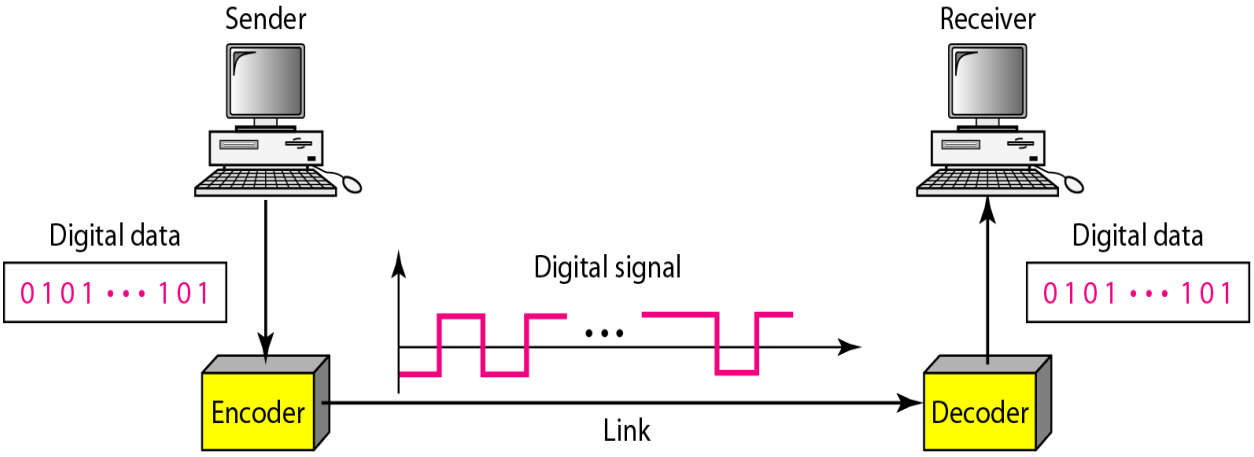
# Digital Transmission

## DIGITAL TO DIGITAL CONVERSION

In this section, we see how we can represent digital data by using digital signals. The conversion involves three techniques: **line coding**, **block coding** and **scrambling**.

### Line Coding

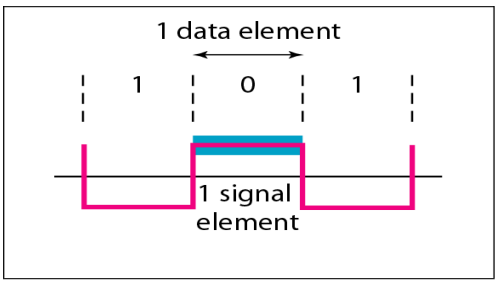
Line coding is the process of converting digital data to digital signals. We assume that data, in the form of text, numbers, graphical images, audio, or video, are stored in computer memory as sequences of bits. Line coding converts a sequence of bits to a digital signal.
At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are recreated by decoding the digital signal. Figure below shows the process.



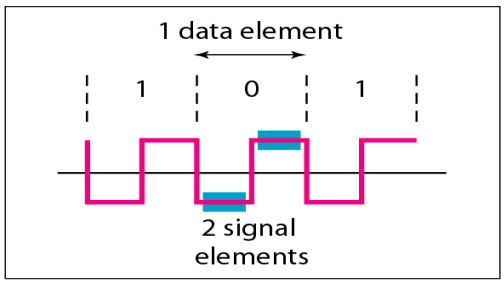

#### *Characteristics*

**i. Signal Element versus Data Element**

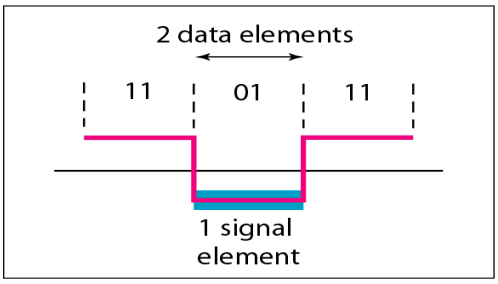
A **data element** is the smallest entity that can represent a piece of information: this is the bit. A **signal element** is the shortest unit (timewise) of a digital signal.
In digital data communications, a signal element carries data elements. Data elements are being carried; signal elements are the carriers.
We define a **ratio *r*** which is the number of data elements carried by each signal element.
Following figure shows several situations with different values of *r.*



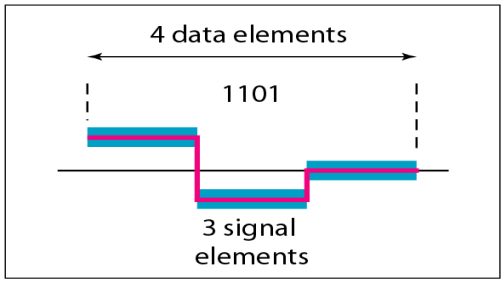

a. One data element per one signal element (r = 1)




b. One data element per two signal elements $\left(r = \frac{1}{2}\right)$




c. Two data elements per one signal element (r = 2)




d. Four data elements per three signal elements $\left(r = \frac{4}{3}\right)$

*(An analogy may help here. Suppose each data element is a person who needs to be carried from one place to another. We can think of a signal element as a vehicle that can carry people. When r = 1, it means each person is driving a vehicle. When r > 1, it means more than one person is travelling in a vehicle (a carpool, for example). We can also have the case where one person is driving a car and a trailer (r = 1/2).)*

### ii. Data Rate Versus Signal Rate

The **data rate** defines the number of data elements (bits) sent in 1s. The unit is bits per second (bps). The **signal rate** is the number of signal elements sent in 1s. The unit is the baud.
The data rate is sometimes called the bit rate; the signal rate is sometimes called the pulse rate, the modulation rate, or the baud rate.
One goal in data communications is to increase the data rate while decreasing the signal rate. Increasing the data rate increases the speed of transmission; decreasing the signal rate decreases the bandwidth requirement. *(In our vehicle-people analogy, we need to carry more people in fewer vehicles to prevent traffic jams. We have a limited bandwidth in our transportation system.)*
We can formulate the relationship between data rate and signal rate as,

$$\mathbf{S = c \times N \times 1/r \ \ baud}$$

Where *N* is the data rate (bps); c is the case factor, which varies for each case; S is the number of signal elements; and *r* is ratio.

