Line Coding Schemes

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.
- Different types of signal elements can be allowing different signal levels.
- If we have **L different levels**, then we can produce **L^n combinations of signal patterns**.

$$2^m \le L^n$$

- The data element and signal element relation is **mBnL coding**, 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.
- B (binary, L=2), T (tenary, L=3), and Q (quaternary, L=4).

Line Coding Schemes

- Now we have 2^m symbols and L^n signals.
- If $2^m > L^n$ then we cannot represent the data elements, we don't have enough signals.
- If $2^m = L^n$ then we have an exact mapping of one symbol on one signal.
- If $2^m < L^n$ then we have more signals than symbols and we can choose the signals that are more distinct to represent the symbols and therefore have better noise immunity and error detection as some signals are not valid.

In *m*B*n*L schemes, a pattern of *m* data elements is encoded as a pattern of *n* signal elements in which $2^m \leq L^n$.

Line Coding Schemes

2B1Q

- 2B1Q (two binary,one quaternary)
- m=2,n=1,and L=4
- The signal rate (baud rate)

$$S = cN\frac{1}{r} = \frac{1}{2} \times N \times \frac{1}{2} = \frac{N}{4}$$

	Previous level: positive	Previous level: negative
Next bits	Next level	Next level
00	+1	-1
01	+3	-3
10	-1	+1
11	-3	+3

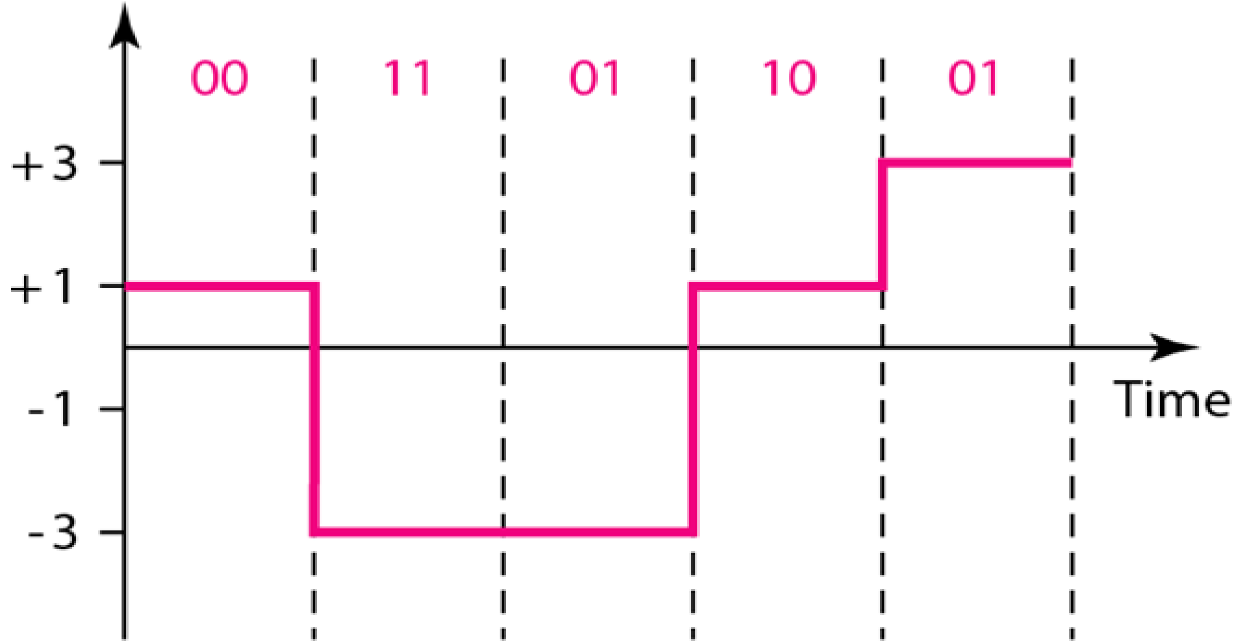
Transition table

- 2B1Q is used in DSL (digital subscriber line) technology to provide a high-speed connection to the Internet by using subscriber telephone lines.

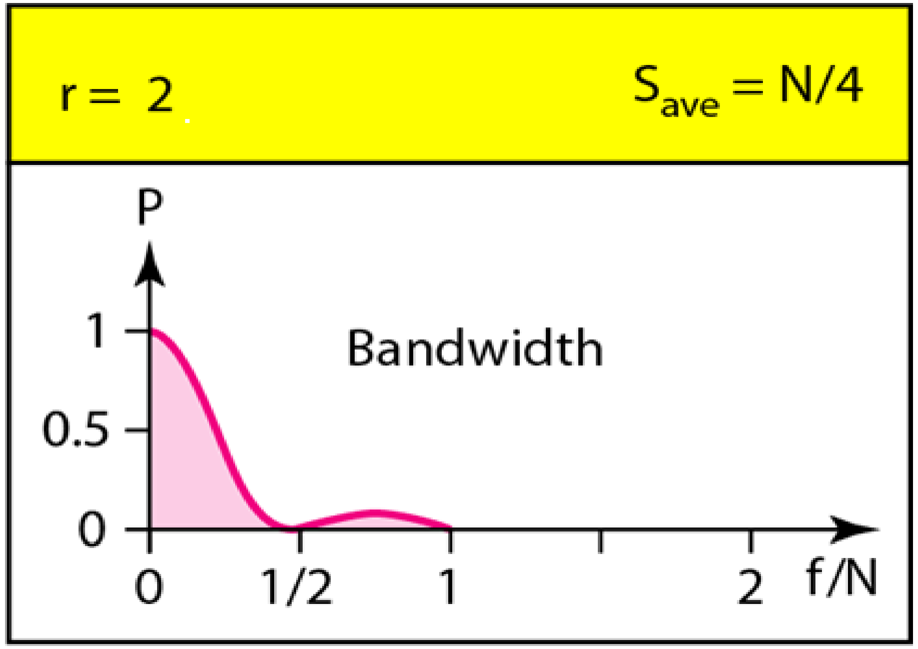
Line Coding Schemes

	Previous level: positive	Previous level: negative
Next bits	Next level	Next level
00	+1	-1
01	+3	-3
10	-1	+1
11	-3	+3

Transition table



Assuming positive original level

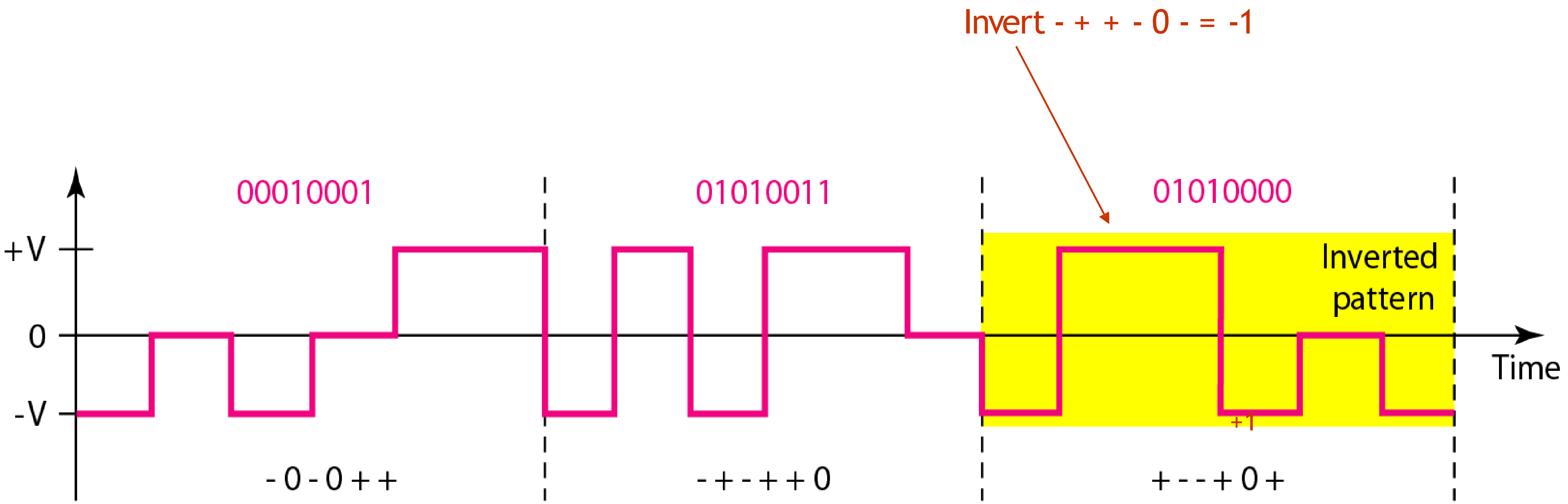


Line Coding Schemes

8B6T

- Eight binary,six ternary (8B6T)
- This code is used with 100BASE-4T cable.
- Encode a pattern of 8 bits as a pattern of 6 signal elements, where the signal has three levels (ternary).
- 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 (direct-current) balance.
- To make whole stream DC-balanced, the sender keeps track of the weight

