Chapter

19

Digital Phase-Locked Loops

The digital phase-locked loop, DPLL, is a circuit that is used frequently in modern integrated circuit design. Consider the waveform and block diagram of a communication system shown in Fig. 19.1. Digital data[1] is loaded into the shift register at the transmitting end. The data is shifted out sequentially to the transmitter output driver. At the receiving end, where the data may be analog (and, thus, without well-defined amplitudes) after passing through the communication channel, the receiver amplifies and changes the data back into digital logic levels. The next logical step in this sequence is to shift the data back into a shift register at the receiver and process the received data. However, the absence of a clock signal makes this difficult. The DPLL performs the function of generating a clock signal, which is locked or synchronized with the incoming signal. The generated clock signal of the receiver clocks the shift register and thus recovers the data. This application of a DPLL is often termed *a clock-recovery circuit* or *bit synchronization circuit.*

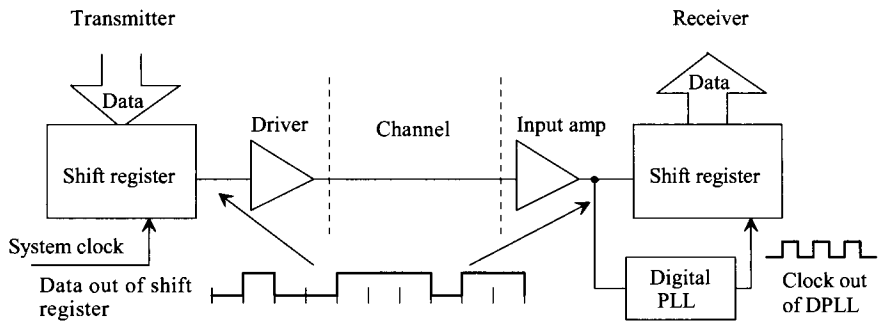


Figure 19.1 Block diagram of a communication system using a DPLL for the generation of a clock signal.

[1] While "data" is plural and "datum" is singular, we will not use, in this chapter, the grammatically correct "data are" in favor of the colloquial "data is".

A more detailed picture of the incoming data and possible clock signals out of the DPLL are shown in Fig. 19.2. The possible clock signals are labeled XOR *clock* and PFD (phase frequency detector) *clock*, corresponding to the type of phase[2] detector (PD) used. For the XOR PD, the rising edge of the clock occurs in the center of the data, while for the PFD, the rising edge occurs at the beginning of the data. The phase of the clock signal is determined by the PD used[3].

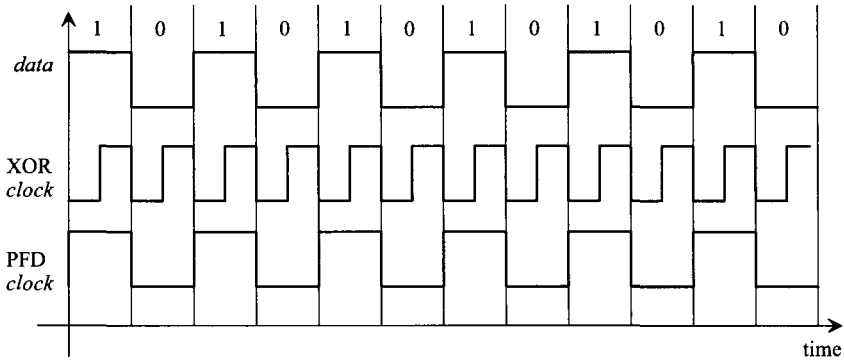


Figure 19.2 Data input to DPLL in lock and possible clock outputs using the XOR phase detector and PFD.

A block diagram of a DPLL is shown in Fig. 19.3. The PD generates an output signal proportional to the time difference between the *data in* and the divided down clock, *dclock*. This signal is filtered by a loop filter. The filtered signal, V_{inVCO}, is connected to the input of a voltage-controlled oscillator (VCO). Each one of these blocks is discussed in detail in the following sections. Once each block is understood, we will put them together and discuss the operation of the DPLL.

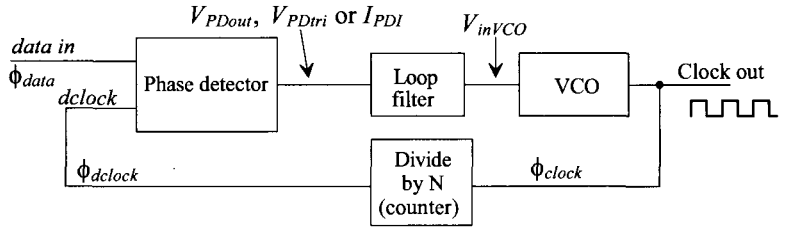


Figure 19.3 Block diagram of a digital phase-locked loop.

---

[2] A more correct name for digital applications is time difference detector (TDD).

[3] The PFD is rarely used in clock recovery applications. Figure 19.2 simply illustrates the different phase relations resulting from an XOR PD or PFD in a locked DPLL.

19.1 The Phase Detector

The first component in our DPLL is the phase detector. The two types of phase detectors, an XOR gate and a phase frequency detector (PFD), have significantly different characteristics. It is therefore very important to understand their performance capabilities and limitations. Selection of the type of phase detector is the first step in a DPLL design.

19.1.1 The XOR Phase Detector

The XOR PD is simply an exclusive OR gate. When the output of the XOR is a pulse train with a 50% duty cycle (square wave), the DPLL is said to be in lock; or in other words, the clock signal out of the DPLL is synchronized to the incoming data, provided the following conditions are met. Consider the XOR PD shown in Fig. 19.4. Let's begin by assuming that the incoming data is a string of zeros and that a divide by two is used in the feedback loop. The output of the phase detector is simply a replica of the *dclock* signal. Since the *dclock* signal has a 50% duty cycle, it would appear that the DPLL is in lock. If a logic "1" is suddenly applied, there is no way to know if the clock signal is synchronized (the clock rising edge coincides with the center of the data bit) to the data. This leads to the first characteristic of an XOR PD;

1. The incoming data must have a minimum number of transitions over a given time interval.

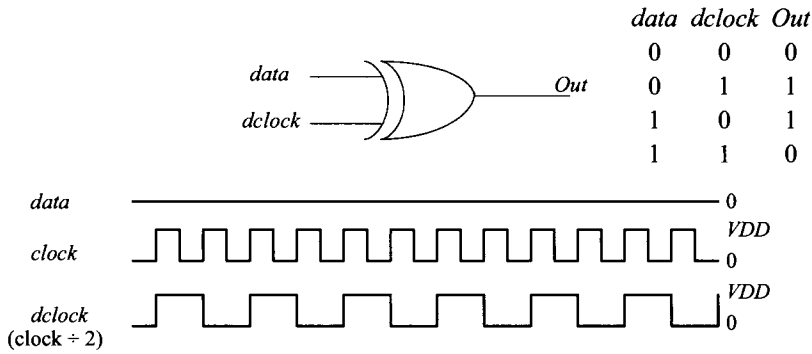


Figure 19.4 Operation of the XOR phase detector.

Now consider the situation when the output of the phase detector, with a string of zeros as the *data* input, is applied to a simple RC low-pass filter (Fig. 19.5). If RC >> period of the clock signal, the output of the filter is simply *VDD*/2. This leads to the second characteristic of the XOR phase detector.

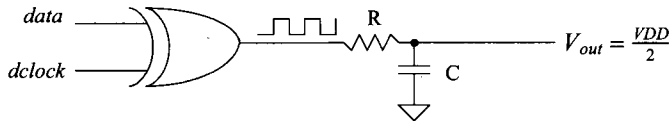


Figure 19.5 How the filtered output of the phase detector becomes VDD/2.

2. With no input data, the filtered output of the phase detector is *VDD*/2.

The voltage out of the loop filter is connected to the input of the VCO, as seen in Fig. 19.3. Consider the typical characteristics of a VCO shown in Fig. 19.6. The frequency of the square wave output of the VCO is f_{center} when V_{in} (= V_{center}) is *VDD*/2 (typically). The other two frequencies of interest are the minimum and maximum oscillator frequencies, f_{min} and f_{max} possible, with input voltage V_{min} and V_{max} , respectively. It is important that the VCO continues to oscillate with no input data. Normally, the VCO is designed so that the nominal data input rate and the VCO center frequency are the same. This minimizes the time it takes the DPLL to lock (and is critical for proper operation of the XOR PD).

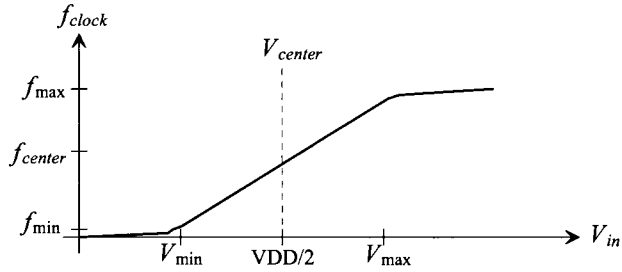


Figure 19.6 Output frequency of VCO versus input control voltage.

Now let's consider an example input to the phase-detector and corresponding output. The data shown in Fig. 19.7 is leading the *dclock* signal. The corresponding output of the phase detector is also shown. If this output is applied to a low-pass filter, the result is an average voltage less than *VDD*/2. This causes the VCO frequency to decrease until the edge of the *dclock* is centered on the data.

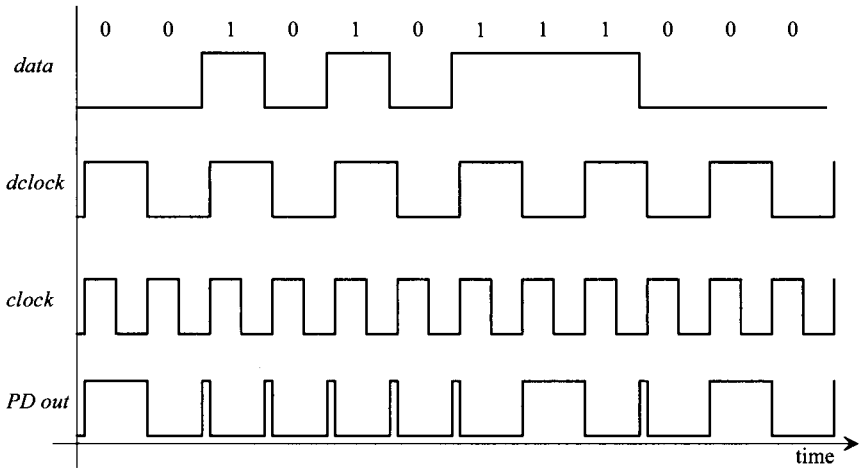


Figure 19.7 Possible XOR phase-detector inputs and the resulting output.

3. The time it takes the loop to lock depends on the data pattern input to the DPLL and the loop-filter characteristics.

Since the output of the PD is averaged, or more correctly integrated, noise injected into the data stream (a false bit) can be rejected. A fourth characteristic is:

4. The XOR DPLL has good noise rejection.

Another important characteristic of a DPLL is whether or not it will lock on a harmonic of the input data. The XOR DPLL will lock on harmonics of the data. To prove that this is indeed the case, consider replacing any of the clock signals of Figs. 19.2, 19.4, or 19.7 with a clock at twice or half the frequency. The average of the waveforms will remain essentially the same. A fifth characteristic of a DPLL using an XOR gate is:

5. The VCO operating frequency range should be limited to frequencies much less than $2f_{clock}$ and much greater than $0.5f_{clock}$, where f_{clock} is the nominal clock frequency for proper lock with a XOR PD.

The loop filter used with this type of PD is a simple RC low-pass filter, as shown in Fig. 19.5. Since the output of the PD is oscillating, the output of the filter shows a ripple as well, even when the loop is locked. This modulates the clock frequency, an unwanted characteristic of a DPLL using the XOR PD. This characteristic can be added to our list:

6. A ripple on the output of the loop filter with a frequency equal to the clock frequency modulates the control voltage of the VCO.

To characterize the phase detector (see Fig. 19.8), we can define the time difference between the rising edge of the *dclock* and the beginning of the *data* as Δt. The phase difference between the *dclock* and *data*, $\phi_{data} - \phi_{dclock}$, is given by

$$\Delta\phi = \phi_{data} - \phi_{dclock} = \frac{\Delta t}{T_{dclock}} \cdot 2\pi \text{ (radians)} \tag{19.1}$$

or, in terms of the DPLL output clock frequency,

$$\Delta\phi = \frac{\Delta t}{2T_{clock}} \cdot 2\pi \tag{19.2}$$

$$f_{clock} = \frac{1}{T_{clock}} = 2f_{dclock} = \frac{2}{T_{dclock}} \tag{19.3}$$

When the loop is locked, the *clock* rising edge is centered on the data; the time difference, Δt, between the *dclock* rising edge and the beginning of the data is simply $T_{clock}/2$ or $T_{dclock}/4$ (see Fig. 19.8c). Therefore, the phase difference between *dclock* and the *data*, under locked conditions, may be written as

$$\Delta\phi = \frac{\pi}{2} \tag{19.4}$$

Note that the phase difference between *clock* and *data* when in lock is π. The average voltage out of the phase detector (Fig. 19.8) may be expressed by

$$V_{PDout} = VDD \cdot \frac{\Delta\phi}{\pi} = K_{PD} \cdot \overbrace{\Delta\phi}^{input} \tag{19.5}$$

where the gain of the PD may be written as

$$K_{PD} = \frac{VDD}{\pi} \text{ (V/radians)} \tag{19.6}$$

As an aid in the understanding of these equations, consider the diagrams shown in Fig. 19.8. If the edges of the clock and data are coincident in time (Fig. 19.8a), the XOR output, V_{PDout}, is 0 V and the phase difference is 0. The loop filter averages the output of the PD and causes the VCO to lower its output frequency. This causes $\Delta\phi$ to increase, thus increasing V_{PDout}. Depending on the selection of the loop filter, this increase could cause the rising edges to increase beyond, or *overshoot*, the desirable center point, as shown in Fig. 19.8b. In this case, the phase difference is $-\frac{3}{4}\pi$, and V_{PDout} is $\frac{3}{4}VDD$. Figure 19.8c shows the condition when the loop is in lock and the phase difference is $\pi/2$.

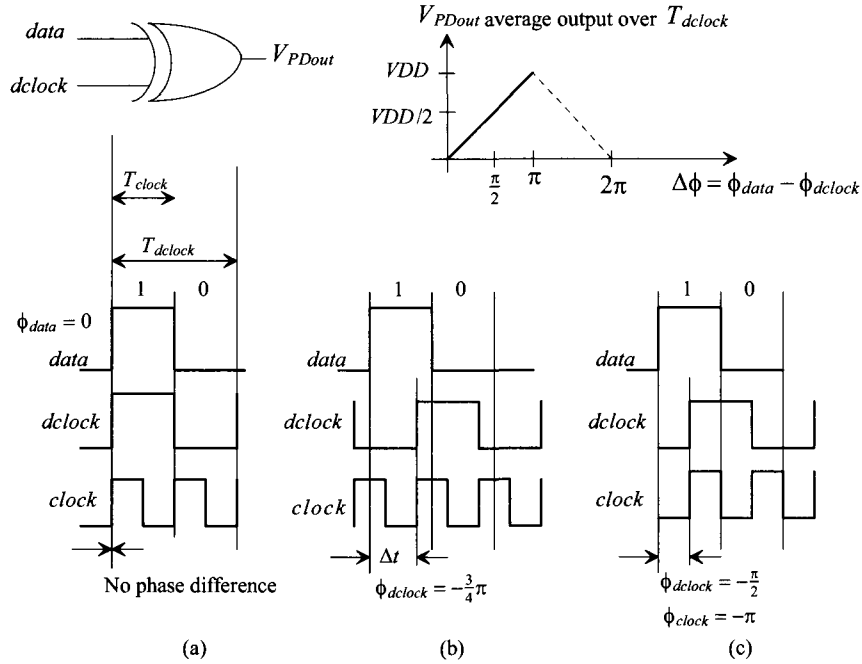


Figure 19.8 XOR PD output for various inputs (assuming input data are a string of alternating ones and zeros).

Here we should point out the importance of the VCO center frequency, f_{center}, being equal to the desired clock frequency. If $f_{center} = f_{clock}$, the VCO control voltage, depending on the loop filter, will look similar to Fig. 19.9a during acquisition (the loop trying to lock). If these frequencies are not equal, the control voltage will oscillate, causing the clock to move, in time, around some other point than the center of the data bit (Fig. 19.9b). The actual control voltages will look different from those portrayed in Fig. 19.9 because the VCO control voltage depends on the input data pattern, as discussed earlier.

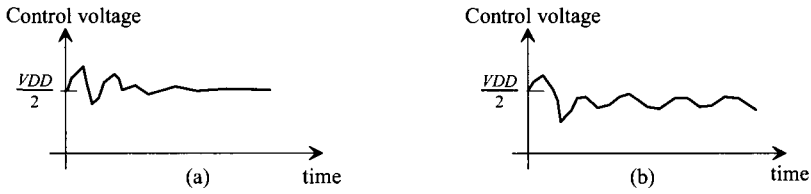


Figure 19.9 Average output voltage of phase detector during acquisition.

To summarize the design criteria of the VCO with the XOR PD, we desire that:

1. The center frequency, f_{center}, should equal the clock frequency when the VCO control voltage is *VDD*/2.

2. The maximum and minimum oscillation frequencies, f_{max} and f_{min}, of the VCO should be selected to avoid locking on harmonics of the input data.

3. The VCO duty cycle is 50%. If this is not the case, the DPLL will have problems locking, or once locked the clock will jitter (move around in time).

19.1.2 The Phase Frequency Detector

A schematic diagram of the phase frequency detector is shown in Fig. 19.10. The output of the PFD depends on both the phase and frequency of the inputs. This type of phase detector is also termed a sequential phase detector. It compares the leading edges of the *data* and *dclock*. A *dclock* rising edge cannot be present without a data rising edge. To aid in understanding the PFD, consider the examples shown in Fig. 19.11. The first thing we notice is that the *data* pulse width and the *dclock* pulse width do not matter. If the rising edge of the *data* leads the *dclock* rising edge (Fig. 19.11a), the "up" output of the phase detector goes high, while the *Down* output remains low. This causes the *dclock* frequency to increase, having the effect of moving the edges closer together. When the *dclock* signal leads the data (Fig. 19.11b), *Up* remains low, while the *Down* goes high a time equal to the phase difference between *dclock* and *data*. Figure 19.11c shows the condition of the locked loop. Notice that, unlike the XOR PD, the outputs remain low when the loop is locked. Again, several characteristics of the PFD can be described:

1. A rising edge from the *dclock* and *data* must be present when making a phase comparison.

2. The widths of the *dclock* and the *data* are irrelevant.

3. The PFD will not lock on a harmonic of the data.

4. The outputs (*Up* and *Down*) of the PFD are both logic low when the loop is in lock, eliminating ripple on the output of the loop filter.

5. This PFD has poor noise rejection; a false edge on either the *data* or the *dclock* inputs drastically affects the output of the PFD.

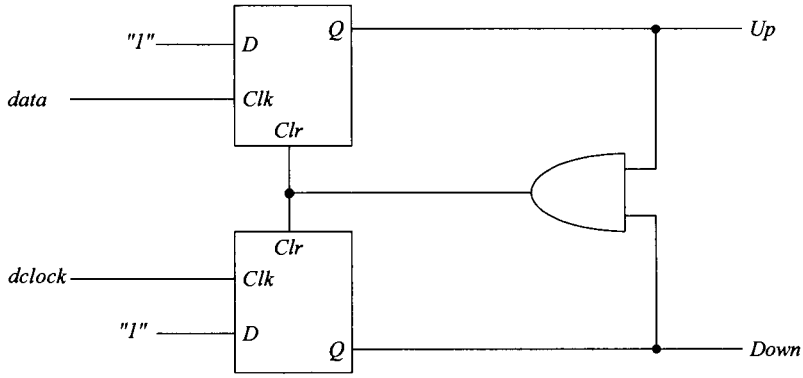


Figure 19.10 Phase frequency detector (PFD).

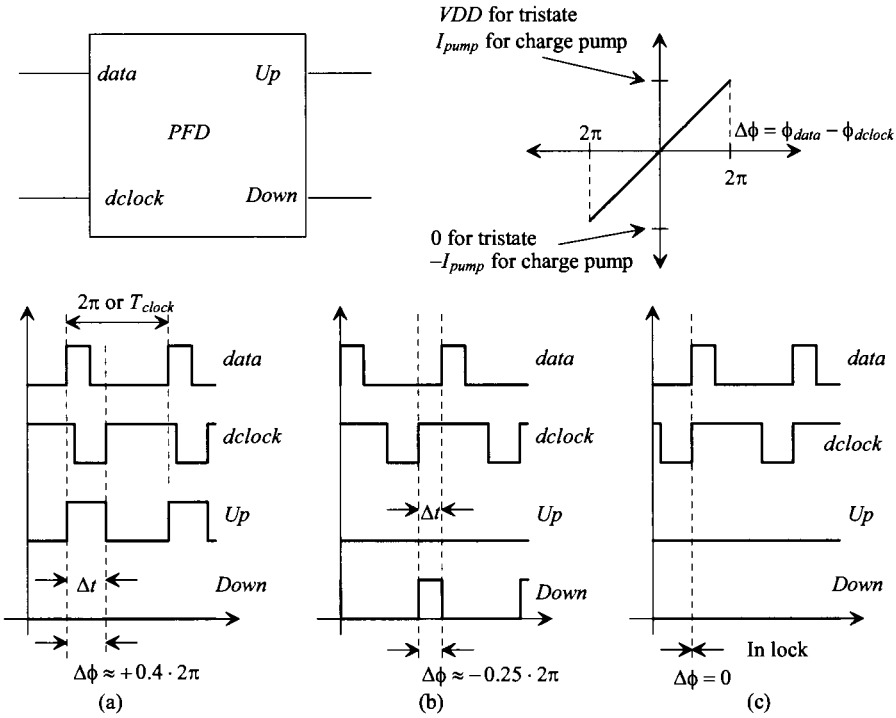


Figure 19.11 PFD phase-detector inputs and outputs.

The output of the PFD should be combined into a single output for driving the loop filter. There are two methods of doing this, both of which are shown in Fig. 19.12. The first method is called a *tri-state* output. When both signals, *Up* and *Down*, are low, both MOSFETs are off and the output is in a high-impedance state. If the *Up* signal goes high, M2 turns on and pulls the output up to *VDD*, while if the *Down* signal is high, the output is pulled low through M1. The main problem that exists with this configuration is that power supply variations can significantly affect the output voltage when M2 is on. The effect is to modulate the VCO control voltage. This wasn't as big a problem when the XOR PFD was used due to the averaging taking place.

