RESEARCH AND DESIGN OF LOW JITTER, WIDE LOCKING-RANGE ALL-DIGITAL PHASE-LOCKED AND DELAY-LOCKED LOOPS

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Feng Lin

March 2000

Major Professor: Russel Jacob Baker, Ph.D.

AUTHORIZATION TO SUBMIT

DISSERTATION

This dissertation of Feng Lin, submitted for the degree of Doctor of Philosophy with a major in Electrical Engineering and titled "Research and Design of Low Jitter, Wide Locking-Range All-Digital Phase-Locked and Delay-Locked Loops," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor ________________________________Date ________________
R. Jacob Baker

Committee Members ____________________________Date ________________
Harry Li

____________________________Date ________________
Richard Wells

____________________________Date ________________
James Frenzel

____________________________Date ________________
Larry Stauffer

Department
Administrator __________________________________Date ________________
Dave Egolf

Discipline's
College Dean ___________________________________Date ________________
David E. Thompson

Final Approval and Acceptance by the College of Graduate Studies

___________________________________Date _______________
Charles R. Hatch

Abstract

PHASE-LOCKED loops (PLLs) and delay-locked loops (DLLs) are often used in integrated circuits in order to compensate for clock distribution delays and to improve overall system timing. PLLs are also widely used in clock recovery and frequency synthesis. When compared to traditional implementations of PLLs and DLLs, an all-digital approach will be found more suitable for monolithic implementation on the same die with other digital circuits. A robust, process-independent performance is expected using all digital techniques.

In this dissertation, several aspects of phase-locked and delay-locked loops are investigated, including building blocks, loop dynamics, noise and jitter. General design criteria are summarized for the all-digital implementation with the comparison to the traditional approaches and popular charge-pump analog implementation.

An all-digital phase-locked loop (ADPLL) using a proposed register-controlled oscillator (RCO) and all-digital phase frequency detector (PFD) is developed and fabricated using 0.18um CMOS technology. The two-loop architecture, hierarchy pull-in process and fine phase adjustment make this RCO-based ADPLL achieve less than 80-cycle lock time, 65MHz-385MHz lock range, 30ps RMS jitter and less than 2% duty cycle distortion when the reference clock is at 200MHz. This ADPLL also shows stable operation when power supply voltage is down to 1.4V, which gives more flexibility in low power applications without significant design modification.

A register-controlled symmetrical DLL (RSDLL), targeted for clock synchronization and de-skewing in double-data rate synchronous DRAM, is implemented based on a symmetrical register-controlled delay line. This RSDLL was fabricated using 0.21μm CMOS technology and achieved 50ps RMS jitter when the operating frequency is in the range of 125MHz to 250MHz. This approach eliminates extra circuitry for duty cycle correction when using both rising and falling edges to latch data. Measurement results are presented to verify its robust operation under different voltage and temperature conditions.

Acknowledgements

I'd like to thank my dissertation advisor Dr. Jacob Baker for his direction, encouragement in this work. Micron Technology Inc. (MTI) funded this project, manufactured the chip and provided measurement supports, which deserves my special thanks. I'd also like to thank Brent Keeth and Ron Harrison for their valuable discussion; Aaron Schoenfeld for his test setup; and Barbara Cobb, Jim Fabbri, Jeremy Gum, and Suzy Mcdonald for their layout support of this work.

During the course of this degree, my wife Hui has been very supportive and understanding. My mother Jingyi and father Chunxun deserve many thanks for their support and help.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

A Phase-Locked Loop (PLL) is a feedback system that causes an output signal to track a reference signal in phase and frequency. When both the frequency and phase of these signals are synchronized, the PLL is said to be in the locked condition. The phase difference between the output and reference is a known value when the loop is locked. Hence the name, "phase-locked loop". PLLs are found in a wide range of applications, including clock recovery, noise and jitter suppression in communications, clock synchronization in memory interface and high-performance microprocessors, and frequency synthesis for instrumentation and RF transceivers.

Another closely related circuit is the delay-locked loop (DLL), which is primarily used for clock synchronization and de-skewing. In high-speed synchronous systems clock skew can be a critical problem caused by clock buffers and propagation delay when the system clock enters a chip from a printed-circuit board (PCB), in Figure 1.1. The buffer is used to sharpen the clock edges and increase the drive capability needed because of the parasitics associated with the interconnect. In order to correctly latch the data and avoid setup / hold time violations of data latches, a PLL or DLL is needed to provide dynamic synchronization of the external (*CLKExt*) and internal (*CLKIn*) clocks, reduce the skew, and improve the system timing for on-chip and inter-chip operations.

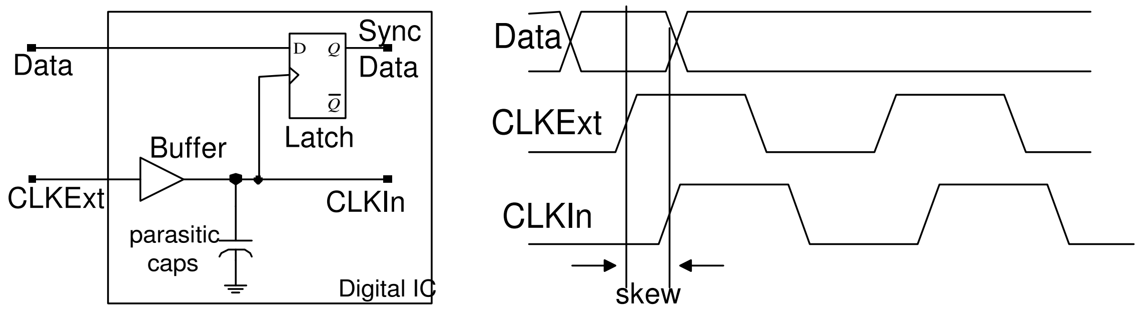


Figure 1.1. Clock skew in an integrated circuit

The theory of phase-lock techniques has been investigated thoroughly by Gardner in [1]. Best, in [2], classified the PLL into four different categories based on different implementations: the linear PLL, classical digital PLL, all digital PLL and software PLL. With tabulated equations and handy software to demonstrate loop responses on different design parameters, this book is found useful for discrete implementations. A tutorial to integrated circuit (IC) related implementation [3] illustrated recent efforts to build phase-locked loops and clock recovery circuits monolithically. Current trends in PLL design techniques are trying to solve the following problems using state-of-the-art IC techniques:

1. Achieving robust operation.
2. Achieving wide locking range.
3. How to get better jitter performance?

Since the basic PLL system is at least a second-order system, its output is susceptible to power supply, process and temperature variations. Traditional phase-locked and delay-locked loops are implemented using analog and digital techniques with some off-chip components to achieve robust operation or tuning. Advances in very large scale integration (VLSI) technology have provided the capacity to implement PLLs and DLLs monolithically with high performance and low cost. Charge-pump PLLs [4] [5], appeared in the early 80s and used a phase-frequency detector to provide attractive tracking and locking performance. Since the tracking range is only limited by the voltage-controlled oscillator (VCO) and the static phase error is zero if mismatches and offsets are negligible, the charge-pump circuit is widely used in many implementations, [6] – [13], of phase-locked and delay-locked loops. Various methods have been proposed to improve the efficiency of charge-pump circuit. Since a higher-order system is inherent in charge-pump PLLs [4] [5], a robust charge-pump PLL design can be a challenge. One implementation proposed a self-bias technique to track the variations caused by process and temperature [15] at the cost of power supply sensitivity. A charge-pump PLL using a current-controlled oscillator is reported in [16], [32]. References [17] [18] analyzed low jitter implementations using charge-pump circuits. Charge-pump phase-locked and delay-locked loops may be considered as analog implementations.

On the other hand, all-digital implementations offer enhanced functionality and flexibility, greater simplicity, and good stability. Here, all digital means the entire system is built with digital components. All digital implementation is primary used in a DLL to provide accurate timing [21]-[27]. A counter-based all digital phase-locked loop (ADPLL) is analyzed in [2]. A digital-controlled oscillator (DCO) uses high frequency source to provide a reference. The operating range is limited by the high frequency reference. Another DCO-based ADPLL is reported in [19] [20]. As the feature size of modern VLSI technologies continue to shrink, fully integrated all-digital approaches will be more scalable and portable in different processes and applications than its analog[1] counterpart. Advantages and disadvantages of both approaches are listed in Table 1.1.

	ALL-DIGITAL PLL	**ANALOG PLL**
Stability (relative)	Good	Poor
Scalability	Good	Poor
System Order	≤ 1	≥ 2
Simplicity	Good	Poor
Drift in Power Management	No Drift	Drift
Tuning	Discrete	Continuous
Lock Range	Limited	Wide
Immune to Variations in Process, Temperature and Power Supply	Good	Poor
Area without passive elements	Large	Small
Noise & Jitter	Predictive	Design Oriented

Table 1.1. Comparisons between two different implementations of PLLs

For a phase-locked loop, wide lock range and low jitter may not be achievable at the same time because, conventionally, the PLL is made narrow bandwidth to suppress the noise. In order to achieve wide lock range, aided frequency acquisition can be used to form a two-loop architecture, with the frequency tracking first, then phase locking [12], [19]. Aided acquisition can substantially increase the lock range. Another way to improve jitter performance is interpolation. The resolution of basic delay element used in the VCO or voltage-controlled delay line can be improved using interpolation [13], [17], [18], [24], [28], [31]. Jitter performance can also be improved by carefully examining the jitter associated with the phase-locked and delay-locked loops [14], [37]-[43].

Application oriented implementation results in different solutions. Besides widely used in the system timing, clock synchronization and clock generating, phase-locked or

[1] the analog PLL mainly refers to Charge-Pump phase-locked loops (CPPLL).

delay-locked loops also have applications in disk drives, and clock / data recovery circuits [32]-[36]. Although complementary MOS (CMOS) implementations are relatively low cost and widely used in current integrated circuit design, in order to achieve high speed, BiCMOS, Bipolar and GaAs may be used to operate PLL in the GHz range [44]-[50].

The objective of the research presented in this dissertation is to develop low jitter, wide lock range phase-locked and delay-locked loops using all digital techniques. Both all digital PLLs and DLLs used in clock synchronization applications are examined in detail. The all digital PLL (ADPLL) implemented in this work can be used in frequency synthesis and clock recovery applications. In this research, an emphasis has also been placed on CMOS implementations.

Chapter 2 first reviews the basic operation involved in phase-locked and delay-locked loops and presents an overview of various building blocks and a linear approximation of second-order system analysis. Some design criteria of PLL and DLLs are explored in Chapter 3, including jitter analysis associated with oscillators. Some tradeoffs in gain, bandwidth, loop dynamics, lock range and lock time are also discussed in detail. Existing approaches, including conventional charge-pump PLL (analog PLL) and counter-based or DCO-based all digital implementations are examined in Chapter 4. Pros and Cons are summarized which lead to the proposed register-based solution presented in this dissertation.

Chapter 5 describes the architecture, circuit design, and implementation of a register-controlled symmetrical DLL (RSDLL) that is suitable for double-data rate (DDR) dynamic random access memory (DRAM). The measurement results verify robust operations and less sensitivity to variations in process, temperature and power supply with this all digital implementation. The simple register-controlled mechanism used in the RSDLL is further explored in a proposed register-controlled oscillator (RCO), which is the main component used in the ADPLL. The RCO-based ADPLL is distinguished from the conventional voltage-controlled or digital-controlled oscillators. The two-loop architecture, specific lock-controlled mechanism and some techniques proposed to improve the resolution of this RCO-based ADPLL are discussed in Chapter 6 and applied to the design of ADPLL for de-skewing and frequency synthesis applications. Some experimental results extracted from a test design provide the proof in silicon of the operation of the RCO-based ADPLL.

Chapter 7 summarizes the principal conclusions and contributions of this research. Suggestions for further investigation are also offered in this chapter.

Chapter 2

Phase-Locked Loop Basics

2.1. Basic operation of a Phase-Locked Loop (PLL)

A basic PLL is a negative feedback system that consists of a phase detector, a low-pass loop filter and a voltage-controlled oscillator (VCO), as shown in Figure 2.1. A divide-by-N counter can be added in the feedback path to provide a frequency synthesis function. Figure 2.2 shows the waveforms of a reference signal and PLL output when the loop is in lock (using a divide-by-2 in the feedback path). The frequency of the PLL output signal is twice the reference with zero-phase difference (assuming N=2).

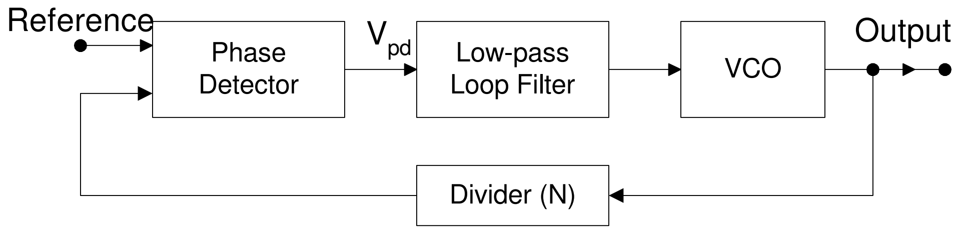


Figure 2.1. Block diagram of a basic PLL

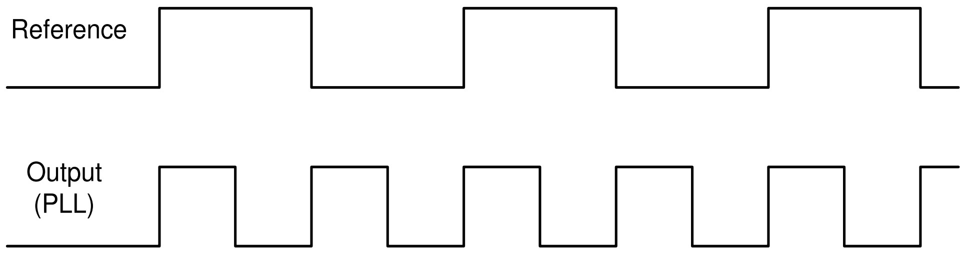


Figure 2.2. PLL in lock (N=2 in this case)

The basic operations of a PLL can be divided into three steps. First, the phase detector (PD) catches the phase difference between two inputs and generates an error signal V_{pd} whose average value is linearly proportional to the phase difference. A Loop filter (LF) is then used to suppress the high-frequency components of the PD output, allowing the average value (DC or low-frequency) to control the VCO frequency. Finally, an oscillator generates an output signal whose frequency is a linear function of the control signal out of the LF. The generated signal is fed back to the input of the PD and another phase comparison is started until the phase difference achieves a fixed relationship (most commonly 0° or 90°, depending on the nature of the PD). Some important aspects of the basic PLL need discussion for further investigation:

- Phase is the interesting value in a PLL rather than the voltage and current in general feedback circuits. It can be changed to voltage or current during the different steps of the PLL operation.
- The loop is said to be locked if the phase difference is constant with time. In other words, the frequency is equal between the two compared signals.
- In general, a PLL exhibits non-linear locking behavior.
- A tiny frequency difference can accumulate fast and the phase error can grow significantly after several cycles.
- A finite time is required for the PLL output to respond to input changes.
- The PLL output is a periodic signal, while the reference could be periodic, frequency or phase modulated signals, or data (1s and 0s).

2.2. Building Blocks in a PLL

From the basic operation of a PLL, three important building blocks are needed to provide phase locking, namely a phase detector (PD), loop filter (LF) and oscillator. Some main features and various implementation methods for different applications are investigated in this section.

2.2.1. Phase Detector

A simplified model of a phase detector (PD) is shown in Figure 2.3. The PD converts the input phase difference $\Delta\phi$, which is the subtraction of the reference phase (ϕ_{ref}) and PLL output phase (ϕ_{pll}), to an output signal V_{pd} with a gain factor K_{pd}.

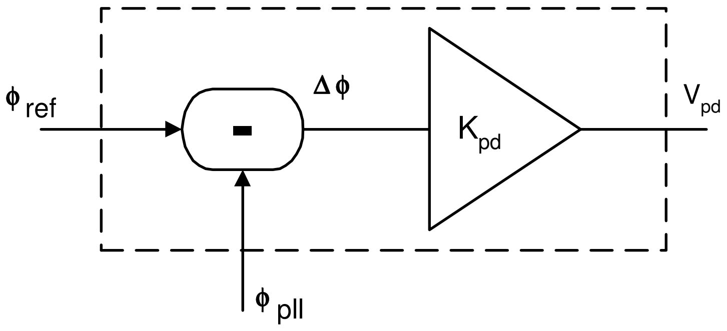


Figure 2.3. Simplified Model of a Phase Detector

The V_{pd} is an average value over cycles, which can be obtained using proper loop filters. A transfer characteristic curve can be derived from the relationship of input (phase) and average output (voltage or current). Since only the linear region, which refers to the steady state of a PLL, is a concern in deriving the PLL linear model, an equation can be used to illustrate such relationship with a gain factor K_{pd}, the slope of the PD linear transfer curve.

$$V_{pd} = K_{pd} \bullet \Delta\phi \tag{2.1}$$

Different types of PDs have their own transfer characteristics and are summarized in Table 2.1. The PD can be classified based on different applications and implementations. In Table 2.1, the two inputs, *R* and *V* refer to the reference (either data or clock) and PLL output respectively. The value V_{PD} stands for average output of the PD, and V_H, V_L are the maximum and minimum average values of V_{PD} respectively. The average output of the PD can be a current when uses in the PD Class IV with a Charge-Pump circuit to generate proper control signal for the VCO.

