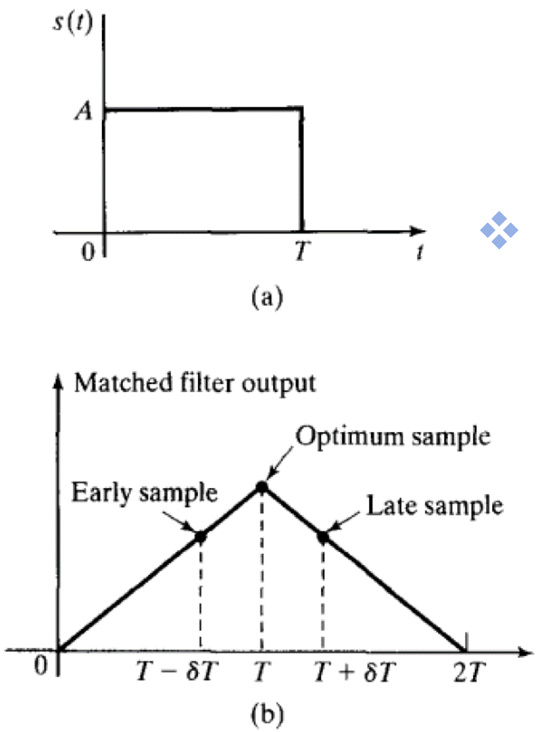
Early-Late Gate Synchronization

- Basic Idea: exploit the symmetry properties of the output signal of matched filter or correlator



- Due to the symmetry, the values of the correlation function at the early samples $t = T - \delta T$ and the late samples $t = T + \delta T$ are equal.
- Thus, the proper sampling time is the midpoint between $t = T - \delta T$ and $t = T + \delta T$

Figure 8.48 (a) Rectangular signal pulse and (b) its matched filter output.

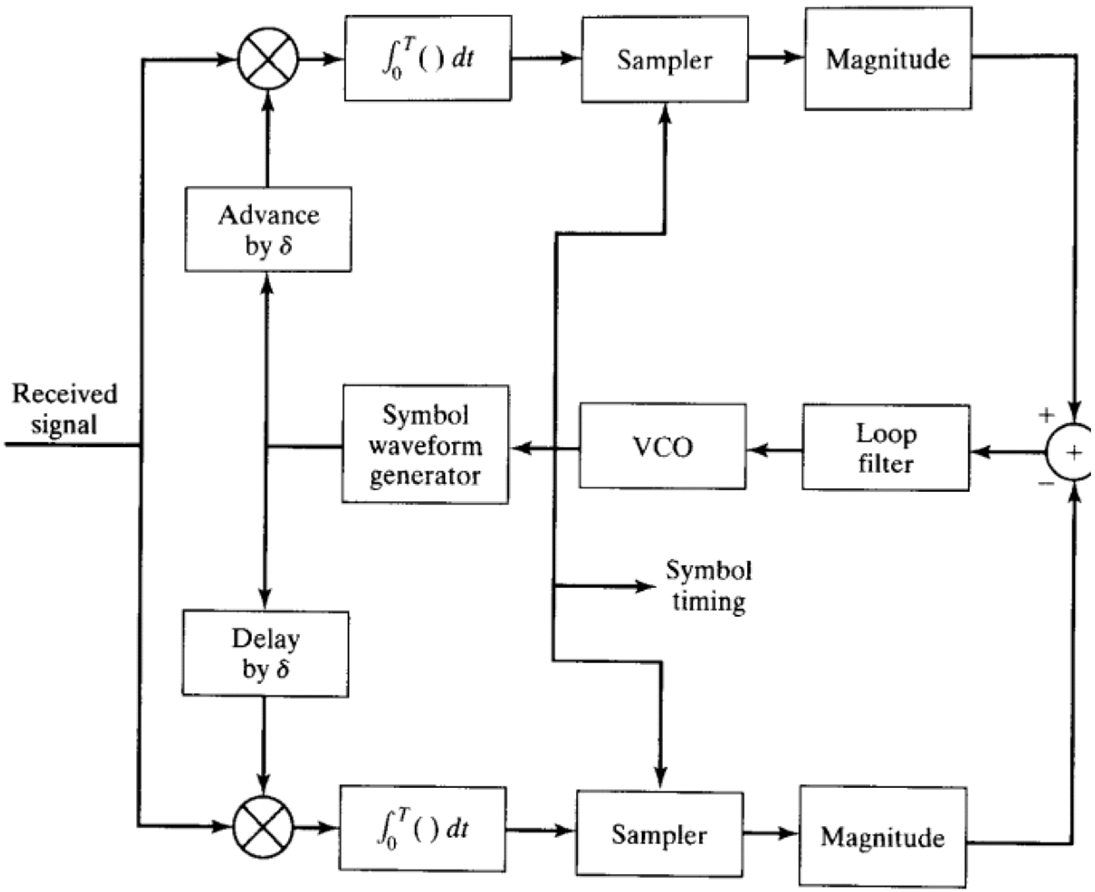
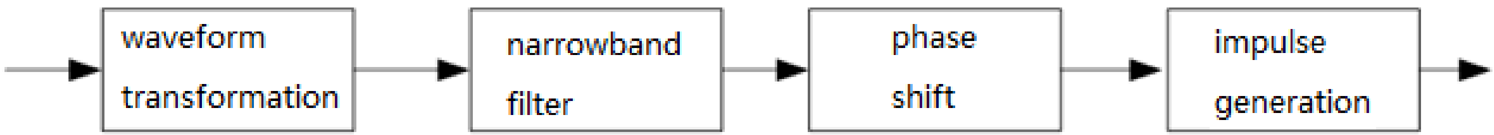