Line Coding Schemes



Line Coding Schemes

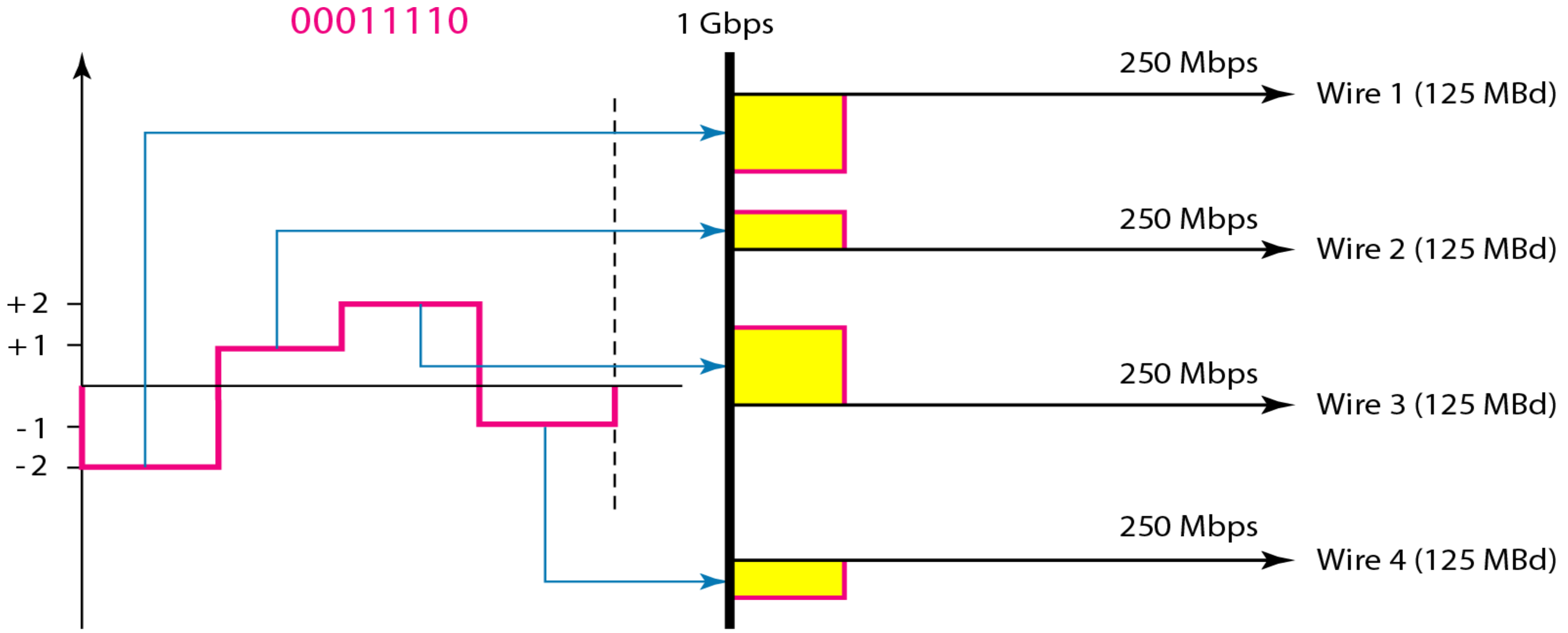
4D-PAM5

- Four-dimensional five-level pulse amplitude modulation (4D-PAM5)
- 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.
- The 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.

Line Coding Schemes

- The worst signal rate for this imaginary one-dimensional version is Nx4/8, or N/2.
- 4D-PAM5 sends data over four channels (four wires).
- This means the signal rate can be reduced to N/8.
- All 8 bits can be fed into a wire simultaneously and sent by using one signal element.
- Gigabit Ethernet use this technique to send 1-Gbps data over four copper cables that can handle 1Gbps/8 = 125Mbaud

Line Coding Schemes



Line Coding Schemes

Multiline Transmission: MLT-3

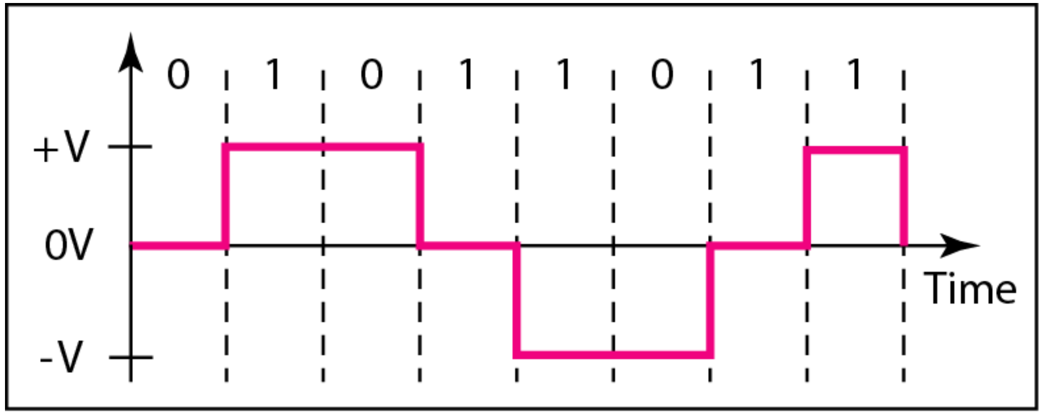
- The multiline transmission, three level (MLT-3)
- Three levels (+V, 0, and –V) and three transition rules to move the levels
  - If the next bit is 0, there is no transition
  - If the next bit is 1 and the current level is not 0, the next level is 0.
  - If the next bit is 1 and the current level is 0, the next level is the opposite of the last nonzero level.

Line Coding Schemes

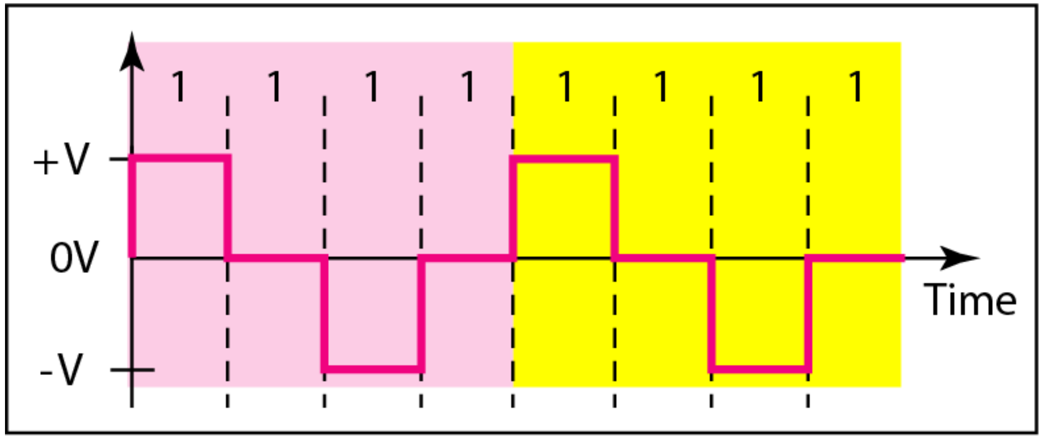
<u>Why do we need to use MLT-3?</u>

- The signal rate for MLT-3 is one-fourth the bit rate (N/4).
- This makes MLT-3 a suitable choice when we need to send 100 Mbps on a copper wire that cannot support more than 32 MHz (frequencies above this level create electromagnetic emission).
- This results in a bandwidth requirement that is equivalent to the bit rate.
- In some instances, the bandwidth requirement may even be lower, due to repetitive patterns resulting in a periodic signal.

