12 SEU/SET Tolerant Phase-Locked Loops

Robert L. Shuler, Jr.

CONTENTS

12.1 INTRODUCTION

The phase-locked loop (PLL) is an old and widely used circuit for frequency and phase demodulation, carrier and clock recovery, and frequency synthesis [1]. Its implementations range from discrete components to fully integrated circuits and even to firmware or software. Often the PLL is a highly critical component of a system, as for example when it is used to derive the on-chip clock, but as of this writing no definitive single-event upset (SET)/single-event transient (SET) tolerant PLL circuit has been described. This chapter hopes to rectify that situation, at least in regard to PLLs that are used to generate clocks.

Older literature on fault-tolerant PLLs deals with detection of a hard failure, which is recovered by replacement, repair, or manual restart of discrete component systems [2,3]. Several patents exist along these lines (6349391, 6272647, and 7089442). A newer approach is to harden the parts of a PLL system, to one degree or another, such as by using a voltage-based charge pump [4,5] or a triple modular redundant (TMR) voted voltage-controlled oscillator (VCO) [6]. A more comprehensive approach is to harden by triplication and voting (TMR) all the digital pieces (primarily the divider)

Au: "voted" seems out of place here.

K11326_C012.indd 293 1/29/10 3:38:57 PM

of a frequency synthesis PLL [7], but this still leaves room for errors in the VCO and the loop filter.

Instead of hardening or voting pieces of a system, such as a frequency synthesis system (i.e., clock multiplier), we will show how the entire system can be voted. There are two main ways of doing this, each with advantages and drawbacks. We will show how each has advantages in certain areas, depending on the lock acquisition and tracking characteristics of the PLL. Because of this dependency on PLL characteristics, we will briefly revisit the theory of PLLs. But first we will describe the characteristics of voters and their correct application, as some literature does not follow the voting procedure that guarantees elimination of errors. Additionally, we will find that voting clocks is a bit trickier than voting data where an infallible clock is assumed. It is our job here to produce (or recover) that assumed infallible clock!

12.2 VOTING ASYNCHRONOUS SIGNALS

When voting synchronous signals, data are latched according to a clock edge and can be unambiguously voted either before or after the clock edge. There are two common ways of doing this, according to what is to be protected. Figure 12.1 shows the two methods.

On the left we have a triplicated functional unit (usually memory, but could be anything). A single voter removes errors introduced within any one of the units but does not protect against errors in the voter itself or in anything prior to the triplicated units. On the right everything is triplicated, including the voters, so all errors are removed. As long as two of the three strings have a correct result, processing will continue correctly.

When voting asynchronous signals, those such as clocks that are not synchronized by some other signal, it is possible to have two correct signals and still get an incorrect result. Suppose, for example, that "correct" means a signal of a given frequency, such as the output of a clock multiplier. In Figure 12.2, signals X and Y are correct, and signal Z is in error. But because X and Y are not perfectly in phase, Z is able to influence the vote this way and that, producing anomalous results for the voting result (***maj*** – top signal), which is of incorrect frequency and has some transitions that are much too fast. These extra transitions will violate timing constraints and produce unpredictable errors if they occur on a clock signal.

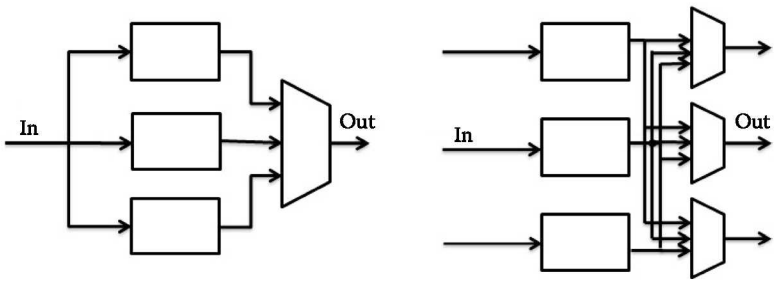

FIGURE 12.1 Single versus triple voters.

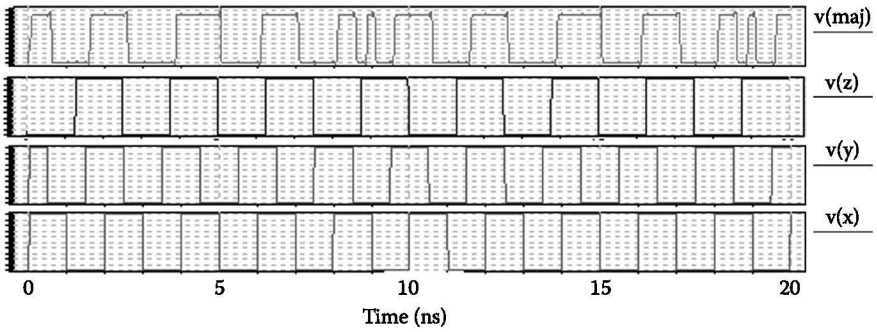

FIGURE 12.2 Phase-induced voting error.

While this example has been exaggerated for illustration, such an error can occur even for a small phase difference between the two correct signals. To guarantee that extra-transition errors (phase-induced voting error) will not occur, the phase difference between the two correct signals has to be smaller than the minimum pulse width to which the voters will respond!

The voting guidelines we have so far may be summarized as (1) use a triple voter configuration to protect against errors even in the voters, and (2) design your PLL so that redundant units will operate closely enough in phase that phase-induced voting errors will be avoided, as for example during the period when one unit is recovering from an SEU/SET and running temporarily at a different frequency or with glitches in its output. One more guideline is needed. What do you do if ultimately you wish to get one reliable output, such as one system master clock? In this case you must eventually rely on a single voter (though you can still use triple voters internal to your PLL system). The only type of voter that does not have a single SEU/SET susceptible point of failure is a heavily overdriven force voter, or conflict voter, that uses many gates to drive a single node. These gates should be spread out so that one single event will not strike several of them, and they should be driven from independent sources, such as a triple of prevoters. A rather elaborate example I have used is shown in Figure 12.3.

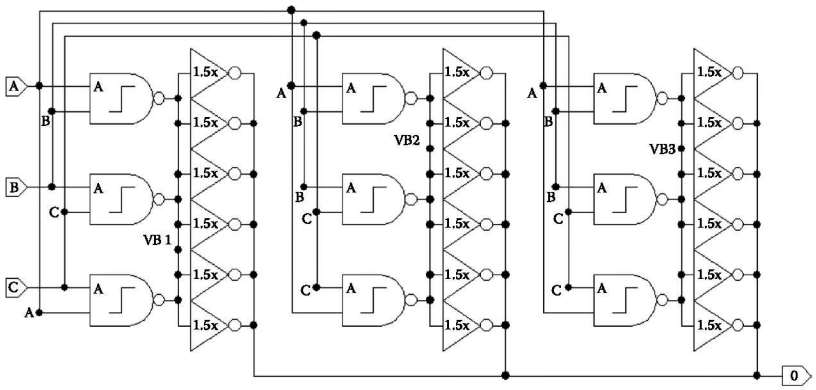

FIGURE 12.3 Force voter for consolidating triple to single string.

