CHAPTER 8

Clock and Data Recovery Circuit

At the output of the limiting amplifier, the amplified data signal with sharpened data edges is available for further processing, but unique interpretation of the received signal requires timing information. Serial communication links do not provide a synchronization signal on a separate channel and therefore the receiver must rely on the extraction of the timing information from the data stream. This *clock and data recovery* process can be performed in a similar manner in optical communication, electrical serial links, hard drive read-out channels, as well as in some memory interfaces. In the latter field, advocates and opponents of the clock forwarding scheme still debate about the advantages and drawbacks of per-channel clock recovery circuits.

Proper denomination of the whole synchronization process would be *clock recovery and data retiming*, better describing the behaviorally independent operations. Indeed the recovered clock is used to re-sample the incoming (or sometimes delayed) data to provide proper timing information, i.e. synchronicity, for the following blocks (Figure 8.1). However, in many implementations, the retiming operation is embedded in the clock recovery part to achieve improved circuit performance.

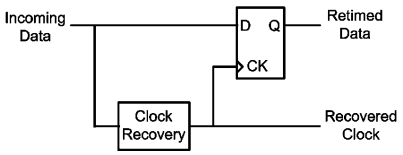

FIGURE 8.1. Clock recovery and data retiming principle

The major challenges of multichannel clock recovery circuit design have already been presented, namely, the achievement of low-power and low-area overhead with minimum impact on the circuit performance. First, an introduction to the important concepts in the design of clock recovery circuits is

P. MULLER, Y. LEBLEBICI,
CMOS Multichannel Single-Chip Receivers for Multi-Gigabit Optical Data Communications,
DOI 10.1007/978-1-4020-5912-4_8, © 2007 Springer

presented. Then, an overview of a variety of clock recovery topologies with their respective advantages and drawbacks allows the reader to fully understand the design choices operated later in this chapter. After discussion of appropriate topologies for multichannel clock recovery, a top-down design approach is presented which validates the jitter performance of the selected topology at the concept level, at the behavioral level and at the transistor level. Finally, the transistor-level design of the CDR is briefly discussed and characterization techniques are introduced.

8.1 Clock Recovery Principles

Historically, clock recovery was performed using resonant filter-based structures, either using discrete passive elements or resonant elements like surface acoustic wave (SAW) devices. In the era of systems-on-chip, requirements for small system form factors and minimum bill of material, the use of such off-chip components is strongly discouraged. In the long-haul market, where fiber-optic communications made their breakthrough first, phase-locked loop (PLL) topologies have become the mainstream solution, thanks to their capability of monolithic integration with a minimum number of external components, typically only one loop filter capacitor.

Beyond the fact that the PLL currently still dominates the field and is the most well-known clock recovery topology, we would like to use this CDR structure as a basis to the explanation of some concepts to the novice reader. A more detailed discussion of PLL-based clock recovery circuits can be found in [83]. For simplification, let us first consider a phase-locked loop with a periodic input signal (Figure 8.2). The three main building blocks are the phase detector (PD), the loop filter (LF) and the voltage-controlled oscillator (VCO). The phase detector compares the phase error between the input signal and the generated signal at the VCO output. The phase error is integrated in the low-pass loop filter and the resulting control signal tunes the VCO frequency, which in turn reduces the phase error.

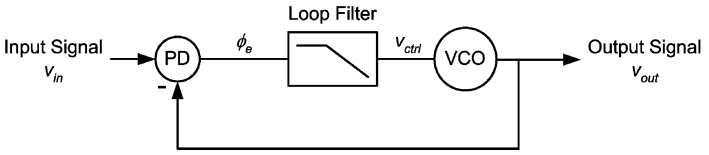

FIGURE 8.2. PLL with periodic input signal

The periodic input signal *f(t)* can be written as the sum of sine and cosine functions, namely, its Fourier series:

$$v_{in}(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cdot \cos(nt) + b_n \cdot \sin(nt)] \quad \text{(EQ 8.1)}$$

In most high data rate applications, we can neglect the higher order harmonics and focus on the fundamental signal, which can be written as:

$$v_{in}(t) = A \cdot \sin(\phi(t)) \quad \text{(EQ 8.2)}$$

The instantaneous value *f(t)* of a sine wave is defined by its peak amplitude *A* and its instantaneous phase $\phi(t)$, which can again be decomposed into a constant angular frequency $\omega_0 = 2\pi f_0$ and a time-varying phase angle $\theta(t)$. In phase and frequency modulation schemes used in wireless data transmission, this time-varying phase angle contains the information to be transmitted. In fiber optic links, it does not contain any useful information, but represents the phase noise due to channel and circuit imperfections like noise and limited bandwidth.

$$v_{in}(t) = A \cdot \sin(\omega_0 t + \theta(t)) = A \cdot \sin(2\pi f_0 t + \theta(t)) \quad \text{(EQ 8.3)}$$

In both types of applications, the phase-locked loop technique allows to extract the time-dependent phase information and track the input signal. The in-phase output signal v_{out}, locked to the input signal, guarantees a very small phase error $\phi_e(t)$ between the input phase $\phi_i(t)$ and the output phase $\phi_o(t)$. This situation is called *phase lock*. As frequency is the derivative of the instantaneous phase, constant phase error provides zero frequency error between input and output signals. The loop bandwidth being limited by the low-pass filter, these characteristics are not guaranteed for variations $d\theta(t)/dt$ at higher frequencies.

As already mentioned, the two-level amplitude modulation of the carrier in serial links can lead to data runs which do not contain any transitions. The low-pass behavior of the loop filter allows the PLL to memorize the signal frequency during these long runs without loss of phase lock.

The *capture range* defines the frequency span of the input signal for which phase lock can be achieved. For frequencies outside this span, the PLL will remain in a free-running situation. *Lock range* on the other hand defines the frequency span for which lock can be maintained once it is acquired. Without going into more details, these parameters depend on the characteristics of the PLL building blocks, among which the VCO's *tuning range*, defining the span of frequencies the oscillator can generate when driving its control voltage from one end to the other of the range.

In the following, the particularities of clock recovery from random data will be addressed. The input signal to be considered will be the NRZ data stream, while the output signal is the recovered clock signal.

8.1.1 NRZ Data Phase Detection

Combining the data retiming mechanism presented in Figure 8.1 with the PLL topology in Figure 8.2, we obtain the general PLL-based clock recovery structure presented in Figure 8.3.

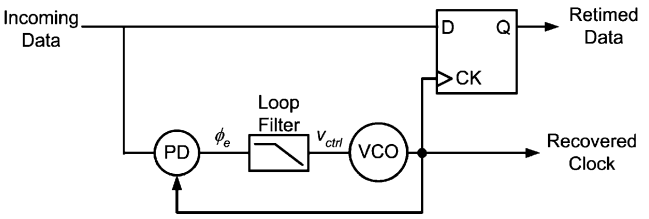

FIGURE 8.3. Overall PLL-based clock recovery structure

Several phase detector topologies with implicit edge detection on random data have been published. One well-known topology uses the incoming data D_{in} to sample the VCO clock signal CK_{VCO} (Figure 8.4, [83]). However, the flip-flop used in the phase detector and in the data sampler present unequal delays in the CK->Q and D->Q paths. This delay mismatch introduces a non-negligible static phase error at high data rates, resulting in reduced jitter tolerance at high frequencies.

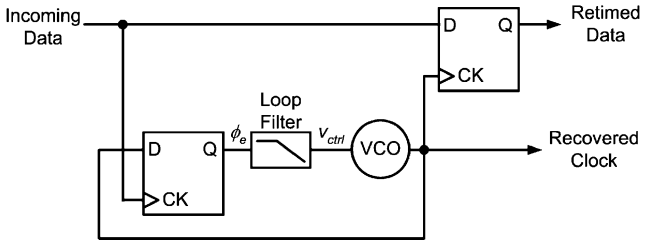

FIGURE 8.4. CDR based on a sampled clock topology

More convenient topologies for high data rate NRZ clock recovery have been published by Hogge [84] and Alexander [85]. Both phase detectors perform inherent retiming, canceling any static phase error resulting from possible path delays between clock and data.

8.1.2 Linear Phase Detector

The linear phase detector, introduced by Hogge, delivers output pulses with a duration that is proportional to the phase error between data and clock. The resulting PLL can be analyzed using linear system theory. The equivalent phase-domain model of the linear CDR is shown in Figure 8.5. The phase detector is represented by a gain block of value K_{PD}. The Laplace representation is used for the loop filter which also contains the charge pump, while the VCO is modeled using a gain factor combined with an integration. This integrative behavior is due to the fact that the signals are considered in the phase, whereas the characteristic tuned by the control voltage is the VCO frequency.

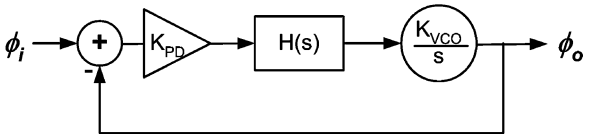

FIGURE 8.5. Equivalent linear model of the CDR

Charge pump topologies are commonly used in integrated PLLs to eliminate the large resistor required in the loop filter, changing at the same time the system behavior to discrete-time operation. Additionally, the use of a charge pump results in a second pole at the origin in the loop transfer function (the first being due to the VCO). A zero is thus required in the loop to guarantee the stability of the system. The resulting loop filter is shown in (Figure 8.6).

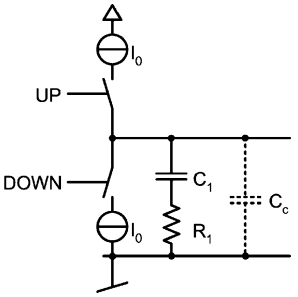

FIGURE 8.6. Second-order loop filter

A resistor R_1 placed in series with the main loop filter capacitor C_1 creates a zero at $1/(R_1C_1)$, but also converts the switched charge pump current into ripple on the control voltage at the output of the loop filter. This ripple can be attenuated by an additional shunt capacitor C_c, resulting in a third-order transfer function (Equation 8.4). This continuous-time representation of an inherently discrete-time circuit is of course based on the assumption that the loop bandwidth is much smaller than the frequency of operation [86]:

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{m\frac{2\zeta}{\omega_n}s^3 + (m+1)s^2 + 2\zeta\omega_n s + \omega_n^2} \quad \text{(EQ 8.4)}$$

In this equation, $m = C_c/C_1$, while the natural frequency ω_n and the damping factor ζ are defined as follows:

$$\omega_n = \frac{I_{PD}K_{VCO}}{2\pi C_1} \quad \text{(EQ 8.5)}$$

$$\zeta = \frac{RC_1}{2}\omega_n \quad \text{(EQ 8.6)}$$

Notice that in a 3rd order function, natural frequency and damping factor do not have the same meaning and do not characterize the loop behavior as they do for a 2nd order. Typical loop filters have $C_1 >> C_c$ ($m << 1$), which means that the third pole does not influence the peaking. With this hypothesis, the loop function reduces to a 2nd order function, where damping factor and natural frequency recover their well-known meaning (Equation 8.7).

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad \text{(EQ 8.7)}$$

As a side note, such a transfer function must exhibit peaking because the closed-loop zero is located at a lower frequencies than the poles. The SONET standard specifies the maximum jitter peaking in the transfer function as 0.1 dB to minimize jitter transfer. In short-distance communications where jitter transfer is not critical, the jitter peaking requirements are less stringent. Loop stability however, is and determines the maximum acceptable peaking.

The relationship between closed-loop bandwidth and natural frequency is given by:

$$\frac{\omega_c}{\omega_n} = \sqrt{2\zeta^2 + \sqrt{4\zeta^4 + 1}} \quad \text{(EQ 8.8)}$$

Provided that low peaking is achieved, i.e. the loop is sufficiently damped ($\zeta > 4$), the transfer function tends to its asymptotic 2nd order function and the closed-loop bandwidth is equal to:

$$\omega_c = 2\zeta\omega_n \quad \text{(EQ 8.9)}$$

The loop bandwidth of conventional PLLs directly determines the system's capture range. As we will see later (paragraph 8.2.1), there are several reasons in clock recovery systems to add a frequency acquisition path, which then also determines the capture range.

In any case, the PLL loop bandwidth determines jitter tolerance performance. As jitter transfer and generation are of no importance to short-distance communications, it is "sufficient" to design the loop bandwidth larger than the jitter tolerance corner frequency. It is easily understood that the loop tracks jitter at frequencies below its bandwidth, exhibiting decreasing jitter tolerance at higher frequencies, as previously shown in the related jitter tolerance specification. Jitter tolerance can also be determined by the VCO tuning range. In this case, exhaustively discussed in [87], a sort of frequency slewing is exhibited by the loop. This issue however only appears at higher jitter frequencies when using a VCO with a reduced frequency-tuning range or in circuits with very large-loop bandwidth.

8.1.3 Binary Phase Detector

At high data rates, it becomes increasingly difficult to generate pulses which widths are proportional to the phase error, as required in linear phase detectors. Lack of proportionality between the phase error and the pulse width leads to a dead zone in the phase detector characteristic. The resulting phase error in the loop reduces the jitter tolerance performance. One attempt to circumvent this problem is the use of binary phase detectors, initiated by J. D. H. Alexander and nowadays extensively used in multi-gigabit receivers. The output of this type of phase detector, also called *bang–bang phase detector* (BB PD) for its hard switching characteristic (Figure 8.7), only indicates whether the recovered clock signal is early or late with respect to the received data edges.

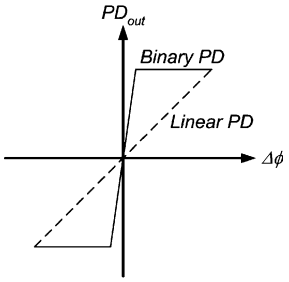

FIGURE 8.7. Bang–bang characteristic compared to a linear characteristic

Although bang–bang phase-locked loops (BB-PLLs) do not allow for linear analysis, some efforts have been undertaken recently to provide some mathematical relationships between the loop specifications and its parameters. While Razavi's work [88] considers both the linear in-lock regime and the binary regime at large phase error, we prefer to summarize here the analysis presented by Walker in [89], focusing on the binary regime of the bang–bang loop. By the way, most bang–bang detector implementation are in fact tri-state designs, which do not deliver any decision in absence of data transitions. This behavior optimally exploits the loop's memory effect during data runs. In the following, the term

bang–bang phase detector defines such tri-state detectors, as opposite to linear detectors, which themselves may also present tri-state implementations.

