Outline

- Introduction
- Signal, random variable, random process and spectra
- Analog modulation
- Analog to digital conversion
- Digital transmission through baseband channels
- Signal space representation
- Optimal receivers
- Digital modulation techniques
- Channel coding
- **Synchronization**
- Information theory

Synchronization

- Synchronization is one of the most critical functions of a communication system with coherent receiver. To some extent, it is the basis of a synchronous communication system.
- Three kinds of synchronization: Carrier synchronization, Symbol/Bit synchronization, and Frame synchronization.
- Carrier synchronization (载波同步): Receiver needs estimate and compensate for frequency and phase differences between a received signal's carrier wave and the receiver's local oscillator for the purpose of coherent demodulation, no matter it is analog or digital communication systems.

Chapter 8.8-8.9

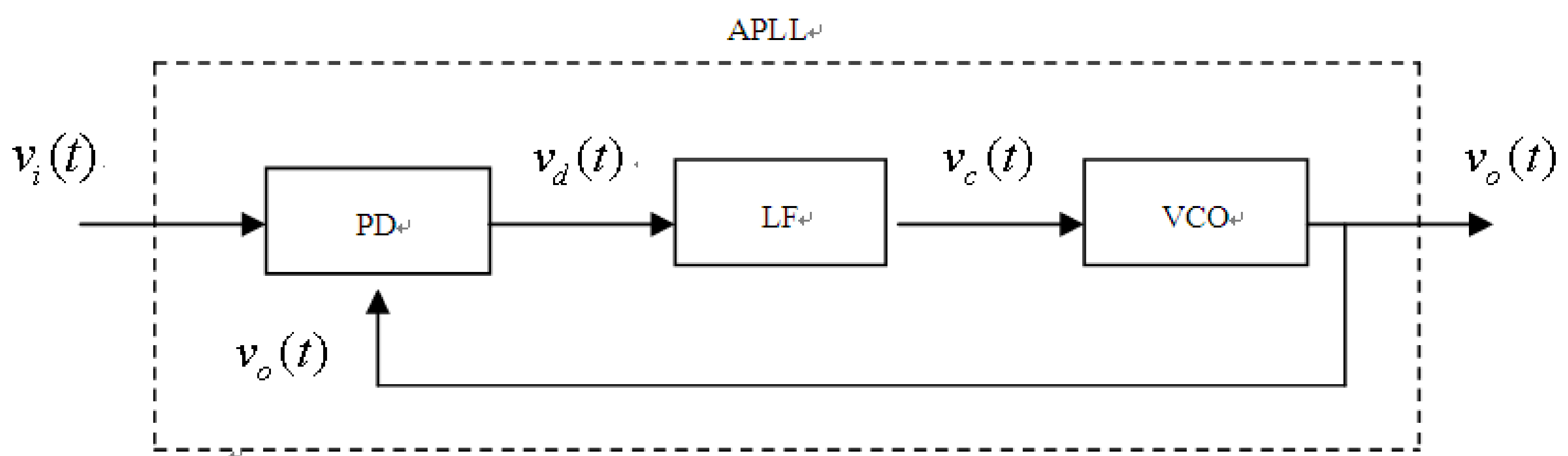
韩声栋等，《通信原理》

Synchronization

- Symbol/Big synchronization (符号/位同步): In digital systems, the output of the receiving filter (i.e. matched filter) must be sampled at the symbol rate and at the precise sampling time instants. Hence, we require a clock signal. The process of extracting such a clock signal at the receiver is called symbol/bit synchronization.
- Frame synchronization (帧同步): In frame-based digital systems, receiver also needs to estimate the starting/stopping time of a data frame. The process of extracting such a clock signal is called frame synchronization.

Synchronization

- Phase-Locked Loop (PLL, 锁相环)
  - ➢ PLL is often used in carrier syn. and symbol syn. It is a closed-loop control system consisting of
  - ➢Phase detector (**PD**): generate the phase difference of $v_i(t)$ and $v_o(t)$.
  - ➢Voltage-controlled oscillator (**VCO**): adjust the oscillator frequency based on this phase difference to eliminate the phase difference. At steady state, the output frequency will be exactly the same with the input



Synchronization

- Phase-Locked Loop (PLL, 锁相环)

$$v_i(t) = v_i \sin[\omega_0 t + \phi(t)]$$

$$v_o(t) = v_o \cos[\omega_0 t + \hat{\phi}(t)]$$

➢A PD contains a multiplier and a low-pass filter. The output of PD is:

$$v_d(t) = \mathrm{K}_d \sin[\phi(t) - \hat{\phi}(t)] = \mathrm{K}_d \sin \phi_e(t)$$

➢LF is also a LPF. The output of the LF is (where F(p) is the transfer function)

$$v_c(t) = F(p) v_d(t)$$

Synchronization

- Phase-Locked Loop (PLL, 锁相环)
  - ➢The output of VCO can be a sinusoid or a periodic impulse train. The differentiation of the output frequency are largely proportional to the input voltage.

$$\frac{d\,\hat{\phi}(t)}{dt} = \mathrm{K}_v v_c(t)$$

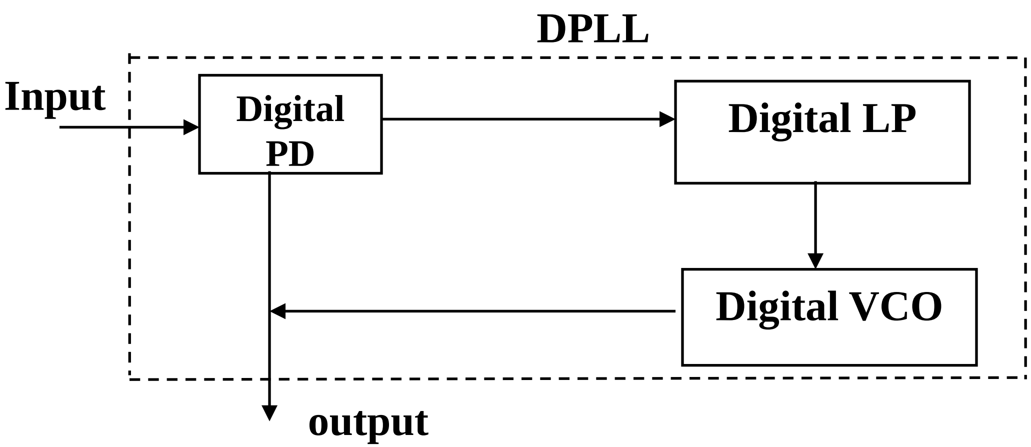
  - ➢If F(p)=1， then

$$\frac{d\,\hat{\phi}(t)}{dt} = \mathrm{K}\sin\phi_e(t)$$

The first kind of loop!