Figure 8.49 Block diagram of early–late gate synchronizer.

Nonlinear-transformation-based method

1. Nonlinear-transformation-based method

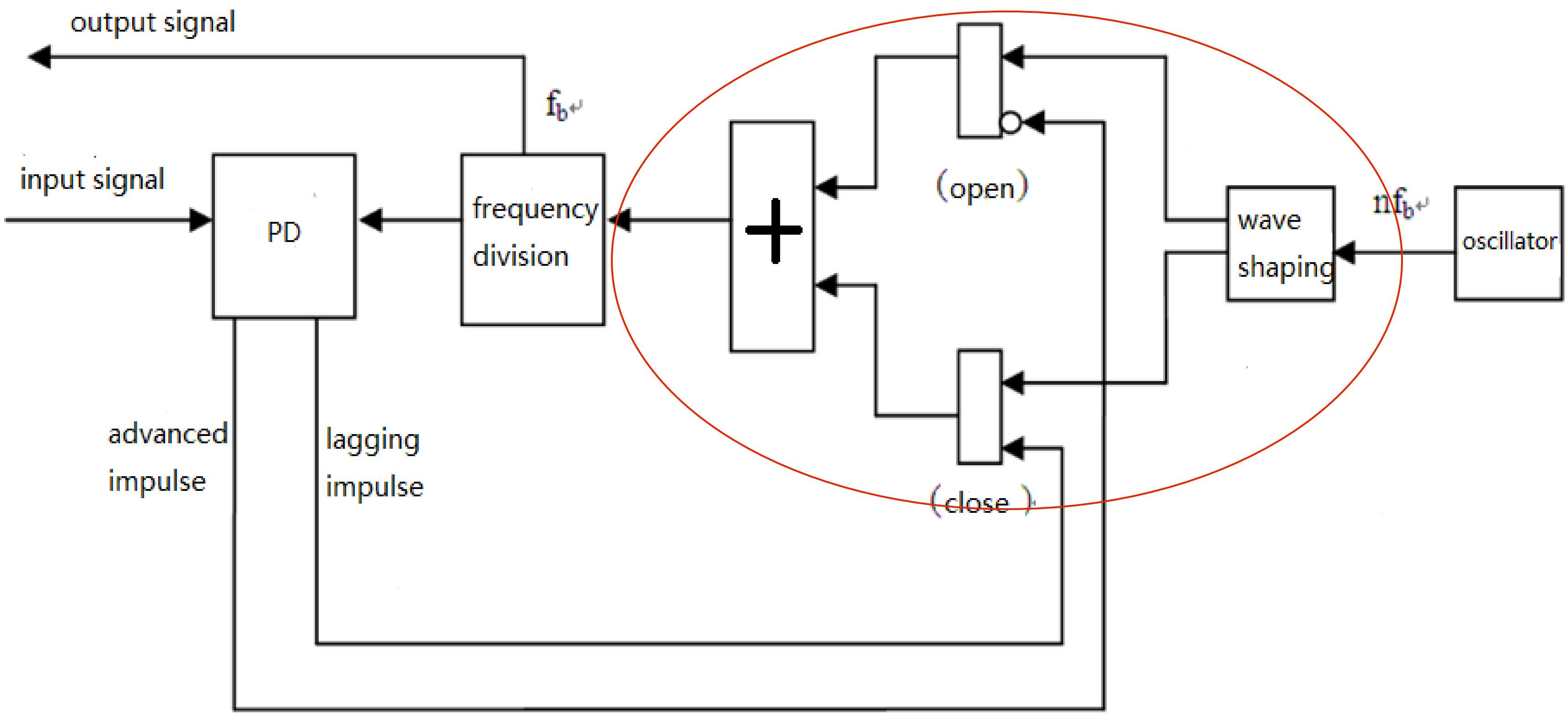


Some transformations can add synchronous signal with f=1/T to the original signal. For example, we can transform the signal to