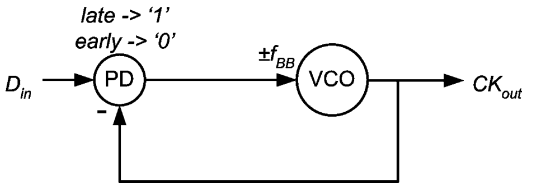

FIGURE 8.8. A first-order bang–bang PLL

In a first order bang–bang loop, shown in Figure 8.8, the VCO frequency toggles between f_0–f_{BB} and f_0+f_{BB}, where f_0 is the oscillator's free running frequency and f_{BB} the frequency step defined by the control voltage step and the VCO gain. The presence or absence of a variation of the control voltage during each cycle depends on the sign of the phase error. A first-order bang–bang loop thus never reaches the exact frequency of the incoming signal, but the average frequency minimizes the phase error. Phase lock is guaranteed as long as the input frequency does not exceed either selectable frequency of oscillation. The frequency switching between both ends of the tuning range results in additional *tracking jitter* (also called *hunting jitter*), which amplitude can be calculated as:

$$TrJ_{PP} = \frac{4\pi}{2\pi} \cdot \frac{f_{BB}}{f_0} = 2 \cdot \frac{f_{BB}}{f_0} \text{ [UI]} \qquad \textbf{(EQ 8.10)}$$

Low tracking jitter requires a relatively small tuning range of $2f_{BB}$. While a small tuning range affects capture and lock properties, it also limits the system's ability to track data jitter. In presence of sinusoidal data jitter of the form $\theta(t) = \frac{SJ_{pp}}{2} \cdot \sin(2\pi f_{SJ} t)$, the maximum jitter amplitude before appearance of a slewing phenomenon, called *slope overload*, is:

$$SJ_{PP} < \frac{2}{2\pi} \cdot \frac{f_{BB}}{f_{SJ}} \text{ [UI]} \qquad \textbf{(EQ 8.11)}$$

Interestingly, this result corresponds to the slope overload of a delta modulator used in early voice encoding schemes. It is valid for a periodic input signal, whereas the lower amount of data transitions in an NRZ data stream further reduces the tolerable amount of jitter.

Due to these limitations and the unsuitable trade-off between jitter tolerance and tracking jitter, a second order loop topology is preferred (Figure 8.9). The integral branch obviously improves jitter tolerance in combination with reduced frequency steps, resulting in less tracking jitter.

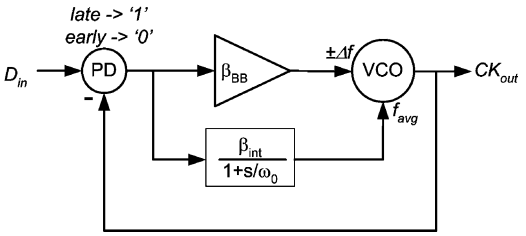

FIGURE 8.9. Second-order bang–bang PLL

The integral branch filters the PD output and adjusts the average VCO frequency with a gain β_{int}, performing low-frequency tracking. The bang–bang branch applies the unfiltered detector output with a gain of β_{BB} to the VCO and thus takes care of the high-frequency tracking. At very high jitter frequencies, jitter tolerance is still limited by the bang–bang step, but at lower frequencies, where more jitter must be tolerated, the integral loop takes over and tracks the low-frequency jitter. As a result, the bang–bang step can be reduced to keep tracking jitter low, while still achieving good jitter tolerance performance thanks to the integral path. The interested reader is referred to [89] for a detailed description.

8.2 CDR Topologies

Power consumption and silicon area are key factors to the success of a multichannel clock recovery circuit, while the influence of interchannel cross talk on the system performance must be considered. Sharing common parts of the CDR structure is an obvious step to reduce the system overhead. Relaxed frequency variations, run length and jitter specifications compared to long-haul systems allow for a simplification of the CDR topology, either by reduction of the loop filter size, or by implementation of alternative topologies, which would be unable to cope with SONET-type specifications. In order to compare their advantages and drawbacks, an overview of the topologies to be considered is presented. Over the last years, the growing number of electrical serial link applications has stimulated the research on alternatives to the conventional PLL/DLL topologies. As the terminology in this field is not totally harmonized, the following classification and especially the association of some published designs to given categories may be subject to discussion. In a general fashion, we will consider the following clock recovery categories, individually discussed below:

- PLL-based topologies
- Delay-locked loop (DLL)-based topologies
- Oversampling topologies
- Phase interpolating (PI) topologies
- Injection locking (IL) topologies
- Gated oscillator (GO) topologies

8.2.1 PLL-Based Topologies

PLL-based clock recovery systems, presented above, have several intrinsic advantages, like exact frequency acquisition, a loop filter memorizing the data frequency over long runs and last but not least, the exhaustive literature and design documentation available on this topology. On the other hand, each PLL is an independent system which cannot easily share components with others, limiting the amount of optimization in multichannel receivers. Although fully integrated DSP-based loop filter topologies have been presented at the transmit side [90], most loop filter topologies rely on passive components of relatively large values. Finally, the presence of multiple oscillators in a common substrate may lead to unwanted *injection locking*, which describes the situation where an oscillator locks to a parasitic signal superimposed to the supply, the substrate bias, a bias current or a reference voltage. This issue may however be alleviated by the usage of silicon-on-insulator technologies in the case of monolithic integration with the proposed photodetector.

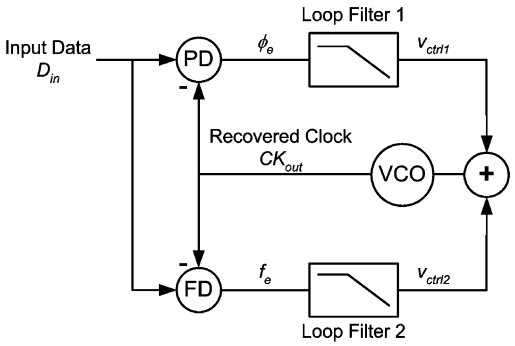

FIGURE 8.10. PLL-based CDR with phase and frequency acquisition loops

Another issue when using PLLs in serial link receivers is frequency acquisition. Due to the non-periodic input data, clock recovery circuits cannot use combined phase-frequency detectors and require an additional loop for frequency acquisition (Figure 8.10). Indeed, the phase detectors able to operate on NRZ data cannot provide frequency acquisition the way conventional phase-frequency detectors do.

The nonlinear operation applied to the input data for generation of spectral content at the data frequency also generates spectral power at different frequencies, depending on the received data pattern. A PLL without frequency acquisition may lock to some harmonic spectral content, leading to erroneous interpretation of the data stream called *harmonic lock* or *false lock* [91].

A frequency acquisition aid may also improve the acquisition speed when out-of-lock. Further more, due to the increasing manufacturing tolerances in modern processes, the free-running frequency of integrated VCOs exhibit considerable die-to-die variations. Again, a frequency acquisition path contributes to the solution of this problem. The design of this additional path however requires some care to guarantee the proper transition between frequency, acquisition and phase-acquisition regimes. Hard switching between the two paths, as suggested in some publications, is usually not recommended because the switching noise may suffice to bring the loop again back out of frequency lock. Frequency detectors with a dead zone are an elegant solution to this problem.

The reader should understand that frequency acquisition is an issue for all closed-loop clock recovery topologies discussed below. Stringent frequency stability requirements on the transmitter side and reduced data run length in 8b/10b encoded data streams however mitigate this issue and frequently allow for omission of this path, especially in receiver schemes based on a very precise reference clock (e.g. phase interpolation, injection locking).

8.2.2 DLL-Based Topologies

A way to circumvent the use of multiple oscillators on a single die is the centralized generation and distribution of a reference clock around the chip. In each receiver channel, this clock can be fed into a voltage/current controlled delay line (VCDL/CCDL) to be aligned with the received data stream (Figure 8.11). In most circuits, such a reference clock is readily available from the transmitter clock source, resulting in significant overhead reduction. It is interesting to graphically present the similarities of the ring oscillator-based PLL and the DLL clock recovery structures (Figure 8.12).

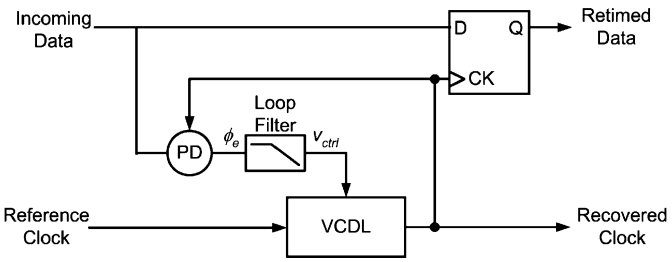

FIGURE 8.11. DLL-based clock recovery and data retiming circuit

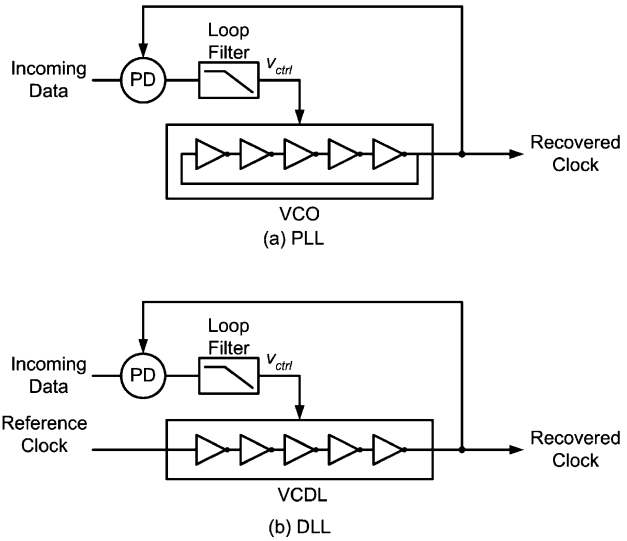

FIGURE 8.12. (a) PLL and (b) DLL clock recovery topologies. (After Yeung [92].)

The phase detector and the loop filter operate the same way than in a PLL structure. The control voltage delivered by the loop filter tunes the delay of the VCDL. While the VCO control voltage acts on the clock frequency, thus on the derivative of the phase, the VCDL control voltage directly acts on the clock phase. This means that the delay line does not introduce a pole in the loop transfer function as the VCO does. Delay-locked loops are thus of the same order than their loop filter, leading to more stable systems. Furthermore, they do not suffer of jitter accumulation due to the signal recirculation in the VCO. However, the delay line does not provide infinite delay tuning capability and for this reason cannot handle even small-frequency offsets between the receiver and the incoming data. Solutions to this problem have been presented. A control voltage correction scheme may compensate for the limited phase-capture range [93], while quadrature mixing is used in a topology somewhere between a DLL and a phase-interpolation scheme in [94]. Another alternative called “dual delay-locked loop” is in fact based on phase interpolation and will be discussed in the corresponding paragraph [95].

8.2.3 Oversampling Receivers

With the increasing speed performance of modern CMOS technologies, the use of oversampling strategies becomes appealing. As shown in Figure 8.13, a multiphase clock, generated either by a ring oscillator in a PLL or by a delay-locked loop, can be used to acquire several samples of the input data at

different instants within a clock cycle. The resulting data word, stored on a parallel array of retiming elements (i.e. flip-flops), is fed to a digital decision logic block, which selects the optimum sample out of the data word [96]. Due to the digital decision algorithm, this feedbackless topology contains a minimum amount of analog blocks. It is however very sensitive to mismatches in and between these blocks, like delay mismatch in the sampling clock paths. This sensitivity becomes critical when supplying multiphase clocks to several of these oversampling receivers using full-chip routing methods. Furthermore, the digital implementation of the so-called *phase picking* algorithm [97] occupies considerable chip area. It can however be expected to shrink conveniently with technology scaling and become appealing in terms of area and power consumption in very deep submicron processes.

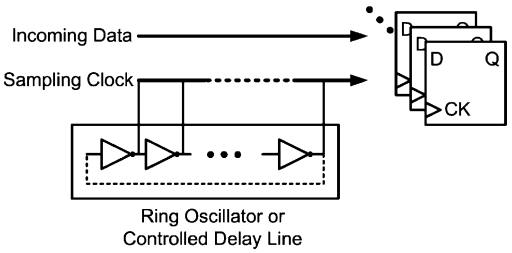

FIGURE 8.13. Multiphase clock generation and signal oversampling

8.2.4 Phase Interpolation

Another way of using a multiphase clock is presented in Figure 8.14, where the best-matching clock phase is selected to minimize the phase error. This *phase selection* strategy is quite close to the *phase alignment* presented in [98] and [99], where the data is sent through a delay line to obtain multiple data phases, followed by a selection multiplexer. As it is preferable to send a periodic signal through the delay line to avoid duty cycle distortion, the phase alignment topology shall be discarded to the benefit of phase selection.

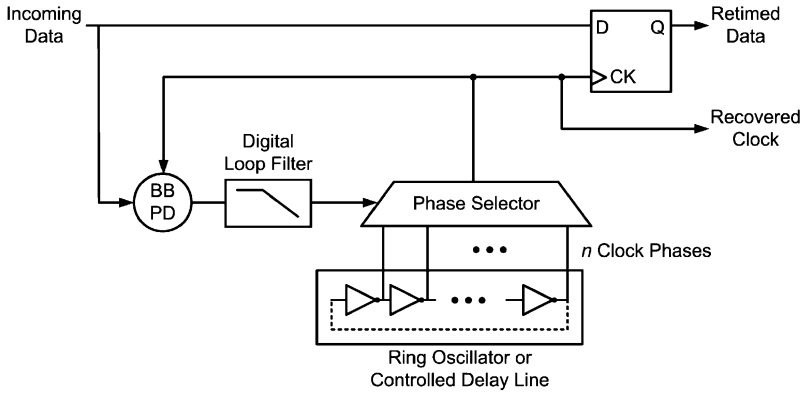

FIGURE 8.14. Phase selection

The granularity of the phase selection can be improved by weighted interpolation between two consecutive clock phases. As shown in Figure 8.15, the clock phase interval corresponding to the arriving data edges is selected first, based on the most significant bits (MSBs) of the loop filter. A finer tuning is achieved by mixing the bracketing phase edges using a weighting method based on the least significant bits (LSBs) of the loop filter [95, 100–102]. This topology can be implemented in a very compact fashion in modern processes using a bang–bang PD with a digital loop filter and a current-steering DAC [103].

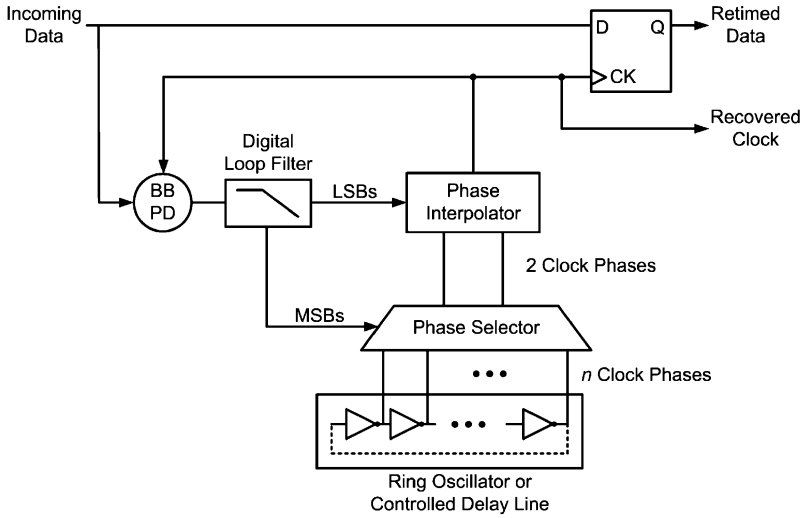

FIGURE 8.15. Phase interpolation structure

As the multiphase clock signal is delivered either by a ring oscillator or by a delay-locked loop and because of the similarities with phase alignment, phase interpolation designs frequently carry different names (e.g. “dual delay-locked loop”, “dual-loop PLL”, etc.). In some designs, quadrature clock signals are used as inputs to the phase interpolator, which in that case is called “phase rotator”. Quadrature phase interpolation requires only two differential clock signals to be routed, minimizing the overhead compared to oversampling techniques and nonetheless eliminating the need for one DLL per channel for phase generation.

