Chapter 9: Synchronization

Shanghai Jiao Tong University 11.2011

Synchronization

Synchronization is one of the most critical functions performed at the receiver of a synchronous communication system. To some extent, it is the basis of a synchronous communication system.

- Carrier synchronization
- Symbol/Bit synchronization
- Frame synchronization

Synchronization

- **Carrier synchronization**

  To recover the signal without distortion, receiver needs to 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.

As to digital communication system, symbol/bit synchronization and frame synchronization are also required.

Synchronization

Symbol/bit synchronization

The output of the receiving 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

Receiver can proceed by every group of symbols instead of every single symbol, such as a frame in TDM system. Similar with symbol/bit synchronization, the process of extracting such a clock signal is called frame synchronization.

Synchronization

The synchronized system required:

- Synchronous signal must have high noise immunity and be reliable.
- Generating synchronous signal should not consume much extra power and increase implementation complexity.
- Synchronous signal should possess few channel resource.

Carrier Synchronization

To extract the carrier:

1. 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.

2. Direct extraction method

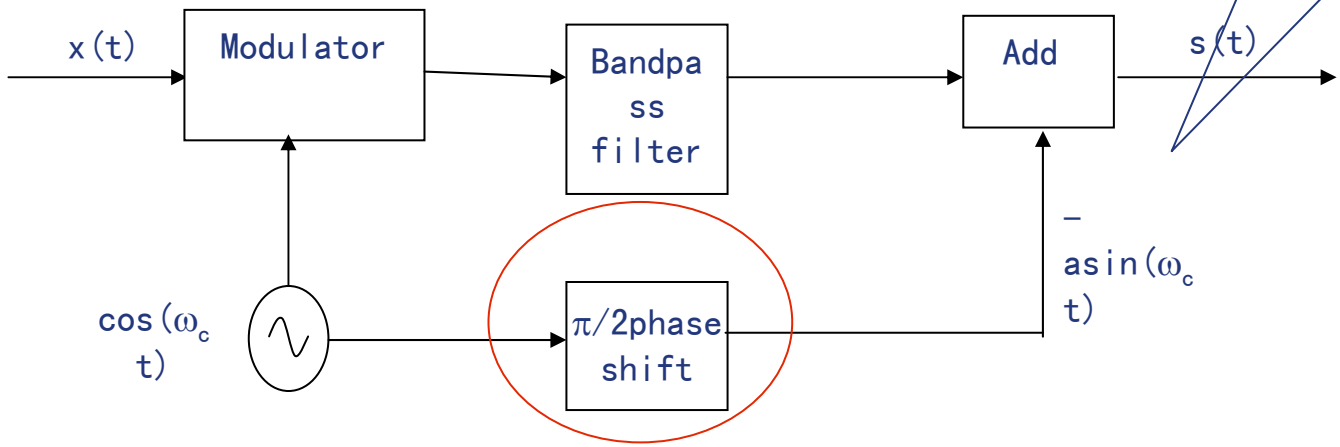
Directly extract the synchronization information from the received signal component.

Pilot-tone insertion method

1. Pilot-tone insertion method
—insert pilot to the modulated signal

Thinking:
Why 90^0 shift?

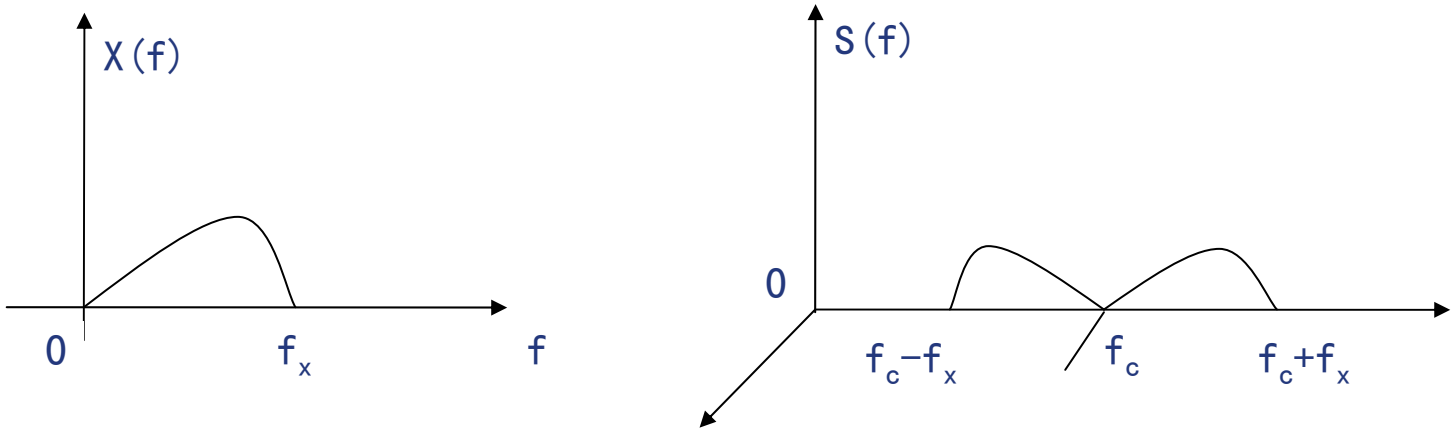
1） Narrowband filter



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

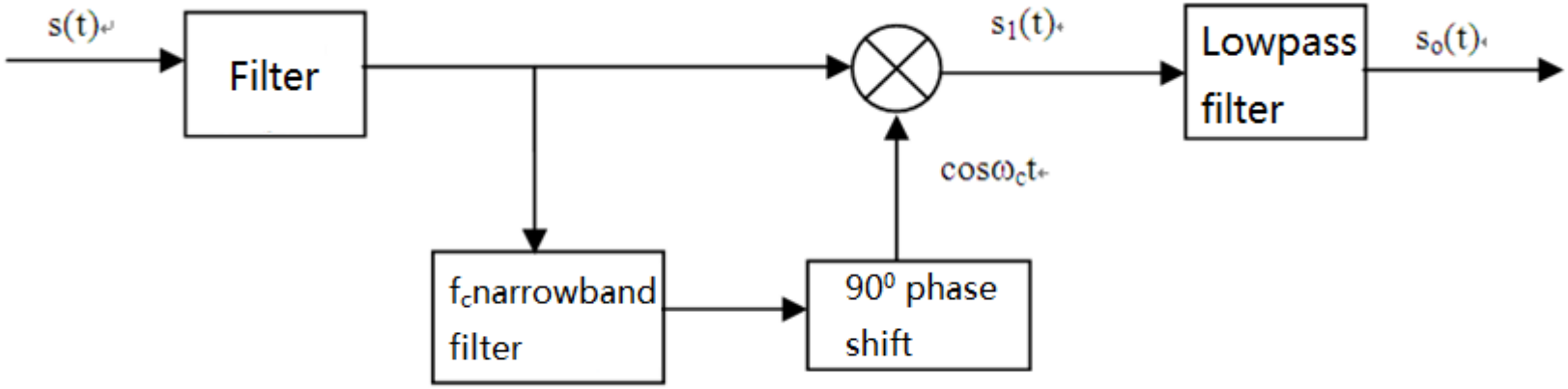
$$s(t) = f(t)\cos\omega_c t - a\sin\omega_c t$$

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.