Synchronization

- Phase-Locked Loop (PLL, 锁相环)
  - ➢Digital PLL.



DPLL

Synchronization

- Phase-Locked Loop (PLL, 锁相环)
  - In a coherence system, a PLL is used for:

    1.PLL can track the input frequency and generate the output signal with small phase difference.

    2.PLL has the character of narrowband filtering which can eliminate the noise introduced by modulation and reduce the additive noise.

    3.Memory PLL can sustain the coherence state for enough time.

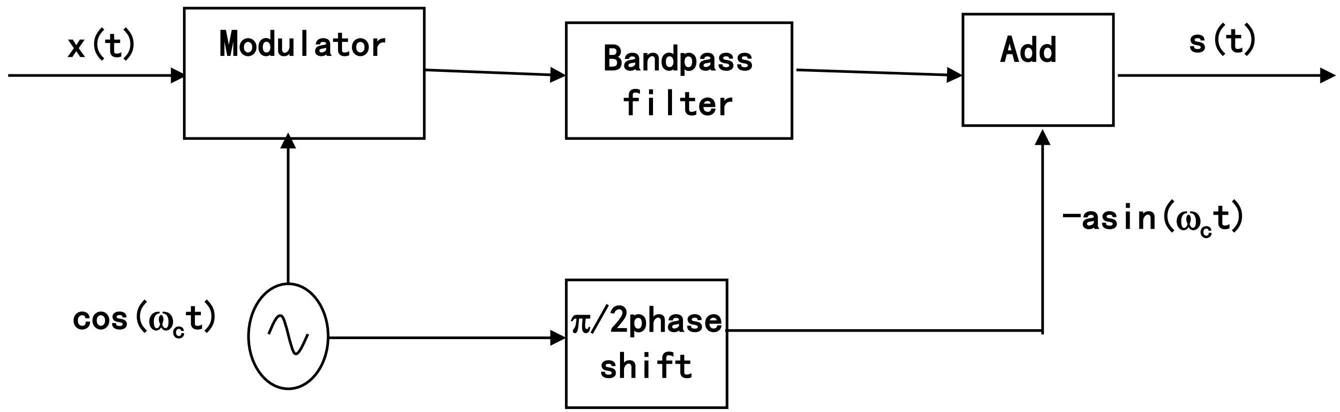
  - CMOS-based integrated PLL has several advantages such as ease of modification, reliable and low power consumption, therefore are widely used in coherence system.

Carrier synchronization

- To extract the carrier:
  - Pilot-tone insertion method: Sending a carrier component at specific spectral-line along with the signal component. Since the inserted carrier component has high frequency stability, it is called pilot (导频).
  - Direct extraction method: Directly extract the synchronization information from the received signal component.

Carrier synchronization

- Pilot-tone insertion method:
  - ➢Insert pilot to the modulated signal

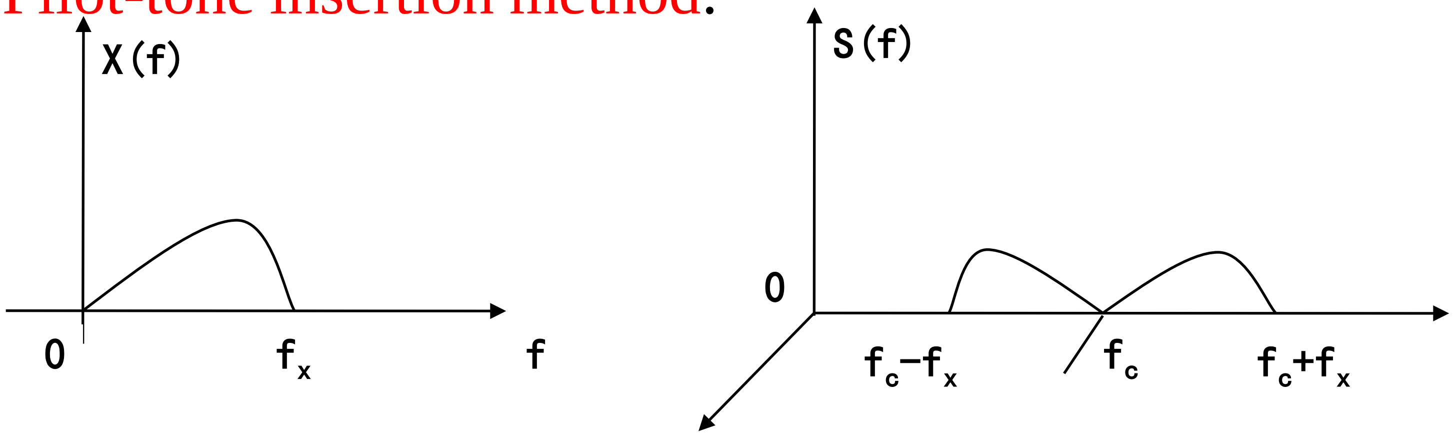


➢The pilot signal is generated by shift the carrier by 90^0 and decrease by several dB, then add to the modulated signal. Assume the modulated signal has 0 DC component, then the pilot is