8.2.5 Injection Locking

While injection locking has been mentioned earlier as a parasitic phenomenon which may harm the performance of an oscillator-based CDR, it can also be used as a means of synchronization of an oscillator with a stimulus [104–105]. Imagine that, in the phase interpolation structure presented above, we replace the phase interpolator with a ring oscillator (Figure 8.16). In this case, we can inject the weighted clock signals at the output of the phase selector into specific nodes of the ring oscillator. Obviously, the injection mechanism must be strong enough to override any parasitic injection from the supplies, the substrate or other potential noise sources.

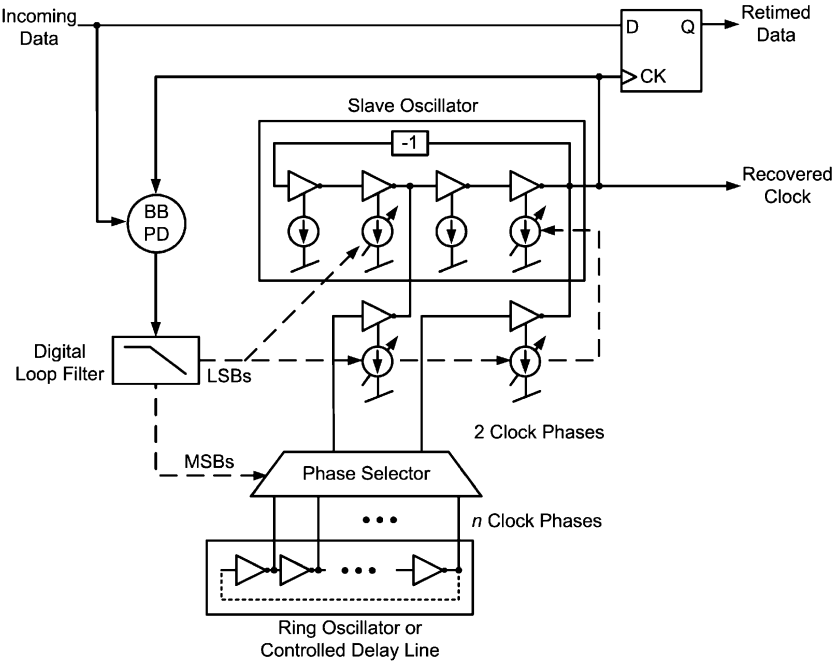

FIGURE 8.16. Injection-locking CDR topology

As the signal injected into the ring oscillator represents only a fraction of the circulating oscillation, the injection-locked oscillator acts as a low-pass filter and somewhat smooths duty-cycle distortion due to the injection mechanism [104]. The generated sampling clock phase thus exhibits much smoother variations compared to the steps of the PI clock. Again, this improvement in tracking jitter trades off with the slave oscillator lock range. Due to the additional ring oscillator, this topology may also result in a small area and power consumption overhead compared to the phase interpolation structure.

8.2.6 Gated Oscillator Topology

The gated oscillator (GO) topology [106] proposes a very different approach to the clock recovery problem. Its open-loop structure does not measure phase error between incoming data and sampling clock in any sense, but simply relies on the resynchronization of the sampling clock at each incoming data edge (Figure 8.17). This structure intrinsically offers a low device count, resulting in a very compact and low-power clock recovery solution. The free-running frequency of each oscillator is determined by the tuning signal delivered by a reference PLL. The possibility of sharing this reference PLL renders this topology even more appealing for multichannel communication links.

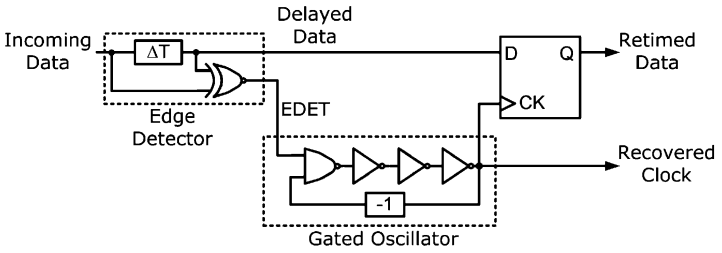

FIGURE 8.17. A single channel gated oscillator CDR

The gated oscillator receiver relies on the arrival time of only one transition, the immediately preceding one, for each bit to be synchronized. As such, it does not provide any jitter rejection mechanism, i.e. its jitter transfer characteristic is equal to one. Neither the immediately following data edge, nor the previous ones, are considered in the selection of the sampling instant. Beyond the oscillator phase information being reset by each arriving data transition, the oscillator runs at its free running frequency in absence of transitions (e.g. during data runs). In addition to its sensitivity to jitter, this structure suffers hence from increased timing errors in presence of frequency offset between the data source and the gated oscillator. However, in most transmission schemes, the incoming data frequency is only known

up to a given tolerance. Potential mismatch between the reference PLL and each local gated oscillator may cause additional frequency offsets to be taken into account in the circuit design.

8.3 Topology Discussion

Table 8.1 presents a summary of the advantages and drawbacks of the various topologies in the scope of clock recovery for short-distance multichannel data links.

TABLE 8.1. CDR Topology Comparison

Topology	Advantages	Drawbacks
PLL-based	Excellent jitter tolerance Tracks input frequency	One oscillator per channel (cross talk) Analog loop filter (area)
DLL-based	Reduced complexity Shared clock source	Complexity for plesiochronous operation Analog loop filter (area) Chip-level routing of clock signal
Oversampling	No feedback loops Little sensitivity to analog components	One DLL or PLL per channel or Chip-level routing of many clock signals Digital circuit complexity
Phase Interpolation (PI)	Good jitter tolerance Shared clock source Intrinsically digital loop filter	Chip-level routing of clock signal(s)
Injection Locking (IL)	Good jitter tolerance Shared clock source Smooth Phase Adjustment Intrinsically digital loop filter	One oscillator per channel (cross talk) Chip-level routing of clock signal(s)
Gated Oscillator (GO)	Low area Potentially low power No chip-level routing of clock signal	Lack of jitter tolerance One oscillator per channel (cross talk) No tracking of input frequency

Focusing on the drawbacks first and considering area, power consumption and straightforward place and route at the chip level as dominant constraints for multichannel receivers, the three latter topologies represent the most promising solutions to the problem. The absence of high-frequency signal routing promote the gated oscillator design to the first place of this selection, but the lack of jitter and frequency tolerance raise major questions regarding its performance. This topology will be used in the following to demonstrate how a systematic top-down design methodology can be applied to analyze the jitter performance of a given CDR topology, as well as guarantee that the final circuit design respects the initial high-level specifications.

8.4 Specifications

Right before going into the detailed analysis of the gated oscillator CDR topology, we will present typical CDR specifications as developed in Chapter 4 (Table 8.2). In oscillator-based multichannel receivers, the different channels may exhibit different free running frequencies due to device mismatch, temperature and supply voltage variations. For this reason, the specified span for tuning range presented below is considerably larger than its initial value determined at the system level. In very short distance processor-to-processor links, the jitter tolerance specifications can be relaxed compared to inter-server data links that are defined for distances up to 300 m.

TABLE 8.2. Clock recovery specifications

Parameter	Acronym	Minimum	Maximum	Unit
Input sensitivity	V_{minCDR}	350		mV_{PP}
Input capacitance	C_{inCDR}		50	fF
Frequency tuning Range	Δf_{TR}	–5%	5%	f_0
Jitter tolerance	$JTOL$	0.46		UI
Total jitter	TJ_{PP}	0.41		UI_{PP}
DJ contribution	DJ_{PP}	0.2		UI_{PP}

8.5 The Gated Oscillator Topology

In order to become a commercially viable alternative to conventional parallel I/O interfaces, multichannel serial receivers must achieve high data rates, low power consumption per channel and low silicon area. The gated oscillator topology is a good choice for such applications. It uses a shared PLL and one current-controlled oscillator (CCO) per channel (Figure 8.18). The advantages of this topology in multichannel applications have already been presented, but before addressing the low-level design, it must be shown that the achievable jitter tolerance and tolerable frequency offset meet the system specification. First, the circuit topology and operation will be described, then a detailed top-down timing analysis will be presented in Section 8.6.

8.5.1 The Multichannel Gated Oscillator CDR

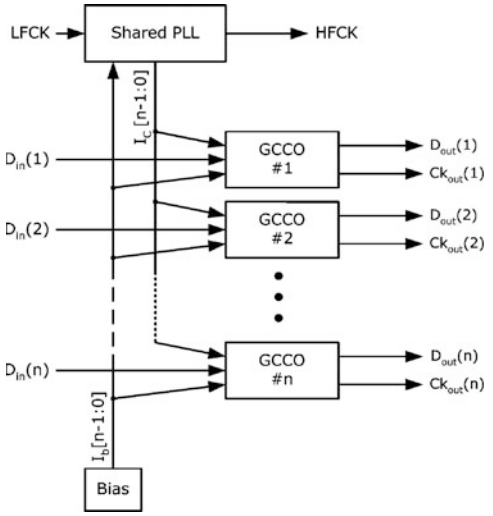

FIGURE 8.18. Multichannel GCCO CDR

As shown in Figure 8.18, the shared PLL generates a high-frequency clock *HFCK* at 2.5 GHz by frequency multiplication by 16 of a very precise low-frequency 156.25 MHz reference clock *LFCK*. It is based on a current-controlled oscillator and the control current of this oscillator is mirrored. One sample of the control current $I_c[n\text{-}1:0]$ is delivered to the CCO in each receiver channel, tuning them to a free-running frequency close to 2.5 GHz. The precision of the frequency tuning is only limited by the oscillator matching and the matching of the control currents. Current-controlled oscillators are preferred to their voltage-controlled counterparts due to better matching. Additional bias currents $I_B[n\text{-}1:0]$ are distributed by a central bias generator for proper biasing of logic gates and buffers.

8.5.2 The Clock Recovery Core

The clock recovery circuit for a single channel is composed of three majors parts (Figure 8.19): the edge detection, the gated oscillator core and a sampling element. For simplicity, the schematic is drawn in a single-ended fashion, whereas the hardware implementation uses fully differential current-mode

logic gates. The "-1" sign in the feedback loop represents the inversion operated on the fully differential signal through wire crossing.

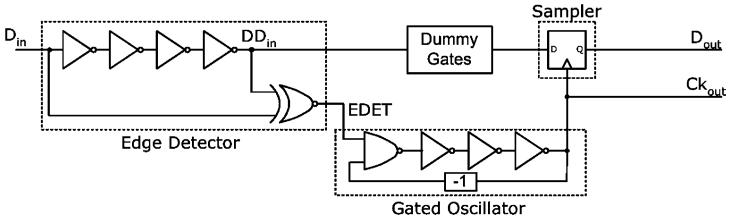

FIGURE 8.19. Gated oscillator schematic with edge detection

The edge detection is performed by an XOR gate comparing the incoming data D_{in} with a delayed copy of the same signal called DD_{in}. At each incoming data edge, a falling pulse is generated on the active low *EDET* signal. This pulse triggers the synchronization operation in the gated oscillator. The duration of the *EDET* pulse is determined by the delay line in the edge detector as well as potential delay mismatch between both inputs in the XOR gate. As we will see later, the *EDET* pulses shall not be shorter than half a clock period, in which case the XOR delay mismatch should be negligible. To allow for delay matching with the oscillator delay stages, the edge detector delay line is built using current-mode logic NAND gates.

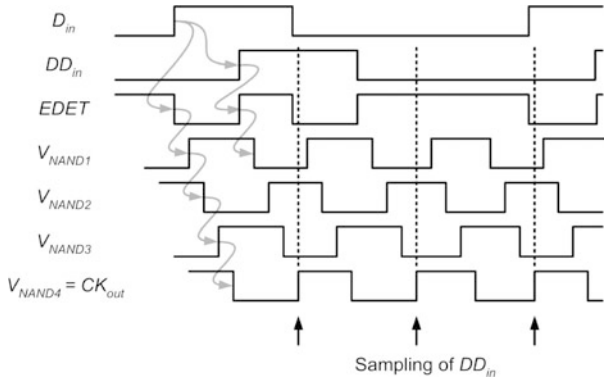

FIGURE 8.20. Gated oscillator timing diagram

The ring oscillator in the GO core is built using four identical NAND gates. The three latter stages have the unused input nodes connected to appropriate bias voltages. The use of identical gates results in better control of the delays compared to a mixed NAND-inverter combination, as well as good matching with the NAND gates used in the edge detector delay line. When the gated oscillator receives the

falling *EDET* signal, the output of the first NAND gate goes high (Figure 8.20). As a property of the ring oscillator, the propagation delay of each gate is equal to $T_{CK}/2n$, where n is the number of stages in the ring oscillator. As long as EDET stays low, this NAND gate inhibits propagation of any data transitions received by the feedback loop from the oscillator output. Once D_{in} propagated through the edge detector delay line to DD_{in}, the rising edge of *EDET* releases the first oscillator NAND gate, which goes back to its inverting operation with respect to the feedback signal.

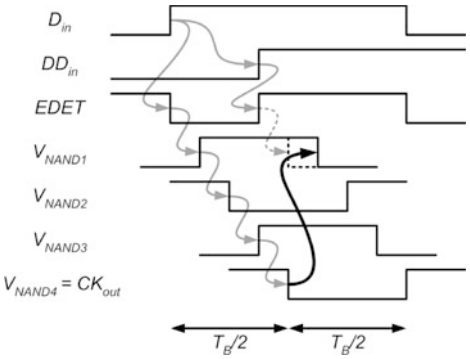

FIGURE 8.21. Race condition due to short *EDET* signal

If the *EDET* signal rises before the feedback signal has risen to "1" under the influence of the same *EDET* edge, an edge will not be generated at the output of the first NAND gate in all situations. If such an edge is not generated, an output clock edge will not occur half a clock cycle later and the data will not be sampled correctly. As this situation can be considered to correspond to a late arrival of the feedback signal, we say the circuit is subject to a *race condition* (Figure 8.21).As the propagation delay from the *EDET* input to the clock output and thus to the feedback input is half a clock cycle, the *EDET* pulse duration must in any case exceed this value. The reader should however also be aware that longer pulse durations result in degraded jitter tolerance. Indeed, in case a second data edge arrives early, a very short EDET rising pulse may be generated (Figure 8.22). The resulting clock pulse may be that

small that it vanishes in the oscillator or results in hold-time violations in the sampling flip-flop. Either case would result in a missing sampling instant, i.e. a dropped bit.

