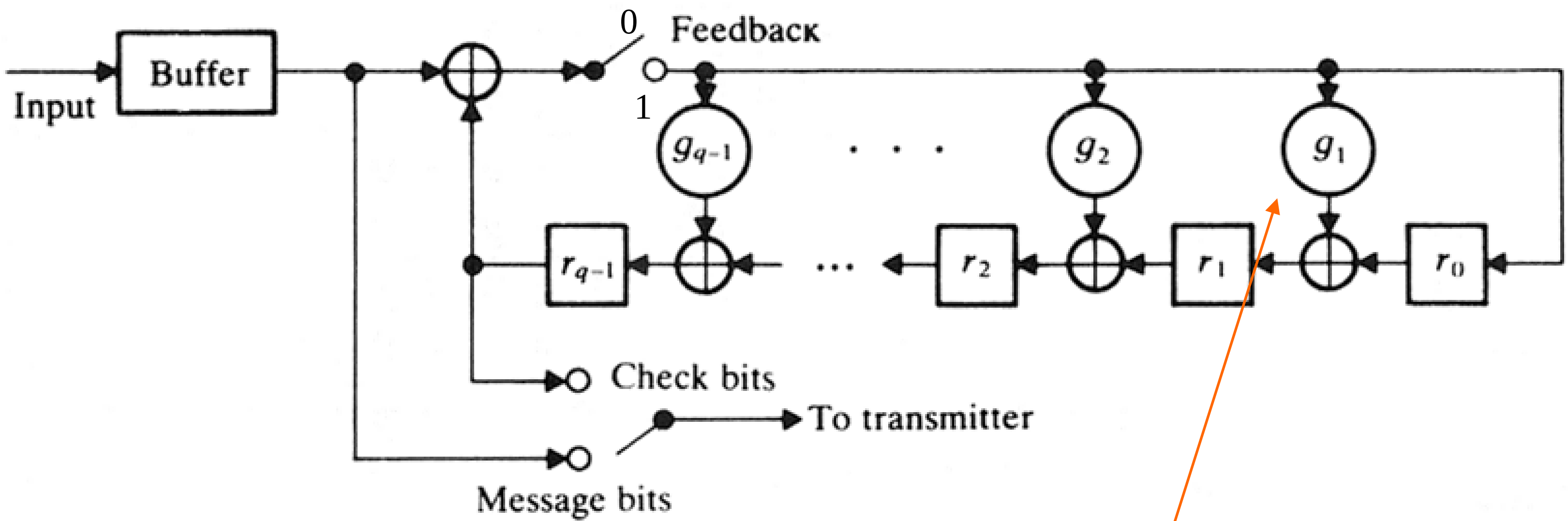
$$p^{n-k}\mathbf{M}(p)/\mathbf{G}(p) = \underbrace{p^3 + p^2 + 1}_{\mathbf{Q}(p)} + \underbrace{1}_{\mathbf{C}(p)}$$

$$p^{n-k}\mathbf{M}(p) + \mathbf{C}(p) = p^3(p^3 + p) + 1 = p^6 + p^4 + 1$$

$$\begin{array}{r|l}
 & \underline{p^3 + p^2 + 1} \\
p^3 + p^2 + 1 & p^6 + p^4 + 1 \\
 & \underline{p^6 + p^5 + p^3} \\
 & p^5 + p^4 + p^3 + 1 \\
 & \underline{p^5 + p^4 + p^2} \\
 & p^3 + p^2 + 1 \\
 & \underline{\underline{p^3 + p^2 + 1}}
\end{array}$$

This can be used for error detection/correction as we inspect later

Circuit for encoding systematic cyclic codes



- We noticed earlier that cyclic codes can be generated by using shift registers whose feedback coefficients are determined directly by the generating polynomial
- For cyclic codes the generator polynomial is of the form

$$\mathbf{G}(p) = p^q + p^{q-1} g_{q-1} + p^{q-2} g_{q-2} + \cdots + p g_1 + 1$$

- In the circuit, first the message flows to the shift register, and feedback switch is set to '1', where after check-bit-switch is turned on, and the feedback switch to '0', enabling the check bits to be outputted

Decoding cyclic codes

- Every valid, received code word **R***(p)* must be a multiple of **G***(p)*, otherwise an error has occurred. (Assume that the probability of noise to convert code words to other code words is very small.)
- Therefore dividing the **R***(p)*/**G***(p)* and considering the remainder as a syndrome can reveal if an error has happed and sometimes also to reveal in which bit (depending on code strength)
- Division is accomplished by a shift registers
- The error syndrome of q=*n*-*k* bits is therefore

$$\mathbf{S}(p) = \mathrm{mod}\left[\mathbf{R}(p)/\mathbf{G}(p)\right]$$

- This can be expressed also in terms of the error **E***(p)* and the code word **X***(p)* while noting that the received word is in terms of error

$$\mathbf{R}(p) = \mathbf{X}(p) + \mathbf{E}(p)$$

hence

$$\mathbf{S}(p) = \mathrm{mod}\left\{\left[\mathbf{X}(p) + \mathbf{E}(p)\right]/\mathbf{G}(p)\right\}$$

$$\mathbf{S}(p) = \mathrm{mod}\left[\mathbf{E}(p)/\mathbf{G}(p)\right]$$

Decoding cyclic codes: syndrome table

Construct the decoding table for the single-error correcting (7, 4) code in Table 16.5. Determine the data vectors transmitted for the following received vectors $\boldsymbol{r}$: **(a) 1101101**; **(b) 0101000**; **(c) 0001100**.