Pilot-tone insertion method



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

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

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 it is more complex.

Narrowband Filter

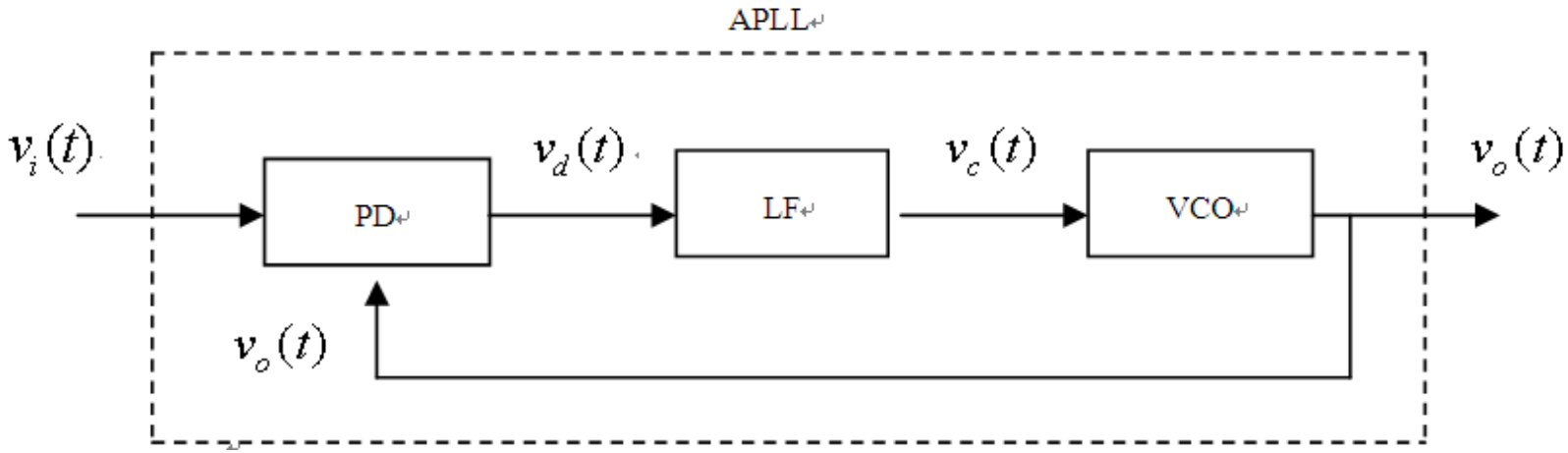
The drawback of narrowband filter:

- The pass band is not narrow enough
- f_c is fixed, do not tolerate any frequency drift with respect to the central frequency

Phase-locked Loop

2）Phase-locked loop（PLL）

PLL is a feedback loop with a voltage-controlled oscillator (VCO), a phase detector (PD)and a loop filter (LF). The PD will generate the phase difference of $v_i(t)$ and $v_o(t)$. The VCO will 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 frequency.



Phase-locked Loop

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

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

A PD contains a multiplexer and a lowpass filter. The LPF is for filtering the extra frequency component generated by multiplexing. The output voltage is:

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

Loop filter is also a LPF. Active/passive PI filter are most commonly used.

The output of the LF is:

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

$$p = \frac{d}{dt}$$

Phase-locked Loop

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

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

This kind of loop is called 1-level loop

Phase-locked Loop

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.

Direct extraction method

2. Direct extraction method

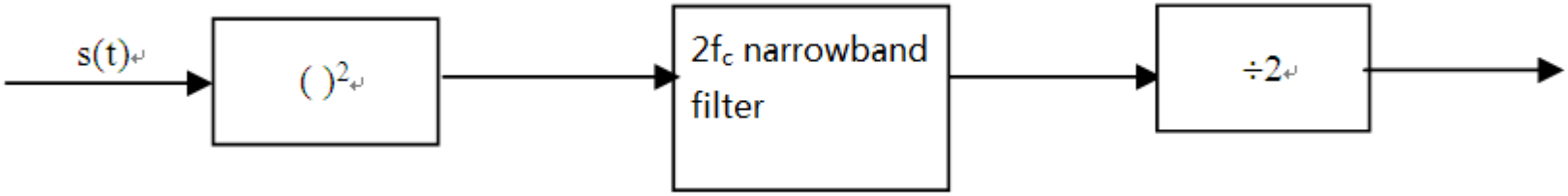
1）. If the spectrum of the received signal already contains carrier component, then the carrier component can be extracted simply by a narrowband filter or a PLL.

2）. If the modulated signal already eliminates the carrier component, then the carrier component can be extracted by performing nonlinear transformation or using a PLL with specific design.

Nonlinear-transformation-based method

1.Nonlinear transformation

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

Nonlinear-transformation-based method

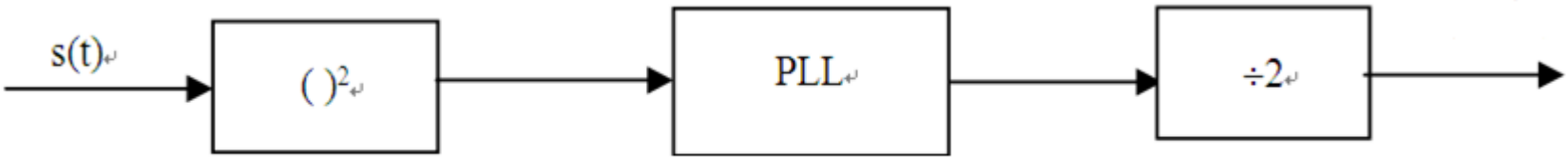
$$s^2(t)=\frac{1}{2}\alpha+\frac{1}{2}f_m(t)+\frac{1}{2}\alpha\cos 2\omega_c t+\frac{1}{2}f_m(t)\cos 2\omega_c t$$

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.

