Behzad Razavi

The Delay-Locked Loop

Delay-locked loops (DLLs) can be considered as feedback circuits that phase lock an output to an input without the use of an oscillator. In some applications, DLLs are necessary or preferable over phase-locked loops (PLLs), with their advantages including lower sensitivity to supply noise and lower phase noise. This article deals with fundamental DLL design concepts.

The origins of DLLs can be traced to a paper published in 1961 [1]. The authors present the topology shown in Figure 1 as a "delay-lock discriminator" operating on random signals. The feedback loop consists of a controlled delay line, a multiplier acting as a phase detector (PD), and a low-pass filter. The use of DLLs in modern CMOS design evidently began with the work by Bazes in 1985 [2] and Johnson and Hudson in 1988 [3].

Basic Idea

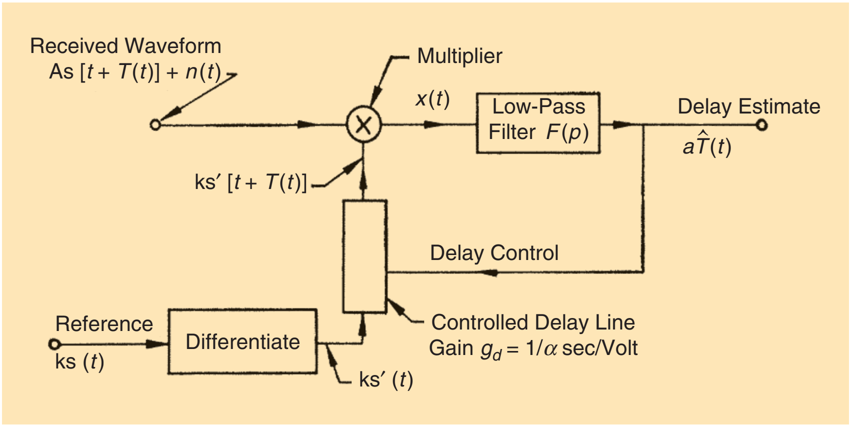
Suppose, as shown in Figure 2(a), an input clock travels on a long interconnect, experiencing a significant skew, ΔT. How do we align CK_{out} with CK_{in}? Since the clock is periodic, we surmise that an *additional* delay can be introduced to make the total delay equal to one clock cycle [Figure 2(b)]. To set the delay properly, we can view ΔT as an error that must be suppressed by means of negative feedback. That is, if the phase of CK_{out} is compared to that of CK_{in}, the resulting error can be used to adjust the delay and force ΔT toward zero. This conjecture leads us to the arrangement depicted in Figure 2(c). Here, a phase detector measures the skew and adjusts the delay of B_2 to reduce ΔT. As with PLLs, the low-pass filter attenuates the high-frequency components generated by the PD. This circuit exemplifies a simple delay-locked loop.

The origins of DLLs can be traced to a paper published in 1961.

The residual phase error in Figure 2(c) depends on the loop gain, i.e., the gain of the PD, K_{PD}, and the gain of the variable-delay stage. The latter is defined as $K_{DL} = \partial\phi/\partial V_{cont}$, where ϕ is the stage's delay in radians. Rather than attempt to maximize $K_{PD}K_{DL}$, we can add an integrator to the loop. Drawing upon our knowledge of PLLs, we thus construct the architecture shown in Figure 2(d), where the cascade consisting of the phase/frequency detector (PFD), charge pump (CP), and capacitor provides an infinite gain, thus driving the skew toward zero. The variable-delay stage is realized as a voltage-controlled delay line (VCDL). Figure 2(e) shows an example of VCDL design employing varactors for delay control. While the DLL does not require frequency detection, the PFD provides a convenient interface with the CP. As explained next, no resistor is necessary in series with C_1. This DLL architecture is commonly used in high-speed systems.

The DLL of Figure 2(d) is of first order, facing no stability issues. Moreover, it benefits from the lower phase noise and supply sensitivity of delay lines compared to oscillators.

In contrast to PLLs, delay-locked loops do not *generate* a frequency;



Received Waveform As $[t + T(t)] + n(t)$
Multiplier
$x(t)$
Low-Pass Filter $F(p)$
Delay Estimate $a\hat{T}(t)$
$ks'[t + T(t)]$
Delay Control
Reference $ks(t)$
Differentiate
Controlled Delay Line Gain $g_d = 1/\alpha$ sec/Volt
$ks'(t)$

FIGURE 1: An early DLL reported in [1].

Digital Object Identifier 10.1109/MSSC.2018.2844615
Date of publication: 13 August 2018