return-to-zero waveform. After narrowband filtering and phase shifting, we can generate the clock signal used for synchronization.

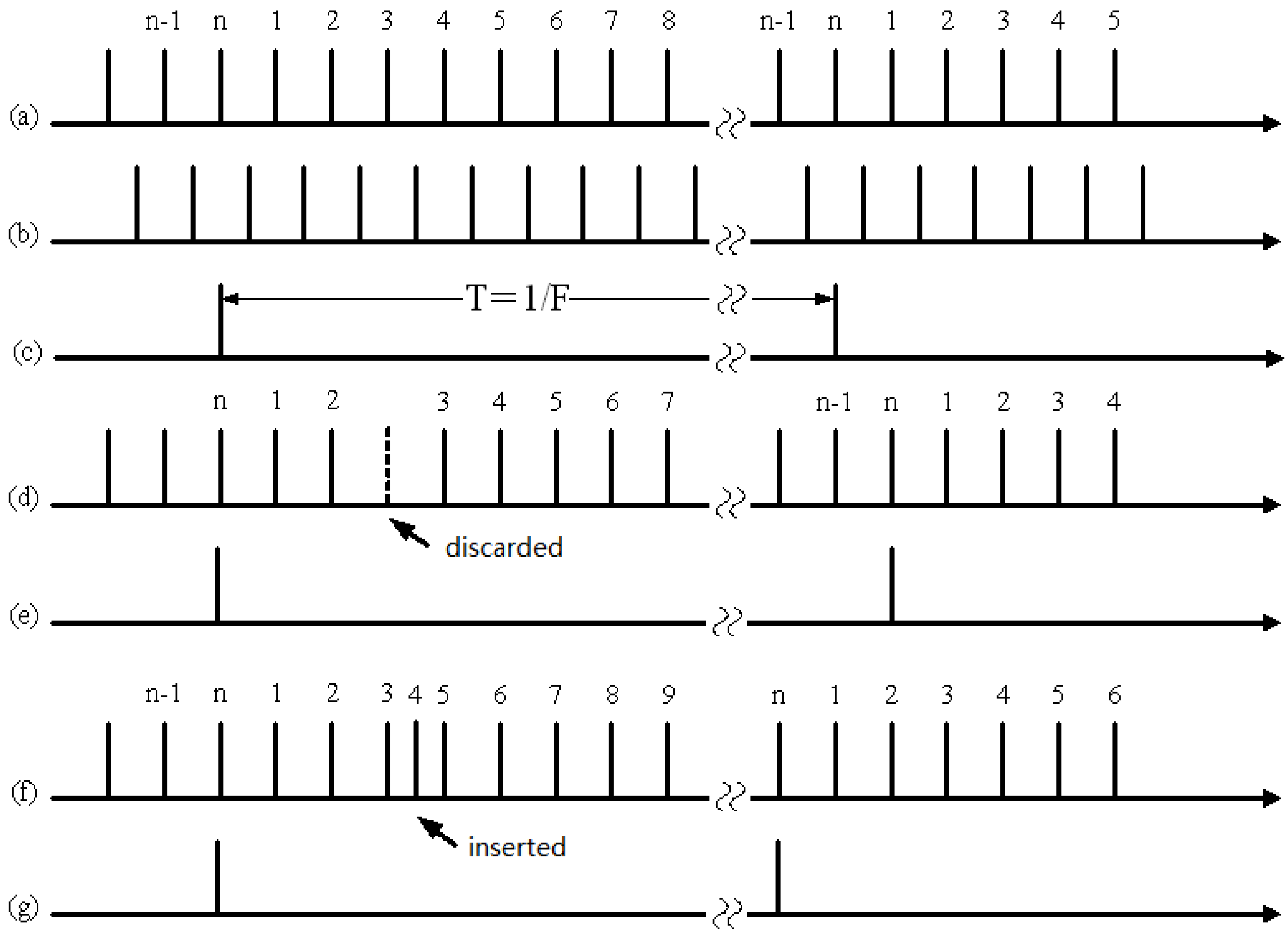
$$P_s(f) = f_s P(1-P)\left|G_1(f) - G_2(f)\right|^2 + f_s^2 \sum_{m=-\infty}^{\infty} \left|P G_1(mf_s) + (1-P) G_2(mf_s)\right|^2 \delta(f - mf_s)$$