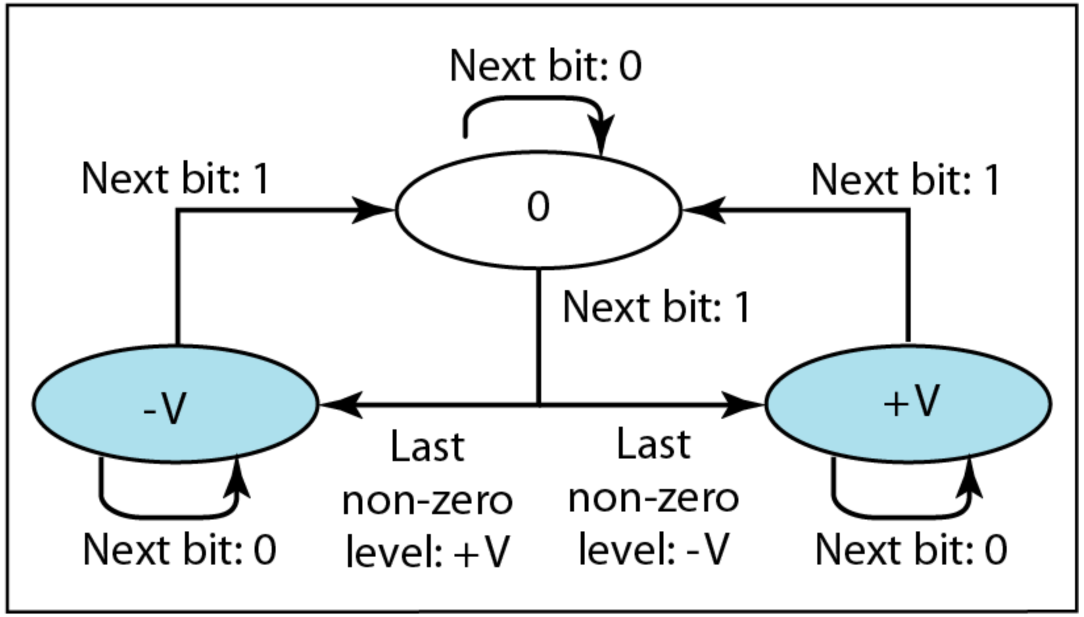
Line Coding Schemes



a. Typical case



b. Worse case



c. Transition states

Summary of line coding schemes

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

Numerical

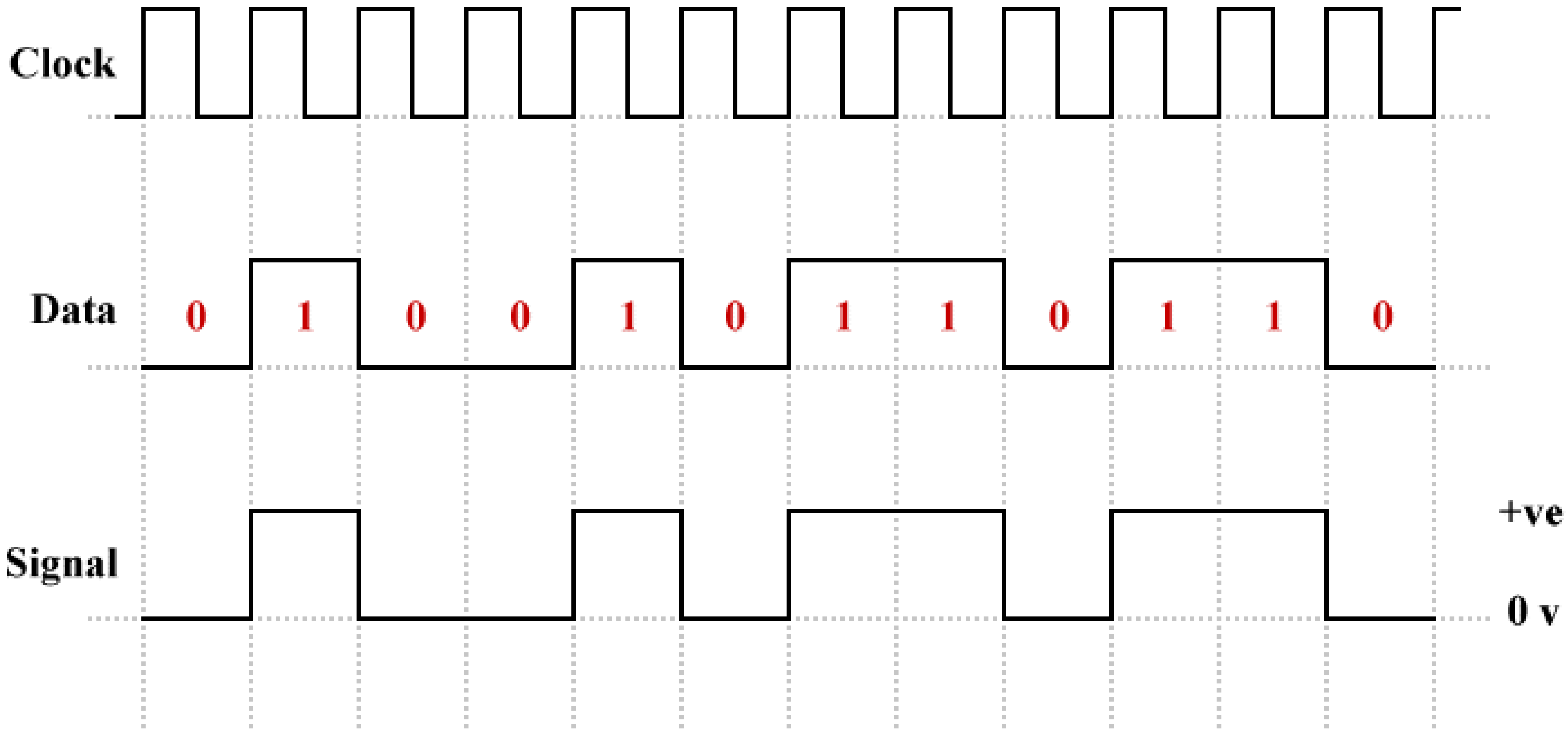
Sketch the line code waveform mentioned below for the following binary data sequence

Sequence = { 0 1 0 0 1 0 1 1 0 1 1 0 }

1. Unipolar NRZ & RZ
2. Polar NRZ-level &RZ
3. Bipolar (AMI)
4. Bipolar (pseudoternary)
5. Manchester encoding
6. Differential Manchester
7. 2B1Q
8. MLT-3