The second configuration shown in this figure is the so-called *charge pump*. MOS current sources are placed in series with M1 and M2. When the PFD *Up* signal goes high, M2 turns on, connecting the current source to the loop filter. Because the current source can be made insensitive to supply variations, modulation of the VCO control voltage is absent.

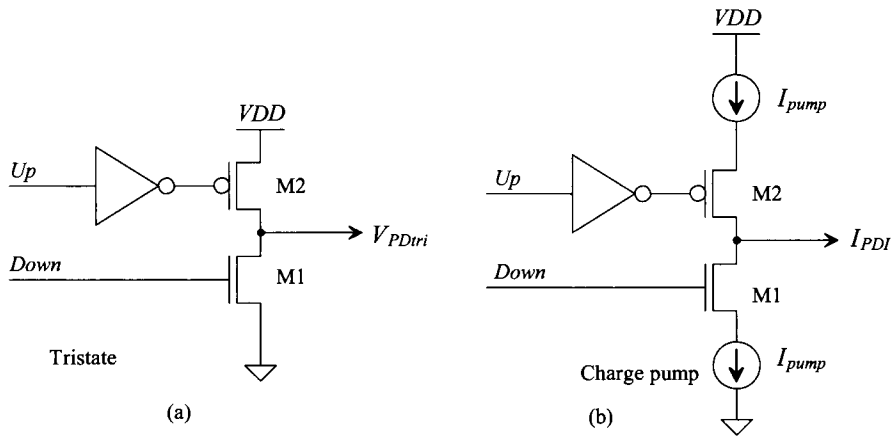


Figure 19.12 (a) Tri-state and (b) charge pump outputs of the PFD.

We can characterize the PFD in much the same manner as we did the XOR PD. We will assume that f_{clock} and f_{dclock} are equal in this analysis. Therefore, the feedback loop, Fig. 19.3, divides by one ($N = 1$). If we again assume that the time difference between the rising edges of the *data* and *dclock* is labeled Δt and the time between leading edges of the *clock* (or the time differences between the leading edges of the *data* since we must have both leading edges present to do a phase comparison) is labeled T_{clock}, then we can write the phase ($T_{clock} = T_{dclock}$) as

$$\Delta\phi = \frac{\Delta t}{T_{clock}} \cdot 2\pi \text{ (radians)} \tag{19.7}$$

The phase difference, $\Delta\phi$, is zero when the loop is in lock. The output voltage of the PFD using the tri-state output configuration (see Fig. 19.11) is

$$V_{PDtri} = \frac{VDD - 0}{4\pi} \cdot \Delta\phi = K_{PDtri} \cdot \Delta\phi \tag{19.8}$$

where the gain is,

$$K_{PDtri} = \frac{VDD}{4\pi} \text{ (volts/radian)} \tag{19.9}$$

If the output of the PFD uses the charge-pump configuration, the output current can be written (again see Fig. 19.11) as

$$I_{PDI} = \frac{I_{pump} - (-I_{pump})}{4\pi} \cdot \Delta\phi = K_{PDI} \cdot \Delta\phi \tag{19.10}$$

where

$$K_{PDI} = \frac{I_{pump}}{2\pi} \text{ (amps/radian)} \tag{19.11}$$

The loop filters used with tri-state and the charge-pump outputs are shown in Fig. 19.13. The first loop filter has a transfer function given by

$$V_{inVCO} = \frac{1 + j\omega R_2 C}{1 + j\omega(R_1 + R_2)C} \cdot V_{PDtri} = K_F \cdot V_{PDtri} \tag{19.12}$$

The charge-pump loop-filter transfer function is given (noting the input variable is a current while the output is a voltage) by

$$V_{inVCO} = I_{PDI} \cdot \frac{1 + j\omega RC_1}{j\omega(C_1 + C_2) \cdot \left[1 + j\omega R \frac{C_1 C_2}{C_1 + C_2}\right]} = K_F \cdot I_{PDI} \tag{19.13}$$

To qualitatively understand how these loop filters work, let's begin by considering the loop filter for use with the tri-state output. For slow variations in the phase difference, the filter acts like an integrator averaging the output of the PD. For fast variations, however, the filter looks like a resistive divider without any integration. This allows the loop filter to track fast variations in the time difference between the rising edges. A similar discussion can be made for the loop filter used with the charge pump. For slow variations in the phase, the current, I_{pump}, linearly charges C_1 and C_2. This gives an averaging effect. For fast variations, the charge pump simply drives the resistor R (assuming C_2 is small), eliminating the averaging and allowing the VCO to track quickly moving variations in the input data.

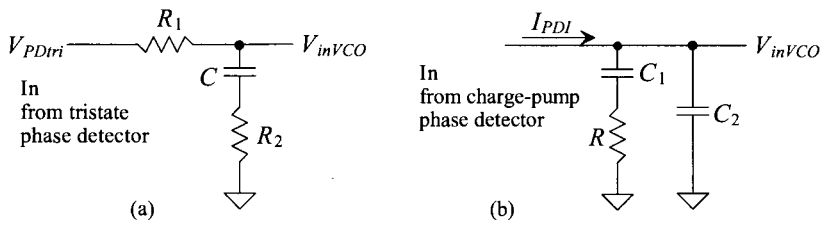


Figure 19.13 Loop filters for (a) tristate output and (b) charge-pump output.

The design requirements of the VCO used with the PFD can be much more relaxed than those for the XOR PD; therefore, it is preferred over the XOR PD. Because the output of the PD can be a voltage from 0 to *VDD*, the requirement that $f_{center} = f_{clock}$ is not present, although it is still a good idea to design the VCO so that this is true. The oscillation range, $f_{max} - f_{min}$, is not limited by the harmonics of f_{clock} because the PFD will not lock on a harmonic of the clock frequency. The VCO clock duty cycle is irrelevant with the PFD because the PD looks only at rising edges. In high-speed or data communications applications, however, one may be forced to use the XOR PD.

19.2 The Voltage-Controlled Oscillator

The gain of the voltage-controlled oscillator is simply the slope of the curves given in Fig. 19.6. This gain can be written as

$$K_{VCO} = 2\pi \cdot \frac{f_{\max} - f_{\min}}{V_{\max} - V_{\min}} \text{ (radians/s} \cdot \text{V)} \tag{19.14}$$

The VCO output frequency, f_{clock}, is related to the VCO input voltage (see Fig. 19.6) by

$$\omega_{clock} = 2\pi \cdot f_{clock} = K_{VCO} \cdot V_{inVCO} + \omega_o \text{ (radians/s)} \tag{19.15}$$

where ω_0 is a constant. However, the variable we are feeding back is not frequency but phase (hence the name of the circuit). The phase of the VCO clock output is related to f_{clock} by

$$\phi_{clock} = \int \omega_{clock} \cdot dt = \frac{K_{VCO}}{j\omega} \cdot V_{inVCO} \text{ (radians)} \tag{19.16}$$

where this signal can be related to the ϕ_{dclock} by

$$\phi_{dclock} = \frac{1}{N} \cdot \phi_{clock} = \beta \cdot \phi_{clock} \tag{19.17}$$

where N is the divide by count and β is the feedback factor.

19.2.1 The Current-Starved VCO

The current-starved VCO is shown schematically in Fig. 19.14. Its operation is similar to the ring oscillator discussed earlier. MOSFETs M2 and M3 operate as an inverter, while MOSFETs M1 and M4 operate as current sources. The current sources, M1 and M4, limit the current available to the inverter, M2 and M3; in other words, the inverter is starved for current. The drain currents of MOSFETs M5 and M6 are the same and are set by the input control voltage. The currents in M5 and M6 are mirrored in each inverter/current source stage. An important property of the VCO used in any of the CMOS DPLLs discussed in this chapter is the input impedance. The filter configurations we have discussed rely on the fact that the input resistance of the VCO is practically infinite and the input capacitance is small compared to the capacitances present in the loop filter. Attaining infinite input resistance is usually an easy part of the design. For the charge-pump configuration, the input capacitance of the VCO can be added to C_2.

To determine the design equations for use with the current-starved VCO, consider the simplified schematic of one stage of the VCO shown in Fig. 19.15. The total capacitance on the drains of M2 and M3 is given by

$$C_{tot} = C_{out} + C_{in} = \overbrace{C'_{ox}(W_pL_p + W_nL_n)}^{C_{out}} + \overbrace{\frac{3}{2}C'_{ox}(W_pL_p + W_nL_n)}^{C_{in}} \quad (19.18)$$

which is simply the output and input capacitances of the inverter. This equation can be written in a more useful form as

$$C_{tot} = \frac{5}{2}C'_{ox}(W_pL_p + W_nL_n) \quad (19.19)$$

The time it takes to charge C_{tot} from zero to V_{SP} with the constant-current I_{D4} is given by

$$t_1 = C_{tot} \cdot \frac{V_{SP}}{I_{D4}} \quad (19.20)$$

while the time it takes to discharge C_{tot} from VDD to V_{SP} is given by

$$t_2 = C_{tot} \cdot \frac{VDD - V_{SP}}{I_{D1}} \quad (19.21)$$

If we set $I_{D4} = I_{D1} = I_D$ (which we will label $I_{Dcenter}$ when $V_{inVCO} = VDD/2$), then the sum of t_1 and t_2 is simply

$$t_1 + t_2 = \frac{C_{tot} \cdot VDD}{I_D} \quad (19.22)$$

The oscillation frequency of the current-starved VCO for N (an odd number ≥ 5) of stages is

$$f_{osc} = \frac{1}{N(t_1 + t_2)} = \frac{I_D}{N \cdot C_{tot} \cdot VDD} \quad (19.23)$$

which is $= f_{center}(@V_{inVCO} = VDD/2 \text{ and } I_D = I_{Dcenter})$

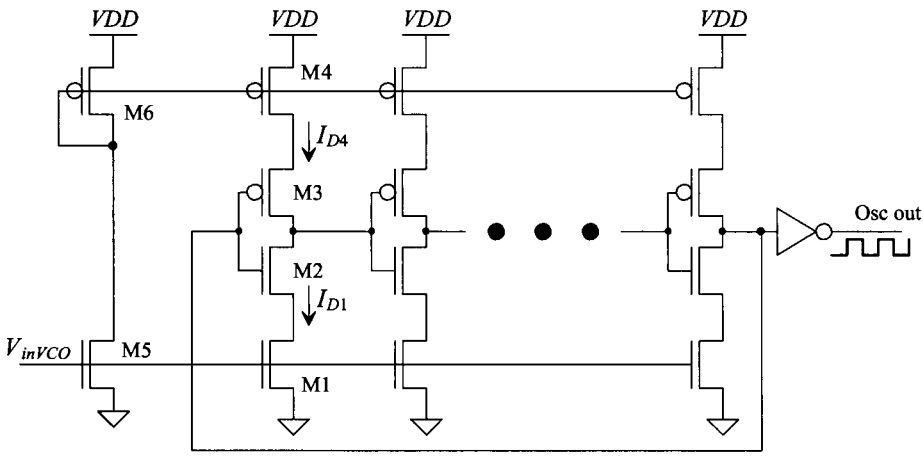


Figure 19.14 Current-starved VCO.

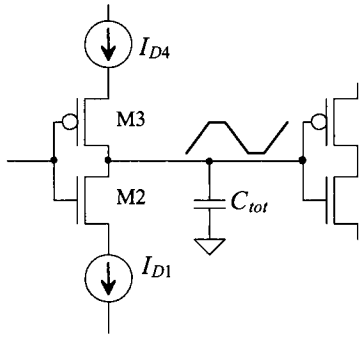


Figure 19.15 Simplified view of a single stage of the current-starved VCO.

Equation (19.23) gives the center frequency of the VCO when $I_D = I_{Dcenter}$. The VCO stops oscillating, neglecting subthreshold currents, when $V_{inVCO} < V_{THN}$. Therefore, we can define

$$V_{\min} = V_{THN} \text{ and } f_{\min} = 0 \tag{19.24}$$

The maximum VCO oscillation frequency, f_{max}, is determined by finding I_D when $V_{inVCO} = VDD$. At the maximum frequency then, $V_{max} = VDD$.

The output of the current-starved VCO shown in Fig. 19.14 is normally buffered through one or two inverters. Attaching a large load capacitance on the output of the VCO can significantly affect the oscillation frequency or lower the gain of the oscillator enough to kill oscillations altogether.

The average current drawn by the VCO is

$$I_{avg} = N \cdot \frac{VDD \cdot C_{tot}}{T} = N \cdot VDD \cdot C_{tot} \cdot f_{osc} \tag{19.25}$$

or

$$I_{avg} = I_D \tag{19.26}$$

The average power dissipated by the VCO is

$$P_{avg} = VDD \cdot I_{avg} = VDD \cdot I_D \tag{19.27}$$

If we include the power dissipated by the mirror MOSFETs, M5 and M6, the power is doubled from that given by Eq. (19.27), assuming that $I_D = I_{D5} = I_{D6}$. For low-power dissipation we must keep I_D low, which is equivalent to stating that for low-power dissipation we must use a low-oscillation frequency.

Example 19.1

Design a current-starved VCO with f_{center} = 100 MHz in the short-channel process. Simulate the design using SPICE.

We begin by calculating the total capacitance, C_{tot}. Using Eq. (19.19) and assuming the inverters, M2 and M3, are sized for equal drive, that is, $L_n = L_p = 1$, $W_n = 10$ and $W_p = 20$, the capacitance is

$$C_{tot} = \frac{5}{2} \cdot 25\frac{fF}{\mu m^2} \cdot (10 \cdot 1 + 20 \cdot 1) \cdot \overbrace{(0.050\ \mu m)^2}^{\text{scale factor}} = 4.7\,fF$$

Let's use a center drain current of 10 μA based on the I_D - V_{GS} characteristics of the MOSFETs. The selection of this current is important. We want, when V_{inVCO} is *VDD*/2, the oscillation frequency to be 100 MHz. Here, just to show the design procedure, we will simply adjust the V_{inVCO} to set the 100 MHz output frequency.

The number of stages, using Eq. (19.23), is given by

$$N = \frac{I_D}{f_{osc} \cdot C_{tot} \cdot VDD} = \frac{10\ \mu A}{100\ MHz \cdot 4.7\,fF \cdot 1\ V} \approx 21$$

The simulation results are shown in Fig. 19.16. For an output frequency of 100 MHz, the input voltage, V_{inVCO} , was set to 450 mV. The big problem with this design is that the output oscillation frequency is not linearly related to the control voltage (easy to verify using the simulation that generated Fig. 19.16). Having a nonlinear VCO gain can greatly reduce the quality of the performance of the DPLL (*this is important*). The output of the phase-locked loop can jitter (move around) or may not lock at all when the VCO gain is nonlinear. ■

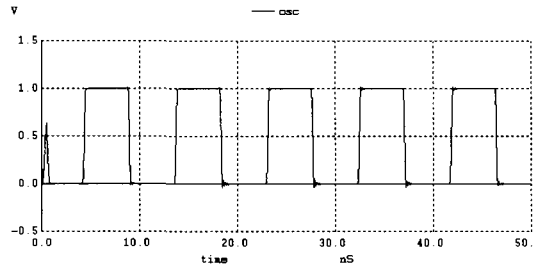


Figure 19.16 Output of the oscillator designed in Ex. 19.1.

Linearizing the VCO's Gain

After studying Fig. 19.14, we see that applying V_{inVCO} directly to the gate of M5 causes the current in M5 (and M6) to be nonlinear. In the long-channel case the drain current of a MOSFET is related to the square of the MOSFET's V_{GS}. To make the current in a MOSFET linearly related to the VCO's input voltage, consider the circuit seen in Fig. 19.17. The width of M5R is made wide so that its V_{GS} is always (independent of V_{inVCO}) approximately V_{THN}. Note that the current in M6R is mirrored over to M6 and M5 to control the current used in the current-starved VCO. Figure 19.18 shows the output of the oscillator (and its transfer curves) in Fig. 19.16 if the linearizing scheme in Fig. 19.17 is used with *R* of 10k and a wide device (M5R) with a size of 100/1. The gain of the VCO is $K_{VCO} = 2\pi \cdot 25MHz/100mV = 1.57 \times 10^9\ radians/V \cdot s$. We'll use this VCO in examples later in this chapter. Note that the gain of this *VCO* is rather high. A 10 mV voltage variation in V_{inVCO} gives a 2.5 MHz variation in the output frequency (e.g., the output frequency varies from 100 to 102.5 MHz). In the time domain, the variation in the period (jitter) is then $1/100\ MHz - 1/102.5\ MHz = 244\ ps$ (roughly 2.5% of the ideal 10 ns period). *For any low jitter PLL design, a VCO with low gain must be used.*

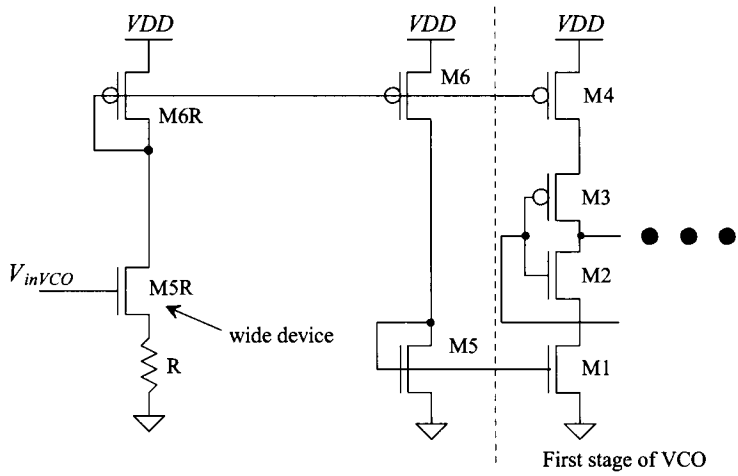


Figure 19.17 Linearizing the current in a current-starved VCO.

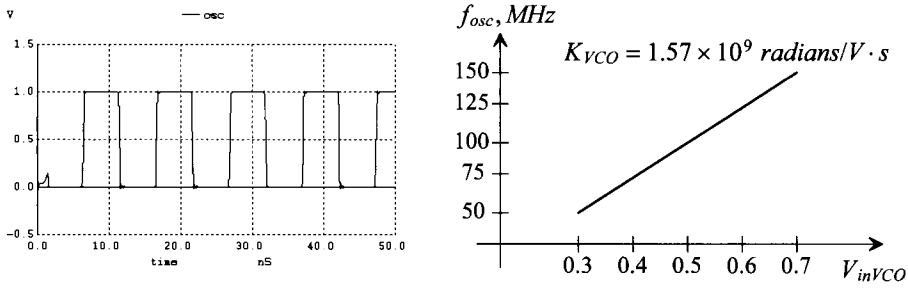


Figure 19.18 Gain of the VCO in Fig. 19.16 after linearizing with Fig. 19.17.

19.2.2 Source-Coupled VCOs

Another variety of VCO, source-coupled VCOs, is shown in Fig. 19.19. These VCOs can be designed to dissipate less power than the current-starved VCO of the last section for a given frequency. The major disadvantage of these configurations is the need for a capacitor, something that may not be available in a single-poly pure digital process without using parasitics for example, a metal1 to metal2 capacitor, and a reduced output voltage swing. However, this configuration is useful when the VCO center frequency is set by an external capacitor; that is, the capacitor shown in the figure is bonded out.

To understand the operation, let's consider the NMOS source-coupled VCO of Fig. 19.19a. The operation of the CMOS source-coupled VCO of Fig. 19.19b is identical to (a) except for the fact that the load MOSFETs M3 and M4 pull the outputs to $VDD - V_{THN}$ (for the NMOS-load VCO) and VDD (for the PMOS-load VCO). The buffer in Fig. 18.17 of the last chapter can be used to restore full logic levels.

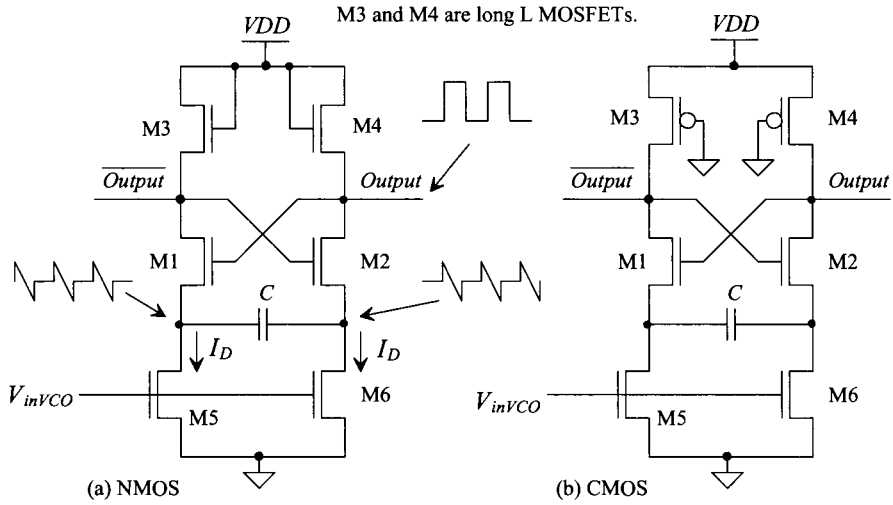
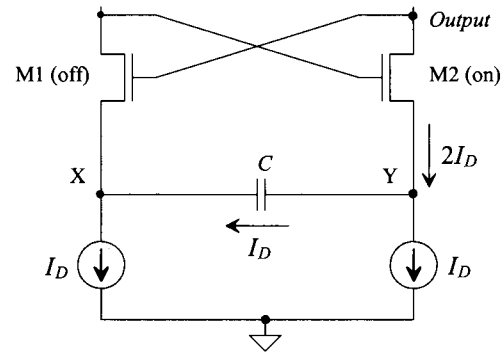


Figure 19.19 Source coupled voltage-controlled oscillators (also known as source coupled multivibrators).

For the circuit in Fig. 19.19a, MOSFETs M5 and M6 behave as constant-current sources sinking a current I_D. MOSFETs M1 and M2 operate as switches. If M1 is off and M2 is on, the drain of M1 is pulled to $VDD - V_{THN}$ by M3. Since the gate of M2 is at $VDD - V_{THN}$, the source and drain (the *Output*) of M2 are approximately $VDD - 2 \cdot V_{THN}$. This is the minimum output voltage. The output voltage swing is limited to V_{THN}. A simplified schematic shown in Fig. 19.20 with M1 off and M2 on is helpful in determining the oscillator frequency. The *Output* gate of M1 is approximately $VDD - 2 \cdot V_{THN}$ and is held at this voltage through M2 until M1 turns on and M2 turns off. Initially, at the moment when M1 turns off and M2 turns on, point X is $VDD - V_{THN}$. The current through C, I_D, causes point X to discharge down toward ground. When point X gets down to $VDD - 3 \cdot V_{THN}$, M1 turns on and M2 turns off. In other words, the voltage at point X changed a



Note: C should be laid out with the same parasitic capacitance at X and Y.