In deriving the transfer function of the Class I PD, assume that $R = A_R \cos(\omega_R t + \Delta\phi)$ and $V = A_V \cos(\omega_V t)$, then the output of this PD is the multiplication of *R* and *V*

$$\begin{aligned} V_{\mathrm{pd,\,out}} &= \mathrm{m}\cdot(\frac{\mathrm{A_R}\cdot\mathrm{A_V}}{2})\cos[(\omega_R+\omega_V)\cdot t+\Delta\phi]+ \\ &\quad \mathrm{m}\cdot(\frac{\mathrm{A_R}\cdot\mathrm{A_V}}{2})\cos[(\omega_R-\omega_V)\cdot t-\Delta\phi] \end{aligned} \tag{2.2}$$

where m is a multiplication factor. When PLL is in lock or in the steady state, $\omega_R = \omega_V$ and the integration over one period results in

$$V_{pd,average} = A\cdot\cos(\Delta\phi) = A\cdot\sin(\frac{\pi}{2}-\Delta\phi) \tag{2.3}$$

where *A* is a gain factor which is equal to $m(A_R A_V/2)$ and dependent on the input amplitudes. Equation (2.3) shows that the slope of this PD is not constant. When the phase difference is small compared to π/2, sin(π/2-Δϕ) is approximately equal to (π/2-Δϕ) and $K_{PD} \approx A$. When large signals are used in the Class I PD, a similar transfer function can be derived as XOR PD, where K_{PD} is linear and independent of the signal amplitudes.

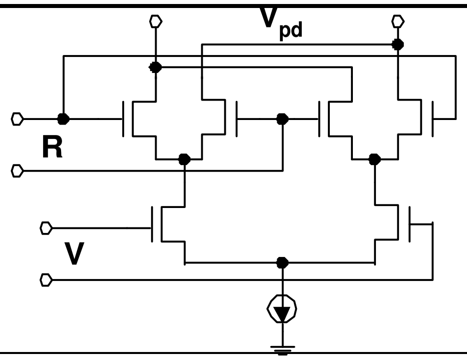
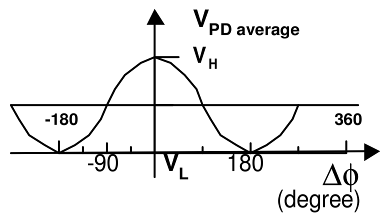
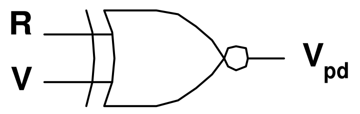
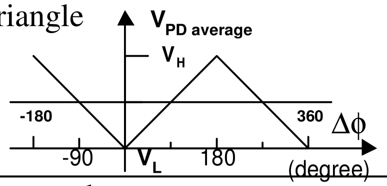
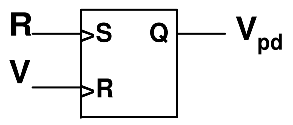
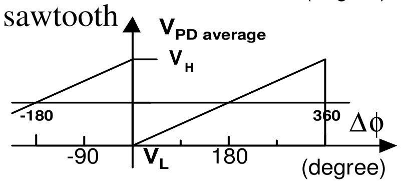
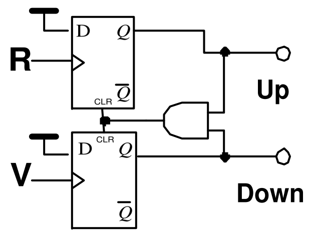
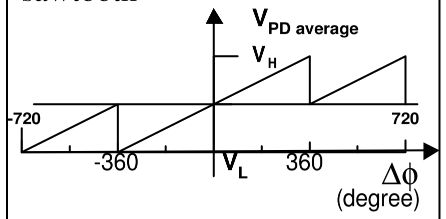
	CLASS	CIRCUIT SYMBOL	PD GAIN (K_{PD})	TRANSFER CURVE	APPLICATIONS
Multiplier (Mixer, Gilbert Cell)	Analog I		Nonlinear and proportional to the amplitude of input signals	sinusoidal 	Frequency Modulation and Demodulation
XOR	Digital II		$\frac{V_H - V_L}{\pi}$	triangle 	Data & Clock Recovery
RS Latch	Digital III		$\frac{V_H - V_L}{2\pi}$	sawtooth 	De-skewing
PFD (phase frequency Detector)	Digital IV		$\frac{V_H - V_L}{4\pi}$	sawtooth 	Clock synchronization, frequency synthesis

Table 2.1 Circuit symbols and transfer characteristics of different phase detectors based on applications

Class III and Class IV are edge-triggered phase comparison, which is sensitive to missing edges. For data recovery applications, these types of PDs may not be suitable without extra circuits to prevent false detection. Class IV PD can perform both phase and frequency detection (PFD) [51], which is powerful in many applications. The state diagram of this sequential detector is shown in Figure 2.4. Three possible states, *O*, *D* and *U* are embedded with this PFD, also known as classic 3-state PFD. The symbol '^' indicates that rising edge of signal *R* or *V* is coming. After "*Reset*", the PFD goes to State *O*. With the memory of previous state, the PFD can generate proper *Up* / *Down* signals even if the frequencies of *R* and *V* are different and it will not lock on the harmonics. Another observation is when PFD-typed PLL is in lock, *Up* and *Down* are both low and no ripples generated by the loop filter. For XOR-based PLL, because *R* and *V* are 90 degree out of phase when in lock, ripples will be generated when the output of the PD passes through the loop filter and modulate the VCO output. Some design considerations of different PDs are summarized in Table 2.2. These features are mainly related to the steady state operation of a PLL, instead of transient or dynamic response.

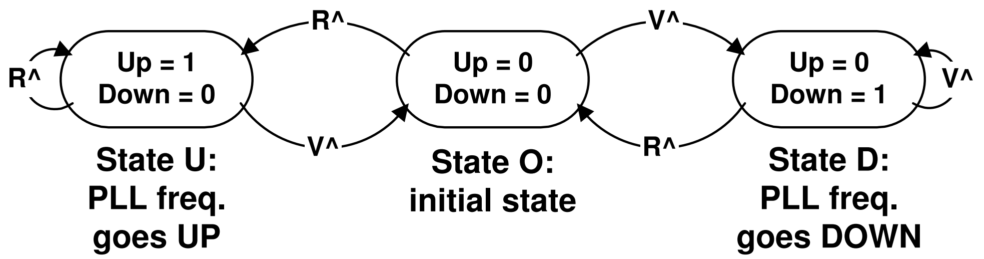


Figure 2.4. State diagram of a Phase Frequency Detector

	MULTIPLIER	XOR	RS-LATCH	PFD
Linear Range	Small	π	2π	4π
Edge Sensitive	No	No	Yes	Yes
Duty Cycle Sensitive	Yes	Yes	No	No
Lock to Harmonics	Yes	Yes	Yes	No
Sequential (w/memory)	No	No	No	Yes
Phase Detection	Yes	Yes	Yes	Yes
Frequency Detection	No	No	No	Yes
Pattern Sensitive	Yes	Yes	No	No
Ripple out of LF	Yes	Yes	No	No
Locking Phase	$\pi/2$	$\pi/2$	0	0

Table 2.2 Summary of lock characteristics of different phase detectors

To illustrate some of the lock characteristics, assume the frequency of V (f_V) equal to 2 times the frequency of R (f_R), and R is leading of V. The responses of different PDs are drawn in the Figure 2.5. From part (a), an XOR is used as the PD. For each cycle, the average value of PD output is equal to $(V_H-V_L)/2$, depending on what kind of power supply is used. In this case, the XOR-PD will declare locking! As a result, the XOR-PD typed PLL has some limitation on the oscillator, whose operating frequency has to be limited between $0.5\,f_R$ and $2\,f_R$.

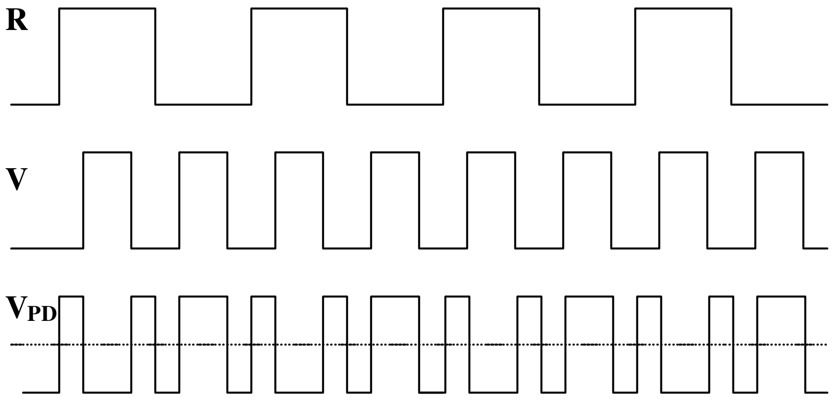


Figure 2.5 (a) XOR PD, the average over cycles is zero

In part (b), a PFD is used under the same situation. Although initially an *Up* is generated, over cycles, the *Down* is asserted more often than *Up*, and the average value used to control the VCO is decreased and the frequency of the VCO is decreased too to pull in the two different signals. When steady state is achieved, both *Up* and *Down* are logic LOW. Not only the phase, but also the frequency is locked when using the PFD. A charge-pump circuit is needed to convert *Up/Down* to a proper control voltage and will be discussed in Chapter 4. With finite delay associated with the reset path of the PFD (an AND gate and CLR-to-Q delay, referring to Table 2.1), a narrow pulse may be generated when changing the states (which is not shown in Figure 2.5 b). A static phase error may exist when a PFD is used.

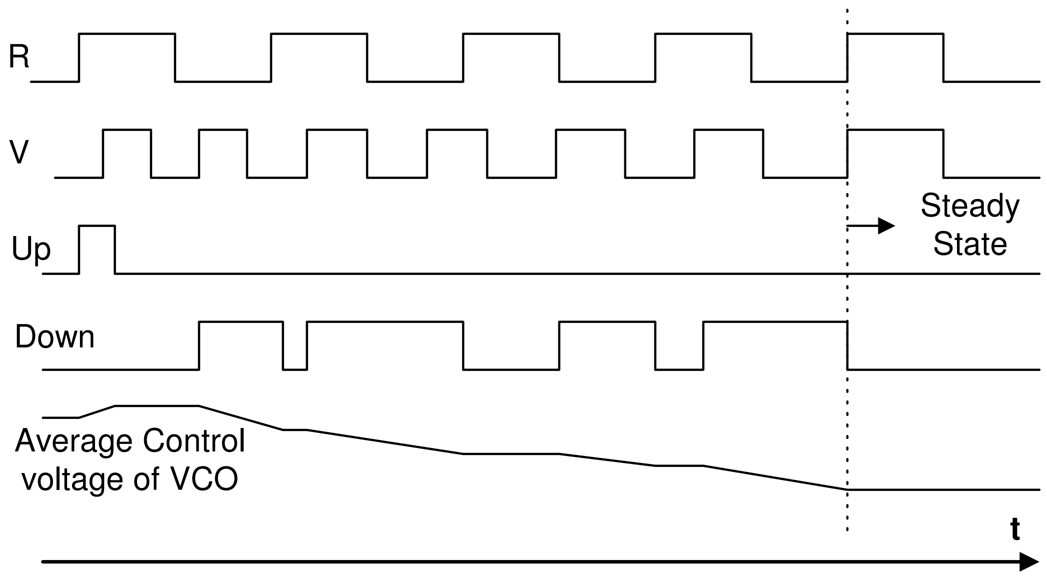


Figure 2.5 (b) PFD PD, a pull-in process is shown and steady state is achieved with both phase and frequency locked.

Another issue is false edges and long strings of 1s or 0s, common for NRZ (not returning-to-zero) data, which are transmitted in data communications to bear the binary information. Another data pattern is "return-to-zero", in which the signal goes to zero between consecutive data bits. Since the PFD is sensitive to the edges, it may not suitable for data / clock recovery.

A sequential phase detector, Hogge's PD [52], shown in Figure 2.6, can be used as self-correcting clock recovery circuit. "Self-correcting" means that the clock signal can align to the center of the data bit independent of the data rate, temperature, or process variations. A "no-action" state is implemented in the event of missing data edges. A re-timed data (*Q1*) is fed into the flip-flop XI2, which is clocked by the complement of the generated clock *V*. Assume clock *V* has a 50% duty cycle and frequency is twice the data rate. The outputs of XI1 and XI2, *Q1* and *Q2*, are half clock cycle delay from data input and *Q1* respectively. The average value of the difference between XOR outputs, *Up* and *Down* over cycles will determine the trend in oscillator frequency, increasing, decreasing or staying the same.

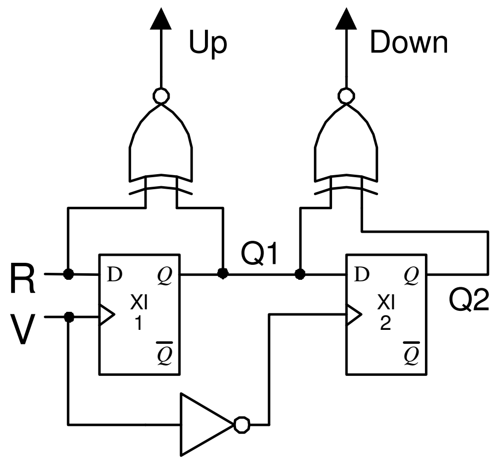


Figure 2.6. Hogge's self-correcting phase detector

The timing diagrams for a Hogge's detector are shown in Figure 2.7. When in lock, the average of *Up/Down* over several cycles is zero. When the clock (*V*) lags the incoming data (*R*), shown in Figure 2.8, *Up* is high more often than *Down* and the average voltage out of the loop filter will increase, as well as the frequency of the oscillator. Unlike the 3-state

PFD, where *Up/Down* are both low when in lock, the Hogge's PD will still generate equal duration *Up* and *Down* signals, which results in ripple after integration via the LF. Another problem is sensitive to the input data density. The charge deposited on the LF is not equal dependent on the different data density, more average charge when 1s and 0s are alternative than a long string of 1s or 0s (Figure 2.7). This will cause a pattern jitter and modulate the VCO output.

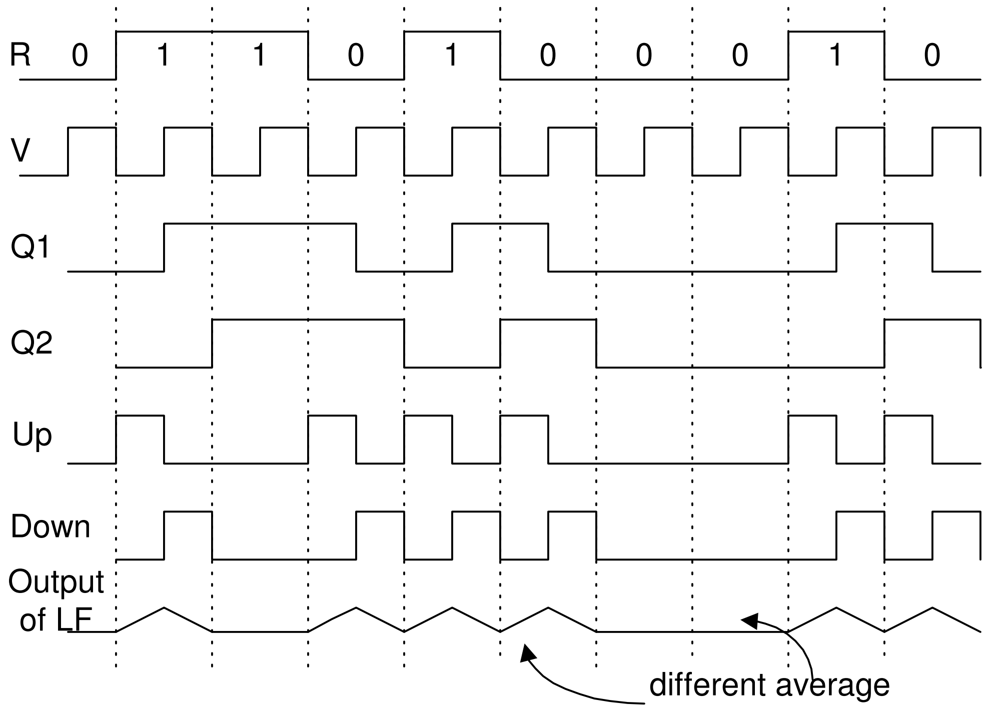


Figure 2.7. Hogge's PD for the clock recovery circuit in lock

A possible solution to the pattern jitter is to center the ripple, shown in Figure 2.9 [34]. The effect is that no net charge will be deposited into the loop filter when the loop is in lock regardless what the data density is. In order to suppress the ripple, a high order pole (HOP) is needed in the LF to effectively get rid of the high-frequency components. This pole should be placed far away from the low frequency poles and have little effect on the low-frequency transfer functions [4].