**iii. Bandwidth**

A digital signal that carries information is non-periodic. The bandwidth of a non-periodic signal is continuous with an infinite range. However, most digital signals we encounter in real life have a bandwidth with finite values. In other words, the bandwidth is theoretically infinite, but the effective bandwidth is finite.

We can say that the baud rate, not the bit rate, determines the required bandwidth for a digital signal. If we use the transportation analogy, the number of vehicles affects the traffic, not the number of people being carried. More changes in the signal mean injecting more frequencies into the signal.

**iv. Baseline Wandering**

In decoding a digital signal, the receiver calculates a running average of the received signal power. This average is called the *baseline.* The incoming signal power is evaluated against this baseline to determine the value of the data element.

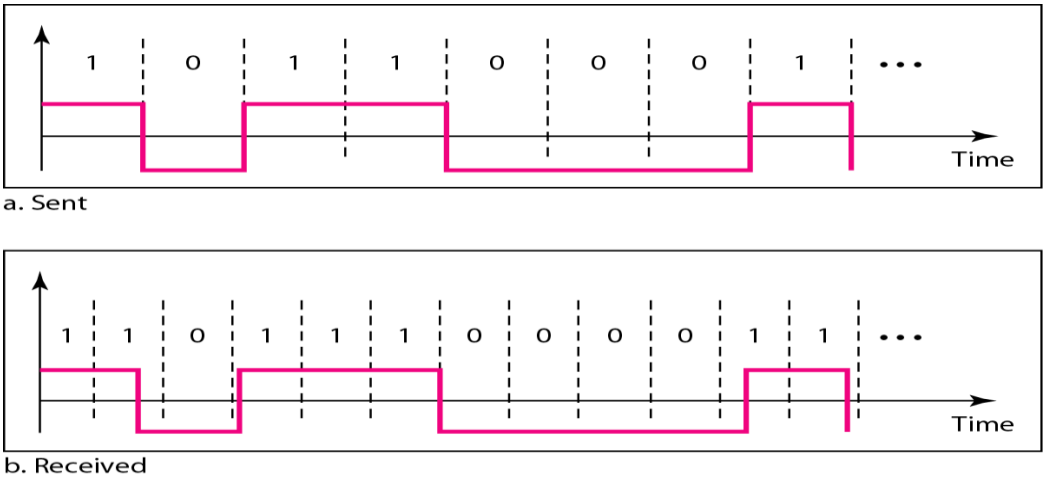
A long string of 0s or 1s can cause a drift in the baseline (baseline wandering) and make it difficult for the receiver to decode correctly. A good line coding scheme needs to prevent baseline wandering.

**v. DC Components**

When the voltage level in a digital signal is constant for a while, the spectrum creates very low frequencies. These frequencies around zero, called DC (direct-current) *components,* present problems for a system that cannot pass low frequencies or a system that uses electrical coupling (via a transformer). For example, a telephone line cannot pass frequencies below 200 Hz.

**vi. Self-synchronization**

To correctly interpret the signals received from the sender, the receiver's bit intervals must correspond exactly to the sender's bit intervals. If the receiver clock is faster or slower, the bit intervals are not matched and the receiver might misinterpret the signals. Figure below shows a situation in which the receiver has a shorter bit duration. The sender sends 10110001, while the receiver receives 110111000011.

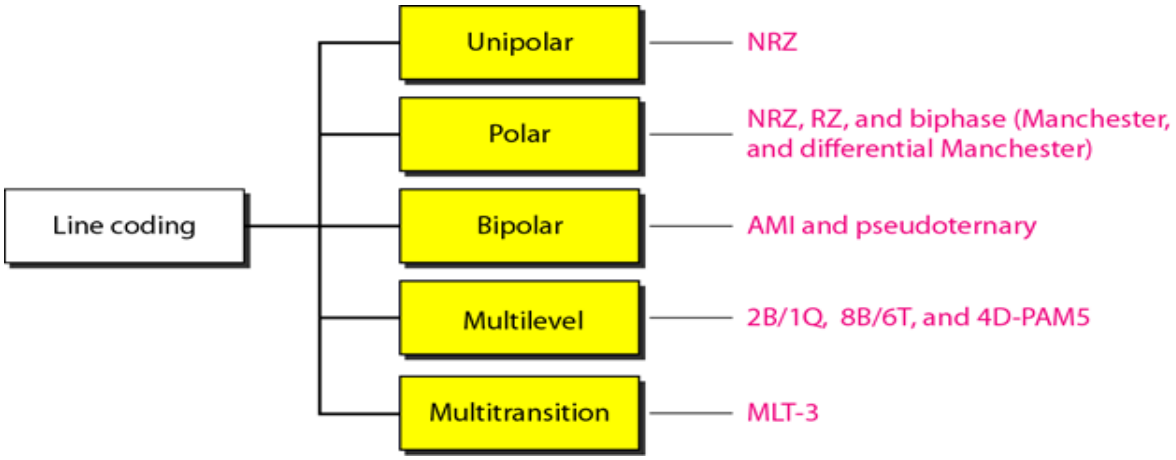


**vii. Built-in Error Detection**
It is desirable to have a built-in error-detecting capability in the generated code to detect some of or all the errors that occurred during transmission.