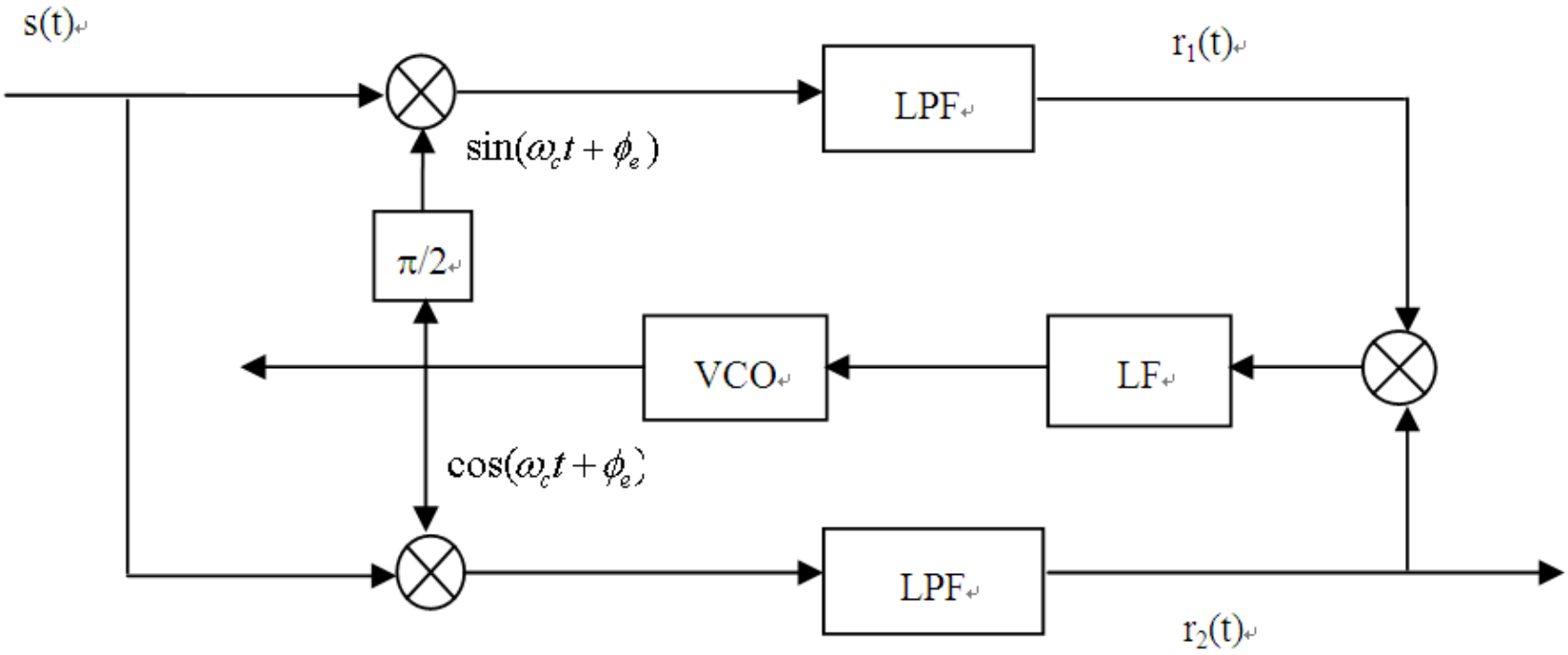
Since the carrier is extracted by frequency division, its phase may shift by 180°. Besides, modulation noise may cause random phase jitter.

Nonlinear-transformation-based method

- Square PLL



2. In-phase orthogonal loop —Costas Loop



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

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

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 $r_2(t)$ is the signal $f(t)$

3. Performance of carrier synchronization technique

1) Phase error: steady-state phase error, random phase error
2) Synchronization build time and hold time

Performance

1. steady-state phase error ϕ_e

(1) Narrowband filter is a simple single tuned loop with a fixed Q value. When the central frequency ω_0 is not equal to the carrier frequency ω_c, steady-state phase error $\Delta\phi$ is arose.

Let $\Delta\omega = |\omega_c - \omega_0|$

When $\Delta\omega$ is small

$$\Delta\phi = 2Q\frac{\Delta\omega}{\omega_0},\quad Q\downarrow,\quad \Delta\phi\downarrow$$

(2)When PLL is applied: $\Delta\phi = \phi_e = \frac{\Delta\omega}{k_0}$

k_0 is the DC gain of the PLL circuit. Apparently, $\Delta\phi$ can have a very small value as long as k_0 is large enough.

Performance

2. Random phase error

Assume Gaussian random noise is the case, we already know when SNR is high (SNR>>1), the phase distribution of the sum of a sinusoid signal and a gaussian noise passed through a narrowband system is a gaussian distribution.

Let $\varphi(0) = 0$, then:

$$f(\phi) = \sqrt{\frac{d}{\pi}} e^{-d\phi^2}$$

$f(\phi)$ with $mean = 0$, $\sigma_\phi^2 = \frac{1}{2d}$, we use σ_ϕ to assess the random phase error.

$$\sigma_\phi = \sqrt{\frac{1}{2d}} \qquad \text{– random phase jitter}$$

Performance

Example: Narrowband filter is a single tuned loop.

The equivalent noise bandwidth: $B_n = \frac{\pi}{2}\frac{f_0}{Q}$

Let the noise's single-sideband PSD is N_0, then the noise power is $P_n = N_0 \cdot B_n$