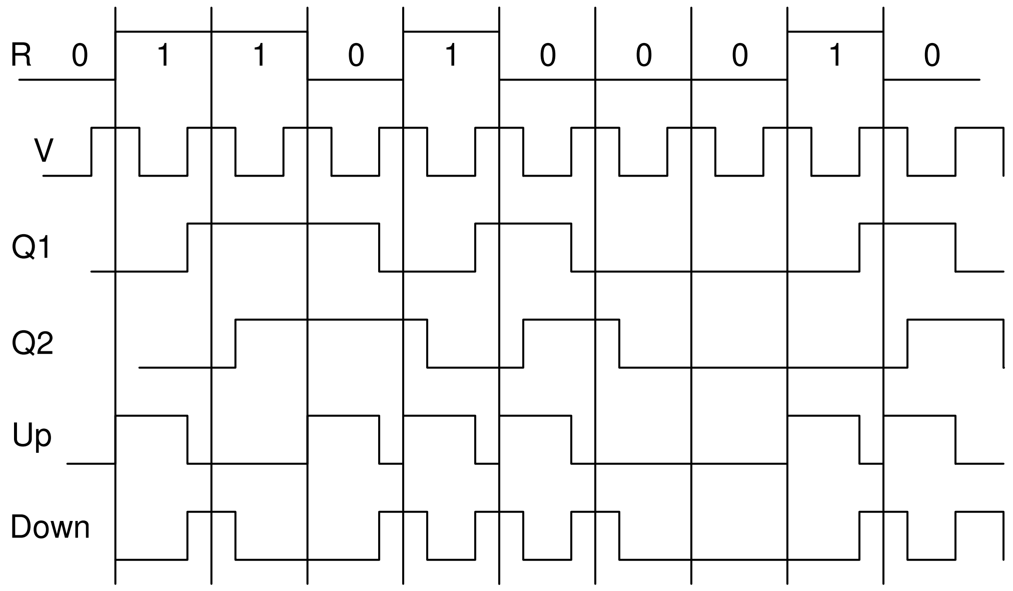


Figure 2.8. Waveforms when the clock (V) lags the center of the data (R)

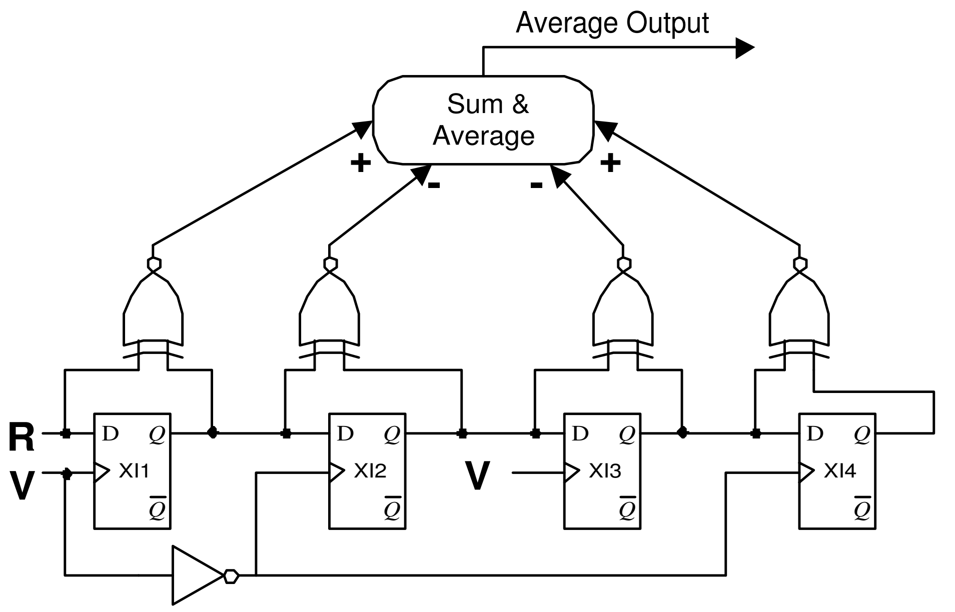


Figure 2.9. A sequential PD used for clock recovery without pattern noise

2.2.2. Loop Filter

After investigating different phase detectors, the averaging or integrating part, a loop filter, need to be addressed. A Loop Filter (LF) is normally a low pass filter, which performs the integration of the PD output over cycles. From the previous section, the pulse width and duty cycle of the PD outputs contain the phase difference information. A LF can convert these digital signals into a continuous average value to control the VCO frequency. In the earlier implementations, the passive elements of a LF were put off-chip to save silicon space. Effort has been made to implement the LF on chip, especially for high-speed applications. The LF is the brain of the PLL, and different topologies of LFs and its parameters can greatly affect the loop dynamics (which will be discussed in Chapter 3).

Generally, first order low-pass loop filters are used in most PLL design, and one pole is included in the transfer function *F(s)* of the LF. A simple RC network can do this job with a transfer function of

$$F(s) = \frac{1}{1 + s \cdot RC} \qquad (2.4)$$

In order to stabilize the operation, a zero is added in series with the capacitor (more on this in Section 2.3). Figure 2.10 shows two other typical loop filters: the passive lag filter (a) and the active PI (proportional + integral) filter (b) which include an op-amp. The transfer functions, F(s), are given by

$$F(s) = \frac{1 + s \cdot \tau_2}{1 + s \cdot (\tau_1 + \tau_2)} \qquad \text{… a passive lag filter} \qquad (2.5a)$$

$$F(s) = \frac{1 + s \cdot \tau_2}{s \cdot \tau_1} \qquad \text{… an active PI filter} \qquad (2.5b)$$

where τ_1 = R$_1$C, τ_2 = R$_2$C. For the active PI filter, slow changes cause the LF to do normal integration. For fast changes, the *s* factor will be cancelled in Equation (2.5b), and the output of the LF is proportional to the input with a gain factor equal to R$_2$/R$_1$. Another approach uses a charge-pump type loop filter which will be discussed in Chapter 4. Compared to the basic RC low pass filter, these types of LFs with stabilized zero suffer the degradation of high-frequency suppression, R$_2$/ (R$_1$+R$_2$) for passive lag LF or R$_2$/R$_1$ for active PI filter. In general, R$_1$ should be chosen at least 10 times larger than R$_2$ for better high frequency performance. A common solution is to add a small capacitor in parallel with serial C and R$_2$ to provide better suppression of high-frequency components. The drawback of this is making a higher order loop filter and all loop parameters need to be adjusted to prevent stability problems. When select the loop parameters, another consideration is the parasitics presented in the PLL, such as the output resistance of the PD, input resistance and capacitance of the VCO. The output resistance of the PD should be small compared to the R$_1$. The input resistance of VCO should be infinite and input capacitance is small compared to the LF.

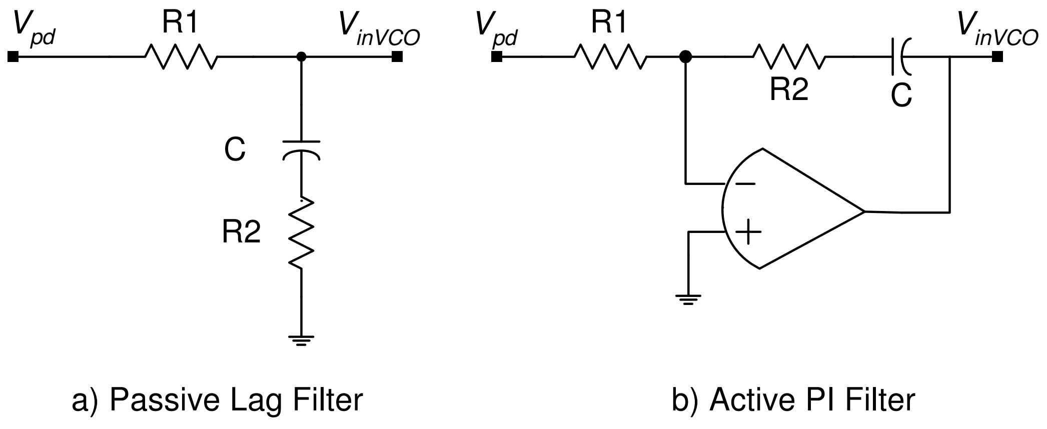


Figure 2.10. Typical first-order loop filters with a zero

Though the LF can be integrated with the PD, with the purpose to generate proper control signal for the oscillator, the important feature of LF and potential effects for the overall loop characteristics will be pointed out in the following sections.

2.2.3. Voltage-controlled oscillator

The last important building block of a basic PLL is the oscillator, which generates the required clock signal with a controlled frequency. If the signal out of the LF is a voltage, it is called voltage-controlled oscillator (VCO); a current-controlled oscillator (ICO) uses a current input to change its output frequency. If the frequency is a linear function of a control voltage V_{invco}, the VCO frequency (f_{out}) can be given by

$$f_{out} = f_0 + K_{vco} V_{invco} \tag{2.6}$$

where the f_0 is the free running frequency of the VCO, and K_{vco} is the gain factor. Figure 2.11 illustrates this relationship.

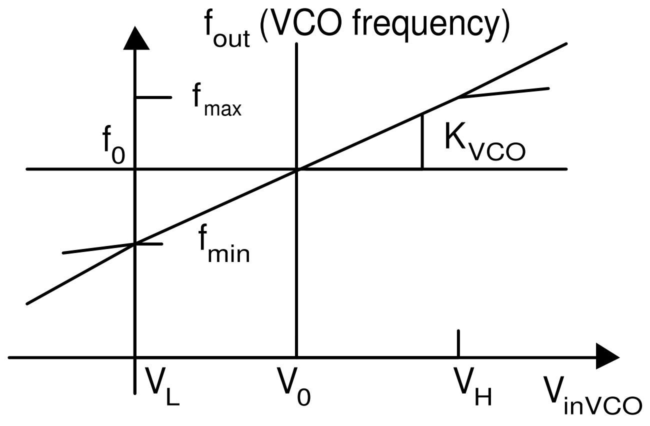


Figure 2.11. VCO characteristics: frequency versus control voltage

In Figure 2.11, f_{max} and f_{min} are the maximum and minimum frequencies that the VCO can achieve respectively. K_{VCO} is equal to the slope of the linear region. When the control voltage, V_{inVCO} is out of range, either greater than V_H or less than V_L, the curve is flatten out and the gain drops to zero. It results in slow loop response and possibly, large phase errors. The tuning range of the VCO, gain, and free running frequency (f_0) are important design issues based on different applications and architectures of PLLs. Since we are interested in the output phase rather than frequency in a PLL system, and phase is the integration of the frequency in time domain, the transfer function of VCO is given by

$$\frac{\phi_{PLL}}{V_{inVCO}} = \frac{K_{VCO}}{s} \qquad (2.7)$$

Unlike the PD, the VCO converts input control voltage (V_{inVCO}) back to phase (ϕ_{PLL}). This will introduce another pole in the system transfer function, and make the basic PLL a second order system.

For the monolithic implementation, a basic LC tank used in the conventional oscillator is not suitable for wide operating range. A common topology using CMOS technology is the ring oscillator, which uses an odd number of inversions to make oscillations. Adjusting the delay of each ring cell according to the control signal can tune the frequency of this ring oscillator when used in a PLL. If the delay of each delay stage is t_d and N delay stages are used, the oscillating frequency (f_{osc}) is equal to

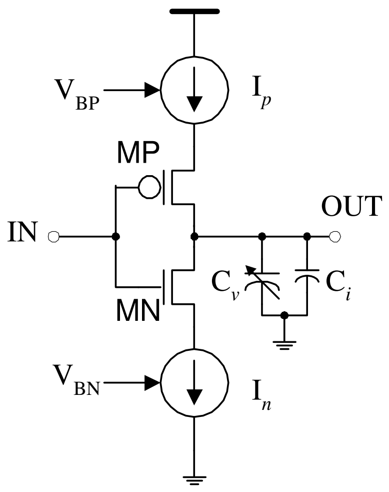
$$f_{osc} = \frac{1}{2 \cdot N \cdot t_d} \qquad (2.8)$$

where the factor of 2 in the denominator is due to both rising and falling edges of each delay stage contributing to the f_{osc}. From equation (2.8), two ways can be used to adjust the frequency:

- *Method 1*: the effective number of delay stages – *N*
- *Method 2*: the delay per stage – t_d

For the first tuning method, one important thing is to keep the odd number of stages for a single-ended delay stage. If a basic inverter is used as a delay stage, two inverters need to be removed or added each time when using this approach. In order to achieve wide lock range, a large number of stages are required. This approach is suitable for an all-digital implementation since changing the number of delay stages can easily be accomplished by using a shift register, where the shifting control signals are generated by all-digital phase detector (more on this in Chapter 6). Because of discrete tuning by adding or removing delay stages, delay interpolation is needed to improve the resolution and achieve tight locking.

Another way to tune the oscillator frequency is by adjusting the delay per stage. This method is widely used in the different types of VCOs. A simplified current-starved VCO delay stage is shown in Figure 2.12.



- V_{BP} and V_{BN} are bias voltages that control the current source and sink. The bias can be generated by V_{invco}.
- C_i is the intrinsic capacitance associated with the inverter MOSFET MP and MN (current stage and next stage).
- C_v can be a voltage-controlled capacitor.
- The delay of this stage can be evaluated by the time used to charge or discharge load capacitance (C_v + C_i) by I_p or I_n.

Figure 2.12. Single stage of a simplified current-starved VCO

Varying the current I_p and I_n, or C_v can change the delay, thus the oscillating frequency. Turning is continuous in this case and easily achieved wide lock range without much area. Some potential problems with this approach are:

1. When the current is small, in order to achieve wide lock range, the slope of transition is sluggish, which is susceptible to noise and may cause excessive jitter of the output.
2. For high-speed applications, the number of stages needs to be limited. For CMOS ring oscillator, at least three stages are needed. In order to provide enough gain for the oscillation, tradeoffs among gain, speed, and number of stages need to be considered.
3. The mismatching may cause $I_p \neq I_n$, and duty cycle distortion in the output (which is unacceptable for XOR PD).
4. It is supply and substrate noise sensitive.
5. The gain of VCO is not constant under the variations of process and temperature, which may cause stability problem in a PLL system.

In order to improve the performance of a VCO, a fully differential delay buffer, shown in Figure 2.13, can be used to reject common-mode noise. The voltage –controlled resistors can be realized by the P-type MOSFET, which is biased on the triode region. Another way to adjust the delay is by changing the tail current. Matched layout is very important to get identical delay per stage. Since the fully differential approach uses cross-point of the output and the complement of output, the signal swing doesn't need to be rail-to-rail, which may be suitable for low supply voltage applications. A level shifting circuit is needed to bring the voltage level back to the logic levels. Another benefit is negligible duty-cycle distortion. However, slow-transition problems are still related to this implementation.

When the input is out of range, the gain drops to zero and oscillations stop. In order to stabilize the operation under variations in process and temperature, self-biasing technologies are used, and discussed in Chapter 4. Noise issues with oscillators are investigated in Chapter 3.

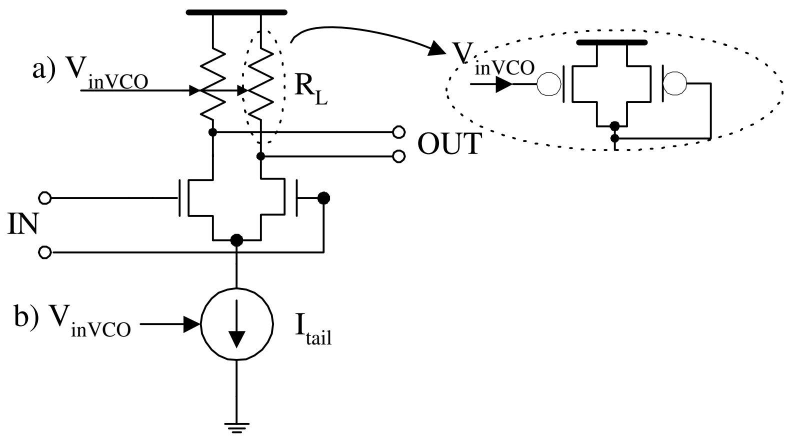


Figure 2.13. Fully differential delay buffer [15]

2.3. System Analysis of a Second-Order PLL

When we put these three building blocks together and apply a linear approximation for the basic PLL system, a simple system model can be derived, and is shown in Figure 2.14. If the loop gain is $K = K_{pd}\ K_{vco}$, the product of the PD gain and VCO gain, the transfer function of the basic PLL system is

$$H(s) = \frac{\phi_{pll}}{\phi_{ref}} = \frac{K \cdot F(s)}{s + K \cdot F(s)} \tag{2.9}$$

where the F(s) is the transfer function of the LF given in section 2.2.2. If the divide-by-N is added in the feedback path, then the loop gain is $K = K_{pd}\, K_{vco} / N$. Without the LF ($F(s) = K_L$, just a gain factor), the PLL is a 1st order system, which is a model for all-digital implementation. The high loop gain also results in a wider bandwidth, which may not be suitable for some applications. With the first-order loop filter, the order of a basic PLL system is two because of the pole in the VCO.