**viii. Immunity to Noise and Interference**
Another desirable code characteristic is a code that is immune to noise and other interferences.

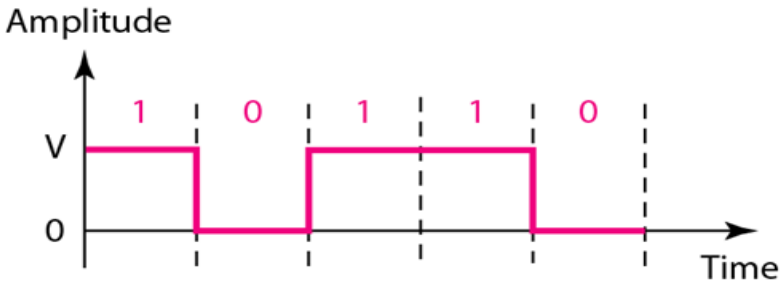
**ix. Complexity**
A complex scheme is more costly to implement than a simple one. For example, a scheme that uses four signal levels is more difficult to interpret than one that uses only two levels.

## Line Coding Schemes




**Unipolar Scheme**
In a unipolar scheme, all the signal levels are on one side of the time axis, either above or below.

**NRZ (Non-Return-to-Zero):** Traditionally, a unipolar scheme was designed as a non-return-to-zero (NRZ) scheme in which the positive voltage defines bit 1 and the zero voltage defines bit 0. It is called NRZ because the signal does not return to zero at the middle of the bit. Following Figure show a unipolar NRZ scheme.

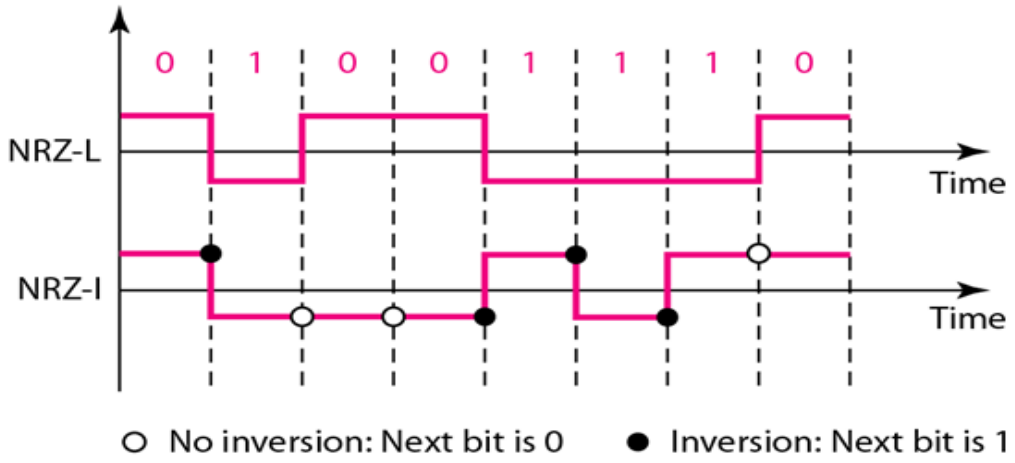



Compared with polar scheme, this scheme is very costly. This scheme is normally not used in data communications today.

**Polar Schemes**

In polar schemes, the voltages are on the both sides of the time axis. For example, the voltage level for 0 can be positive and the voltage level for I can be negative.

**Non-Return-to-Zero (NRZ)**: In polar NRZ encoding, we use two levels of voltage amplitude. There are two versions of polar NRZ: **NRZ-L** and **NRZ-I**, as shown in Figure. In the first variation, NRZ-L (NRZ-Level), the level of the voltage determines the value of the bit. In the second variation, NRZ-I (NRZ-Invert), the change or lack of change in the level of the voltage determines the value of the bit. If there is no change, the bit is 0; if there is a change, the bit is 1.



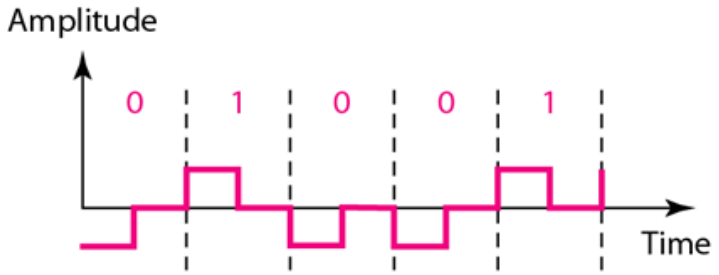

Although **baseline wandering** is a problem for both variations, it is twice as severe in NRZ-L. If there is a long sequence of 0s or 1s in NRZ-L, the average signal power becomes skewed. The receiver might have difficulty discerning the bit value. In NRZ-I this problem occurs only for a long sequence of 0s. If somehow we can eliminate the long sequence of 0s, we can avoid baseline wandering.

The **synchronization** problem (sender and receiver clocks are not synchronized) also exists in both schemes.