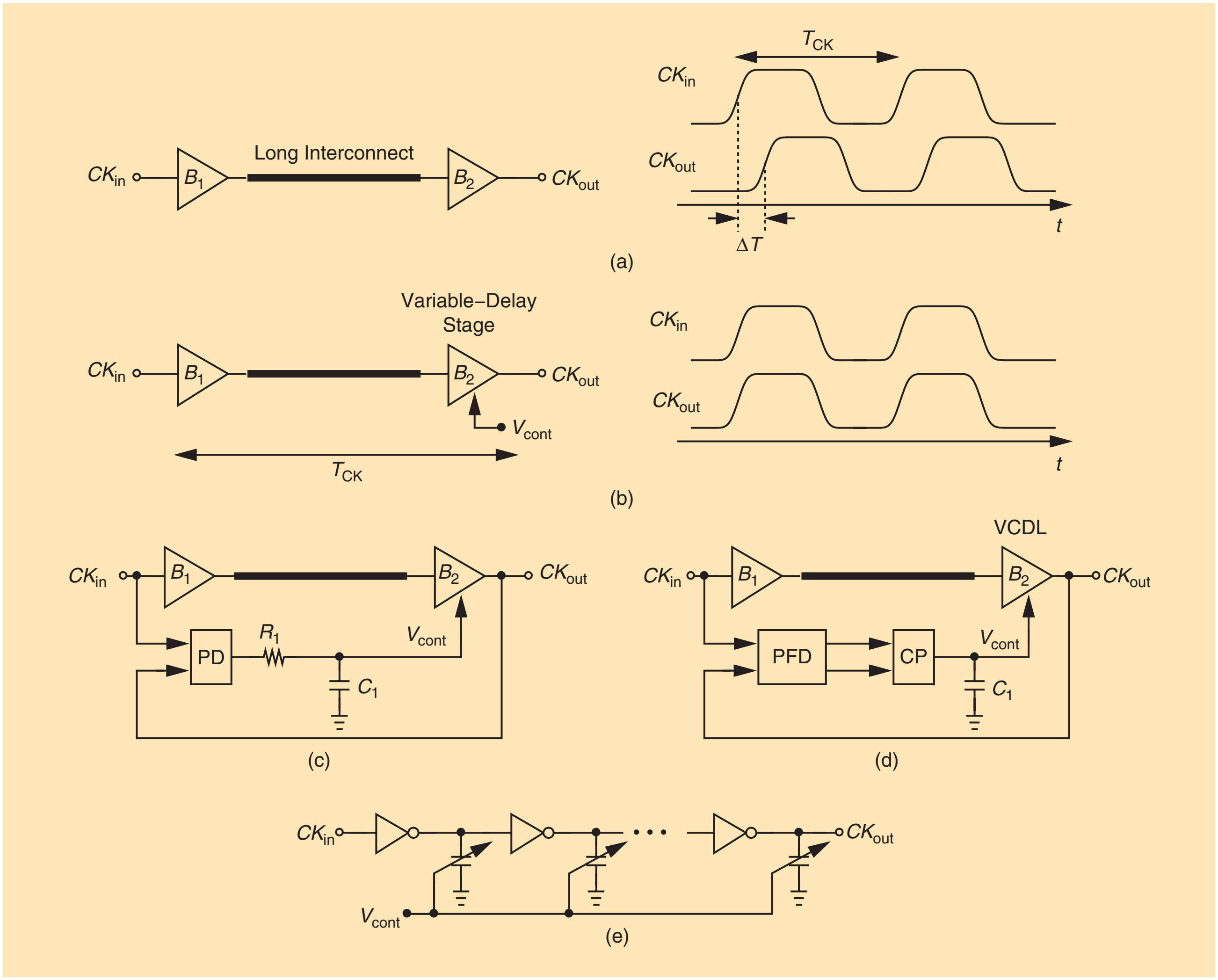


FIGURE 2: (a) An interconnect with skew, (b) the correction of skew by a delay stage, (c) a simple feedback system for controlling the delay line, (d) a basic DLL, and (e) a VCDL implementation example.

rather, they simply delay the input. As such, DLLs are less versatile than PLLs. For example, in practice, a DLL would not be able to generate a 5-GHz clock from a 20-MHz reference.

Another drawback of DLLs is that they allow the input duty cycle error to propagate to the output. In fact, the delay line may further increase this error. Thus, the VCDL is typically preceded or followed by a duty cycle correction stage. A third drawback of DLLs is that they operate the PFD and the CP at high speeds.

The dynamic behavior of DLLs determines how they respond to such effects as input phase noise, and supply noise. We therefore study this behavior in the next section.

Loop Dynamics

We wish to analyze the dynamic behavior of the DLL shown in Figure 3(a). In the locked state, the phase difference between CK_{in} and CK_{out} is constant and, in principle, equal to zero. Thus, the VCDL provides a delay of one clock period, T_{CK}.

Before delving into the overall loop dynamics, let us understand those of the VCDL itself. The circuit has a clock input and a control input. What happens if CK_{in} in Figure 2(e) incurs a phase step? This step propagates through the chain and emerges at the output T_{CK} seconds later [Figure 3(b)]. That is, the transfer function associated with this path can be expressed as $\exp(-T_{CK}s)$. In practice, T_{CK} is much less than the overall DLL time constant, allowing the approximation $\exp(-T_{CK}s) \approx 1$.

How about the path from V_{cont} to CK_{out}? If we apply a step at V_{cont} in Figure 2(e), how long does it take to affect the output phase? From the waveforms shown in Figure 3(c), we recognize that this path too has a delay of at most one T_{CK}. Based on these observations, we can construct an approximate, static model for the VCDL; as shown in Figure 3(d), it simply adds a phase equal to $K_{DL}V_{cont}$ to the input phase. (The one-cycle delay is neglected here.)

