6.6 Error Detection Code

6.7 Automatic Repeat Request (ARQ)

6.8 Hybrid automatic repeat-request

6.6 Error-Detection Codes

- Cyclic Redundancy Check (CRC) Codes CRC codes were invented by W.W.Peterson and D.T. Brown in 1961.
- They are error-detecting codes typically used in automatic repeat request (ARQ) systems.
- CRC codes have no error-correction capability but they can be used in combination with an error-correcting code to improve the performance of the system. CRC codes are often (but not always) constructed of polynomials of the form

$$G(x) = (x+1)\,p(x)$$

where $p(x)$ is a primitive polynomial of degree $(r-1)$. .

The $(x+1)$ factor ensures that all odd weight error patterns are detectable.

$n = 2^{r-1} - 1$

Example

- CRC in spread spectrum cellular system (CDMA)

  $g(x) = x^{30} + x^{29} + x^{21} + x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^2 + x + 1$

- CRC-32

  $g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

- CRC - CCITT

  $g(x) = x^{16} + x^{12} + x^5 + 1$

- CRC-16 (ANSI)

  $g(x) = x^{16} + x^{15} + x^2 + 1$

- CRC-5-EPC (Generation-2 RFID)

  $g(x) = x^5 + x^3 + 1$

- CRC-5-USB (USB token packet)

  $g(x) = x^5 + x^2 + 1$

- CRC are simply cyclic codes, so the same encoding and decoding concepts as any other cyclic codes applies.
- The entire encoding operation can be written as

$$c(x) = x^{L} m(x) + \text{Rem}\left[x^{L} m(x) / g(x)\right]$$

The remainder is denoted as

$$p(x) = p_0 x^{L-1} + p_1 x^{L-2} + \ldots + p_{L-2} x + p_{L-1}$$

Assume we use a degree-L CRC generator polynomial g(x) to generate the code.

$$g(x) = x^{L} + g_1 x^{L-1} + g_2 x^{L-2} + \ldots + g_{L-1} x + g_L$$

In general , for message

$$m(x) = m_0 x^{M-1} + m_1 x^{M-2} + \ldots + m_{M-2} x + m_{M-1}$$

the CRC encoding is performed in a systematic form.