$$\therefore\ d = \frac{P_S}{P_n} = \frac{P_s}{N_0} \cdot \frac{2Q}{\pi f_0}$$

then $\sigma_\phi = \sqrt{\frac{1}{2d}} = \sqrt{\frac{\pi N_0 f_0}{4QP_S}}$

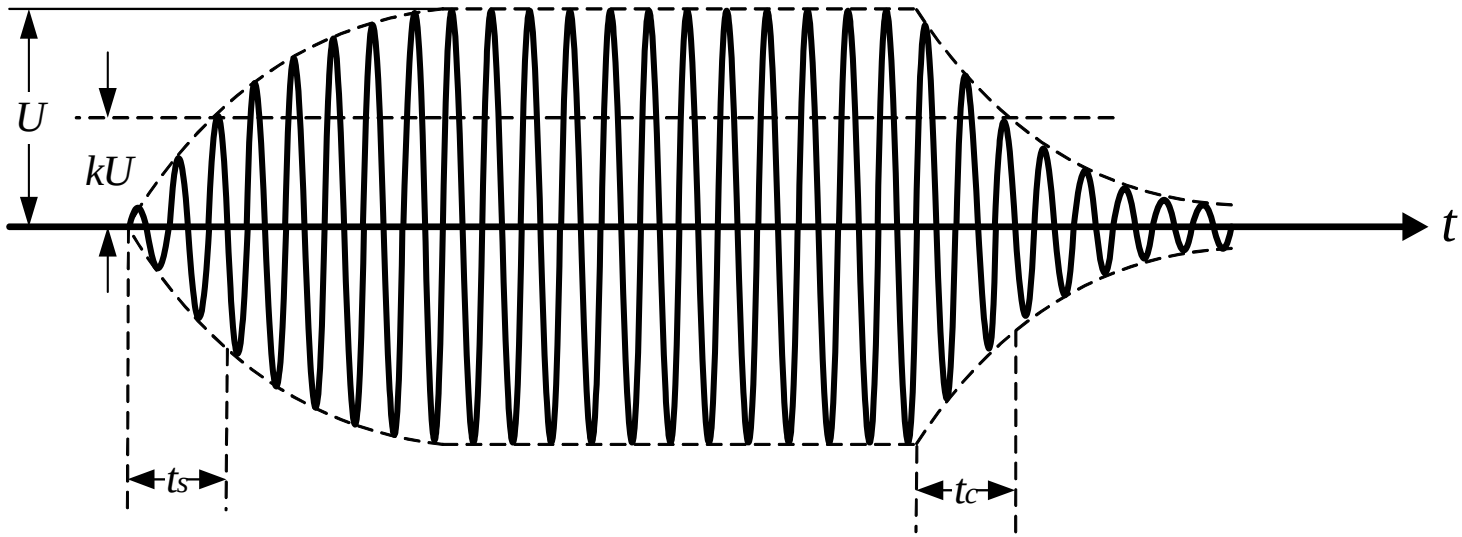
Apparently, Q↑, σ_ϕ↓.

Based on above discussion, we can see there is a trade-off between minimize the steady-state phase error and the random phase error.

Performance

3. Synchronization build time and hold time

Example: using single tuned loop to realize narrowband filtering



Performance

(1) When t = 0, the output voltage is:

$$u(t) = u_0\left(1 - e^{-\alpha t}\right)\cos\omega_0 t$$

$\alpha = \omega_0 / 2Q$ ω_0 is the resonant frequency

When $t = t_S$, synchronization is build when the amplitude goes to $Ku_0\ (0 < K < 1)$

As: $$Ku_0 = u_0\left(1 - e^{-\alpha t_s}\right)$$

So $t_s = \frac{1}{\alpha}\ln\frac{1}{1-k}$ ——synchronization build time

Performance

(2) When synchronization is build, cut off the input signal at $t = 0$, then the output is:

$$u(t) = Ue^{-\alpha t}\cos\omega_0 t$$

When $t = t_c$, synchronization is hold until the amplitude decreases to Ku_0

As $$KU = Ue^{-\alpha t_s}$$

So $t_c = \frac{1}{\alpha}\ln\frac{1}{k}$ ——synchronization hold time

Performance

$$t_s = \frac{1}{\alpha}\ln\frac{1}{1-k} \qquad t_c = \frac{1}{\alpha}\ln\frac{1}{k}$$

$$\alpha = \omega_0 / 2Q$$

In practice, we want $t_s \downarrow$, $t_c \uparrow$. However, when Q $\uparrow$, $\alpha \downarrow$, both t_s and t_c $\uparrow$, vice versa. Therefore, we need consider both factors when designing the parameters.

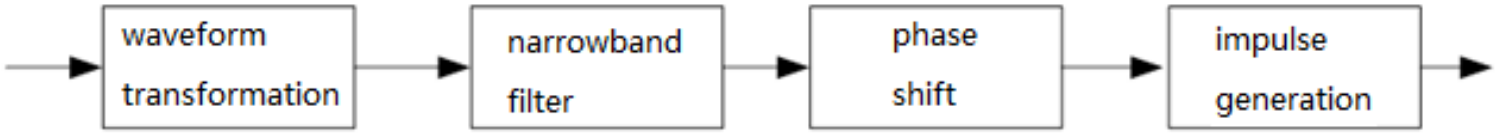
When using PLL, t_s is the lock time, t_c is the hold time. Their values depend on the parameters of circuits. Similarly, there is also a conflict when picking parameters. However, we can change the parameters after lock is build, therefore let $t_s \downarrow$ and $t_c \uparrow$.

Symbol Synchronization

If the baseband signal already contains information for symbol synchronization, then we can extract it by narrowband filter or PLL. When the signal does not contain synchronous signal such as a random polar signal, then we need to insert pilot signal or perform some kind of transformation. Since pilot-tone insertion method is rarely used in practical systems, in this section, we only focus on nonlinear-transformation-based method or using a digital PLL (DPLL)with specific PD.