$$s(t) = f(t)\cos\omega_c t - a\sin\omega_c t$$

Carrier synchronization

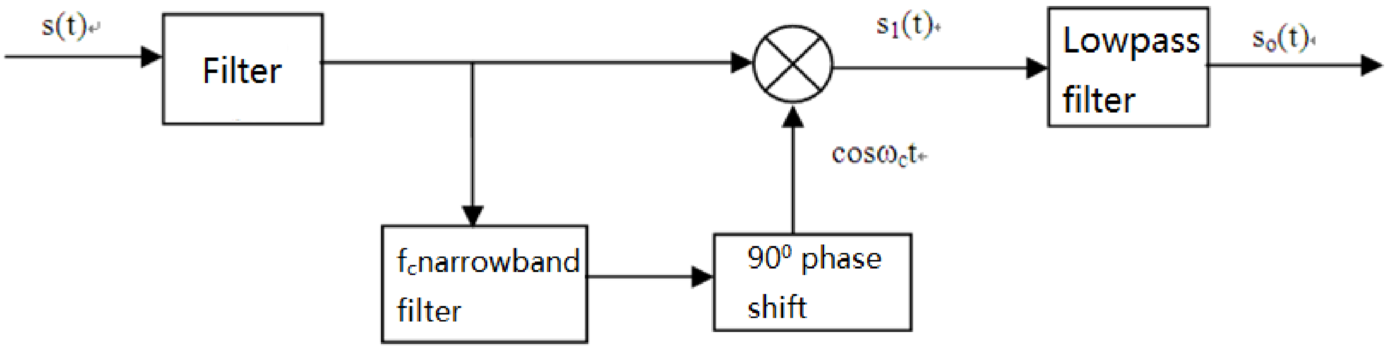
- Pilot-tone insertion method:



  - The receiver uses a narrowband filter with central frequency f_c to extract the pilot $a\sin\omega_c t$ and then the carrier $a\cos\omega_c t$ can be generated by simply shifting 90^0.

Carrier synchronization

- Pilot-tone insertion method:
  - Narrowband filter receiver structure



$$s_1(t) = s(t)\cdot\cos\omega_c t = f(t)\cos^2\omega_c t - a\sin\omega_c t\cos\omega_c t$$

$$= \frac{1}{2}f(t) + \frac{1}{2}f(t)\cos 2\omega_c t - \frac{1}{2}a\sin 2\omega_c t$$

$$\textit{After the LPF}\ \ s_0(t) = \frac{1}{2}f(t)$$

  - DSB, SSB and PSK are all capable of pilot-tone insertion method. VSB can also apply pilot-tone insertion method but with certain modification.

Carrier synchronization

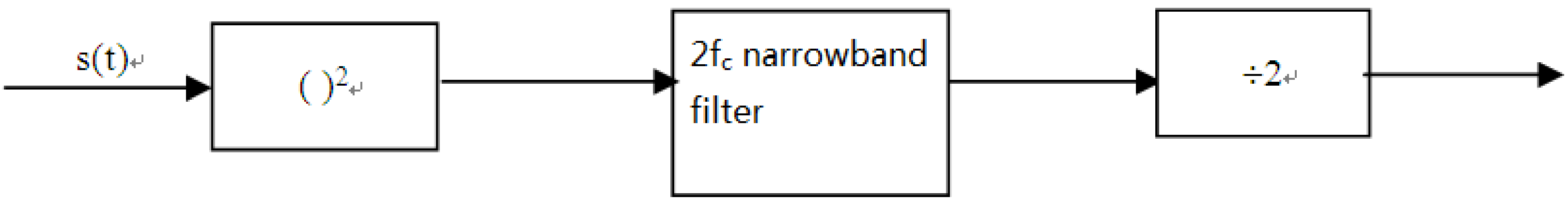
- Pilot-tone insertion method:
  - The drawback of narrowband filter receiver includes:
    1.The pass band is not narrow enough
    2.f_c is fixed, cannot tolerate any frequency drift with respect to the central frequency
    3.Can be replaced by PLL
  - Pilot-tone insertion method is suitable for DSB, SSB, VSB and 2PSK

Carrier synchronization

- Direct extraction method:
  - ➢If the spectrum of the received signal already contains carrier component, then the carrier component can be extracted simply by a narrowband filter or a PLL
  - ➢If the modulated signal suppresses the carrier component, then the carrier component may be extracted by performing nonlinear transformation or using a PLL with specific design.

Carrier synchronization

- Nonlinear transformation based method:
  - ➢Square transformation



Example： a DSB signal $s(t)= f(t)\cos\omega_c t$

If $f(t)$ has 0 DC component， then $s(t)$ *does not have carrier component*

square transformation: $s^2(t)=\frac{1}{2}f^2(t)+\frac{1}{2}f^2(t)\cos 2\omega_c t$

now $f^2(t)$ contains DC component, let it be α so: $f^2(t)=\alpha+f_m(t)$

then $s^2(t)=\frac{1}{2}\alpha+\frac{1}{2}f_m(t)+\frac{1}{2}\alpha\cos 2\omega_c t+\frac{1}{2}f_m(t)\cos 2\omega_c t$

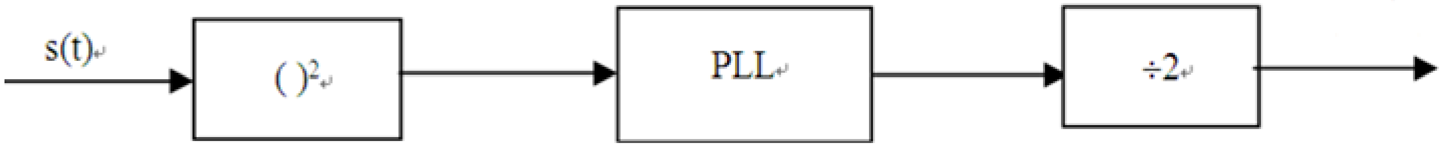
Carrier synchronization

- Nonlinear transformation based method:
  - ➢Square transformation

$$s^2(t)=\frac{1}{2}\alpha+\frac{1}{2}f_m(t)+\frac{1}{2}\alpha\cos 2\omega_c t+\frac{1}{2}f_m(t)\cos 2\omega_c t$$