That is , $c(x) = x^{L} m(x) + p(x)$

- **Example**

  $g(x) = x^{16} + x^{12} + x^5 + 1$

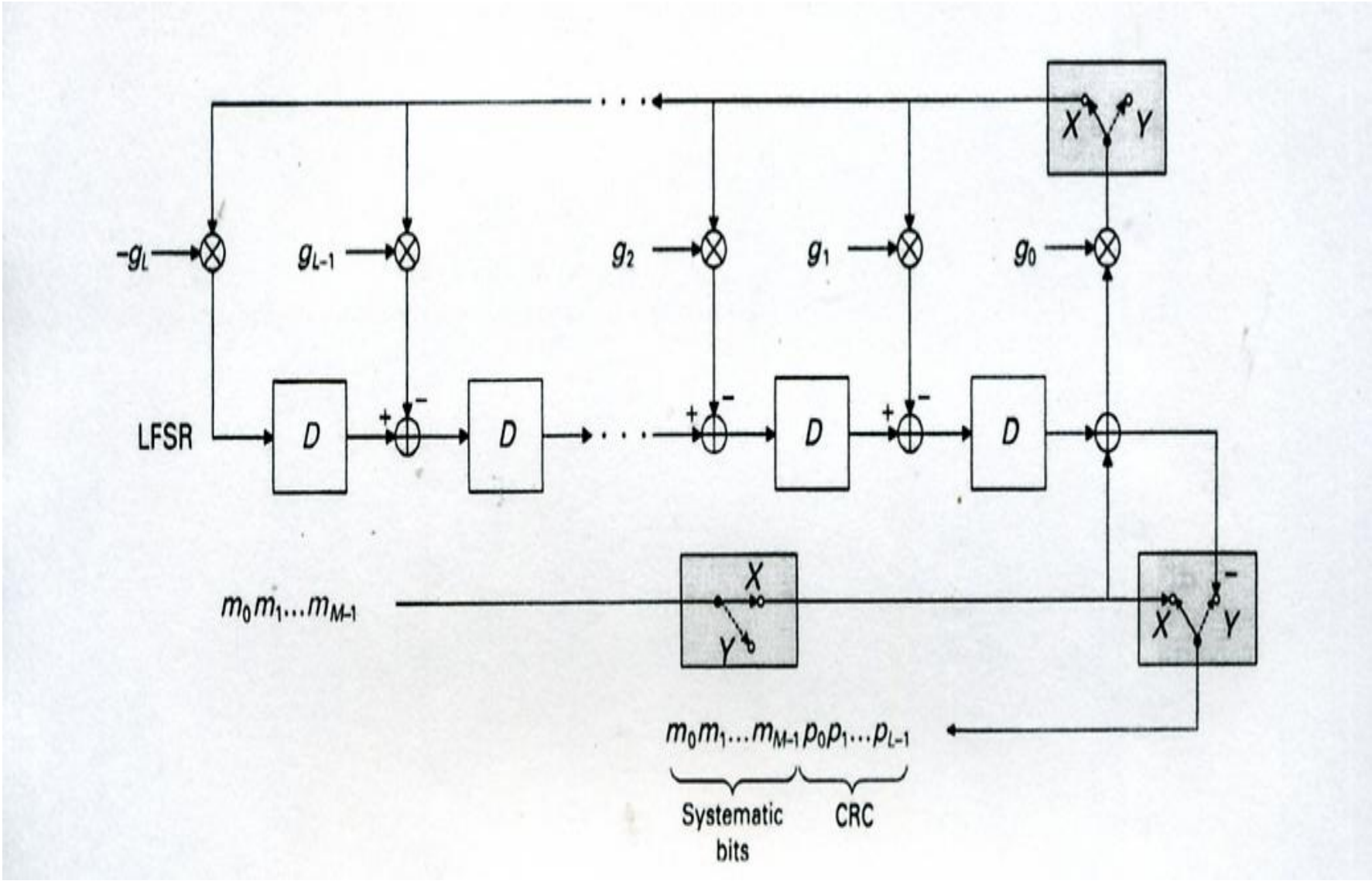
  $m(x) = x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^5 + x^2 + x + 1$

  Then the remainder of $x^{16}\, m(x)$ divided by $g(x)$ is

  $d(x) = x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^4 + x^2$

  The code polynomial is

  $c(x) = x^{30} + x^{29} + x^{27} + x^{26} + x^{24} + x^{21} + x^{18} + x^{16} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^4 + x^2$


X
Y
$-g_L$
g_{L-1}
g_2
g_1
g_0
LFSR
D
D
D
D
+
−
$m_0 m_1 \ldots m_{M-1}$
X
Y
X
Y
$m_0 m_1 \ldots m_{M-1} p_0 p_1 \ldots p_{L-1}$
Systematic bits
CRC

- Figure xx shows a systematic encoder for a general CRC code.

  The linear feedback shift register (LFSR) can be used as a circuit for polynomial division .

  Assume that g(x) is a L-degree polynomial , the LFSR has L shift registers The switches are initially placed at position X. The message bits $m_0 m_1 ... m_{M-2}$ and m_{M-1} are fed into the shift register one at a time in order of increasing index . After the last bit m_{M-1} has been fed into the LFSR , the switches are moved to position y. The LFSR is shifted by another L times to produce the CRC at the output of the rightmost register.

- At the receiver the decoder simply computes the CRC of the message part and adds the results to the CRC bytes, and then tests to see whether the result equals zero.

6.7 Automatic Repeat Request (ARQ)

- When the error control consists of error detection only , the transmission system generally needs to provide a means of alerting the transmitter that an error has been detected and that a retransmission is necessary. Such error control procedures are known as automatic repeat request (ARQ) methods.
- ARQ strategies are particularly suited to transmission systems where transmission delay is small and information occurs naturally in blocks.
- Figure XX illustrates three of the most popular procedures : Stop-and-wait ARQ , Continuous ARQ with pullback , and Continuous ARQ with selective repeat. The three strategies differ with respect to their average rate of information transfer for a fixed binary transmission rate and propagation delay.