Nonlinear-transformation-based method

1. Nonlinear-transformation-based method

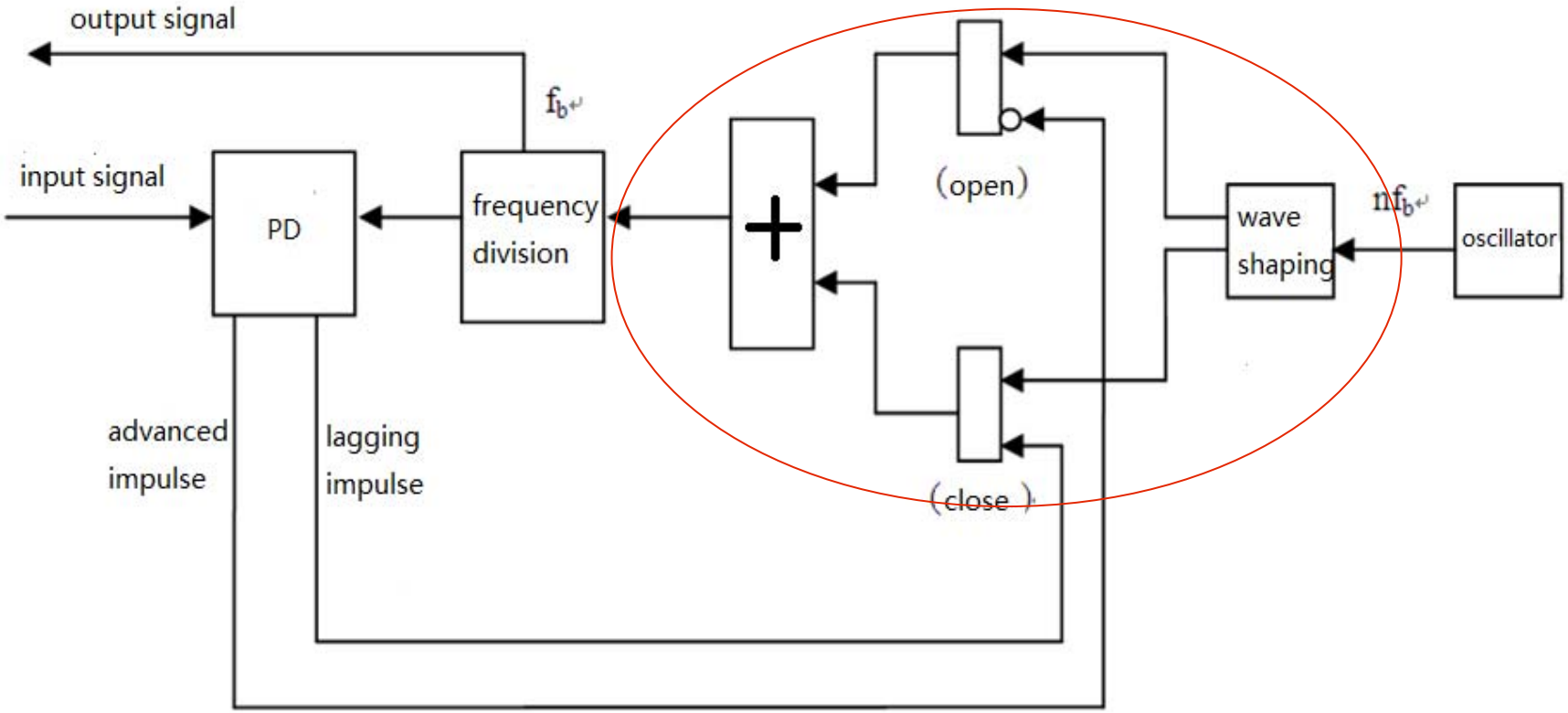


Some transformations can add synchronous signal with f=1/T to the original signal. For example, we can transform the signal to return-to-zero waveform. After narrowband filtering and phase shifting, we can generate the clock signal used for synchronization.

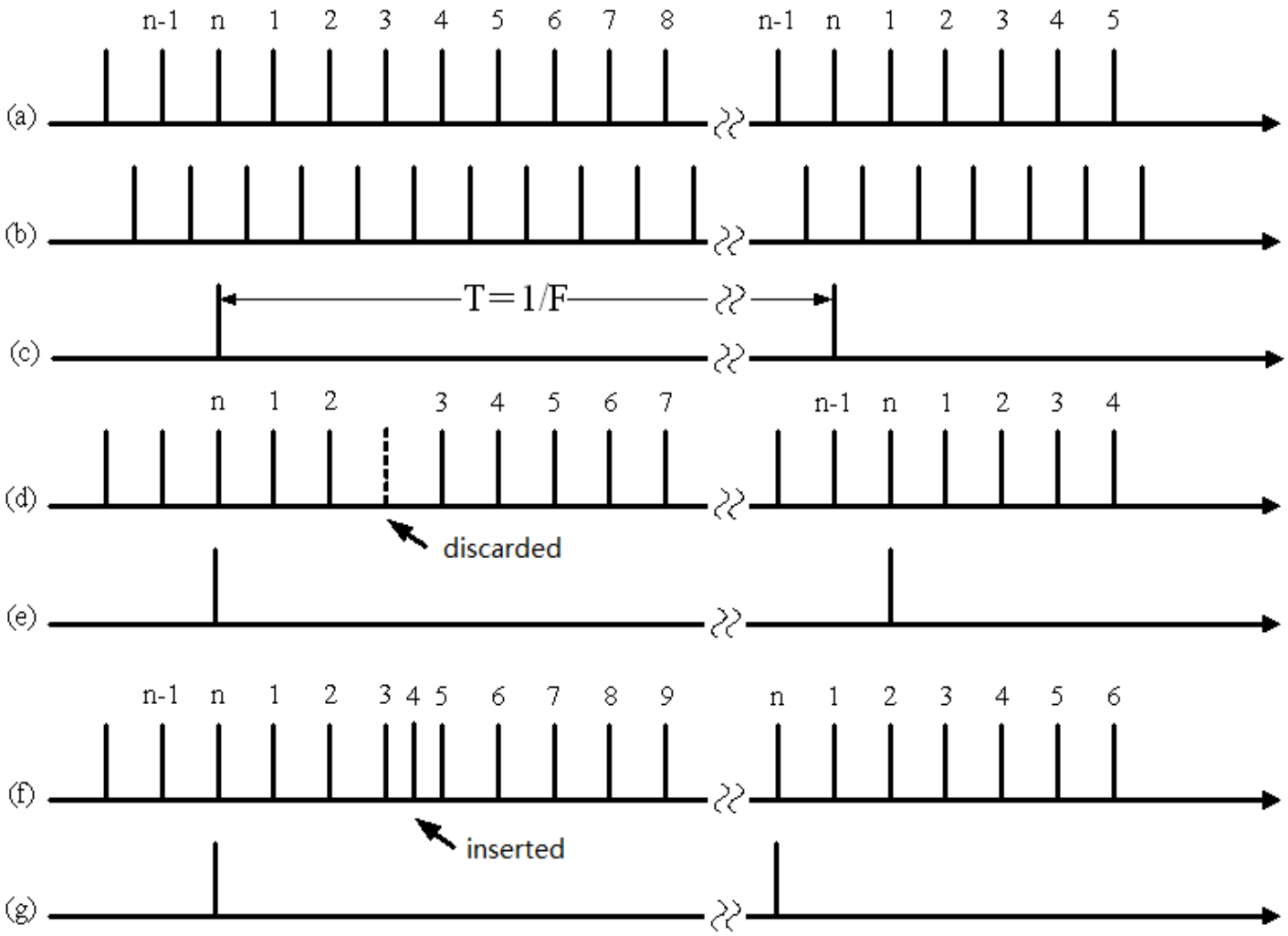
$$P_s(f) = f_s P(1-P)\left|G_1(f) - G_2(f)\right|^2 + f_s^2 \sum_{m=-\infty}^{\infty} \left|PG_1(mf_s) + (1-P)G_2(mf_s)\right|^2 \delta(f - mf_s)$$

Digital PLL (DPLL)

2. DPLL



Digital PLL (DPLL)



Performance

3. Performance of symbol synchronization system

—DPLL

1). Phase error

The phase error occurs because of for a DPLL, the phase cannot change with arbitrary small value, each time the phase can only be modified by 2 π /n（n is the frequency division number）.
Therefore the maximum of phase error is:

$$\varphi_e = 2\pi/n(rad)$$

In time domain, it is:

$$T_e = T_b/n(s)$$

Performance

2. Synchronization build time t_s

Synchronization build time is the maximum period from the system lost synchronization to the system back to synchronization state.