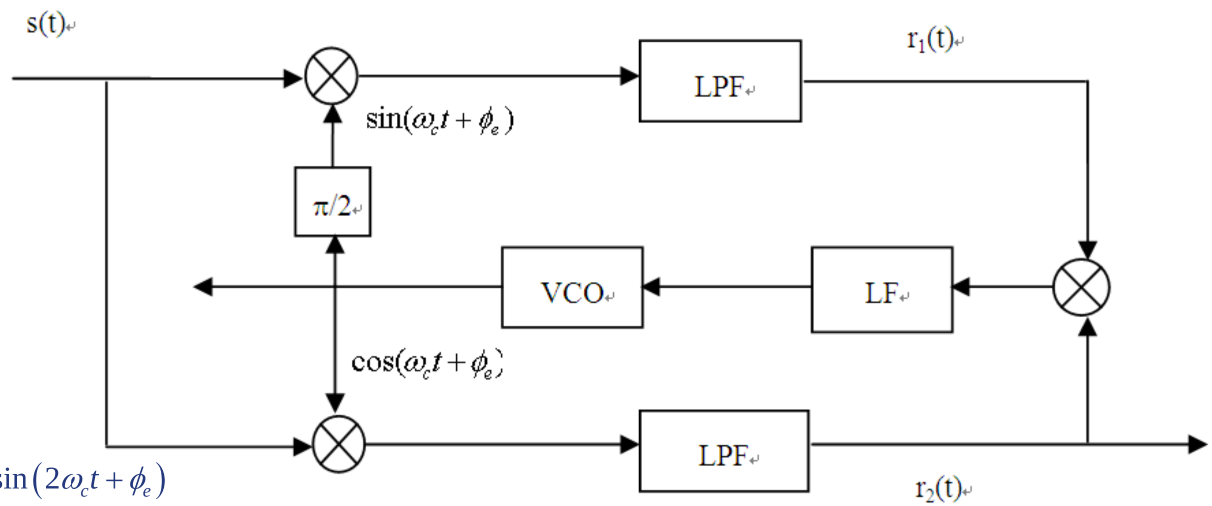
The first term is the DC component. The second term is the low frequency component. The third term is the $2\omega_c$ component. The 4th term is the frequency component symmetrical distributed of $2\omega_c$—modulation noise. After narrowband filtering, only the 3rd term and a small fraction of 4th term left, then the carrier component can be extracted by frequency division.

Since the carrier is extracted by frequency division, its phase may shift by 180°. Besides, modulation noise may cause random phase jitter.



Carrier synchronization

- Nonlinear transformation based method:
  - In-phase orthogonal loop (**Costas loop**)



Let $s(t) = f(t)\cos\omega_c t$

(1) upper branch

$$f(t)\cos\omega_c t \cdot \sin(\omega_c t + \phi_e) = \frac{1}{2} f(t)\sin\phi_e + \frac{1}{2} f(t)\sin(2\omega_c t + \phi_e)$$

ϕ_e is the phase difference between generated carrier and the original carrier

After LPF $r_1(t) = \frac{1}{2} f(t)\sin\phi_e$

When ϕ_e is small, $r_1(t) = \frac{1}{2} f(t)\phi_e$

(2) lower branch

$$r_2(t) = \frac{1}{2} f(t)\cos\phi_e \to \frac{1}{2} f(t)$$

$$(3) r_1(t) \cdot r_2(t) \to \frac{1}{4} f^2(t)\phi_e = v_d(t)$$

Contains in-phase branch and orthogonal branch. All parts except LF and VCO are similar with a "phase detector".

Carrier synchronization

- Nonlinear transformation based method:
  - In-phase orthogonal loop (Costas loop)

    Advantages of Costas loop:

    1.Costas loop works on f_c instead of $2f_c$, so when f_c is large Costas loop is easier to realize

    2.The output of in-phase loop $\mathrm{r}_2(t)$ is the signal $\mathrm{f}(t)$

- Performance of carrier synchronization technique

  1) Phase error: steady-state phase error, random phase error

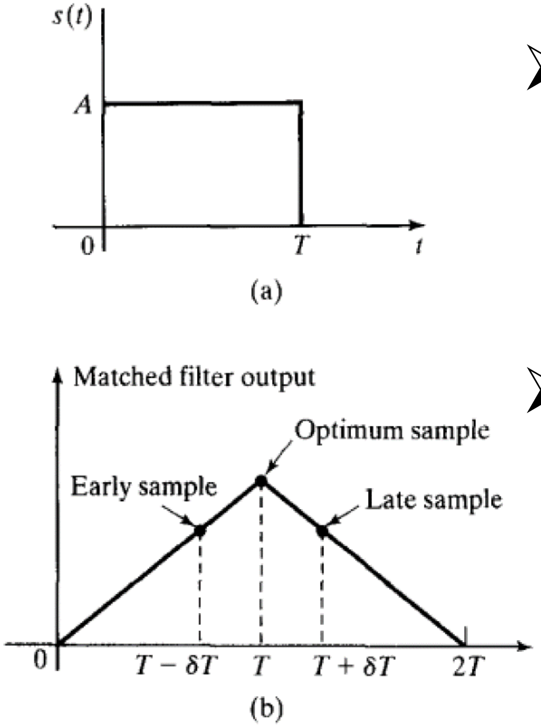
  2) Synchronization build time and hold time

Symbol synchronization

- In a digital communication system, the output of the receiving filter must be sampled periodically at the symbol rate and at the precise sampling time instance.
- To perform this periodic sampling, we need a clock signal at the receiver
- The process of extracting such a clock signal is called symbol synchronization or timing recovery
- One method is for the transmitter to simultaneously transmit the clock frequency along with the information signal. The receive can simply employ a narrowband filter or PLL to extract it. This method requires extra power and bandwidth and hence, but frequently used in telephone transmission systems.
- Another method is to extract the clock signal from the received data signal by using some kind of non-linear transformation.