It is instructive to first examine the overall DLL's response qualitatively. If the phase of CK_{in} in Figure 3(a) fluctuates slowly, the DLL maintains a high loop gain, keeping

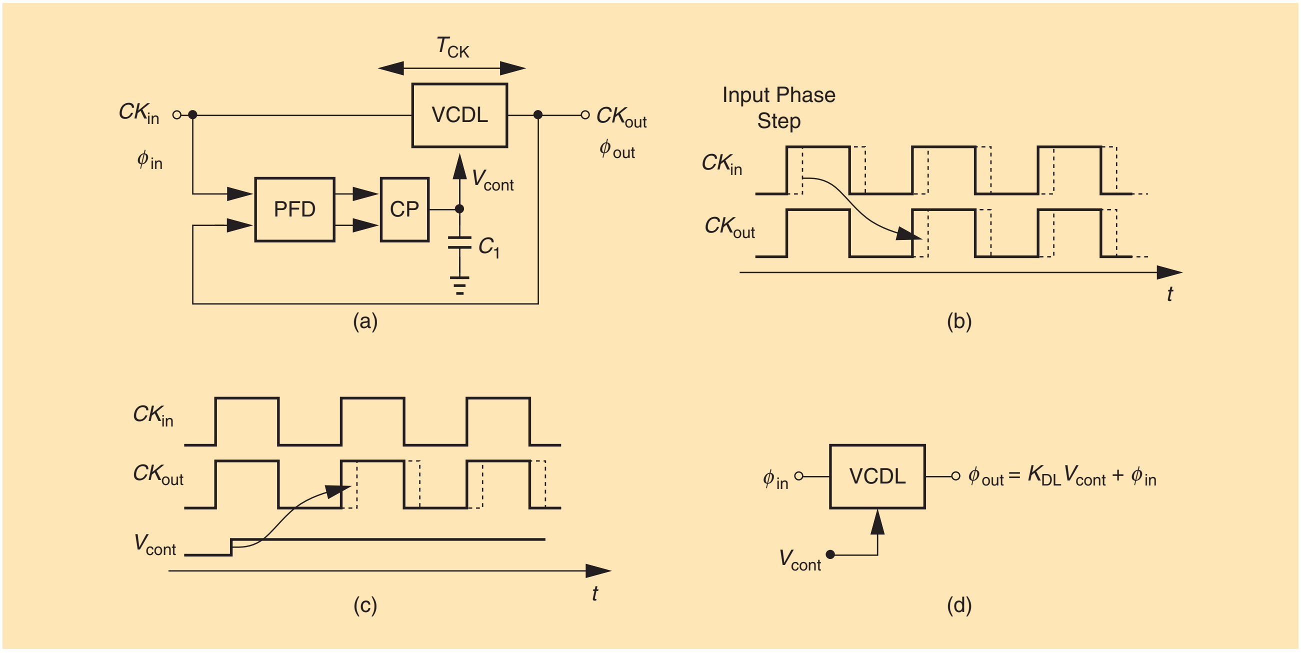


FIGURE 3: (a) A DLL with a VCDL, (b) the propagation of input phase step to output, (c) the propagation of step on V_{cont} to output, and (d) a linear model of a VCDL.

CK_{out} aligned with CK_{in}. That is, the closed-loop transfer function has a unity magnitude for slow phase variations. Now suppose CK_{in} experiences very fast phase changes. Then, the DLL has little loop gain, V_{cont} does not change, and CK_{in} simply propagates to CK_{out}. In this case, too, the closed-loop response is around unity because the input phase changes appear at the output with only a delay of T_{CK} seconds. We thus conclude that DLLs exhibit an all-pass response, a point of contrast to the low-pass behavior of PLLs.

The all-pass nature of DLLs can also be confirmed mathematically. For the DLL of Figure 3(a), we draw the phase model as shown in Figure 4(a), noting that V_{cont} is given by $(\phi_{in} - \phi_{out})[I_p/(C_1 s)]$, where I_p denotes the charge pump current, and hence

$$\phi_{in} + (\phi_{in} - \phi_{out}) \frac{I_p}{C_1 s} K_{DL} = \phi_{out}. \quad (1)$$

That is,

$$-(\phi_{in} - \phi_{out}) = (\phi_{in} - \phi_{out}) \frac{K_{DL} I_p}{C_1 s}, \quad (2)$$

which implies $\phi_{in} = \phi_{out}$. In practice, the response exhibits a small amount of peaking [Figure 4(b)] [4].

We should remark that some DLLs apply an independent reference clock to the PFD and do not follow these dynamics [4].

The aforementioned study reveals two points: 1) DLLs do not generally face stability issues and can operate with a wide range of values for I_p and C_1, and 2) the lack of filtering ability precludes the use of the foregoing DLLs in applications where the input jitter must be removed. The latter issue is resolved by a different DLL architecture [4].

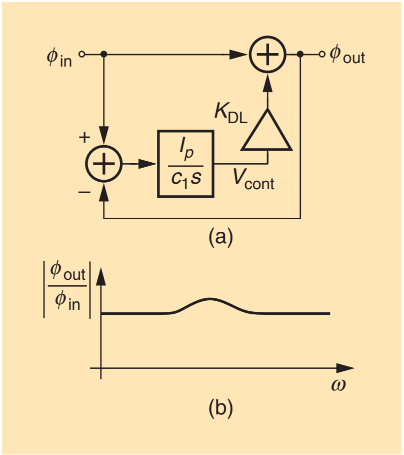


FIGURE 4: (a) A linear model of DLL and (b) the DLL phase response.