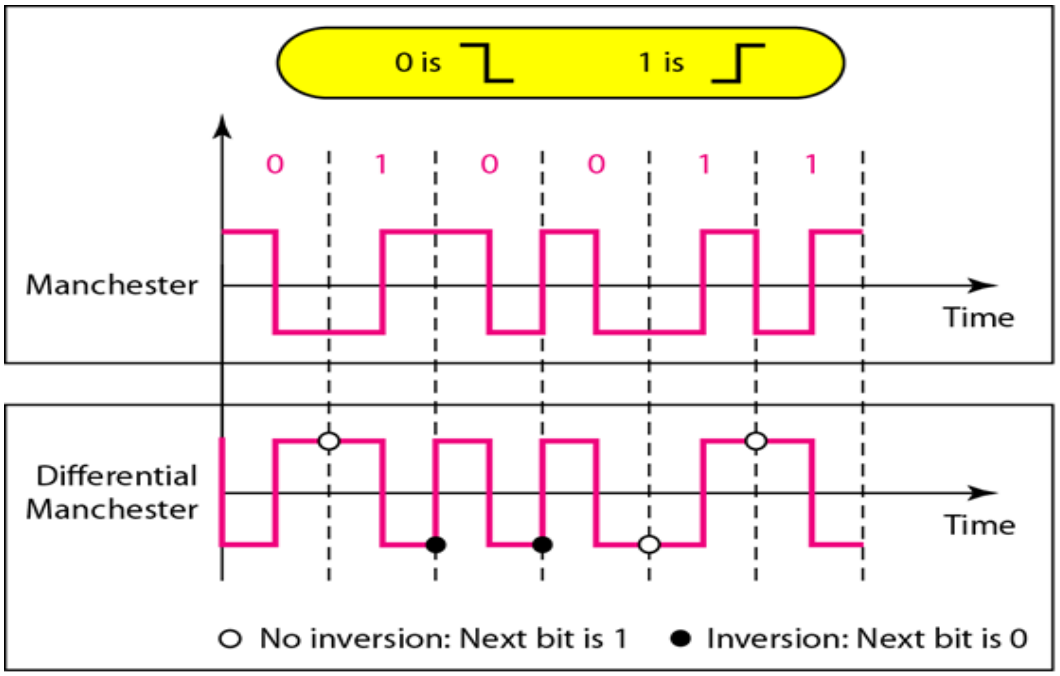
Another problem with NRZ-L occurs when there is a sudden change of polarity in the system. For example, if twisted-pair cable is the medium, a change in the polarity of the wire results in all 0s interpreted as 1s and all 1s interpreted as 0s. NRZ-I does not have this problem.

NRZ-L and NRZ-I both have a **DC component** problem.

**Return to Zero (RZ):** The return-to-zero (RZ) scheme, uses three values: positive, negative, and zero. In RZ, the signal changes not between bits but during the bit. In Figure, we see that the signal goes to 0 in the middle of each bit. It remains there until the beginning of the next bit. RZ encoding provides **self-synchronization**. The main disadvantage of RZ encoding is that it requires two signal changes to encode a bit and therefore occupies greater bandwidth. A sudden change of polarity resulting in all 0s interpreted as 1s and all 1s interpreted as 0s. There is **no DC component problem**. Another problem is the complexity: RZ uses three levels of voltage, which is more complex to create and discern.




**Biphase: Manchester and Differential Manchester:** The idea of RZ (transition at the middle of the bit) and the idea of NRZ-L are combined into the Manchester scheme.
In **Manchester encoding**, the duration of the bit is divided into two halves. The voltage remains at one level during the first half and moves to the other level in the second half.
**Differential Manchester**, combines the ideas of RZ and NRZ-I. There is always a transition at the middle of the bit, but the bit values are determined at the beginning of the bit. If the next bit is 0, there is a transition; if the next bit is 1, there is none. Following figure shows both Manchester and differential Manchester encoding.

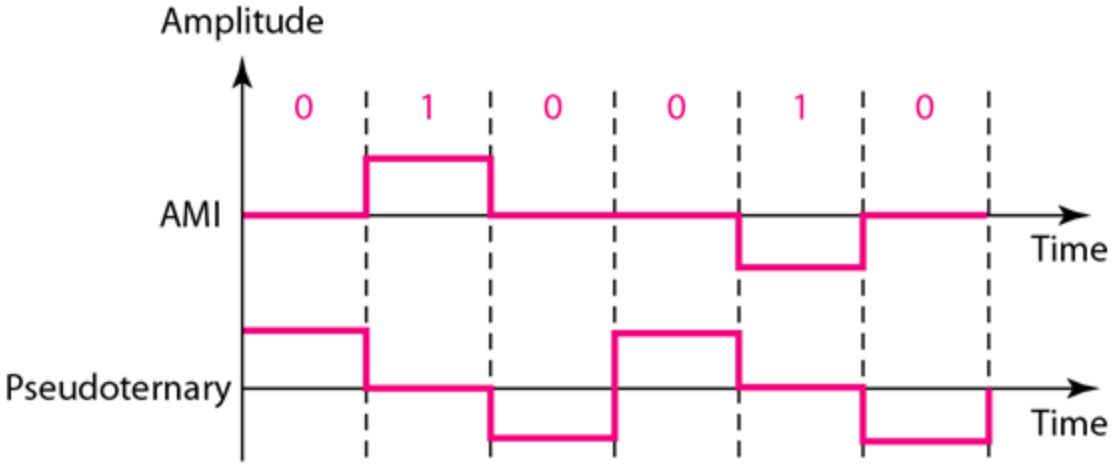



In Manchester and differential Manchester encoding, the transition at the middle of the bit is used for synchronization. There is no baseline wandering, no DC component because each bit has a positive and negative voltage contribution. The only drawback is the signal rate. The signal rate for Manchester and differential Manchester is double that of NRZ. Note that Manchester and differential Manchester schemes are also called **biphase schemes**.

**Bipolar Schemes**

In bipolar encoding (sometimes called *multilevel binary),* there are three voltage levels: positive, negative, and zero. The voltage level for one data element is at zero, while the voltage level for the other element alternates between positive and negative.