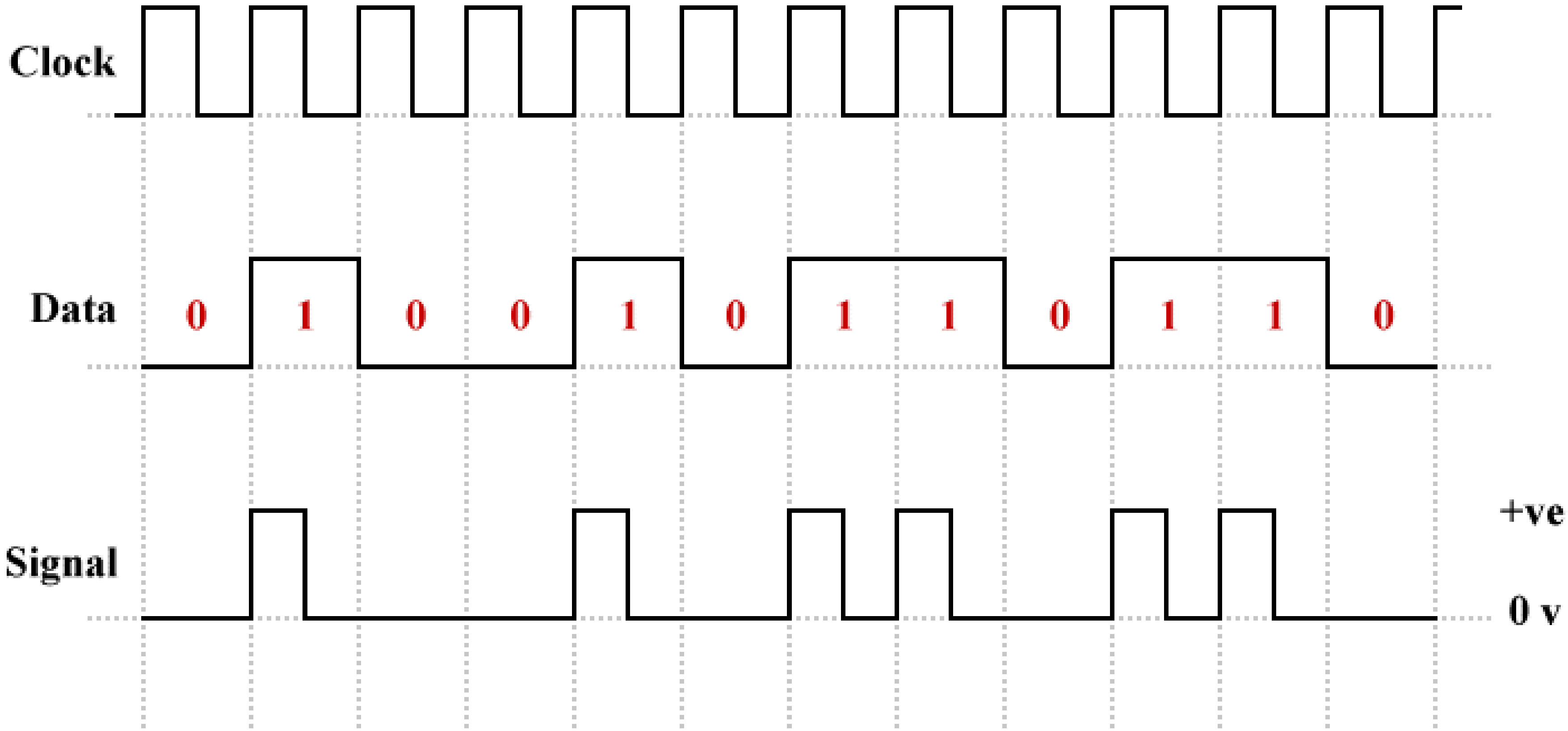
Numerical

- **Unipolar Non-Return-to-Zero (NRZ)**



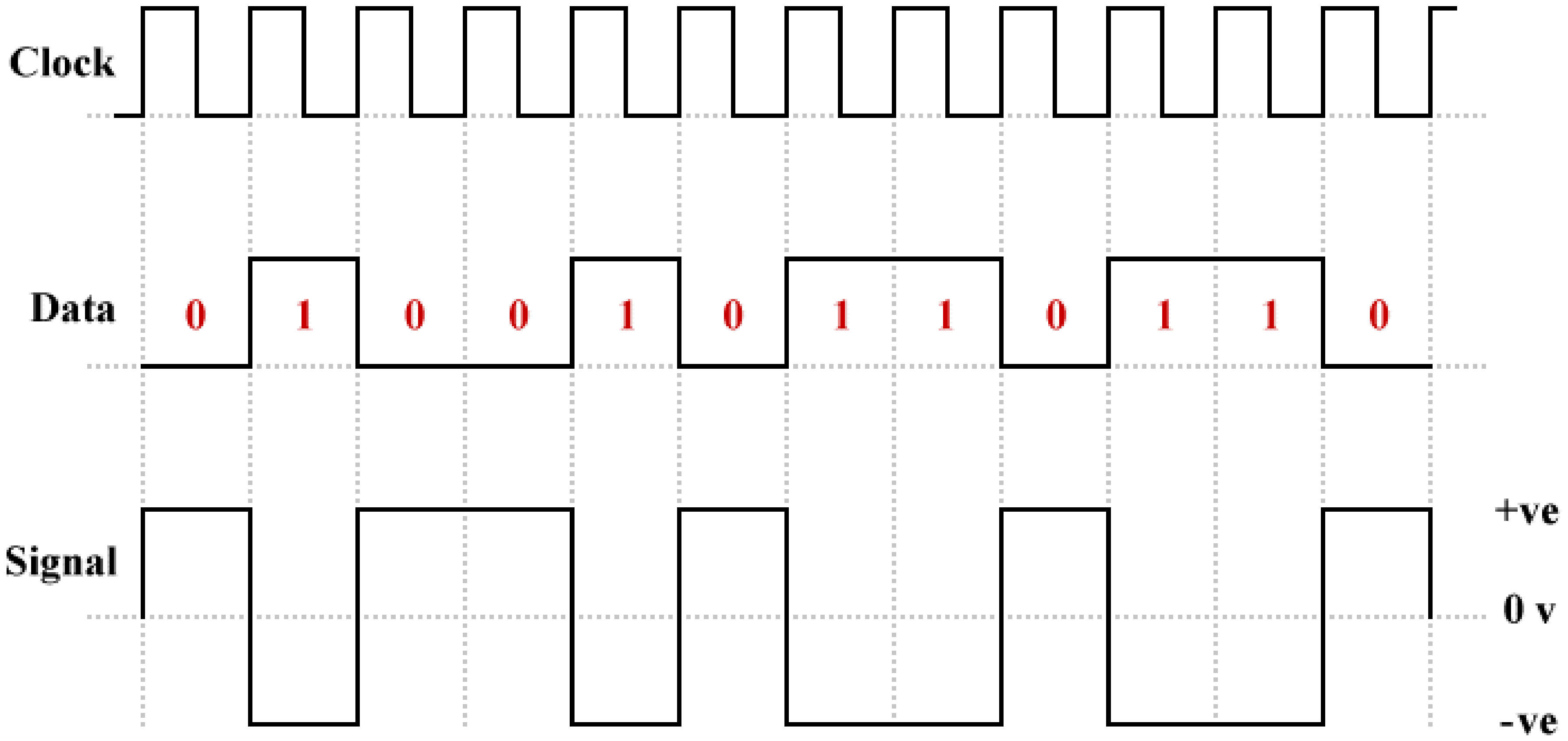
Numerical

- **Unipolar Return-to-Zero (RZ)**



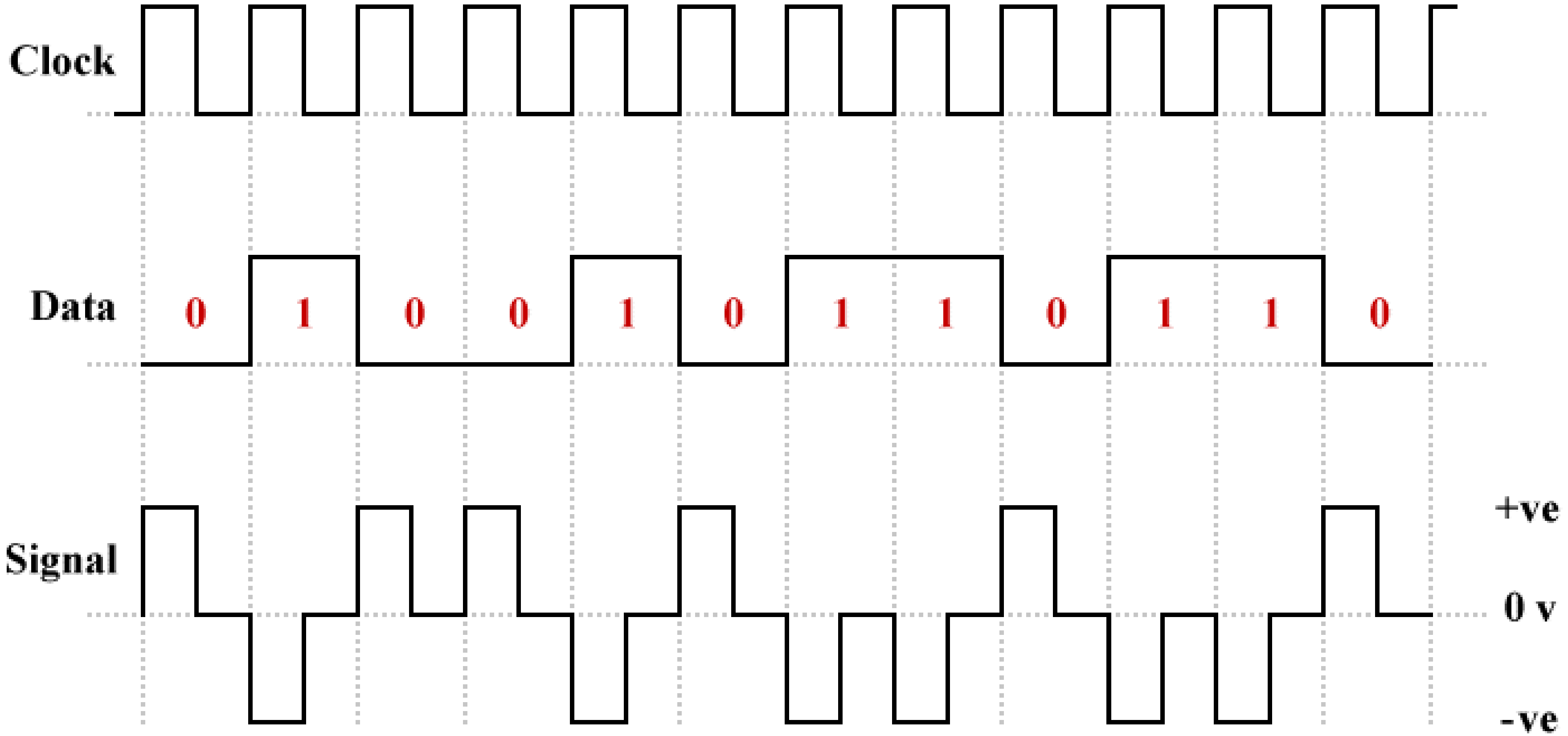
Numerical

- **Polar NRZ-level (NRZ-L)**



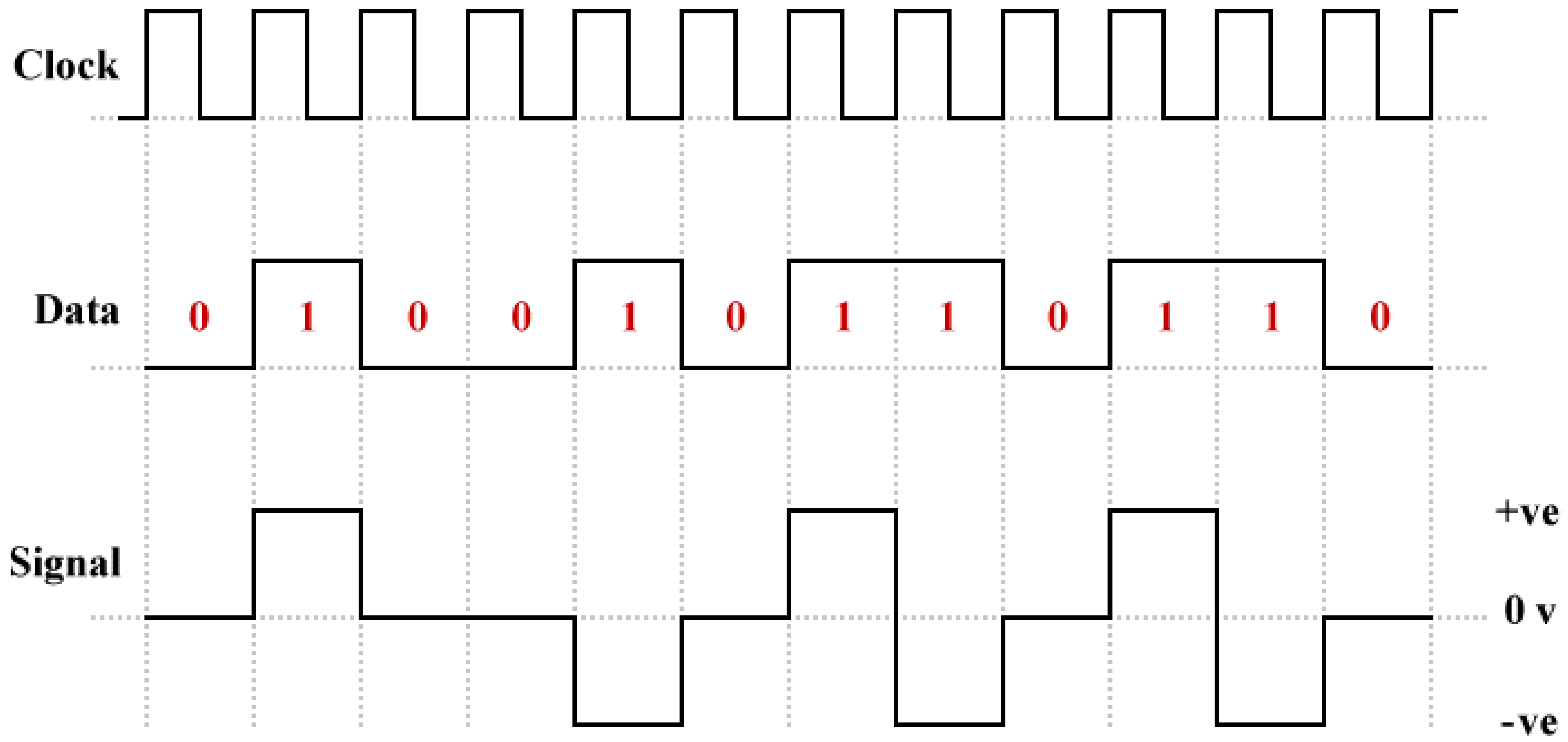
Numerical

- **Polar RZ**