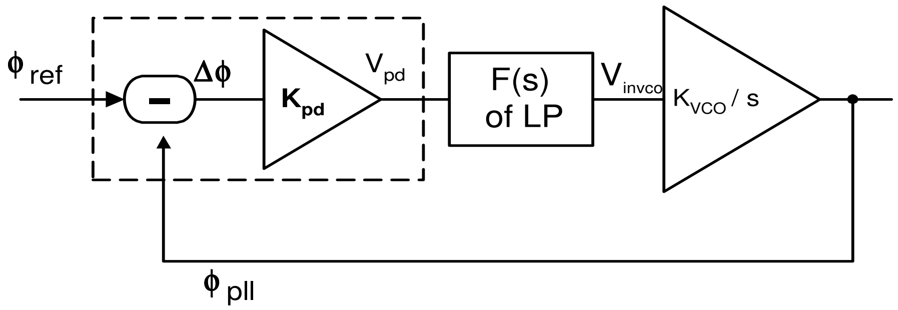


Figure 2.14. Linear model of second-order PLL system

With the consideration of damping factor ζ and natural frequency ω_n, the general form of second-order system is

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta \cdot \omega_n \cdot s + \omega_n^2} \tag{2.10}$$

Different topologies of the loop filter result in different ζ and ω_n, which is listed in the Table 2.3. From Table 2.3, we find that damping factor and natural frequency of the basic PLL system has a direct relationship with the loop gain (K) and LF parameters. For the basic RC filter, the ζ and ω_n have an inverse relationship with the loop gain. The added zero in the passive and active LFs will minimize this dependence and stabilize the loop operation. It's important to keep the loop gain constant to achieve ideal dynamic response of the PLL.

Another interesting point is the relationship between the static phase error ($\Delta\phi$) and input phase (ϕ_{ref}). From Figure 2.13 and Equation (2.9), it's easy to derive the phase error transfer function *$H_e(s)$* which is given by

$$H_e(s) = \frac{\Delta\phi}{\phi_{ref}} = \frac{s}{s + K \cdot F(s)} \tag{2.11}$$

In the steady state, the PLL is locked with its transient dead. The steady state condition can be expressed by the static phase error, $\phi_e(t \to \infty)$. Assume $\omega_{ref} = \omega_0$, at t <0 and a frequency step $\Delta\omega$ is applied at t = 0. The phase difference of the input is $\Delta\omega$ u(t), where u(t) is a unit step function. In the frequency domain, it can be expressed as $\Delta\omega/s^2$. Using the final value theorem, the static phase error is given as follows:

$$\phi_e(t \to \infty) = \lim_{s \to 0} s \cdot H_e(s) \cdot \frac{\Delta\omega}{s^2} \tag{2.12}$$

The static phase error for the step in frequency based on different loop filters is listed on Table 2.3. Type[1] C filter has 0 static phase error if the op-amp is idea, where higher loop gain can suppress the static phase error for Type A and Type B filters. However, higher loop gain of Type A LF will degrade the damping factor and make the PLL unstable. That's the reason why basic RC filter is rarely found in practical PLL implementations.

If $K >> \omega_n$ (high gain loop) for Type B and C filters, Equation (2.9) and (2.11) turn out to be

$$H(s) = \frac{\omega_n^2 + 2\zeta\omega_n \cdot s}{s^2 + 2\zeta \cdot \omega_n \cdot s + \omega_n^2} \tag{2.13}$$

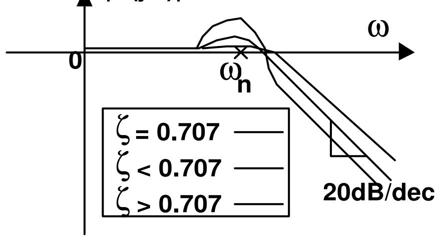
$$H_e(s) = \frac{s^2}{s^2 + 2\zeta \cdot \omega_n \cdot s + \omega_n^2} \tag{2.14}$$

[1] Here, the type of loop filter should not be confused with the type of PLL, which refers to the number of pure integrator (a pole at 0) included in the feedback system. A PLL with active PI filter is Type II.

	TYPE	NATURAL FREQUENCY	DAMPING FACTOR	STATIC PHASE ERROR
Basic RC network	A	$\sqrt{\frac{K}{RC}}$	$\frac{1}{2}\cdot\sqrt{\frac{1}{K\cdot RC}}$	$\frac{\Delta\omega}{K}$
Passive lag filter	B	$\sqrt{\frac{K}{\tau_1+\tau_2}}$	$\frac{1}{2}\cdot\sqrt{\frac{K}{\tau_1+\tau_2}}\cdot(\tau_2+\frac{1}{K})$	$\frac{\Delta\omega}{K}$
Active PI filter	C	$\sqrt{\frac{K}{\tau_1}}$	$\frac{\tau_2}{2}\cdot\sqrt{\frac{K}{\tau_1}}$	0 (with idea op-amp)

Table 2.3. Different loop parameters for the second-order system based on the various loop filters (Referring to Figure 2.10, $\tau_1 = R_1C$, $\tau_2 = R_2C$; $\Delta\omega$ refers to the input frequency step)

From equation (2.13), the phase transfer function of basic PLL system shows low-pass characteristics with bandwidth ω_n. The ζ determines the dynamic response of the second-order system, shown in the Figure 2.14 left. For ζ >1, the transfer function flattens out, and the dynamic response becomes sluggish (overdamped); for ζ <1, results in overshoot and oscillating. The optimal point may choose ζ = 0.707. On the other hand, Equation (2.14) shows that phase error has high-pass transfer function, shown in Figure 2.15 right plot. In the low frequency range, the phase error is attenuated to be small, where in high frequency range (>ω_n), large phase error can cause the PLL to lose tracking. A higher-order (> 2) system may be used to improve high-frequency attenuation, but this system may not be stable and may have some penalty in the settling time.



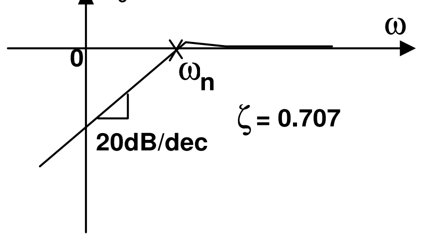


Figure 2.15. Magnitude Plots of the PLL Phase Transfer Function (left) and Error Transfer Function (right)

The steady state condition and linear approximation of a PLL system is the start point to understand a PLL operation. The transient response and loop dynamics will be investigated in Chapter 3. The challenges in design a PLL bring out a new candidate for clock synchronization: the delay-locked loop.

2.4. A new candidate -- Delay-locked loop (DLL)

A delay-locked loop, shown in Figure 2.16, uses a voltage-controlled delay line (VCDL) to replace the VCO used in a PLL. The basic operation of DLL is to delay the input reference signal (clock) through a delay line to get a zero phase shift at the output (one period delay according to the input reference). The frequency of the input and output signals are same. The big difference between DLL and PLL is that VCDL doesn't generate a signal. This limits a DLL from being used in frequency synthesis applications. Another difference is no separated reference signal is needed in a DLL.

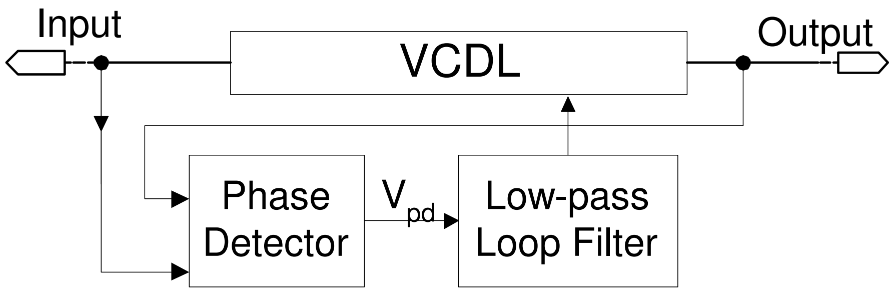


Figure 2.16. Block diagram of a DLL

A (N-1)-tap of VCDL is shown in Figure 2.17. Suppose the input frequency is 400MHz (T_{clk} = 2.5ns) and N=16. If the DLL is in the locked state (the rising edges of *CLKIn* and *CLKOut* occur at the same time), the VCDL can provide 16 different time-shifted taps of input clock signal, and each delay-per-tap is equal to 156ps. These equal spaced taps can be used to provide optimum clock strobe for different DQ outputs in high-speed memory interface. It's also easy to get the quadrature signal out of Tap 4, which is shifted 90 degrees from *CLKIn*. Each delay buffer used in VCDL can be the same as was discussed in section 2.2.3. If mismatches between taps are negligible, the tap delay is independent of device parameters even in the presence of temperature and process variations.

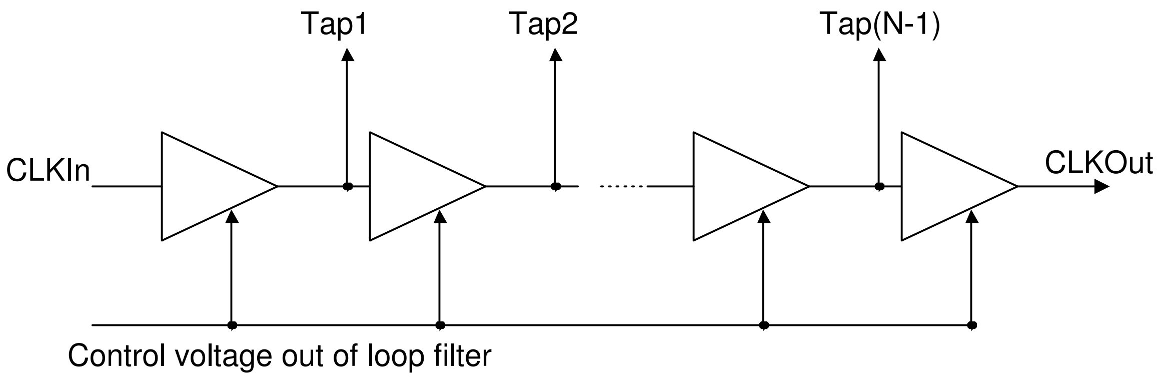


Figure 2.17. Block diagram of the voltage-controlled delay line (VCDL)

DLLs have much more relaxed tradeoffs among gain, bandwidth and stability since it can be designed as a first-order system [15] [37], a simple capacitor for the loop filter. Noise injected into a DLL disappears at the end of the delay line, eliminating the noise recirculating occurring in an oscillator. A false lock should be avoided when the maximum delay of VCDL equals to the twice of input clock period. Figure 2.18 shows the relationship between the *CLKIn* and the quadrature clock output (for the previous example, the 4th tap). If the VCDL electrical length is $2T_{CLK}$, the quadrature output is T_{CLK} / 2 from *CLKIn* instead of T_{CLK} / 4.

Another way to implement DLL uses all-digital techniques and it can be designed as a 0th-order system, where no integration is taken place in such a system. The delay is changed using method one, i.e. changing the number of delay taps. The big benefit of all digital DLLs is easy to scale for other processes and applications. The detail of this all digital implementation will be examined in Chapter 5.

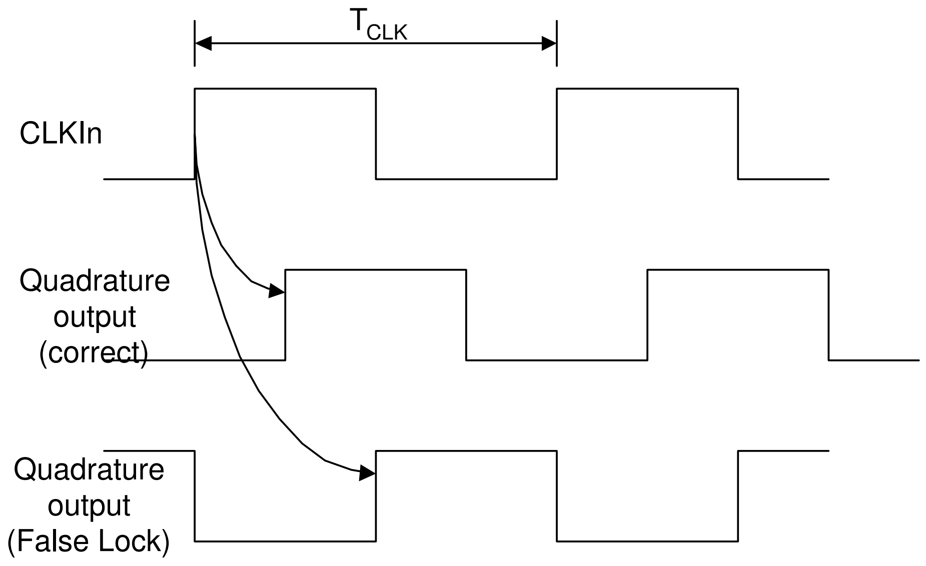


Figure 2.18. False lock of a DLL (input clock and quadrature output are shown)

Chapter 3

Loop Dynamics and Jitter

3.1. Tracking and Acquisition

Since the basic PLL system is not a linear system, it can be difficult to analyze its transient

different conditions:

1. ***tracking*** capability of the PLL system? That is, how does the PLL track the input variations while keeping the loop locked? The input variations may be small to validate a statically linear analysis, or large for dynamic tracking characteristics. This brings out two design parameters.
   - *Hold Range*, $\Delta\omega_H$, is the frequency range which the PLL can maintain phase tracking statically. Out of this range, the loop falls out of lock and loses tracking forever. In reality, the hold range may not be reached since other restrictions around the loop may take effects.
   - *Change Rate* of the input frequency, $d\Delta\omega/d$t, can not exceed some limit otherwise the loop will fall out of lock. This, $d\Delta\omega/d$t, defines the loop response of input frequency ramp.
2. If the loop is initially unlocked, how does the loop ***acquire*** lock? Two different processes exist during acquisition, *pull-in* and *lock-in*.

- *Pull-in* refers to the process of bringing a loop into lock after several cycles. The frequency range which the loop can finally acquire lock is defined as *Pull-in Range*, $\Delta\omega_P$. The pull-in process may be very slow and the time needed to pull in the loop is *Pull-in Time*, T_p.
- Compared to the slow pull-in process, if the initial frequency difference is small, the loop may lock up within one single-beat note between reference frequency and PLL output frequency. The frequency range of this quick *lock-in* process is defined as *Lock Range*, $\Delta\omega_L$, or *Capture Range*, and the time required to lock is *Lock Time*, T_L.

To achieve robust operation, different loop topologies are examined based on the tracking and acquisition characteristics. Some tradeoffs among the order of loop, loop gain, and bandwidth are discussed in detail. In order to achieve wide lock range, aided acquisition is used and discussed at the end of this section. The following analysis is based on the classic second-order PLL system [1].

3.1.1. Tracking

There are three different kinds of changes associated with the reference input of a PLL, phase step ($\Delta\Phi$), frequency step ($\Delta\omega$) and frequency ramp, illustrated in Figure 3.1. The frequency domain analysis using Laplace transform for different changes is also listed in Figure 3.1. For a small phase error, the linear model of a PLL discussed in Chapter 2 is valid and allows performing static analysis of the steady-state errors with difference input variations. Exploring the static phase error using Equation (2.11) and applying the final value theorem of Laplace transform, several results can be derived accordingly.