**AMI and Pseudoternary:** Figure below shows two variations of bipolar encoding: AMI and pseudoternary. A common bipolar encoding scheme is called bipolar alternate mark inversion (AMI). In the term *alternate mark inversion,* the word *mark* comes from telegraphy and means 1. So AMI means alternate 1 inversion. A neutral zero voltage represents binary 0. Binary 1s are represented by alternating positive and negative voltages. A variation of AMI encoding is called pseudo ternary in which the 1 bit is encoded as a zero voltage and the 0 bit is encoded as alternating positive and negative voltages.



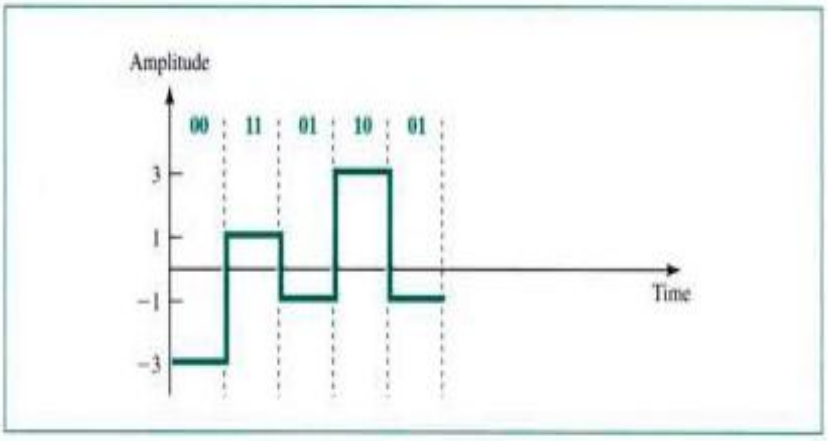

If we have a long sequence of 1s, the voltage level alternates between positive and negative; it is not constant. Therefore, there is **no DC component**. For a long sequence of 0s, the voltage remains constant, but its amplitude is zero, which is the same as having no DC component. In other words, a sequence that creates a constant zero voltage does not have a DC component.
AMI is commonly used for long-distance communication, but it has a synchronization problem when a long sequence of 0s is present in the data.

**Multilevel Schemes**

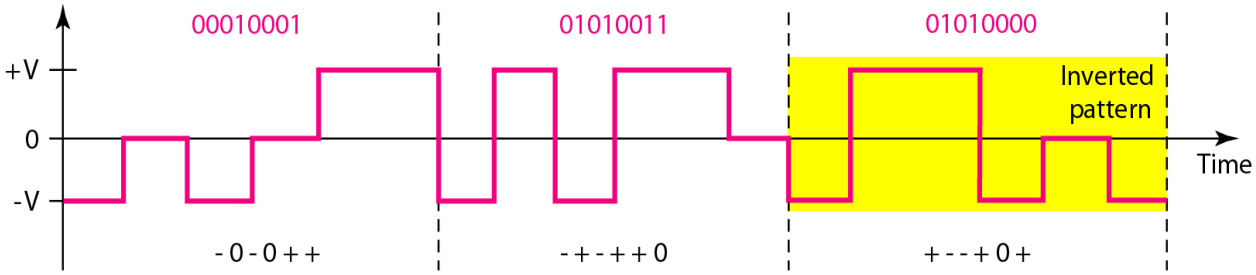
The goal is to increase the number of bits per baud by encoding a pattern of *m* data elements into a pattern of *n* signal elements.
We have two types of data elements (0s and 1s), which means that a group of m data elements can produce a combination of **$2^m$ data patterns**. We can have different types of signal elements by allowing different signal levels. If we have L different levels, then we can produce **$L^n$ signal patterns**. If $2^m = L^n$, then each data pattern is encoded into one signal pattern. If $2^m < L^n$, data patterns occupy only a subset of signal patterns. The subset can be carefully designed to prevent baseline wandering, to provide synchronization, and to detect errors that occurred during data transmission. Data encoding is not possible if $2^m > L^n$ because some of the data patterns cannot be encoded.

The code designers have classified these types of coding as ***mBnL,*** where *m* is the length of the binary pattern, *B* means binary data, *n* is the length of the signal pattern, and *L* is the number of levels in the signaling. A letter is often used in place of L: *B* (binary) for *L*=2, *T* (ternary) for *L*=3, and Q (quaternary) for *L*=4. Note that the first two letters define the data pattern, and the second two define the signal pattern.

**2B1Q:** Two binary, one quaternary (2BIQ), uses data patterns of size 2 and encodes the 2-bit patterns as one signal element belonging to a four-level signal. In this type of encoding *m* =2, *n* =1, and *L* =4 (quaternary). The average signal rate of 2B1Q is S =*N/4.* This means that using 2B1Q, we can send data 2 times faster than by using NRZ-L. There are no redundant signal patterns in this scheme because $2^2 = 4^1$.