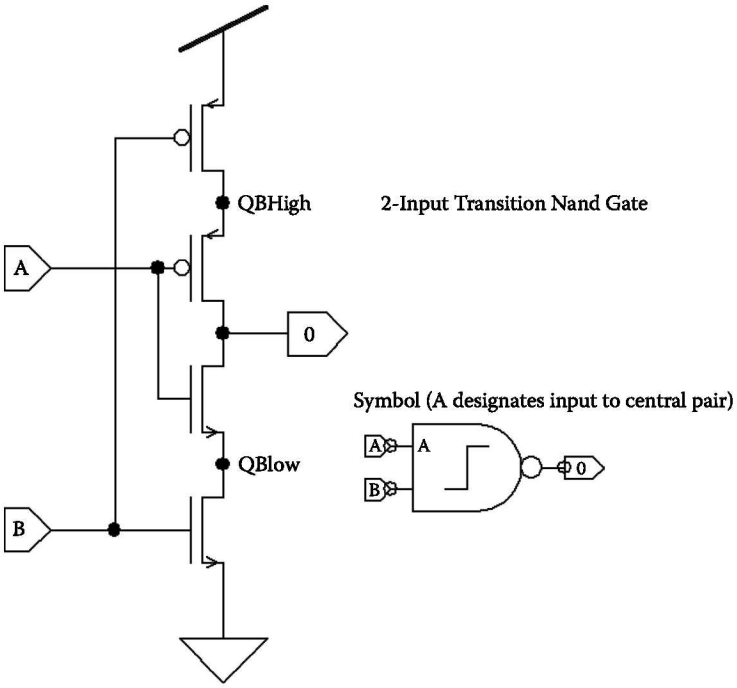

FIGURE 12.4 Transition nAnd gate.

The special symbol in Figure 12.3 is not a NAND gate. It is a transition nAnd gate (TAG) [8,9], also sometimes called a guard gate [10,11], broadly useful in radiation-hardening-by-design (RHBD) technology. The circuit for it is shown in Figure 12.4.

12.3 STABLE PLLS THAT MINIMIZE PHASE-INDUCED VOTING ERROR

A PLL is a feedback circuit that measures the phase of a signal compared with some reference and attempts to correct the phase of a local oscillator to match the reference. The local oscillator is frequently a VCO but may also be a numerically controlled oscillator, and in signal processing applications the whole PLL may be implemented mathematically in firmware rather than using actual components. But when a PLL must operate very fast and produce the clock on which digital logic depends, there is no alternative but to implement it directly in hardware.

An analysis of charge-pump PLLs by Gardner [14] points out several stability issues. First, the continuous-time approximation used for the analysis is not valid if the PLL loop bandwidth is high, and this introduces stability problems. For the fastest recovery from SEU/SET tolerance we will want the highest bandwidth practical. For clock generator PLL applications, it turns out it is very practical to increase loop bandwidth. For frequency synthesizer PLL applications, used to generate channel frequencies for communications systems, higher loop bandwidth

Au: Refs 12 and 13 need to be cited before ref 14. Please revise.

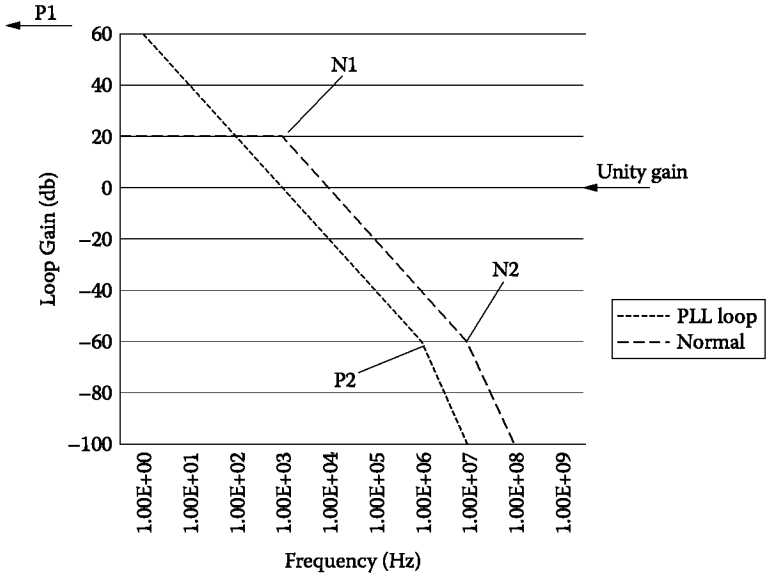

FIGURE 12.5 PLL versus normal feedback loop bode plot.

is not so practical. We will examine redundant PLL architectures appropriate to both situations.

Second, frequency synthesis PLLs need to have second- or third- or higher-order loop filters to reduce ripple on the VCO control voltage, V_{ctl}, caused by charge-pump operation and by the workings of fractional-N and sigma-delta frequency dividers and associated compensation systems. Whereas second-order analog PLLs are unconditionally stable, a second-order charge-pump-based PLL is a sampled data system and is unstable with high loop gain. The characteristics of VCOs currently preferred for high speed PLLs, which will be described in the next section, virtually guarantee excessively high gain unless the designer takes careful steps to ensure otherwise. With a third-order loop the situation is even worse. We will consider the most effective ways to manage loop gain.

Because of the additional pressure toward instability due to the requirements of an SEU/SET tolerant PLL, we will briefly review PLL stability and introduce nonlinear circuit considerations.

For any feedback loop to be stable and not oscillate, the feedback must be negative at all frequencies for which the feedback loop gain is equal to or greater than unity. PLLs are rarely completely stable. Their residual instability, the reasons for which we will explore herein, shows up as continual oscillation in frequency, or phase, of the local oscillator. This residual instability is a source of excess phase jitter and can impair the ability to synchronize redundant PLLs in a fault tolerant circuit, and the design techniques often used to combat it (lower bandwidth loop filters) can slow and interfere with recovery from an SET or SEU.

Stability of a feedback circuit is often understood by means of a Bode plot, such as in Figure 12.5. Loop gain is asymptotically plotted as a function of frequency.

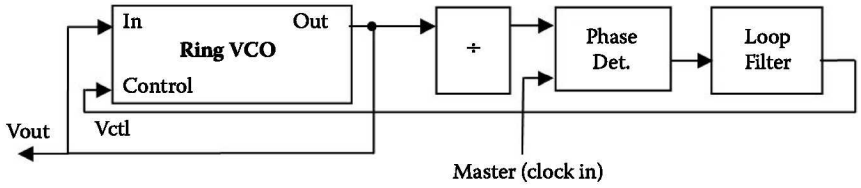

FIGURE 12.6 PLL block diagram with divider for frequency synthesis.

In the case of the PLL, this is the frequency with which the loop control voltage varies, not the frequency of oscillation of the VCO. The two are related of course by the transfer function of the VCO, which in the circuits we will be using is highly nonlinear.

First consider a normal operational amp (op amp) feedback loop, with loop gain represented by the dashed line. It is shown with two example poles in the loop response, N1 and N2. A 90 degree phase shift is associated with each pole. As long as no more than one pole is above the unity gain line, the circuit should be stable. If not naturally the case, this is often arranged by use of a Miller effect equalization capacitor to move one pole lower in frequency than any others.