- **Stop-and –wait ARQ**

  This strategy requires a half-duplex connection only. The transmitter waits for an acknowledgement (ACK) of each transmission of message before it proceeds with the next transmission of message. In the figure, the third message block is received in error , therefore , the receiver responds with a negative acknowledgement (NAK) and the transmitter retransmits this third message block before transmitting the next in the sequence.

- **Continuous ARQ with pullback (Go-Back-N ARQ)**

  In this strategy , a full-duplex connection is necessary. Both terminals are transmitting simultaneously ;the transmitter is sending message data and the receiver is sending acknowledgement data. Each block of data has to be assigned a sequence number for reference.

  Also, the ACKs and NACKs need to reference such numbers ,

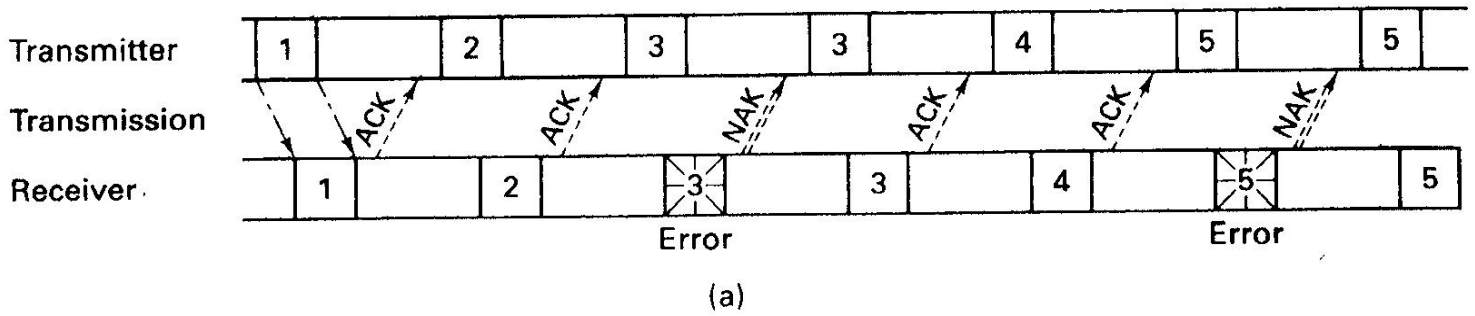
There is a fixed separation between the message being transmitted and the acknowledgement being simultaneously received. When transmission error occurs and are detected by the receiver , the transmitter backs up to the incorrectly received message and retransmits that message and all subsequent messages.

For example, in Fig.xx, when message 8 is being sent , a NACK corresponding to the corrupted message 4 is being received.

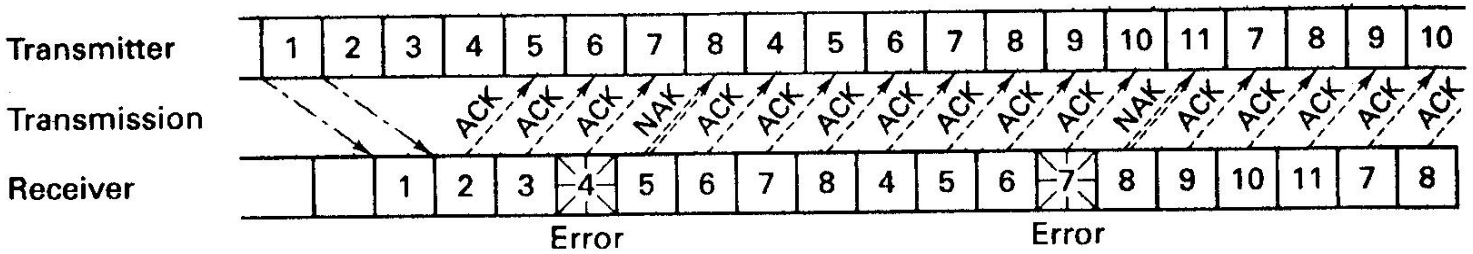
- Selective Repeat ARQ

  In this strategy, as with the Go-BACK-N ARQ , a full- duplex connection is needed. However , only the corrupted message is repeated, and then the transmitter continues the transmission sequence