Figure 19.20 Simplified schematic of source coupled oscillator, M1 is on and M2 is off.

total of $2 \cdot V_{THN}$ before switching took place. The time it takes point X to change $2 \cdot V_{THN}$ is given by

$$\Delta t = C \cdot \frac{2 \cdot V_{THN}}{I_D} \tag{19.28}$$

Since the circuit is symmetrical, two of these discharge times are needed for each cycle of the oscillation. The frequency of oscillation is given by

$$f_{osc} = \frac{1}{2\Delta t} = \frac{I_D}{4 \cdot C \cdot V_{THN}} \tag{19.29}$$

The waveforms at the points X, Y, and *Output* are shown in Fig. 19.21 for continuous time operation.

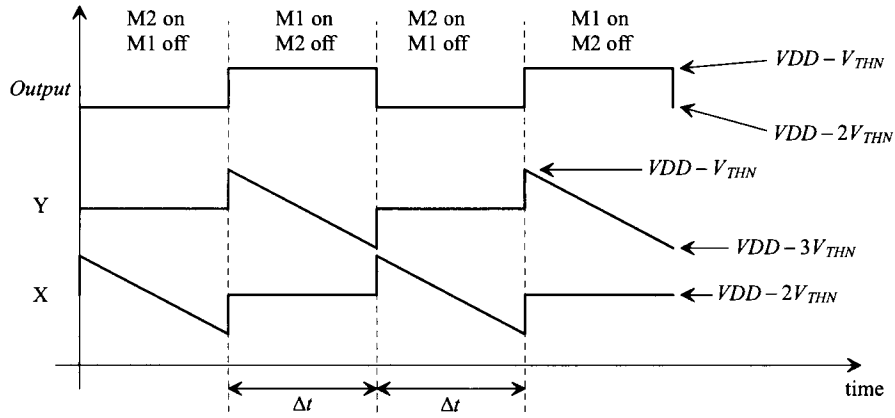


Figure 19.21 Voltage waveforms for the NMOS source-coupled VCO.

19.3 The Loop Filter

The loop filter is the brain of the DPLL. In this section, we discuss how to select the loop-filter values in order to keep the DPLL from oscillating (i.e., keep the V_{inVCO} voltage from oscillating, causing the frequency out of the VCO to wander). If the loop-filter values are not selected correctly, it may take the loop too long to lock, or once locked, small variations in the input data may cause the loop to unlock.

In the following discussion, we will be concerned with the *pull-in range* and the *lock range*. The pull-in range, $\pm\Delta\omega_P$, is defined as the range of input frequencies that the DPLL will lock to. The time it takes the loop to lock is labeled T_p and can be a very long time. If the center frequency of the DPLL is 10 MHz and the pull-in range is 1 MHz, the DPLL will lock on an input frequency from 9 to 11 MHz in a time T_p (assuming $N = 1$). The lock range, $\pm\omega_L$, is the range of frequencies in which the DPLL locks within one single beat note between the divided down output (*dclock*) and input (*data*) of the DPLL. *The operating frequency of the DPLL should be limited to the lock range for normal operation.* Once the DPLL is locked, it will remain locked as long as abrupt frequency changes, $\Delta\omega$, in the input frequency (input frequency steps) over a time interval t are much smaller than the natural frequency of the system squared, that is, $\Delta\omega/t < \omega_n^2$.

19.3.1 XOR DPLL

Consider the block diagram of the DPLL using the XOR DPLL shown in Fig. 19.22. The phase transfer function (neglecting the static or DC behavior) is given by

$$H(s) = \frac{\phi_{clock}}{\phi_{data}} = \frac{K_{PD}K_FK_{VCO}}{s + \beta \cdot K_{PD}K_FK_{VCO}} \tag{19.30}$$

with $s = j\omega$ and the feedback factor, β, is

$$\beta = \frac{1}{N} \tag{19.31}$$

The transfer function of the loop filter is given by

$$K_F = \frac{1}{1 + j\omega RC} = \frac{1}{1 + sRC} \tag{19.32}$$

Substituting Eqs. (19.31) and (19.32) into Eq. 19.30 yields

$$H(s) = \frac{\phi_{clock}}{\phi_{data}} = \frac{K_{PD}K_{VCO} \cdot \frac{1}{1+sRC}}{s + \frac{1}{N} \cdot K_{PD}K_{VCO} \cdot \frac{1}{1+sRC}} \tag{19.33}$$

This is a second-order system. $H(s)$ can be rewritten as

$$H(s) = \frac{\frac{K_{PD}K_{VCO}}{RC}}{s^2 + \frac{s}{RC} + \frac{1}{N} \cdot \frac{K_{PD}K_{VCO}}{RC}} = \frac{f_{clock}}{f_{data}} = \frac{(K_{VCO}V_{inVCO})/2\pi + f_o}{f_{data}} \tag{19.34}$$

since $j\omega \cdot \phi = f$. The natural frequency, ω_n, of this system is given by

$$\omega_n = \sqrt{\frac{K_{PD}K_{VCO}}{N \cdot RC}} \tag{19.35}$$

and the damping ratio, ζ, is given by

$$\zeta = \frac{1}{2RC\omega_n} = \frac{1}{2} \cdot \sqrt{\frac{N}{K_{PD}K_{VCO} \cdot RC}} \tag{19.36}$$

The pull-in range is given by

$$\Delta\omega_P = \frac{\pi}{2} \cdot \sqrt{2\zeta\omega_n K_{VCO}K_{PD} - \omega_n^2} \tag{19.37}$$

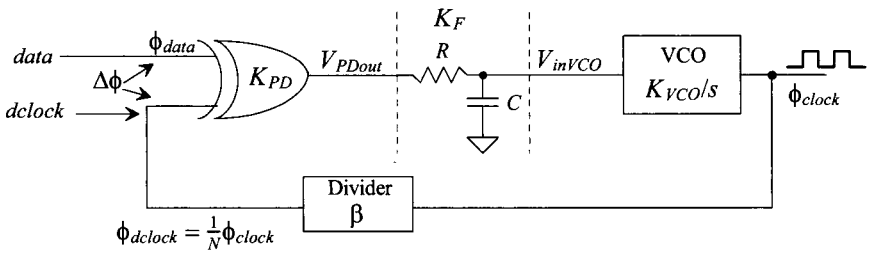


Figure 19.22 Block diagram of a DPLL using a XOR phase detector

while the pull-in time is given by

$$T_P = \frac{4}{\pi^2} \cdot \frac{\Delta\omega_{center}^2}{\zeta\omega_n^3} \tag{19.38}$$

The lock range of the loop is given by

$$\Delta\omega_L = \pi\zeta\omega_n = \frac{\pi}{2} \cdot \frac{1}{RC} \tag{19.39}$$

while the lock time is

$$T_L = \frac{2\pi}{\omega_n} \tag{19.40}$$

Probably the best way to understand how these equations affect the performance of a DPLL is through an example.

Example 19.2

Design and simulate the operation of a DPLL with the topology seen in Fig. 19.22 using the VCO from Fig. 19.18. Assume that the input data rate is 100 Mbits/s and has a format as in Fig. 19.7 (so the DPLL's output is a 100 MHz square wave).

Let's start off by drawing the schematic of the XOR phase detector, Fig. 19.23, and the divide by two circuit, Fig. 19.24. Notice that the XOR gate won't have zero output resistance, so the value of R used in the loop filter may need to be modified accordingly. The divide by two circuit is implemented using true single phase logic.

To calculate the values for the loop filter, let's write the gain of the VCO in Fig. 19.18 as

$$K_{VCO} = 1.57 \times 10^9 \; radians/V \cdot s$$

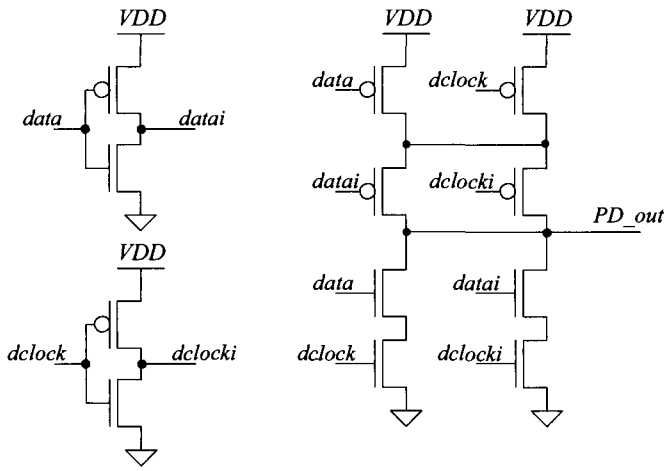


Figure 19.23 XOR phase detector.

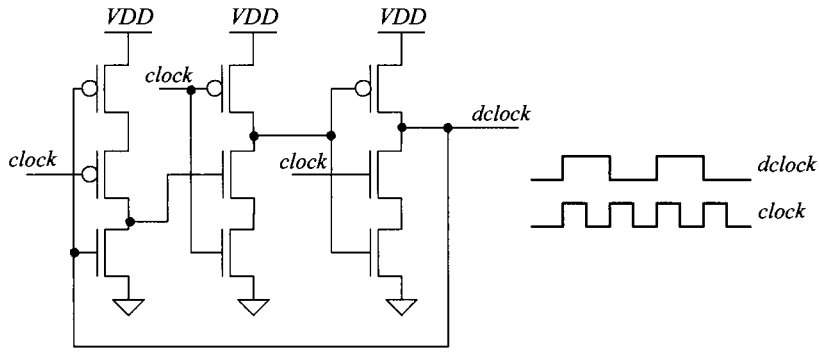


Figure 19.24 A divide by two circuit using true single phase clocking (TSPC) logic.

and the gain of the XOR phase detector

$$K_{PD} = \frac{VDD}{\pi} = \frac{1}{\pi} \text{ (V/radians)}$$

Let's begin by setting the damping ratio ζ to 1. Using Eq. (19.36) results in

$$1 = \frac{1}{2} \cdot \sqrt{\frac{2}{\frac{1}{\pi} \cdot (1.57 \times 10^9) RC}} \rightarrow RC = 500\, ps$$

clearly not practical! We must reduce the gain of the VCO. This means the VCO output frequency range must be reduced. The problem with reducing the output frequency range is that it becomes very difficult to fabricate the VCO in CMOS at the precise center oscillation frequency. Nonetheless, let's modify the VCO from Fig. 19.18 so that the output frequency range is limited.

Consider the schematic seen in Fig. 19.25 of the current generator portion of the VCO (see Fig. 19.17). Here, we've added a resistor to set the lower frequency range of the VCO (which makes the VCO very dependent on the power supply voltage). The range is still set by M5R. However, now we've increased the value of the resistor to limit the VCO's output frequency range. The resulting VCO gain, with the values seen in Fig. 19.25, appears in Fig. 19.26. Using this modified VCO gain in the above equation gives

$$RC = 5 \text{ ns}$$

The natural frequency of the second-order system is, from Eq. (19.36),

$$\omega_n = \frac{1}{2 \cdot RC \cdot \zeta} = 100 \times 10^6 \text{ radians/s}$$

The lock range is

$$\Delta\omega_L = \pi\zeta\omega_n = 314 \times 10^6 \text{ radians/s or } \Delta f_L = 50\ MHz$$

which is outside the VCO operating range. Therefore, this DPLL will lock over the entire VCO operating range, that is, from 95 to 105 Mbits/s. We do not need

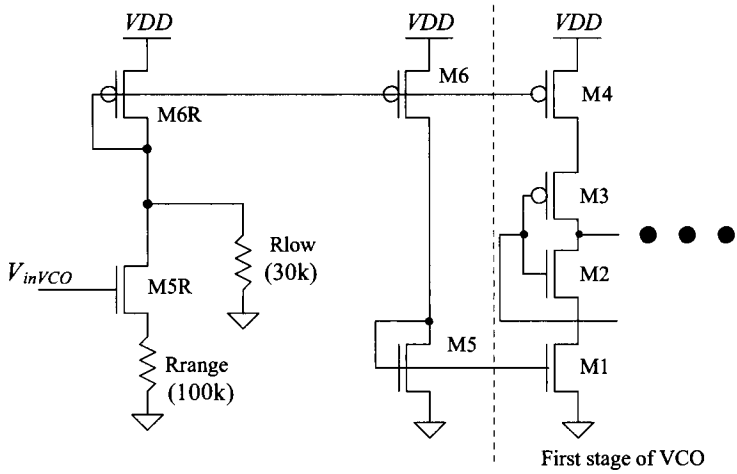


Figure 19.25 Limiting the current in a current-starved VCO.

to calculate the pull-in range since it is greater than the lock-in range. Again, the design of the oscillator is important in this DPLL. The lock time, T_L, of the DPLL is 40 ns. SPICE simulation results of this DPLL are seen in Fig. 19.27. The top waveform is the input control voltage of the VCO. The time that it takes the loop to pull-in the frequency and lock, from this figure, is roughly 250 ns. The lock time would be the time it takes the loop if the input frequency jumped after the loop was locked. Note how, in the bottom traces, the output clock rising edge is centered on the data. Notice that the VCO's control voltage varies by 200 mV. From Fig. 19.26, we can see a 5 MHz variation, Δf_{VCO}, in the VCO's output frequency. To estimate the jitter, Δt_{jitter}, in the output, we can use

$$\Delta t_{jitter} = \frac{1}{f_{center}} - \frac{1}{f_{center} + \Delta f_{VCO}} = \frac{1}{100\ MHz} - \frac{1}{105\ MHz} = 480\ ps \qquad (19.41)$$

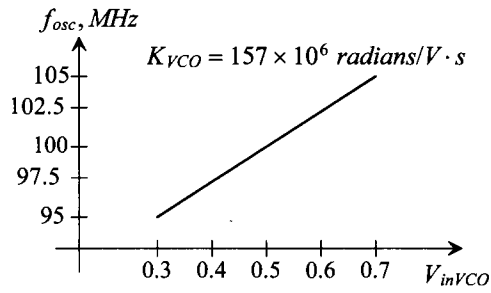


Figure 19.26 Gain of the VCO in Fig. 19.16 after linearizing and limiting with Fig. 19.25.

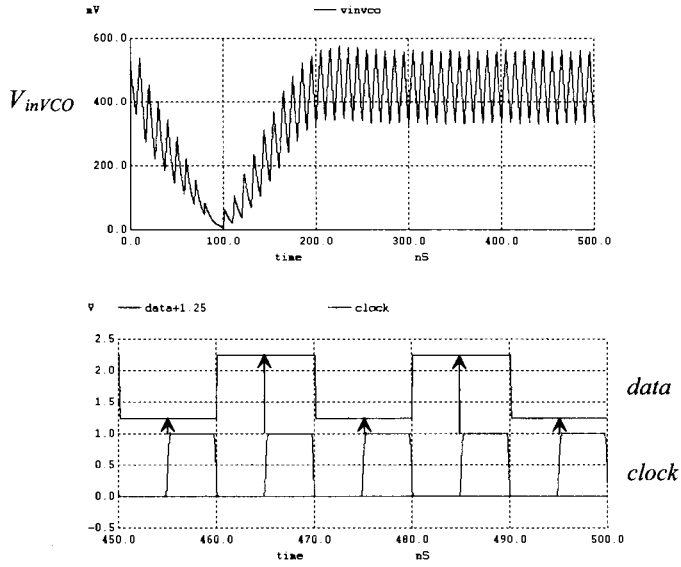


Figure 19.27 Simulating the DPLL in Ex. 19.2.

In this simulation (Fig. 19.27), we used the ideal input data, that is, a string of alternating ones and zeroes. Figure 19.28 shows what happens when the data is a one followed by seven zeroes. The rising edge of the clock is occurring a little offset from the center of the data. This error is sometimes called a *static phase error*. As mentioned earlier, when the input data is not changing, the VCO control voltage starts to wander back towards *VDD*/2.

Finally, *it's very important* to realize that if we were to increase the *RC* time constant of the filter, as seen above, the damping factor, ζ , would decrease and the loop would never lock. This is counter-intuitive and the reason we should *always use the design equations when designing DPLLs.* ■

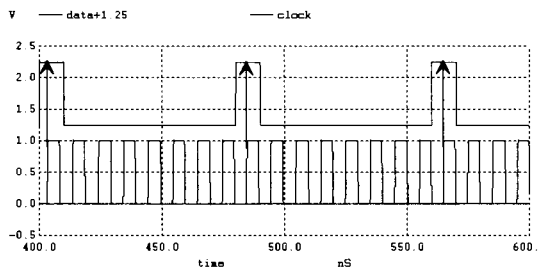
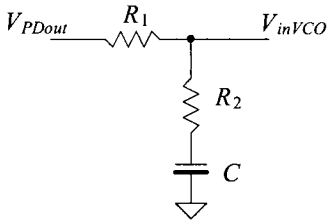


Figure 19.28 Showing how the DPLL doesn't lock up to the center of the data when the data isn't alternating ones and zeroes.

Adding a zero to the simple passive RC loop filter, Fig. 19.29 (called a passive lag loop filter), the loop-filter pole can be made small (and thus the gain of the VCO can be made larger) while at the same time a reasonable damping factor can be achieved. The result is an increase in the lock range of a DPLL using the XOR PD and a shorter lock time see Eqs. (19.37) – (19.40). The passive lag loop filter is, in most situations preferred over the simple RC. Again, as Ex. 19.2 showed, if the center frequency of the VCO doesn't match the input frequency, the clock will not align at $\pi/2$ (in the center of the data).



$$K_F = \frac{V_{inVCO}}{V_{PDout}} = \frac{1 + j\omega R_2 C}{1 + j\omega(R_1 + R_2)C}$$

$$\omega_n = \sqrt{\frac{K_{PD}K_{VCO}}{N(R_1 + R_2)C}} \qquad \Delta\omega_L = \pi\zeta\omega_n$$

$$\zeta = \frac{\omega_n}{2} \cdot \left(R_2 C + \frac{N}{K_{PD}K_{VCO}}\right)$$

Figure 19.29 Passive lag loop filter used to increase DPLL lock range.

Active-PI Loop Filter

The clock misalignment encountered in a DPLL using an XOR PD and passive loop filter can be minimized by using the active proportional + integral (active-PI) loop filter shown in Fig. 19.30. The integrator allows the V_{inVCO} voltage to move, and stay, away from *VDD*/2. This then eliminates the static phase error (the clock will align to the middle of the data). The transfer function of this filter is given by

$$K_F = \frac{1 + sR_2C}{sR_1C} = \underbrace{\frac{R_2}{R_1}}_{\text{proportional}} + \underbrace{\frac{1}{sR_1C}}_{\text{integral}} \tag{19.42}$$

The natural frequency of the resulting second-order system is given by

$$\omega_n = \sqrt{\frac{K_{PD}K_{VCO}}{NR_1C}} \tag{19.43}$$

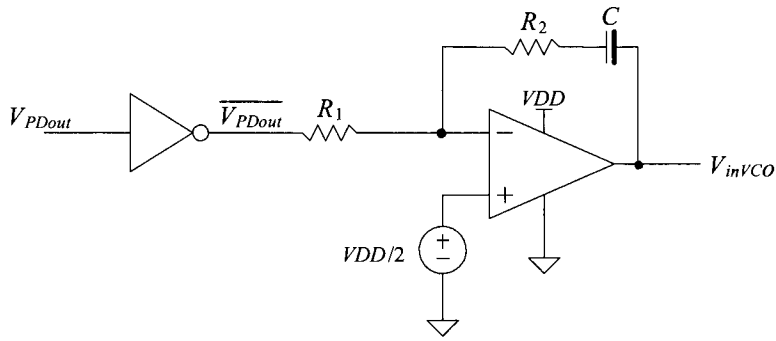


Figure 19.30 Active-PI loop filter.

and the damping ratio is given by

$$\zeta = \frac{\omega_n R_2 C}{2} \tag{19.44}$$

The lock range is

$$\Delta\omega_L = 4\pi\zeta\omega_n \tag{19.45}$$

while the lock time remains $2\pi/\omega_n$. The pull-in range, using the active-PI loop filter, is limited by the VCO oscillator frequency. Consider the following example.

Example 19.3

Repeat Ex. 19.2 using the active-PI loop filter. The SPICE model for an op-amp (a voltage-controlled voltage source) seen in Fig. 20.19 can be used for the op-amp.

Using Eq. (19.43) in (19.44) gives

$$\zeta = \frac{R_2 C}{2}\sqrt{\frac{K_{PD}K_{VCO}}{NR_1 C}}$$

If we set the damping factor again to 1 we can write

$$4 = \frac{157 \times 10^6 R_2^2 C}{2\pi R_1}$$

Setting C to 10 pF and R_2 to 25k, then R_1 is 39k. The simulation results are seen in Fig. 19.31. The output clock is aligned to the center of the data. It's important to make sure that the step size used in the SPICE transient simulation isn't too big. For example, if a 100 MHz clock is used with a step size of 1 ns, it is likely that the loop either won't lock or if it does lock, both the VCO control voltage and thus the output data will oscillate. A maximum step size for 100 MHz signals may be 100 ps. Also note the sparse data pattern will increase the lock time (thus the reason for the long simulation time in Fig. 19.31).

Finally, it is important to remember that the VCO tuning scheme in Fig. 19.25 is not practical unless the components can be set manually (either off-chip or using fuses or switches on chip). Also, something very important that we are not discussing (yet) is power supply noise. The current-starved VCO's output frequency is not very stable with changes in *VDD*. ■

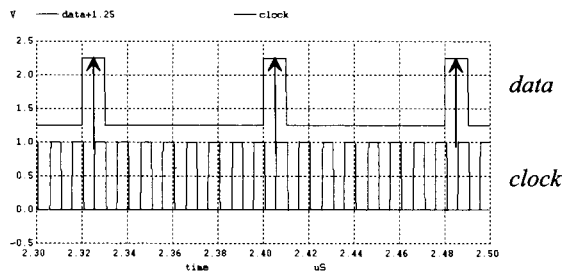


Figure 19.31 How the loop locks up on the center of the data.