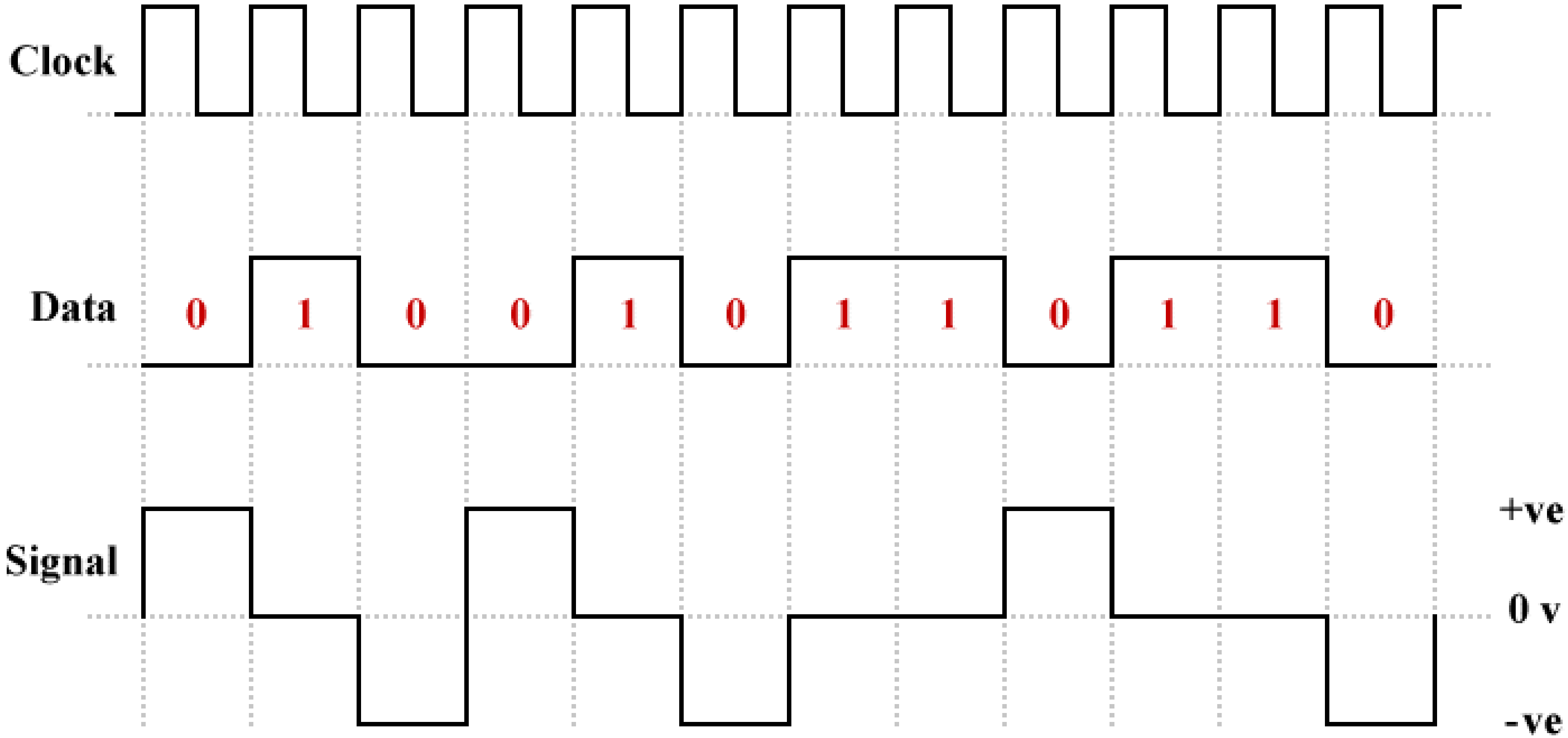
Numerical

- **Bipolar (AMI)**



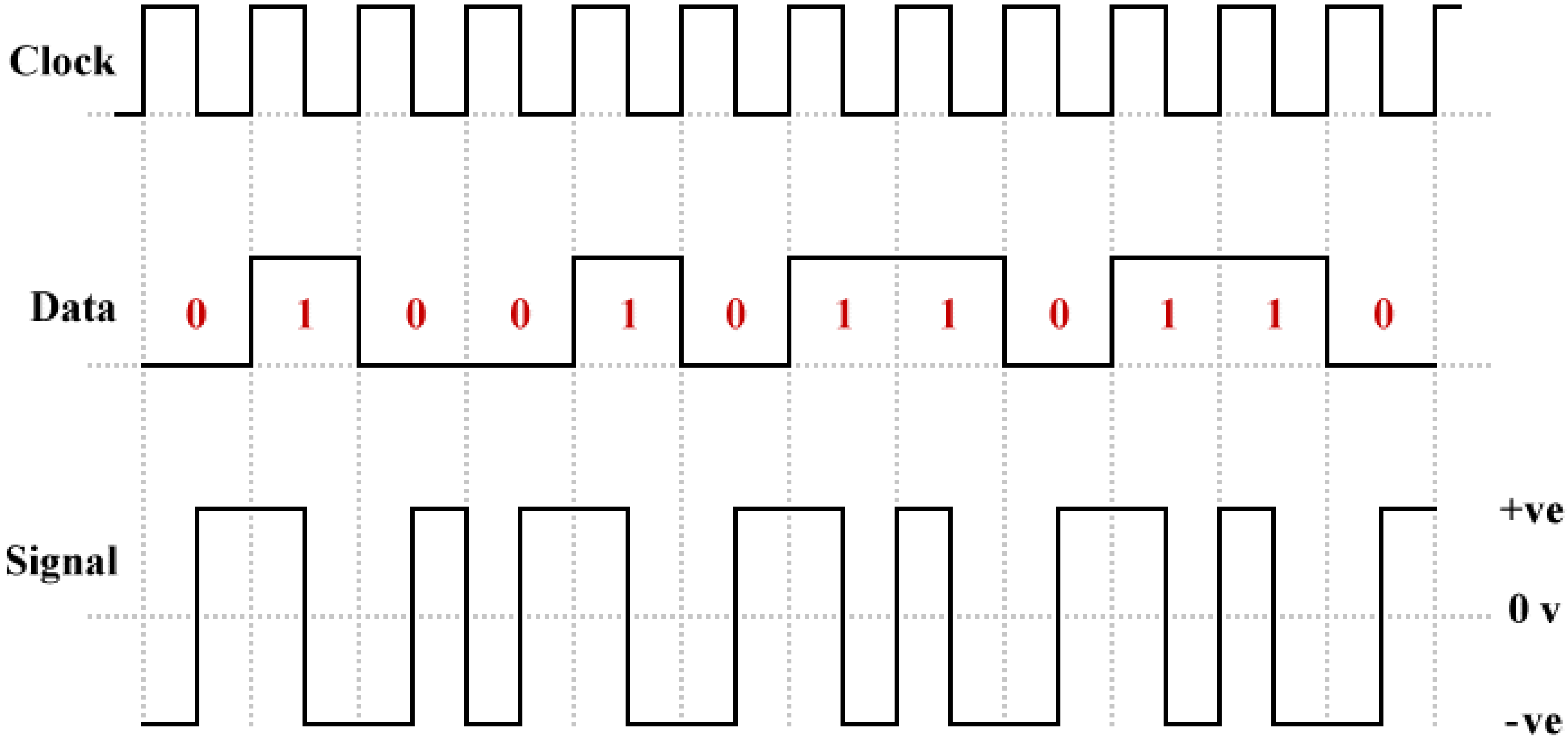
Numerical

- **Bipolar (Pseudoternary)**



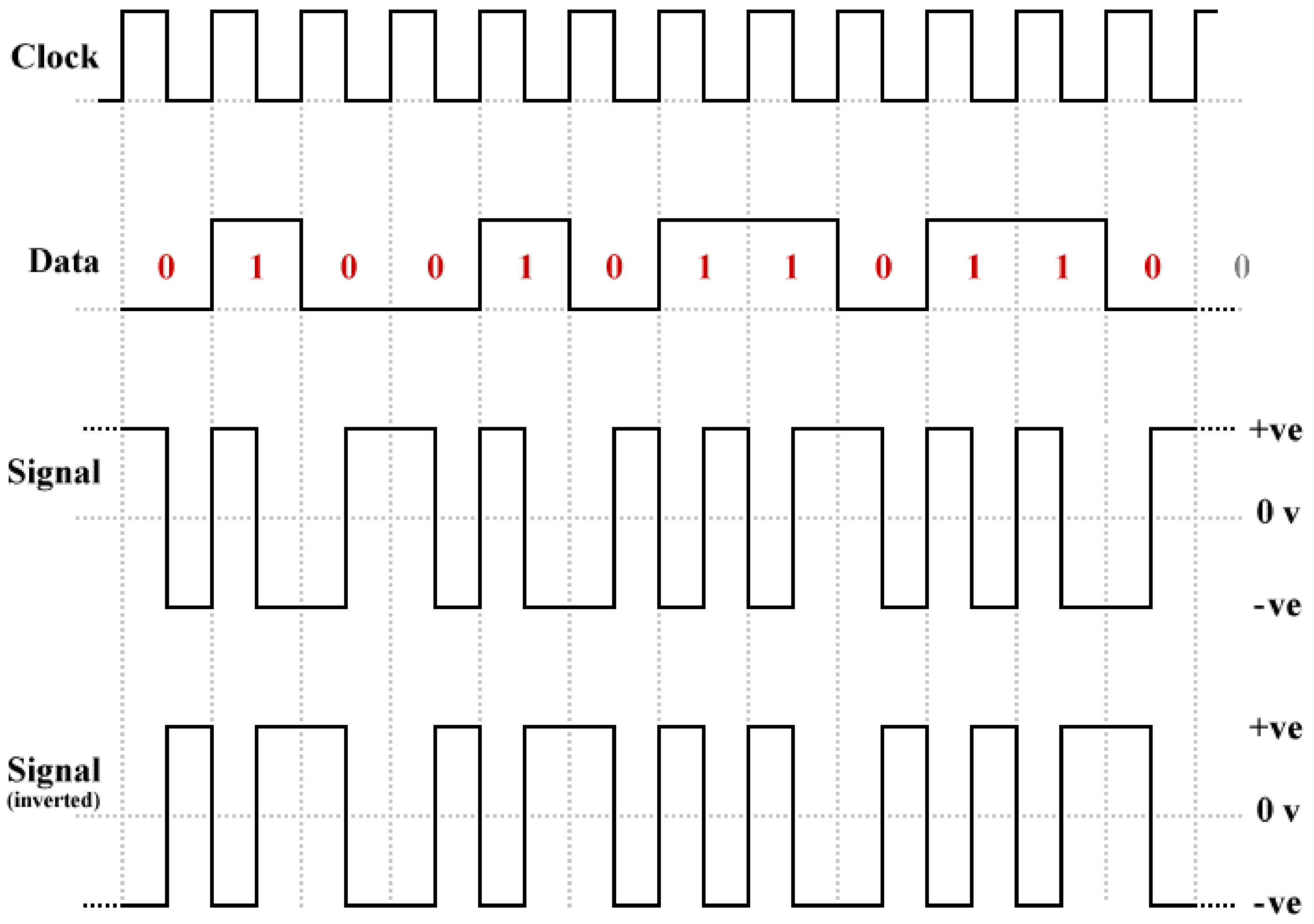
Numerical

- **Manchester encoding**



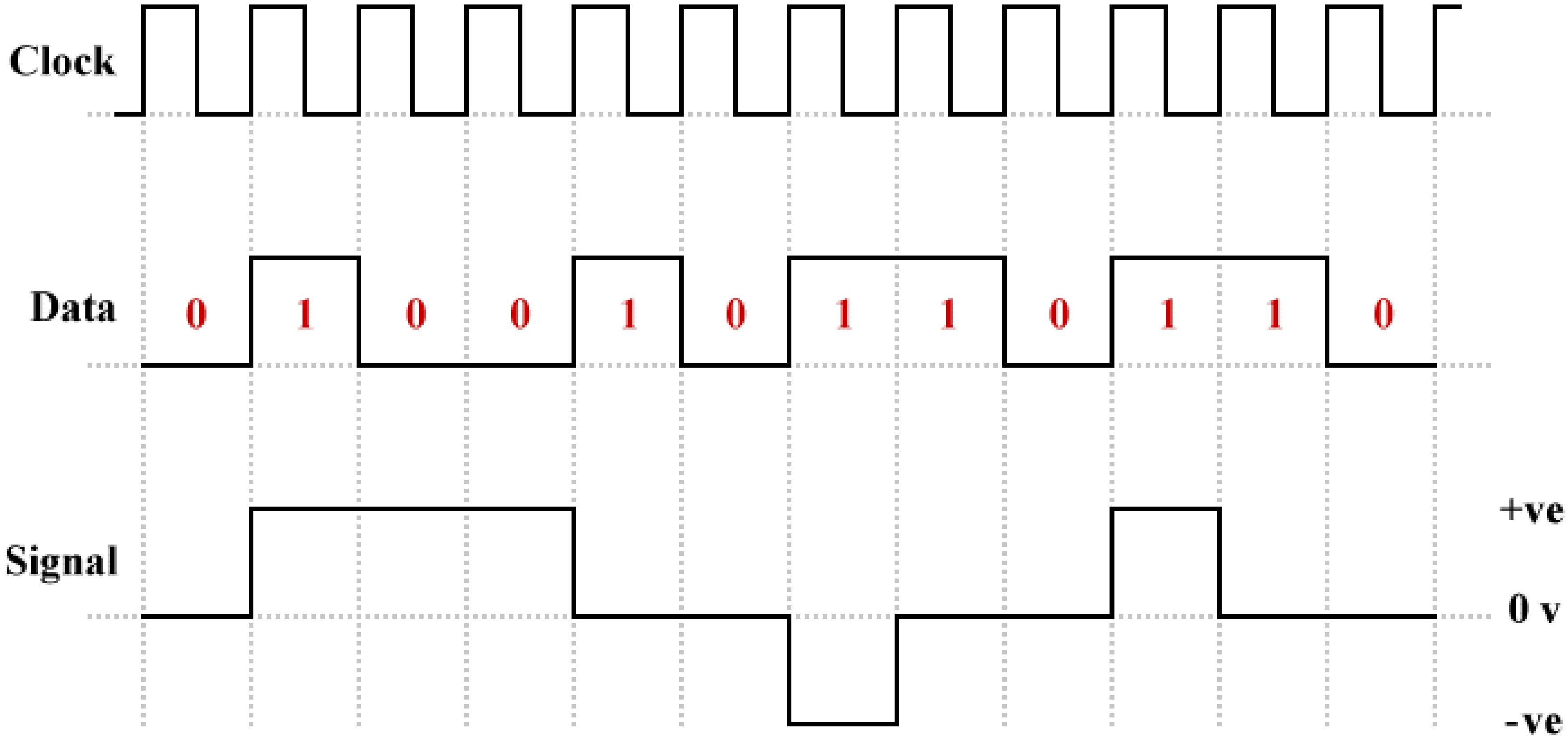
Numerical

- **Differential Manchester**



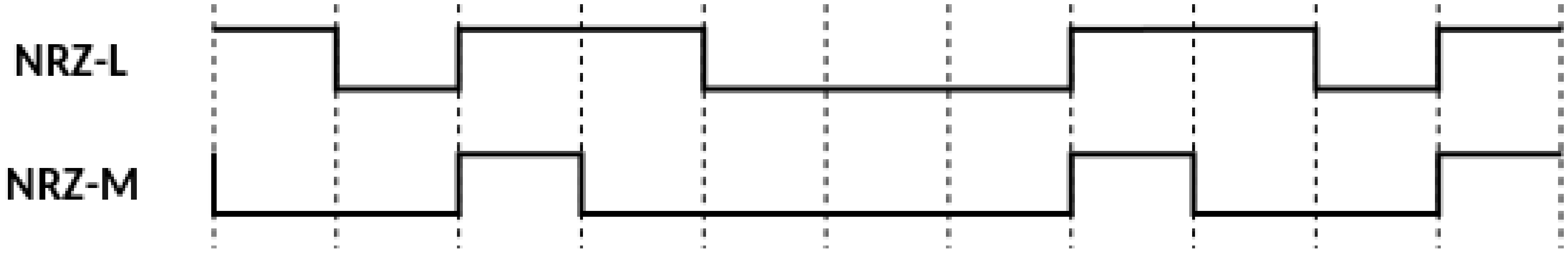