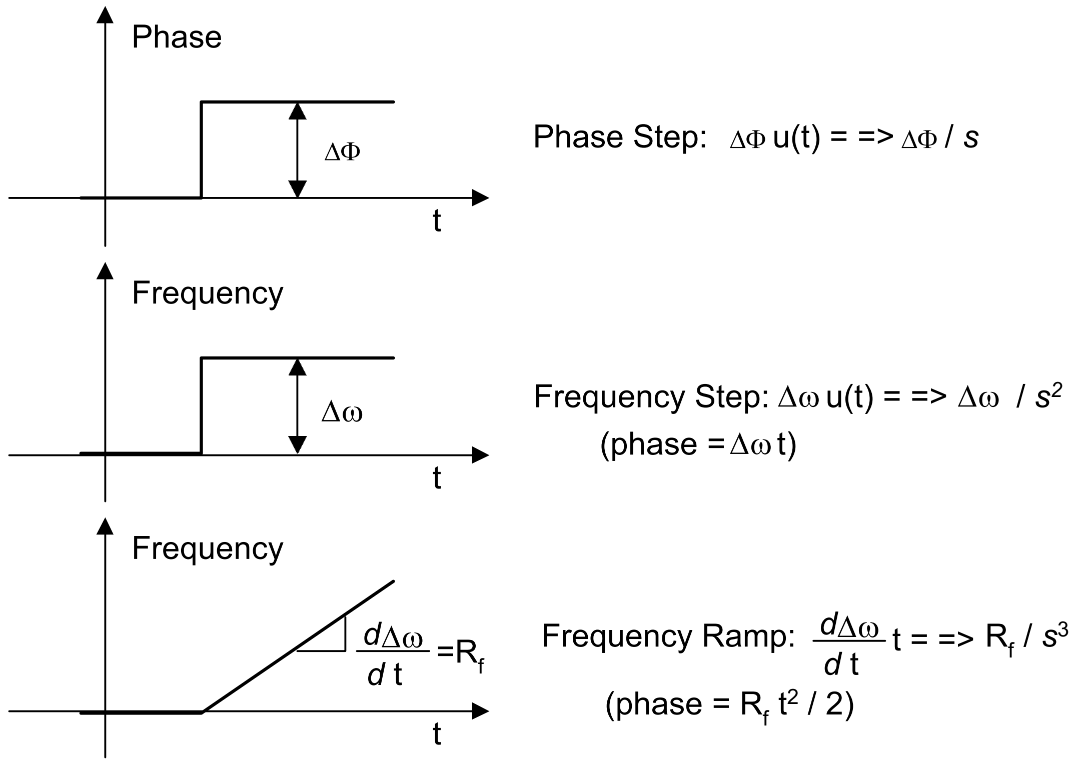

Phase
ΔΦ
t
Phase Step: ΔΦ u(t) = => ΔΦ / s
Frequency
Δω
t
Frequency Step: Δω u(t) = => Δω / s2
(phase = Δω t)
Frequency
dΔω / dt =Rf
t
Frequency Ramp: dΔω / dt t = => Rf / s3
(phase = Rf t2 / 2)

Figure 3.1. Three types of input variations: phase step, frequency step and frequency ramp with their time domain and frequency domain representations.

1. For a step in phase,

$$\phi_e(t \to \infty) = \lim_{s\to 0} s \cdot H_e(s) \cdot \frac{\Delta\Phi}{s} = 0 \tag{3.1}$$

   Regardless of different loop filters, the steady-state phase error is zero resulting from a step change of phase.

2. For a step in frequency[1], a general formula is

$$\phi_e(t \to \infty) = \lim_{s\to 0} s \cdot H_e(s) \cdot \frac{\Delta\omega}{s^2} = \lim_{s\to 0} \frac{\Delta\omega}{s + K \cdot F(s)} = \frac{\Delta\omega}{K \cdot F(0)} \tag{3.2}$$

[1] In the servo system, it's called a velocity error.

For Type A and B loop filters, $F(0) = 1$. For the active LF (Type C), ideally the $F(0) \to \infty$ and static phase error is equal to 0, which is consistent with Table 2.3. The integrator in the LF of a second-order loop builds a proper value to control the VCO while still permitting small phase error. By contrast, a first-order loop will need a large bandwidth to get high gain in order to suppress the static phase error. A large bandwidth will degrade the noise performance. It's the reason why the second-order PLL is predominant.

3. For the frequency ramp, assuming high gain loop and the slope of the frequency ramp is equal to R_f, which may vary with time.

$$\phi_e(t \to \infty) = \lim_{s \to 0} s \cdot H_e(s) \cdot \frac{R_f}{s^3} = \frac{R_f}{\omega_n^2} \tag{3.3}$$

   In a second-order system, a large natural frequency (ω_n) is required to handle a rapid changing input frequency. A higher order (e.g. 3) loop may have better tracking of the frequency ramp with small noise bandwidth [53].

4. The transient response of different input variations is listed in Gardner's [1] when inverse Laplace transforms of error transfer function are computed and evaluated for the time response.

5. When the phase error is large and the linear model is invalid, a nonlinear tracking occurs and the Hold Range can be derived based on Equation (3.2) and the gain factor (F_{PD}) listed in Table 3.1. For the active LF, because of the ideally infinite DC gain, $\Delta\omega_H \to \infty$ regardless of what class of PDs is used. Another exception is the PFD with charge-pump loop filter (discussed in Chapter 4). With the help of Figure 2.5 (b), this sequential PD acts like a real integrator to build up a proper control voltage for the VCO. Hence, the

hold range also approaches infinity without other limitations around the loop. When using the passive RC loop filter,

$$\Delta\omega_H = \pm K\,F_{PD} \tag{3.4}$$

Since the loop gain (K) can be made extremely large, the hold range may not be the real limit of input changes. The PLL operating range should be far less than the hold range in most cases.

6. According to Equation (3.3), the maximum permissible rate of change of input frequency is deduced based on different classes of phase detectors.

$$R_f = F_{PD} \cdot \omega_n^2 \tag{3.5}$$

where F_{PD} is a factor listed in Table 3.1. A rule of thump is chosen when the rate is less than ω_n^2.

	MULTIPLIER	XOR	RS-LATCH	PFD
Class	I	II	III	IV
Gain Factor (F_{PD})	1	$\pi/2$	π	2π

Table 3.1. A gain factor according to different kinds of phase detectors

The nonlinear tracking behavior is analyzed in [1] with some other limits. A pull-out range, $\Delta\omega_{PO}$, is defined as the dynamic limit for stable operation of a PLL. The value of $\Delta\omega_{PO}$ is related to the natural frequency and damping factor and mainly determined by computer simulation [2]. In the pull-out range, a PLL will lose phase tracking, but may reacquire lock via pull-in or lock-in process. Generally, $\Delta\omega_{PO} < \Delta\omega_H$. If the loop is initially unlocked, which is common when the VCO free-running frequency is not the same as input frequency, an acquisition process is started to bring the loop back into lock.

3.1.2. Acquisition

Acquisition is a nonlinear process. If the frequency difference is small and the loop can acquire lock without cycle slips, it can be considered as phase acquisition where a PLL locks up with just a phase transient. A more difficult analysis is the frequency acquisition, where the reference frequency is initially far away from the VCO free-running frequency (the limit is the pull-in range). For a typical second-order PLL, this pull-in process is fairly slow compared to the lock-in process.

I. Lock-in

The lock-in process can be illustrated using Figure 3.2. A simple Class I PD is used and the output of the PD is sinusoidal. Similar analysis can be applied for other class of PDs. Assume the initial frequency difference is $\Delta\omega$. After the loop filter, only the low frequency terms remain and can be expressed as

$$V_{inVCO} = K_{PD} \cdot \left| F(j \cdot \Delta\omega) \right| \cdot \sin(\Delta\omega \cdot t) \tag{3.6}$$

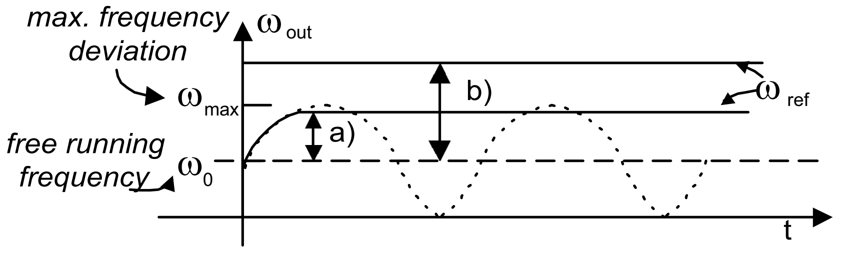
This time-variant control signal will modulate the VCO frequency, as shown in Figure 3.2, assuming the reference frequency $\omega_{ref} = \omega_0 + \Delta\omega$. ω_0 is the free running frequency of the VCO. The peak frequency deviation is equal to

$$\omega_{\Delta\max} = K \cdot \left| F(j \cdot \Delta\omega) \right| \cdot F_{PD} \tag{3.7}$$

where K is the loop gain and F_{PD} is the factor dependent on different PDs. For Class I PD, $F_{PD} = 1$. If $\Delta\omega < \omega_{\Delta\max}$, then the PLL can achieve lock within the first proper peak of the beat signal, shown in Figure 3.2 (a). This maximum range is defined as the *Lock Range*, $\Delta\omega_L$. If we make an assumption that the lock range is much greater than the corner frequencies of the loop filter, the magnitude of $F(j\Delta\omega_L)$ is approximately equal to τ_2 / τ_1. The relationship between ζ, ω_n and $\Delta\omega_L$ can be established using equations in Table 2.3, assuming high-gain loops.

$$\Delta\omega_L \approx K \cdot F_{PD} \cdot \frac{\tau_2}{\tau_1} \approx 2 \cdot \zeta \cdot \omega_n \cdot F_{PD} \tag{3.8}$$

Compared to the *Hold Range*, $\Delta\omega_H$, the lock range $\Delta\omega_L$ is approximately τ_2 / τ_1 of the hold range for the second-order loop. For the first-order loop, there is no distinguishing between the hold range and lock range. Another observation is that the PFD can achieve widest lock range and is 2π times the lock range of the analog PD.



a) Frequency offset within the lock range -- lock-in process
b) Frequency offset out of the lock range -- pull-in process

Figure 3.2. Processes of acquisition with the different frequency offset ($\Delta\omega$)

The other design consideration is how long the loop can acquire lock. Within the lock range, for a damped oscillation (ζ<1), the lock-in time can be computed as

$$T_L = \frac{2 \cdot \pi}{\omega_n} \tag{3.9}$$

when the transient is considered to die out after one cycle. It's highly desired to limit the input frequency step within the lock range, where a fast acquisition can be achieved. Wide bandwidth can help to achieve wide lock range and fast lock, but the noise performance may be degraded. Equation (3.9) provides an initial guess of the settling time of the PLL, which is important in the power management of a microprocessor or memory system. In practice, the linear model of the PLL may not be satisfied because of the variation of *K*. Simulations are needed to predict the more accurate lock time.

II. Pull-in

When $\Delta\omega$ is greater than $\Delta\omega_L$, a pull-in process is launched to gradually pull the frequency back to the lock range, and the lock-in process takes place to finally acquire lock. The limit which pull-in process can occur is defined as pull-in range, $\Delta\omega_P$. In Figure 3.2 (b), the initial frequency offset is out of the lock range, but the modulation of the output frequency of the PLL is not symmetrical, with longer positive half cycle. It will cause ω_{out} slightly increasing toward ω_{ref}. After several cycle slips, the pull-in process keeps going until $\Delta\omega$ falls into the lock range. Figure 3.3 demonstrates this pull-in process. This process is nonlinear and mathematical analysis is complicated. The derivations for the pull-in range and time can be found in [2] and the results are shown here. For the active PI filter, or Class IV PD, a pure integrator in the loop filter can provide ideally infinite DC gain and acquire lock eventually, which means that $\Delta\omega_P \to \infty$.

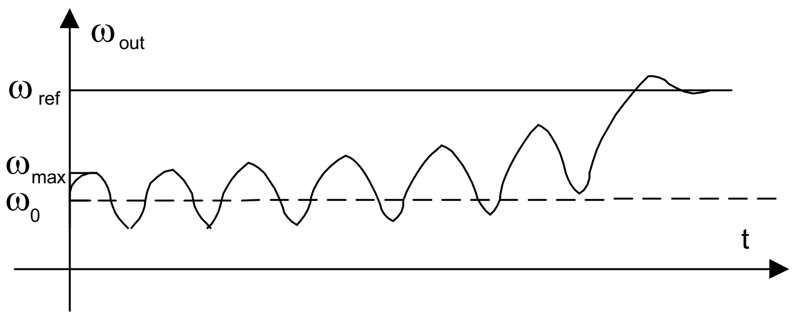


Figure 3.3. Pull-in process

For the passive LF, assuming high-gain loop, the pull-in range is given by

$$\Delta\omega_P = F_{PD} \cdot \sqrt{2\zeta\omega_n K} \tag{3.10}$$

where F_{PD} is the gain factor listed in Table 3.1 excluding the PFD. To increase the pull-in range, high loop gain is desired whereas the loop bandwidth can still be small. If the frequency offset ($\Delta\omega$) is less than 80% of the pull-in range, the pull-in time can be approximated as (excluding the PFD):

$$T_P \approx \frac{1}{F_{PD}^2}\frac{\Delta\omega^2}{\zeta\omega_n^3} \tag{3.11}$$

If $\Delta\omega$ approaches $\Delta\omega_P$, the pull-in time goes infinitely. Narrow bandwidth has large penalty on the pull-in time, which is inversely proportional to the cube of loop bandwidth. For the sequential phase detector, the pull-in time is equal to the time required to charge the loop capacitor to proper value ($\Delta\omega$ / K_{VCO}), as shown in Figure 2.5 (b). Since the sequential PD has memory, the initial conditions of the loop, as well as the initial phase of the VCO will affect the pull-in time.

In general, the frequency acquisition is slow and unreliable for a second-order PLL. Large bandwidth is needed to increase the acquisition range and decrease the acquisition time. In practical, noise needs to be considered and small loop bandwidth is desired to have better jitter performance (more in next section), tradeoffs about gain, bandwidth, lock range and time need to be concerned. The relationship among hold range, lock range, pull-out range and pull-in range is

$$\Delta\omega_L < \Delta\omega_{PO} < \Delta\omega_P < \Delta\omega_H \tag{3.12}$$

For the delay-locked loop, it can be designed as a first-order system, and only phase acquisition is needed. From previous discussion, the phase variation can always be corrected statically and dynamically. The DLL design is more concerned about how to improve jitter performances while still achieve wide lock range.

3.1.3. Aided Acquisition – Two-loop Architecture

In order to effectively increase the capture range, most practical PLLs employ additional techniques to aid the frequency acquisition. A sequential phase detector can be used to achieve both phase and frequency detection simultaneously. Another popular aided acquisition uses a two-loop architecture, a frequency-locked loop (FLL) to pull in the frequency and a phase-locked loop to lock the phase, as shown in Figure 3.4. If the frequency difference is large, the DC value generated by the PD loop is negligible and the VCO is driven by the FLL. The FLL can be designed as a first-order loop to achieve wide bandwidth and quick pull-in if the input signal-to-noise ratio is permitted [1]. When the frequency difference is zero, the FLL declares lock and the phase-locked loop is dominated to finally acquire phase lock.

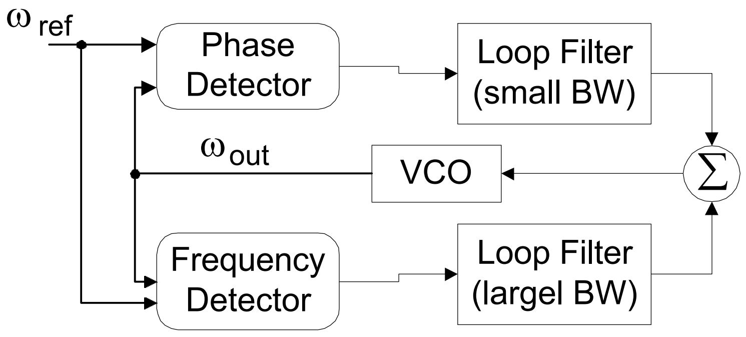


Figure 3.4. Two-loop architecture for the aided acquisition

For the delay-locked loop, no frequency detection is needed since the DLL only adjusts phase. Similar two-loop architecture can be used to improve the resolution of delay line and achieve better jitter performance [12], [18], [27], [28]. It is also called a phase or delay interpolation.

Another limitation of acquisition is from the VCO. From Figure 2.11, if the operating frequency is out of the VCO linear range, the gain drops to zero and the VCO can cease to oscillate. This is the physical limitation and the acquisition range should be within this range. With process and temperature variations, the linear range of VCO may vary substantially, thereby requiring a wider acquisition range even if the input frequency is tightly controlled.

3.2. Phase Noise and Jitter

Jitter can be considered as time variant of the clock period. When a clock/data signal travels through a non-ideal channel and corrupted with noise, there are some uncertainties about the clock/data edges, which move in time, shown in Figure 3.5. With large noise, the data eye may close and make data/clock recovery extremely difficult. If the generated clock is jittering, it may not be placed on the center of data eye and make a wrong decision. Such random variation cannot be recovered by simple amplification or clipping. A PLL circuit can be used to efficiently recovery or regenerate the clock/data with low jitter. In the frequency domain, such timing jitter is called phase noise.