A PLL, on the other hand, does not control the same thing it measures. It measures phase, but through the VCO it controls frequency. Phase is the integral of frequency. Therefore, every PLL has an unavoidable pole at zero frequency. You cannot move any other pole below it! This is illustrated by the dotted line, with poles P1 (infinitely off to the left on this logarithmic frequency scale plot) and P2. We have optimistically shown P2 below unity gain, but that is often difficult to arrange.

Unfortunately, determining gain for a highly nonlinear circuit is problematic. The best way is usually to run a transient simulation in Spice to see if the loop is stable. The point of this discussion is that one must be careful in designing the loop filter in a PLL. Below is a high-level block diagram level view of a PLL clock multiplier, in Figure 12.6.

Phase or phase-frequency detectors usually output a series of pulses, not a nicely behaved analog signal, so the loop filter must be introduced to smooth this signal. Otherwise, the VCO would vary between some very high and very low frequency, and its output would be unusable. The loop filter determines the bandwidth of the PLL [1]. Here we get conflicting requirements. For recovery from SETs and SEUs, the loop filter should have a high bandwidth so that the PLL performs like a tracking PLL and rapidly resynchronizes after an upset. But in clock or synthesis applications, a narrow frequency range is desired, meaning a low filter bandwidth. This is so that the VCO deviation over the counter cycle of the frequency divider will be small. In a fractional-N or sigma-delta PLL, the VCO deviation over several counter cycles must be small. A low filter bandwidth slows acquisition, either initially or after an upset. Additionally, the loop filter forms a second pole in the loop gain, and a low bandwidth loop filter moves that pole to the left on a Bode plot, toward a position of higher gain on the PLL's unavoidable pole-at-zero loop response characteristic.

Sometimes PLLs are designed with a tight (low bandwidth) loop for normal operation, and a separate means for initial "acquisition" of the target signal (the master

clock, in the case of Figure 12.6). We do not recommend this for SEU/SET tolerant PLLs, because an SET or SEU can cause an unplanned reacquisition at any time. Having a separate means for acquisition might be possible but is difficult to verify for every case.

The requirement for unplanned reacquisition at any time implies that we should use a phase-frequency detector (PFD), not a pure phase detector (e.g., XOR gate, traditional frequency multiplier). Phase-only detectors often are tricky to design for initial or reacquisition, especially when using tight loop filters. The reason is that without intrinsic frequency information, the phase-only detector can slip phase repeatedly, and a tight loop filter will average the varying output of the phase-only detector and produce false lock. The tighter the loop filter, the closer the false lock frequency can be to the desired frequency. It does not have to be at a harmonic of the desired frequency.

A frequency synthesis or clock generator PLL can easily find itself operating in the unstable region. The tight loop filter needed to control the frequency of the synthesized signal, coupled with the inherently high gain of some high-speed VCOs (which we will describe in the next section), places their loop filter pole above unity gain. If the instability is small, they work anyway. A PFD is not a pure phase detector, so to some extent the unfortunate pole-at-zero is eliminated, but not completely. But over the region where the PFD functions as a phase detector, basically when the PLL is "locked" and tracking its input, the PLL control loop oscillates, producing unwanted phase jitter. If this source of jitter can be reduced below the cycle driven jitter (from alternating pulses out of the charge pump), it is no longer a concern. But changing the loop parameters to make a PLL more SEU/SET tolerant can increase instability. Figure 12.7 shows a plot of this instability in an original case and with some modifications that a designer might use to minimize the jitter.

A quick way to understand what is going on in a PLL is to examine the loop control voltage (V_{ctl} in Figure 12.6), that is, the input to the VCO. In Figure 12.7 this is the dark wavy line. Ideally the line would be flat, indicating no variation in the

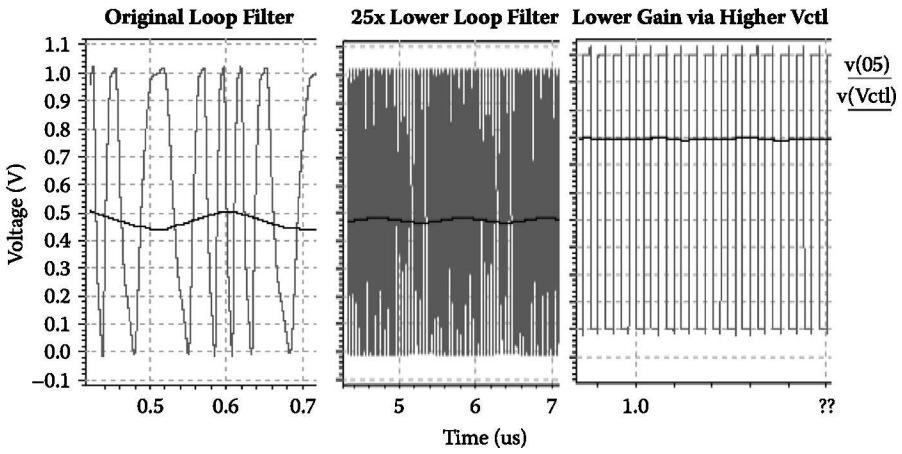

FIGURE 12.7 PLL instability management, comparing approaches.

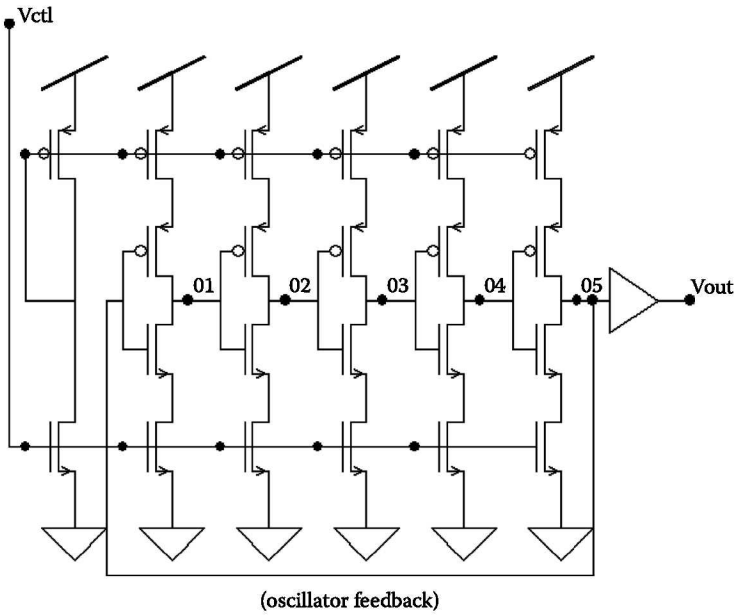

FIGURE 12.8 Current-starved inverter loop VCO.

frequency the VCO is requested to produce. A common type of VCO used in high-speed complementary metal-oxide semiconductor (CMOS) PLLs is the current-starved inverter loop, such as shown in Figure 12.8.