Symbol synchronization

- 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.

Symbol synchronization

- Early-late gate synchronization
  - Block diagram.

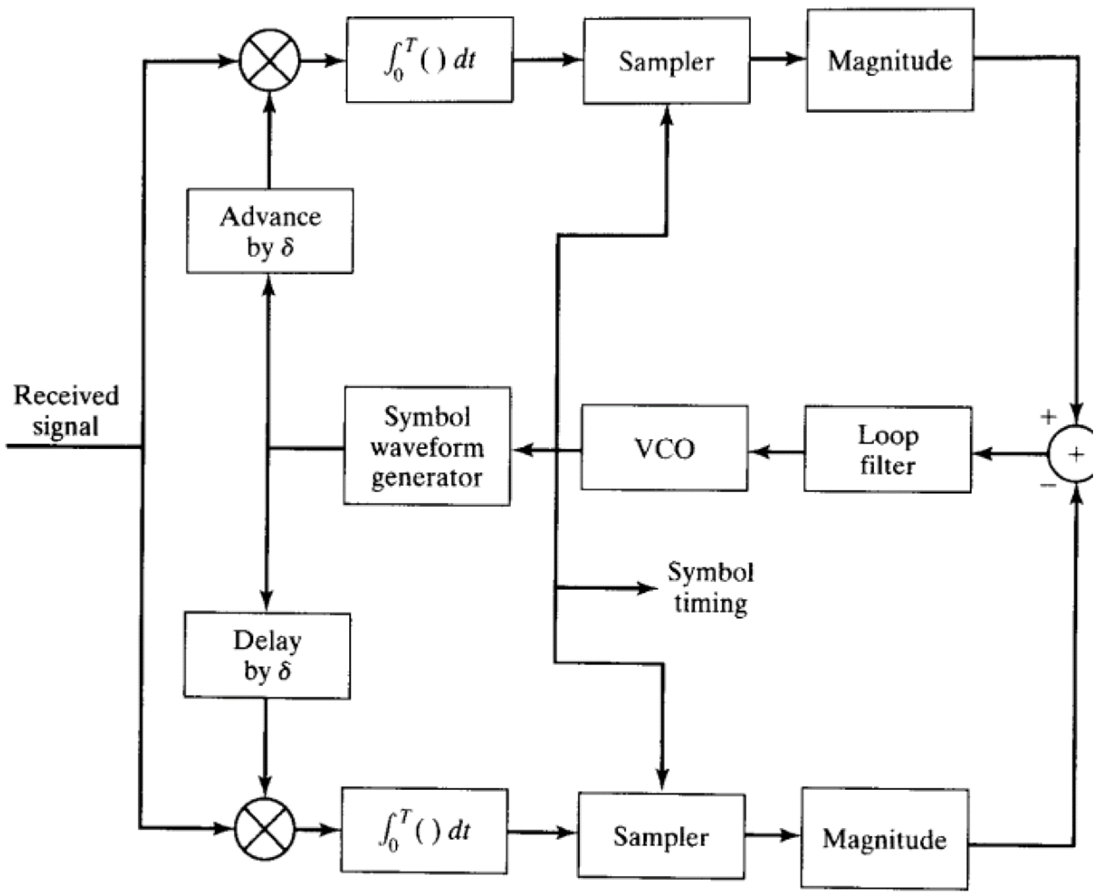
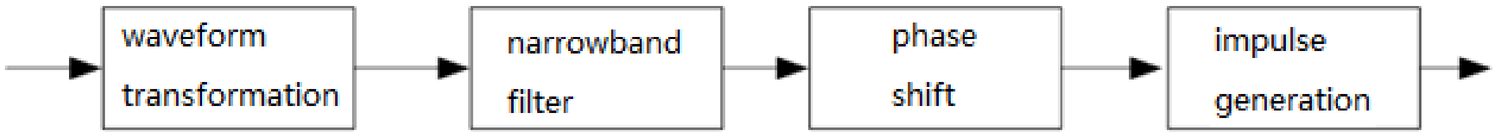


Figure 8.49 Block diagram of early–late gate synchronizer.