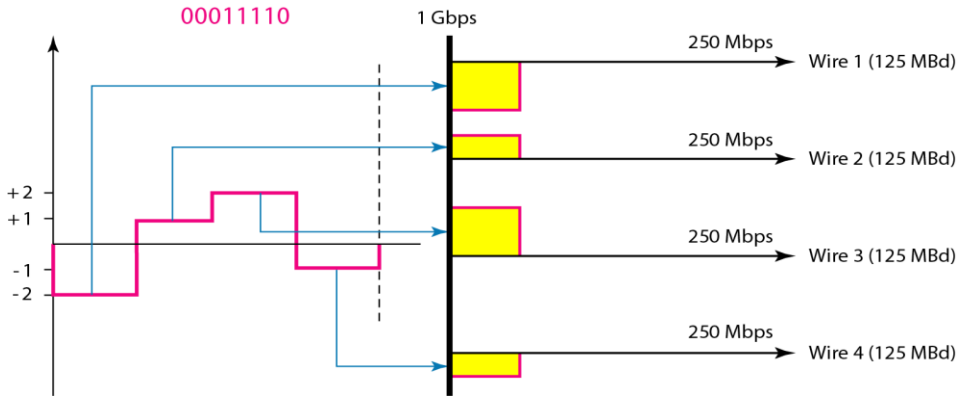

**8B6T:** encodes a pattern of 8 bits as a pattern of 6 signal elements, where the signal has three levels (ternary). In this type of scheme, we can have $2^8$ =256 different data patterns and $3^6$ =729 different signal patterns. There are 729 - 256 =473 redundant signal elements that provide synchronization and error detection. Part of the redundancy is also used to provide DC balance. Each signal pattern has a weight of 0 or +1 DC values. This means that there is no pattern with the weight -1. To make the whole stream DC-balanced, the sender keeps track of the weight. If two groups of weight 1 are encountered one after another, the first one is sent as is, while the next one is totally inverted to give a weight of -1.




The three possible signal levels are represented as -, 0, and +. The first 8-bit pattern 00010001 is encoded as the signal pattern -0-0++ with weight 0; the second 8-bit pattern 01010011 is

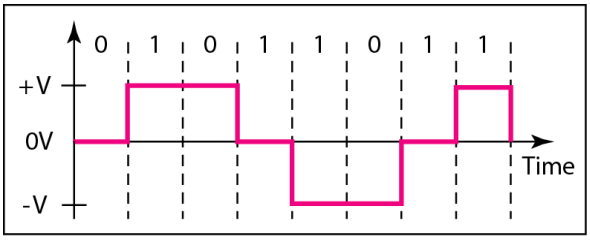
encoded as - + - + + 0 with weight +1. The third bit pattern should be encoded as + - - + 0 + with weight +1. To create DC balance, the sender inverts the actual signal. The receiver can easily recognize that this is an inverted pattern because the weight is -1. The pattern is inverted before decoding.

**4D-PAM5:** Four dimensional five-level pulse amplitude modulation (4D-PAM5). The 4D means that data is sent over four wires at the same time. It uses five voltage levels, such as -2, -1, 0, 1, and 2. However, one level, level 0, is used only for forward error detection. If we assume that the code is just one-dimensional, the four levels create something similar to 8B4Q. In other words, an 8-bit word is translated to a signal element of four different levels.
Figure below shows the imaginary one-dimensional and the actual four-dimensional implementation.

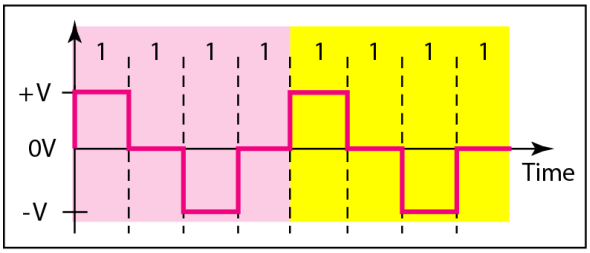



**Multiline Transition: MLT-3**
NRZ-I and differential Manchester are classified as differential encoding but use two transition rules to encode binary data (no inversion, inversion). If we have a signal with more than two levels, we can design a differential encoding scheme with more than two transition rules. The multiline transition, **three level (MLT-3) scheme** uses three levels (+V, 0, and - V) and three transition rules to move between the levels.
1. If the next bit is 0, there is no transition.
2. If the next bit is 1 and the current level is not 0, the next level is 0.
3. If the next bit is 1 and the current level is 0, the next level is the opposite of the last nonzero level.
The behavior of MLT-3 can best be described by the state diagram shown in Figure. The three voltage levels (-V, 0, and +V) are shown by three states (ovals).

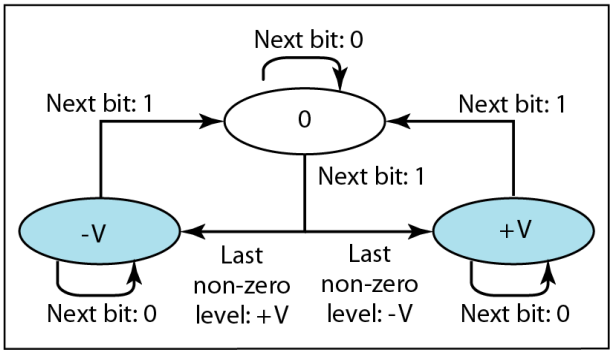
Let us look at the worst-case scenario, a sequence of 1s. In this case, the signal element pattern *+V0 – V0* is repeated every 4 bits. A non-periodic signal has changed to a periodic signal with the period equal to 4 times the bit duration. This worst-case situation can be simulated as an analog signal with a frequency one-fourth of the bit rate.