Digital PLL (DPLL)

2. DPLL



Digital PLL (DPLL)



Performance

3. Performance of symbol synchronization system

—DPLL

1). Phase error

2). Synchronization build time

3). Synchronization hold time

4). Synchronous bandwidth

Frame Synchronization

- Recall that carrier and symbol synchronization needs to estimate the phase of synchronous signal which can be realized by using a PLL.
- Frame synchronization is to insert frame alignment signal (distinctive bit sequence) and then detect the alignment symbol.
- Besides adding frame alignment bits, some code such as self-synchronizing code can be synchronized without adding extra bits.
- Here, we only focus on the first method ——inserting frame alignment signal.

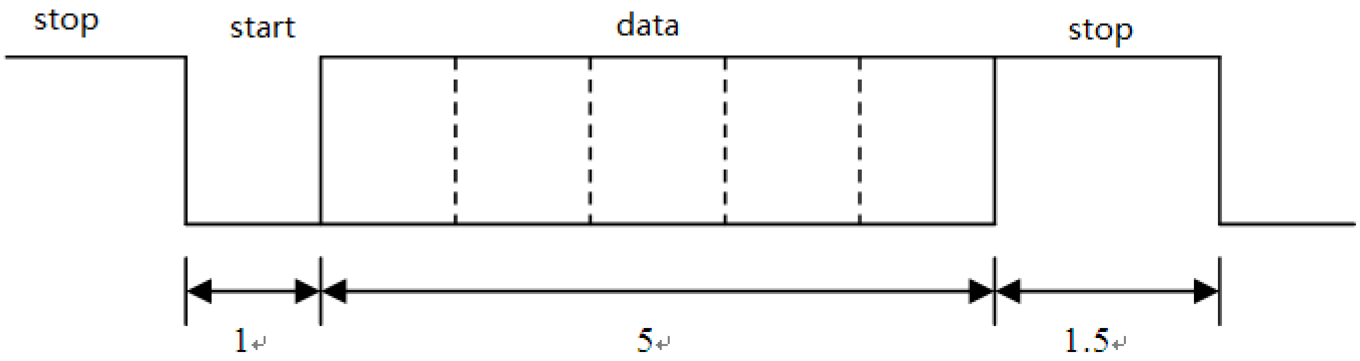
Frame Synchronization

- Start-stop method
- Bunched frame alignment signal
- Distributed frame alignment signal

Start-stop Method

1. Start-stop method

It is widely used in teleprinter. Each symbol contains 5-8 data bits, a start bit and a stop bit.



start bit: "0", width: T_b

stop bit: "1", width $\geq T_b$

System will keep sending stop bit when it is idle. When "1" $\rightarrow$ "0", the receiver will start to receive a data symbol.

Start-stop Method

Drawbacks:

1). Low transmission efficiency

2). Low timing accuracy

Bunched frame alignment signal

2. Bunched frame alignment signal

This method inserts synchronous code at a particular place in each frame. The code should have a sharp self-correlation function. The detector should be simple to implement.

Frame synchronization code:

- Barker code
- optimal synchronous code
- pseudo-random code.