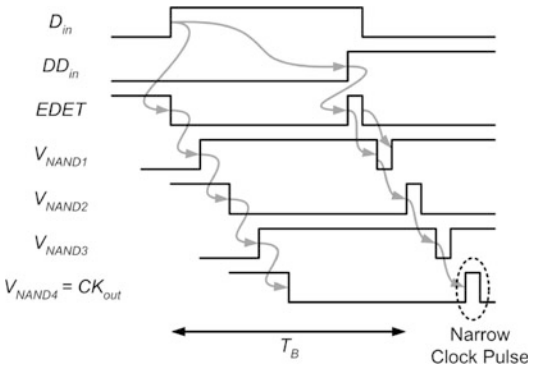

FIGURE 8.22. Race condition due to short *EDET* signal

When released, the oscillator resumes operation at its free-running frequency set by the control current delivered by the shared PLL. As the rising *EDET* edge takes half a clock cycle to propagate to the oscillator output, the delayed data is sampled in the sampling flip-flop (FF) with this same delay with respect to the data edge. Parasitic delays in the *EDET* path are compensated for using dummy gates in the data path. It is important to understand that, in presence of jitter, sampling half a clock period after the data edge does not always correspond to sampling in the middle of the eye, as performed by most conventional CDR topologies (Figure 8.23).

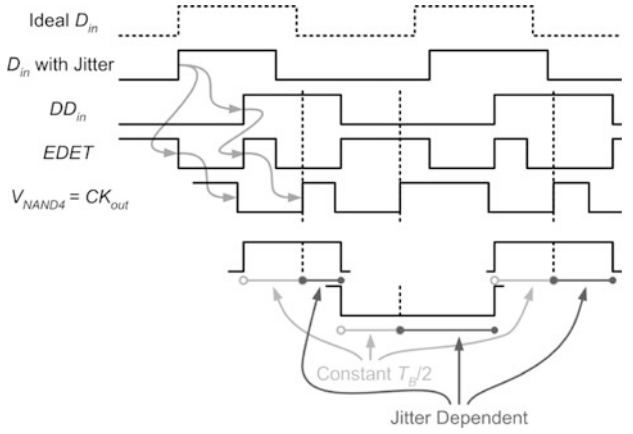

FIGURE 8.23. Sampling after $T_{CK}/2$

The alignment of the sampling instant on each data edge results in unity jitter transfer, which does not only affect the periodicity of the recovered clock signal, but also enhances the jitter seen by the sampling FF on the following data edge. Imagine an incoming edge being late due to jitter, shifting the generated clock edge by the same amount (bottom part of Figure 8.23). If the next data edge is early, it will not only be shifted by its own jitter to the center of the eye (defined by the clock edge), but by the sum of both jitter contributions. In consequence, all jitter contributions on both data edges are appearing in the eye diagram as being applied to the second data edge.

Finally, the sampled data D_{out} and the recovered clock CK_{out} are delivered to the output. As this topology does not have any long-term memory with respect to data rate and low-frequency jitter, it is fairly intuitive that in presence of frequency offsets, frequent data edges must be received to avoid excessive drift of the sampling instant from the desired location (Figure 8.24). This issue is accentuated by the fact that device mismatch and supply voltage variations between the CDR oscillator and the shared PLL oscillator are enhanced by physical distance on the silicon die.

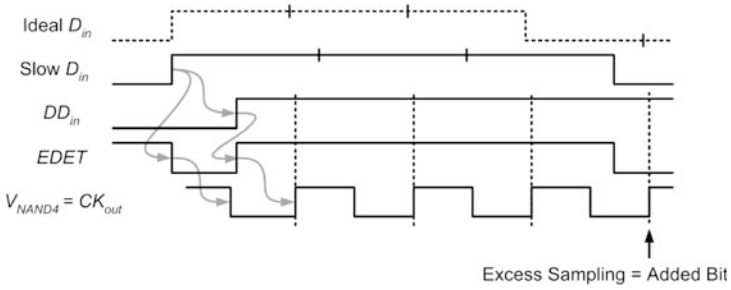

FIGURE 8.24. Drift of sampling point in presence of frequency offsets (exaggerated)

8.5.3 CDR Top-Down Design Methodology

While gated oscillator structures have been used in the past in burst-mode communications, jitter and frequency tolerance requirements in the present application are more stringent. For the above-mentioned reasons, a thorough analysis of jitter tolerance (JTOL) and frequency tolerance (FTOL) performance of the gated oscillator topology is mandatory before addressing the transistor-level design. In order to propagate and refine the bit error ratio estimations at the different abstraction levels, from the system level down to the device level, we introduce a specification driven top-down design methodology for such gated oscillator clock recovery circuits (Figure 8.25). After the selection of the clock recovery topology, statistical Matlab-based simulation is used to get an estimate of jitter and frequency tolerance. A time-domain topology-based behavioral model is developed using a hardware description language (HDL) that allows in-depth sensitivity analysis of various topology parameters, like gate

delays, thermal noise and voltage levels. If the simulation results of the time-domain analysis are in agreement with the statistical results, we can step over to the transistor-level design. The HDL model includes most block-level parameters of the device-level design. Particularly the acceptable oscillator jitter is first determined at the statistical level, then verified at the behavioral level. The bias conditions of the ring oscillator are obtained from the obtained value using Hajimiri's theory [108], as discussed in Section 8.8. Finally, the classical analog flow through layout, parasitic extraction and post-layout simulation phases, guarantees the verification of the design before submission for manufacturing. The presented top-down design methodology may of course be adapted to other CDR structures as well.

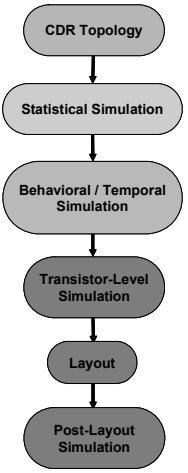

FIGURE 8.25. Specification driven top-down design methodology

8.6 Statistical Modeling of the Gated Oscillator

The highest level of abstraction in this flow models jitter components based on their statistical distributions. Jitter tolerance and tolerance of frequency offsets are estimated based on the resulting bit error ratio through purely statistical analysis, using Matlab. This model does not take into account implementation details, but delivers good insight on the jitter and frequency tolerances in limited simulation time.

8.6.1 Jitter Statistics

Before discussion of the statistical model, we describe the jitter statistics of the different jitter components applied to the receiver (Table 8.3).

TABLE 8.3. Jitter specifications for statistical modeling

Jitter Type	Acronym	Units	Value	PDF
Random jitter	*RJ*	UI_{RMS}	0.015(1)	Gaussian
Deterministic jitter	*DJ*	UI_{PP}	0.2	Dual-Dirac
Sinusoidal jitter	*SJ*	UI_{PP}	swept(2)	Sinusoidal
GCCO Clock jitter	*CKJ*	UI_{RMS}	TBD(3)	Gaussian

(1) The RMS value is obtained from a total jitter (TJ) specification of 0.41 UI_{RMS}, resulting in $RJ_{PP} = TJ_{PP} - DJ_{PP} = 0.21$ UI_{PP}, then divided by the PP to RMS conversion factor for a BER of 10^{-12}.

(2) Sinusoidal jitter amplitude and frequencies are swept to obtain the jitter tolerance graph to be compared with Figure 4b.

(3) To be defined. An acceptable upper bound for the clock jitter is to be obtained from this analysis.

The basic concepts of statistical jitter analysis, as discussed in [107], will be briefly introduced. In this reference, the bit error ratio is defined by the integral of the error probability over the range of possible sampling instants t_s.

$$BER = \int_{-\infty}^{\infty} P[\varepsilon, t_s] \qquad \text{(EQ 8.12)}$$

The bit error probability over the full range of t_s itself depends on the probability of error at each possible sampling instant $P[\varepsilon|t_s]$ and the probability of sampling at this instant $P[t_s]$.

$$P[\varepsilon, t_s] = P[\varepsilon|t_s] \cdot P[t_s] \qquad \text{(EQ 8.13)}$$

In this approach, channel jitter contributions are assigned to the data edges, while jitter sources in the CDR oscillator are associated with the sampling instant. This calculation is suitable for conventional clock recovery circuits, which use the loop filter memory to average out jitter and track the average data edge location in time. It is not suitable, however, for the analysis of the gated oscillator CDR, where the oscillator phase is realigned at each incoming data edge. As a result of this retiming strategy, channel and receiver jitter, as well as frequency offsets, accumulate differently on both edges of the eye diagram. In the following, we will propose an alternative model, which considering an ideal sampling clock and assigns all jitter contributions to the data edges. This approach results in more accurate modeling of the gated oscillator CDR performance.

First, the jitter sources and offsets corresponding to both data edges in the eye diagram will be calculated. The total jitter probability density function, calculated separately for the first (i.e. left) and second (i.e. right) data edge, is obtained by convolution of the different jitter distributions. The reader should understand that in this development, the term *jitter* is used in its largest sense, i.e. it also includes gate delays and frequency offsets. The sources of jitter can be easily found by analyzing the data and clock paths in the schematic, starting from the *EDET* generation at the output of the edge detector and closing the loop at the sampler.

Possible static delays between data and clock can be due to mismatch between active gates and dummy gates (e.g. data-clock delay `d2ck_del`), as well as intentionally introduced delays obtained by tapping a different output of the ring oscillator (`delta_samp_point`). Frequency offset of the oscillator is accumulated as long as no new data edge arrives. This duration is determined by the number of consecutive identical bits `CID` (Figure 8.26).

```
edge1_off = delta_samp_point + d2ck_del + delta_f_rel*0.5;
edge2_off = delta_samp_point + d2ck_del + delta_f_rel*(CID-0.5);
```

FIGURE 8.26. Modeling of timing offset due to static delays and frequency offset

8.6.2 Random Jitter Components

Oscillator jitter accumulates over a time period ΔT according to Equation 8.14. Unlike the free-running oscillator modeled in [108], the gated oscillator is regularly reset. We can consider the case of short-term accumulation of jitter only and neglect correlated noise sources like up-converted 1/f noise.

$$\sigma_{CKJ} = \kappa\sqrt{\Delta T} \qquad \text{(EQ 8.14)}$$

In this relationship, κ is a technology and design dependent proportionality constant used in the calculation of the oscillator bias currents. Knowing the maximum number of CID defining ΔT, we can calculate κ to achieve a given phase noise, respectively a given oscillator jitter performance. The present bit error ratio estimation determines the maximum tolerable oscillator jitter to meet the system specifications. A shown in Section 8.8, this approach allows us to obtain design parameter values for minimum system power, which is one of the main constraints of the application. Random data jitter, due to amplitude noise to jitter conversion in the receiver's amplification stages, is not subject to any recirculating mechanism the way oscillator is and does thus not exhibit comparable accumulation properties. Random jitter sources in the edge detection delay line of the CDR are supposed negligible.

```
edge1_sigma = gvco_k * sqrt(0.5);
edge2_sigma = gvco_k * sqrt(CID-0.5) + sqrt(2) * RJ;
```

FIGURE 8.27. Oscillator jitter accumulation model

As the jitter affecting the first data edge is transferred to the clock edge, the effective random jitter of the second data edge with respect to the clock edge is weighted by a factor √2 (Figure 8.27). This model does not take into account the fact that random jitter on either data edge is correlated, as shown by the nonzero correlation coefficient of Gaussian white noise of limited bandwidth BW (Equation 8.15).

$$\rho_x(\Delta t) = \mathrm{sinc}(2BW\Delta t) \quad \textbf{(EQ 8.15)}$$

Indeed, for a single-period time interval between two data edges and a bandwidth of $0.75 f_B$, the correlation is 66.5%. Inclusion of this reduction in effective random jitter seen by the CDR in the model is left to the interested reader.

8.6.3 Deterministic Jitter Components

As for random jitter, the deterministic jitter components on an incoming data edge are transferred to the clock. The left data edge is thus exempt of deterministic jitter when being sampled. However, deterministic jitter on the second data edge is doubled, because DJ components on consecutive data edges are considered uncorrelated. The deterministic jitter probability density function (PDF) is modeled with a so-called dual-Dirac distribution, as suggested in [21]. A detailed discussion of the dual-Dirac model and how it is used to estimate the BER from statistical measurement results is presented in [109].

The convolution of the dual-Dirac distribution with a conventional random jitter PDF leads to the results Figure 8.28. The reader can easily understand the split of the RJ Gaussian PDF at each data edge into a sum of two Gaussian PDFs offset by the DJ Dirac functions (Equation 8.16).

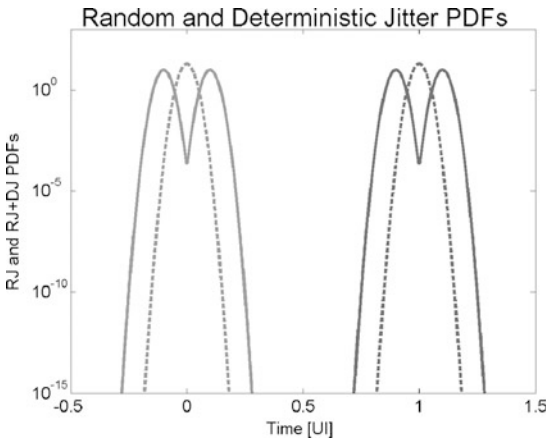

FIGURE 8.28. RJ and combined RJ–DJ probability density functions

$$p_{RJDJ}(t) = 0.5 \cdot \frac{1}{RJ_{RMS}\sqrt{2\pi}} \cdot e^{-\frac{\left(t-\frac{DJ}{2}\right)^2}{2RJ_{RMS}^2}} + 0.5 \cdot \frac{1}{RJ_{RMS}\sqrt{2\pi}} \cdot e^{-\frac{\left(t+\frac{DJ}{2}\right)^2}{2RJ_{RMS}^2}} \quad \text{(EQ 8.16)}$$

This equation, written for the PDF of a single data edge, also holds for the particular case of the gated oscillator, where different RJ and DJ values are applied. The implementation code is shown in Figure 8.29. As for random jitter, the second edge jitter is affected by deterministic jitter doubling due to the jitter transfer of the first edge jitter to the clock edge.

```
pdf_rjdj_edge1=normpdf(x, edge1_off, edge1_sigma);
pdf_rjdj_edge2=0.5*normpdf(x, 1.0-2*DJ/2+edge2_off, edge2_sigma)
              +0.5*normpdf(x, 1.0+2*DJ/2+edge2_off, edge2_sigma);
```

FIGURE 8.29. Combined RJ–DJ PDF Model

The resulting RJ–DJ PDF affecting the second data edge in the case of a gated oscillator CDR is illustrated in Figure 8.30. One can observe the broadening of the probability density function compared to the Figure 8.28.

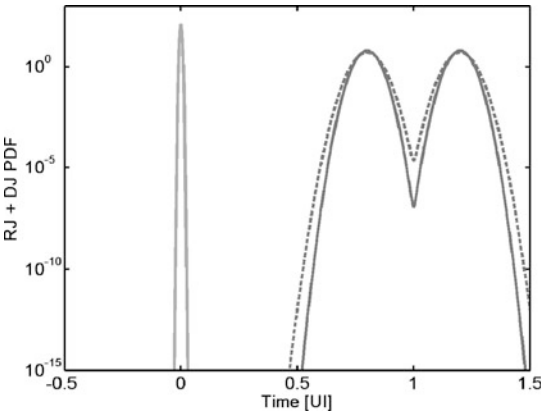

FIGURE 8.30. Combined RJ–DJ PDF in a GO CDR for CID = 1 (solid) and CID = 5 (dashed)

8.6.4 Sinusoidal Jitter Model

For sinusoidal jitter, the fundamental goal is the consideration of the variation of the sinusoidal modulation between two data edges, which will be called *differential sinusoidal jitter* (*dSJ*).