a. Typical case



b. Worse case



c. Transition states

**Summary of Line Coding**

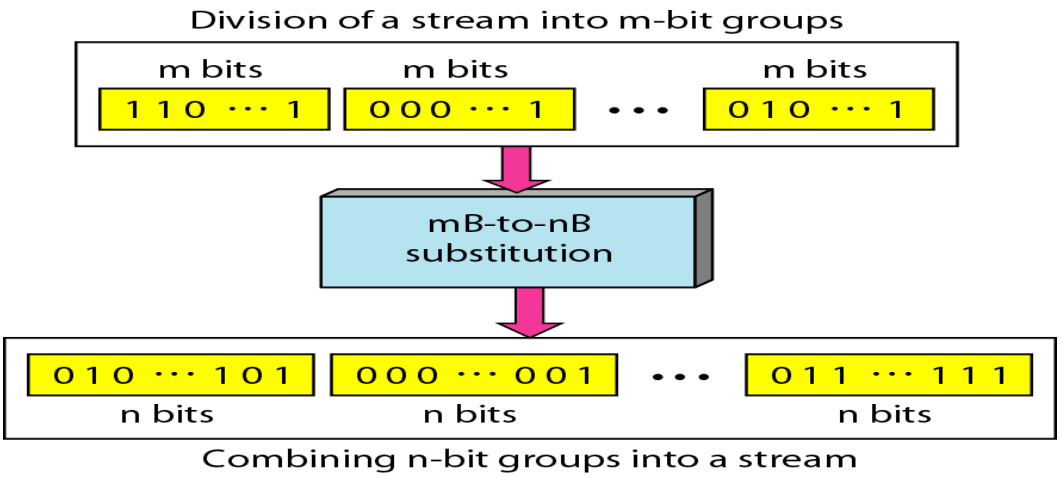
| *Category* | *Scheme* | *Bandwidth (average)* | *Characteristics* |
|---|---|---|---|
| Unipolar | NRZ | $B = N/2$ | Costly, no self-synchronization if long 0s or 1s, DC |
| Unipolar | NRZ-L | $B = N/2$ | No self-synchronization if long 0s or 1s, DC |
| | NRZ-I | $B = N/2$ | No self-synchronization for long 0s, DC |
| | Biphase | $B = N$ | Self-synchronization, no DC, high bandwidth |
| Bipolar | AMI | $B = N/2$ | No self-synchronization for long 0s, DC |
| Multilevel | 2B1Q | $B = N/4$ | No self-synchronization for long same double bits |
| | 8B6T | $B = 3N/4$ | Self-synchronization, no DC |
| | 4D-PAM5 | $B = N/8$ | Self-synchronization, no DC |
| Multiline | MLT-3 | $B = N/3$ | No self-synchronization for long 0s |

**Block Coding**

Block coding changes a block of *m* bits into a block of *n* bits, where *n* is larger than *m.* Block coding is referred to as an *mB/nB* encoding technique.

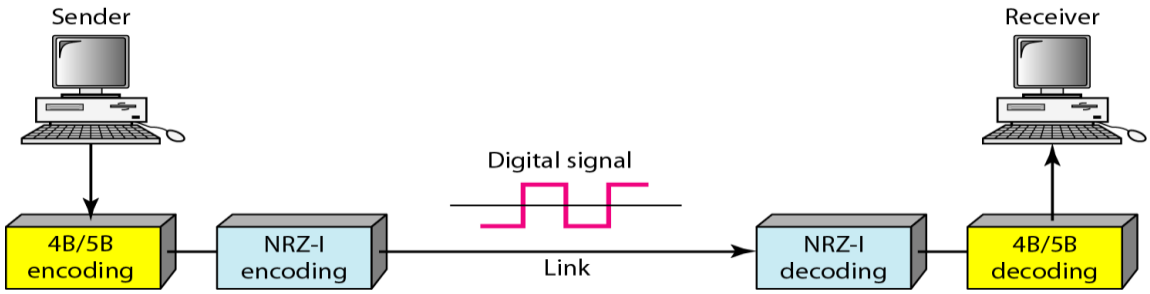
We need redundancy to ensure synchronization and to provide some kind of inherent error detection. Block coding can give us this redundancy and improve the performance of line coding. Block coding normally involves three steps: **division, substitution, and combination**. In the division step, a sequence of bits is divided into groups of *m* bits. For example, in 4B/5B encoding, the original bit sequence is divided into 4-bit groups. In substitution step, we substitute an m-bit group for an n-bit group. For example, in 4B/5B encoding we substitute a 4-bit code for a 5-bit

group. Finally (combination step), the n-bit groups are combined together to form a stream. The new stream has more bits than the original bits.




**4B/5B**

The four binary/five binary (4B/5B) coding scheme was designed to be used in combination with NRZ-I. A long sequence of 0s can make the receiver clock lose synchronization. One solution is to change the bit stream, prior to encoding with NRZ-I, so that it does not have a long stream of 0s. The 4B/5B scheme achieves this goal. The block-coded stream does not have more than three consecutive 0s. At the receiver, the NRZ-I encoded digital signal is first decoded into a stream of bits and then decoded to remove the redundancy.



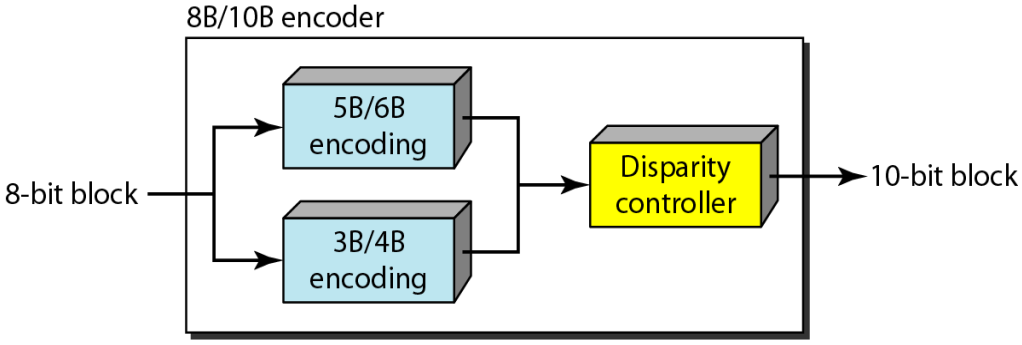

In 4B/5B, the 5-bit output that replaces the 4-bit input has no more than one leading zero (left bit) and no more than two trailing zeros (right bits). So when different groups are combined to make a new sequence, there are never more than three consecutive 0s. (Note that NRZ-I has no problem with sequences of 1s.). Below Table shows the corresponding pairs used in 4B/5B encoding.