Barker Code

(1) Barker code:

A n bits barker code $\{x_1, x_2, x_3 \cdots$ =+1 or −1. its self-correlation function satisfies:

$$R_x(j) = \sum_{i=1}^{n-j} x_i x_{i+j} = \begin{cases} n & j = 0 \\ 0 \text{or} \pm 1 & 0 < j < n \\ 0 & j \geq n \end{cases}$$

Barker code is not a periodic sequence. It is proved that when $n < 12100$, we can only find barker code with $n = 2, 3, 4, 5, 7, 11, 13$.

Barker Code

n	barker code
2	+ +
3	+ + -
4	+ + + - , + + - +
5	+ + + - +
7	+ + + - - + -
11	+ + + - - - + - - + -
13	+ + + + + - - + + - + - +

Barker Code

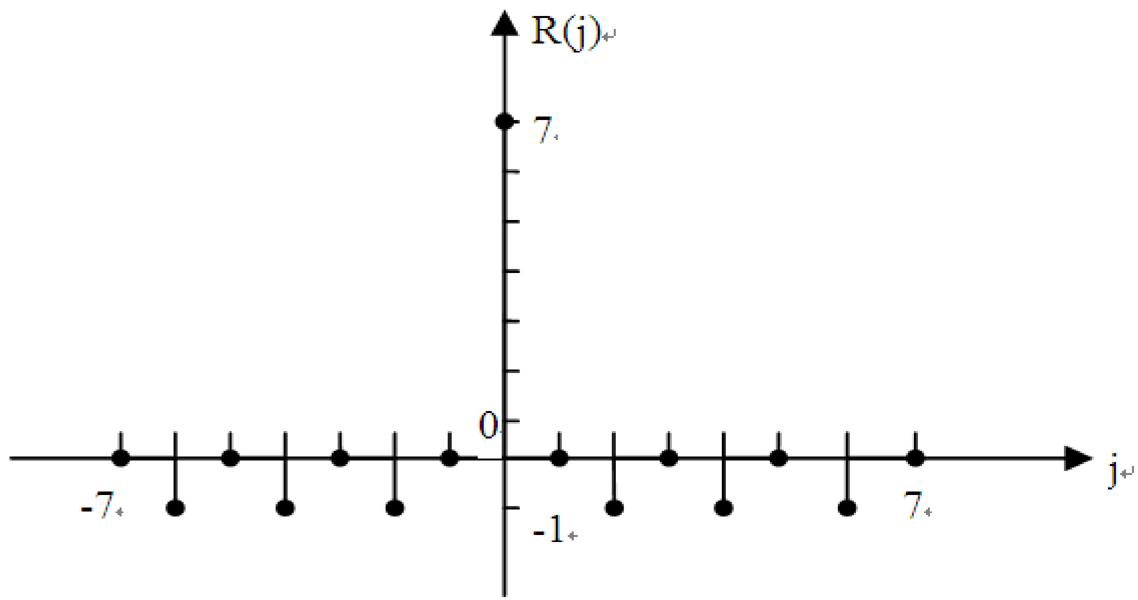
Example: A barker code with $n = 7$, find its self-correlation function

$j = 0 : R_x(0) = x_1 x_1 + x_2 x_2 + \cdots = 7$

$j = 1 : R_x(1) = x_1 x_2 + x_2 x_3 + \cdots$

Similarly, we can determine $R_x(j)$.

The result is shown below, we can see it has a sharp peak when $j = 0$.