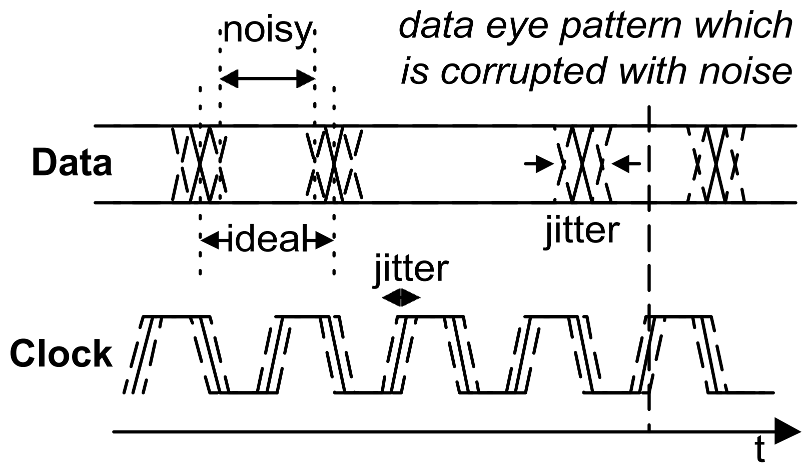


Figure 3.5. Timing jitter related to data and clock.

Phase noise and jitter are a very important issue when design a phase-locked and delay-locked loops. Different applications may have different emphasis on the jitter specifications. "Cycle-to-cycle" jitter refers to the time difference between two consecutive cycles of a period signal. A RMS (root mean square) or peak-to-peak value is used to describe a random jitter. According to the noise sources, it can be classified as internal jitters, caused by the building blocks of PLLs and DLLs, and external jitters. Jitters in an oscillator have been examined for almost half a century and still a hot topic. Some latest thoughts about the CMOS implementation, deep sub-micron effects, and fully differential approach will be discussed in this section.

3.2.1. Noisy Input – External Jitter

Receiving data corrupted by noise is very common in modern communication system because of the distortion, inter-symbol interference (ISI) etc. In order to conserve the channel bandwidth, no separate clock signal is transmitted along with the data and not-return-to-zero (NRZ) data pattern is used because of its good bandwidth efficiency [54]. A clock recovery circuit is needed to recovery the clock signal from the noise buried data. A phase-locked loop is well suited for this application because of its ability to cope with large amounts of noise. A delay-locked loop may not be accommodated since any input jitter will also appear on the output.

I. Noise Bandwidth

Analysis the input phase noise is straightforward. Assume the input noise and internal noise are not correlated. Superposition can be used to analyze the effect of input phase noise. From the discussion in Chapter 2, the basic second-order PLL system can be treated as a low-pass filter with the transfer function of Equation (2.13), shown in Figure 2.14 left. If input phase noise is white and time invariant, the output noise is sharpened for a narrow band PLL. The noise bandwidth (B_L) can be defined as

$$B_L = \int_0^\infty \left|H(j2\pi f)\right|^2 df \tag{3.13}$$

where $H(j2\pi f)$ is the transfer function of the close loops [1]. To suppress the input noise, a loop with narrow bandwidth is desired for better noise shaping. Higher order loop is also helpful to provide a sharp cutoff. These requirements to improve jitter performance caused by input noise are contradict to ones for loop dynamics, which require wide bandwidth to achieve wide lock range and fast lock, and lower order to avoid stability problems.

II. Jitter Tolerance

The ability that a PLL system can track the input jitter is called Jitter Tolerance. The phase error transfer function in Equation (2.14) and Figure 2.14 right shows high-pass characteristic, which means that in some frequency range, the loop cannot track the input noise and pass the input jitter with litter suppression. For clock and data recovery applications, the jitter tolerance range should be wide to accommodate lots of jitter while not degrading the performance. On the other hand, narrow range is required to get better signal-to-noise performance of the output. For a long distance communication, a string of repeaters is needed to rebuild the corrupted data with recovered clock synchronized on the center of the data eyes. Narrow bandwidth PLLs are required to avoid jitter accumulation from each repeater.

III. Jitter Peaking

As seen from Chapter 2, a zero is placed in the loop filter to provide stability, otherwise, damping factor and loop gain are dependent and the loop is unstable if large gain is exploited to provide better tracking. If the active PI filter is used and Equation (2.13) is rewritten as

$$H(s) = \frac{\frac{K}{\tau_1}(1+\tau_2 s)}{s^2 + \frac{K}{\tau_1}\tau_2 s + \frac{K}{\tau_1}} \tag{3.14}$$

where a zero is located around $1/\tau_2$. The denominator of Equation (3.14) gives two roots as closed loop poles, P_{low} and P_{high}.

$$P_{low} = \frac{K\tau_2}{\tau_1}(-1+\sqrt{1-\frac{4\tau_1}{K\tau_2^2}}) \tag{3.15}$$

$$P_{high} = \frac{K\tau_2}{\tau_1}(-1-\sqrt{1-\frac{4\tau_1}{K\tau_2^2}}) \tag{3.16}$$

When the zero in Equation (3.14) occurs at a lower frequency than the first closed loop pole (P_{low}), a gain greater than 1 (> 0dB) occurs and is called *jitter peaking*. If the jitter is fallen in this range, the jitter will be amplified and cause problems. A simple Bode plot of this jitter peaking is shown in Figure 3.6, which is same as Figure 2.14 left.

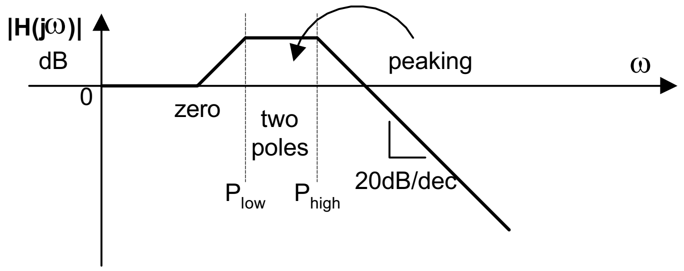


Figure 3.6. Jitter peaking for a typical second-order PLL

One way to minimize jitter peaking is to put the zero and the low frequency pole as closely as possible. This requires overdamping the loop (choosing large damping factor ζ) and results in slow acquisition process and a large capacitor value in the loop filter. Higher order loop can be used to offer more design parameters and place the zero after the first pole to eliminate the peaking. First-order PLLs and DLLs have no peaking in their transfer curves but have to tradeoff their gain with bandwidth. A good way to get rid of the jitter peaking was reported in [34], which using a DLL instead of stabilizing zero for a clock recovery circuit.

If the input jitter is negligible for clock synchronization or frequency synthesis applications (the reference comes from crystal oscillator), internal jitter, especially jitter in the oscillator is dominant for the overall jitter performance. For a DLL, input jitters will also appear at the output with similar low-pass characteristics. Generally, DLLs are not suitable for clock recovery applications because no signal can be generated in DLLs.

3.2.2. Noise in the Oscillator – Internal Jitter

An oscillator can be treated as a frequency selector which has a band-pass characteristic, passing the selected frequency and filtering out other components. A feedback feature is embedded in an oscillator when the close loop gain is greater than 1 and the fed back signal has a 180° phase shift. Any jitter in one transition of the oscillator will be fed back to the input and affects all the following transitions. This is called *Jitter Accumulation*. In a PLL, jitter in the oscillator is dominant in the internal noise.

In a PLL, an oscillator is designed to track small disturbances which come from the phase detector and loop filter. Intuitively, a large bandwidth is desired to track the disturbances, as described in the jitter tolerance criteria. From the frequency acquisition point of view, a PLL with wider bandwidth can correct the timing error more quickly, referring to Equation (3.9), (3.11). This results in a smaller overall jitter, which is in direct contradiction to the narrow bandwidth requirement for the input jitter suppression. This is an application-oriented decision, a narrow band for input noise critical designs and wide bandwidth for better frequency selection and fast acquisition in the internal noise-sensitive applications.

To examine jitter in a CMOS ring oscillator, several effects need to be focused:

- Intrinsic delay per stage
- Supply and substrate noise
- Thermal and Flicker noise
- Number of delay stages
- Process scaling

- Power consumption
- Oscillating frequency
- Symmetrical edges of transition

I. Noise in a Delay Cell

As seen from Chapter 2, a CMOS ring oscillator is made up with N identical delay stages, and each delay stage contributes delay of $2t_d$ for the oscillator period. The delay can be tuned to achieve different oscillating frequency. When considering timing jitter, we can treat each stage of the ring oscillator as an independent noise source, causing timing error Δt_r for the rising edge and Δt_f for the falling edge, shown in Figure 3.7. The RMS time error (standard deviation) is denoted as $\sigma_{\Delta t}$ for each delay cell. C_n is the node capacitance including output capacitance of current delay stage and input capacitance of next delay stage. The intrinsic delay (t_d) is the time required to charge or discharge C_n by the tail current. Two observations can be drawn from Figure 3.7.

- Timing jitter is sensitive to the transition times (or slope).
- Asymmetrical transitions may have different Δt_r and Δt_f.

Intuitively, large number of delay stages result in sharp transitions and less sensitive to the noise. A mathematical derivation in [40] shows that the sensitivity factor is proportional to $1/N^{1.5}$. As will be seen, it cannot conclude that a ring oscillator with large N has a better jitter performance since the number of noise sources also increases for a given oscillating frequency and power dissipation.

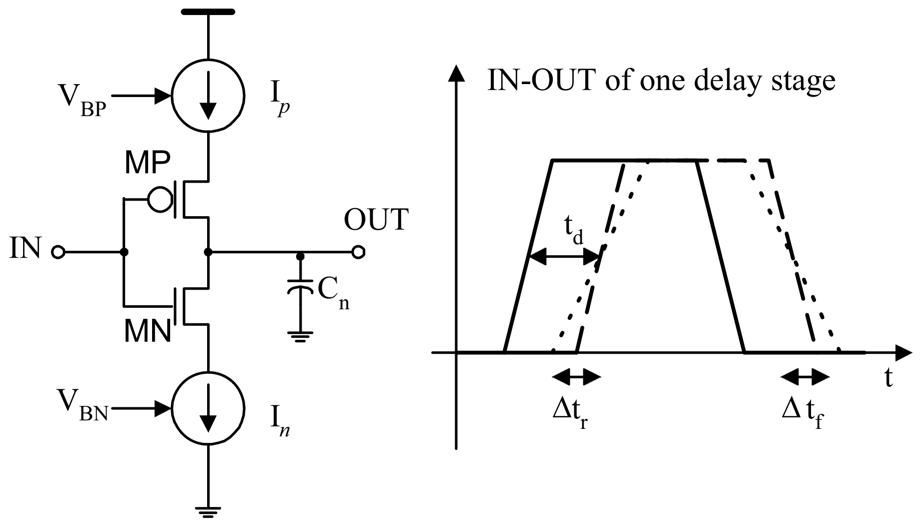


Figure 3.7. Timing error of one delay stage

Noise in an oscillator can be classified as independent noise sources such as thermal noise with each device used in a delay cell, and correlated noise sources, such as substrate and supply noise, low frequency noise (1/*f* noise). For the intrinsic jitter per delay stage (thermal noise), the overall timing error $\sigma_{\mathrm{N},\Delta t}$ is given by

$$\sigma_{\mathrm{N},\Delta t} = \sqrt{2N} \cdot \sigma_{\Delta t} \tag{3.17}$$

For a single-ended delay stage, the ratio of the overall timing error to the period ($T_o = 2Nt_d$) is analyzed in [40] and the results are

$$\frac{\sigma_{N,\Delta t}}{T_o} = \alpha \cdot \sqrt{\frac{kT}{P} \cdot \frac{V_{DD}}{V_c}} \tag{3.18}$$

$$P \approx 2NV_{DD}q_{\max} f_o \tag{3.19}$$

where α is a factor, V_c is the characteristic voltage of the device, P is the total power dissipation, V_{DD} is the supply voltage, and f_o is the oscillating frequency. For long channel devices, V_c is the gate overdrive voltage (V_{GS}-V_T). For short channel devices, V_c is smaller and jitter performance is degraded. Similar analysis for fully differential delay stages is shown in [40]. The results are

$$\frac{\sigma_{N,\Delta t}}{T_o} = \alpha \cdot \sqrt{N \cdot \frac{kT}{P} \cdot \left(\frac{V_{DD}}{V_c} + \frac{V_{DD}}{R_L \cdot I_{tail}}\right)} \tag{3.20}$$

$$P = N I_{tail} V_{DD} \tag{3.21}$$

Refer to Figure 2.12, R_L is the resistive load, I_{tail} is the tail current. Another approach for the CMOS differential oscillator can be found in [39] that gives result as

$$\frac{\sigma_{N,\Delta t}}{T_o} = \alpha \cdot \sqrt{\frac{kT}{I_{tail}} \cdot \frac{A_v}{V_c} \cdot f_o} \tag{3.22}$$

where A_v is the small signal gain for one delay stage. Some important design implications can be drawn from Equation (3.18) to (3.22) for low jitter.

1. The gate overdrive should be chosen as large as possible for both single-ended and differential implementation. Low threshold voltage (V_T) reduces phase noise.
2. Phase noise degrades with scaling because of V_c (not obvious for the first order), which is a function of velocity saturation (critical electric field) and channel length [40]. As the channel length is scaled down, V_c is smaller and poorer jitter behavior. If the speed is not critical, a delay stage with longer channel length has better noise performance.
3. Jitter is inversely proportional to the power dissipation. Increasing the tail current (I_{tail}) of the differential delay stage improves noise performance, which is not a direct function of power supply voltage.
4. From Equation (3.22), the jitter improves with lower gain per stage. Any noise, either internal or external, will be amplified if the gain is large. There is also a down limit of the gain to maintain oscillating.

5. If the power dissipation (P) is fixed, as well as f_o and N, the phase noise of the differential oscillator is about $N\,[1 + V_c\,/\,(R_L\,I_{tail})]$ times larger than that of a single-ended oscillator. It can be understood that a single-ended structure is simple and has less noise sources than the differential architecture. If the common-mode noise, such as supply and substrate noise, can be controlled and limited, the single-ended delay stage, even a two-transistor inverter is preferred to achieve low jitter.
6. If P is fixed, the number of stage (N) is independent for the single-ended oscillator but will degrade the jitter performance of the differential oscillator for a large N. On the other hand, if oscillating frequency is fixed, as well as I_{tail} and q_{max}, the N is factored out for differential implementation but related to the single-ended approach. When choosing the *optimum* number of stages, power dissipation, oscillating frequency, and charge swing need to be considered.
7. Generally, higher oscillating frequency degrades the jitter performance of a ring oscillator. For GHz frequency range, an inductor along with a capacitor may need to provide better noise performance.
8. Higher temperature introduces more jitter in a ring oscillator. It's possible to cancel or reduce the temperature effects with a carefully designed bias or tail current.

II. Symmetrical Transitions

Symmetrical transition means equal rising and falling times for the oscillating signal. It will affect the low frequency noise or $1\,/\,f$ noise [40] [41]. A large number of stages is helpful to reduce large $1\,/\,f$ noise. Nonlinear loads for the differential stage will degrade the noise performance (noise coupling from power supply). In order to reject the common-mode noise from power supply and substrate, fully differential delay buffer is preferred in the IC implementation. A resistive load has better linearity. For the IC implementation, a voltage-controlled resistor that includes a diode-connected MOSFET is shown to have better linearity [8] [15]. The tail current source will also introduce a switching noise if the differential buffer is not well balanced. Matched each half circuit of the differential buffer is required to minimize the $1 / f$ noise. Frequency-domain analysis of phase noise in an oscillator can be found in [41], [42].

The above design implications are also suitable for a low-jitter DLL design when the tuning method is to adjust the delay per tap. When all-digital implementation is used and delay is tuned by changing *N*, a Up-Down counter or shift-register is needed to filter out the noise and the jitter is determined by the resolution of the delay element. Delay interpolation can greatly improve the jitter performance of an all-digital DLL.

IV. Jitters from Other Building Blocks

Besides the oscillator, the other building blocks of the PLL will also contribute to the overall phase noise if careful design is not applied. From the discussion in Chapter 2, different phase detectors have different characteristics and may cause jitter which modulates the VCO frequency. For the clock/data recovery applications, the recovered clock is usually aligned with the center of the data bits, which results in 90° out of phase when locked. With finite bandwidth of the loop filter, a ripple whose frequency is equal to the clock frequency is always presented on the control signal of the VCO. For the PFD along with the charge-pumped loop filter, ideally, there is no net charge transition when the loop is in lock (the PFD is in State *O* in Figure 2.4.). Device mismatch, finite switch time, and supply / substrate noise injection will cause a static phase error for this type of PD. A high order pole is needed to effectively suppress the high frequency components and minimize the ripples. This will bring the second-order PLL to a higher order system and may cause stability problems. Another possibility is data pattern jitters, discussed in Chapter 2. Different data density may cause the control signal out of the LF to drift and have different average value over several cycles. Modified PD architectures can be used to get rid of these pattern jitters.