Let us consider an input signal *ck(t)* subject to the sinusoidal phase modulation $\varphi_{SJ}(t)$. For simplicity of the developments, we consider in the following a modulated periodic (sinusoidal) signal instead

of a random data signal. The results in terms of SJ contributions however remain valid for any data sequence with known maximum run length. Equation 8.17 shows the time-domain description of a sinusoidal clock signal of peak amplitude A_{ck} and frequency f_{ck} modulated by a sinusoidal jitter component $\varphi_{SJ}(t)$ of peak amplitude A_{SJ} and frequency f_{SJ}. As the clock recovery circuit operates on the data transitions only, the assumption of the modulated signal being sinusoidal does not limit the generality of the development.

$$\begin{aligned} ck(t) &= A_{CK} \cdot \sin[2\pi \cdot f_{ck} \cdot t + \varphi_{SJ}(t)] \\ \varphi_{SJ}(t) &= A_{SJ} \sin(2\pi \cdot f_{SJ} \cdot t) \end{aligned} \qquad \text{(EQ 8.17)}$$

The probability density function of a sinusoidal waveform of peak amplitude A_{SJ} is given by:

$$p_{SJ}(x) = \begin{cases} \dfrac{1}{\pi \cdot \sqrt{A_{SJ}^2 - x^2}} & \text{if} \quad (|x| < A_{SJ}) \\ 0 & \text{if} \quad (|x| > A_{SJ}) \end{cases} \qquad \text{(EQ 8.18)}$$

Based on the fact that the relationship between the sampling instant and the jitter waveform affecting consecutive samples is known, sinusoidal jitter cannot be considered to be a stochastic process. In this paragraph, the probability density function of the differential sinusoidal jitter from a time instant t_1 to a time instant $t_2 = t_1 + \Delta t$ will be developed. Figure 8.31 shows how the amount of differential jitter depends not only on the time interval Δt, but also on the initial time instant t_1.

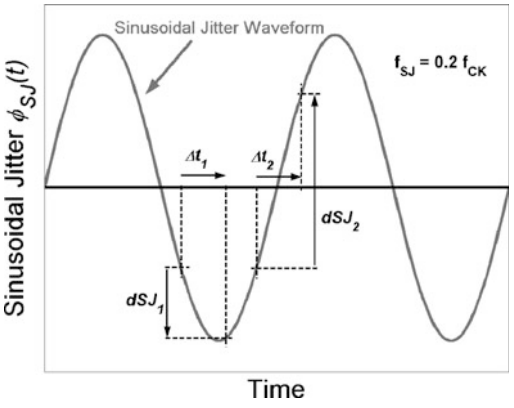

FIGURE 8.31. Influence of initial time and time interval on the differential sinusoidal jitter

The general approach to the problem will first develop the independent sinusoidal jitter PDF at a randomly chosen time t_1, then the conditional probability of the differential sinusoidal jitter, which depends on the initial position on the jitter sine wave at instant t_1. Finally, the time-independent differential SJ PDF will be obtained from these two distributions.

Let us first consider an arbitrarily chosen rising clock edge at time t_1, the PDF of the phase modulation due to sinusoidal jitter being given by Equation 8.18. The difference in SJ phase modulation between instant t_1 and arbitrarily chosen time instant $t_2 = t_1 + \Delta t$ is defined by:

$$\Delta\varphi_{SJ} = \varphi_{SJ}(t_2) - \varphi_{SJ}(t_1) = A_{SJ} \cdot [\cos(2\pi \cdot f_{ck} \cdot t_1) \cdot \sin(2\pi \cdot f_{SJ} \cdot \Delta t) - \sin(2\pi \cdot f_{ck} \cdot t_1) \cdot (1 - \cos(2\pi \cdot f_{SJ} \cdot \Delta t))] \quad \text{(EQ 8.19)}$$

This phase difference $\Delta\varphi$ represents the difference in phase modulation of the clock signal. If we take t_2 corresponding to another rising clock edge, Δt can be written as a function of the clock period T_{ck} and the number of clock periods n_{ck} between the two edges (Equation 8.20),

$$\Delta t = n_{ck} \cdot T_{ck} - (\varphi_{jit}(t_2) - \varphi_{jit}(t_1)) \quad \text{(EQ 8.20)}$$

where φ_{jit} represents the phase modulation due to all jitter sources. In order to solve this recursive relationship, combined with the dependency on other a priori unknown jitter contributions, we only consider the average value of Δt given by Equation 8.21.

$$\Delta t \approx CID \cdot T_{ck} = \frac{CID}{f_{ck}} \quad \text{(EQ 8.21)}$$

Knowing the PDF of the SJ phase at instant t_1 (Equation 8.18), as well as the probability $P[\Delta\varphi_{SJ}|\varphi_{SJ}(t_1)]$ of an SJ phase shift of $\Delta\varphi_{SJ}$ is known for each $\varphi_{SJ}(t)$, we can calculate the PDF associated with *dSJ*. The corresponding Matlab code, which calculates the probability density function of the differential sinusoidal jitter as a function of the SJ amplitude, normalized frequency and the number of

```
if abs(sin(TWOPI_fmod_t(i))) < 1
  prob_phi_t1_ft(i) = 0.5 /
    (pi*sqrt((sj_amp_pp/2)^2-(sj_amp_pp/2*sin(TWOPI_fmod_t(i)))^2));
  % 0.5 because the total probability is related to two points on the
  % single-period sine wave
else if abs(sin(TWOPI_fmod_t(i))) == 1
  % provides correct calculation of the boundary probabilities
  prob_phi_t1_ft(i) = 1 / (pi*sqrt((sj_amp_pp/2)^2
    -(sj_amp_pp/2*sin(TWOPI_fmod_t(i)-2*pi*fmod_t_step))^2));;
  else
    prob_phi_t1_ft(i) = 0;
  end
end
```

FIGURE 8.32. Calculation of the basic sinusoidal jitter PDF

CID, is shown below and explained step-by-step (Figures 8.32–8.34). The index `i` is the vector index used in the calculations to define the number of bins (or samples) used to represent the probability density functions.

According to Equation 8.18, the first part calculates the sinusoidal jitter PDF of $P[\varphi_{SJ}(t_1)]$ (represented by `prob_phi_t1_ft`) as a function of the instantaneous SJ phase (represented by `TWOPI_fmod_t`, corresponding to $2\pi f_{mod}\, t$), thus as a function of time. In this calculation, the boundary conditions require some particular attention and a factor 0.5 is introduced to take into account that each probability level in Equation 8.18 corresponds to two points on the sine wave in the time domain, which is considered here.

The next part calculates the phase $\varphi(t)$ for each instant t, as well as the associated-phase increment $\Delta\varphi$ (corresponding to `phi_t2_phi_t1`) as a function of the data rate `Tck`, the number of CID and the instantaneous SJ phase. This value is further scaled and rounded (`phi_t2_phi_t1_roundscale`), because it is later on used as a vector index of the final probability density function.

```
phi_t1(i) = sj_amp_pp/2 * sin(TWOPI_fmod_t(i));
phi_t2_phi_t1(i) = delta_phi_offs +
   sj_amp_pp/2 * (cos(TWOPI_fmod_t(i))*sin(TWOPI_fmod_Tck_CID)
   - sin(TWOPI_fmod_t(i))*(1-cos(TWOPI_fmod_Tck_CID)));
phi_t2_phi_t1_scale(i) = phi_t2_phi_t1(i) / prob_phi_t2_step;
phi_t2_phi_t1_roundscale(i) = round(phi_t2_phi_t1_scale(i));
```

FIGURE 8.33. Calculation of the initial SJ phase and the SJ phase increment

Up to this point, the probability of obtaining $\Delta\varphi_{SJ}$ is represented as a function of time. In order to calculate the conditional probability $P[\Delta\varphi_{SJ}|\varphi_{SJ}(t_1)]$, we need the probability to be expressed as a function of the phase. For this purpose, we use the scaled and rounded value of the differential phase `phi_t2_phi_t1_roundscale` as an index for the probability of the differential phase, called `prob_delta_phi_temp`. The '`temp`' extension is due to the fact that the result will be subject to some additional conditioning before revealing the true differential sinusoidal jitter PDF. Incremental addition is needed as several initial phase values $\varphi_{SJ}(t)$ combined with their associated $\Delta\varphi_{SJ}$ may be mapped to a single bin of the differential phase histogram.

```
x_delta_phi_temp(phi_t2_phi_t1_roundscale(i)) =
   phi_t2_phi_t1(i) - delta_phi_offs;
prob_delta_phi_temp(phi_t2_phi_t1_roundscale(i)) =
prob_delta_phi_temp(phi_t2_phi_t1_roundscale(i)) + prob_phi_t1_ft(i);
```

FIGURE 8.34. Calculation of the differential SJ PDF

Zero values in some histogram bins are due to the limited number of samples used for the calculations. In order to obtain a smooth plot, these bins are removed, which does not alter the integral of the histogram. Then, the histogram is smoothed by averaging with the two neighboring bins. These operations are not required from a mathematical point of view, but results in a more explicit probability density plot.

Figure 8.35 shows the resulting probability density functions for a given set of jitter amplitudes and frequencies. While jitter amplitude SJ_{PP} determines the maximum possible extent of the differential sinusoidal jitter distribution, the differential SJ amplitude and the location of its peaks depend on the normalized jitter frequency and the number of consecutive identical bits.

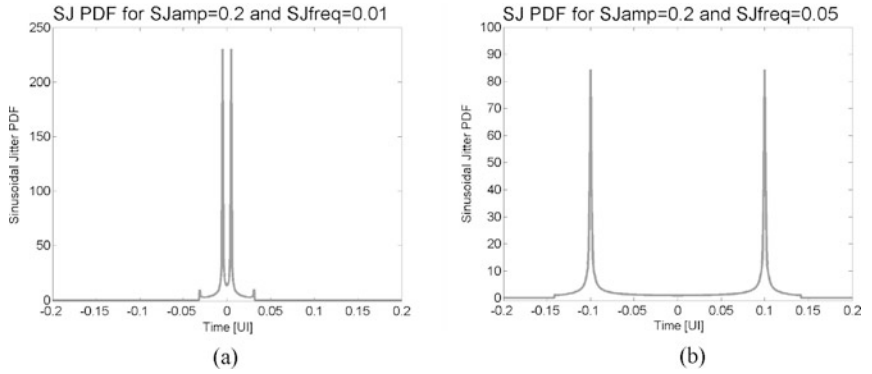

FIGURE 8.35. SJ probability density functions for CID = 5 and (a) SJ_{amp} = 0.2 UI_{PP} and SJ_{freq} = 0.01 f_B, (b) SJ_{amp} = 0.2 UI_{PP} and SJ_{freq} = 0.05 f_B

8.6.5 Bathtub Curves and BER Estimation

Based on the probability density functions for the different jitter contributions, we can calculate the the total jitter PDF for both data edges by convolution of the RJ–DJ and the SJ PDFs. The cumulative distribution functions (CDF) for both data edges, resulting from integration of the corresponding PDFs (Equation 8.23), are shown in Figure 8.36. In the case of gated oscillators, the first data edge

exclusively suffers from oscillator jitter, while the second jitter distribution is obtained by convolution of the RJ–DJ PDF with the sinusoidal jitter previously described.

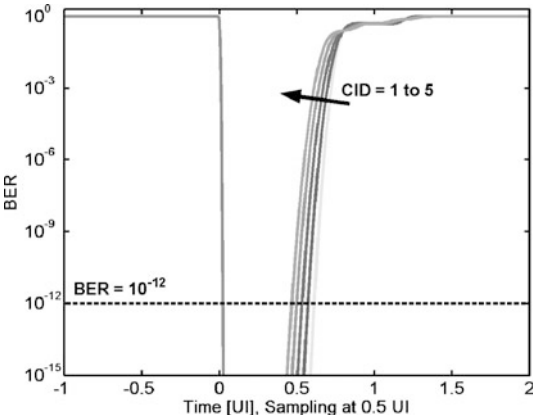

FIGURE 8.36. Bathtub curves for SJ_{amp} = 0.2 UI_{PP}, SJ_{freq} = 0.05 f_B, RJ = 0.015 UI_{RMS}, DJ = 0.2 UI_{PP}, $FTOL$ = 0 and CID = 1...5 (not weighted)

The amount of random jitter to be applied is derived from the total peak–peak jitter specification, then converted to an RMS value according to Equation 8.22.

$$RJ_{RMS} = \frac{TJ_{PP} - DJ_{PP}}{k_\sigma} \quad \text{(EQ 8.22)}$$

$$CDF_{edge1} = \int_t^\infty PDF_{edge1}\,dt$$
$$CDF_{edge2} = \int_{-\infty}^t PDF_{edge2}\,dt \quad \text{(EQ 8.23)}$$

The bit error ratio can be calculated (Equation 8.24) from the sum of both CDFs at the sampling point, multiplied by the transition probability (generally considered to be 0.5). Summation of the CDFs is equivalent to integration of the area under the PDF curves. In order to consider data edges beyond the previous or next sampling instants (–0.5 UI and 1.5 UI) as full errors, i.e. without weighting with the transition density, the bit error ratio is augmented by the sum of integrals of these intervals.

$$BER = p_{trans} \cdot (CDF_{edge1} + CDF_{edge2}) \quad \text{(EQ 8.24)}$$

Estimating the bit error ratio for the maximum number of CID overestimates the number of errors, as it does not take into consideration the fact that only few data transitions in an encoded data pattern

are subject to these conditions. In the presented model, the total BER is calculated as a weighted sum of the bit error ratios corresponding to n CID, where n is swept from 1 to CID_{max} = 5. The weights are determined by the probability of occurrence of n CID in an encoded random data stream. Considering a transition probability of 0.5, the probability of n CID is given by $p(\text{CID} = n) = 2^{-n}$. As 8 b/10 b encoding does not allow for n exceeding 5, each probability is the divided by $\sum_{n=1}^{5} p(\text{CID} = n)$ to guarantee that the sum of all probabilities equals 1. The calculated bit error ratio for given sinusoidal jitter amplitude-frequency pairs is shown in Figure 8.37. According to the jitter tolerance mask plotted at SJ_{PP} = 0.1 UI, the specified JTOL can tightly be achieved at a BER=10^{-12}, keeping in mind the relaxation of the deterministic jitter specifications with respect to the InfiniBand standard initially considered.

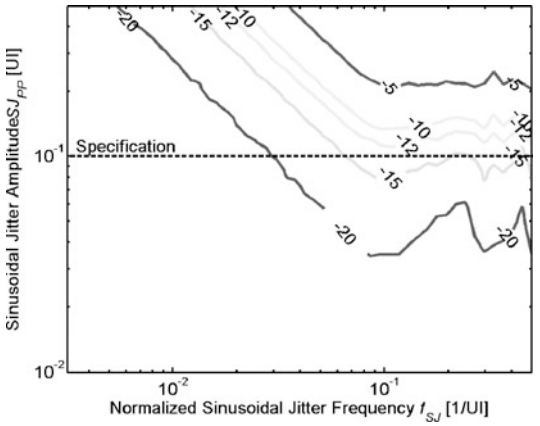

FIGURE 8.37. BER estimation for *RJ* = 0.015 UI_{RMS}, *DJ* = 0.2 UI_{PP}, FTOL = 0

Figure 8.38 shows the BER estimation in presence of a 1% frequency offset. The result highlights the sensitivity to frequency offsets between the transmitter and the receiver, as the specified bit error ratio cannot be met within all cases of the jitter tolerance mask. The bumps in the BER curves are due

to the dependency of differential sinusoidal jitter on the normalized jitter frequency (i.e. the ratio between the SJ frequency and the data rate), as shown in the SJ analysis.