The transfer characteristic of this type of VCO is such that when V_{ctl} is near the threshold voltage, a very wide variation in frequency occurs for tiny changes in V_{ctl}. In other words, the voltage-to-frequency gain (often denoted K_{vco}) is very high near $V_{threshold}$. A small K_{vco} is beneficial in achieving low phase noise [15].

K_{vco} is not only high near $V_{threshold}$; it is rapidly varying (i.e., highly nonlinear), which in itself can cause stability problems [14]. The plots of Figures 12.7a, 12.7b, and 12.7c were obtained using the schematic of a well-designed PLL, but operating it at a lower than designed master input clock frequency so that V_{ctl} would be too low and the loop gain very high. The result is Figure 12.7a in which a steady oscillation in V_{ctl} produces a frequency instability that is unacceptable.

An alternative to the current-starved inverter VCO is a tank circuit. These typically have smaller K_{vco} but also more narrow tuning ranges. And the necessity of having an inductor makes them less desirable for fully integrated or redundant applications.

Figure 12.7b shows what happens if one uses the instinctive solution feedback circuit designers would apply, that is, to "compensate" or dramatically lower the pole that the designer can control, the loop filter cutoff frequency. This is lowered by a factor of 25 in the middle figure. It appears to help slightly, perhaps by a factor of two, but the acquisition and tracking characteristics are dramatically reduced, in fact by a factor of 25! So while this sort of works, it might conflict with SEU/SET recovery performance requirements. Why doesn't this technique work better? It is because the PFD is highly nonlinear. A large frequency deviation can have approximately the

same PFD output as a modest deviation. This makes the loop less sensitive than it should be to changes in the loop bandwidth.

Figure 12.7b shows what happens if instead of tampering with the loop filter, we lower the VCO gain. Since the gain, if we are using a current-starved inverter VCO, is determined by the bias of V_{ctl}, we must figure how to raise V_{ctl}. This can be done by making the VCO slower, so a higher V_{ctl} is needed to operate at the desired frequency. To achieve this, 2.5 pF of capacitance was added to nodes O2 and O3. Modifying at least two nodes, and an even number of them, assures a symmetric waveform within the VCO, and that one node is not operated far beyond its cutoff frequency. One could modify all nodes if one wished. Adding stages to the VCO is not a particularly effective alternative for raising V_{ctl}, because very many stages are required. Making the circuit larger also would increase its SEU/SET error cross section.

Adding capacitance raised V_{ctl} from around 0.46 V to 0.8 V, and the output frequency stability was vastly improved. This was accomplished without significant degradation of the acquisition and tracking characteristics of the PLL! It might seem to an experienced feedback loop designer that changing the loop filter or loop gain to have a given stability effect should be more equivalent. But due to the nonlinear nature of the PFD, changes to the VCO predominantly affect stability, not acquisition. When there is a frequency mismatch between the VCO and the reference input, the PFD outputs either low or high, with no indication of how low or how high. So, excess gain in the VCO increases the frequency error without increasing the tracking "force."

Figure 12.7d shows a charge-pump PLL in acquisition mode. Notice that V_{ctl} jogs around in an irregular way, due to the nonlinear interactions among the VCO, PFD, and loop filter. It may take a long time for this to stabilize. During acquisition, a frequency deviation is produced by the PLL to bring phase into a matching condition. At match, the PFD produces no output. The phase momentarily matches. But the frequency deviation persists until enough phase error accumulates to drive the loop back the other way. For stability, one must guarantee this process eventually damps out, which may require a very slow (low bandwidth) loop, not what we'd like for fast SEU/SET recovery. There are four parameters by which such a "slow" PLL might be judged: acquisition time, jitter, spectral purity, and phase error (with regard to the input reference). In the slow design, both acquisition time and phase error are traded for jitter and spectral purity. A small and varying phase error would be of no consequence in clock generator applications of a PLL. But in a redundant voted PLL, if the three component PLLs have independent phase errors, then phase-induced voting error can result.

Figure 12.7e shows a PLL in which the PFD is modified to always produce an output. This is done by presuming that if the phase is not ahead, it is behind, so the PFD is always outputting a signal, or "always on." Such an architecture reintroduces the drawbacks of older more linear PLLs (the phase error bias), but notice that it also has a more linear behavior without the chaotic jogs of the classic three-state PFD. It is also free of dead-zone nonlinearity near-zero phase error and is easier to analyze. Figure 12.7f combines VCO gain reduction with the always-on PFD to produce a very well-behaved loop. A clever designer could match the V_{ctl} of the desired

operating frequency of the VCO to the 50% duty cycle of the always-on PFD to create a PLL that would have no static phase error, low jitter, and also quick recovery from SEU/SET.

The faster a PLL acquires, the faster it will recover from an SEU/SET. Recovery time is important in a voting arrangement, because while one module is recovering, a second SEU/SET will cause an output error. Tight phase tracking is important to prevent phase-induced voting error. The key to making a good redundant fault-tolerant PLL is to start with a fast acquisition, low phase error, and single-string PLL design.

12.4 SEU/SET CHARACTERISTICS OF PLL BUILDING BLOCKS

Figure 12.6 shows five building blocks in a basic PLL for frequency synthesis. We describe the ring VCO in connection with Figure 12.8. The other four blocks can in principle be anything the designer chooses, but we discuss a representative design of each block here for purposes of understanding how their SEU/SET characteristics might affect our overall design.

12.4.1 Ring VCO

The SEU/SET characteristic of the Ring VCO is straightforward and somewhat unfortunate. It comes to an erroneous phase, from which it does not return on its own. Each stage of the VCO has four transistors, about as many as a typical digital logic gate. There are usually at least 5 stages, and sometimes 10 or more, so the cross section of the VCO rivals that of any other part of the PLL, such as the frequency divider. To make matters worse, the VCO is operated in current starved mode, which means that less current is available for charge clearing after an ion strike than would be the case in a high drive digital circuit.

Furthermore, an SET in any part of the VCO causes a phase displacement, whereas in a digital circuit only half the circuit is susceptible most of the time. This at least doubles the error cross section of the VCO. There are three effects in play. In a digital circuit with multiple input gates, the state of the logic ignores many of the inputs. For example, a NAND gate with one input low effectively ignores the second input and any error on the second input. The VCO stages do have multiple inputs (the signal and the control voltage), but neither of them is ever ignored.

Second, digital logic transistors that are in the ON or conducting state do not experience a state change when an ion strikes, because ion strikes only increase conduction and do not decrease it. If the excess charge is cleared by the time the state changes, no effect is noticed. Timing of the digital logic is important only insofar as it meets minimum timing requirements. However in the VCO timing of the delay through each stage is always critical. Even a slight change in timing due to extra time required to clear charge from an ion strike will result in clock jitter and possible timing violations in the target circuit served by the clock.

Third, digital logic is sampled by the clock, and errors are counted only if they persist through a clock edge. But the clock, and thus the VCO, is not sampled by anything, and errors occurring at any time may result in system errors.

The first goal of an SEU/SET tolerant design is to quickly return the VCO to a correct state. This should be done regardless of the method of eliminating errors. If, for example, one of three VCOs lingers in an incorrect state, the chances it will cause phase-induced voting error increase. If it lingers long enough, there may even be a second SEU/SET in another part of the circuit, causing an error.