The maximum delay between the synchronous impulse and the received signal is $T_b/2$. Each time the DPLL can only modify the phase by $T_b/2n$, so synchronization is build after N times:

$$N = \frac{T_b/2}{T_b/n} = \frac{n}{2}$$

The PDLL can only modify the phase when the received code crosses zero point. As to random binary code, we can approximately think '01','10','11','00' appears with same probability. Therefore, the code will cross zero point with probability 0.5. Thus, a modification happens each $2T_b$ seconds.

Synchronization build time is:

$$t_s = 2T_b \cdot N = nT_b$$

Performance

3. Synchronization hold time t_c

If the input signal is interrupted after synchronization is build, since there is a frequency difference ΔF between the transmitter and the receiver, the phase of the synchronous signal will keep drifting. We consider the system is not synchronous any more if the phase is drifted by a centain value. This period is called synchronization hold time.

If the transmitter and the receiver have $T_1 = \frac{1}{F_1}$ and $T_2 = \frac{1}{F_2}$, separatively, then

$$|T_1 - T_2| = \left|\frac{1}{F_1} - \frac{1}{F_2}\right| = \frac{|F_2 - F_1|}{F_1 F_2} = \frac{\Delta F}{F_0^2}$$

$F_0 = \sqrt{F_1 F_2}$ and $T_0 = 1/F_0$

It can also be written as: $F_0 |T_1 - T_2| = \frac{\Delta F}{F_0}$

$$\therefore \quad \frac{|T_1 - T_2|}{T_0} = \frac{\Delta F}{F_0}$$

Performance

When frequency difference ΔF exists, the phase will drifted by $|T_1 - T_2|$ after every T_0. If the system can only allow the drift to be as large as T_0/K (K is relevant with Pe), then we can calculate the synchronization hold time t_c as:

$$\frac{T_0/k}{t_c} = \frac{\Delta F}{F_0}$$

$$\therefore \qquad t_c = \frac{1}{\Delta F \cdot k}$$

If t_c fixed, then the system requires ΔF to be:

$$\Delta F = \frac{1}{t_c \cdot k}$$

If the oscillators in transmitter and receiver have the same stability, then the stability should not below:

$$\frac{\Delta F}{2F_0} = \pm\frac{1}{2t_c k F_0}$$

Performance

4. Synchronous bandwidth Δf_s

Synchronous bandwidth is the range of ΔF.

The drift occurs during one symbol interval is:

$$\Delta T = |T_1 - T_2| = \frac{\Delta F}{F_0} T_0 = \frac{\Delta F}{F_0^2}$$

As mentioned, the DPLL modifies the phase every 2 symbols. Each time spends T_0/n seconds. To mentain synchronazation, obviously we need:

$$\Delta T \le \frac{T_0}{2n} = \frac{1}{2nF_0} \qquad \frac{\Delta F}{F_0^2} \le \frac{1}{2nF_0}$$

$$\therefore \qquad |\Delta f_s| = |\Delta F| \le \frac{F_0}{2n}$$

when $F_0 \to f_b$, we have: $|\Delta f_s| \le \frac{f_b}{2n}$

Performance

In DPLL, frequency division number n will affect ϕ_e and Δf_s, so n must be selected properly to let ϕ_e satisfies the system's requirement and let $t_s \downarrow$, $t_c \uparrow$, $\Delta f_s \uparrow$

Frame Synchronization

As mentioned, carrier synchronization and symbol synchronization needs to estimate the phase of synchronous signal which can be realized by using a PLL. Frame synchronization is realized in a different way——inserting frame alignment signal (distinctive bit sequence). Therefore, the basic task of frame synchronization is how to detect the alignment symbol.

Besides add frame alignment bits, some code such as self-synchronizing code can be synchronized without add extra bits. In this section, we only focus on the first method ——inserting frame alignment signal.

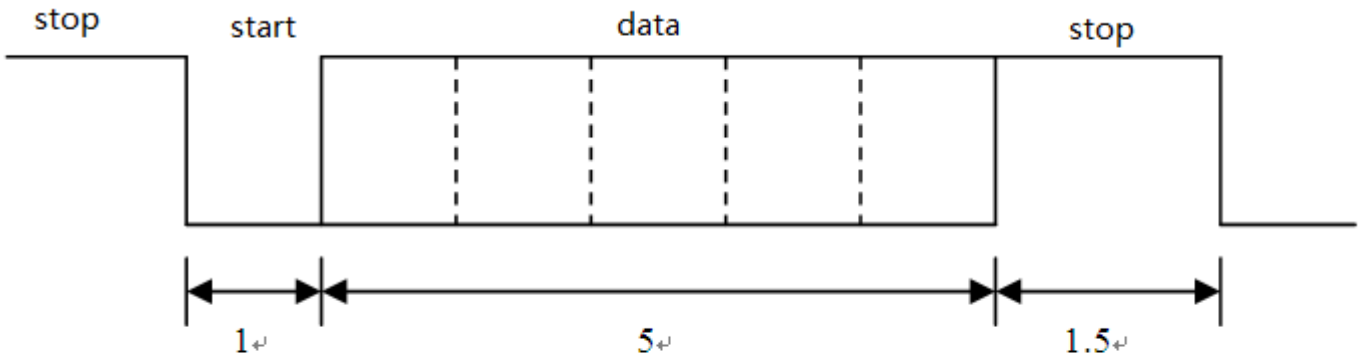
Frame Synchronization

- 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 transport efficiency

2). Low precision of timing

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 and pseudo-random code.

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 $n = 2, 3, 4, 5, 7, 11, 13$.