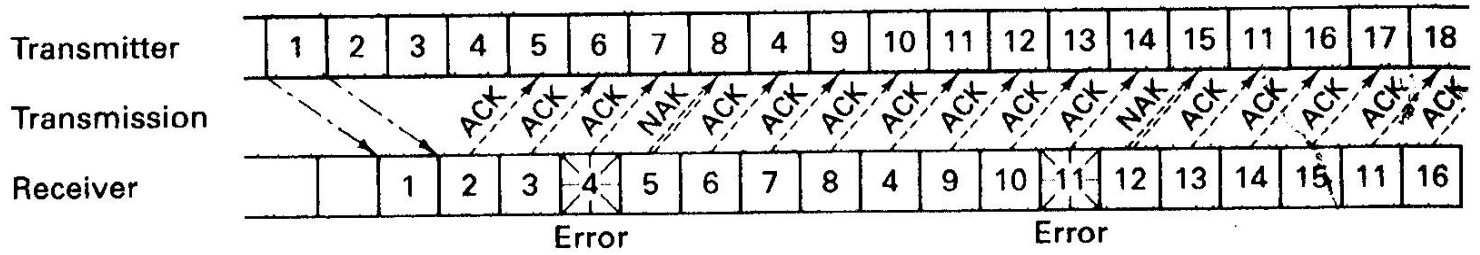
Fig. (a) Stop-and-wait ARQ (b) Continuous ARQ with pullback
(c) Selective repeat ARQ



(a)



(b)



- **The major advantage of ARQ over forward error correction (FEC) is that error detection requires much simpler decoding process and much less redundancy than does error correction. Also, ARQ is adaptive in the sense that message is retransmitted only when error occurs.**
- **FEC may be desirable if a reverse channel is not available or the delay with ARQ is excessive**

6.8 Hybrid automatic repeat-request

- Hybrid ARQ (HARQ) is a variation of the ARQ error control method.
- In standard ARQ, error-detection information (ED) bits are added to data to be transmitted (such as cyclic redundancy check, CRC).
- In Hybrid ARQ, forward error- correction (FEC) bits (such as Reed-Solomon code or turbo code). are also added to the existing Error Detection (ED) bits. As a result Hybrid ARQ performs better than ordinary ARQ in poor signal conditions.

  There is typically a signal quality cross-over point below which simple Hybrid ARQ is better, and above which basic ARQ is better.
- In Hybrid ARQ with soft combining , the erroneously received packet is stored in a buffer memory and later combined with the retramission to obtain a single , combined packed which is more reliable than its constituents . Decoding of error-correcting code operates on the combined signal .

- Hybrid ARQ with soft combining is usually categorized into two types : chase combining (Type I HARQ) and incremental redundancy (Type II HARQ) .
- In Type I HARQ, a both ED and FEC are applied to each message prior to transmission. When the coded data block is received, the receiver first decodes the error-correction code. If the channel quality is good enough, all transmission errors should be correctable, and the receiver can obtain the correct data block. If the channel quality is bad , and not all transmission errors can be corrected, the receiver will detect this situation using the error-detection code, then the received coded data block is discarded and a retransmission is requested by the receiver, similar to ARQ.

  After each retransmission , the receiver uses maximum-ratio combining (MRC) to combine the current and all previous HARQ transmissions of the data-block in order to decode it.

- In **Type II HARQ**, each retransmission does not have to be identical to the original transmission. Instead , multiple sets of coded bits are generated , each of them representing the same set of information bits . When ever a retransmission is required , the retransmission typically uses a different set of coded bits than the previous transmission . The receiver combines the retransmission with previous transmission attempts of the same packet. As the retransmission may contain additional parity bits , not included in the previous transmission attempts , the resulting code rate is generally lowered by a retransmission. The modulation scheme can be different for different retransmission. Typically , incremental redundancy is based on a low-rate code and the different redundancy versions are generated by puncturing of the low-rate mother code. Rate-compatible convolutional codes can be used for this purpose.

- **In summary, in chase combining ,very retransmission contains the same information (data and parity bits). One could think of every retransmission adding extra "energy" to the received transmission.**
- **In incremental redundancy, every retransmission contains different information than the previous one. At every retransmission the receiver gains knowledge of extra information.**