Effect of Supply Noise

The principal effect of supply noise, $V_{DD}(t)$, in DLLs is to modulate the delay of the VCDL. How does the DLL of Figure 3(a) respond to $V_{DD}(t)$? If the noise varies slowly, the loop has enough "strength" to keep ϕ_{out} close to ϕ_{in}, i.e., V_{cont} opposes $V_{DD}(t)$ and ϕ_{out}

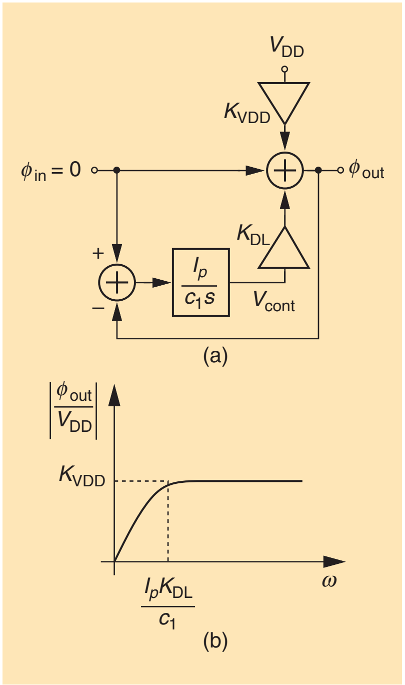


FIGURE 5: (a) The supply noise in a DLL and (b) the DLL phase response.

is not affected. For high-frequency noise, on the other hand, the loop gain drops, and ϕ_{out} is directly modulated by V_{DD}.

Let us define for the VCDL a gain from V_{DD} to ϕ_{out} as $K_{VDD}=\partial\phi_{\text{out}}/\partial V_{DD}$. Shown in Figure 5(a) is the DLL model with supply noise and $\phi_{\text{in}}=0$. Beginning from the output, we can write V_{cont} as $-\phi_{\text{out}}[I_p/(C_1 s)]$ and hence

$$-\phi_{\text{out}}\frac{I_p}{C_1 s}K_{\text{DL}}+V_{DD}K_{VDD}=\phi_{\text{out}}. \quad (3)$$

It follows that

$$\frac{\phi_{\text{out}}}{V_{DD}}(s)=\frac{K_{VDD}C_1 s}{C_1 s+I_p K_{DL}}. \quad (4)$$

Plotted in Figure 5(b), the response begins to flatten out beyond the pole frequency, $\omega_p = I_p K_{DL}/C_1$. We must therefore choose a high value for ω_p so as to maximize the supply rejection.

Effect of Phase Noise

DLLs are generally considered to generate much less phase noise than PLLs, but the comparison must be done carefully. We begin with the input phase noise. As exemplified by the transfer function plotted in Figure 4(b), this noise experiences no attenuation and simply propagates to the output.

The case of the VCDL phase noise is more interesting. We make two observations. First, as shown in [5], the phase noise of a delay line, $S_{\phi,\text{DL}}$, and that of a ring oscillator using such a line, $S_{\phi,\text{ring}}$, are related as follows:

$$S_{\phi,\text{ring}}(f)=S_{\phi,\text{DL}}\left(\frac{f_0}{\pi\Delta f}\right)^2, \quad (5)$$

where f_0 is the oscillation frequency (Figure 6). We conclude that the ring produces much higher phase noise. One interpretation of this result is that, in a ring, an edge continues to accumulate phase noise as it circulates, whereas, in a delay line, an edge experiences the phase noise of the delay stages only once before it reaches the output [6].

Second, we model the VCDL phase noise as shown in Figure 7(a) and write

$$-\phi_{\text{out}}\frac{I_p}{C_1 s}K_{\text{DL}}+\phi_{n,\text{DL}}=\phi_{\text{out}}, \quad (6)$$

obtaining

$$\frac{\phi_{\text{out}}}{\phi_{n,\text{DL}}}(s)=\frac{C_1 s}{C_1 s+I_p K_{\text{DL}}}. \quad (7)$$

Similar to the effect of supply noise, this result indicates a first-order high-pass behavior [Figure 7(b)]. As expected, the loop rejects slow phase fluctuations caused by the VCDL. In general, the dominant source of phase noise in VCDLs is the supply noise.

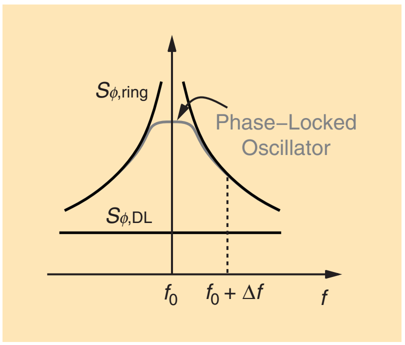


FIGURE 6: The comparison of ring oscillator phase noise and delay line phase noise.

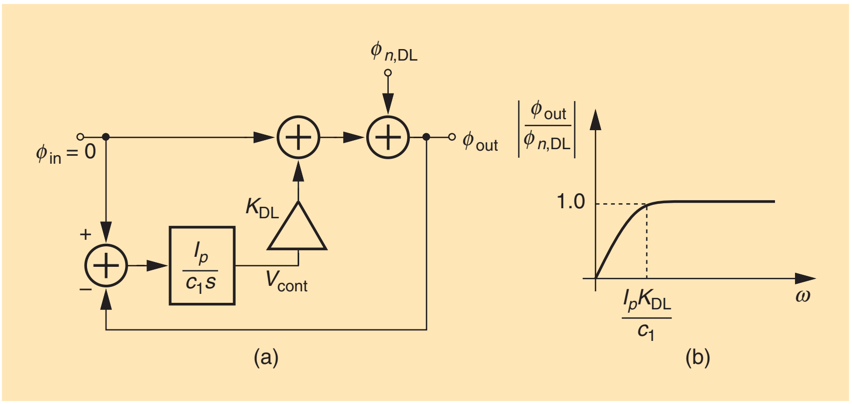


FIGURE 7: (a) A DLL model including delay line phase noise and (b) the DLL response.