There are several ways of quickly returning a VCO to a correct state, and in later sections we will explore two of them in detail. One method is to use PLL parameters that produce fast acquisition and tracking. Presumably this will also result in fast correction. Fast correction will not prevent an error on the output and so must be used in combination with some other scheme for eliminating errors. But it will prevent the VCO from lingering in an incorrect state and thus minimize the probability of phase-induced voting error, or accumulation of a second error. The problem with fast acquisition and tracking is that, as described already, it is often at odds with frequency and phase stability, or tightness of tracking.

Another method of quickly returning a VCO to a correct state is to have three VCOs and vote them [12]. This works so quickly that it also eliminates errors. However, it eliminates only errors from the output of the VCO, if taken from the voter output, not from other parts of the PLL. Still another method is to vote only the output of the entire PLL (with two other identical PLLs) and to allow the feedback loop to resynchronize any PLL that experiences an SEU/SET induced error. We will explore both of these in a subsequent section.

12.4.2 Frequency Divider

The frequency divider, needed when the PLL is to provide frequency synthesis, is a digital state machine, and an SEU putting it in a different state is probably the most disruptive of any SEU/SET effect in a PLL. It is possible of course to vote every bit in the state machine [7]. Or one can allow the feedback loop of the PLL to eventually resynchronize the divider by reacquiring lock on the master clock input. In either case, a divider that minimizes SEU susceptibility is a good idea, such as a fully synchronous design.

It is very important to note that if the frequency divider is protected by voting, this cannot be done using the same components that otherwise participate in some other PLL voting scheme. For example, if three complete PLLs are voted, differences in the frequency dividers are essential to allow a failed PLL to resynchronize itself with the others.

12.4.3 Sigma-Delta Fractional-N Frequency Dividers

Sigma-delta or fractional-N frequency dividers use a dithering scheme to divide the VCO frequency by a sequence of integers that averages to a noninteger value. This creates two problems for the designer of a fault tolerant PLL. First, there is a lot more logic in the frequency divider, which must be protected from SEU/SET. This is inconvenient but not conceptually difficult. If voting is used, care must be taken that no internal state is left unprotected and allows an error to persist.

Second, either the variability of the output of the dithered frequency divider must be averaged over a longer period, implying a lower bandwidth loop filter and leading to problems we have already discussed, or compensation circuitry must be used to tune out the expected variations. Compensation circuitry can be analog in nature, getting involved with the charge pump, and can be both more susceptible to SET and more difficult to protect. A strategy of protecting an entire PLL is advantageous for such a situation. In the case where this circuit is protected by voting, care must be taken to vote every internal state variable so that there are no persistent errors.

12.4.4 Phase-Frequency Detector

There are many types of phase detectors and phase-frequency detectors [1,13]. For clock recovery PLLs, which examine data transitions and synthesize the implied clock, a PFD that tolerates missing transitions is required. For frequency synthesis, we are already in the position that the PLL is generating many more transitions than are in the master clock and, so for synthesis, PFDs are desired that use every clock edge, both leading and trailing, and produce immediate correction signals if the edge is leading or lagging. In the interest of setting a manageable scope for the current discussion, we limit ourselves to the second type. These are usually three-state or higher-logic circuits. While better acquisition performance can be obtained with complex higher-state PFDs, the additional states also increase SEU susceptibility and increase the difficulty of resynchronization. An ordinary three-state PFD as shown in Figure 12.9 always resets itself when a clock edge has occurred on both inputs.

For most radiation-tolerant applications, we would use fully synchronous flip-flops. However, this PFD circuit works only with an asynchronous reset. The circuit of Figure 12.9 updates on leading edges of master and slave signals. One source of phase jitter in a frequency synthesizer is the PFD update cycle, because V_{ctl} will typically vary from some minimum to maximum value between PFD updates. The update cycle can be cut in half, reducing phase jitter, by using two PFD circuits and negating the inputs to the second one so that it updates on trailing (falling) edges.

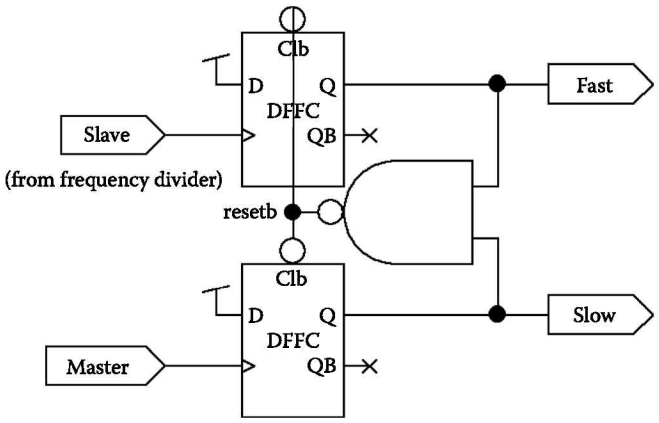

FIGURE 12.9 Single-edge three-state PFD.

As soon as both flip-flops have triggered (i.e., an edge is detected on both input signals) the flip-flops are reset, clearing any SEU condition.

12.4.5 Charge Pumps

There is a bit of an overlap when both the signals "fast" and "slow" are high, because of the time it takes the reset to operate. In a high-speed PLL this can be a significant error factor. If perfect charge pumps are used, in theory the "fast" and "slow" signals each result in a fixed current pulse into the loop filter and cancel out. But current-oriented charge pumps are relatively more susceptible to SEU/SET than a voltage-oriented charge pump (voltage with a high impedance switch) [4]. The larger number of transistors in a near-ideal current pump, and their lower drive strengths, increase both the exposure to SETs and the time required for recovery from SETs.

In the case of the voltage-oriented charge pump, the overlapping "fast" and "slow" signals do not exactly cancel, and phase error is produced. This can be eliminated by using pulse trimmers to reduce the length of these two signals by exactly the amount of the reset delay. Figure 12.10 shows a trimmer circuit that uses the same flip-flop reset to time the amount of trimming. Figure 12.11 shows the complete dual-edge PFD with trimmers and voltage-oriented charge pumps (switches are minimum-size pass gates).

A good bit of the literature on precision PLL design, such as frequency synthesizers for communication circuits, depends on a sophisticated current-based charge pump with precisely matched currents. There are a couple of approaches for dealing with this situation. One is to simply use the strategy of protecting the entire PLL, as we have been advocating, rather than its parts. SET recovery time will not be fast, but there will be errors only if a second SET occurs before the recovery from the first is complete. While this would be disastrous in a clock circuit, it is probably acceptable in a communication circuit where the communication protocol provides other means of handling errors. In other words, SEU/SET performance of frequency synthesizers is less critical than for clock generator circuits.