19.3.2 PFD DPLL

Tri-State Output

A block diagram of a DPLL using the PFD with tri-state output is shown in Fig. 19.32. The phase transfer function is the same as Eq. (19.30)

$$H(s) = \frac{\phi_{clock}}{\phi_{data}} = \frac{K_{PDtri}K_F K_{VCO}}{s + \beta \cdot K_{PDtri}K_F K_{VCO}} \tag{19.46}$$

where

$$K_F = \frac{1 + sR_2C}{1 + s(R_1 + R_2)C} \tag{19.47}$$

When this filter is driven with the tri-state output, no current flows in R_1 or R_2 with the output in the high impedance state. The voltage across the capacitor remains unchanged. We can think of the filter, tri-state output as an ideal integrator with a transfer function

$$K_F' = \frac{1 + sR_2C}{s(R_1 + R_2)C} \tag{19.48}$$

Substituting this equation into Eq. 19.46 and rearranging results in

$$H(s) = \frac{K_{PDtri}K_{VCO}\frac{1+sR_2C}{(R_1+R_2)C}}{s^2 + s\frac{K_{PDtri}K_{VCO}R_2C}{N(R_1+R_2)C} + \frac{K_{PDtri}K_{VCO}}{N(R_1+R_2)C}} = \frac{\phi_{clock}}{\phi_{data}} = \frac{f_{clock}}{f_{data}} \tag{19.49}$$

From this equation, the natural frequency is

$$\omega_n = \sqrt{\frac{K_{PDtri}K_{VCO}}{N(R_1 + R_2)C}} \tag{19.50}$$

and the damping factor is determined by solving

$$2\zeta\omega_n = \frac{K_{PDtri}K_{VCO}R_2C}{N(R_1 + R_2)C} \tag{19.51}$$

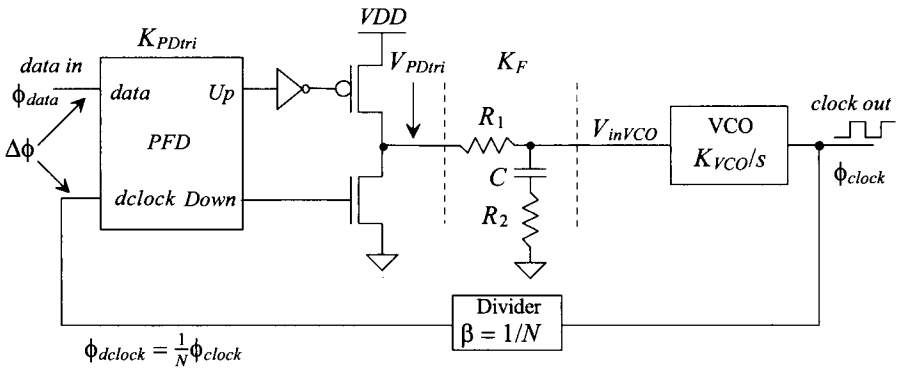


Figure 19.32 Block diagram of a DPLL using a sequential phase detector (PFD).

which results in a damping factor given by

$$\zeta = \frac{\omega_n}{2} \cdot R_2 C \tag{19.52}$$

The lock range is given by

$$\Delta\omega_L = 4\pi\zeta\omega_n \tag{19.53}$$

while the lock time, T_L, remains $2\pi/\omega_n$. The pull-in range is limited by the VCO operating frequency. The pull-in time is given by

$$T_P = 2R_1 C \cdot \ln \frac{(K_{VCO}/N) \cdot (VDD/2)}{(K_{VCO}/N)(VDD/2) - \Delta\omega} \tag{19.54}$$

where $\Delta\omega$ is the magnitude of the *input* frequency step.

Implementing the PFD in CMOS

The PFD seen in Fig. 19.10 can be implemented using inverters and nand gates as seen in Fig. 19.33. Simulating this circuit gives the results seen in Fig. 19.34. When the *dclock* is lagging, the *data* the output of the PFD is the *up* pulse (indicating that the edge of *dclock* needs to speed up or occur earlier in time, that is, the VCO's control voltage should increase). When *data* is lagging *dclock* the *down* pulse goes high indicating *dclock* should slow down.

We have some fine points that need to be discussed here. For example, what happens to *up* and *down* as the rising edges of *dclock* and *data* move close together? What we may get is some small glitches in the *up* and *down* signals. Or we may see both *up* and *down* staying low as the two pulses move closer together. In the PFD of Fig. 19.33, the delay through the two inverters can be used to set how *up* and *down* behave as

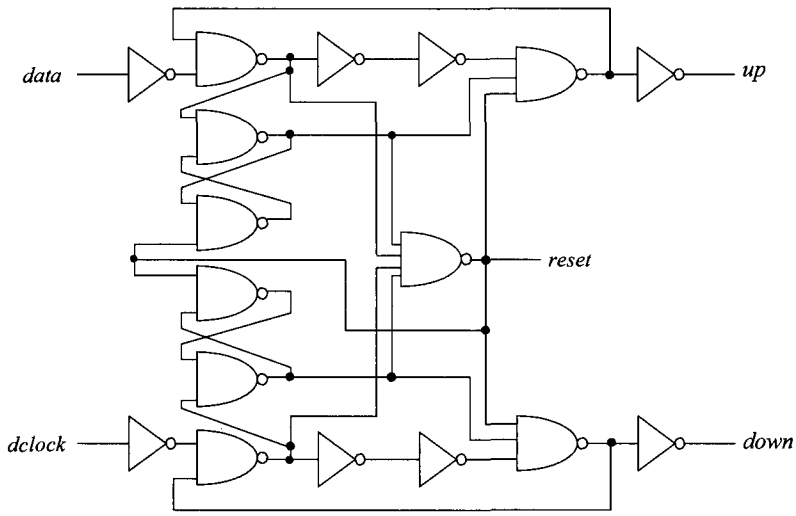


Figure 19.33 CMOS implementation of a PFD.

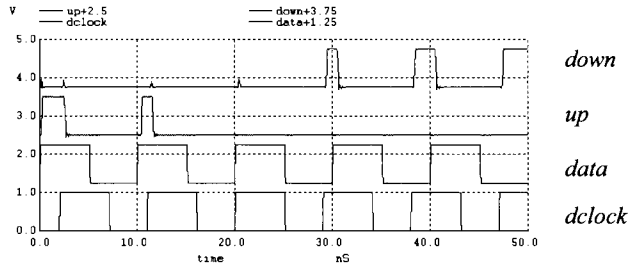


Figure 19.34 Simulation results for the PFD in Fig. 19.33.

the PFD's inputs move close together. If, for example, both stay low when the PFD's inputs get within 100 ps of each other, we get a *static phase error*. The loop won't lock any tighter than within 100 ps of its ideal position. Another way of looking at this is that as the phase difference, $\Delta\phi$, moves towards zero (see Fig. 19.11), the gain (the slope of the curve in Fig. 19.11) decreases. The phase detector is then said to have a *dead zone* where the gain of the phase detector (the slope of the curves) decreases.

Example 19.4

Design a DPLL using the tri-state topology seen in Fig. 19.32 that generates a clock signal at a frequency of 100 MHz from a 50 MHz square wave input. This application of the DPLL is called *frequency synthesis*.

The feedback path contains a divide by 2 circuit ($N = 2$). Let's use the VCO with the characteristics seen in Fig. 19.18

$$K_{VCO} = 1.57 \times 10^9 \; radians/V \cdot s$$

The gain of the phase detector (knowing *VDD* is 1 V) is

$$K_{PDtri} = \frac{1}{4\pi}$$

The lock range, Δf_L will be set to 20 MHz. Again let's set $\zeta = 1$. Using Eq. (19.53)

$$\Delta\omega_L = 4\pi\zeta\omega_n \rightarrow \omega_n = 10 \times 10^6 \; radians/V \cdot s$$

Using Eq. (19.52) gives

$$R_2 C = 200 \; ns$$

Let's set the capacitor to 10 pF and R_2 to 20k. Solving Eq. (19.50) for R_1 gives 42.5k ($= R_1$).

The simulation results are seen in Fig. 19.35. We should first notice that the VCO's control voltage doesn't have the excessive ripple like we had in the DPLL using the XOR gate phase detector. The response of the loop shows a nice $\zeta = 1$ shape. Again note that a DPLL loop's pull-in range is limited by the VCO's operating frequency (which, in this example uses the VCO from Fig. 19.18, which ranges from 50 to 150 MHz). A good exercise to perform at this point is to change the divider in the feedback path (to divide by 1, 2, 4, etc.) and the input frequency (a signal we've called *data*) and look at the robustness of the loop. ■

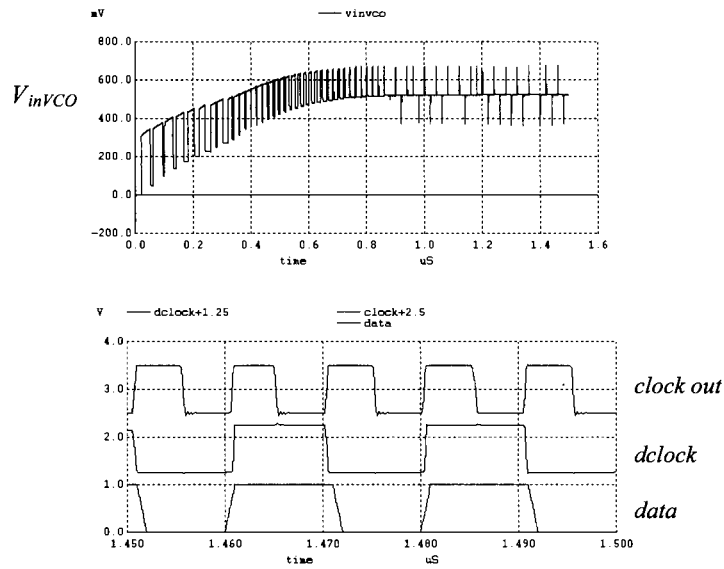


Figure 19.35 Simulation results for Ex. 19.4.

PFD with a Charge Pump Output

The PFD with a charge-pump output is seen in Fig. 19.36. A CMOS implementation of a DPLL using this configuration is, in general, preferred over the tri-state output because of the better immunity to power supply variations. In the tristate configuration seen in Fig. 19.32 note that when either the NMOS or PMOS switches are on, either *VDD* or ground are connected directly to the loop filter. Power supply or ground noise can thus feed directly into the loop filter and then into the VCO's control voltage.

The loop filter integrates the charge supplied by the charge pump. The capacitor C_2 prevents $I_{pump} \cdot R$ from causing voltage jumps on the input of the VCO and thus frequency jumps in the DPLL output. In general, C_2 is set at about one-tenth (or less) of C_1. The loop-filter transfer function neglecting C_2 is given by

$$K_F = \frac{1 + sRC_1}{sC_1} \tag{19.55}$$

The feedback loop transfer function is given by

$$H(s) = \frac{\phi_{clock}}{\phi_{data}} = \frac{K_{PDI}K_{VCO}(1 + sRC_1)}{s^2 + s(\frac{K_{PDI}K_{VCO} \cdot R}{N}) + \frac{K_{PDI}K_{VCO}}{NC_1}} \tag{19.56}$$

From the transfer function the natural frequency is given by

$$\omega_n = \sqrt{\frac{K_{PDI}K_{VCO}}{NC_1}} \tag{19.57}$$

and the damping factor is

$$\zeta = \frac{\omega_n}{2} \cdot RC_1 \tag{19.58}$$

The lock range and lock time remain the same (using the different values for the natural frequency and damping ratio) as the PFD with the tri-state output. Again, the pull-in range is set by the VCO oscillator frequency range. The pull-in time is given by

$$T_P = 2RC_1 \ln\left[\frac{(K_{VCO}/N)\cdot(I_{pump})}{(K_{VCO}/N)\cdot(I_{pump}) - \Delta\omega}\right] \tag{19.59}$$

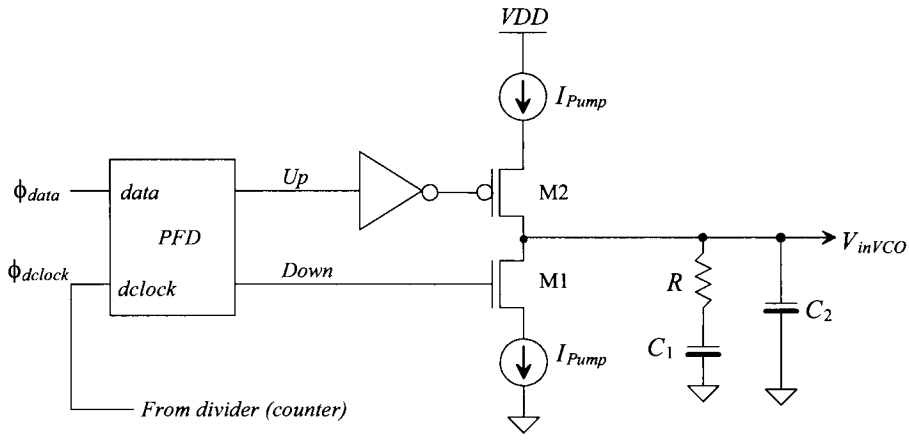


Figure 19.36 PFD using the charge pump.

Practical Implementation of the Charge Pump

The circuit seen in Fig. 19.36 is useful for illustrating the concept of the PFD with charge pump. However, in a practical circuit the fact that the sources of M1 and M2 charge to ground and *VDD* respectively when M1 and M2 are off creates some design issues. For example, suppose that the source of M1 is discharged to ground when the *down* signal goes high. When M1 turns on, it doesn't supply the charge set by I_{pump} to the loop filter but rather, until the voltage across the current sink increases, behaves like a switch simply connecting the filter's input to ground (meaning that we are not controlling the signal that is applied to the loop filter). To get around this problem, consider the circuit seen in Fig. 19.37. The bias voltages come from diode-connected MOSFETs (to form current mirrors, see Ch. 20). When *up* and *down* are low, M1L and M2L are on. The pump-up current source, MPup, is driving the pump-down current source, MNdwn. When either *up* or *down* goes high, the output is connected to one of the current sources. The MOSFETs M1L,R and M2L,R simply steer the current to the loop filter. The only other concern we have with this topology is the fact that when M1L and M2L are both on, the voltage on their drains won't precisely match the voltage across the loop filter (V_{inVCO}). In other words, the voltage across each of the current sources (MPup and MNdwn) won't be the same as it is when they are connected to the output node. The result is that charge sharing between the parasitic capacitance on the drains of MPup and MNdwn and the capacitance used in the loop filter cause a static phase error or jitter. To eliminate this problem, an amplifier (see dashed lines in the figure) can be inserted to set the drain voltage of M1L and M2L to the same value V_{inVCO}.

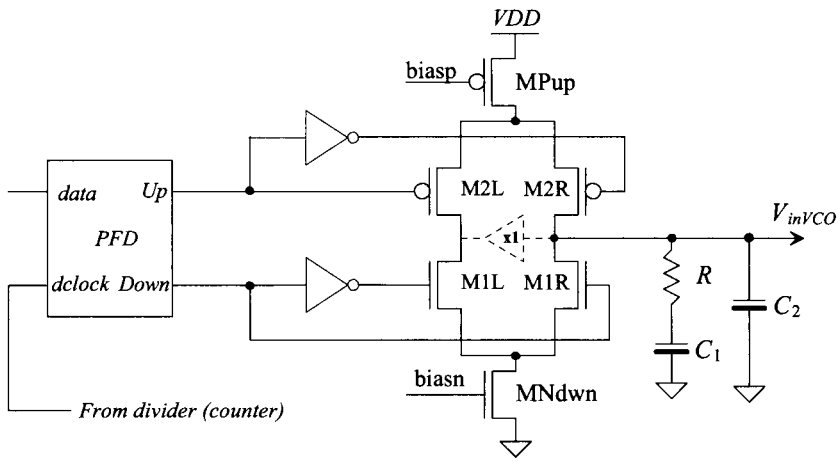


Figure 19.37 Practical implementation of the charge pump.

Example 19.5

Repeat Ex. 19.4 using the PFD and charge pump seen in Fig. 19.37.

One of the other benefits of using the charge pump is the fact that we can select I_{pump}, that is,

$$K_{PDI} = \frac{I_{pump}}{2\pi}$$

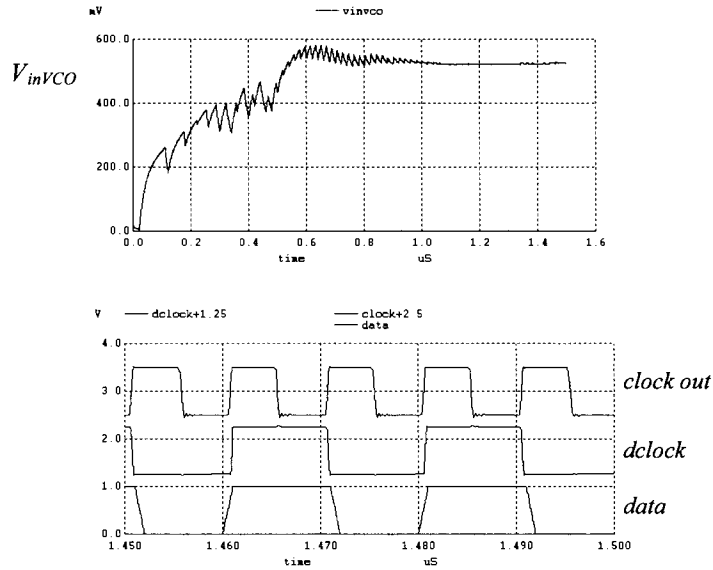
Again, let's set the lock range to 20 MHz. Using Eq. (19.53) with $\zeta = 1$ gives, again, $\omega_n = 10 \times 10^6$ *radians/V* · *s*. From Eq. (19.58) we get

$$RC_1 = 200\ ns \text{ so let's use } R = 20\text{k}, C_1 = 10\text{ pF}, \text{ and } C_2 = 1\text{ pF}$$

remembering that we generally set C_2 to one-tenth of C_1. Using Eq. (19.57), we can now find the value of I_{pump}

$$10 \times 10^6 = \sqrt{\frac{I_{pump} \cdot 1.57 \times 10^9}{2\pi \cdot 2 \cdot 10 \times 10^{-12}}} \rightarrow I_{pump} = 8\ \mu A$$

Because this value isn't that critical, we'll round it up to 10 μA. Figure 19.38 shows the simulation results. Notice the $\zeta = 1$ (or maybe a little less because the voltage does overshoot its final value by a little bit) shape of the VCO's control voltage. Let's modify the loop filter to show what V_{inVCO} would look like if $\zeta = 0.1$. All we have to do, from Eq. (19.58), is drop R by a factor of 10. The result is seen in Fig. 19.39. The control voltage is oscillating and the loop isn't locking. Of course, we can also increase the damping factor. The loop now behaves sluggishly and does not respond to fast changes in the input signal. For general design, where the process and temperature vary, it's better to center the design on a larger damping factor to avoid instability. ■



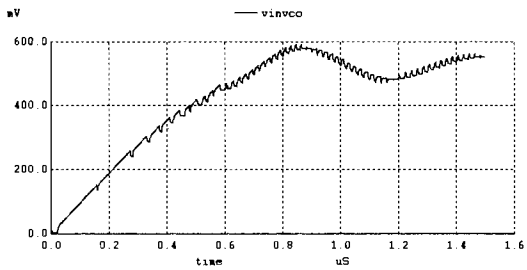
V_{inVCO}

Figure 19.38 Simulation results for Ex. 19.5.

Discussion

When selecting values for the loop filter, we assumed that the output resistance of the phase detector was small (for the XOR and tri-state PD) compared to the impedances used in the loop filter. We also assumed that the input resistance of the VCO was infinite and the input capacitance of the VCO was small compared to the capacitance used in the loop filter. Considering the parasitics present in the DPLL is an important part of the design.

The center frequency of the VCO is critical for good DPLL performance when using the XOR gate with *RC* loop filter. If the center frequency, f_{center}, of the VCO (i.e., $V_{inVCO} = VDD/2$) does not match twice the input data rate, the DPLL will lock up at a phase different from $\pi/2$ (the input frequencies to any phase detector must be equal). The need for a precision center frequency is eliminated by using the XOR PD with an active-PI loop filter or by using the PFD. The other big benefit of using the PFD is that the VCO's gain can be larger.



V_{inVCO}

Figure 19.39 Showing what happens when the damping factor is reduced to 0.1.

Finally, selecting of the loop's damping factor, ζ, is very important. If the value of ζ is too small, the loop will have trouble locking or the output will jitter excessively when the loop is locked. This last problem is sometimes (gratuitously) called *jitter peaking* since a step in the DPLL's input frequency causes excessive overshoot in V_{inVCO} with small ζ.

19.4 System Considerations

System concerns are often the driving force behind the design of a DPLL. Referring to Fig. 19.1, we observe that the data transmitted through the channel should ideally arrive at the receiver with the same shape with which it was transmitted. In reality, the data becomes distorted. Distortion arises from nonlinearities in the receiver input amplifier and the finite bandwidth of the channel. To understand the conditions for distortionless transmission, consider the block diagram shown in Fig. 19.40. The system has a transfer function in the frequency domain of $H(f)$ and in the time domain $h(t)$. For distortionless transmission, we can relate the input and output of the system by

$$y(t) = K \cdot x(t - t_o) \tag{19.60}$$

where t_o is the time delay through the system and K is a constant. This equation shows that for distortionless transmission through a system the output is simply a scaled, time-delayed version of the input. An interesting observation can be made by taking the Fourier Transform of both sides of this equation,

$$Y(f) = K \cdot X(f)e^{-j2\pi f t_o} \tag{19.61}$$

The transfer function of a distortionless system can then be written as

$$H(f) = \frac{Y(f)}{X(f)} = Ke^{-j2\pi f t_o} \tag{19.62}$$

Figure 19.41 shows the magnitude and phase responses of a distortionless system. A system is distortionless when its amplitude response, $|H(f)|$, is a constant, K, and its phase response, $\angle H(f)$, is linear with a slope of $-2\pi t_0$ over all frequencies of interest.

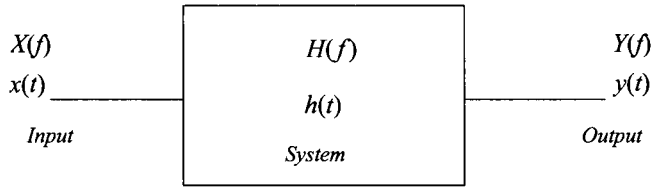


Figure 19.40 Representation of a system with input and output.

The responses shown in Fig. 19.41 are ideal. In practice, the magnitude response of a system may look similar to Fig. 19.42a. At higher frequencies, the magnitude rolls off. To compensate for this roll-off, or other imperfections, a circuit called an equalizer is added in series with the system (Fig. 19.42b). The equalizer has a transfer function in which its magnitude response increases with increasing frequency beyond a point (Fig. 19.42c). If the low-frequency gain of the equalizer is A/K and the low-frequency gain of the system is K, then the resulting gain of the system/equalizer combination is A.