Numerical

- **Multi-level transmit (MLT-3)**



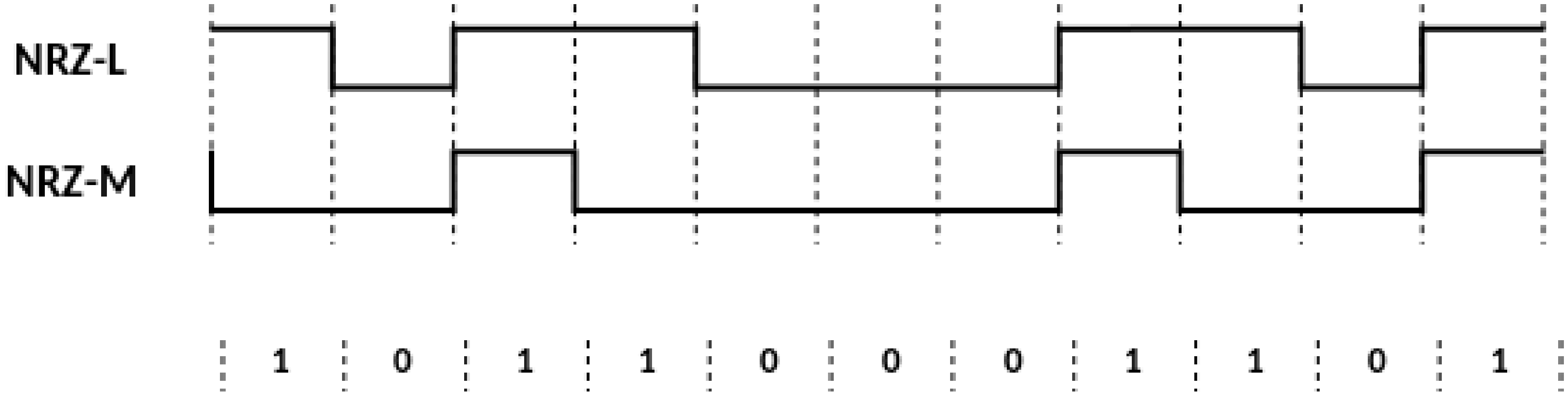
Numerical

- **Find the bit sequence**



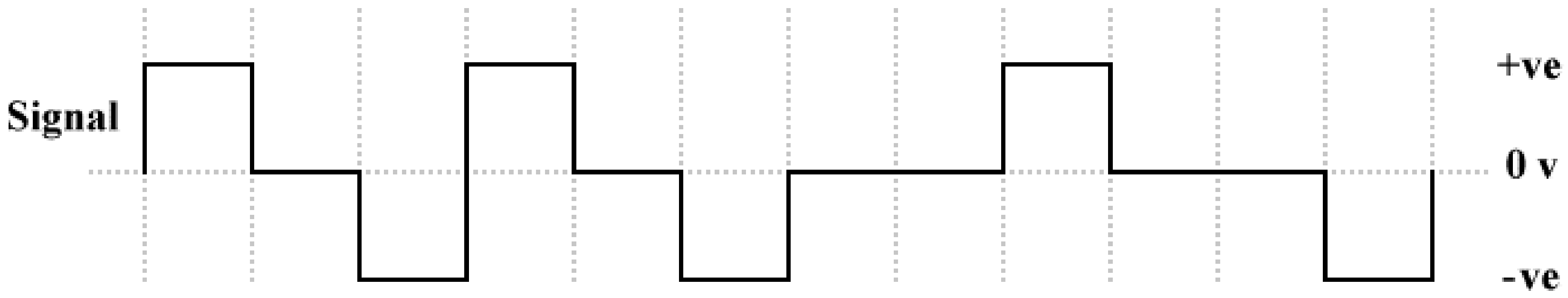
Numerical

- **Find the bit sequence**



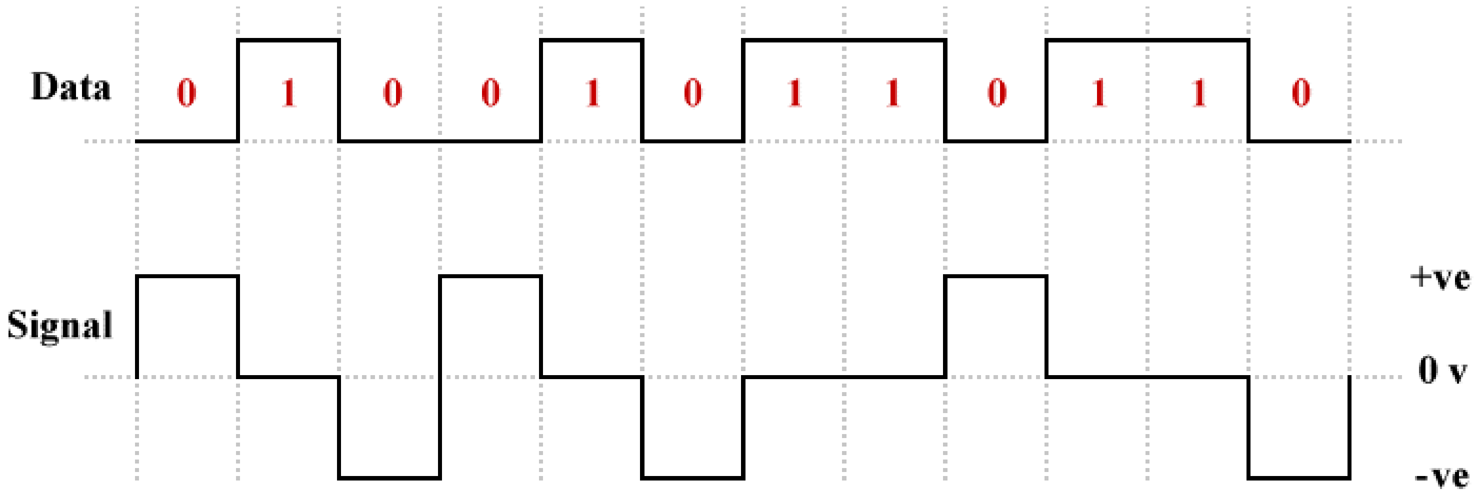
Numerical

- **Find the bit sequence of Pseudoternary code**



