Digital Data Communication Techniques

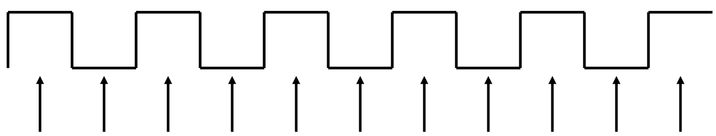
Raj Jain
Washington University
Saint Louis, MO 63131
Jain@cse.wustl.edu

These slides are available on-line at:
http://www.cse.wustl.edu/~jain/cse473-05/

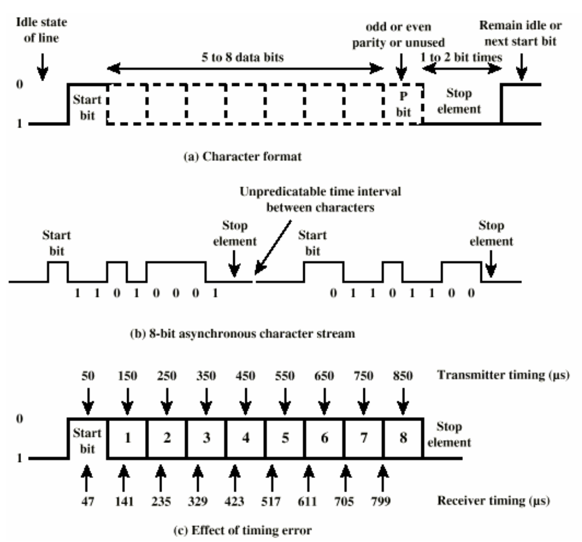
1. Asynchronous vs Synchronous Transmissions
2. Types of Errors
3. Error Detection: Parity, CRC
4. Error Correction

Clock Synchronization

- Suppose, data rate = 1 Mbps
  One bit = 1 μs
- Clock rate is 1% faster,
  Sampling every 0.99 μs
- After 50 bits: 50% away from center ⇒ Error



Asynchronous Transmission



Asynchronous Transmission (Cont)

- Used for short bit sequences
- Idle = No signal, negative voltage, 1
- One Start bit, 7 or 8 data bits
- One parity bit: Odd, Even, None
- Minimum Gap = Stop bits = 1, 1.5, or 2 bits
- Efficiency = data bits/total bits
  8N1 = 1 Start bit + 8 Data bits + 1 Stop bit + 1 parity bit (even though the parity is not being used by this site)
  $\Rightarrow$ 8/(1+8+1+1) = 73%
- Faster clock: 7% $\Rightarrow$ 56% off on 8th bit $\Rightarrow$ Error
- Framing error $\Rightarrow$ False start/end of a frame

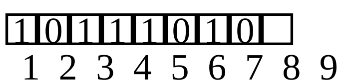
Synchronous Transmission

- Used for longer bit sequences
- Requires clock transmission
  Use codes with clock information (Manchester)
- Beginning of block indicated by a preamble bit pattern called "Syn" or "flag"
- End of block indicated by a post-amble bit pattern
- Character-oriented transmission: Data in 8-bit units
- Bit-oriented transmission: Preamble = Flag
- Efficiency: Data bits/(Preamble+Data+Postamble)
- High-Level Data Link Control (HDLC) uses bit-oriented synchronous transmission.

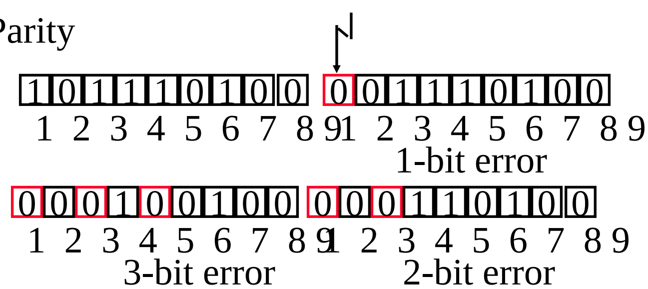
Types of Error

- An error occurs when a bit is altered between transmission and reception
- Single bit errors
  - One bit altered
  - Adjacent bits not affected
  - White noise
- Burst errors
  - Length *B*
  - Contiguous sequence of *B* bits in which first last and any number of intermediate bits in error
  - Impulse noise
  - Fading in wireless
  - Effect greater at higher data rates

