Investigates how digital data is encoded using signal elements

Competency Level 6.3

Signal Encoding Techniques

- **Encoding** is the process of converting the data or a given sequence of characters, symbols, alphabets etc., into a specified format, for the secured transmission of data.
- **Decoding** is the reverse process of encoding which is to extract the information from the converted format.

Data Encoding

- Encoding is the process of using various patterns of voltage or current levels to represent **1s** and **0s** of the digital signals on the transmission link

Modulation.

- Modulation is the process of encoding source data onto a carrier signal with frequency

Encoding/modulation Techniques

- **Digital data, digital signal**
  - The equipment for encoding digital data into a digital signal is less complex and less expensive than digital-to-analog modulation equipment.
- **Analog data, digital signal**
  - Conversion of analog data (e.g., voice, video) to digital form permits the use of modern digital transmission & switching.
- **Digital data, analog signal**
  - Optical system and unguided media (wireless system) only propagate analog signals.
- **Analog data, analog signal**
  - Baseband: easy and cheap, e.g., in voice-grade telephone lines, voice signals are transmitted over telephone lines at their original spectrum
  - Modulation permits frequency division multiplexing, e.g., AM/FM radios

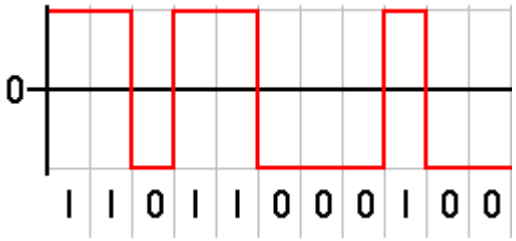
Encoding Schemes

1. **Nonreturn to Zero-Level (NRZ-L)**
2. **Nonreturn to Zero Inverted (NRZI)**
3. **Manchester**
4. Pseudoternary
5. Return to Zero (RZ)
6. Bipolar – AMI

*your learning only 1,2,3 only

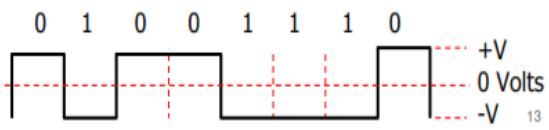
Non-Return to Zero (NRZ) Encoding

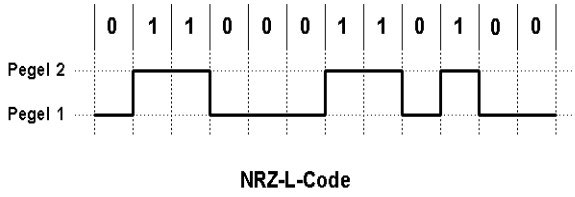
- Non-return to zero is one of the encoding formats used in digital signals.
- It is commonly used in slow speed communications interfaces for both synchronous and asynchronous transmission.
- Using NRZ, a logic 1 bit is sent as a high value and a logic 0 bit is sent as a low value (the line driver chip used to connect the cable may subsequently invert these signals).



NON-RETURN TO ZERO LEVEL (NRZ-L)

- In NRZ-L, binary 1 is represented by positive voltage and 0 by negative voltage. This scheme, though simple, creates problems: if there is a synchronization problem, it is difficult for the receiver to synchronize, and many bits are lost.
- In NRZ-L, 1 is represented by positive voltage and 0 by negative voltage. Synchronization is a problem in this encoding scheme.
- Example:

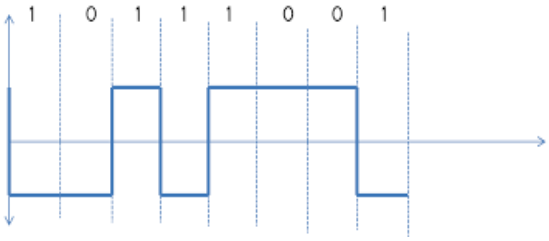


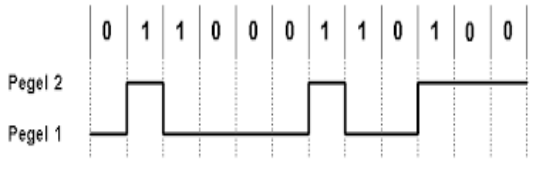


NRZ-L-Code

Non-return to Zero Inverted (NRZI)

- Non-return to zero inverted on ones
- Constant voltage pulse for duration of bit time
- Data encoded as presence or absence of signal transition at beginning of bit time
  —Transition (low-to-high or high-to-low) denotes a binary 1
  —No transition denotes binary 0
- If a data bit is 1, NRZI transitions at the clock boundary. If a data bit is 0, there is no transition.
- NRZI may have long series of 0s or 1s, resulting in clock recovery difficulties.