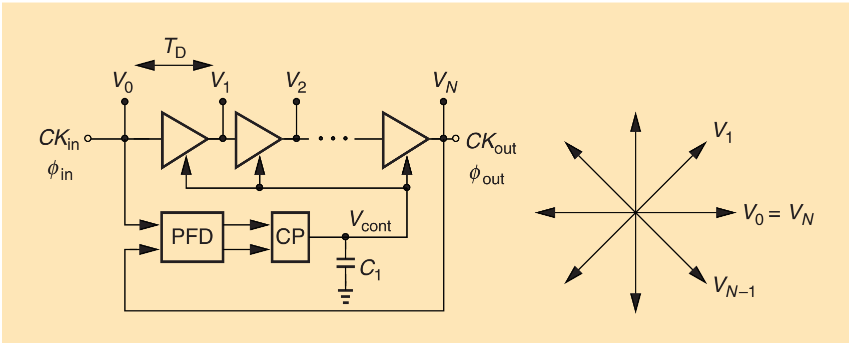


FIGURE 8: A DLL generating multiple phases.

Generation of Multiple Phases

In addition to the deskewing function illustrated in Figure 2(d), DLLs also find application in systems requiring multiple clock phases. For example, some clock and data recovery circuits demand 32 or 64 equally spaced clock phases, a difficult situation for ring oscillators as their operation frequency is inversely proportional to the number of stages that they employ.

Figure 8 depicts a DLL that delivers multiple clock phases. Incorporating N nominally identical delay stages, the VCDL provides N phases with a minimum spacing equal to the delay of one stage, T_D. The key point here is that $T_D = T_{CK}/N$ because the loop locks such that CK_{out} and CK_{in} have a phase difference of T_{CK}. In other words, by virtue of the feedback around the loop, T_D remains well defined and relatively precise even with process, voltage, and temperature (PVT) variations. By comparison, a "free-running" delay line can

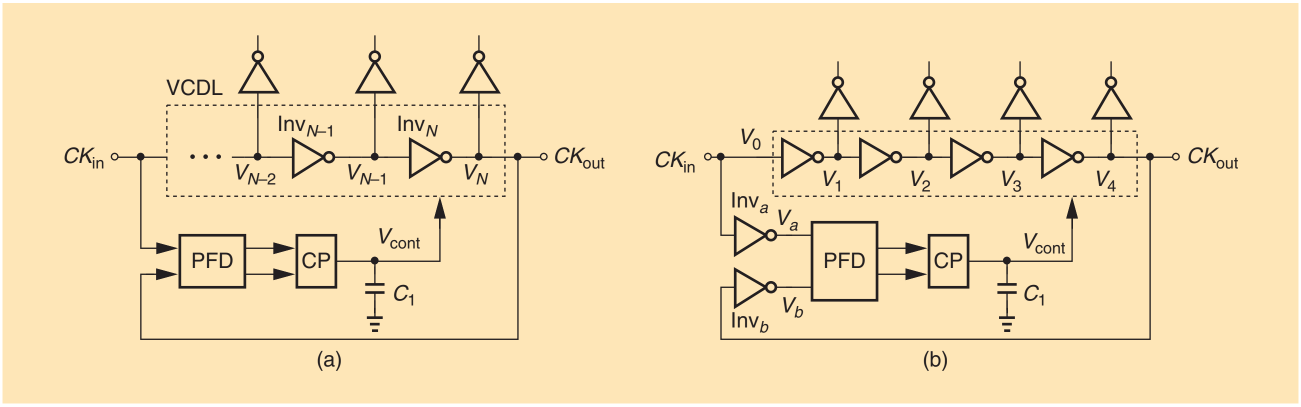


FIGURE 9: (a) The DLL of Figure 8 with output inverters added and (b) uniform fanouts at all VCDL taps.

experience a nearly twofold change in its delay as a function of PVT.

The multiphase DLL shown in Figure 8 faces several issues. First, due to unequal loading, the phase spacings at the boundaries of the VCDL can be different from those in the middle. To understand this point, consider the situation illustrated in Figure 9(a), where inverters Inv_{N-1} and Inv_N see different fanouts; the former drives two inverters but the latter, an inverter and a PFD. As a result, the phase difference between V_{N-2} and V_{N-1} is not the same as that between V_{N-1} and V_N. This issue is overcome as shown in Figure 9(b), with two inverters inserted at the PFD inputs. Assuming all of the inverters are identical, we observe that 1) the fanout at V_4 is equal to that at V_1, V_2, and V_3, and 2) the loop drives the phase difference between V_a and V_b to zero, thus aligning V_0 and V_4 as well. One assumption here is that the waveform arriving at V_0 has approximately the same rise and fall times as that at V_4; otherwise, the delays through Inv_a and Inv_b are slightly different.

Another issue in the DLL of Figure 8 is the problem of "false lock." Assume the circuit is designed to provide a total delay of T_{CK} at the typical-typical (TT), 27 °C corner. Now, suppose the DLL operates in the slow-slow (SS), high-temperature corner, and, upon startup, the total VCDL delay is slightly *greater* than $2T_{CK}$ (Figure 10). Then, the DLL simply attempts to align V_0 and V_N, and it can do so if the phase difference between these two signals reaches $2T_{CK}$ rather than T_{CK}. As a result, the phase spacings will be equal to $2T_{CK}/N$.