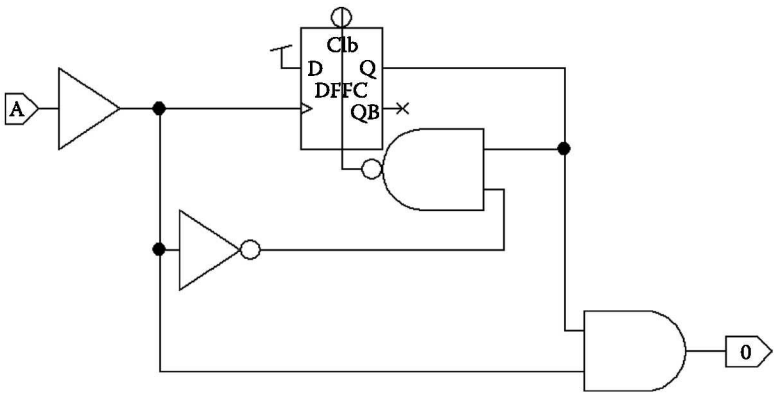

FIGURE 12.10 Reset pulse trimming circuit.

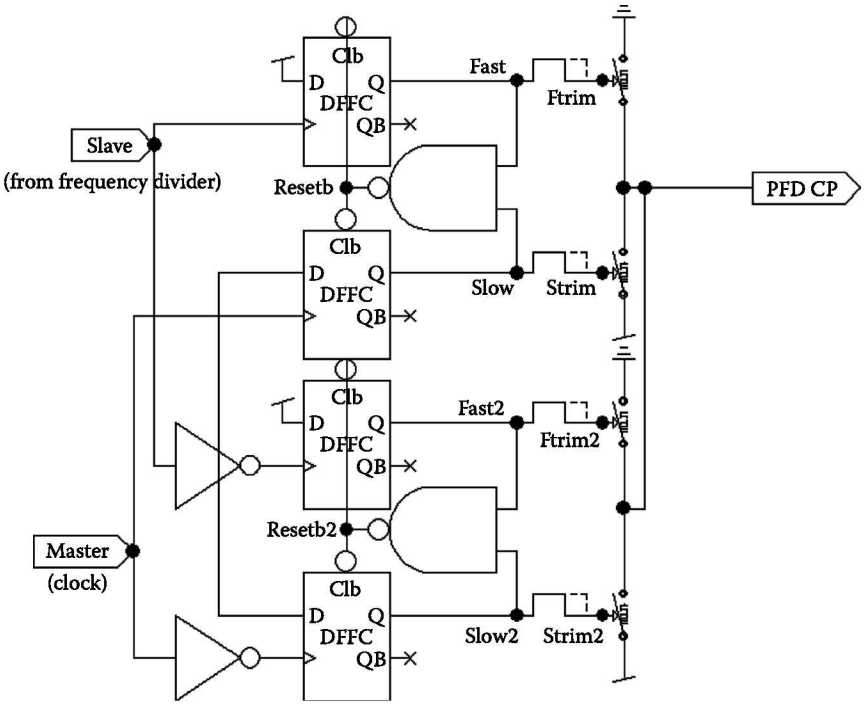

FIGURE 12.11 Dual-edge PFD with voltage-based charge pump.

A second strategy is to address issues with the voltage-based charge pump. Unbalanced charge injection can be addressed by V_{ctl} tuning as already described, although this might be hard to make process-independent. Power supply noise is also a commonly voiced concern for voltage-based charge pumps (though also for current-based pumps). Lee and Wang [16] have shown significant benefits from using separate regulators for the VCO and the charge pump.

12.4.6 Loop Filter

The last block of the PLL is the loop filter. As emphasized, the ideal loop filter would not be anything more than a single-pole RC filter. Second- and third-order loop filters will lead to longer acquisition time and also longer recovery times from SET/SET errors. With the description in Section 12.3 of how PLLs can be stabilized, one may be able to solve phase jitter problems by modifying the VCO gain. If necessary, the always-on PFD technique could be used. However, as with the charge pump, a large amount of technical literature would have to be disregarded to follow our most aggressive recommendation. We caution only that, when using a higher-order filter, make sure to examine the SET recovery time. If an SET disturbance causes milliseconds of thrashing around, consider reducing the VCO gain.

Another consideration is simulation time. If you are designing a nonredundant PLL, it is sufficient to get one correct simulation for each frequency of operation, and

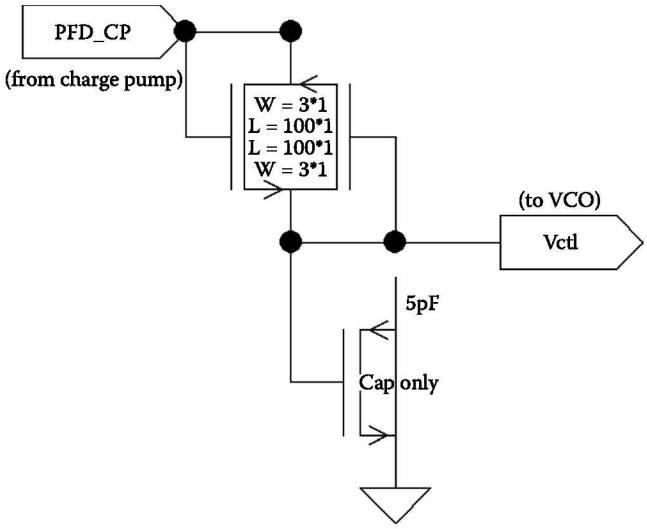

FIGURE 12.12 Loop filter.

you are done. But when designing a fault-tolerant PLL, it is necessary to consider the response to a variety of faults, increasing the number of simulations. In each simulation, one must wait for acquisition, inject a fault, and wait for it to settle, so each simulation is longer. It is tempting to incompletely verify the design in such a situation. The techniques we have outlined to reduce acquisition time will greatly speed up verification by allowing shorter simulations.

Ideally the RC loop filter would be just resistors and capacitors. In practice, in a CMOS circuit you can obtain an approximate RC filter by using resistor-connected field-effect transistors (FETs) for the resistor and the gates of FETs for the capacitor. This has the advantage of being process-independent and relatively scalable, possibly requiring no change when moving to new processes.

Figure 12.12 shows such a loop filter. It is important to use a symmetric pair of resistor connected FETs connected in opposite directions to avoid a nonlinear preference for charging the filter in one direction or another. This circuit uses a width/length (W/L) of 3/100, giving quite a high impedance. Because of threshold voltages, the circuit will not quite charge to either supply rail. For the capacitor, enough FETs with large area gates are connected in parallel to make whatever value is needed.

Au: Long form of W/L ok?

It would seem that an SET in the loop filter might be significant, but in practice it might be of less consequence. If there is a lot of charge stored on the capacitor compared with the charge generated by an ion strike, an SET has less effect. Any effect it does have will be eliminated by the redundant voting techniques that will be used to protect other parts of the circuit.