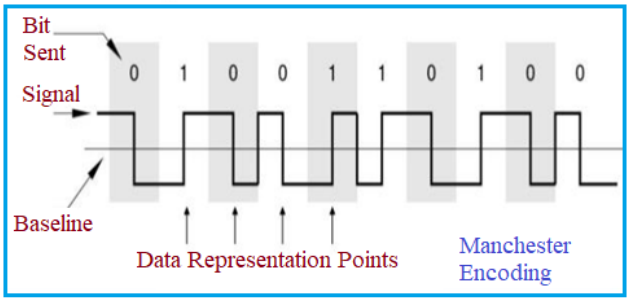


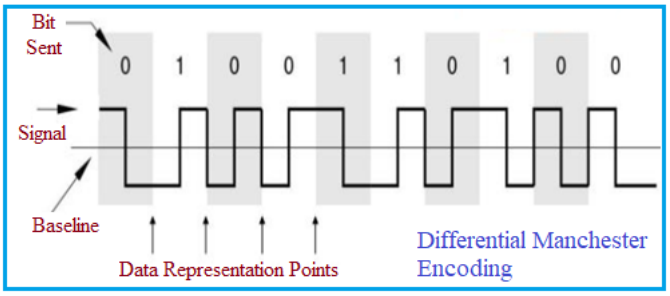


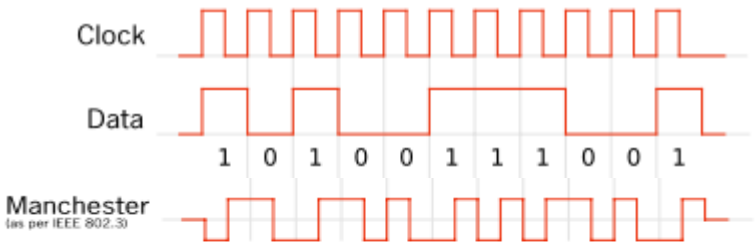
NRZ-I-Code

Manchester code

- The Manchester encoding, a simple and effective way to improve digital communication for high-speed or wireless
- The Manchester encoding is a data modulation technique that can be used in many situations, but it is particularly useful in the transfer of binary data based on analog signals, RF, optical, digital high speed or long-distance digital signals.
- Manchester encoding is named after the University of Manchester, where the first recorded use of it occurred in the late 1940's
- The signal transitions do not always occur at the 'bit boundary' but there is always a transition at the center of each bit.
- The **Manchester code** is a binary encoding which also encodes the clock signal. This is very useful, because an external clock signal is no longer needed.
- Advantages of Manchester code is that the DC component of the signal carries no information.
- A logic 1 is indicated by a 0 to 1 transition at the center of the bit 
- Logic 0 by 1 to 0 transition. 
- Used by IEEE 802.3 specification for Ethernet LAN (Short distances)



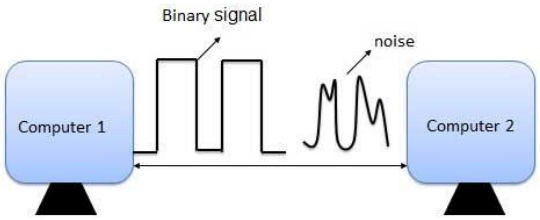
- Used by IEEE 802.5 specification for Token Ring LAN
  - Dedicated mid-bit transition used only for clocking
  - Data representation is mapped as per bit start time instant.
  - No transition at start of a bit period represents as 1
  - Transition at start of a bit period represents as 0
- 



Error Control

What is Error?