If the PLL is used for frequency synthesis, a divide-by-N is needed to provide frequency multiplication in the oscillator output. As seen from Chapter 2, the loop gain is degraded by factor of *N*. This will slow down the loop acquisition and amplify the input jitter.

For a monolithic implementation, an all-digital phase detector along with charge-pump loop filter is popular for an *analog* PLL design. However, all digital implementation has features of low-cost, simplicity, portable and scalable for different processes. The design criteria for charge-pump PLL and all digital PLL are examined in the next chapter.

Chapter 4

Charge-Pump and All-Digital Loops

4.1. Charge-Pump Phase-Locked Loops

From the previous discussion, the phase-frequency detector (PFD) has supreme features over other types of phase detectors, especially for its wide acquisition range (both phase and frequency detection) and fast lock. Two outputs, *Up* and *Down*, and three possible states of the PFD require a special circuit to convert the output digital signal to proper analog control value of the VCO and generate three states, increasing, decreasing and no change for the VCO frequency. The "*no change*" state is distinguished from other type of LFs and provides the charge-pump LF a unique feature: a real integrator.

A simple tri-state loop filter can do this job and is shown in Figure 4.1. Two MOSFETs (*MP* and *MN*) are used as switches. A passive loop filter is used to generate the proper average voltage to control the VCO. When both *Up* and *Down* are LOW, the tri-state LF is in the high impedance and V_{inVCO} stays same. One big problem with tri-state loop filter is nonlinear gain if the V_{inVCO} is far away from $V_{DD}/2$ [4]. The asymmetrical charge or discharge voltage will cause the gain to vary which is intolerable for most PLLs. Another problem is sensitive to the supply and substrate noise. An active filter can also be used in the tri-state configuration with some limitations. With the concern of charge-pump circuit, an active LF has no extra benefits over a simple passive LF.

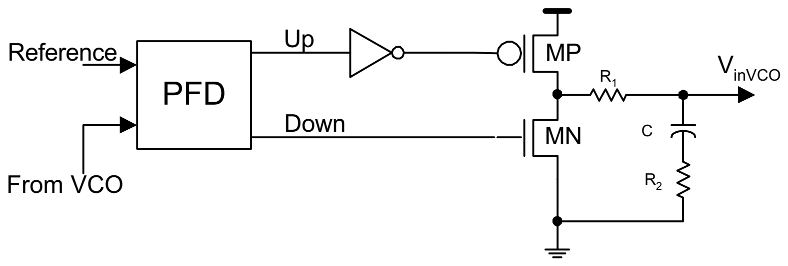


Figure 4.1. A tri-state loop filter

Another way is a current-steering charge-pump loop filter, shown in Figure 4.2. If the current sink and source are well matched, the gain for the charge-pump LF is constant regardless what the initial voltage in the capacitor is. Supply and substrate noise effects can be minimized if the pump current can be made as constant as possible. The current switching circuit with a simple passive LF is widely used for the charge-pump phase and delay-locked loops.

4.1.1. Charge-Pump Basics

In Figure 4.2, the two switches, *M1* and *M2* are connected to a current sink and source respectively. When the PFD indicates *Up* or *Down*, ideally, a constant current I_{pump} charges or discharges the loop capacitor (C_1) respectively to build a proper control voltage for the VCO. The gain of the PFD with a charge-pump LF is given by (referring to Table 2.1)

$$K_{PFD} = \frac{i_{LF}}{\Delta\phi} = \frac{I_{pump} - (-I_{pump})}{4\pi} = \frac{I_{pump}}{2\pi} \tag{4.1}$$

where i_{LF} is the current injected into the loop filter, and K_{PFD} has the unit of amps/radian. With the LF transfer function F(s), the control voltage of the VCO is given by

$$V_{inVCO} = F(s) \cdot i_{LF} \tag{4.2}$$

From the previous discussion, a PFD with the charge-pump LF acts like a real integrator with a pole at the origin. Therefore, a charge-pump PLL is Type II with another integrator from the VCO. A stabilizing zero is needed in the LF to provide a stable operation (*R* in the Figure 4.2).

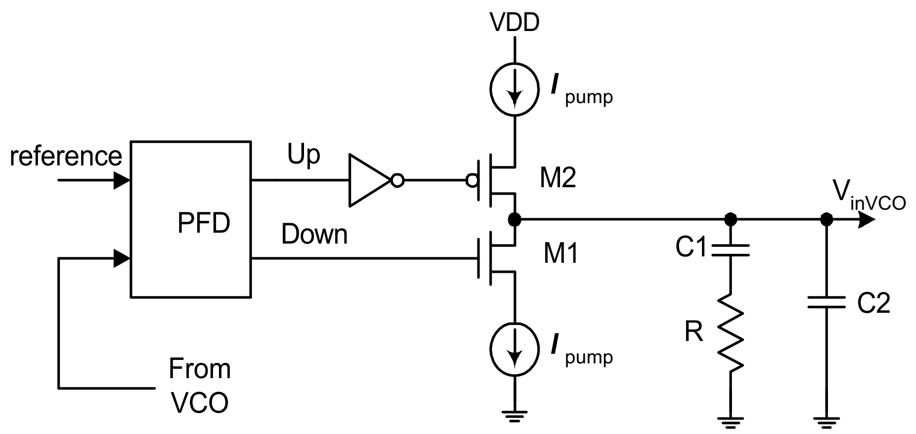


Figure 4.2 A charge-pump loop filter with the PFD

If we ignore the C_2 in Figure 4.2 first, the charge-pump PLL is a second order system and the transfer function of the LF is

$$F(s) = R + \frac{1}{sC_1} \qquad (4.3)$$

The switching characteristic with a charge pump loop filter makes the PLL a discrete-time system, and simple transfer-function analysis is not directly applicable to such system. However, if only a small change of the PLL is made during each input cycle and the loop bandwidth is much less than the input frequency, a continuous-time system analysis can be applied [4]. The closed loop phase transfer function is

$$H(s) = \frac{K(1+sRC_1)}{s^2 + s(KR) + \frac{K}{C_1}} \tag{4.4}$$

where the loop gain $K = K_{PFD}\,K_{VCO}$. Following the same analysis in Chapter 2, the natural frequency and damping factor can be derived as

$$\omega_n = \sqrt{\frac{K}{C_1}} \tag{4.5}$$

$$\zeta = \frac{RC_1}{2}\sqrt{\frac{K}{C_1}} = \frac{\omega_n RC_1}{2} \tag{4.6}$$

With the help of Equation (3.2), the steady state phase error for an input frequency step is

$$\phi_e(t \to \infty) = \lim_{s\to 0} s \cdot H_e(s) \cdot \frac{\Delta\omega}{s^2} = \frac{\Delta\omega}{K \cdot F(0)} = 0 \tag{4.7}$$

Ideally, no static phase error exists in the charge-pump PLL for an input step in frequency. If an M-counter is included in the feedback path to provide the frequency multiplication, $K = K_{PFD}K_{VCO}/M$. Although a wide bandwidth is desired to provide wide lock range and fast lock, there is a stability limit [4], which is given by

$$\omega_n^2 < \frac{\omega_{ref}^2 \cdot R}{\pi(RC_1\omega_{ref} + \pi)} \tag{4.8}$$

where ω_{ref} is the input frequency. Another inherent problem is the ripple out of the charge-pump LF even the loop is locked [4]. For the single-ended charge pump, the pump current charging or discharging C_1 via R will generate a ripple and modulate the VCO frequency. A shunt capacitor C_2 can be used to suppress the ripple, which makes the charge-pump PLL a third-order system. If this high-order pole is put far away from the low-frequency pole, the third-order system can still hold the second-order system properties with the suppression of the high frequency ripples. The transfer function of this second-order LF is given by

$$F(s) = \frac{sRC_1 + 1}{s(C_1 + C_2)(\frac{sRC_1C_2}{C_1 + C_2} + 1)} \tag{4.9}$$

The stability of this third-order PLL is given in [4].

The charge-pump DLL is simple compared to the PLL because the voltage-controlled delay line (VCDL) doesn't act as an integrator since it only adjusts the phase not the frequency. If a single capacitor (*C*) is used for the charge-pump loop filter, the DLL is a first-order system and shown in Figure 4.3. The phase detector will generate proper *Up* / *Down* for the charge pump. V_{ctrl} is the average control voltage of the VCDL and t_o is the delay of the VCDL.

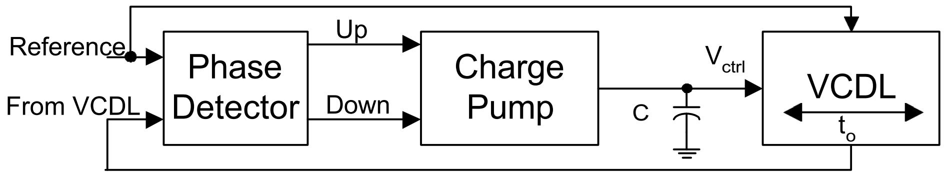


Figure 4.3. A typical charge-pump DLL

The close loop response of the charge-pump DLL can be analyzed with a continuous time approximation, same as assumed for the charge-pump PLL. Assume that input reference signal is at frequency of ω_{ref} and the clock period is $T_{ref} = 2\pi/\omega_{ref}$. The DLL output phase ϕ_{out} can be given by

$$\phi_{out} = t_o \frac{2\pi}{T_{ref}} \tag{4.10}$$

The delay of the VCDL (t_o) can be expressed as the input control voltage V_{ctrl} times the gain of the VCDL, K_V (seconds/V).

With a simple capacitor (C) as the LF, the transfer function for the charge-pump PD is given by

$$\frac{V_{ctrl}}{\Delta\phi} = K_{PD} \cdot \frac{1}{sC} \tag{4.11}$$

where K_{PD} is the gain of the charge-pump phase detector, and $\Delta\phi = \phi_{ref} - \phi_{out}$. For the PFD, $K_{PD} = I_{pump}/2\pi$, for RS-Latch or Hogge's PD, $K_{PD} = I_{pump}/\pi$. The overall closed-loop transfer function is

$$H(s) = \frac{\phi_{out}}{\phi_{ref}} = \frac{K_{PD}K_V\omega_{ref}}{sC + K_{PD}K_V\omega_{ref}} = \frac{1}{1+\dfrac{s}{\omega_L}} \tag{4.12}$$

$$\omega_L = K_{PD}K_V \cdot \omega_{ref} \cdot \frac{1}{C} \tag{4.13}$$

where ω_L can be defined as the loop bandwidth with the relationship of the input frequency (ω_{ref}) shown in Equation (4.13). For the constant charge-pump current and K_V (which can be considered as a constant loop gain $K = K_{PD}\,K_V$), the loop bandwidth will track the operating frequency. Generally, the gain of the VCDL is nonlinear, with an increasing gain as the delay per stage is large. A simulated delay per stage versus control voltage of the VCDL is plotted in Figure 4.4. Five fully differential delay stages are used in the VCDL with a bias generator to provide bias voltage for the voltage-controlled resistor and tail current source [15], referring to Figure 2.12 for each delay stage. A delay interpolation is needed to provide a more linear operation of the VCDL.

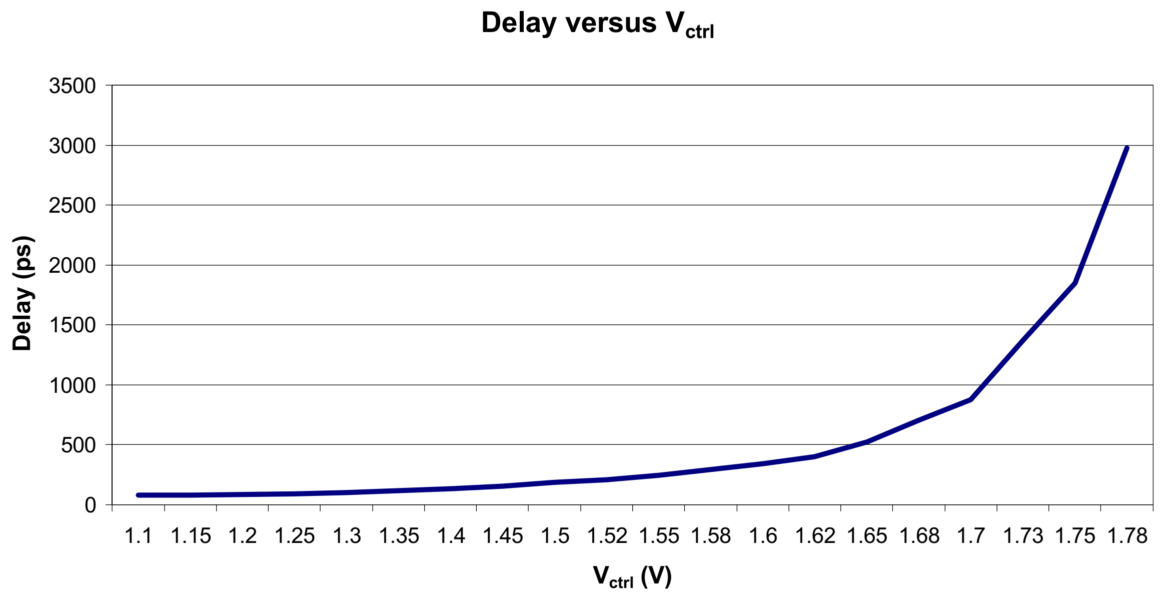


Figure 4.4. Delay versus control voltage of the VCDL

4.1.2. Self-Bias Techniques

Generating constant pump current over the wide operating range is very important for a charge-pump circuit design. Fully differential charge-pump circuit is required to give a better matching. Process and temperature variations make the charge-pump design a challenge. Post-fabrication tuning is needed to provide desire performance. A self-bias technique discussed in [15] uses a half-replica circuit of the differential delay stage to track process and temperature variations, shown in Figure 4.5. The bias voltage for P-devices, V_{BP}, is essentially equal to V_{ctrl} with the aides of the op-amp. V_{BN}, the bias voltage for the tail current source, is generated by using the half-replica of the differential delay stage. The buffer stage is used to isolate the control voltage (V_{ctrl}) from the delay stages. The gain of the VCDL is given by

$$K_V = \frac{N \cdot C_n}{k \cdot (V_{DD} - V_{ctrl} - V_T)^2} \tag{4.14}$$

where *N* is the number of delay stages, C_n is the effective node capacitance of each stage, *k* is the PMOS device transconductance, V_T is the threshold voltage of PMOS devices used as the linear voltage-controlled resistor, and V_{DD} is the supply voltage. Although the bandwidth tracking can be made constant if careful layout is applied, from Equation (4.14), the loop gain is inversely proportional to the square of the supply voltage. Noise from the power supply will degrade the self-bias performance.

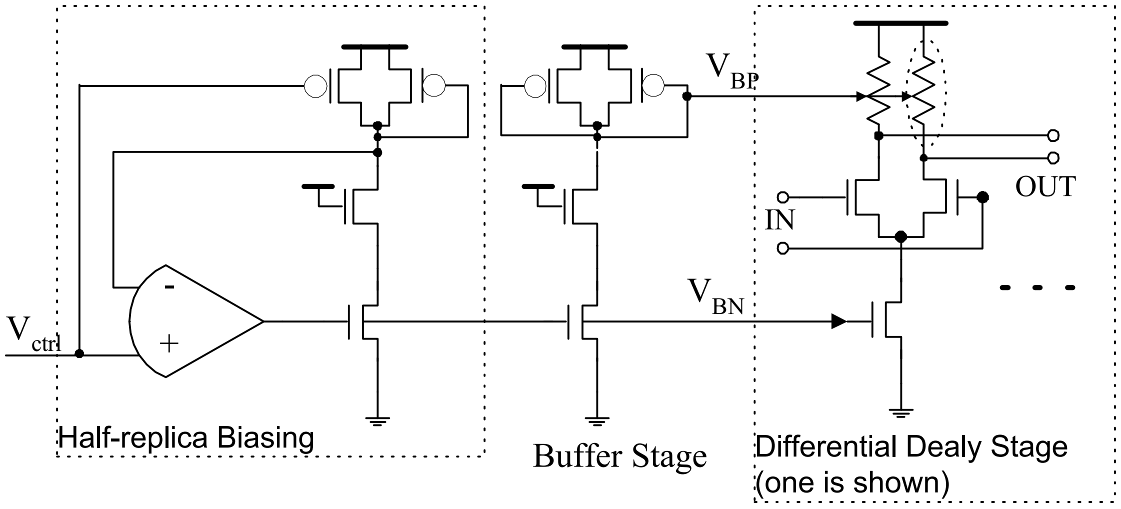


Figure 4.5 A half-replica self-bias generator

A similar differential delay stage can be used as a differential charge-pump circuit to generate V_{ctrl}, shown in Figure 4.6. This architecture can effectively eliminate the dead zone since no net charge deposits into the loop capacitor when both *Up* and *Down* are asserted for a same duration. With the replica bias generator, when in lock, the output stage of this charge-pump circuit will source and sink same amount of current which results in no net charge deposit.