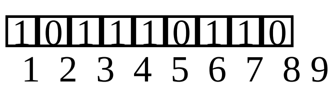
Parity Checks



- Odd Parity



- Even Parity



Check Digit Method

- Make number divisible by 9

Example: 823 is to be sent

1. Left-shift: 8230
2. Divide by 9, find remainder: 4
3. Subtract remainder from 9: 9-4=5
4. Add the result of step 3 to step 1: 8235
5. Check that the result is divisible by 9.

Detects all single-digit errors: 7235, 8335, 8255, 8237

Detects several multiple-digit errors: 8765, 7346

Does not detect some errors: 7335, 8775, ...

Modulo 2 Arithmetic

```
  1111      11001          110
 +1010     ×    11     11 | 1010
 --------  --------       /  11
  0101      11001          --------   010  2
           11001            x11       011  3
           ---------         11       ----  --
           101011          --------   001  1  Mod 2
                             x00      101  5  Binary
                              00
                            -----
                              x0
```

Cyclic Redundancy Check (CRC)

- **Binary Check Digit Method**
- Make number divisible by P=110101 (n+1=6 bits)

Example: M=1010001101 is to be sent

1. Left-shift M by n bits 2^nM= 101000110100000
2. Divide 2^nM by P, find remainder: R=01110
3. ~~Subtract remainder from P~~ ← Not required in Mod 2
4. Add the result of step 2 to step 1 :
   T=101000110101110
5. Check that the result T is divisible by P.

Modulo 2 Division

```
         Q=1101010110
P=110101)101000110100000=2ⁿM
            110101
             111011
             110101
              011101
              000000
               111010
               110101
                011111
                000000
                 111110
                 110101

                  010110
                  000000
                   101100
                   110101
                    110010
                    110101
                     001110
                     000000
                      01110 = R
```

Checking At The Receiver

```
       1101010110
110101)101000110101110
       110101
        111011
        110101
         011101
         000000
          111010
          110101
           011111
           000000
            111110
            110101
```

```
010111
000000
 101111
 110101
  110101
  110101
   00000
```


Polynomial Representation

- Number the bits 0, 1, ..., from right
  $b_n b_{n-1} b_{n-2} \ldots b_3 b_2 b_1 b_0$
  $b_n x^n + b_{n-1} x^{n-1} + b_{n-2} x^{n-2} + \ldots + b_3 x^3 + b_2 x^2 + b_1 x + b_0$
- Example:
  543210
  ↓↓↓↓↓
  110101 = $x^5+x^4+x^2+1$
  1101 1001 0011 = $x^{11}+x^{10}+x^8+x^7+x^4+x+1$
  11 10 9 8 1 0

Cyclic Redundancy Check (CRC)

Polynomial Division Method

Make $T(x)$ divisible by $P(x) = x^5+x^4+x^2+1$ (Note: n=5)

Example: M=1010001101 is to be sent
$M(x) = x^9+x^7+x^3+x^2+1$

1. Multiply M(x) by x^n, $x^nM(x)= x^{14}+x^{12}+x^8+x^7+x^5+$ ....
2. Divide $x^nM(x)$ by $P(x)$, find remainder: $R(x)$=01110=x^3+x^2+x

CRC (Cont)

3. Add the remainder $R(x)$ to $x^nM(x)$:
$T(x)= x^{14}+x^{12}+x^8+x^7+x^5+x^3+x^2+x$
4. Check that the result $T(x)$ is divisible by $P(x)$.