- Error is a condition when the output information does not match with the input information.
- Network is responsible for transmission of data from one device to another device. The end to end transfer of data from a transmitting application to a receiving application involves many steps,
- During data transmission, sometimes data bits may get flipped due to various reasons. In such situations the data bit received is in error.
- Transmission errors are caused by:
    - Thermal Noise
    - Impulse Noise
    - Signal Distortion During Transmission (Attenuation)
    - Crosstalk
    - Voice Amplitude Signal Compression



Error detection

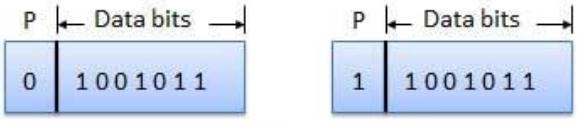
- Error detection is the detection of errors caused by noise or other impairments (the data bit has been altered) during transmission from the transmitter to the receiver.

- The only way to do error detection is to send extra data with each message. These error detection data are added to each message by the data link layer of the sender on the basis of some mathematical calculations performed on the message.
- The receiver performs the same mathematical calculations on the message it receives and matches its results against the error-detection data that were transmitted with the message. If the two match, the message is assumed to be correct. If they don't match, an error has occurred.
- Three error-detection methods are
    1. parity checking,
    2. checksum,
    3. Cyclic redundancy checking.

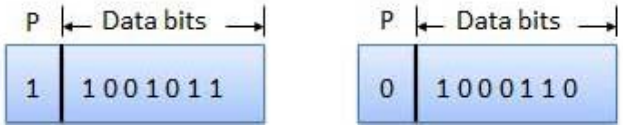
Error correction (recovery mechanisms)

- Are used to correct the data bits received in error and to recover the actual data bits.
- Error correction is the process of detecting errors in transmitted messages and reconstructing the original error-free data.
- Error correction ensures that corrected and error-free messages are obtained at the receiver side

1. **parity bit**
   - A **parity bit**, also known as a **check bit**, is a single bit that can be appended to a binary.
   - To detect and correct the errors, additional bits are added to the data bits at the time of transmission.
     - The additional bits are called **parity bits**. They allow detection or correction of the errors.
     - The data bits along with the parity bits form a **code word**.
   -
   - It is set to either **1** or **0** to make the total number of **1**-bits either even ("even parity") or odd ("odd parity").
   - For even parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is even.

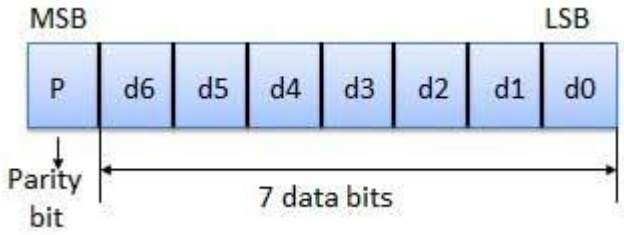


   - For odd parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is odd.



Parity Checking of Error Detection

The MSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.



Even parity -- Even parity means the number of 1's in the given word including the parity bit should be even (2,4,6,....).

Odd parity -- Odd parity means the number of 1's in the given word including the parity bit should be odd (1,3,5,....).

Parity bit examples

sequence of seven bits	with eighth even parity bit:	with eighth odd parity bit:
0100010	01000100	01000101
1000000	10000001	10000000

Example parity checking process

At sender place

The data is 10101

Given the **even parity bit of** 1

Resulting in the bit sequence 101011.

At receiver place

This data is transferred to another computer.

In transit, the data is corrupted,
And the computer receives the incorrect data 100011.

The receiving computer computes the parity: 1+0+0+0+ 1+ 1 = 3.

It then performs **3 modulo 2**

Expecting the result 0 which would indicate that the **number is even**.

Instead, it receives the result 3 modulo 2 = 1, indicating that the **number is odd**.

It is looking for numbers with even parity, *It is **Parity Checking of Error Detection**

Do the process again

- It asks the original computer to send the data again.
- This time, the data comes through with no errors: 101011. The receiving computer calculates 1+0+1+0+1+1 = 4.
- 4 modulo 2 = 0, indicating even parity. The parity bit is stripped from the end of the sequence, and the data 10101 is accepted.