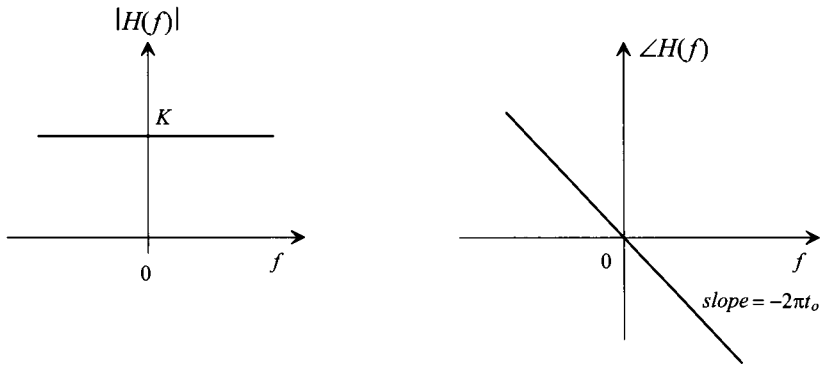


Figure 19.41 Magnitude and phase response of a distortionless system.

Another source of potential distortion occurs when the receiver input data is regenerated into digital levels. This was discussed back in Sec. 18.3. Timing errors occur when the input data is not precisely sliced through its middle (see Fig. 18.12). What makes slicing the data correctly even more difficult is the fact that the amplitude response of the channel can change with time and the data pattern can affect the average level of the data. There are two solutions to this problem. The first uses a circuit (see Fig. 18.29) that determines the peak positive and negative input analog amplitudes, averages the values, and feeds back the result to the comparator in the decision-making circuit. The second method encodes the digital data so that the duty cycle of the resulting encoded data is 50%. The encoding increases the channel bandwidth for a constant data rate.

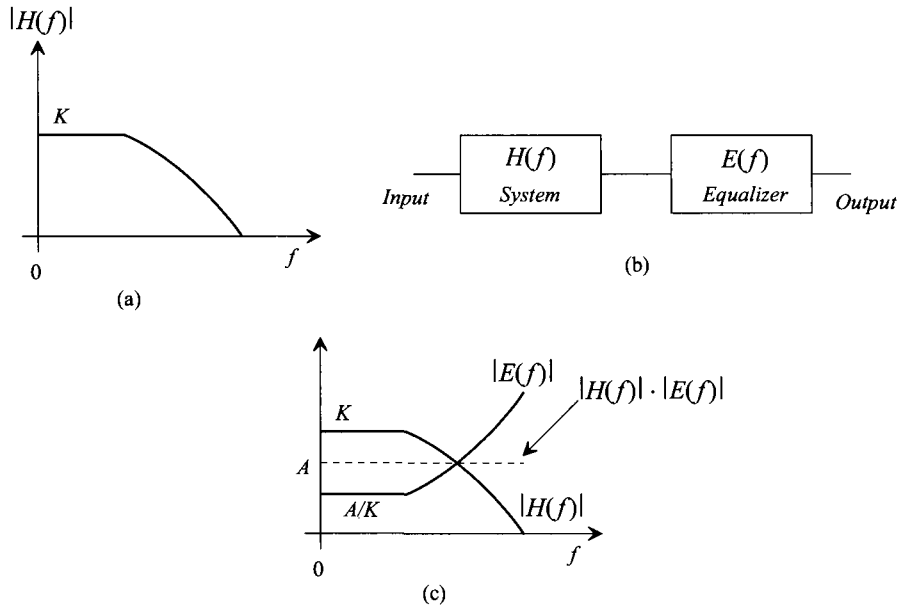


Figure 19.42 Using an equalizer to lower distortion in a system.

Encoding the data can eliminate the need for a decision circuit. If the resulting encoded data has a 50% duty cycle, it can be passed through a capacitor in the receiver, resulting in an analog signal centered around ground. The noninverting input of the comparator will then be connected to ground. The comparator will then be able to slice the data at the correct moments in time (in the middle of the data bit). An example of an encoding scheme is shown in Fig. 19.43. Encoding occurs in the transmitter prior to transmission over the channel. This particular encoding scheme is referred to as the bi-phase format, or more precisely, the bi-phase-level (sometimes called bi-phase-L or Manchester NRZ) format. The cost of using this scheme over the NRZ data format is increased channel bandwidth. Other encoding schemes are shown in Fig. 19.44.

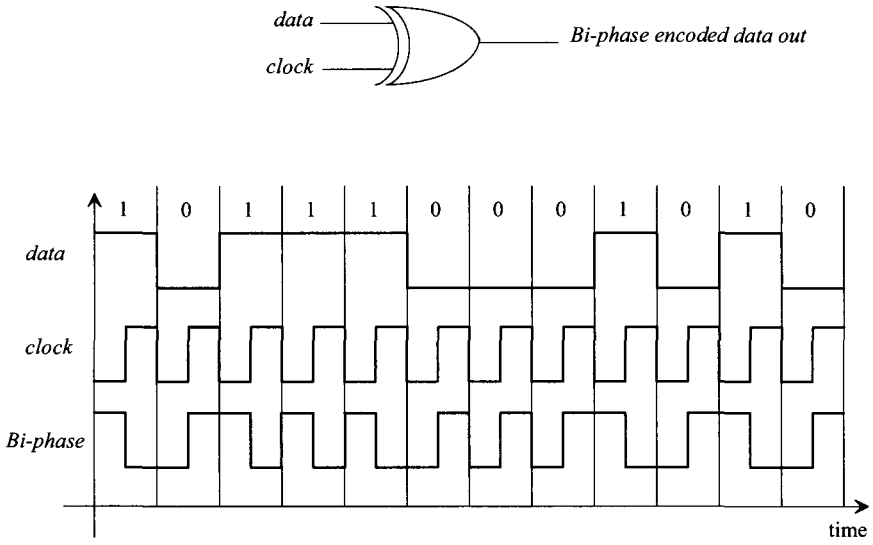


Figure 19.43 Bi-phase data encoding.

19.4.1 Clock Recovery from NRZ Data

One of the most important steps in the design of a communication system is the selection of the transmission format, that is, NRZ, bi-phase, or some other format, together with use of parity, cyclic redundancy code, or some other encoding format. In this section we discuss some of the considerations that go into the design of a clock-recovery DPLL in a system that uses NRZ.

Let's begin this discussion by considering the NRZ *data* and *clock* shown in Fig. 19.45. Let's further assume that these signals are the inputs to an XOR PD in a DPLL, which is not in lock since the clock is not aligned properly to the data. The resulting output of the XOR PD is shown in this figure as well. If we were to average this output using a loop filter, we would get *VDD*/2. In fact, it is easy to show that shifting the *clock* signal in time has no effect on the average output of the PD. Why? To answer this question, let's use some numbers. Assume that a bit width of *data* is 10 ns (which is also the period of the *clock*). The frequency of the square wave resulting from the alternating

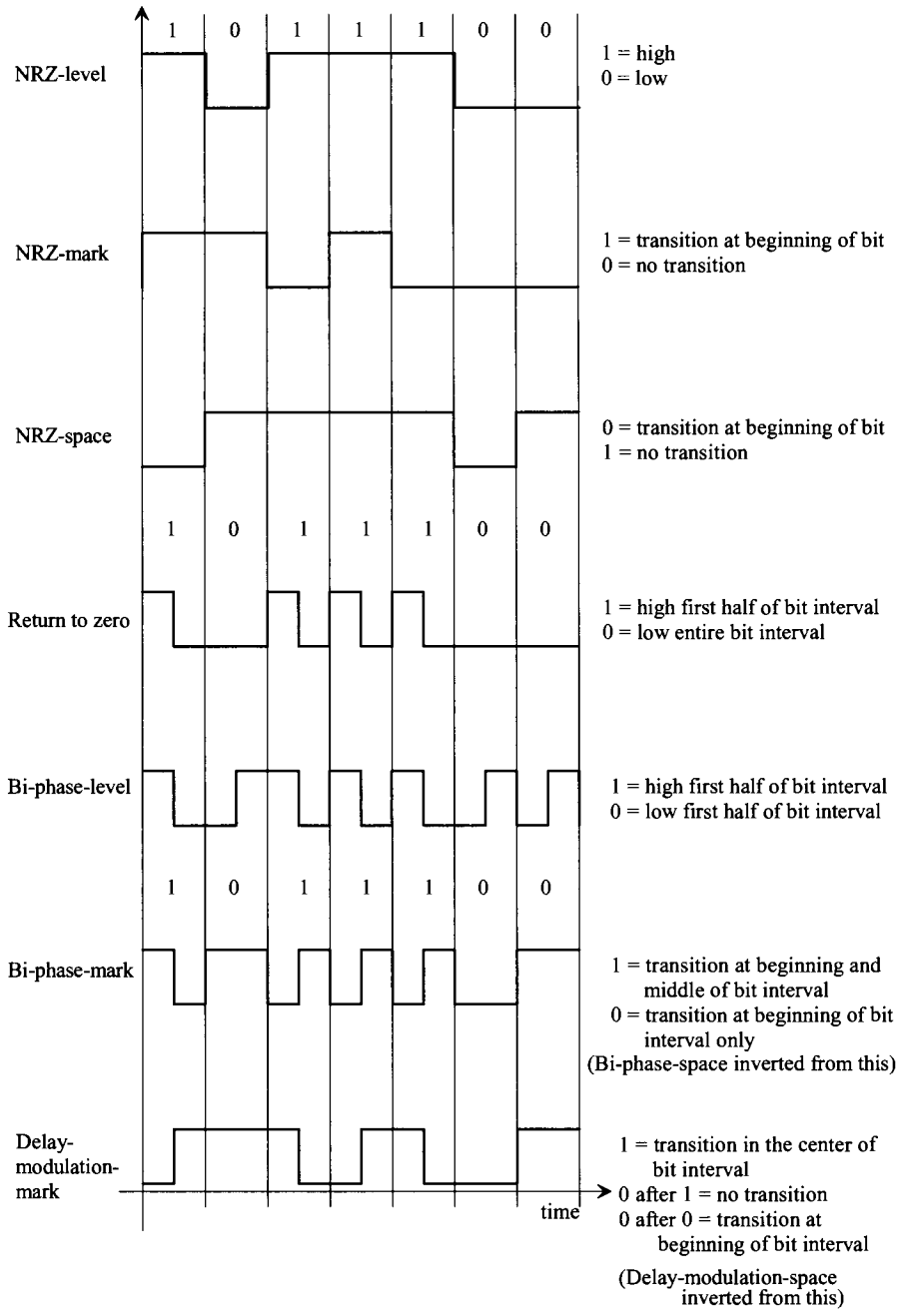


Figure 19.44 Data transmission formats.

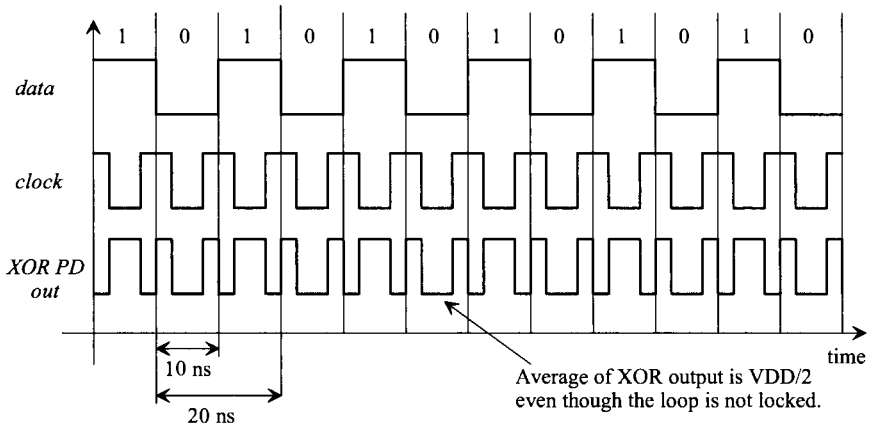


Figure 19.45 The problems of using clock without the divide by 2 to lock on data.

strings of ones and zeros is 50 MHz. We know that if we take the Fourier Transform of a square-wave, only the odd harmonics (i.e., 50, 150, and 250 MHz) are present. Since the *clock* signal is at 100 MHz, there is no energy or information common between the *clock* and *data* signals. To remedy this, we divide the clock down in frequency, *dclock*, so that it is at the same frequency as the alternating ones and zeros of the data (divide by two).

The next problem we encounter, if we use the divide by two in the feedback loop, occurs when we get *data* that is a repeating string of 2 ones followed by 2 zeroes, Fig. 19.46 (the *dclock* is not locked to the data). Again, there is no common information between the two inputs, and the resulting XOR PD output will always have an average of *VDD*/2. In this case, however, the *dclock* is running at 50 MHz, and the *data* is a square-wave of 25 MHz. Should we divide the *clock* down further to avoid this situation? The answer is no. However many times we divide the clock down, we can still come up

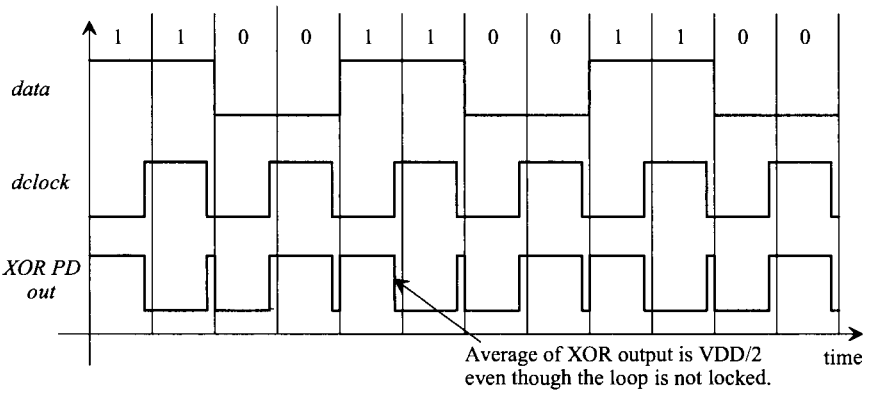


Figure 19.46 Problems trying to lock on a data stream that is one-half the dclock frequency.

with a data string that will not allow the loop to lock. Also, it is the actual edge transitions (the frequency of the *data* and *dclock*) that is used when the inputs are not pure squarewaves. Increasing the width of *dclock* has the effect of removing information and making it more difficult to lock up to the data. One solution to this problem is to use odd-parity with an 8-bit word (9 bits total) and eliminate, at the transmitting end, the possibility of all 8 bits being high, that is, 11111111 or 255. It is then impossible to generate a square-wave.

The restrictions on the data pattern in a communications system using NRZ data can be reduced by detecting the edges of the input data with an edge detector circuit (Fig. 19.47). The delay through the inverters sets the width of the output pulses, labeled "*Edge out*" in this circuit. The frequency content of the output pulses will always contain energy at the *clock* frequency and thus the loop can lock up on the data.

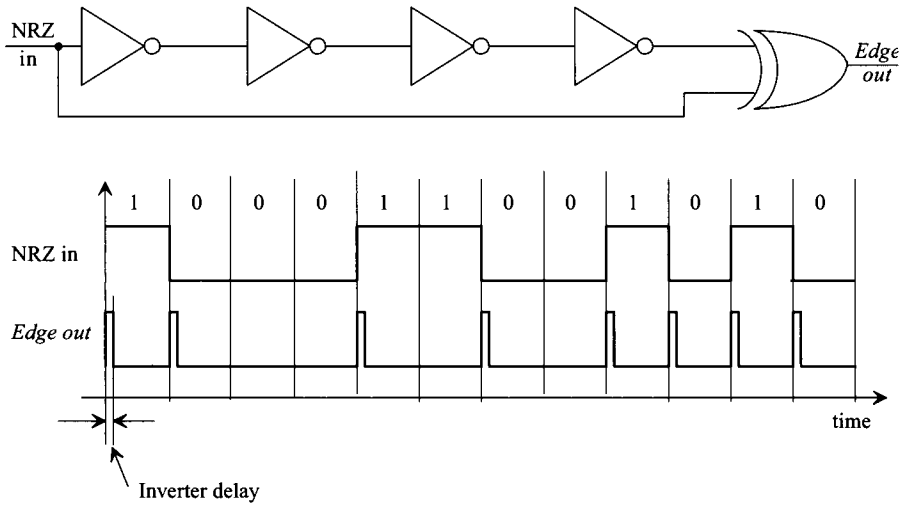


Figure 19.47 Detecting the edges in NRZ data.

As an example, consider the block diagram and data shown in Fig. 19.48. The *Edge output* is connected as the *input* of an XOR-based DPLL. The output of the VCO, *clock*, will lock up on the center of the *Edge output*, that is, the rising edge of the *clock* signal will become aligned with the center of *Edge out*. Averaging *PD out*, in this figure, results in *VDD*/2. If the clock is shifted to the left or right in time, the average value of *PD out* will shift down or up, causing the VCO frequency to change and keep the *clock* aligned to the center of *Edge out*. Several practical problems exist with this configuration. The delay through the inverters should be constant whether a high-to-low or a low-to-high transition is propagating through the inverters. Also, for best performance the delay of the inverters, or whatever element is used for the delay (one common element for high-speed applications is a microstrip line) should be close to one-half of the bit-interval time. This delay is important as it directly affects the gain of the phase detector and therefore the transient properties of the DPLL.

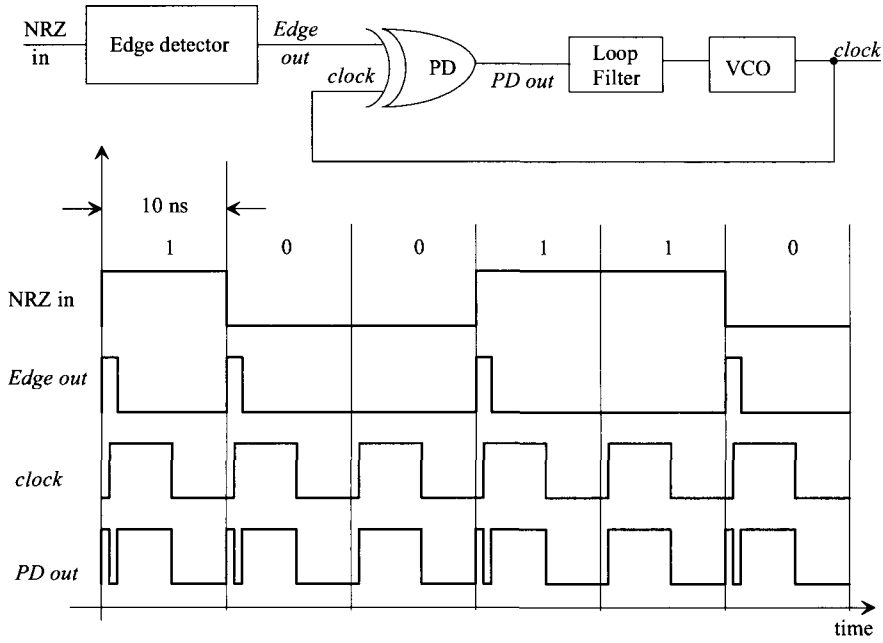


Figure 19.48 Clock-recovery circuit for NRZ using an edge detector. Note that the DPLL is in lock, when the rising edge of clock is centered on the edge output pulse.

Not having the *clock* aligned to the center of the data bit can cause problems in high-speed, clock-recovery circuits. Simply adding a delay in series with the *clock* signal does not solve this problem since the temperature dependence and process variations of the associated circuit do not guarantee proper alignment. What is needed is a circuit that is *self-correcting*, causing the clock signal to align to the center of the data bit independent of the data-rate, the temperature, or process variations.

The Hogge Phase Detector

The PD portion of a self-correcting, clock-recovery circuit is shown in Fig. 19.49 along with associated waveforms for a locked DPLL. Nodes A and B are simply the input NRZ data shifted in time by one-half bit-interval and one bit-interval, respectively. The outputs of the phase detector are labeled *Increase* and *Decrease*. If *Increase* is low more often than *Decrease*, the average voltage out of the loop filter and thus the frequency out of the VCO will decrease. A loop filter that can be used in a self-correcting DPLL is shown in Fig. 19.50a. This filter is the active-PI loop filter discussed in Sec. 19.3.1, with an added input to accommodate both outputs of the PD. Figure 19.50b shows the resulting waveforms in a DPLL where the clock is leading the center of the NRZ data, and thus *Increase* is high less often than *Decrease*. If the NRZ data was lagging the center of the data bit, *Decrease* would be high less often than *Increase,* resulting in an increase in the loop-filter output voltage. Note that in this discussion we have neglected the propagation delays present in the circuit. For a high-speed, self-correcting PD design, we would have to analyze each delay in the PD to determine their effect on the performance of the DPLL.

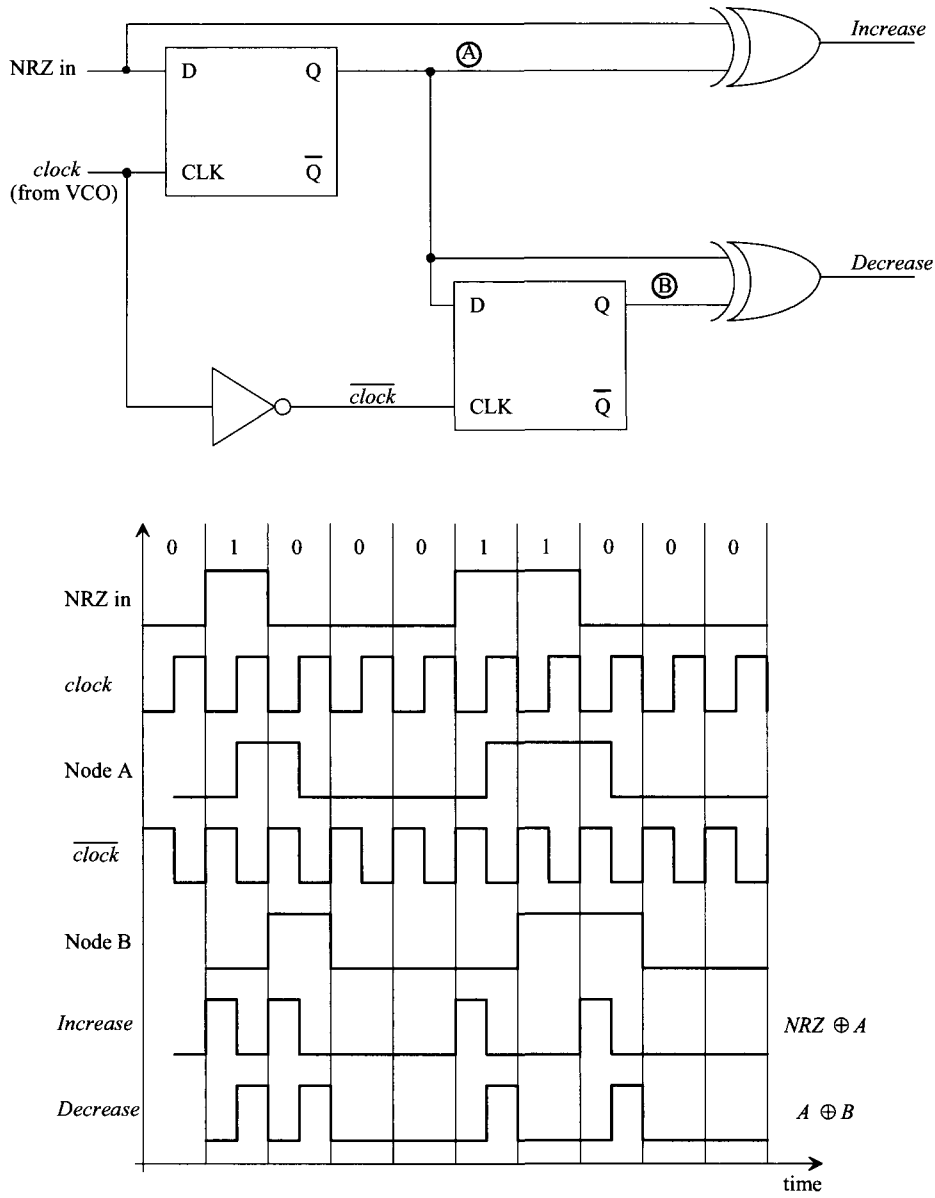


Figure 19.49 The PD (Hogge) portion of a self-correcting, clock-recovery circuit in lock.