Transmit the bit pattern corresponding to $T(x)$:
101000110101110

Popular CRC Polynomials

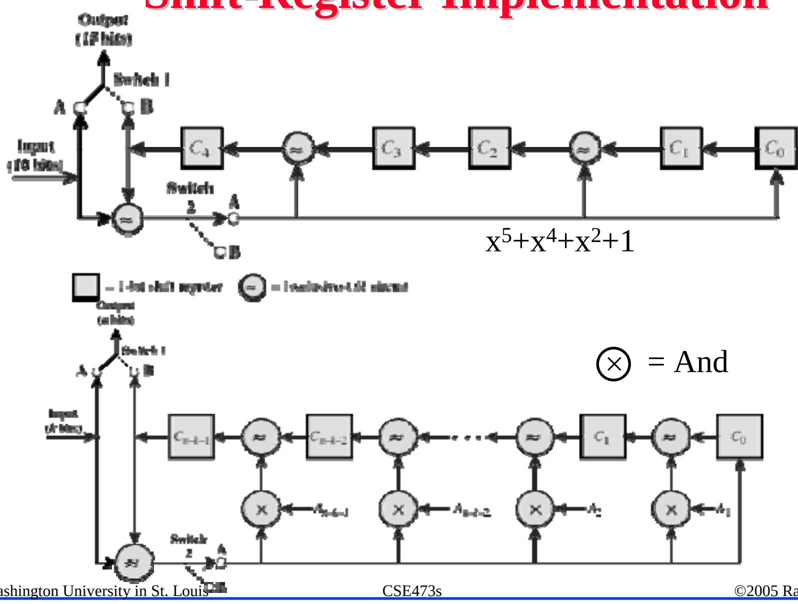
- CRC-12: $x^{12}+x^{11}+x^3+x^2+x+1$
- CRC-16: $x^{16}+x^{15}+x^2+1$
- CRC-CCITT: $x^{16}+x^{12}+x^5+1$
- CRC-32: Ethernet, FDDI, ...
  $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}$
  $+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Even number of terms in the polynomial
  ⇒ Polynomial is divisible by 1+x
  ⇒ Will detect all odd number of bit errors

Errors Detected by CRC

- All single bit errors
- Any burst error of length *n* bits or less, *n*=degree of the polynomial
- Most larger burst errors
  P(undetected burst errors|error has ocurred) = 2^{-n}
- Any odd number of errors if P(x) has 1+*x* as a factor, i.e., has even number of terms
- Any double bit errors as long as *P(x)* has a factor with 3 terms, e.g., $(1+x^4+x^9)(....)$

Shift-Register Implementation



Hamming Distance

- Hamming Distance between two sequences
  = Number of bits in which they disagree
- Example: 011011
  110001

  Difference 101010 ⇒ Distance =3

Error Correction

- Appropriate for wireless applications
  - Bit error rate is high ⇒ Lots of retransmissions
- Appropriate for satellite
  - Propagation delay can be long
    ⇒ Retransmission is inefficient.
- Need to correct errors on basis of bits received

Error Correction Process

- Each *k* bit block mapped to an *n* bit block (*n*>*k*)
- Received code word passed to FEC decoder
  - If no errors, original data block output
  - Some error patterns can be detected and corrected
  - Some error patterns can be detected but not corrected
  - Some (rare) error patterns are not detected
    Results in incorrect data output from FEC

Error Correction Example

- 2-bit words transmitted as 5-bit/word

Data	Codeword
00	00000
01	00111
10	11001
11	11110

Received = 00100 ⇒ Not one of the code words ⇒ Error

Distance (00100,00000) = 1   Distance (00100,00111) = 2

Distance (00100,11001) = 4   Distance (00100,11110) = 3

⇒ Most likely 00000 was sent. Corrected data = 00

b. Received = 01010 Distance(…,00000) = 2 = Distance(…,11110)
Error detected but cannot be corrected

c. Three bit errors will not be detected. Sent 00000, Received 00111.

Summary

- Asynchronous and Synchronous transmission
- Parity, CRC
- CRC Polynomials
- Hamming Distance
- Error Correction

Reading Assignment

- Read sections 6.1 through 6.4 of Stallings' 7th edition

Homework

- Submit solution to Exercise 6.12 (CRC) in Stallings' 7th edition. Use a polynomial representation for all bit sequences.