The first step is to construct the decoding table. Because $n - k - 1 = 2$, the syndrome polynomial is of the second order, and there are seven possible nonzero syndromes. There are also seven possible correctable single-error patterns because $n = 7$. Using Eq. (16.20), we compute the syndrome for each of the seven correctable error patterns. For example, for $\boldsymbol{e}$ = **1000000**, $e(x) = x^6$. Because $g(x) = x^3 + x^2 + 1$

$$\begin{array}{r} x^3 + x^2 + x \\ x^3 + x^2 + 1 \overline{) \; x^6 \qquad\qquad\qquad} \\ \underline{x^6 + x^5 + x^3} \\ x^5 + x^3 \\ \underline{x^5 + x^4 + x^2} \\ x^4 + x^3 + x^2 \\ \underline{x^4 + x^3 + x} \\ x^2 + x \quad \leftarrow s(x) \end{array}$$

s = **110**

Using denotation of this example:

$$16.20 \quad s(x) = \mathrm{mod}\left[e(x)/g(x)\right]$$

Table 16.5

d	c
1111	1111111
1110	1110010
1101	1101000
1100	1100101
1011	1011100
1010	1010001
1001	1001011
1000	1000110
0111	0111001

e	s
1000000	110
0100000	011
0010000	111
0001000	101
0000100	100
0000010	010
0000001	001

Decoding cyclic codes: error correction

When the received word $\boldsymbol{r}$ is **1101101**,

$$r(x) = x^6 + x^5 + x^3 + x^2 + 1$$

We now compute $s(x) = \operatorname{mod}[r(x)/g(x)]$

$$\begin{array}{r|l} & x^3 \\ \hline \underbrace{x^3 + x^2 + 1}_{g(x)} & x^6 + x^5 + x^3 + x^2 + 1 \\ & x^6 + x^5 + x^3 \\ \hline & \qquad\qquad\qquad x^2 + 1 \end{array}$$

Hence, $\boldsymbol{s} = \mathbf{101}$. From Table 16.6, this gives $\boldsymbol{e} = \mathbf{0001000}$, and

$$\boldsymbol{c} = \boldsymbol{r} \oplus \boldsymbol{e} = \mathbf{1101101} \oplus \mathbf{0001000} = \mathbf{1100101}$$

Hence, from Table 16.5 we have

$$\boldsymbol{d} = \mathbf{1100}$$

In a similar way, we determine for $\boldsymbol{r} = \mathbf{0101000}$, $\boldsymbol{s} = \mathbf{110}$ and $\boldsymbol{e} = \mathbf{1000000}$; hence $\boldsymbol{c} = \boldsymbol{r} \oplus \boldsymbol{e} = \mathbf{1101000}$, and $\boldsymbol{d} = \mathbf{1101}$. For $\boldsymbol{r} = \mathbf{0001100}$, $\boldsymbol{s} = \mathbf{001}$ and $\boldsymbol{e} = \mathbf{0000001}$; hence $\boldsymbol{c} = \boldsymbol{r} \oplus \boldsymbol{e} = \mathbf{0001101}$, and $\boldsymbol{d} = \mathbf{0001}$.

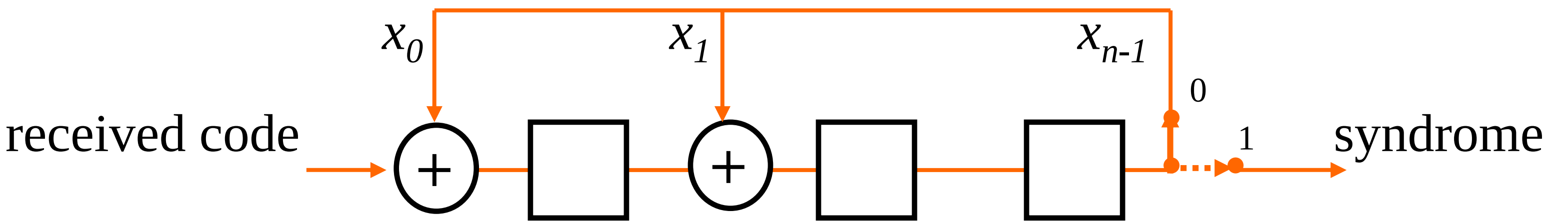
Table 16.6

e	s
1000000	110
0100000	011
0010000	111
0001000	101
0000100	100
0000010	010
0000001	001

Table 16.5

d	c
1111	1111111
1110	1110010
1101	1101000
1100	1100101
1011	1011100
1010	1010001
1001	1001011
1000	1000110
0111	0111001

Decoding circuit for (7,4) code syndrome computation



$$\mathbf{G}(p) = p^3 + p + 1$$

- To start with, the switch is at “0” position
- Then shift register is stepped until all the received code bits have entered the register
- This results is a 3-bit syndrome (n - k = 3):

$$\mathbf{S}(p) = \mathrm{mod}\left[\mathbf{R}(p)/\mathbf{G}(p)\right]$$

  that is then left to the register
- Then the switch is turned to the position “1” that drives the syndrome out of the register
- Note the tap order for Galois-form shift register

Lessons learned

- You can construct cyclic codes starting from a given factored p^n+1 polynomial by doing simple calculations in GF(2)
- You can estimate strength of designed codes
- You understand how to apply shift registers with cyclic codes
- You can design encoder circuits for your cyclic codes
- You understand how syndrome decoding works with cyclic codes and you can construct the respect decoder circuit