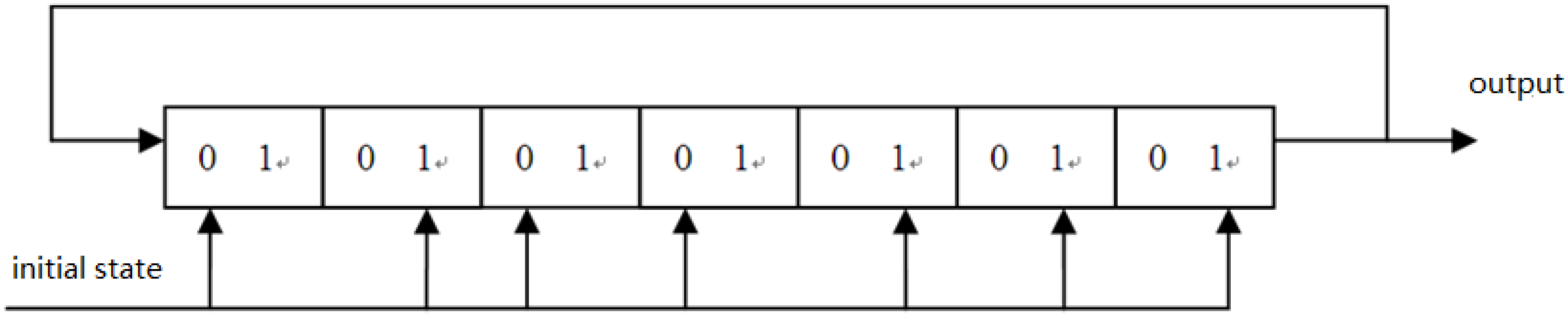


Barker Code

(2) Barker code generator

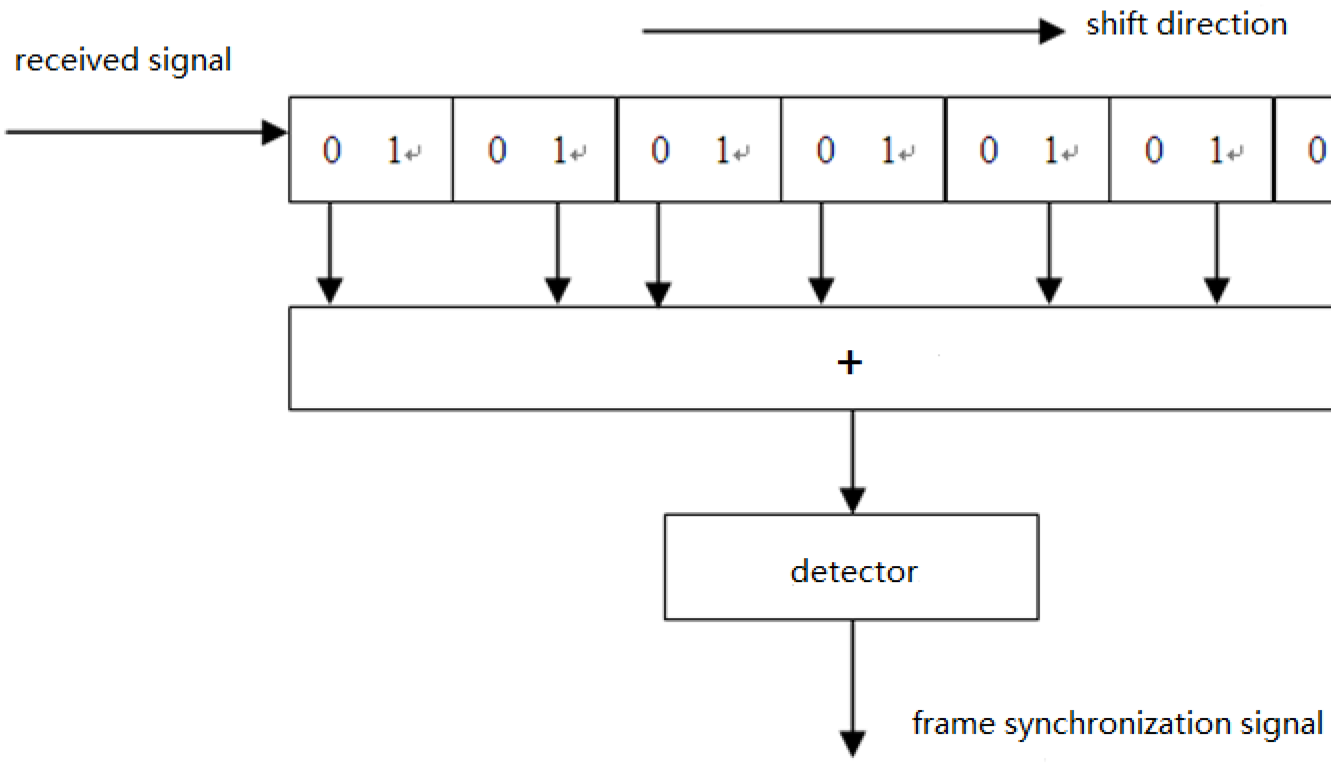
—shift register

Example： when n=7, a 7 bits shift register. The initial state is a barker code.