Avoiding false lock generally requires substantial added complexity, especially if the DLL must operate across a wide frequency range. Depicted in Figure 11 is a solution employing a PLL. A replica of the VCDL is configured as a ring oscillator and phase locked to the main input, thus guaranteeing that the delay from A to B is equal to T_{CK} and hence V_{cont1} reaches the desired value. Now, this voltage serves as the coarse control for the main VCDL, allowing the DLL to provide only a fine adjustment through V_{fine}. For example, if the two VCDLs have a delay mismatch of 10%, then V_{fine} must vary the delay by only this amount, thereby avoiding false lock. The filter preceding V_{coarse} suppresses the ripple and noise present in V_{cont1}. This architecture approximately doubles the area and power consumption. Another method is described next.

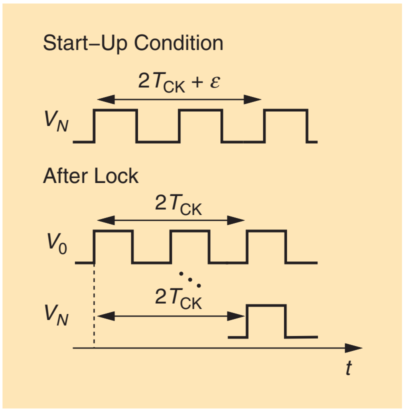


FIGURE 10: The problem of false lock.

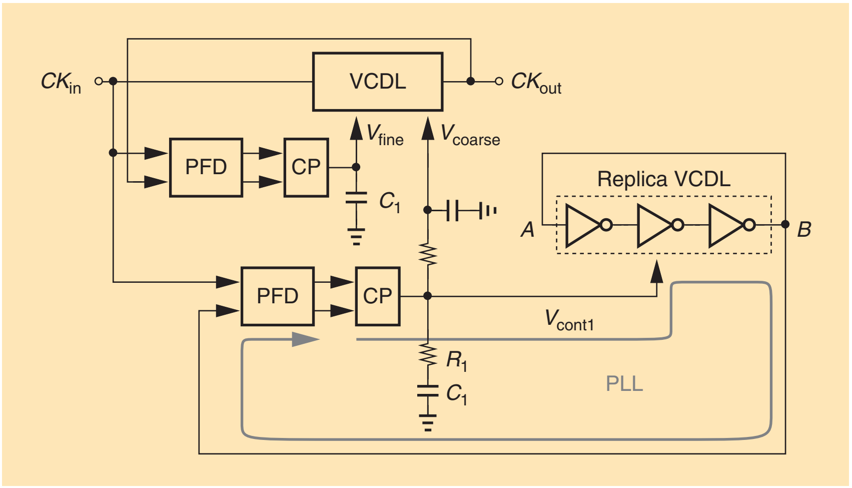


FIGURE 11: The addition of a PLL to a DLL to avoid false lock.

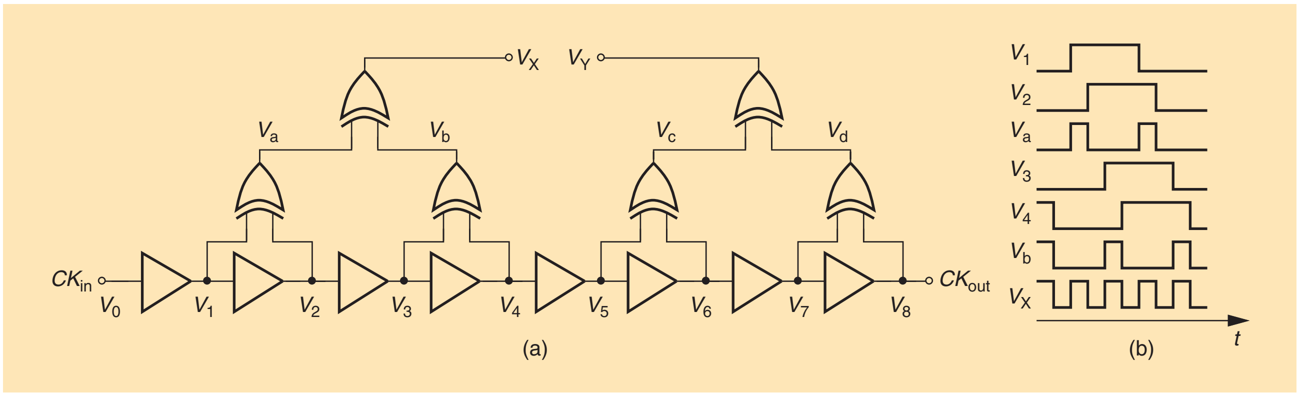
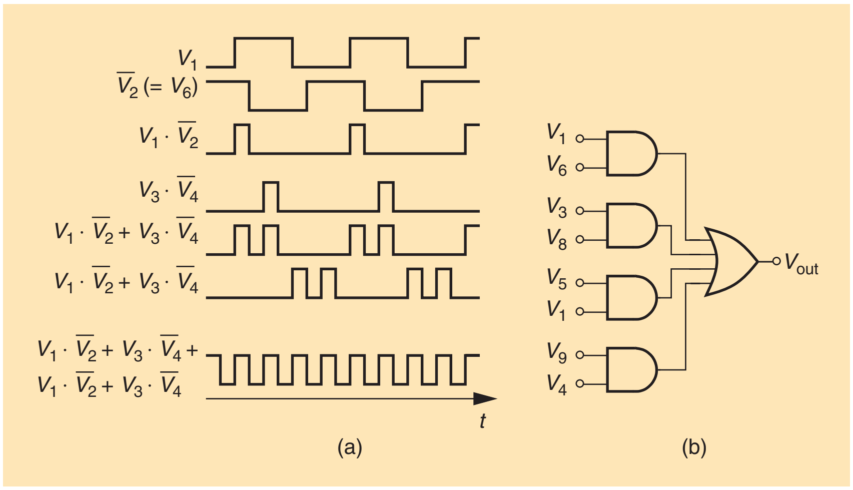


FIGURE 12: (a) An edge-combining circuit for frequency multiplication and (b) its waveforms.