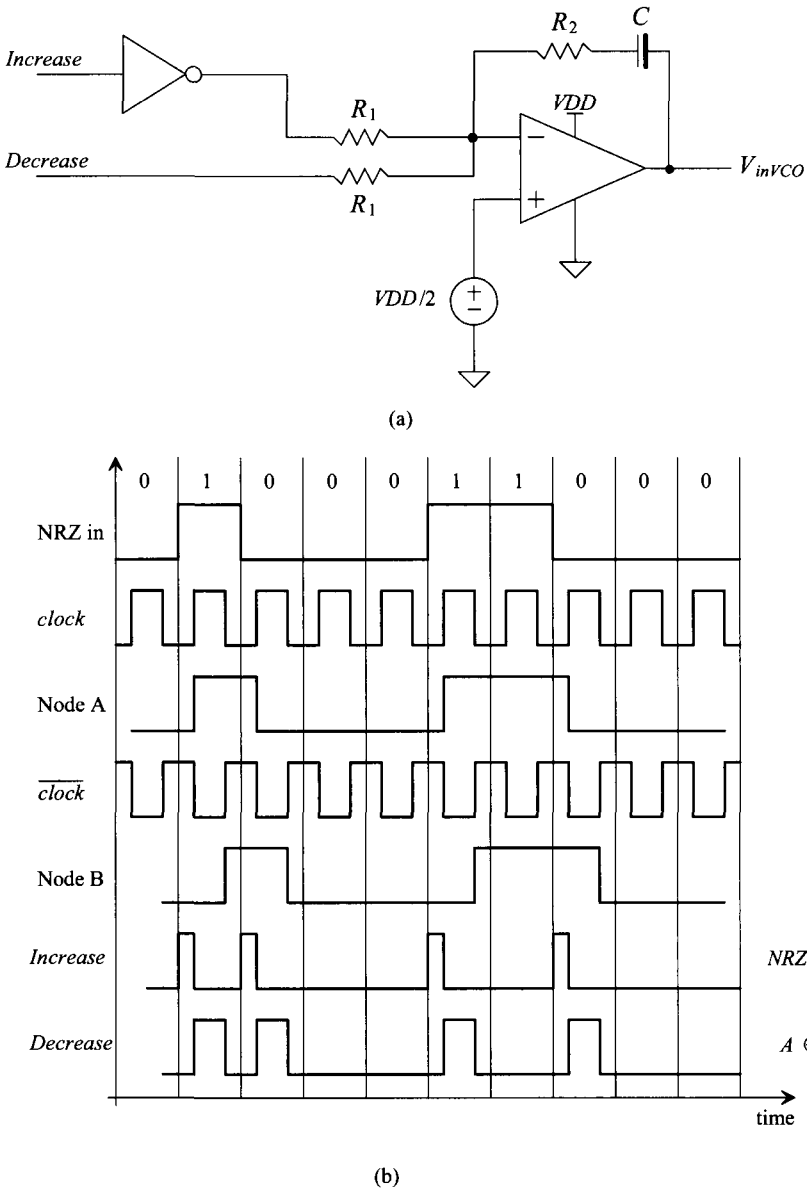


Figure 19.50 (a) Possible loop filter used in a self-correcting (Hogge) DPLL and (b) waveforms when the loop is not in lock and the clock leads the center of the data.

Jitter

Jitter, in the most general sense, for clock-recovery and synchronization circuits, can be defined as the amount of time the regenerated clock varies once the loop is locked. Figure 19.51a shows the idealized case when the *clock* doesn't jitter, while Fig. 19.51b shows the actual situation where the clock-rising edge moves in time (jitters). In these figures, the oscilloscope is triggered by the rising edge of the *data*. In the following discussion, we neglect power supply and oscillator noise, that is, we assume that the oscillator frequency is an exact number that is directly related to the VCO input voltage. In the section following this one, we cover delay-locked loops and further discuss the limitations of the VCO.

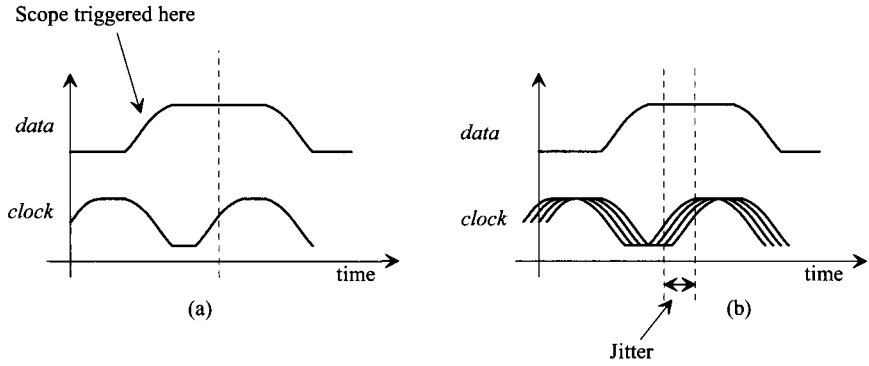


Figure 19.51 (a) Idealized view of clock and data without jitter and (b) with jitter.

Consider using the charge pump with the self-correcting PD shown in Fig. 19.52. When the loop is locked (from Fig. 19.49), both increase and decrease occur (with the same width) for every transition in the incoming data. Note that unlike a PFD/charge pump combination where the output of the charge pump remains unchanged when the loop is locked, the self-correcting PD/charge-pump combination generates a voltage ripple on the input of the VCO (similar to the XOR PD with RC or Active PI filter)[4]. Let's assume that this ripple is 10 mV and use the values for VCO gain and frequencies given in Ex. 19.5 to illustrate the resulting jitter introduced into the output clock. The change in output frequency resulting from this ripple is $10\text{mV}\cdot(1.57 \times 10^9$ radians/V·s)·$(1/2\pi)$ or 2.5 MHz. This means that the output of the DPLL will vary from, say, 100 MHz to 102.5 MHz. In terms of a jitter specification (see Eq. (19.41)), the clock jitter is (roughly) 250 ps (2.5% of the output clock's period).

From this example, *data dependent jitter*, can be reduced by

1. *Reducing the gain of the VCO.* Ripple on the input of the VCO has less of an effect on the output frequency. The main disadvantage, in the most general sense, of reducing the gain of the VCO is that the range of frequencies the DPLL can

[4] Of course, the self-correcting PD should not be used in most frequency synthesis applications. Similarly, the PFD should not be used in most clock-recovery applications.

lock up on is reduced. Also, the VCO gain strongly affects the ability to fabricate the VCO without postproduction tuning.

2. *Reducing the bandwidth of the loop filter.* This has the effect of reducing the actual ripple on the input of the VCO. The main drawback here is the increase in chip real estate needed to realize the larger components used in the filter.

3. *Reducing the gain of the PD.* This also has the effect of reducing the ripple on the input of the VCO. This method is, in general, the easiest when using the charge pump since reducing the gain is a simple matter of reducing I_{pump}. Substrate noise can become more of a factor in the design.

Another way of stating the above methods of reducing jitter is simply to say that the forward loop gain of the DPLL, that is, $K_{PD}K_FK_{VCO}$, should be made small. The main problems with using small forward loop gain are the reductions in lock range and pull-in range coupled with the associated increases in pull-in and lock times.

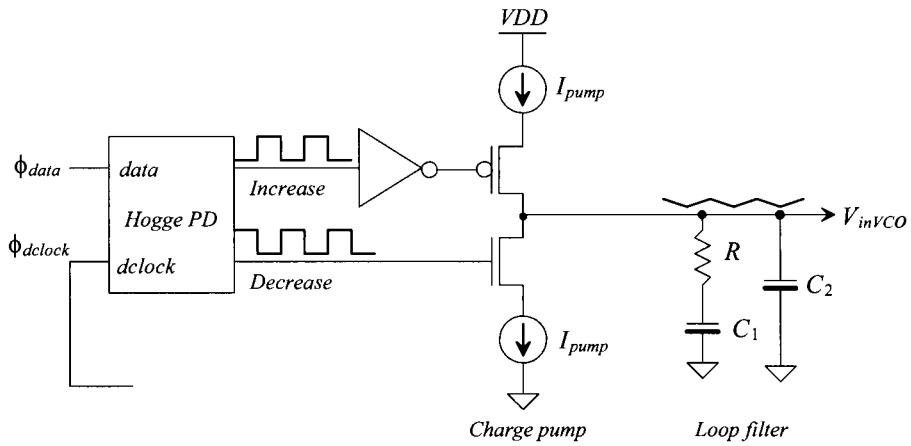


Figure 19.52 Self-correcting PD with charge pump output.

19.5 Delay-Locked Loops

Problems with PLL output jitter resulting from the VCO output frequency changing (often called oscillator or phase noise) with a constant input voltage (V_{inVCO} = constant) has led to the concept of a delay-locked loop (DLL). Figure 19.53 shows the basic block diagram of a DLL. *Assuming* that a reference clock is available at exactly the correct frequency, the input data is delayed through a voltage-controlled delay line (VCDL) a time t_o until it is synchronized with the reference clock. Jitter is reduced by using an element, the VCDL, that does not generate a signal (like the VCO did). The transfer function ϕ_{clock}/ϕ_{out} is zero (the phase of the reference clock is taken as the reference for the other signals in the DLL, i.e., $\phi_{clock} = 0$), so that oscillator noise and the resulting jitter are not factors in DLL design. The jitter considerations discussed in the last section, however, are still a concern since any ripple on the output of the loop filter will cause jitter.

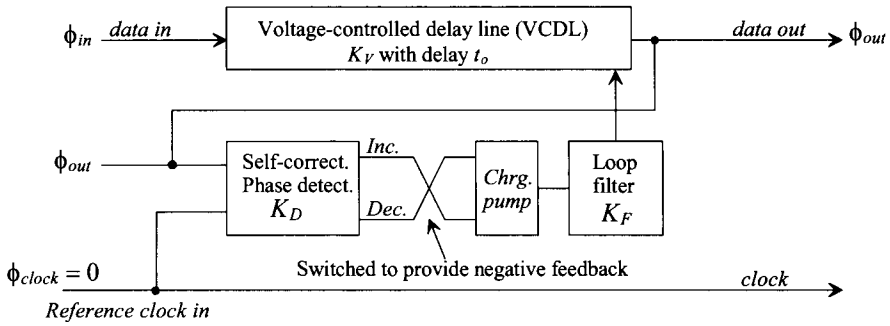


Figure 19.53 Block diagram of a delay-locked loop.

The phase (in radians) of the input data is related to the phase of the output data by

$$\phi_{out} = \phi_{in} + t_o \cdot \frac{2\pi}{T_{clock}} \tag{19.63}$$

where T_{clock} is the period of the reference clock (or half of the period of the *data in* for a string of alternating ones and zeros). The gain of the VCDL can be written in terms of the delay, t_o, by

$$t_o = K_V \cdot V_{indel} \tag{19.64}$$

where K_V has units of seconds/V and V_{indel} is the voltage input to the VCDL from the loop filter. The minimum and maximum delays of the VCDL should, in general, lie between $T_{clock}/2$ and $1.5T_{clock}$ for proper operation. The output of the loop filter (input to the VCDL) can be written as

$$V_{indel} = \phi_{out} \cdot K_D \cdot K_F \tag{19.65}$$

The overall transfer function may now be written as

$$\frac{\phi_{out}}{\phi_{in}} = \frac{1}{1 - K_D K_F K_V \cdot \omega_{clk}} \tag{19.66}$$

where $\omega_{clk} = 2\pi/T_{clock}$. The gain of the self-correcting PD with charge-pump output, with the help of Fig. 19.54 and noting that *Increase* and *Decrease* can occur at the same time, is

$$K_D = -\frac{I_{pump}}{\pi} \text{ (amps/radian)} \tag{19.67}$$

The negative sign is the result of switching the *Increase* and *Decrease* outputs of the self-correcting PD when connecting to the charge pump. This switch is required to provide negative feedback around the loop. Another benefit of the DLL is that the loop filter can be a simple capacitor, which results in a first-order feedback loop, that is,

$$K_F = \frac{1}{sC_1} \tag{19.68}$$

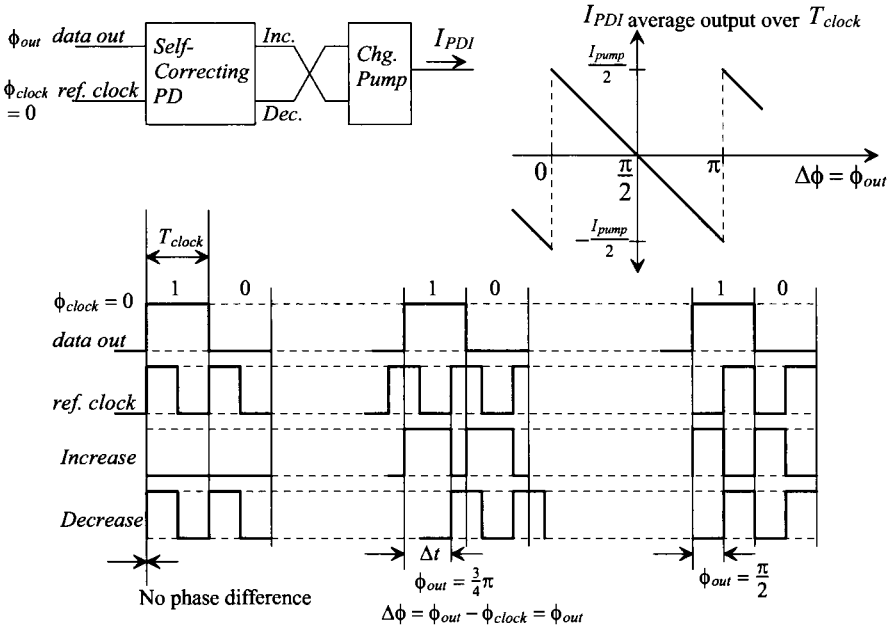


Figure 19.54 Self-correcting PD output for various inputs (assuming input data is a string of alternating ones and zeros).

The transfer function of the DLL relating the input data to the time-shifted output is

$$\frac{\phi_{out}}{\phi_{in}} = \frac{1}{1 + \frac{I_{pump}}{\pi} \cdot \frac{1}{sC_1} \cdot K_V \cdot \omega_{clk}} = \frac{s}{s + K_V \cdot \frac{2I_{pump}}{C_1 T_{clock}}} \tag{19.69}$$

We know that the frequency of the reference clock must be exactly related to the frequency of the input data. However, there will exist instantaneous changes in the phase of the input data which the output of the DLL should follow. Modeling instantaneous changes in ϕ_{in} by $\Delta\phi_{in}/s$ (a step function with an amplitude of $\Delta\phi_{in}$), we get a change in output phase given by

$$\Delta\phi_{out} = \frac{\Delta\phi_{in}}{s + K_V \cdot \frac{2I_{pump}}{C_1 T_{clock}}} \tag{19.70}$$

The time it takes the DLL to respond to an input step in phase is simply

$$T_r = 2.2 \cdot \frac{C_1 T_{clock}}{K_V \cdot 2I_{pump}} = \text{number of clock cycles} \cdot T_{clock} \tag{19.71}$$

This time can be decreased by making C_1/I_{pump} small, which from our discussion in the last section, has the result of increasing the output pulse jitter (i.e., jitter dependent on the input data pattern). Decreasing C_1/I_{pump}, increases the ripple on the control voltage of the VCDL. Similarly, increasing K_V (the time/volt delay of the VCDL) increases jitter since a given ripple on the control voltage of the VCDL will have a larger effect on the delay. Again, trade-offs must be made between responses to input variations and output jitter.

Delay Elements

The VCDL is an important component of the DLL. Figure 19.55a shows the basic implementation of a VCDL using adjustable delay inverters. The last two inverters in the VCDL ensure that clean digital signals are output from the line. Figure 19.55b shows the circuit schematics for possible delay elements. The first delay element should be recognized as a current-starved inverter discussed earlier in the chapter. The second delay element is nothing more than an inverter with a variable load. In practice, these delay elements are rarely used because of the susceptibility to noise and power supply variations. Rather, fully-differential delay elements are used.

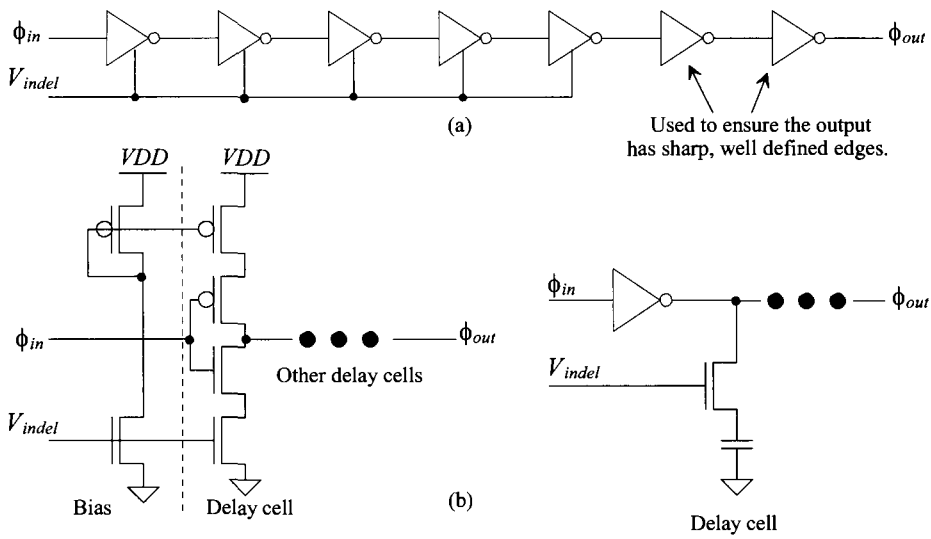


Figure 19.55 (a) VCDL made using inverter delay cells and (b) possible delay cells.

Figure 19.56 shows the connection of a fully differential VCDL. Using any number of stages, a fully differential VCO can be implemented with the delay elements (and the proper feedback), while using an even number with the inverting (noninverting) output fed back and connected to the noninverting (inverting) input in-phase (I) and quadrature (Q) signals can be generated.

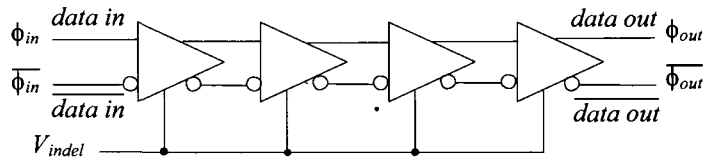


Figure 19.56 Implementation of a fully differential VCDL.

Practical VCO and VCDL Design

An example of a practical delay stage is shown in Fig. 19.57. The control voltage (that controls the delay if used in a VCDL or the oscillation frequency if the element is used in a VCO) can be generated from some linear voltage to current converter (like the one seen in Fig. 19.17 using a resistor, M5R, and M6R). The reason that this circuit is labeled "practical" can be understood by first considering what happens when there is noise on *VDD*. The noise causes a voltage variation across the PMOS current source, MPC. Ideally, this doesn't change the current through MPC. The output voltages of the delay cell swing between V_{REF} and ground. If there is noise on ground, it feeds equally into each output. This common-mode noise is then, ideally, rejected by the differential amplification action of the next stage. The bias circuit is a half-replica of the delay stage and is used to bias the delay elements so that the swing is up to V_{REF} when the gate of one of the PMOS switches is at ground.

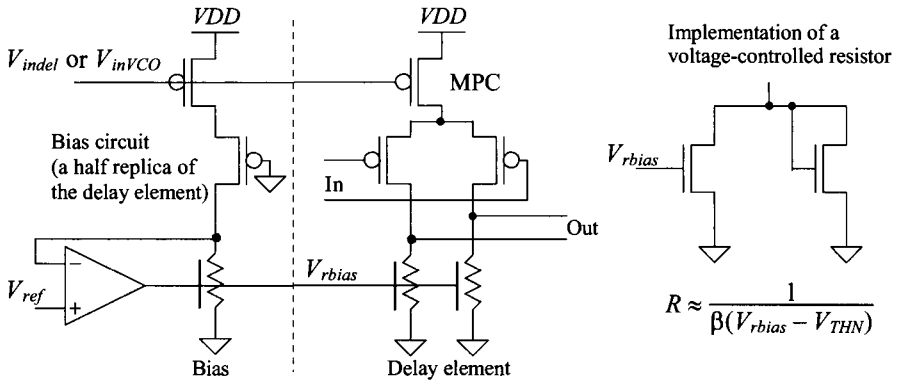


Figure 19.57 A differential delay element based on a voltage-controlled resistor. The bias circuit adjusts the value of the resistors used in the delay elements to sink the current sourced by the p-channel MOSFETs.

19.6 Some Examples

To finish up this chapter, let's present some examples with discussions and simulations.

19.6.1 A 2 GHz DLL

To begin, let's design a 500 ps delay line for a DLL that is used for *clock synchronization*. Let's design the delay line so that when it is used in a DLL, with the input and output of the delay line synchronized, we have eight phases of the input clock (we need eight stages in the delay line).

The first circuits we need to design are the bias circuit and the voltage-to-current converter used to generate $V_{outfilter}$ and V_{rbias} (see Fig. 19.57). We'll use the basic schemes of Fig. 19.17 and 19.57, as seen in Fig. 19.58. The important issues concerning this circuit are: 1) the linearity of the V_{indel} voltage against the generated I_{REF}, 2) the sensitivity of I_{REF} to changes in *VDD*, and 3) how well the amplifier regulates node n2 to V_{REF}.