Barker Code

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

Barker Code

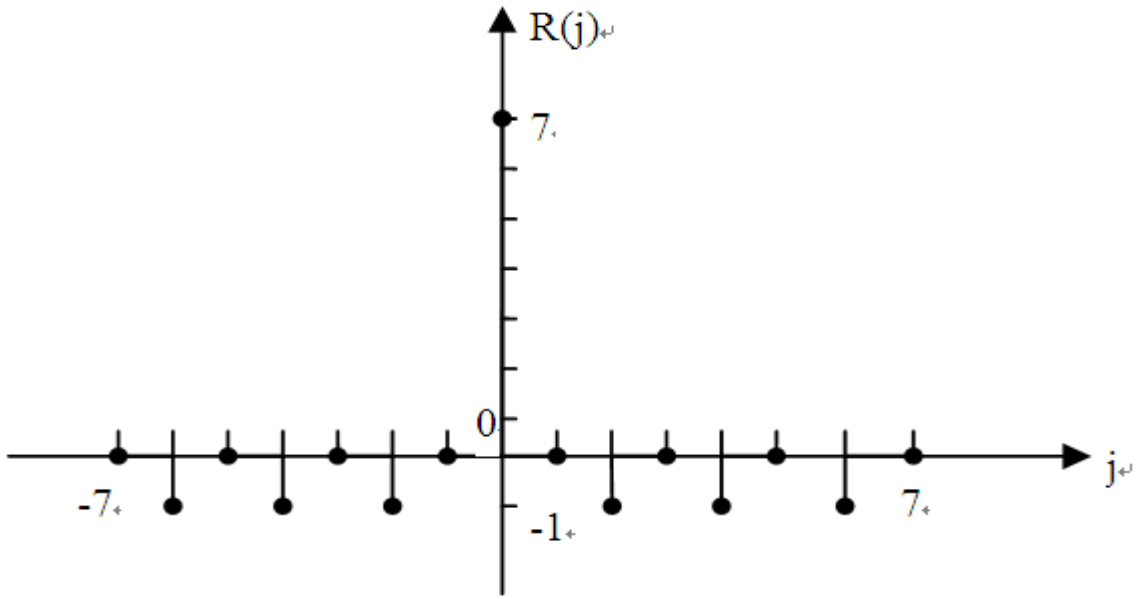
Example: A barker code with $n = 7$, find its self-correlation function

$$j = 0 : R_x(0) = x_1 x_1 + x_2 x_2 + \cdots + x_7 x_7 = 7$$

$$j = 1 : R_x(1) = x_1 x_2 + x_2 x_3 + \cdots = 0$$

Similarly, we can determine $R_x(\mathrm{j})$.