Symbol synchronization

- Nonlinear transformation based synchronization

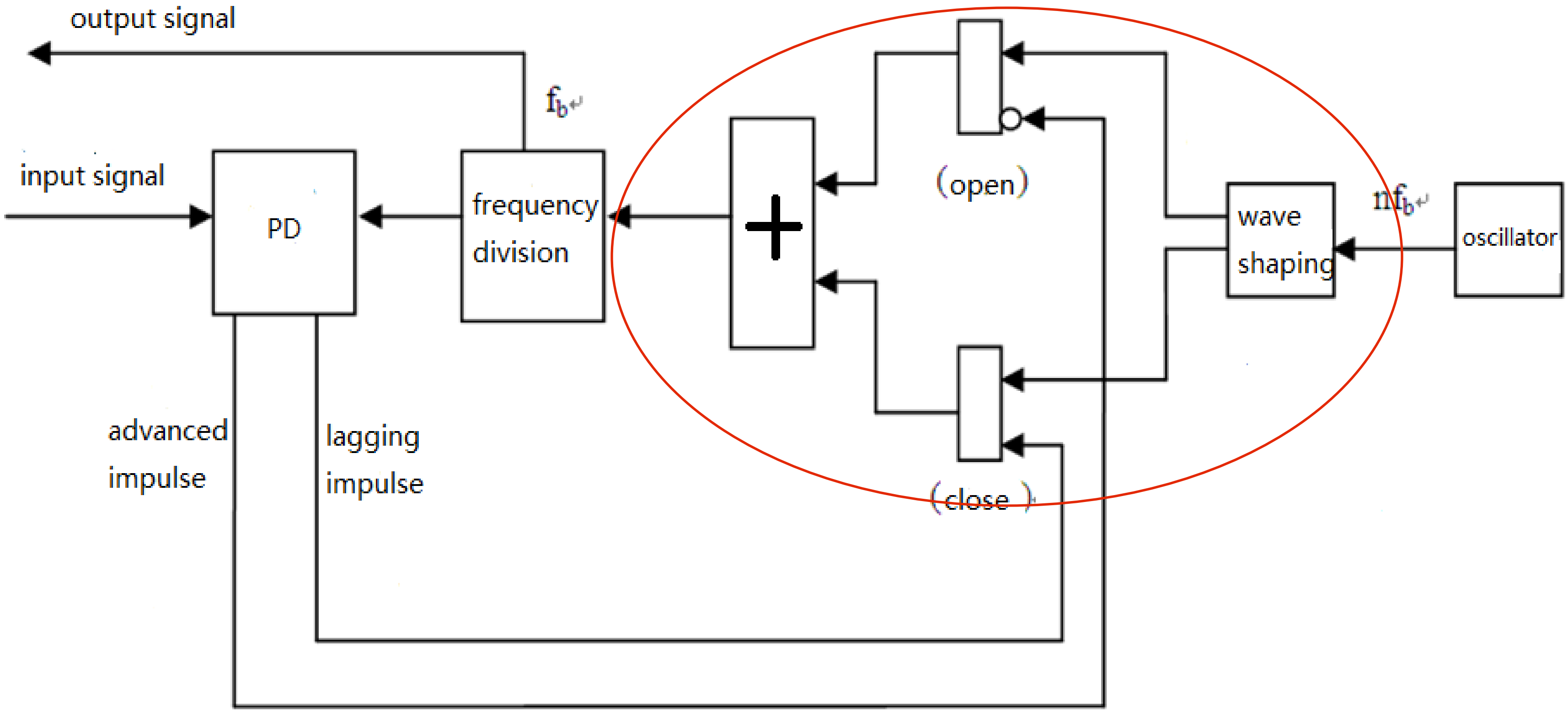


  - Some transformations can add synchronous signal with f=1/T to the original signal. For example, we can transform the signal to return-to-zero waveform. After narrowband filtering and phase shifting, we can generate the clock signal used for synchronization.

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

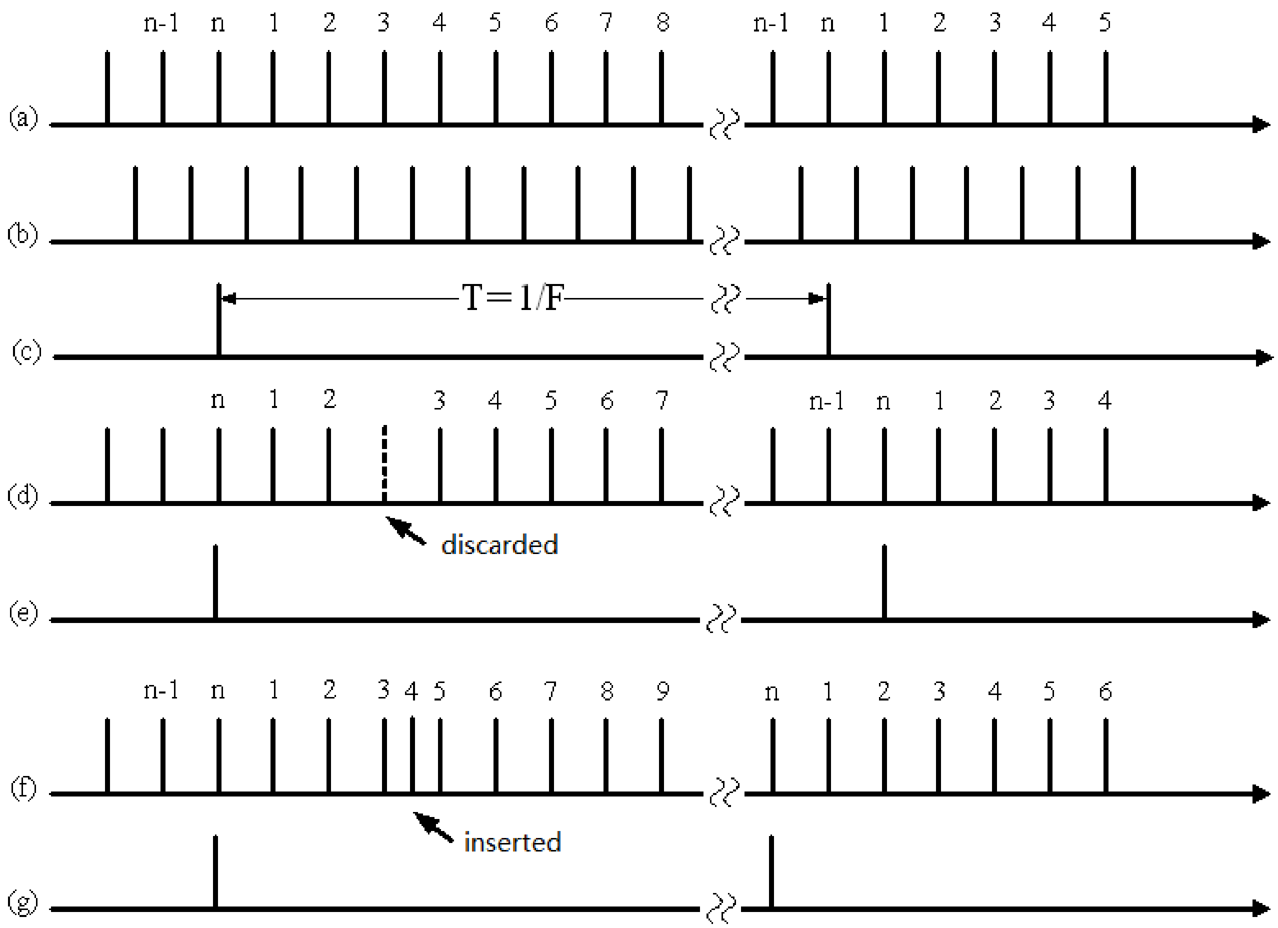
Symbol synchronization

- DPLL



Symbol synchronization

- DPLL



Symbol synchronization

- DPLL
  - Performance.