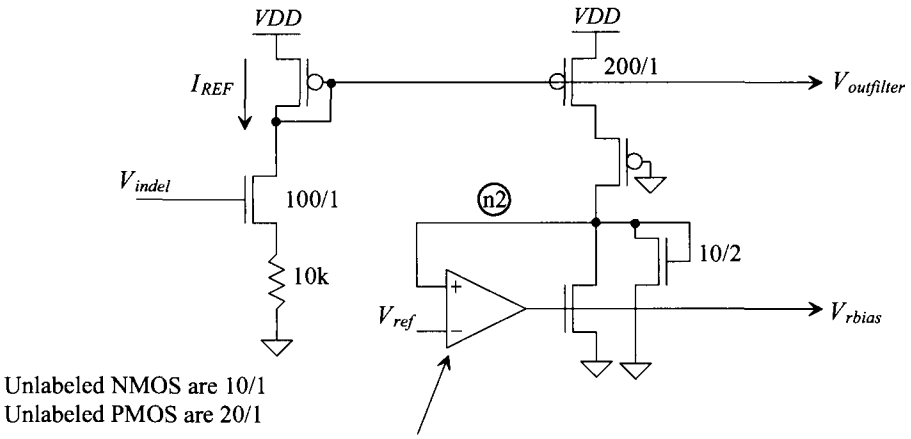


Figure 19.58 Bias circuit for a 500 ps delay line.

Figure 19.59 shows the simulation results for the bias circuit in Fig. 19.58. In (a) the x-axis is the delay line's control voltage, V_{indel} , while the y-axis is the generated reference current. Also seen in the figure, although hard to discern, is the change in *VDD* from 900 mV to 1.1 V (*VDD* is changed from 900 mV to 1.1 V in 50 mV steps, while V_{indel} is swept from 300 mV to 800 mV). Figure 19.59b shows how well node n2 is regulated to V_{REF}. V_{REF} can be generated with the beta-multiplier discussed in the next chapter (see Eq. (20.38) and Fig. 20.22).

Note that we aren't setting a lower value of current in this circuit like we did in Fig. 19.25. If we were to do this (which may be useful to reduce jitter since the resulting delay line or VCO would have lower gain), we would use a topology like the one seen in Fig. 19.60. The reference voltage is used to set the minimum current (when V_{indel} is small) through Rlow. The problem with setting the lower current with the method seen in Fig. 19.25 is that any changes in *VDD* feed directly across Rlow and change the bias current. Using the method in Fig. 19.60 with wide NMOS devices, the lower current is (roughly) $(V_{REF} - V_{THN})$/Rlow.

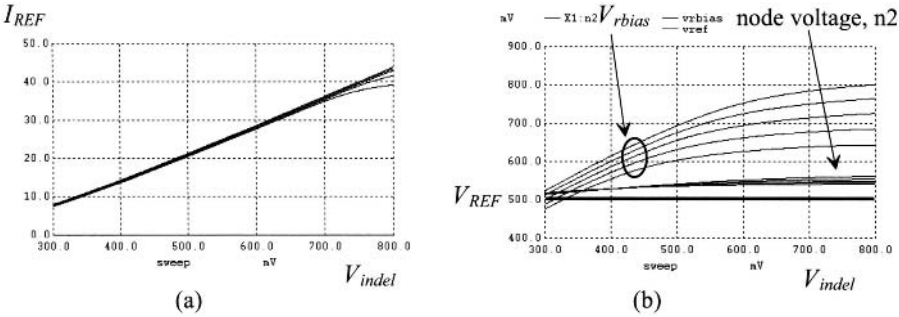


Figure 19.59 The performance of the bias circuit in Fig. 19.58.

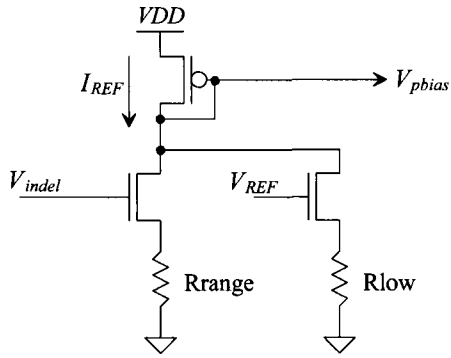


Figure 19.60 Lowering the current range in the bias circuit.

A schematic of the delay line is seen in Fig. 19.61. Notice that the current in the each delay stage was sized up by a factor of ten (the current sources in the delay cell are sized 200/1). This VCDL was simulated using the topology seen in Fig. 19.62. The control voltage was set to *VDD*/2 while the differential inputs were generated using an inverter and a transmission gate (to attempt to equalize the delays that the input signal sees to the VCDL). The outputs of the even stages are shown in this figure. Notice how they are evenly spaced. Notice, also, that the outputs only swing up to V_{REF} (= 500 mV here). As discussed earlier, this was done to minimize the effects of power supply and ground noise on the delay of the circuit.

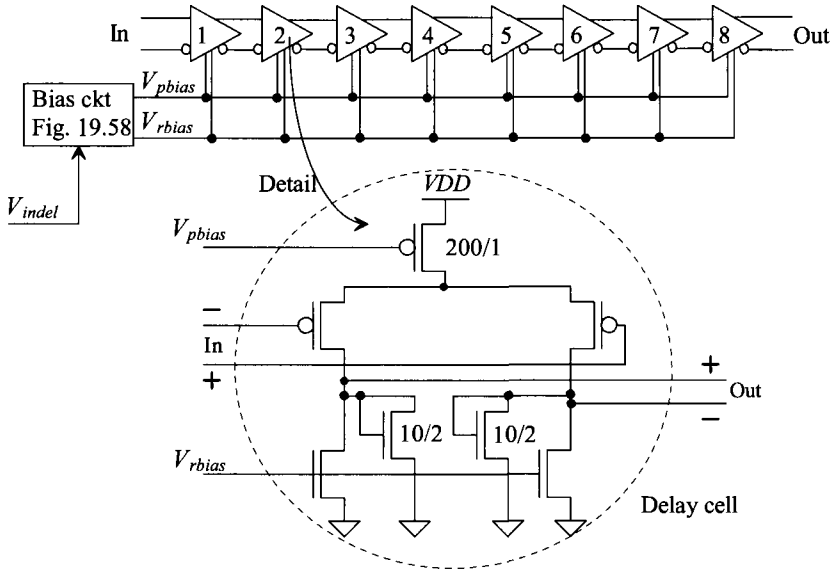


Figure 19.61 An eight-stage VCDL.

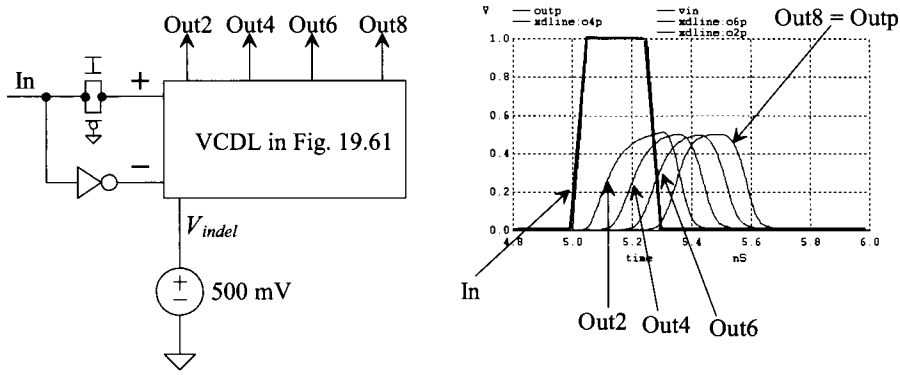


Figure 19.62 Simulating the VCDL in Fig. 19.61.

An important concern is how we regenerate full logic levels without introducing skew into our signals (see Sec. 18.3). Consider the modified VCDL seen in Fig. 19.63. Here we've changed the last stage to two diff-amps with swapped input signals (so we can generate the positive and minus outputs). Figure 19.64 shows the simulation results with this change. The shift in the eighth stage's output is small, in (a), and not much different than what is seen in Fig. 19.62. However, as needed, the output amplitude is larger. To get full logic levels, we pass these signals through inverters (that add more delay as seen Fig. 19.64b).

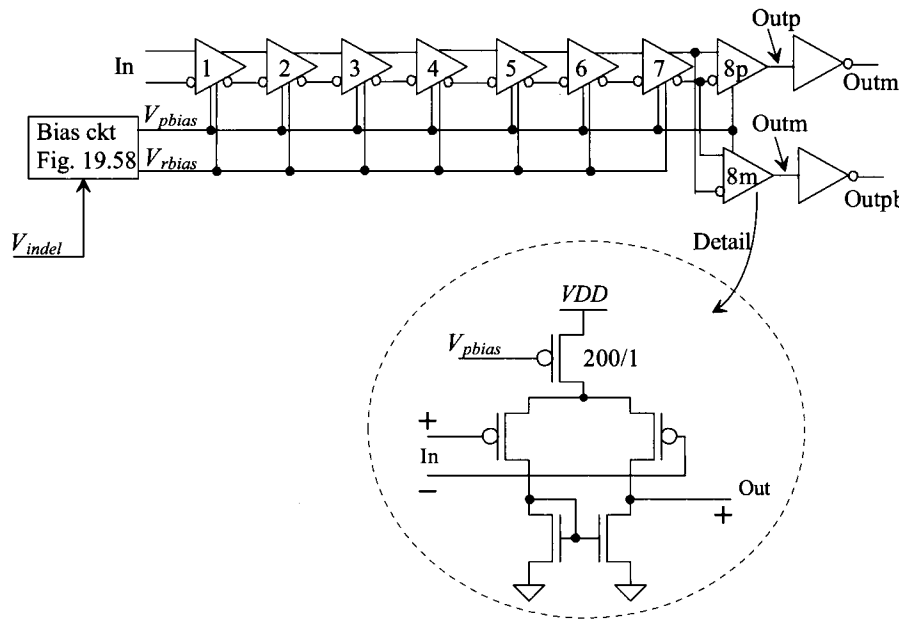


Figure 19.63 Modifying the VCDL to generate full output logic levels.

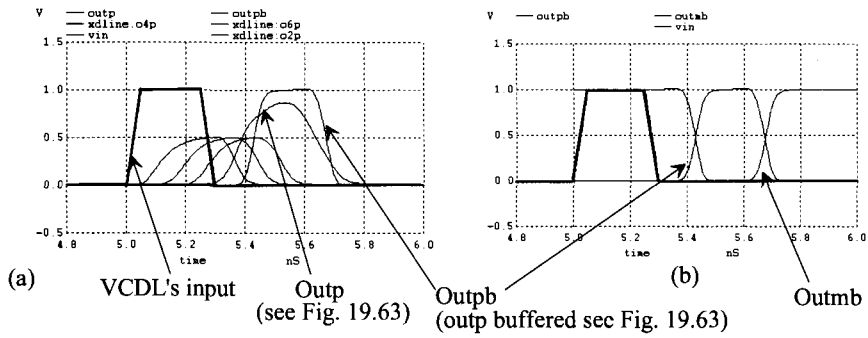


Figure 19.64 The operation of the VCDL in Fig. 19.63.

Using the simulations that generated Fig. 19.64, we can get a reasonable estimate for the VCDL's gain as

$$K_V = \frac{75\,ps}{100\,mV} = 750\,\frac{ps}{V}$$

The minimum delay through the VCDL is roughly 300 ps, while the maximum delay is infinite when V_{indel} moves towards V_{THN}. As seen in Fig. 19.60, we can set the minimum current that flows, I_{REF}, and thus the maximum delay. While this is an important concern we won't address it any further here.

For the PD in our DLL, let's use the PFD detector and charge pump from Ex. 19.5. The PFD's gain can be written as

$$K_D = \frac{I_{pump}}{2\pi} = \frac{10\,\mu A}{2\pi} = 1.59 \times 10^{-6}\ amps/radian$$

Using Eq. (19.71) with a lock time of 50 cycles gives

$$C_1 = \frac{(750 \times 10^{-12}) \cdot 2 \cdot (10 \times 10^{-6}) \cdot 50}{2.2} = 340\,fF$$

As indicated in the discussion following Eq. (19.71), however, this capacitor's selection is based, in some designs, not on lock time but rather the allowable jitter in the output signal once the loop is locked. A block diagram of our DLL is seen in Fig. 19.65. Simulations show that, indeed, the loop locks within 50 cycles with this value of loop filter capacitor (340 *fF*). However, the ripple on the control voltage is 20 mV. With the VCDL gain given above, we can estimate the output jitter as 15 ps. Since the period is 500 ps, this is 3% of the period. This much jitter, in a general application, may not be too bad. When we derived Eq. (19.71), we assumed that the gain of the VCDL was linear. If it is not linear the DLL can exhibit some second-order locking effects (meaning the response isn't truly first-order), resulting in a static phase error or a dead zone (the loop doesn't lock tightly to the input signal). To reduce the jitter and to linearize the loop's response, let's use a large capacitor for the loop filter (a 5 pF) in this example.

The simulated input and output signals of the DLL in Fig. 19.65 using a 5 pF capacitor are seen in Fig. 19.66. Notice the static phase error. The center of the rising edge of the input signal isn't exactly occurring at the same time as the center of the rising edge of the output signal. Further, notice the asymmetry in the output signal, which stays

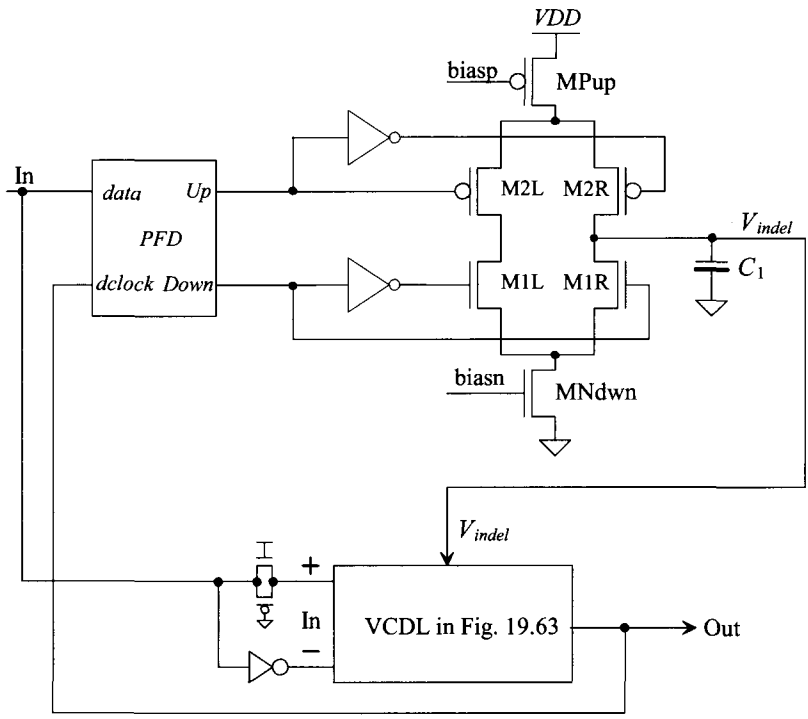


Figure 19.65 Block diagram of a DLL.

high longer than it stays low. This skew, as discussed in Sec. 18.3, is caused by different drive strengths of NMOS and PMOS devices or by differing slopes of input signals. We used the half-replica bias circuit in Fig. 19.58 to make the delay line more tolerant of power supply and ground noise. The drawback is that, at the end of the line, we do have to generate full logic levels. This causes skew between the final output of the VCDL (stage 8 in Fig. 19.63) and the outputs of the other stages in the line. The simple solution

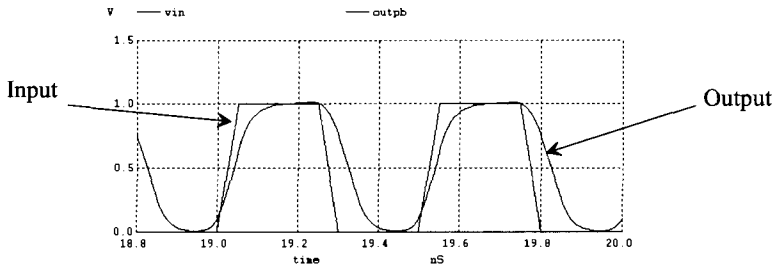


Figure 19.66 Input and output of the DLL in Fig. 19.65.

to this problem is to use a delay stage where the outputs swing to full logic levels. To reduce the DLL's susceptability to power supply and ground noise, we can power it with its own on-chip regulated power supply, supply power and ground to the DLL from separate *VDD* and ground pads, or use some simple filtering to ensure the noise on *VDD* is slow enough that the DLL can respond and correct for variations in the VCDL's delay circuits. For example, we might, in Fig. 19.61, reduce the number of stages to four and add, on the output of each of the delay cells seen in this figure, two of the PMOS buffers seen in Fig. 18.21. We need two buffers because we swap the inputs of each buffer to generate both true and compelement output logic levels. An example of this delay cell is seen in Fig. 19.67. We only use four stages now (for the 500 ps delay line in this section) because of the extra capacitive loading on the output of each stage. The buffered delay cell outputs drive an external load while the basic delay cell outputs are used to drive the input of the next delay cell in the VCDL (only).

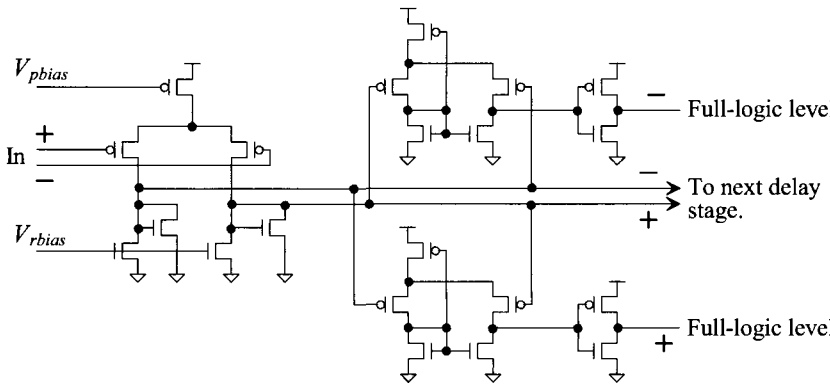


Figure 19.67 Generating full-logic levels in each delay stage.

19.6.2 A 1 Gbit/s Clock-Recovery Circuit

As another example, let's discuss and simulate the design of a 1 Gbit/s clock recovery circuit using the NRZ data format. Let's use the VCDL we've already developed in the VCO (see Fig. 19.68). The output of the VCDL is fed back to its input (with an inversion) to get the positive feedback needed for oscillations. Note that the VCO oscillates at 1 GHz when V_{inVCO} is (roughly) 350 mV (which is not *VDD*/2). While, in a production part, centering the VCO's operating frequency is important, we don't modify the VCDL in Fig. 19.63 for the PLL design here.

We will be using the Hogge phase detector in this design. Since the Hogge PD doesn't perform frequency detection (like the PFD), we must be concerned with locking on harmonics. For example, we might have an input NRZ data stream of 1000 0000. Our loop locks with the clock centered on the one and then six other edges centered on the seven zeroes. The loop is locked just not at the right clock frequency (it's locked at 7/8 of the correct frequency). While this is an **important practical** concern, we won't discuss it further until the end of the section. We will use initial conditions on the loop filter to set the initial clock frequency close to the correct value to avoid locking on a harmonic.

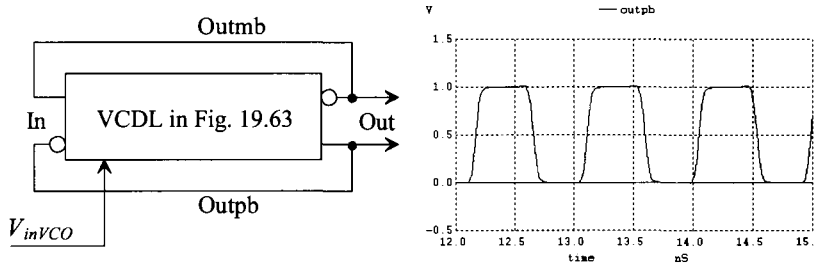


Figure 19.68 Making a VCO with the delay line in Fig. 19.63 and its output when Vinvco = 350 mV.

We know that the gain of the VCDL is 750 ps/V. Since, for one complete oscillation of the VCO, the signal must travel through the VCDL twice, we can estimate the gain of the VCO as

$$K_{VCO} = 2\pi \cdot \frac{1.175\,GHz - 1\,GHz}{100\,mV} = 11 \times 10^9\ radians/V \cdot s$$

where we assume at 350 mV the output frequency is 1 GHz (period is 1 ns). If we increase V_{inVCO} by 100 mV, we get a decrease in the period by 150 ps ($2 \times 100\,mV \cdot 750\,ps/V$) resulting in an output frequency of 1.175 GHz (period is 850 ps).

To implement the Hogge PD seen in Fig. 19.49, we'll use the same TSPC edge-triggered latch that we used for the divide-by-2 seen in Fig. 19.24. The result is seen in Fig. 19.69. We've been very careful to buffer the inputs and outputs of the PD to ensure high-speed output edges and to square up the input signals. To verify the operation, we've simulated the PD with the NRZ and clock signals seen in Fig. 19.49, Fig. 19.70. Looking at the *increase* and *decrease* signals, we see that the width of the *increase* signal is too large. On careful inspection, the cause of this error is the delay through the DFF the NRZ data sees to node A. To compensate for this delay, we'll add a delay in parallel with this path to the XOR gate, as seen in Fig. 19.71. Figure 19.72 shows the resulting simulation output. Again note that unlike the PFD, which only looks at the edges, the Hogge PD, like the XOR PD, uses the widths of its inputs to drive the loop filter.