V_1
$\overline{V_2}\,(= V_6)$
$V_1 \cdot \overline{V_2}$
$V_3 \cdot \overline{V_4}$
$V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4}$
$V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4}$
$V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4} +$
$V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4}$
t
(a)
V_1
V_6
V_3
V_8
V_5
V_1
V_9
V_4
V_{out}
(b)

FIGURE 13: (a) The use of AND and OR operations for frequency multiplication and (b) the logical implementation.

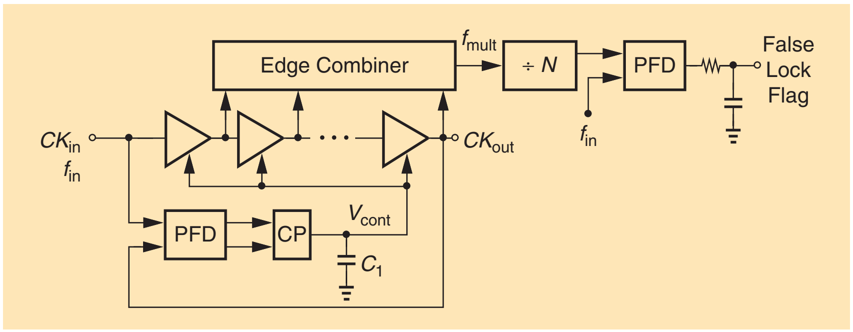


FIGURE 14: A false lock detector using a frequency multiplier.

The third issue in the multiphase DLL of Figure 8 relates to the mismatches between the delay units, which translate to departures in the phase spacings from T_{CK}/N. Present in both the driving strength of the stages and their load capacitances, the mismatches must be managed by proper sizing, careful layout, and Monte Carlo simulations.

Frequency-Multiplying DLLs

An important shortcoming of DLLs is their inability to perform frequency synthesis, i.e., generate arbitrary output frequencies. In some applications, a DLL can multiply the input frequency by an integer, thereby acting as a "poor man's" frequency synthesizer.

Recall that the DLL of Figure 8 produces N equally spaced clock edges with a resolution of T_{CK}/N seconds. If we *combine* these edges, we can generate an output having a higher frequency. Shown in Figure 12(a) is an eight-phase delay line, with its outputs applied to XOR gates. Noting that the DLL aligns V_8 and V_0, we observe a phase difference of 45° between adjacent taps. As illustrated in Figure 12(b), the XOR result of V_1 and V_2 exhibits pulses every $T_{CK}/2$ seconds and so does the XOR result of V_3 and V_4. Since V_a and V_b have a phase difference of 90°, their XOR result, V_X, has a period of $T_{CK}/4$ seconds. From another perspective, the first rank of XOR gates doubles the frequency, and the second rank doubles again. The eight-stage DLL thus multiplies the input frequency by a factor of four. Note that the XORs introduce uniform loading along the delay line (if the XOR gates are symmetric with respect to their two inputs). The XOR stages form an "edge combiner" here.

It is possible to design an edge combiner using AND and OR gates. The delay line shown in Figure 12(a) provides delayed signals and their complements. For example, $V_5 = \overline{V_1}$, $V_6 = \overline{V_2}$, etc. If we AND V_1 and $\overline{V_2}$, we obtain one pulse of width $T_{CK}/8$ every T_{CK} seconds [Figure 13(a)]. Similarly, $V_3 \cdot \overline{V_4}$ exhibits the same shape but shifted by $T_{CK}/4$. It follows that $V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4}$ yields two such pulses every T_{CK} seconds. Noting that $V_5 \cdot \overline{V_6} + V_7 \cdot \overline{V_8}$ has the same behavior

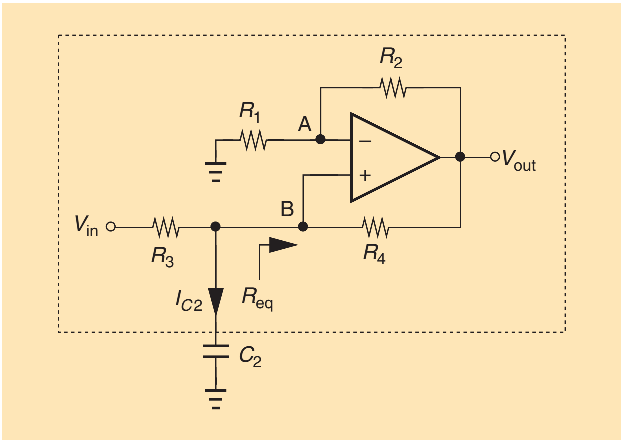


FIGURE 15: A noninverting integrator.