Issues of Parity Checking

- This technique cannot detect an even number of bit errors (two, four, six and so on).
- If two bits have changed on the way over then the receiver wouldn't be able to detect an error

Example

- Consider the data unit to be transmitted is 10010001 and even parity is used.
- Then, code word transmitted to the receiver = 100100011
- Consider during transmission, code word modifies as 101100111. (2 bits flip)
- On receiving the modified code word, receiver finds the number of 1's is even and even parity is used.
- So, receiver assumes that no error occurred in the data during transmission though the data is corrupted.

0ut of syllabus

*not for a/l (BUT LEARN THIS)

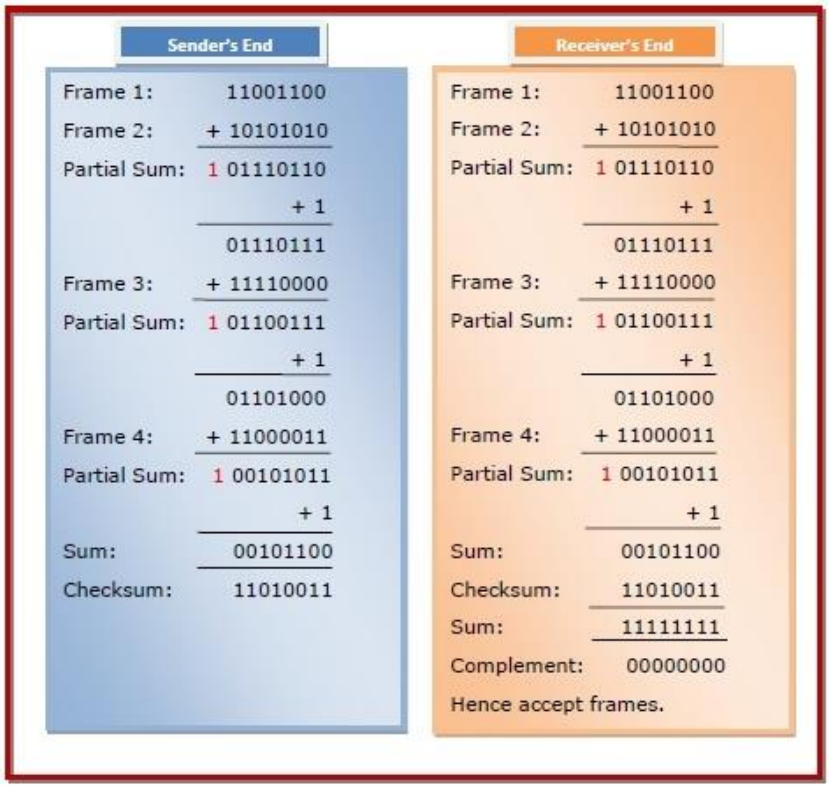
*Checksum

This is a block code method where a checksum is created based on the data values in the data blocks to be transmitted using some algorithm and appended to the data. When the receiver gets this data, a new checksum is calculated and compared with the existing checksum. A non-match indicates an error.

Data is divided into fixed sized frames or segments.

- **Sender's End** − the sender adds the segments using 1's complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
- **Receiver's End** − the receiver adds the incoming segments along with the checksum using 1's complement arithmetic to get the sum and then complements it.

If the result is zero, the received frames are accepted; otherwise they are discarded.



*Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Check (CRC) is a block code invented by W. Wesley Peterson in 1961.
- It is commonly used to detect accidental changes to data transmitted via telecommunications networks and storage devices.
- CRC involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system.
- The divisor is generated using polynomials. CRC is also called polynomial code checksum.

Sender using CRC

IF the sender want to send the data 100100 with deviser 1101

Divisor (L) =1101

Length of Divisor =4

Applied bits to the sending data
=L-1
=4-1
=3
=000 added

```
        111101
1101 ) 100100000
       1101
       01000
        1101
        01010
         1101
         001100
          1101
          0001
```

CRC bits → 001

Transmitted bits = Original Message +CRC bits
= 100100000+001
= 100100001

⊕ represents bitwise XOR

Receiver using CRC

Receiver receive data as 1000100001

And Divisor (L) =1101

```
        111101
1101 ) 100100001
       1101
       01000
        1101
        01010
         1101
         01110
          1101
          001101
           1101
           0000
```

Remainder is zero, So data is accepted