A group of 4 bits can have only 16 different combinations while a group of 5 bits can have 32 different combinations. This means that there are 16 groups that are not used for 4B/5B encoding. Some of these unused groups are used for control purposes; the others provide a kind of error detection.

| *Data Sequence* | *Encoded Sequence* | *Control Sequence* | *Encoded Sequence* |
|---|---|---|---|
| 0000 | 11110 | Q (Quiet) | 00000 |
| 0001 | 01001 | I (Idle) | 11111 |
| 0010 | 10100 | H (Halt) | 00100 |
| 0011 | 10101 | J (Start delimiter) | 11000 |
| 0100 | 01010 | K (Start delimiter) | 10001 |
| 0101 | 01011 | T (End delimiter) | 01101 |
| 0110 | 01110 | S (Set) | 11001 |
| 0111 | 01111 | R (Reset) | 00111 |
| 1000 | 10010 | | |
| 1001 | 10011 | | |
| 1010 | 10110 | | |
| 1011 | 10111 | | |
| 1100 | 11010 | | |
| 1101 | 11011 | | |
| 1110 | 11100 | | |
| 1111 | 11101 | | |

**8B/10B**

In eight binary/ten binary (8B/10B) encoding, a group of 8 bits of data is substituted by a 10-bit code. The 8B/10B block coding is actually a combination of 5B/6B and 3B/4B encoding, as shown in Figure.

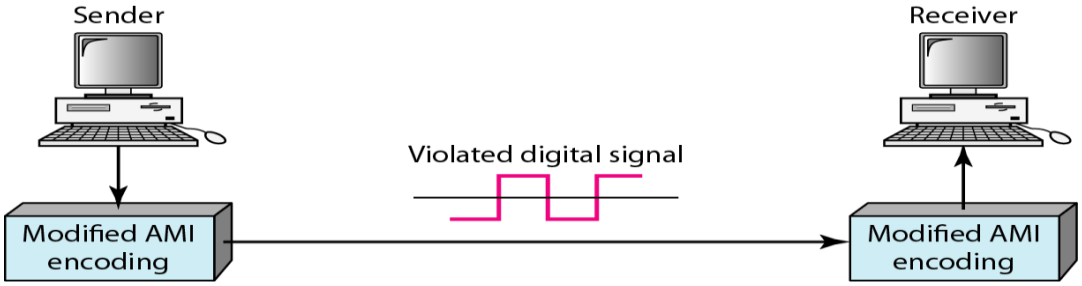



The most 5 significant bits of a 10-bit block is fed into the 5B/6B encoder; the least 3 significant bits is fed into a 3B/4B encoder. The split is done to simplify the mapping table. To prevent a long run of consecutive 0s or 1s, the code uses a disparity controller which keeps track of excess 0s over 1s (or 1s over 0s). The coding has $2^{10} - 2^8 = 768$ redundant groups that can be used for disparity checking and error detection.

**Scrambling**

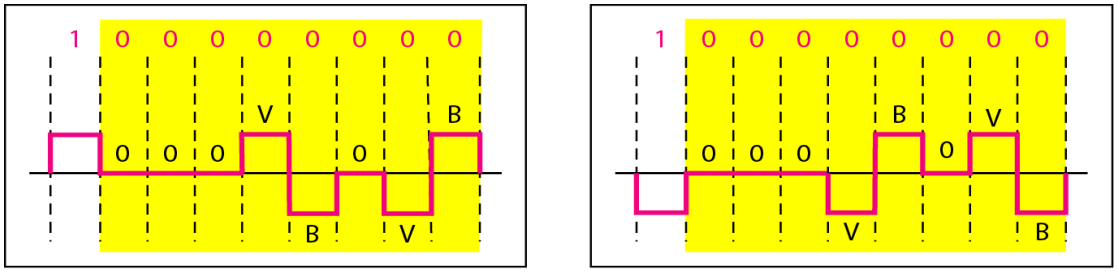
Biphase schemes are not suitable for long-distance communication because of their wide bandwidth requirement. The combination of block coding and NRZ line coding is not suitable for long-distance encoding, because of the DC component. Bipolar AMI encoding, has a narrow bandwidth and does not create a DC component. However, a long sequence of 0s upsets the

synchronization. We are looking for a solution that substitutes long zero-level pulses with a combination of other levels to provide synchronization. One solution is **scrambling**.




**B8ZS (Bipolar with S-zero substitution)**

In this technique, eight consecutive zero-level voltages are replaced by the sequence **000VB0VB**. The V in the sequence denotes violation; this is a nonzero voltage that breaks an AMI rule of encoding (opposite polarity from the previous). The B in the sequence denotes bipolar; which means a nonzero level voltage in accordance with the AMI rule.




a. Previous level is positive.

b. Previous level is negative.

**HDB3 (High density bipolar 3 zero)**

In this technique, four consecutive zero-level voltages are replaced with a sequence of 000V or B00V. The reason for two different substitutions is to maintain the even number of nonzero pulses after each substitution.

The two rules can be stated as follows:

1. If the number of nonzero pulses after the last substitution is **odd**, the substitution pattern will be **000V**, which makes the total number of nonzero pulses even.
2. If the number of nonzero pulses after the last substitution is **even**, the substitution pattern will be **B00V**, which makes the total number of nonzero pulses even.