    1). Phase error

    2). Synchronization build time

    3). Synchronization hold time

    4). Synchronous bandwidth

Frame synchronization

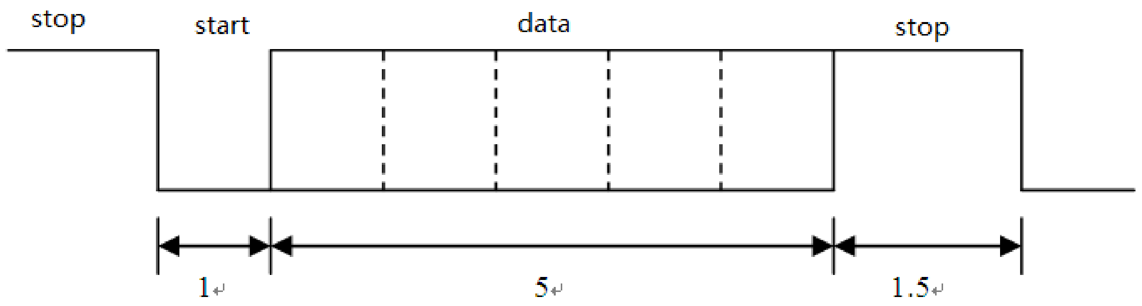
- Recall that carrier and symbol synchronization needs to estimate the phase of synchronous signal which can be realized by using a PLL.
- Frame synchronization is to insert frame alignment signal (distinctive bit sequence) and then detect the alignment symbol.
- Besides adding frame alignment bits, 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

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

Frame synchronization

- 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.

Low transmission efficiency and low timing accuracy

Frame synchronization

- 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 synchronous code includes
  1. Barker code
  2. Optimal synchronous code
  3. Pseudo-random code

Frame synchronization

- Bunched frame alignment signal
  - Barker code

(1) Barker code:

A n bits barker code $\{x_1, x_2, x_3 \cdots x_n\}$, x_i=+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 $n = 2,3,4,5,7,11,13$.

Frame synchronization

- Bunched frame alignment signal
  - Barker code

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

Frame synchronization

- Bunched frame alignment signal
  - Barker code

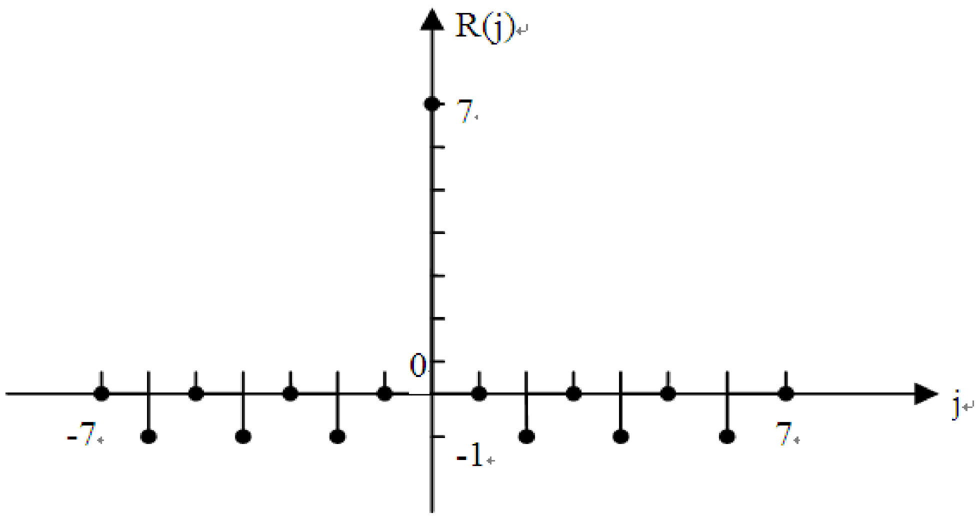
Example: A barker code with $n = 7$ find its self-correlation function

$$j = 0: R_x(0) = x_1x_1 + x_2x_2 + \cdots + x_7x_7 = 7$$

$$j = 1: R_x(1) = x_1x_2 + x_2x_3 + \cdots = 0$$

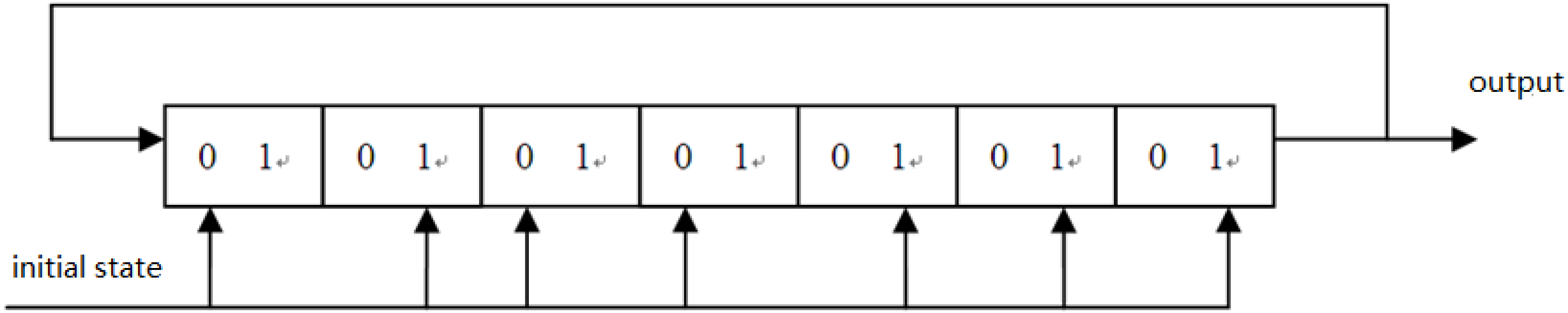
Similarly, we can determine $R_x(j)$.