When the Hogge PD is used with a charge pump the gain is, see Eq. (19.67),

$$K_D = \frac{I_{pump}}{\pi}$$

If we use the charge pump from the DLL in Fig. 19.65 with I_{pump} = 10 μA, the gain of the PD is 3.2 μA/radian. If we set the natural frequency, ω_n, to, 100×10^6 radians/s and ζ = 1, then using Eq. (19.58)

$$RC_1 = 20\,ns$$

and using Eq. (19.57) we can solve for C_1 (knowing that there isn't a feedback divider so N is 1) as

$$C_1 = \frac{K_D K_{VCO}}{\omega_n^2} = \frac{(3.2 \times 10^{-6})(11 \times 10^9)}{(100 \times 10^6)^2} = 3.5\,pF$$

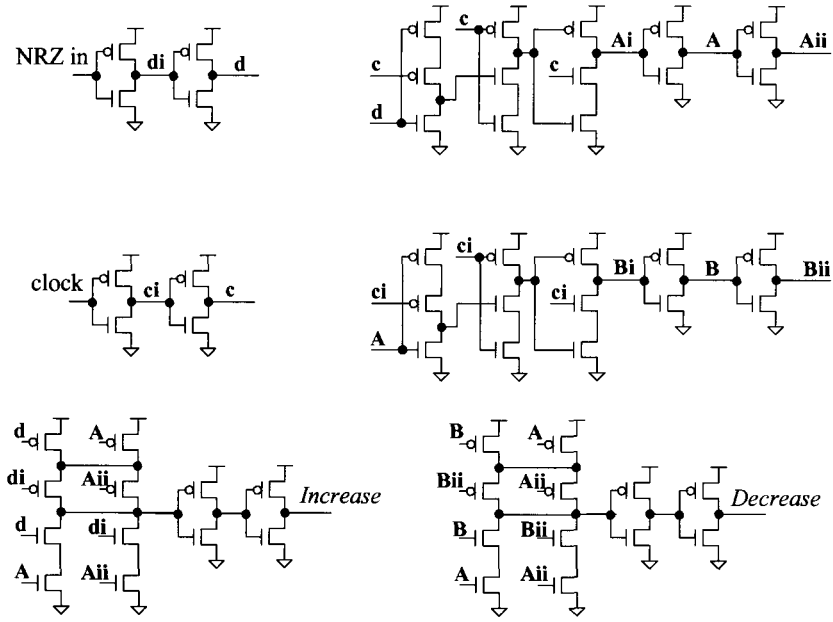


Figure 19.69 CMOS implementation of the Hogge PD.

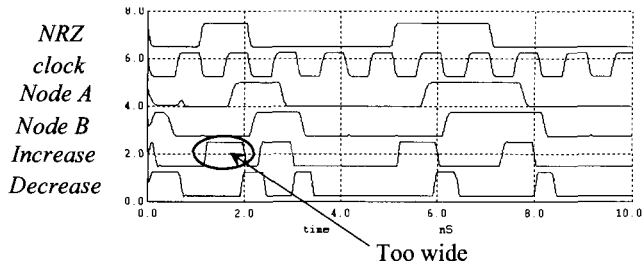


Figure 19.70 Simulating the Hogge PD in Fig. 19.69. These results should be compared to Fig. 19.49. Notice how the increased pulse widths are too wide. The result is that the Vinvco control voltage will increase above the desired value (and cause a static phase error or false locking).

Let's set C_1 to 3.5 pF, R to 5k, and C_2 to 0.35 pF. The schematic of the complete DPLL clock-recovery circuit is seen in Fig. 19.73.

In the first simulation, Fig. 19.74, we apply an alternating string of ones and zeroes to the PLL clock-recovery circuit. Since the data rate is 1 Gbit/s, the width of a one or a zero is 1 ns. In this simulation we didn't apply any initial conditions to the loop filter to start the simulation out. However, because the NRZ data is full of transitions, V_{inVCO} quickly moves to the correct voltage (around 350 mV). The important thing to note is that, again, the lock time depends on the input data.

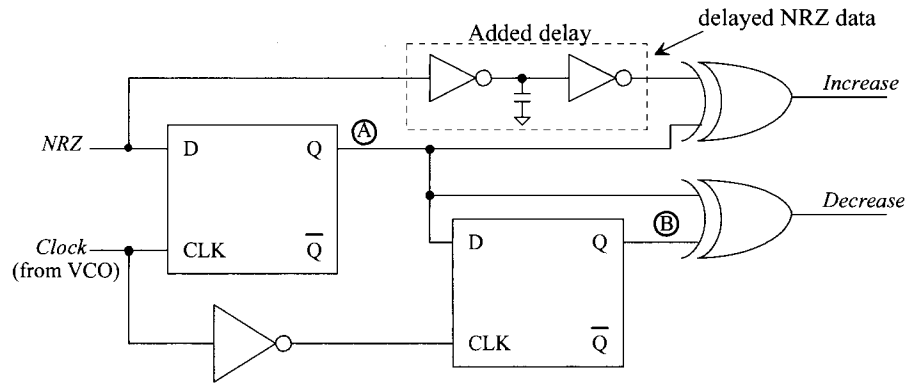


Figure 19.71 Adding a delay to the Hogge PD to compensate for the delay the NRZ data sees to node A.

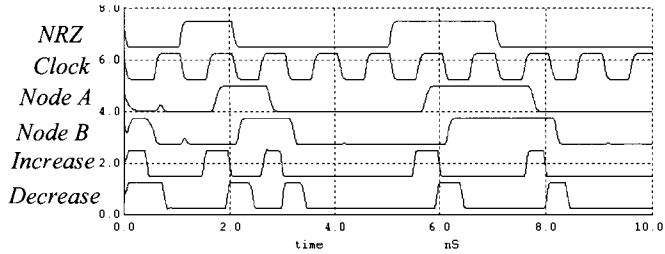


Figure 19.72 Resimulating the operation of the Hogge PD with the delay seen in Fig. 19.71 included.

For the second simulation, Fig. 19.75, we start the voltage across the loop filter, V_{inVCO} , out at 340 mV (slightly below the final value of around 350 mV). After approximately 600 ns, the loop locks. Had we not used this initial condition, the lock time (and the simulation time required to attain lock) would be quite long. Because of the large gain of the VCO, not using an initial voltage for V_{inVCO} would result in the VCO oscillating at the wrong frequency and the loop not locking correctly (locking on the wrong frequency) when the data is, as seen, a single one followed by seven zeroes (or some other sparse pattern). To avoid this problem, we can require the PLL sees a long pattern of alternating ones and zeroes to ensure lock prior to sending data (called sending a preamble) or we can reduce the gain of the VCO. The problem with sending a preamble is that it reduces the data rate through the channel. Further, large VCO gain makes it easier for the PLL to lose lock with steps in the phase shift through a communication channel. The practical solution is to reduce the gain of the VCO. In this example if we were to limit the VCO's oscillation range to, say, 950 MHz to 1050 MHz, then (with proper design of the loop filter) the PLL could acquire lock quickly and stay locked with reasonable variations in the phase shifts of the input NRZ data (these phase shifts are common in channels whose path length is not fixed, as in wireless communications). The practical problem with low VCO gain, again, is manufacturability. It's impossible to

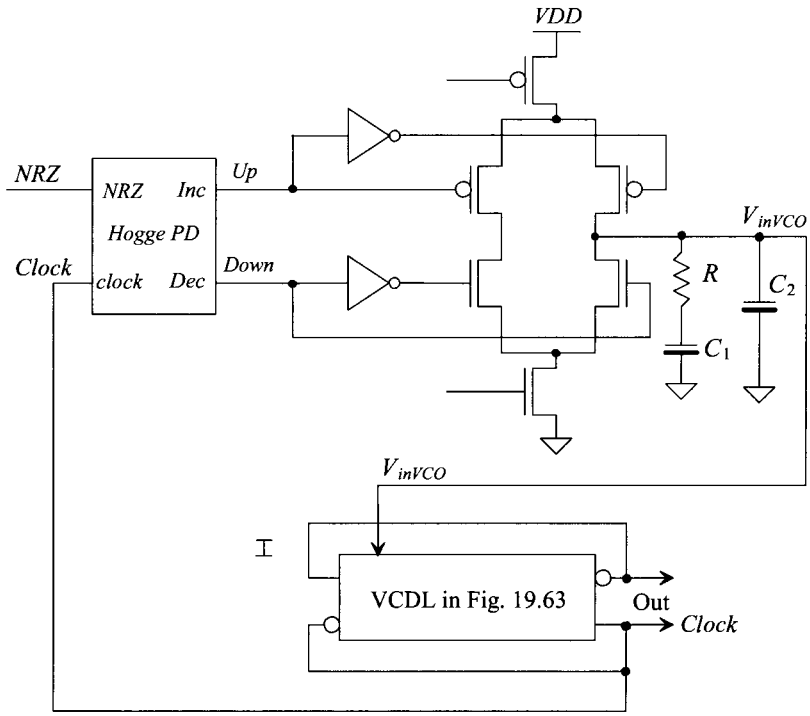


Figure 19.73 Block diagram of the DPLL used for clock-recovery discussed in this section.

design a purely monolithic CMOS oscillator (meaning no off-chip components) without some sort of hand tuning (meaning using fuses or some other sort of digital adjustment to center the oscillation frequency and limit the gain after the chips are made). Further, even if we hand-tailor the performance of each VCO in a production line, the temperature and power supply sensitivity will limit the minimum gain we can attain. Tuned circuits

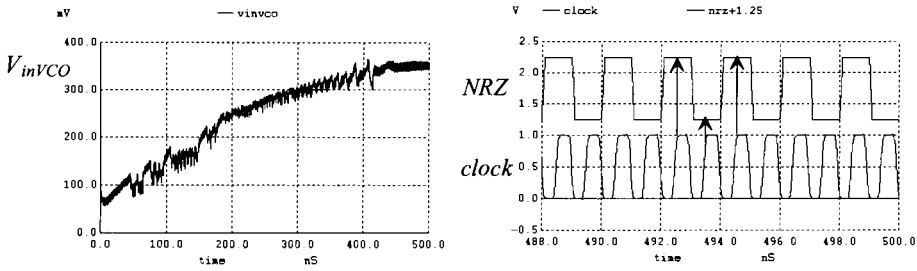
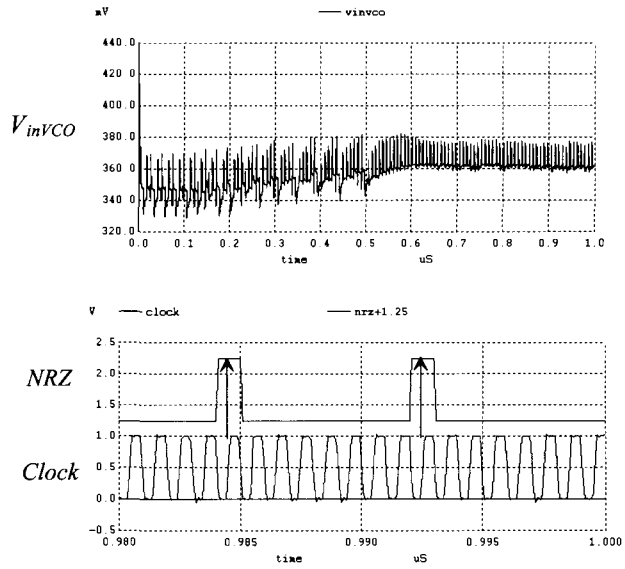


Figure 19.74 Simulating the PLL in Fig. 19.73 when the input NRZ data is an alternating string of ones and zeroes.



V_{inVCO}

NRZ

Clock

Figure 19.75 Simulating the PLL in Fig. 19.73 when the input data is a string of seven zeroes followed by a single one.

(parallel inductor-capacitor "tanks") are commonly used to help make purely monolithic clock-recovery circuits manufacturable. However, hand-tuning during Probe is still common to set the oscillation frequency and oscillation range.

ADDITIONAL READING

[1] R. E. Best, *Phase-Locked Loops: Design, Simulation, and Applications,* McGraw-Hill, 6th ed., 2007. ISBN 978-0071493758. Excellent book covering PLL design.

[2] W. F. Egan, *Phase-Lock Basics*, John Wiley and Sons, 1998. ISBN 0-4712 -4261-6 Good introduction to PLLs. See also the same author's *Frequency Synthesis by Phase Lock.*

[3] B. Razavi, *Monolithic Phase-Locked-Loops and Clock Recovery Circuits,* IEEE Press, 1996. ISBN 0-7803-1149-3. Good tutorial and collection of papers.

[4] J. Maneatis, "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques", *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1723-1732, 1996. Excellent paper discussing the design of PLLs and DLLs using self-biased techniques for very wideband operation.

[5] D. H. Wolaver, *Phase-Locked Loop Circuit Design*, Prentice-Hall, 1991. ISBN 0-1366-2743-9. Another excellent book covering PLL design.

[6] S. Haykin, *An Introduction to Analog and Digital Communications,* John Wiley and Sons, 1989. ISBN 0-471-85978-8. Covers distortionless transmission.

[7] M. G. Johnson and E. L. Hudson, "A Variable Delay Line PLL for CPU - Coprocessor Synchronization," *IEEE Journal of Solid-State Circuits,* vol. SC-23,

pp. 1218–1223, October, 1988. Classic paper presenting the concept of a delay-locked loop.

[8] C. R. Hogge, Jr., "A Self Correcting Clock Recovery Circuit," *IEEE Journal of Lightwave Technology*, vol. LT-3, pp. 1312–1314, December 1985. Paper presenting the "Hogge" phase detector.

[9] F. M. Gardner, "Charge-Pump Phase-Lock Loops," *IEEE Transactions on Communications,* COM-28, no. 11, pp. 1849–1858, November 1980. The paper first discussing charge pump PLLs.

PROBLEMS

19.1 The XOR gate PD seen in Fig. 19.4 can exhibit input-dependent skew. In other words, the delay from one of the inputs changing to the output of the PD will not be precisely the same as the delay from the other input to the output. Design an XOR PD that doesn't exhibit input-dependent skew. Hint: see Fig. 18.16.

19.2 Verify, using simulations, that a locked PLL using an XOR PD will exhibit, after RC filtering, an average value of *VDD*/2. Show, using simulations and hand calculations, the filter's average output if the XOR PD sees a phase difference in its inputs of $-\pi/4$.

19.3 Why, in your own words, is it so important for the VCO's center frequency to match the input NRZ data rate when a passive loop filter is used. Use simulations to show the problems. Why does using the active loop filter eliminate this requirement?

19.4 Suppose, for a robust high-speed PLL using an XOR PD, that it desirable to adjust the PD's gain. Show a charge pump can be used towards this goal. Should the loop filter have two outputs? Discuss the PD's gain and how it would be adjusted. What topology would be used for a passive loop filter? for an active loop filter?

19.5 Demonstrate, using simulations, how the outputs of the XOR PD can be equivalent when the loop is true or false locking (locking on a harmonic).

19.6 Describe, in your own words and using simulations, what the dotted line in Fig. 19.8 indicates.

19.7 We know that a PFD isn't generally used in clock recovery applications because both edges (*data* and *dclock*) must present when doing a phase comparison. Suggest a scheme where the edge of the input *data* enables a phase comparison. When an edge of data is not present your circuit will "swallow" the pulse from *dclock* resulting in no edges being applied to the PFD (no phase- frequency comparison).

19.8 Demonstrate, using simulations, charge sharing between the charge pump and loop filter using the topology seen in Fig. 19.12b (and Fig. 19.13b). Show how the topology in Fig. 19.37 helps with charge sharing (simulate with and without the x1 amplifier). For the x1 amplifier, use the n-type diff-amp from Fig. 18.17 without the inverter on its output. Connect the loop filter to the gate of M1 (in the diff-amp). To make the gain "1," tie the output (which is connected to drains of M1L/M2L in Fig. 19.37) of the diff-amp (the drains of M2/M4) to the gate of M2.

19.9 Using the VCO that generated the simulation data in Fig. 19.18, plot the VCO's center frequency against changes in *VDD*. Is this VCO insensitive to changes in *VDD*? Where does the sensitivity come from?

19.10 Discuss, and demonstrate with simulations, how to reduce the gain of the VCO used to generate the data in Fig. 19.18 (see Fig. 19.25 and the associated discussion). Your discussion should include some insight into the manufacturability of low-gain VCOs.

19.11 Design and simulate the operation of a 100 MHz VCO using the topology seen in Fig. 19.19a. The output of your VCO should be full logic levels. Your design should show a linearly frequency against V_{inVCO} curve. What is the gain of your design?

19.12 Using the step response of an RLC circuit, Fig. 19.76, demonstrate how selection of the resistor, inductor, and capacitor affect the output voltage's damping factor and natural frequency. From this plot show how natural frequency and lock time are related.

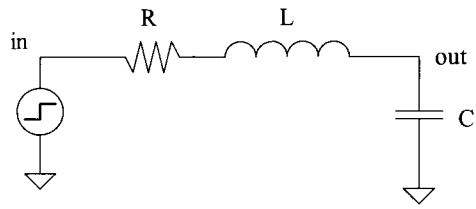


Figure 19.76 Using a second-order circuit to demonstrate how lock time and natural frequency are related.

19.13 Replace, in the DPLL that generated the waveforms in Figs. 19.27 and 19.28, the simple RC loop filter with the passive lag loop filter in Fig. 19.29. Show how the filter's component values are selected. Comment on, using simulations to support your conclusions, how the performance of the DPLL is affected.

19.14 Using the PFD in Fig. 19.33 and the charge pump in Fig. 19.36, demonstrate, with simulations, how the gain of PFD/charge-pump configuration can have a dead zone (the gain, K_{PDI}, decreases) as seen in Fig. 19.77.

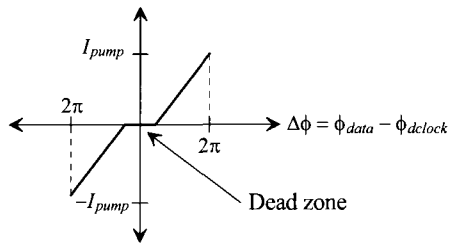


Figure 19.77 Showing the dead zone in a PFD/charge-pump.

19.15 Suppose that the pump currents used in the DPLL in Ex. 19.5 are mismatched by 50%. Will this cause a static phase error? Why or why not? Use simulations to support your answer.

19.16 Redesign the DPLL in Ex. 19.5 so that the output remains a 100 MHz square wave signal when the input signal is changed to 25 MHz.

19.17 We used pF values for the capacitors in the loop filters in this chapter. Why not reduce the filter's layout area by using fF size capacitors? Demonstrate the problems with using such small loop filter capacitors.

19.18 What are we sacrificing by using an equalizer? What does the minus sign indicate in the slope of the phase in Fig. 19.41?

19.19 Using the DPLL from Ex. 19.2, demonstrate the false locking as seen in Fig. 19.46.

19.20 Design an edge detector, like the one seen in Fig. 19.47, for use in the DPLL in Ex. 19.2. Regenerate the simulation data seen in Figs. 19.27 and 19.28. Remember to eliminate the divide by two in the feedback path.

19.21 Derive Eq. (19.71).

19.22 Design a nominal delay line of 1 ns (when V_{indel} = 500 mV) using the current starved delay element seen in Fig. 19.55. Determine the delay's sensitivity to variations in *VDD*.

19.23 Repeat Problem 19.22 for the inverter delay cell in Fig. 19.55.

19.24 Suppose, as seen in Fig. 19.78, that instead of using a half-replica of the delay cell in Fig. 19.58, the full delay cell is used to generate V_{rbias}. Electrically is there any difference in V_{rbias} when comparing the full- and half-replica circuits? What may be the benefit of using a full-replica of the delay cell?

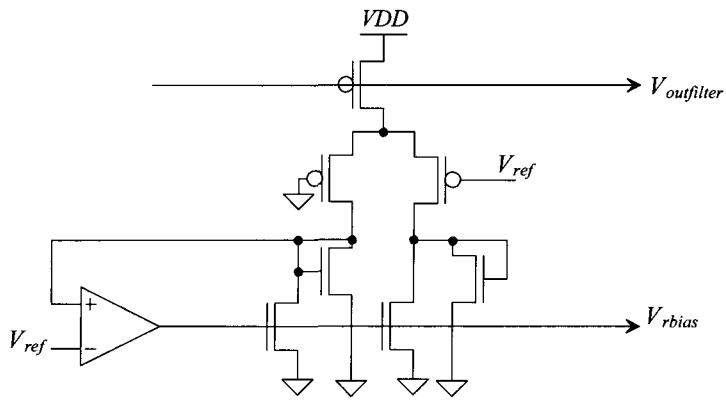


Figure 19.78 Using a full-replica of the delay element for generating Vrbias.

19.25 For the delay line that generated the simulation results in Fig. 19.62 determine, using simulations, the delay's sensitivity to changes *VDD*. Plot the VCDL's delay as a function of *VDD* with V_{indel} held at 500 mV.

19.26 The delay element seen in Fig. 19.79 doesn't use a reference voltage. Show how this element can be used in the VCDL of Fig. 19.61. Note that the input capacitance of this delay stage is twice as large as the input capacitance of the element in Fig. 19.61 (and so you may have to use fewer stages to attain the same overall delay). How does the NMOS gate potential change with the inputs/outputs switching? Why? Is the output amplitude a function of *VDD*?

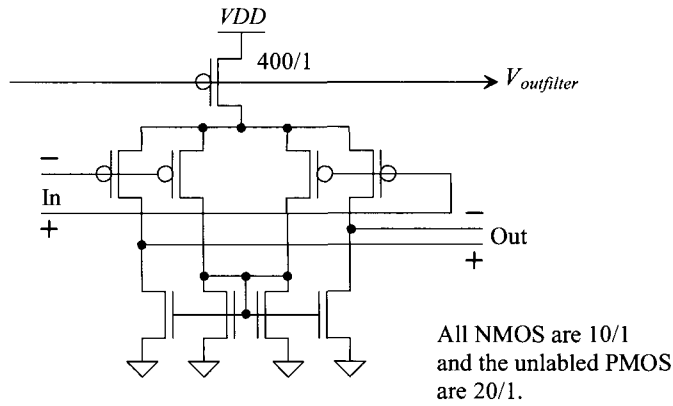


Figure 19.79 A delay element that doesn't use a reference voltage.

19.27 Use the delay element in Fig. 19.67 to implement a VCDL. Use the VCDL in the DLL seen Fig. 19.65 to generate the waveforms in Fig. 19.66. Show the 90, 180, 270, and 360 degree outputs of the VCDL swinging to full-logic levels.

19.28 The VCO output seen in Fig. 19.68 doesn't have an exactly 50% duty cycle. Will this affect a clock-recovery circuit's operation? Why or why not? Use simulations to support you answer.

19.29 Suggest another method, other than the one seen in Fig. 19.72, to equalize the widths of the *Increase* and *Decrease* outputs of the Hogge PD. Verify your design with simulations.

19.30 Must the currents in the charge pump in Fig. 19.73 be equal for proper DPLL clock-recovery operation? Why or why not? What happens if the currents aren't equal? What happens if the NMOS switches turn on at different speeds than the PMOS switches? Use SPICE to support your answers.

19.31 Modify the simulation inputs in Fig. 19.74 to show false locking. Comment on what can be done in a practical clock-recovery circuit to eliminate false locking.

19.32 Replace the charge pump used in the DPLL in Fig. 19.73 with the active-PI loop filter seen in Fig. 19.50. Calculate the loop filter component values. Using the new loop filter, regenerate Figs. 19.74 and 19.75.