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

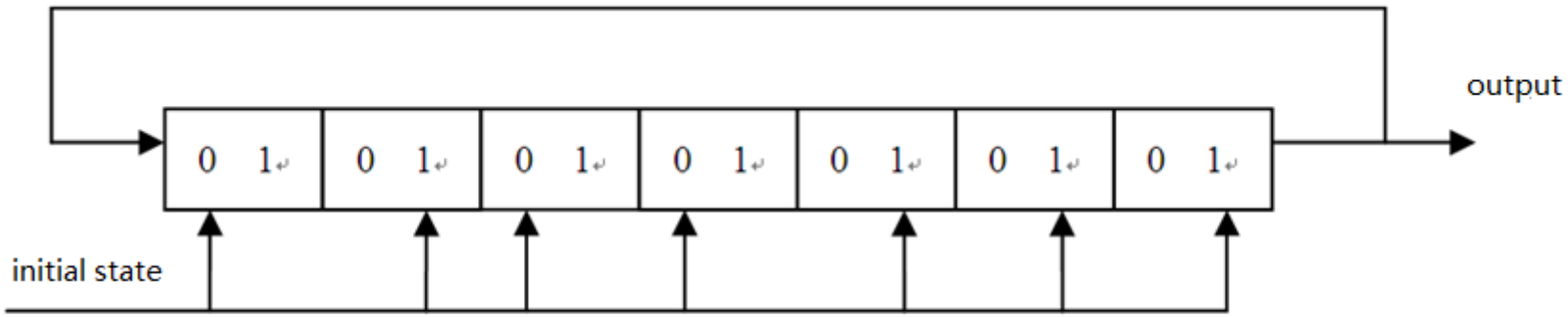


Barker Code

(2) Barker code generator

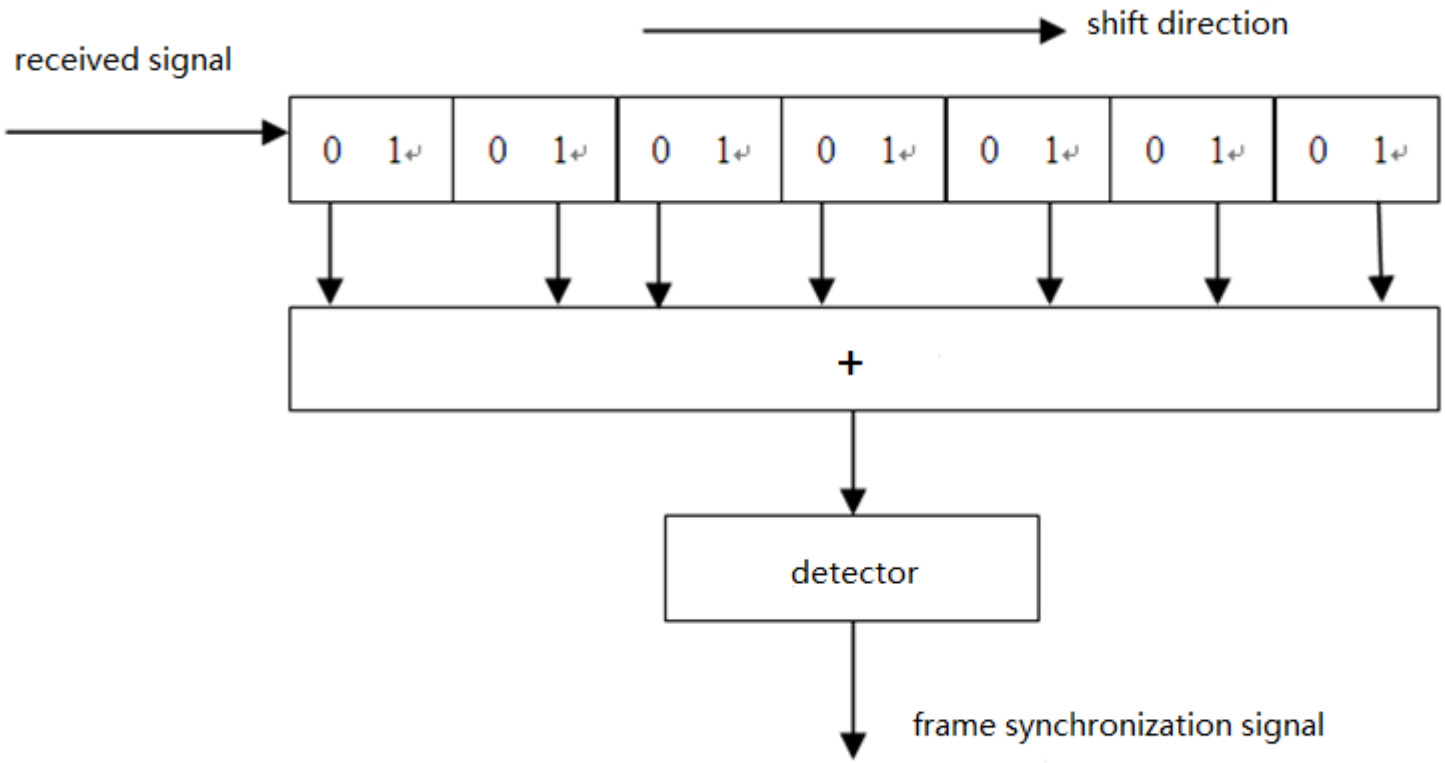
—shift register

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



Barker Code

(3) Barker code detector



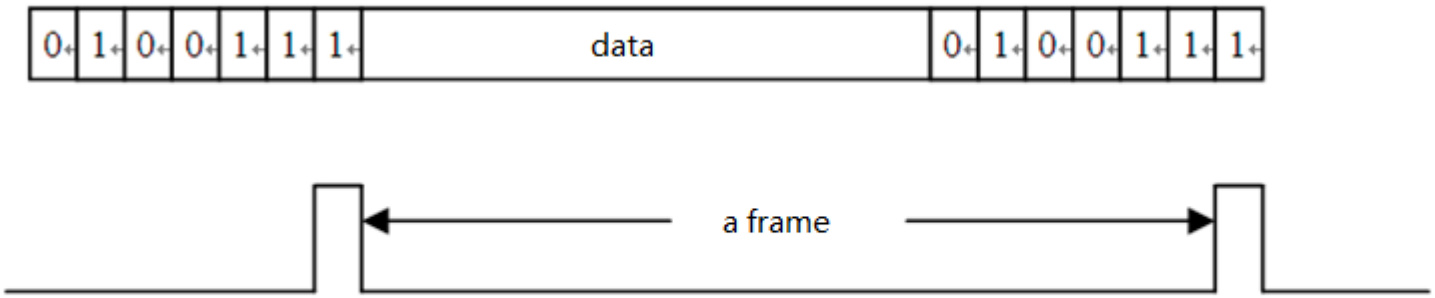
Barker Code

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.

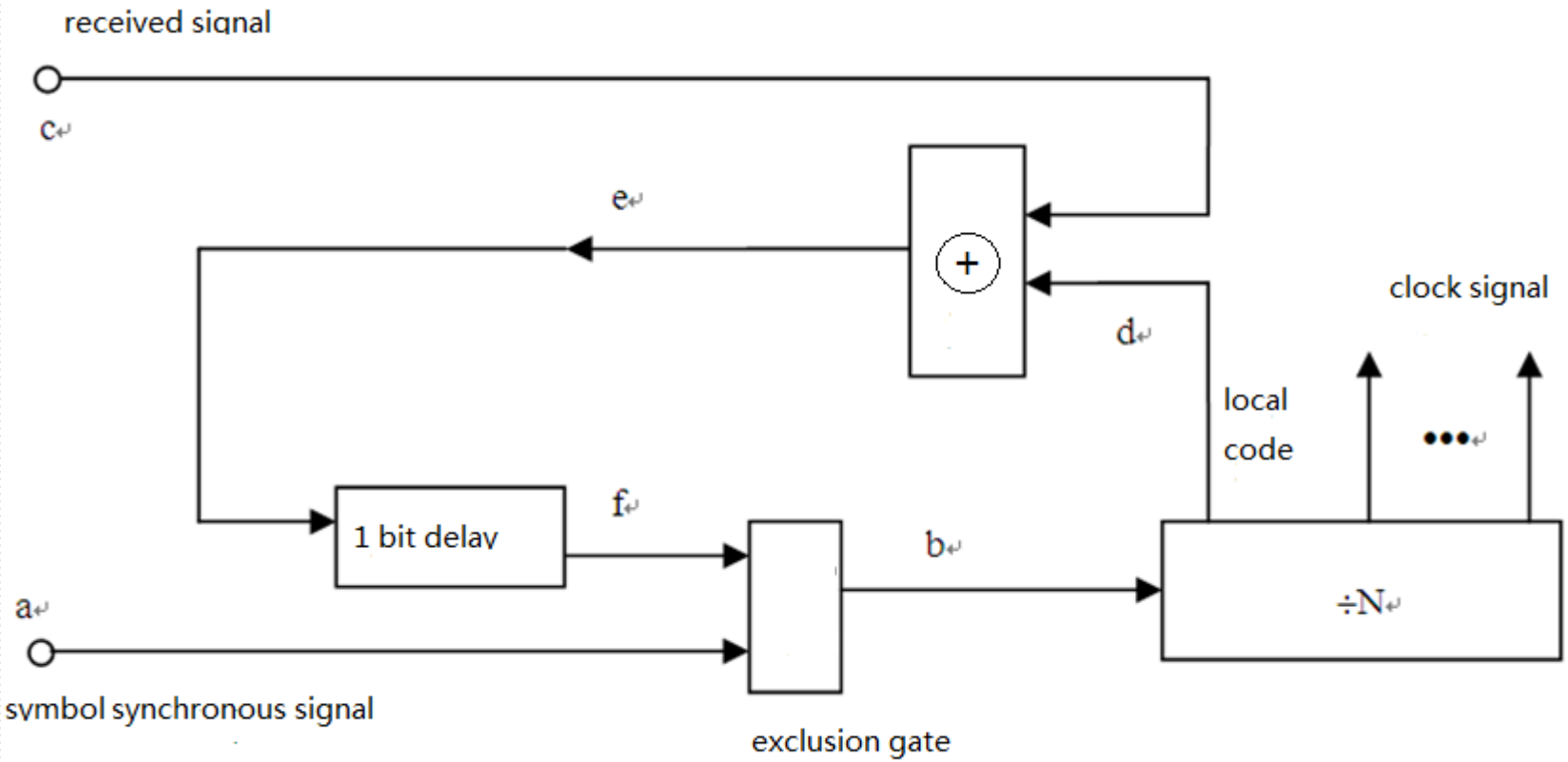
How to design synchronous code:

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

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

To determine the synchronous code, receiver need to detect the code bit by bit. Generally, the code can be detected by shifting the signal code by code.

Distributed frame alignment signal



Distributed frame alignment signal

Example： data code is all “0”， synchronous code is all“1”

The synchronous code is generated by frequency division (N=4)of the symbol synchronous impulse (a). In practical system, the local code (d) will not be exact the same with (c).Therefore, the output of the XOR gate will have nonzero waveform (e). After one bit delay (f), the exclusion gate will discard one symbol synchronous impulse (b). By repeating this procedure, finally (d) and (c) will be exactly the same, frame synchronization is realized.

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， symbol 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.