12.5 APPLYING REDUNDANCY TO PLLS

It is possible to improve or vote or otherwise mitigate SEU/SET for the individual components of PLLs. This usually leaves some component, such as the loop

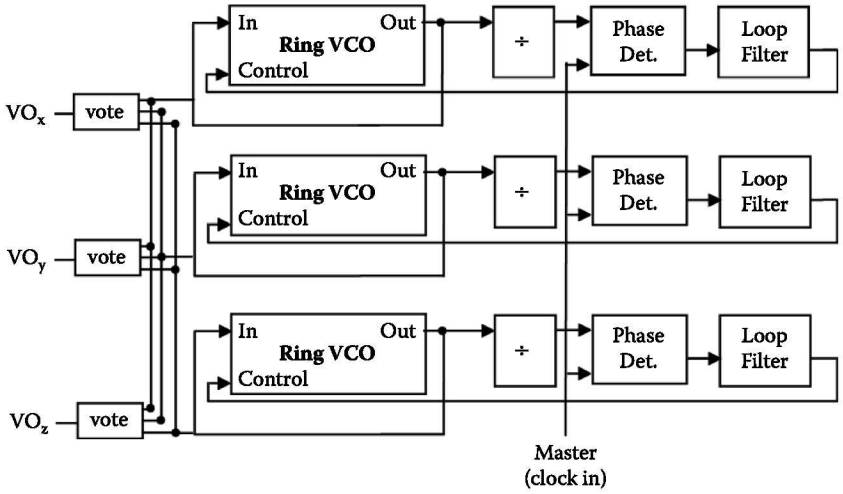

FIGURE 12.13 Output voted PLL.

filter with an analog output, unprotected. It is the purpose of this chapter to propose comprehensive treatments. These can be simpler, since a good PLL design is merely repeated three times and voted, with a single voter instead of many voters. But there are tricks and considerations to such an arrangement that require a little more examination.

12.5.1 Output-Only Voting Method

The first and simplest arrangement, shown in Figure 12.13, is to vote only the output. This arrangement relies on the individual PLLs to resynchronize themselves with the master input clock after an upset condition. This is a function they are already designed to do when they are turned on. No PLL ever looks at what is going on in another PLL, but only at the input signal. So if it works at power up, it will also work following an SEU/SET. The question with this design is whether it will work well enough to avoid phase-induced voting errors. And that will largely depend on whether you have designed a good PLL!

When you run a SPICE simulation on three identical PLLs with identical starting conditions, the results will appear identical. But real PLLs will not be identical. However, when you disturb one of the PLLs by injecting a simulated SET, then as it recovers you will see what the relative tracking of your PLL design might be in the real circuit. Figure 12.14 shows a plot of two PLLs.

The light gray line is the master clock, and the darker highlighted waveforms at 8× frequency are the two PLL outputs. The two lines at about 0.65 V are the two V_{ctl}'s. Each horizontal axis tick mark is about 1 ns, so the difference in the two PLL outputs is a small fraction of a nanosecond, probably around 100–150 ps. A large voter such as shown in Figure 12.3 will be slow enough in this technology (90 nm) that voting these signals should not produce any phase-induced errors. However, if it gets any worse, it won't work.

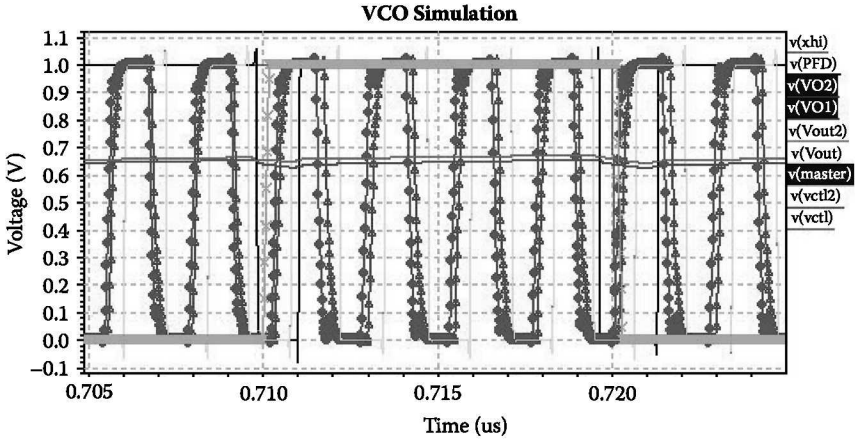

FIGURE 12.14 Comparing outputs of identical PLLs after disturbance.

The thing that might make it get worse is trying to fix a stability problem by dramatically lowering the loop filter bandwidth. This makes the control loop very slow, and it could be a long time, if ever, before the outputs line up again. Whether you fix stability problems by lowering the VCO gain or with the loop filter, you should carefully check the worst-case disturbances to make sure your PLL outputs will line up again. You can also check for the effect of process variation by changing the W/L of a bias transistor in one of the ring VCOs by an amount of half a lambda, or half the minimum feature size of your process. This model approximates the worst-case process parameter variation that you would normally see.

12.5.2 The VCO Voting Method

As mentioned earlier, the VCO itself can be voted, which guarantees that all three VCOs quickly return to lock step. So what do you then do about the rest of the PLL?

As far as the PFD, charge pumps and loop filter are concerned; whether you have to do anything depends on the amount of jitter caused by an SEU/SET on these components. The effect of an SEU/SET on any of these components is some delta to V_{ctl}, which will result in an erroneous delta to the VCO frequency, but not any discontinuity in the output. It must be determined whether the worst case delta to VCO frequency, and resultant phase deviation, is within desired operating limits of the PLL.

The PFD will be cleared out every master (input) clock cycle. The worst case is the delta frequency caused by one master clock cycle. Adjustments can be made by changing charge-pump current, loop filter time constant, or VCO gain. While the PFD could be protected by a complicated voting scheme, it is likely these other adjustments will suffice.

An SET in the charge pump could last longer than a master clock cycle, since tiny high impedance transistors are used. This is difficult to determine directly by heavy-ion testing. It can be estimated by using laser testing or SPICE analysis, but in either

case a calibration based on the transistor sizes used is advisable. Once the longest error state is determined, analysis and adjustment proceeds in the same manner as for the PFD.

An SET in the loop filter makes a small change in the charge stored in the loop filter. Erroneous charge on the loop filter capacitor will eventually be eliminated by operation of the PLL control loop. It will produce a small change in V_{ctl} and, in turn, a small change in VCO frequency. The problem is that it is difficult to quantify.

So with the PFD, the charge pump, and the loop filter we have three progressively more difficult to quantify effects on V_{ctl}. Here is the dilemma. The voted VCO behaves like a single VCO, not three independent ones as in the voted output PLL. If we are to have a single V_{ctl} to control it, this V_{ctl} will either be subject to SET errors, or we will have to design a complex analog voter, which will itself be extremely challenging. If we replicate the PFD, charge pump, and loop filter and produce three V_{ctl} signals and separately drive the three VCOs, then there is no independent correction of errors by loop action, because with the voted VCO it is just one loop. Errors in the PFD will clear in one cycle. SETs in the charge pump will be removed by charge collection processes. Errors in the three loop filters will eventually decay away if the filters are passive RC circuits. So replicating these components and producing three V_{ctl} signals might be a feasible option. It seems to lack the positive assured error removal of the voted output PLL design, but it is not particularly susceptible to phase-induced voting error.

The problem with the frequency divider is different. An SEU on the frequency divider will never be cleared unless you do something special to clear it. Because the VCO is voted internally, the control loop of the affected PLL will never be allowed to correct the divider. The PLL control loop and the VCO voting will be fighting each other. This condition will persist, and the PFD of the failed PLL will constantly be trying to adjust its loop filter to "make up" the lost phase, stored in the divider, but it will never be able to. With only two good remaining PLLs, the next time one of them experiences an SET/SEU the circuit output will be in error.