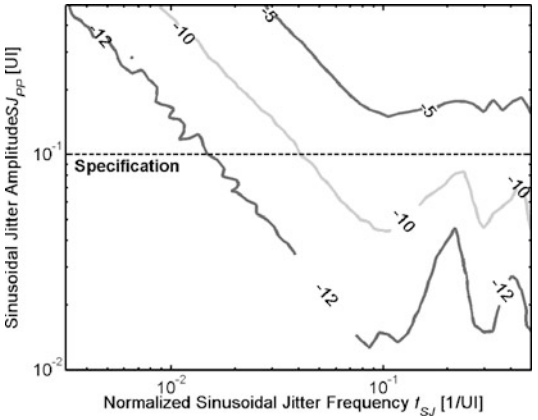

FIGURE 8.38. BER Estimation for *RJ* = 0.015 UI_{RMS}, *DJ* = 0.2 UI_{PP}, *FTOL* = 1%

In conclusion, the statistical modeling reveals that the simplicity and compactness of the gated oscillator topology result in limited tolerance to frequency mismatch and jitter. This topology cannot achieve sufficient jitter tolerance to apply for long-haul clock recovery or inter-server interconnects of 10–300 m length. However, in the box, where jitter is limited due to the reduced communications distances, this clock recovery topology can fully exploit its benefits: small silicon, low power and optimal scalability due to the use of digital building blocks only. Based on the presented analysis, the acceptable RMS clock jitter *CKJ* is specified as $0.01UI_{RMS}$ at very low-frequency offsets, allowing for a good power consumption/bit rate efficiency.

8.7 Time-Domain Modeling

The pattern generator code presented in this section has been previously released in [110].

8.7.1 CDR Model

The statistical model previously discussed gives a good estimate of the achievable performance, but it cannot verify the time-domain behavior of the selected topology. This verification is however necessary to analyze the influence of gate delays, various jitter sources, mismatch effects and also amplitude restoring mechanisms on jitter and frequency tolerance. Behavioral modeling using a hardware description language (HDL) is a convenient approach to this problem.

As the clock recovery circuit has an event-driven behavior and operates on two-level (binary) signals, the CDR time-domain model is based on a pure VHDL description. VHDL-AMS components are used in the test bench to generate the input stimuli and allow for proper representation of the output waveforms in the form of eye diagrams. The behavioral model of the clock recovery circuit is very close to the actual implementation shown in Figure 8.19 and contains all important gate delays and jitter contributions to be considered.

```
entity cdr_gcco is
  generic (
  cdr_gcco_k: real;-- CCO gain [Hz/A]
  cdr_gcco_fc: real;-- Free-running frequency [Hz]
  cdr_gcco_cc0: voltage;-- Control current mid-point [C]
  cdr_gcco_jit_sigma: real); -- defined as a ratio, e.g 1%->0.01
  port (
    […]);
end entity cdr_gcco;

architecture bhv of cdr_gcco is
  […]
begin
  calc_delay0: process
  begin
    awgn(seed1, seed2, mean, sigma, jitter);-- gaussian random gen.
    delay0 <= 1 ps * 1.0e12/ (8.0*(cdr_gcco_fc+cdr_gcco_k*(cctrl-
cdr_gcco_cc0))) * (1.0+jitter);
    wait for delay0;
  end process calc_delay0;

  […]    -- calculation of the three remaining delays

  (0) <= transport (vinv4(0) and cdr_gcco_trig(0)) and (cdr_gvco_enable
and cdr_gcco_nreset) after delay0;
  vinv1(1) <= transport (vinv4(0) nand cdr_gcco_trig(0)) and
(cdr_gvco_enable and cdr_gcco_nreset) after delay0;
  vinv2 <= transport not(vinv1) after delay1;
  vinv3 <= transport not(vinv2) after delay2;
  vinv4 <= transport not(vinv3) after delay3;
  cdr_gcco_ckout <= not(vinv4);
end bhv;
```

FIGURE 8.39. VHDL code of gated CCO

As shown in Figure 8.39, the delay of each gate in the ring oscillator is adjusted with respect to the control current `cctrl` coming from the shared PLL (the nominal value being `cdr_gcco_cc0`) and a jitter amplitude which is computed as a random white noise source following a Gaussian distribution [111, 112]. The output of each stage (`vinv1-vinv4`) is affected with the output of the preceding stage after the delay calculated above. In order to avoid suppression of events in the event stack under variable delay conditions, **`transport`** statements are required. In the current implementation, the

pseudo-random number generators used by the different delay cells use different seed values, but their sufficiency to achieve complete statistical decorrelation has not been further analyzed.

8.7.2 Input Data Source with Extended Jitter Model

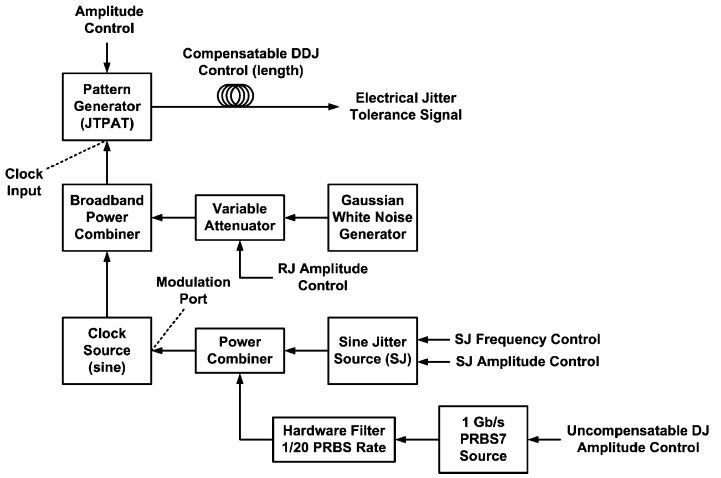

FIGURE 8.40. Block diagram of the electrical signal tolerance source. (After INCITS Technical Commitee [113].)

Time-domain generation of a data pattern exhibiting jitter characteristics with the previously introduced statistical properties is difficult. For this reason, we built a behavioral model of the data source recommended for hardware jitter tolerance compliance testing by most communication standards (Figure 8.40). The different components of this data source are discussed in the following. Most jitter contributions are applied to the locally generated reference clock. At each rising edge of this reference clock, the pattern generator emits a new bit of the data pattern.

Some of the contributions to deterministic jitter mentioned in Paragraph 3.7.1, especially those related to bandwidth limitations, can be compensated for by signal processing manipulations like predistortion and equalization. In order to obtain a close to real-life data pattern where part of the deterministic jitter can be compensated for, DJ is generated based on two independent mechanisms. The *compensatable deterministic jitter*, designated by the acronym DDJ (which stands for *data-dependent jitter*), is introduced by a bandwidth limitation mechanism applied at the output of the pattern generator. The *uncompensatable deterministic jitter* source, designated by UDJ, is built of a 1.0 Gb/s PRBS7 pattern generator, which data is sent through a 50 MHz bandwidth 1st-order low-pass filter.

$$\frac{v_{UDJ}}{v_{PRBS7}} = \tanh\left(\frac{7 \cdot T_{ck}}{2 \cdot \tau}\right) \cong \tanh(1.1) \cong 0.8 \qquad \text{(EQ 8.25)}$$

Due to low-pass filtering, the peak amplitude of the UDJ signal is lower than the input PRBS7 pattern as given by Equation 8.25 as a function of the longest run length (i.e. $7T_{ck}$ in PRBS7) and the filter time constant $\tau = \frac{20 \cdot T_{ck}}{2\pi}$. To compensate for this undesired attenuation, a gain block must be added after the filter. In the final model, this gain is numerically adjusted by simulation to improve the compensation according to the particular PRBS7 pattern. It is interesting to notice that the DDJ mechanism also results in partial vertical eye closure, while the UDJ mechanism does not.

The UDJ component is then combined with the sinusoidal jitter component to modulate the phase of the reference clock. When setting the parameters of the SJ component, care should be taken to avoid the sinusoidal jitter frequency to be derived from the clock frequency by an integer or rational division. Insufficient coverage of the stochastic properties of sinusoidal jitter may otherwise result in excessive worst or best case simulations. As a side comment, an interesting note on the validity of sinusoidal jitter testing at large jitter amplitudes has been presented in [114]. Indeed, as shown by the probability density function of a sine wave, the data edge density of the incoming data will be located close to the center of the eye for high SJ amplitudes (>0.5 UI), which may lead loop-based clock recovery systems to false lock.

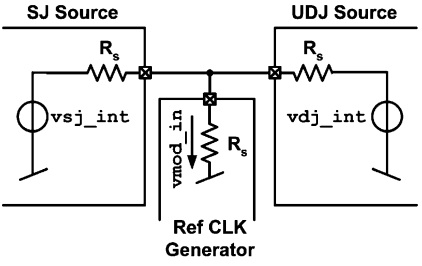

FIGURE 8.41. Equivalent schematic of the power combiners

After combination of the sinusoidal jitter component and the UDJ component, this signal is applied to the modulation input of the reference clock generator. The power combiners are modeled as Thévenin-equivalent voltage sources with a source resistance of 50 Ω driving a shared load resistance of 50 Ω (Figure 8.41), where the ideal voltage sources deliver twice the output voltage, as used in conventional measurement equipment. For a number *n* of sources driving the load according to this principle,

the attenuation of the individual signal is given by Equation 8.26, the equivalent code is shown in Figure 8.42.

$$v_{mod\_in} = \sum_{i=1}^{n} \frac{2 \cdot v_n}{1+n} \quad \text{(EQ 8.26)}$$

The values of the internal voltages `vsj_int` and `vdj_int` are twice the desired output voltages, in order for the output voltages `vsj_out` and `vdj_out` to correspond to the expected value when terminated with a matched load.

```
vsj_out == vsj_int + isj_out * 50.0;
vdj_out == vdj_int + idj_out * 50.0;
imod_in == isin_clk + idj_out
vmod_in == imod_in * 50.0;
```

FIGURE 8.42. Equivalent VHDL-AMS code of the power combiner

A sinusoidal clock source is used to allow proper phase modulation by the previously mentioned components (Equation 8.27). All voltages are centered around v_{mid} (for simplicity equal to ground, could also be set to $V_{DD}/2$), A_{clk} and A_{mod} are the respective amplitudes set for the reference clock and the voltage at the modulation input.

$$vsin\_clk = v_{mid} + A_{clk} \cdot \sin(2\pi f_{clk} t + d\phi)$$
$$\text{with} \quad d\phi = \frac{vmod_{in} - v_{mid}}{A_{mod}} \cdot k_{VCO} \cdot 2\pi \quad \text{(EQ 8.27)}$$

Figure 8.43 shows the corresponding code, where again the generated internal value is equal to twice the desired value. The `NOW` statement is used to recover the time information from the simulator and `dph` is the phase modulation represented by $d\phi$ in Equation 8.27.

```
vsin_clk_int == 2.0 * (0.0 + 0.9 * sin(MATH_2_PI*freq*NOW+dph));
```

FIGURE 8.43. Sinusoidal reference clock code

Random jitter is added in power to the modulated clock signal, which is then applied to the clock input of the pattern generator. The conversion from additive noise to jitter being illustrated in Figure 8.44, the conversion factor is given by the slope at the mid-point crossing of the clock signal (Equation 8.28).

$$\sigma_{AN} = RJ_{RMS} \cdot \frac{1\text{UI}}{\pi \cdot A_{clk}} \quad \text{(EQ 8.28)}$$

The principle of adding random noise is arguable due to the resulting invalid additional clock edges, as discussed in [115]. In our implementation, the probability of immediate successive zero crossings is somewhat mitigated by the hysteresis implemented at the clock input of the pattern generator. When applying larger amounts (e.g. 0.02 UI_{RMS}) of random jitter in simulation, the large number of invalid additional clock edges rendered the analysis of the bit error ratio and eye diagram impossible. For this reason, it is frequently recommended to add the RJ component to the combined SJ and UDJ components at the modulation input of the clock generator.

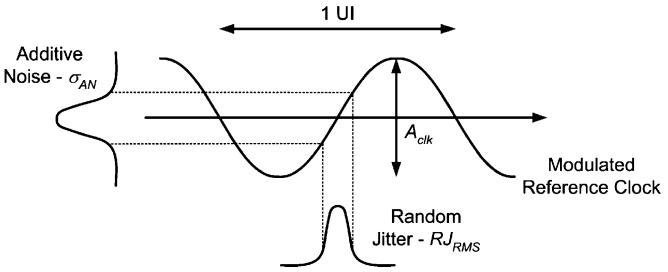

FIGURE 8.44. Additive noise to random jitter conversion. (After Baumer and Engelen [115].)

The pattern generator of the standardized jitter tolerance source for system compliancy verification generates a so-called *compliant jitter tolerance pattern* (CJTPAT), composed of the data words representing the worst-case condition for jitter tolerance measurement of the receiver. The *compliant* term indicates that the effective stress pattern is embedded into data frames including delimiters, fill words and CRC, which are required to test physical links. For verification of a CDR circuit, use of the *jitter tolerance pattern* (JTPAT), which corresponds to the payload of the CJTPAT, is recommended and implemented in the present model. In the current version of the model, the JTPAT is hardcoded as a data vector repeated iteratively (Figure 8.45).