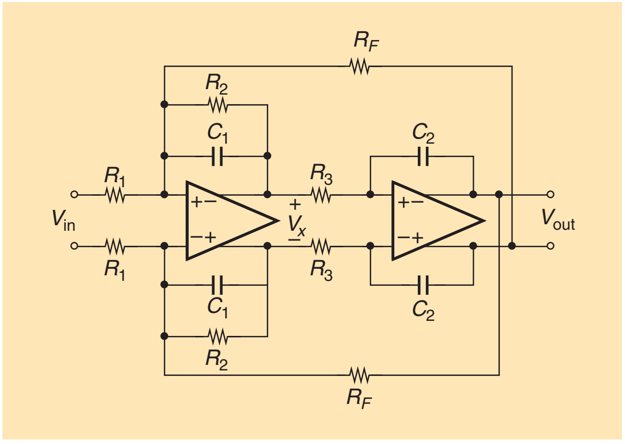


FIGURE 16: The Tow–Thomas biquad.

but shifted by $T_{CK}/2$, we conclude that $V_1 \cdot \overline{V_2} + V_3 \cdot \overline{V_4} + V_5 \cdot \overline{V_6} + V_7 \cdot \overline{V_8}$ is a signal with four times the input frequency. The implementation is shown in Figure 13(b). In reality, the AND and OR gates are replaced by NANDs.

The multiplication factor in Figures 12 and 13 is difficult to change, a point of contrast to PLLs. Moreover, delay mismatches among the stages give rise to jitter and spurs.

The frequency multiplication ability of DLLs can be exploited to detect false locking. Consider the architecture shown in Figure 14, where an edge combiner multiplies the frequency by a factor of N. This result, f_{mult}, is then divided by N and compared to f_{in}. With correct locking, $f_{mult} = Nf_{in}$, leading to a low average value for the PFD output. In the presence of false lock, on the other hand, the total delay from CK_{in} to CK_{out} is equal to or greater than $2T_{CK}$, and $f_{mult} < Nf_{in}$. As a result, the PFD output exhibits a higher average. The false lock flag can then be used to adjust the tuning range of the delay line so that the total delay remains less than $2T_{CK}$.

Questions for the Reader

1) Suppose the up and down currents in the charge pump of Figure 2(d) have a mismatch of ΔI.How does the DLL react to this mismatch?
2) The CP imperfections in Figure 2(d) create a periodic ripple in V_{cont}. What is the effect of this ripple on the output waveform?

Answers to Last Issue's Questions

1) Figure 15 shows a noninverting integrator. Derive the condition for the elements so that the circuit acts as an ideal integrator. What is the principal difficulty with this topology?

We have $V_B \approx V_A = V_{out}R_1/(R_1+R_2)$. Also, $(V_{in}-V_B)/R_3+(V_{out}-V_B)/R_4 = V_B C_2 s$. Thus,

$$\frac{V_{in}}{R_3} = V_{out}\frac{R_1 C_2 s}{R_1+R_2} + V_{out}\left(\frac{R_1}{R_1+R_2}\cdot\frac{R_3+R_4}{R_3R_4} - \frac{1}{R_4}\right). \quad (8)$$

For ideal integration, the second term on the right-hand side must vanish, yielding $R_2/R_1 = R_4/R_3$.

Another perspective provides additional insight. If V_{out} is proportional to the integral of V_{in}, then so are V_A and V_B. Thus, $I_{C2} = C_2 dV_B/dt$ is also proportional to V_{in}, a condition that is met only if the Norton equivalent of the circuit in the dashed box reduces to an ideal current source. Since the Norton resistance is given by $R_3 || R_{eq}$, we set R_{eq} to $-R_3$ and hence obtain $R_2/R_1 = R_4/R_3$.

The principal issue here is that the circuit relies on equal positive and negative feedback factors and is prone to latch up in the presence of component mismatches.

2) Suppose the Tow–Thomas biquad of Figure 16 senses a large, narrowband undesired channel at $\omega = \omega_{-3dB}$. Which of the two integrators produces greater voltage swings and hence experiences more nonlinearity?

We have $V_{out}/V_X = -1/(R_3C_2s)$. The relative swings in the two integrator outputs depend on the component values. For example, if $C_1 = C_2$, $R_2 = R_3 = R_F$, and $Q = 1$, then $\omega_n = 1/(R_3C_2)$ and $\omega_{-3dB} = 1.27\omega_n = 1.27/(R_3C_2)$. That is, $|V_{out}/V_X| = 1/1.27$. In this case, the first integrator compresses first. If the undesired channel occurs at $2\omega_{-3dB}$, we have $|V_{out}/V_X| = 1/2.54$, observing even a greater swing disparity.

References

[1] J. J. Spilker and D. T. Magill, "The delay-lock discriminator: An optimum tracking device," *Proc. IEEE*, vol. 49, pp. 1403–1416, Sept. 1961.
[2] M. Bazes, "A novel precision MOS synchronous delay line," *IEEE J. Solid-State Circuits*, vol. 20, pp. 1265–1271, Dec. 1985.
[3] M. G. Johnson and E. I. Hudson, "A variable delay line PLL for CPU-coprocessor synchronization," *IEEE J. Solid-State Circuits*, vol. 23, pp. 1218–1223, Oct. 1988.
[4] M. J. E. Lee, W. J. Dally, T. Greer, H. T. Ng, R. Farjad-Rad, J. Poulton, and R. Senthinathan, "Jitter transfer characteristics of delay-locked loops: Theories and design techniques," *IEEE J. Solid-State Circuits*, vol. 38, pp. 614–621, Apr. 2003.
[5] A. Homayoun and B. Razavi, "Relation between delay line phase noise and ring oscillator phase noise," *IEEE J. Solid-State Circuits*, vol. 49, pp. 384–391, Feb. 2014.
[6] J. Sonntag and R. Leonowich, "A monolithic CMOS 10 MHz DPLL for burst-mode data retiming," in *Proc. Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1990, pp. 194–195.