So to make the VCO voting method work, you have to provide some means of correcting the dividers. The most obvious method is to replicate and vote the dividers. The internal signals of the divider must be voted so that the voting corrects their internal state. If only the outputs are voted, an erroneous count will simply be perpetuated. It is possible to dream up other schemes, such as forcing the three dividers to reset all at once, but these would become complicated. A simple scheme is to drive each divider with one of the three VCO outputs. The output of the dividers can either be from a triple output voter (preferred), driving replicated PFDs and so forth as previously described, or a single output driving a single-string PFD, charge pump, and loop filter if it can be determined that an SET impact does not disturb the VCO frequency by more than the desired design specification.

The advantage of the VCO voting method is not having to worry about phase-induced voting error. The VCO voting method can be used whenever it is too difficult to design robust PLLs that will remain in lock in spite of process parameter variations and will return to lock after a disturbance. The drawback of the VCO voting method is the difficulty of verifying that SETs in the PFD, charge pump, and loop filter are within tolerance, or the ambiguity of the lack of positive removal of

errors if these components are replicated without independent feedback correction. Simulations suggest that errors in replicated charge pumps and loop filters decay rather quickly, in tens or at most hundreds of nanoseconds, but this is has not been confirmed by heavy-ion testing.

12.6 CONCLUSIONS

We have seen that to design a fault-tolerant PLL, we must start with a good PLL design that will remain in tight lock to avoid phase-induced voting errors and will reacquire quickly so that the vulnerable time after one PLL has been upset is minimized. To design a good PLL has required a revisit of PLL stability theory and consideration of nonlinear factors, such as the differing effects of stabilizing by changing VCO gain versus changing the loop bandwidth.

We then examined two redundancy topologies that produce an entirely redundant and fault-tolerant PLL. The simpler one votes only the output and relies on the PLL feedback to resynchronize the failed PLL but is subject to phase-induced voting errors if the PLLs do not sync up tightly.

The more complicated method keeps the VCOs in lock step but requires other mitigation, especially of the frequency divider, to avoid a fight over control of the loop, which would result in making a transient error permanent.

For frequency synthesizer applications, the required phase error is likely already so small that phase-induced voting errors will not be an issue. These applications also are more likely to require a complex frequency divider of the fractional-N or sigma-delta type, which is tedious to mitigate for SEU/SET. So for frequency synthesizer applications, the output voted PLL is highly recommended.

For clock generator applications, the phase error should be carefully reviewed for potential voting problems. If there are any, they can probably be eliminated by the techniques we have given, such as VCO gain reduction. In that case, again the simpler output voted PLL is recommended.

If you have a PLL design that you simply must use but that does not control the phase tightly enough for the simpler output voting method or that is sensitive to device parameter or other variations, then the more complicated VCO voting method should work, provided all state variables in the frequency divider are voted, and the remaining components have no source of persistent error following an SET.

This analysis has been done for a frequency synthesis PLL of the sort needed for an on-chip clock multiplier. Similar principles can be applied to other types of PLLs. The output voted PLL has been fabricated and tested by the author. Components of the VCO voted PLL have been investigated by Loveless at Vanderbilt [5,6,12], and a VCO voted PLL has been simulated by the author.

REFERENCES

1. Wolaver, D.H., *Phase-Locked Loop Circuit Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.
2. Van Alen, D.J. and Somani, A.K., "An All Digital Phase Locked Loop Fault Tolerant Clock," *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 3170–3173, 1991.

3. Kessels, J.L.W., "Two Designs of a Fault-Tolerant Clocking System," *IEEE Transactions on Computers*, vol. C-33, no. 10, pp. 912–919, Oct. 1984.
4. Loveless, T. D., Massengill, L. W., Bhuva, B. L., Holman, W. T., Witulski, A. F., and Boulghassoul, Y., "A Hardened-by-Design Technique for RF Digital Phase-Locked Loops," *IEEE Transactions on Nuclear Science,* vol. 53, no. 6, pp. 3432–3438, Dec. 2006.
5. Loveless, T.D., Massengill, L.W., Bhuva, B.L., Holman, W.T., Reed, R.A., McMorrow, D. et al., "A Single-Event-Hardened Phase-Locked Loop Fabricated in 130 nm CMOS," *IEEE Transactions on Nuclear Science,* vol. 54, no. 6, pp. 2012–2020, Dec. 2007.
6. Loveless, T.D., Massengill, L.W., Holman, W.T., and Bhuva, B.L., "Modeling and Mitigating Single-Event Transients in Voltage-Controlled Oscillators," *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2561–2567, Dec. 2007.
7. Nemmani, Anantha, N., "Design Techniques for Radiation Hardened Phase-Locked Loops," master's thesis, Oregon State University, A874174, August 2005.
8. Shuler, R.L., Kouba, C., and O'Neill, P.M., "SEU Performance of TAG Based Flip-Flops," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2550–2553, Dec. 2005.
9. Shuler, R.L., Balasubramanian, A., Narasimham, B., Bhuva, B.L., O' Neill, P.M., and Kouba, C., "The Effectiveness of TAG or Guard-Gates in SET Suppression Using Delay and Dual-Rail Configurations at 0.35 μm," *IEEE Transactions on Nuclear Science*, vol. 53, no. 6, pp. 3428–3431, Dec. 2006.
10. Mongkolkachit, P. and Bhuva, B., "Design Technique for Mitigation of Alpha-Particle-Induced Single-Event Transients in Combinational Logic," *IEEE Transactions on Device and Materials Reliability*, vol. 3, no. 3, pp. 89–92, Sept. 2003.
11. Balasubramanian, A., Bhuva, B.L., Black, J.D., and Massengill, L.W., "RHBD Techniques for Mitigating Effects of Single-Event Hits Using Guard-Gates," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2531–2535, Dec. 2005.
12. Loveless, T.D., Massengill, L.W., Bhuva, B.L., Holman, W.T., Casey, M.C., Reed, R.A. et al., "A Probabilistic Analysis Technique Applied to a Radiation-Hardened-by-Design Voltage-Controlled Oscillator for Mixed-Signal Phase-Locked Loops," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3447–3455, Dec. 2008.
13. Abramovitch, D., "Phase-Locked Loops: A Control Centric Tutorial," *Proceedings of the 2002 American Control Conference*, vol. 1, pp. 1–15, 2002.
14. Gardner, F., "Charge-Pump Phase-Lock Loops," *IEEE Transactions on Communications*, vol. 28, no. 11, pp. 1849–1858, Nov. 1980.
15. Ti, C.-L., Liu, Y.-H., and Lin, T.-H., "A 2.4-GHz Fractional-N PLL with a PFD/CP Linearization and an Improved CP Circuit," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.1728–1731, May 18–21, 2008.
16. Lee, T.-J. and Wang, C.-C., "A Phase-Locked Loop with 30% Jitter Reduction Using Separate Regulators," *VLSI Design*, vol. 2008, Article ID 512946, 2008.