At the output of the pattern generator, the resulting pattern is bandwidth limited by a first order filter of time constant τ, resulting in compensatable deterministic jitter. Considering a periodic pattern centered around its DC value, the filter bandwidth required to obtain a given amount of deterministic jitter is given by Equation 8.29, where $T_B = 1/f_B$ is the bit period and T_{MIN} and T_{MAX} respectively the shortest and longest run length in the pattern. Convenient cutoff frequency settings for some common values of deterministic jitter are proposed in Table 8.4. However, applying the JTPAT or a PRBS signal results in more severe jitter due to the transitions between long and short CID sequences, which are difficult to estimate mathematically. Some simulated correspondence points between cutoff frequency and DDJ for the JTPAT are also given in Table 8.4.

```
architecture dig of jtpat_core is
  constant d30p3_pat: std_logic_vector(19 downto 0) :=
  "10000111000111100011";
  constant d21p5_pat: std_logic_vector(19 downto 0) :=
  "10101010101010101010";
  signal counter : natural := 100;
begin
  gen_proc : process (clk, rst_b)
  begin
    if rst_b = '0' then
      data_out <= '0';
    elsif clk'EVENT and clk='1' then
      if counter < 100 then
        data_out <= d30p3_pat(counter mod 20);
      else
        data_out <= d21p5_pat(counter mod 20);
      end if;
      if counter = 130 then
        counter <= 0;
      else
        counter <= counter + 1;
      end if;
    end if;
  end process gen_proc;
end dig;
```

FIGURE 8.45. JTPAT pattern generator code

$$DDJ[UI] = \frac{\tau}{T_B} \cdot \ln\left(\frac{1 + e^{\frac{-T_{MIN}}{\tau}}}{1 + e^{\frac{-T_{MAX}}{\tau}}}\right) \qquad \text{(EQ 8.29)}$$

TABLE 8.4. Filter cutoff frequencies and associate DDJ values for $T_{min}=T_B$ and $T_{min}=5T_B$

f_c [MHz]	DDJ – periodicity assumption [UI_{PP}]	DDJ – JTPAT [UI_{PP}] (simulated)
292.0	0.50	
340.0	0.40	
400.0	0.30	
411.0		0.41
500.0	0.20	0.27
672.0	0.10	
1000.0	0.031	0.033
2000.0	0.001	0.001

Figure 8.46 shows the eye diagram of the data output by the JTPAT data source with different jitter components applied. Of course the simulator time step influences the accuracy of the obtained eye opening. Simulation of a 1μs pattern at 2.5 Gb/s (2′500 bits) with a maximum time step of 2 fs on a SPARC Ultra 10 (400 MHz CPU) takes approximately 7 min.

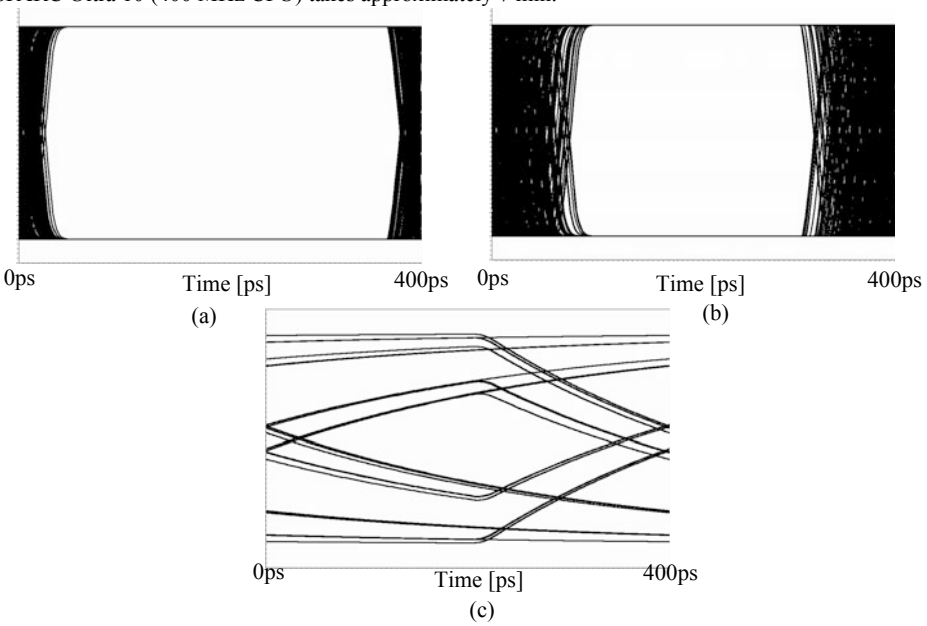

FIGURE 8.46. VHDL-AMS JTPAT data source with normalized amplitude for (a) SJ_{amp} = 0.1 UI_{PP} SJ_{freq} = 0.0211 f_B, (b) UDJ = 0.4 UI_{PP} and (c) DDJ = 0.4 UI_{PP}

The generated data pattern is sent to the gated oscillator-based clock recovery circuit. When aligning the data to be sampled (input signal of the sampling element) with respect to the sampling edges of the recovered clock, we obtain the eye diagram shown in Figure 8.47. Unlike an eye diagram obtained from the output data of the sampling element, this figure allows us to observe the remaining timing

margins with respect to the sampling instant located at 0.5 UI. The difference in jitter accumulation between both data edges is easily distinguishable.

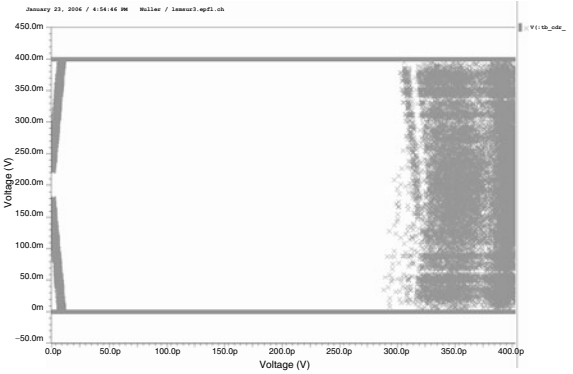

FIGURE 8.47. Eye diagram at the CDR output when applying data from the JTPAT source with SJ_{amp} = 0.1 UI_{PP}, SJ_{freq} = 0.0211 f_B, UDJ = 0.1 UI_{PP}, $DDJ \approx$ 0.1 UI_{PP} (f_C = 672 MHz), RJ = 0.001 UI_{RMS} and CKJ = 0.01 UI_{RMS} (no frequency offset)

Based on the presented models, the accurate behavior of the CDR circuit have been verified. Among other issues, the importance of the accuracy of the edge detection pulses has been analyzed and will be presented below. In combination with a bit error ratio test module (which can be included in the form of a VHDL or VHDL-AMS model), BER calculations can also be performed. Due to the large number of bits required to achieve statistical significance [117], the determination of the bit error ratio based on a given number of errors in the data stream however requires several weeks of simulation time on a state-of-the-art computing system.

8.7.3 Duration of Edge Detection Pulse

In the initial design, the edge detector delay was specified to be exactly half a nominal clock cycle, so that the reset signal would not disturb the oscillator duty cycle. Behavioral time-domain simulation showed however that this choice leads in presence of delay variations to a race condition between the oscillator output feedback and the EDET signal, resulting in the absence of synchronization. As shown in Figure 8.48, for synchronization to be effective, the clock edge generated in the oscillator by the falling EDET signal must propagate to the oscillator output and be fed back to the input gate before the

EDET signal may rise. Otherwise the information of this edge be used a single time for sampling the data, then vanish.

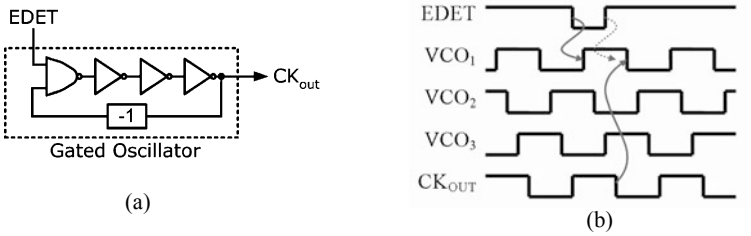

FIGURE 8.48. (a) Gated oscillator core, (b) race condition in the oscillator reset

This issue is solved by increasing the duration of the EDET pulses, achieved by increasing the delay of the delay line. However, this results in duty-cycle distortion of the recovered clock and an optimum trade-off between both issues is to be found. This optimum depends on the variability of the delay line delay, which can also be simulated by adding delay mismatch in the model. This example indicates the strength of combining statistical simulation with time-domain simulation, where both levels of abstraction have an important role to play in the complete design flow.

8.8 Transistor-Level Design

From statistical simulations we have determined the estimate of acceptable jitter performance of the oscillator, which can now be used for power-aware transistor-level design of the clock recovery circuit. As transistor-level clock recovery design is widely discussed in the literature, only the important issues of the design optimization of the gated oscillator topology will be presented here, completing the top-down design flow introduced earlier.

8.8.1 Current-Mode Logic Cells

The gated oscillator clock and data recovery circuit has been implemented using current-mode logic (CML) cells. The conventional CMOS logic design style has several drawbacks when operating at high speed:

- The load is the equivalent of two gate capacitances
- The signal swing is $V_{DD} - V_{SS}$
- The power consumption scales with f_{CLK}^2
- Supply and substrate noise is generated at the frequency of operation

According to Kwan and Shams [119], the differential signaling current-mode logic cells (Figure 8.49) can operate about two times faster than conventional CMOS cells at comparable power dissipation. This result probably has to be mitigated by the increased wire load in a larger design due to the increased number of routing constraints. Nevertheless, this logic family certainly has a speed advantage and comes with additional benefits:

- Reduced input capacitance
- Lower power supply and substrate noise
- Reduced sensitivity to environmental perturbations (capacitive or magnetic cross talk) due to routing symmetry.

As a result of the reduced noise sensitivity, this architecture can afford lower noise margins and thus operate at reduced signal swing. Beyond the intrinsic speed improvement resulting of reduced signal swing, this in turn also results in reduced capacitive coupling effects and closes the noise versus noise-margin loop. Additionally, a differential implementation of the CDR matches perfectly the differential limiting amplifier output interface.

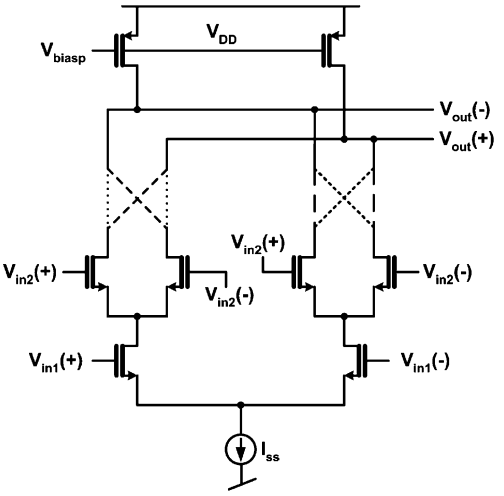

FIGURE 8.49. A simple current-mode logic cell (multifunction two-input gate)

Finally, a two-stage CML gate can perform multiple logical operations, only depending on the routing of its input nodes and output nodes of each highest stage.

8.8.2 Low-Power CDR Design

The four-stage current-controlled ring oscillator in the CDR is built using current-mode logic (CML) gates [116]. It was previously mentioned that completely switching differential stages do not offer the improved supply noise immunity over single-ended stages. However, in the case of oscillators, we are mostly concerned about phase noise and not about amplitude noise. As shown by Hajimiri et al. [108], the amplitude restoring mechanism of an oscillator renders it most sensitive to noise injection around the zero crossing instant. In this situation, the gain stage is in equilibrium and thus optimally rejects common-mode perturbations like supply noise. As a result, differential ring oscillator implementations provide optimal first-order rejection of supply and substrate noise.

In order to obtain perfect symmetry between all four stages, the two-stage CML gate performing the gating operation at the oscillator input is used for all stages (Figure 8.50). The tuning current delivered by the shared PLL determines the biasing of each stage, while the values of the load resistors operating in triode region are adjusted by a so-called *replica-bias circuit* to guarantee constant signal swing [120].

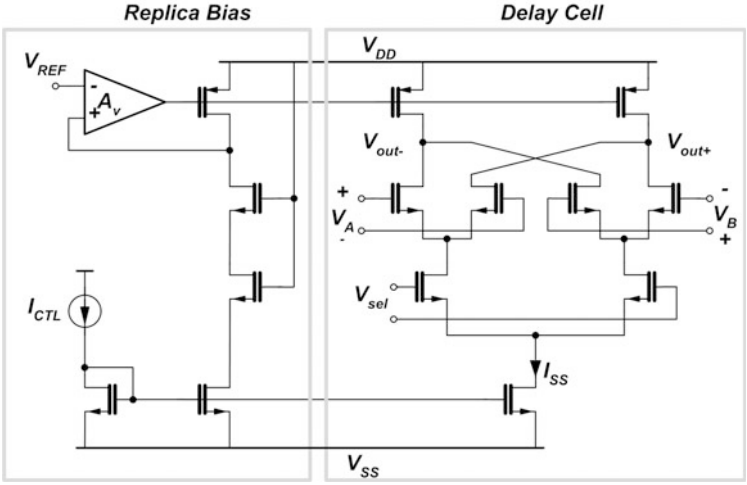

FIGURE 8.50. The two-stage CML gate used in the gated oscillator

From the oscillator jitter parameter κ_{min} determined in Paragraph 8.6.5, the bias current of one oscillator cell can be determined. Indeed, Hajimiri et al. [108] introduces the relationship between the κ_{min} and the oscillator bias current of one stage I_{ss}, as well as other parameters defined below:

$$\kappa_{min} = \sqrt{\frac{8}{3\eta} \cdot \frac{k_B T}{I_{ss}} \cdot \left(\frac{1}{V_{char}} + \frac{1}{V_{swing}}\right)} \qquad \text{(EQ 8.30)}$$

η is a proportionality constant between the rise/fall-time and the gate delay and is typically close to 1. k_B is the Boltzmann constant, while T is the absolute temperature expressed in Kelvin. The characteristic voltage V_{char} of the logic gate is given, according to the above-mentioned reference, by $(V_{GS} - V_T)/\gamma$ under the long-channel assumption, while it becomes $(E_c L)/\gamma$ in the short-channel regime, where γ is the excess thermal noise factor, E_c the critical electric field and L the channel length. The increase of the excess thermal noise factor in short-channel devices has been discussed in Paragraph 6.4.5 on page 91 and the use of $\gamma = 2$ has been suggested. V_{swing} is the maximum amplitude at the output of an oscillator stage determined by the setting of the reference voltage V_{ref}. V_{swing} has to be somewhat larger than the effective amplitude of oscillation, because the effective oscillation amplitude V_{eff} of an M-stage ring oscillator is given by:

$$V_{eff} = V_{swing} \cdot \frac{4}{\pi} \cdot \cos\left(\frac{\pi}{M}\right) \qquad \text{(EQ 8.31)}$$

The detailed development of the quasi-linear analysis leading to this result is presented in [121]. Based on the expression of κ_{min}, we can design the circuit to operate at the minimum power level required by the jitter specifications. The specified bias current and signal swing determine the value of the load resistor. Sufficient gain per stage is obtained by correct dimensioning of the differential pair transconductance, which in return also determines the load capacitance seen by the preceding stage. In order to obtain a good wire load model, post-layout simulation is essential. The wire parasitics indeed contribute to around 30% additional load capacitance compared to the schematic design. The simulated eye diagram in Figure 8.51 shows the output data after retiming with the recovered clock signal, which triggers the sampling flip-flop.

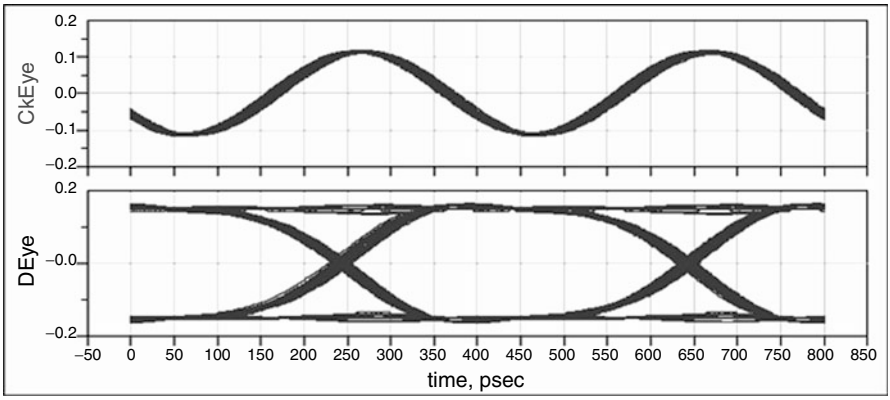

FIGURE 8.51. Eye diagram of the output data (bottom) with the recovered clock (top) [118]

A seven-channel CDR has been fabricated in a 0.18 μm CMOS technology, occupying a silicon area of 45′000 μm^2 per channel (Figure 8.52). For test purposes, it uses three separate replica bias circuits, controlling respectively the delay line, the oscillator and the fixed-delay gates like the sampler. Merging the delay line bias circuit with the fixed delay bias circuit would result in a 10% silicon area reduction.