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



Frame synchronization

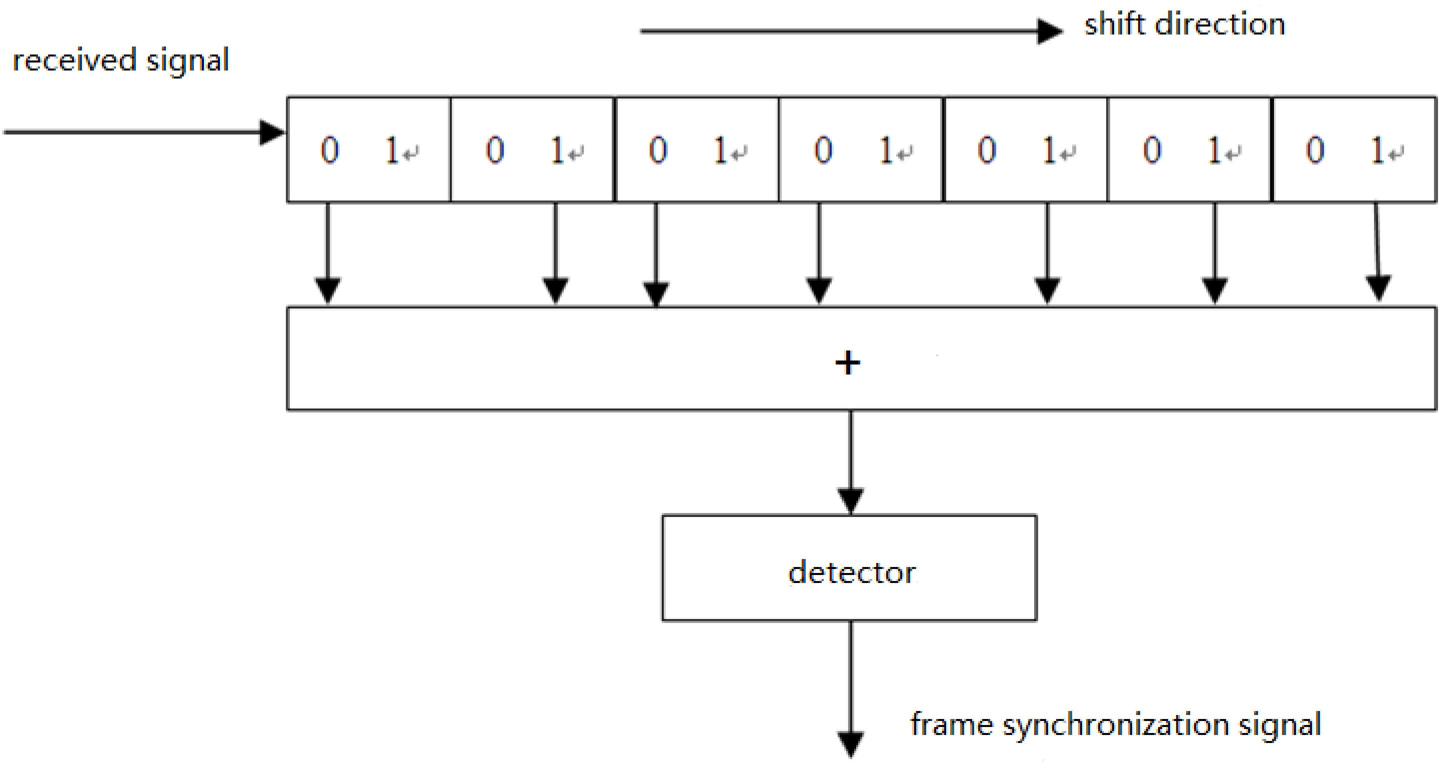
- Bunched frame alignment signal
  - Barker code generator – shift register
  - Example: when n=7, a 7 bits shift register. The initial state is a barker code.



Frame synchronization

- Bunched frame alignment signal

➢ Barker code detector



Frame synchronization

- Bunched frame alignment signal

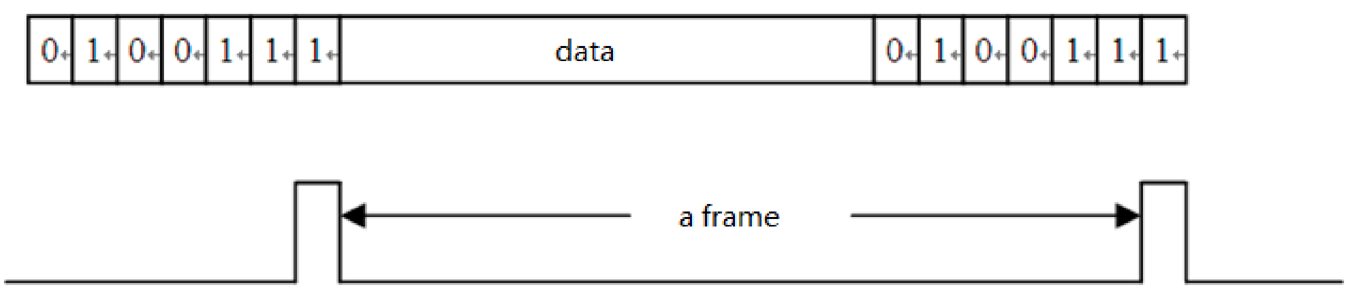
➢ Barker code detector

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.



Frame synchronization

- 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”

Frame synchronization

- Performance of Bunched frame alignment signal

➢ Probability of missing synchronization P_L

1. 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.

2. 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:

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

Frame synchronization

- Performance of Bunched frame alignment signal

➢ 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.

Frame synchronization

- Performance of Bunched frame alignment signal

➢ Probability of false synchronization PF

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:

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

Frame synchronization

- Performance of Bunched frame alignment signal

➢ 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↑

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:

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

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

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

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