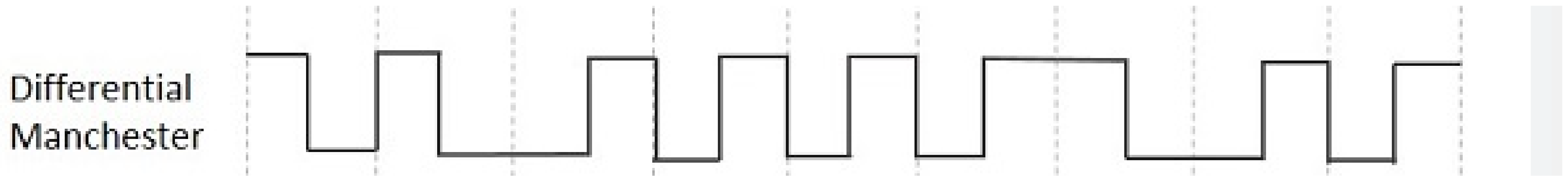
Numerical

- **Find the bit sequence of Pseudoternary code**



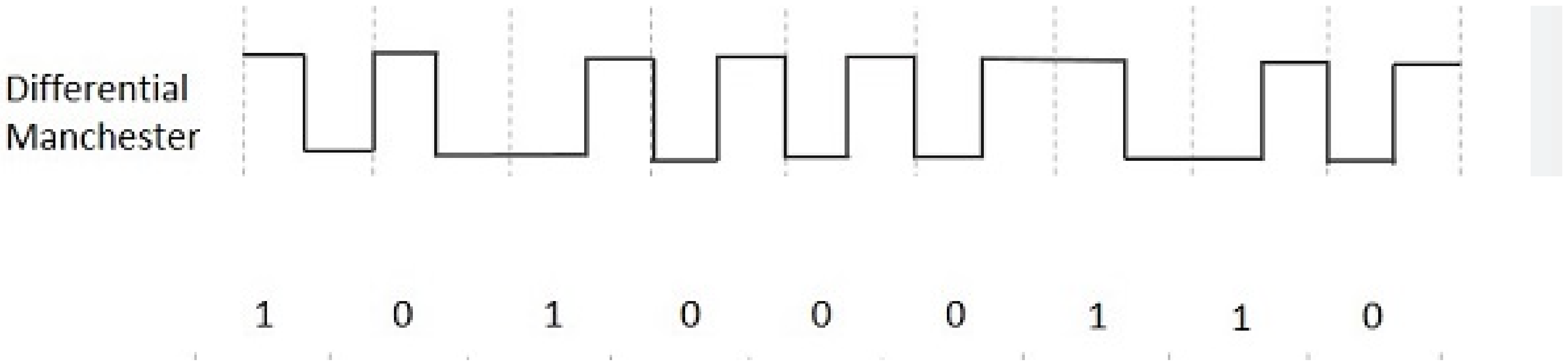
Numerical

- **Find the bit sequence.**



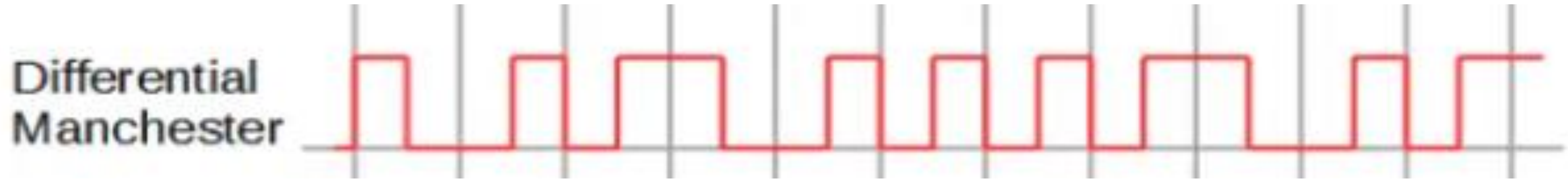
Numerical

- **Find the bit sequence.**



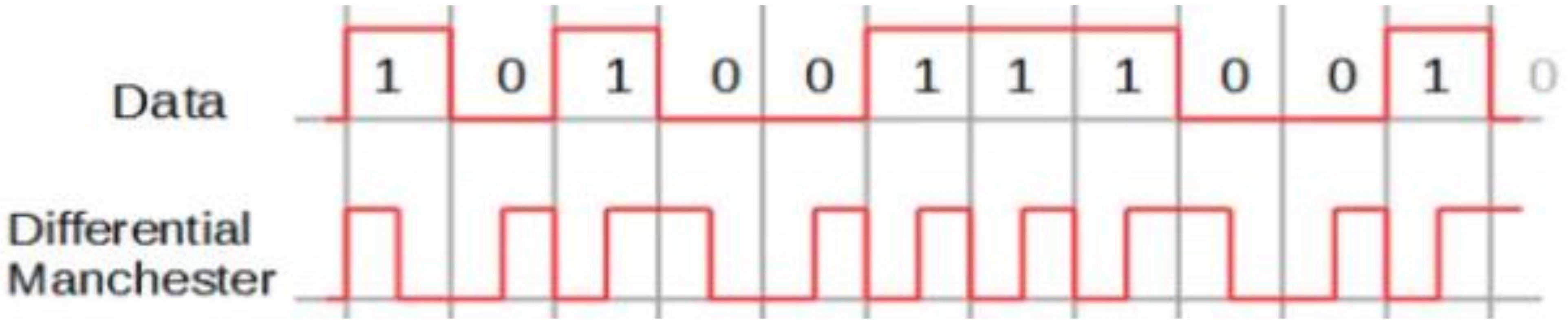
Numerical

- **Find the bit sequence.**