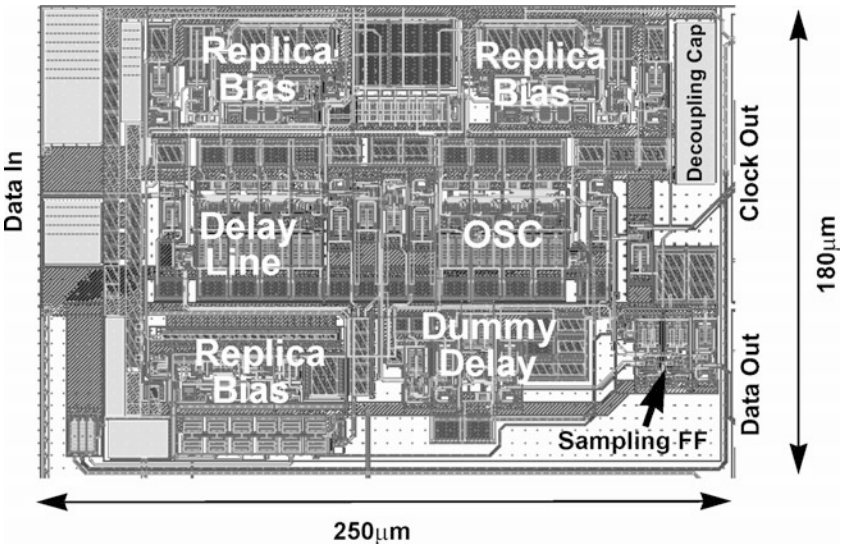

FIGURE 8.52. Gated oscillator CDR layout [118]

While the shared PLL occupies 0.1 mm^2, one CDR channel consumes 7.2 mW and the PLL 10.2 mW from a 1.8 V power supply.

8.8.3 Scaling Theory

As the presented topology focuses on area and power awareness, it is interesting to estimate its potential when porting to smaller feature size technologies. This development is based on first-order models neglecting some device parameters that do not scale perfectly. In deep submicron technologies, we must consider a scaling approach where dimensions and voltages scale differently [122]. Let us consider the scaling factors *a* for dimension scaling and *b* for voltage scaling, where both parameters are larger than one in a smaller feature size technology. In high-speed wireline circuits, the large majority

of the devices operating not in, but close to the velocity saturated regime. Using the assumption of velocity saturation to simplify the analysis, we obtain the scaling rules presented in Table 8.5.

TABLE 8.5. General scaling rules for short-channel devices

Parameter	Scales with
W, L, t_{ox}	$1/a$
C_{ox}	a
V_{DD}, V_{T0}	$1/b$
Area	$1/a^2$

Under the same assumption, the transconductance can be approximated by Laker and Sansen [26],

$$g_{msat} = WC_{ox}\mu\varepsilon_c \quad \text{(EQ 8.32)}$$

where ε_c is the critical longitudinal electric field and μ the mobility, which are considered constant with scaling. The transit frequency f_T, an indicator of the intrinsic device speed has been defined in Equation 4.1, reproduced in Equation 8.33. As the gate-source capacitance in saturation is proportional to the total gate capacitance, constant transconductance and scaled gate capacitance result in f_T increasing proportionally with K.

$$f_T = \frac{g_m}{2\pi \cdot C_{gs}} \sim \frac{WC_{ox}\mu\varepsilon_c}{2\pi \cdot C_{ox}WL} = \frac{\mu\varepsilon_c}{2\pi \cdot L} \sim a \quad \text{(EQ 8.33)}$$

As the velocity saturation operation is characterized by a linear relationship between the drain current and the gate voltage, the corresponding drain current I_{Dsat} is given by $g_{msat}(V_G\text{-}V_{T0}\text{-}nV_S)$, scaling as $1/b$. In conclusion, the area of the CDR scales as $1/a^2$, the power consumption as $1/b^2$, while the data rate is increased by a.

For illustration purposes, let us estimate the performance improvements when porting the current 0.18 μm design into a 90 nm CMOS technology. This step corresponds to two major technology generations, represented by $a = 180/90 = 2$ and $b = 1.8/1.2 = 1.5$. The new design is expected to operate at 5 Gb/s, occupying an approximate area of $11'000\mu m^2$ and consuming 4.8 mW, resulting in a normalized area of $2'200\mu m^2$ /Gb/s and a normalized power consumption below 1 mW/Gb/s. As speed/power and speed/area figures improve when data rates get closer to the technology limits, it can be safely assumed that a 90 nm GO-CDR operated at 10 Gb/s would achieve even better performance with respect to these metrics.

8.9 CDR Measurements

Now that we have seen how the performance of a particular CDR topology can be evaluated beforehand and how the jitter specifications can be propagated down to the transistor-level design, let us discuss the measurement techniques which allow for the characterization of such a CDR circuit.

8.9.1 Eye Diagram Measurements

Eye diagram measurements show the output data in an eye diagram, triggered by the recovered clock (Figure 8.53). In the measurement shown below, the clock recovery circuit receives a PRBS7 input data stream at 2.5 Gb/s and delivers both the recovered clock and retimed data signals. Proper measurement of the receiver jitter tolerance requires jitter to be fed into the pattern generator for phase modulation. Alternatively a complete link using a nonideal transmitter and a nonideal communication medium can be measured, varying the transmit power and/or the transmit jitter characteristics.

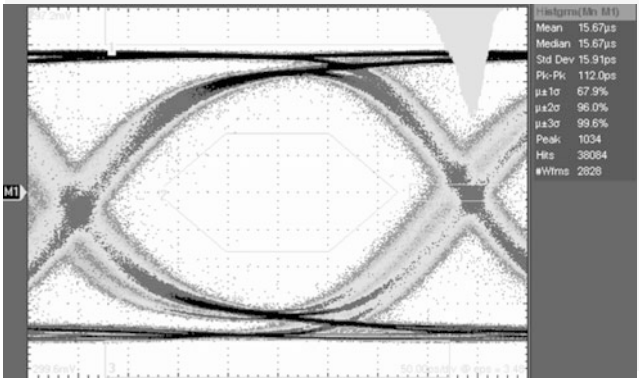

FIGURE 8.53. Measured eye diagram of a gated oscillator CDR output data operated at 2.5 Gb/s

The receiver output is typically taken after the retiming flip-flop. While this is the correct procedure for analyzing the bit error ratio of the complete receiver, it does not allow for direct determination of the receiver's jitter tolerance based on the eye diagram. Determination of JTOL from an eye diagram requires the input of the retiming element to be fed to the oscilloscope, where it is sampled with the recovered clock signal.

However, even in this configuration, exact eye diagram measurement is not always possible due to the way the oscilloscope uses the trigger signal. In the case of a *sampling* oscilloscope, the delay between the trigger event and the first sample is in the nanosecond range. This does not allow the scope to take into account short-term variations of the clock frequency as they appear due to the very short-term

synchronization of the gated oscillator CDR. *Real-time* oscilloscopes provide various sampling modes. In *single-shot* configuration, a user-defined number of samples is taken based on a single trigger event, which again does not consider individual sampling events. In *sequence* mode, acquisition of a single bit interval based on a trigger event is possible, but the dead-time between samples results in the loss of a large number of bits in the sequence.

8.9.2 Bit Error Ratio Measurements

Accurate estimation of the CDR performance and in particular the bit error ratio at the CDR output, as well as related jitter tolerance, can only be obtained from a bit error ratio tester (BERT), which compares the input and output data streams of the device under test (DUT, Figure 8.54).

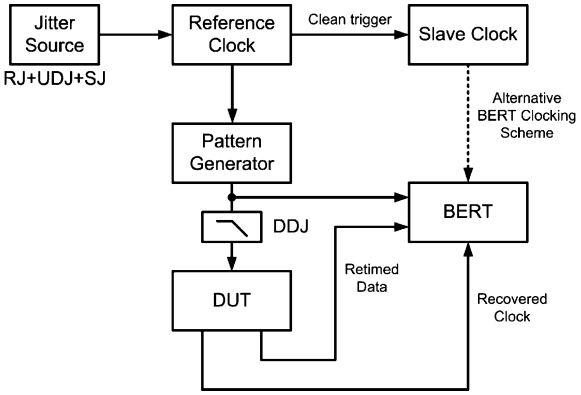

FIGURE 8.54. BER test measurement setup

In this scheme, data jitter is applied to the reference clock source, which triggers the pattern generator. The data pattern, known by the BERT, can be fed through a low-pass filter to add deterministic jitter. For reasons of setup complexity, the explicit addition of a DDJ component may in some cases be omitted, while the limited bandwidth of the board-level routing contributes some DDJ to the input data. The data pattern is applied to the DUT (i.e. the CDR), which recovers the clock signal from the received data and samples the data with this recovered clock signal. The same recovered clock signal is then used to determine the sampling instants in the BERT. By adjusting the BERT sampling delay between recovered clock and retimed data, the minimum BER value can then be determined. In the case of the GO-CDR, the possibility of using the recovered clock as a trigger for the BER tester depends on the shape of the recovered clock, which is subject to duty-cycle distortion inherent to this topology. In this situation, a stable slave clock can be used to trigger the BERT under certain conditions on jitter and frequency offsets ("Alternative BERT Clocking Scheme" in Figure 8.54).

An exhaustive measurement of the sampled eye opening requires the possibility to tune the sampling instant and the threshold voltage inside the retiming part of the CDR, using, e.g. a tunable delay line and a variable decision threshold. In such a configuration, the BERT can sweep the complete eye and determine the error ratio at each point of the eye, resulting in excellent knowledge of the design margins.

The gated oscillator CDR measurements revealed an interesting phenomenon. While short-term measurements (e.g. 5 min) result in zero detected bit error, longer-term measurements reveal a higher bit error ratio. Indeed, the bit errors appear by bursts of several million errors a time, the repetition rate of the bursts being related to the center frequency of the reference PLL and thus to the control current of the CDR (Figure 8.55, measured over intervals of 5 min).

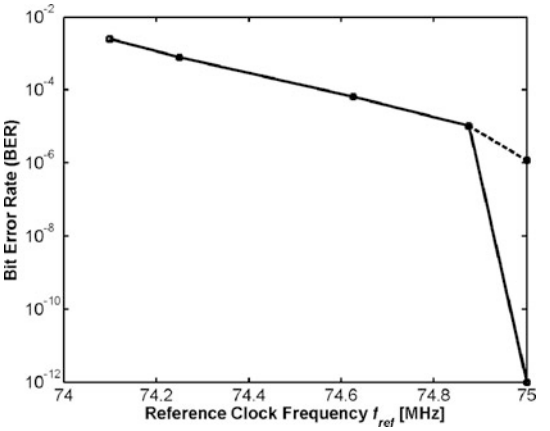

FIGURE 8.55. BER sensitivity to CDR center frequency measured over 5 min (dashed result is measured over 30 min)

Tuning of the center frequency resulted in a single error interval during measurements done over a duration of 30 min, equivalent to 99,9444% of error free intervals, however also equivalent to a BER of $1.147\ 10^{-6}$. We believe that residual frequency offset combined with ambient deterministic and random jitter result in eventual drop of a sampling event due to a very short clock pulse. Unlike the case of regular bit errors, this "lost" bit results in loss of synchronization in the PRBS sequence and thus appears as a large amount of bit errors in the BER result. As expected, the channel that was imple-

mented with a three-stage delay line instead of a four stage delay line experienced dramatically higher BER due to lacking synchronization.

TABLE 8.6. Jitter components and BER measured over 30 min

Jitter component	Jitter amplitude	BER	# Error intervals
no jitter		1.1473e-6	1
RJ_{RMS}	0.0077 mUI		
UDJ_{PP}	0.185 UI		
DDJ_{PP}	–		
SJ_{PP}	0.74 UI		
f_{SJ}	50 kHz	3.7574e-6	2
	2 MHz	4.0455e-6	3

This fact renders accurate jitter tolerance measurements very difficult, as the measurement duration required to achieve statistical significance in presence of such sporadic events is very long. Nevertheless, we tested the jitter tolerance up to the maximum jitter amplitude accepted by the pattern generator and the maximum sinusoidal jitter frequency achievable with the arbitrary waveform generator used. The jitter parameters applied and the corresponding BER results obtained in measurements over 30 min are shown in Table 8.6. Again, bursts of bit errors strongly determine the bit error ratio and limit the statistical significance of these measurements. It has not been verified in how far the combination of the CDR with an appropriate elastic buffer would suffice to suppress these error bursts. Table 8.7 summarizes measurement results of the gated oscillator CDR, implemented in a 0.18 μm CMOS technology.

TABLE 8.7. CDR simulation and measurement results overview

Specification	Simulated	Measured	Unit
Supply voltage	1.6–2.0	1.8	V
Data rate	2.5	1.0–2.5	Gb/s
Clock Jitter @ 2.5 GHz			
w/o trimming	4	12	ps
with trimming		4.1	ps
Bit error ratio/error free intervals	@ 2.5 Gb/s	@ 1.0 Gb/s	
no jitter	$< 10^{-12}$ / –	1.1473e-6 / 99.9444	– / %
jitter according to Table 8.3, FTOL = 0	$< 10^{-11}$ / –		–
jitter according to Table 8.6		4.0455e-6 / 99.8333	– / %

TABLE 8.7. CDR simulation and measurement results overview

Specification	Simulated	Measured	Unit
Current Consumption @ 25°C, 1.8 V			
PLL	10.5 @ 2.5 Gb/s		mW
CDR - 1 channel	8.5	~ 60mW @ 2.5Gb/s	mW
Output Drivers - 1 channel	~2 x 23.5		mW
Bias	~1.5		mW
Area			
PLL	0.10		mm^2
CDR (w/o Drivers)	0.06		mm^2

8.10 Discussion

In this chapter, we presented different CDR topologies with their advantages and drawbacks when used in the context of short-distance multichannel data transmission. A jitter-estimation-based top-down design flow was introduced and applied to the design of a gated oscillator based clock recovery circuit. The flow highlights the strength and limitations of the studied topology at an early stage. While the time-domain simulation allows for functional verification of the design, the specification propagation to the transistor-level leads to a hardware implementation that meets the high-level specifications. Furthermore, the consistent oscillator jitter specifications available to the designer result in a power-jitter optimized schematic, essential to the achievement of the power constraints. In a general fashion, such a specification-driven design approach can be applied to the analysis and design of most CDR topologies and delivers consistent early performance estimates for such a design.