Barker Code

(3) Barker code detector



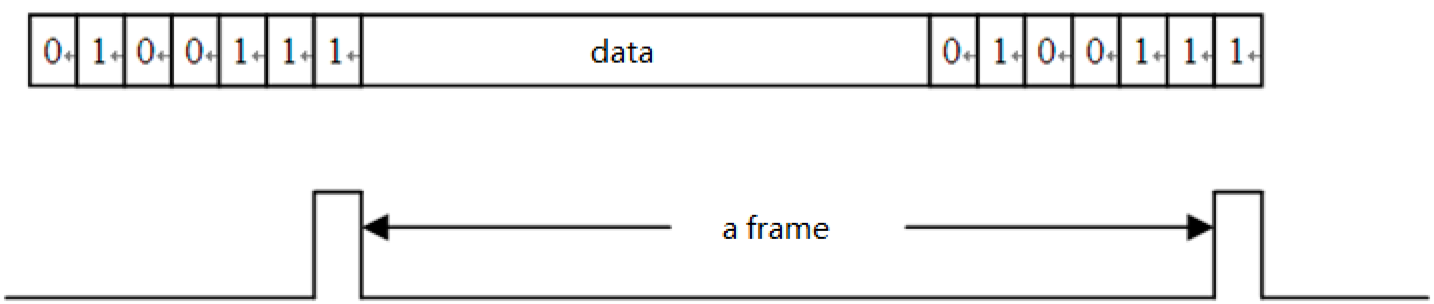
Barker Code

The barker code detector follows:

$$input: "1" \begin{cases} \text{output } ''1'' :? \ +1 \\ \text{output } ''0'' :? \ -1 \end{cases}$$

$$input: "0" \begin{cases} \text{output } ''1'' :? \ -1 \\ \text{output } ''0'' :? \ +1 \end{cases}$$

If the output connection of the shift register is the same with a barker code，then when the input is a barker code, the output of the shift register is "1111111". The detector will send a synchronous impulse.



Distributed frame alignment signal

3. Distributed frame alignment signal

The synchronous code is distributed in the data signal. That means between each n bits, a synchronous bit is inserted.

Design criteria of synchronous code:

1. Easy to detect. For example: “11111111”or ”10101010”

2. Easy to separate synchronous code from data code. For example: In some digital telephone system， all ”0” stands for ring, so synchronous code can only use “10101010”.

Performance

Performance of frame synchronization system

—Bunched frame alignment signal

1. Probability of missing synchronization P_L

Affected by noise, the detector may not be able to detect the synchronous code. The probability of this situation is called probability of missing synchronization P_L.

Assume the length of synchronous code is n， bit error rate is Pe. The detector will not be able to detect if more than m bit errors happen, then:

$$P_L = 1 - \sum_{x=0}^{m} C_n^x P_e^x \left(1 - P_e\right)^{n-x}$$

Performance

2. Probability of false synchronization P_F

Since data code can be arbitrary, it may be the same with synchronous code. The probability of this situation is called probability of false synchronization P_F.

P_F equals to the probability of appearance of synchronous code in the data code.

a. In a binary code, assume 0 and 1 appears with the same probability. There are 2^n combinations of a n bit code.

b. Assume when there are more than m bit errors, the data code will also be detected as synchronous code.

When $m = 0$, only $1(C_n^0)$ code will be detected as synchronous code;

When $m = 1$, there are C_n^1 codes will be detected as synchronous code;

……

Therefore, the probability of false synchronization is:

$$P_F = \frac{\sum_{x=0}^{m} C_n^x}{2^n} = \left(\frac{1}{2}\right)^n \sum_{x=0}^{m} C_n^x$$

Performance

P_L and P_F depends on the length of synchronous code n and the maximum bit error m. When n↑, P_F↓, P_L↑; when m↑, P_L↓, P_F↑

Performance

3. Average build time t_s

Assume both P_L and P_F will not happen, the worst case is we need one frame to build frame synchronization. Assume each frame contains N bits, each bit has a width T_b, then one frame costs NT_b.

Now assume a missing synchronization or a flase synchronization also needs NT_b to rebuild the synchronization, then:

$$t^1{}_s = NT_b\left(1 + P_L + P_F\right)$$

Bisedes, the average build time of using the distributed frame alignment signal is:

$$t^2{}_s = N^2 T_b\left(N >> 1\right)$$

Apparently, $t^1{}_s < t^2{}_s$, so the previous method is more widely used.