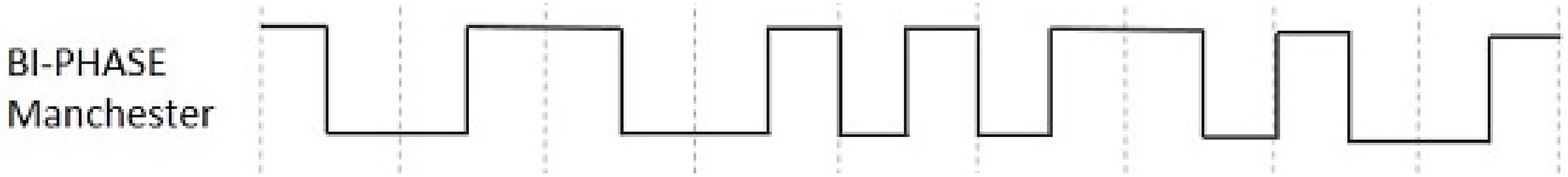
Numerical

- **Find the bit sequence.**



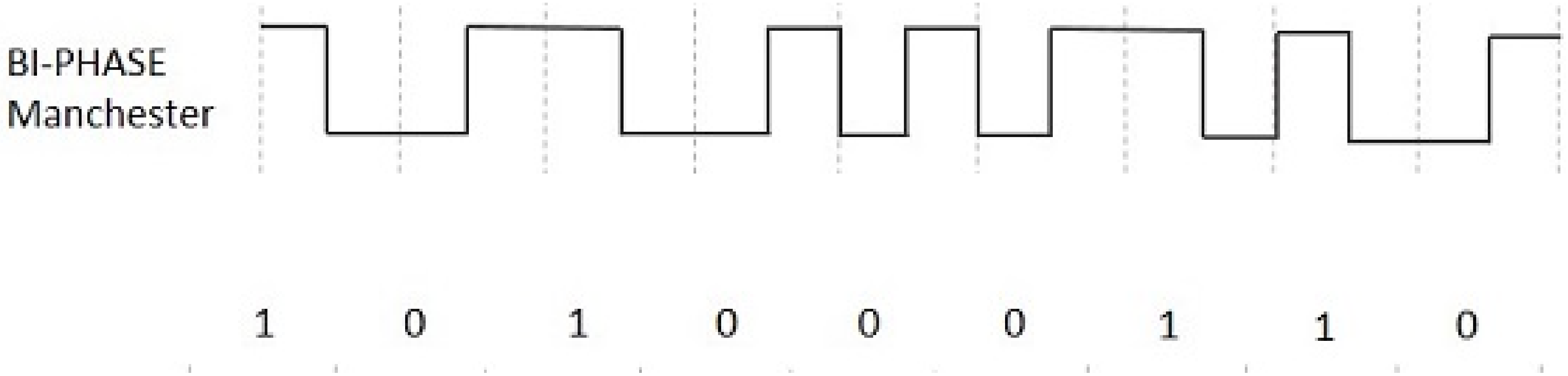
Numerical

- **Find the bit sequence.**



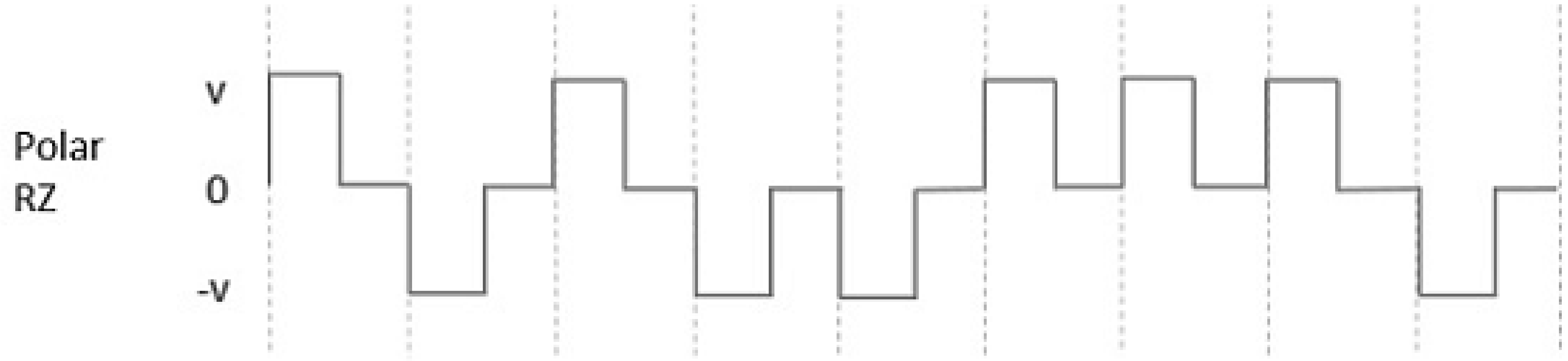
Numerical

- **Find the bit sequence.**



Numerical

- **Find the bit sequence.**



Numerical

- **Find the bit sequence.**