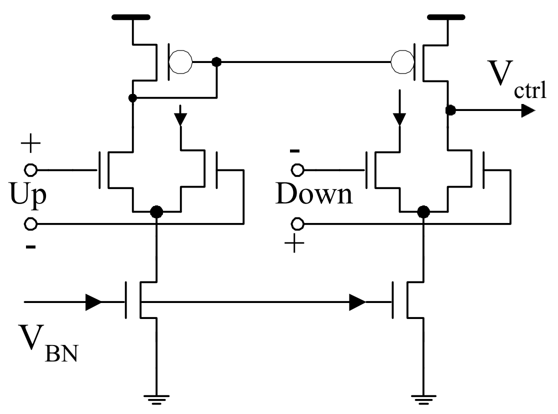


Figure 4.6. A differential charge-pump circuit

4.1.3. Dead Zone

Consider a CMOS implementation of the phase-frequency detector (PFD) in Figure 4.7. The input latches are used to detect the edges. Unlike the traditional edge-triggered D flip-flop which uses master and slave architecture, the NAND-gate-based latch has a symmetric sampling window and shorter setup time and hold time requirements. No complementary signals for *R* and *V* are needed for this PFD and the critical path includes three gate delays: two from the cross-coupled NAND gates, and one from the four-input NAND gate. High frequency operation can be achieved using this configuration.

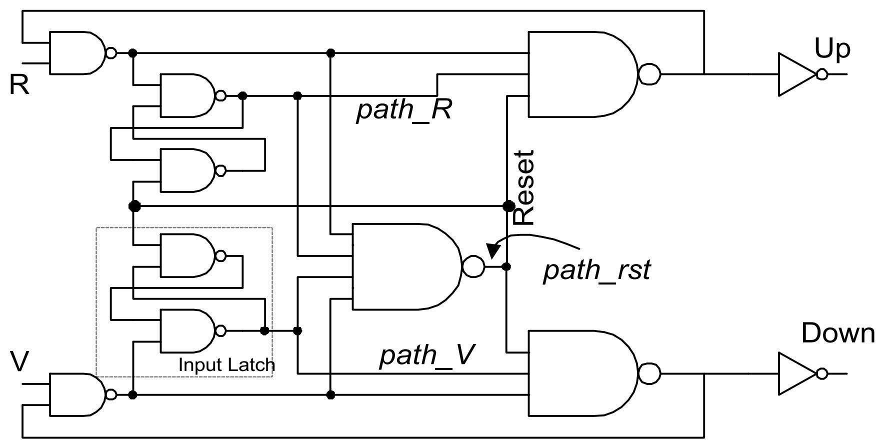


Figure 4.7. A CMOS Phase Frequency Detector

When the loop is approaching lock, the phase difference between *R* and *V* is small, which cause narrow pulses at the PFD outputs. The pulse width depends on the timing between three paths: *path_R*, *path_V* and *path_rst* shown in Figure 4.7, as well as process and temperature variations. Since the *Up* and *Down* are used to control the switches in the charge-pump circuit, it may cause the switches to malfunction if the pulse width is too narrow. This phenomenon is called "Dead Zone" and is critical for a low-jitter design. With the help of Figure 4.8, the dead zone for a charge-pump loop is the area where the switches don't response to the input changes and fail to bring the voltage above the switching point. In the transfer curve of the PFD, a dead zone is the region close to lock ($\Delta\phi \approx 0$) where the gain drops to zero.

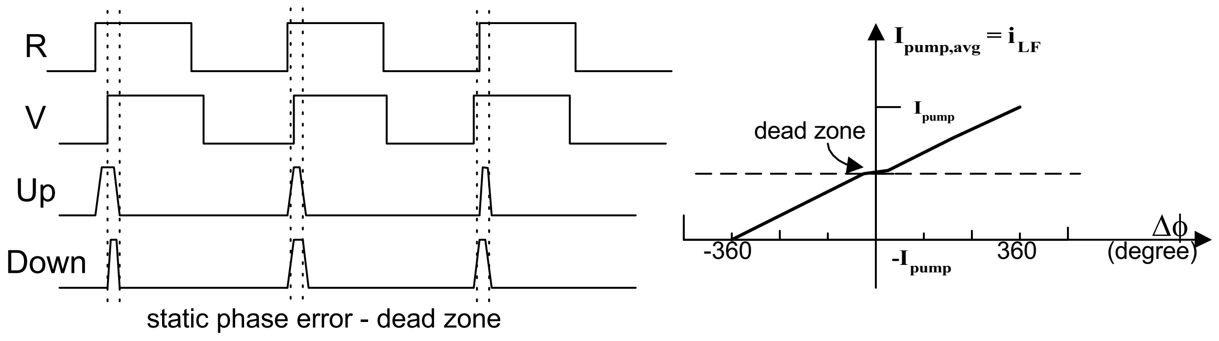


Figure 4.8. Dead zone in the charge-pump circuit

In order to minimize the dead zone and static phase error, a large pump current is desired and parasitic capacitance with switch M1 and M2 should be minimized to improve the slew rate. Adding some delay in the *path_R* and *path_V* in Figure 4.7 will help to eliminate the narrow pulse at the PFD output with the penalty of a static phase error. Adding delay in the *path_rst* will increase the pulse width and result in the equal *Up*/*Down* when in lock. If charge-pump circuit is well matched, no net charge is deposited on the loop capacitor and thus, no dead zone. However, the delay in the reset path will limit the maximum operating frequency. Another way uses a pair of switches for either charging or discharging leg to eliminate the static phase error, shown in Figure 4.9.

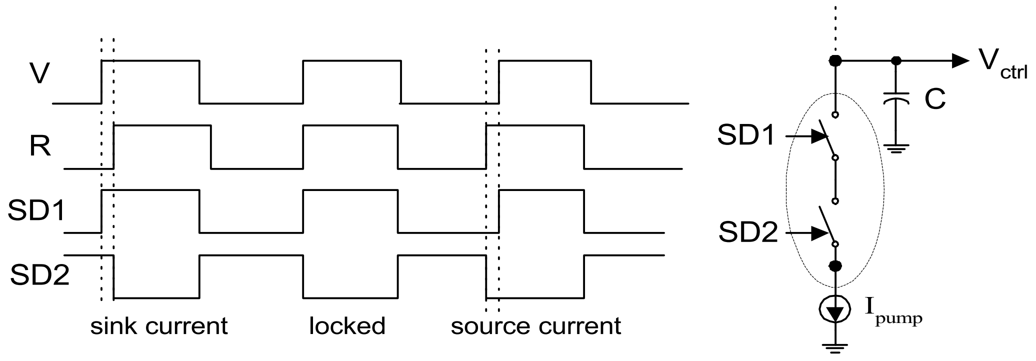


Figure 4.9. A charge-pump circuit using a pair of switches to eliminate the dead zone (Only the lower part is shown)

The small phase error is equal to the difference of signal *SD1* and *SD2*. The current flows only when both switches are ON. The signal swing is limited for the bias circuit to provide constant current (keep the MOSFETs in the saturation region). Differential configuration and wide swing bias circuit are needed to provide desire performance.

4.1.4. Charge Sharing

Because of switching, charge sharing exists in the simple charge-pump circuit and can be illustrated using Figure 4.10. Assume initially *M2* is OFF and *M1* is ON and the loop capacitor is discharged via *M1*and *MN*. With no current flowing in *MP* and *M2*, node *SP* is at V_{DD} and charge is stored in the $C_{gd,MP}$, the gate-drain parasitic capacitor of *MP*. When *M2* is ON and *M1* is OFF, the charge stored in the $C_{gd,MP}$ will cause a spike at the charge pump output. Charge sharing is unwanted and may cause the VCO output jittering. A possible solution [8] is shown in Figure 4.10 with a voltage follower (can be a unit gain buffer). A current path is always available with this configuration. The voltage at node *X* follows the change of node *Y* and node *SP* or *SN* is precharged to a proper value to avoid charge sharing.

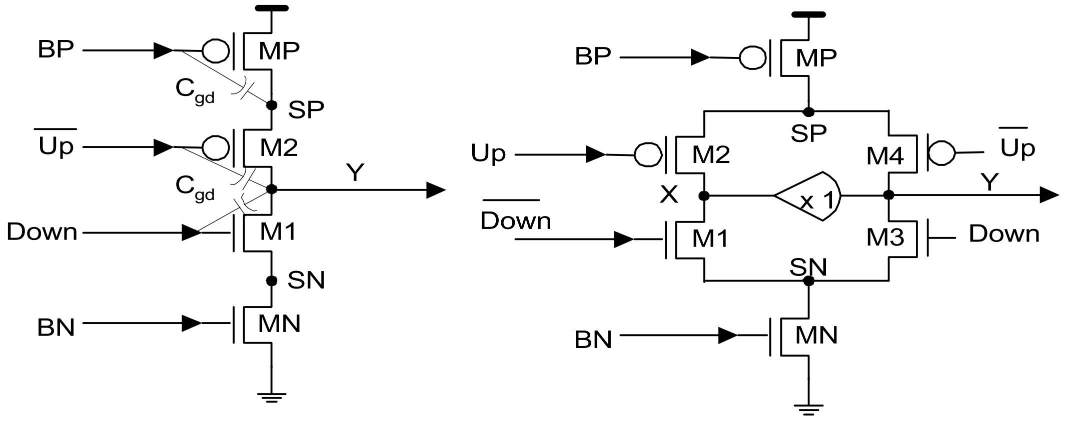


Figure 4.10. Charge sharing and a possible solution

The gate-drain capacitors of the switches also cause signal feedthrough and switching noise at node Y. Minimum size can be used for the charge-pump switches to reduce the signal feedthrough. Another problem is the charge injection, which is caused by the charge stored under the gate oxide from the inverted channel when the switch is ON [55]. As the switch turns OFF, the charge will inject into the node *Y* and cause the voltage to change. A dummy switch can be used to reduce the charge injection.

To design a low-jitter charge-pump loop, biasing circuits, dynamic characteristics such as charge sharing and injection, and dead zone all need to be careful accommodated for each single application. Matched layout is crucial and some critical node may need to be shielded to prevent digital noise. Parasitics extraction and thorough simulations under different corner conditions are important to provide stable operation over process and temperature variations.

4.2. All-Digital Loops

Compared to charge-pump-based analog implementations for phase and delay-locked loops, all-digital techniques can be used to achieve scalability and portability for different process and applications. Existing cell library can be used as all digital building blocks. Generally, all digital loops are more flexible and simple than the charge-pump loops, but also have some limitations in lock range, lock time, and jitter performance. Designing all-digital loops, which can offer comparable locking performance of analog loops, is the purpose of this work.

4.2.1. All-Digital Building Blocks

I. Phase Detector

The digital phase detectors examined in Chapter 2 are also suitable for all-digital loops. In order to drive the digital loop filter, extra circuits are needed to convert narrow width outputs of the digital PD (such as *Up* or *Down* from the PFD when the loop is close to lock) to well-defined digital signals. A simple counter-based all digital PD is shown in Figure 4.11.

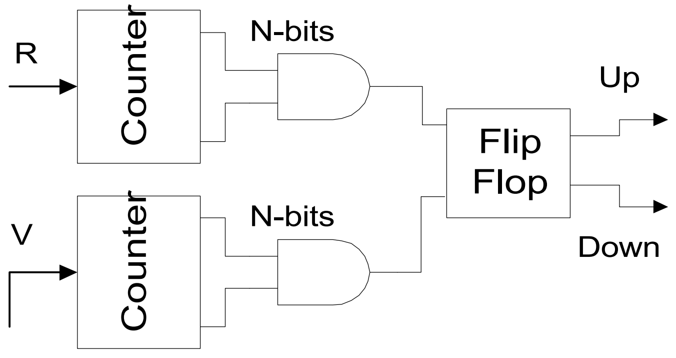


Figure 4.11. A counter-based all-digital phase detector

Inputs, *R* and *V* are periodic signals. The counter is used to count the clock cycles. After the counter reaches N-cycle, the flip-flop will be set or reset to generate the proper *Up* or *Down* signals to indicate the phase difference between *R* and *V*. Another counter-based PD uses simple edge-triggered RS latch at the front to catch the phase difference and feeds the result to an N-bit counter, which is clocked by a high-frequency clock. The phase difference information is stored in the N-bit counter.

If the input data is analog signal, an analog-to-digital convert (ADC) is needed to convert the analog input to a digital signal with a sampling rate greater than the Nyquist rate [54]. Anti-aliening circuit is used to properly restore the data information [55]. Some variations of this kind of PD are discussed in [2].

II. Loop Filter

Since no capacitors are used in the all-digital loop filter, no integration will take place in the all-digital loops. A simple shift register can be used as LF and filter out occasionally signals generated by the PD. An n-bit up-down counter can let the digital-controlled oscillator (DCO) to change according to the number of *Up / Down* from the PD. The pulse width of *Up* or *Down* from the PFD, for example, has no phase error information when an all digital LF is used. Any *Up* pulse will cause the up-down counter to count up by 1 regardless the pulse width. The content of the up-down counter is used to control a binary-weighted current sources or sinks to vary the frequency. When alternative *Up* and *Down* are generated, the loop is close to lock and the minimum resolution of the DCO is reached. This situation is shown in Figure 4.12, where a divide-by-4 is used in the feedback path to scale down the

DCO frequency by 4. It is so called a “bang-bang” -typed loop, which has phase error and jitter because of the discrete tuning.

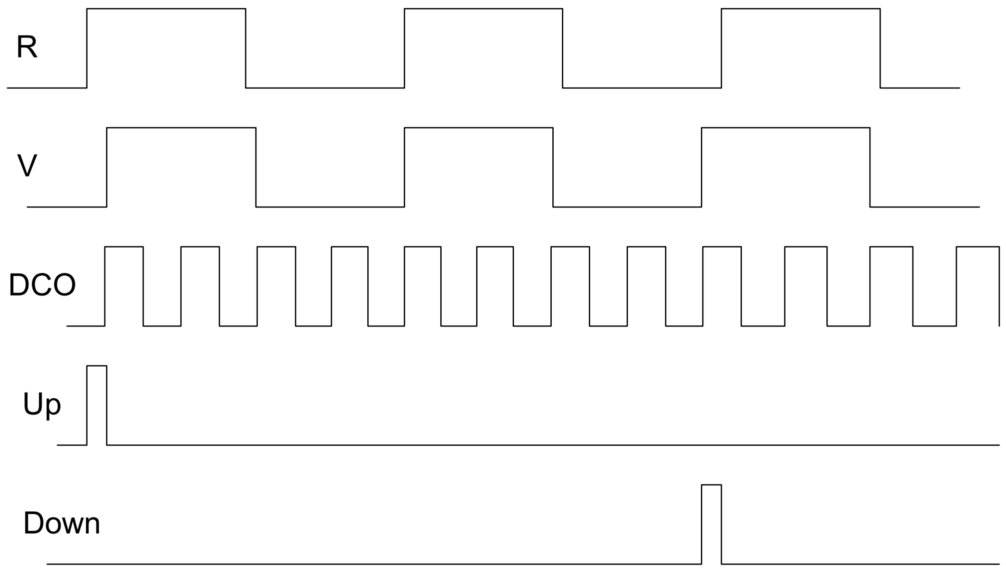


Figure 4.12. The “bang-bang” typed all-digital loops

Another digital LF separates the up-counter and down-counter to count the coming *Up* and *Down* pulses respectively using a high frequency reference. It is the so-called *K* counter and used in the 74xx297 all-digital PLL. The pulse duration of *Up/Down* is counted by the high-frequency K-clock [2] and a *Carry* or *Borrow* is generated when the counter reaches a preset limit. This type of digital LF can work with the phase-detection only PDs. When the *Carry* and *Borrow* occur alternatively, the loop is locked and static phase error still exists in this configuration. A similar *N*-before-*M* counter operates in conjunction with a PFD and the *Carry* or *Borrow* is generated when majority of *M* is *Up* or *Down* (*N*), where *M* is always greater than *N* [2].

III. Digital-Controlled Oscillators (DCOs)

The control of the digital oscillator is typically a digital word rather than the continuous analog signal for the VCO. In some literatures, it is also called a number-controlled oscillator (NCO). In contrast to the VCO, the DCO has a frequency granularity problem which is determined by the minimum resolution of the DCO. Interpolation is needed to improve jitter performance.

A simple DCO is shown in Figure 4.13, where an N-bit counter is used to scale down the frequency from the fixed high-frequency oscillator. The input frequency (f_{in}) should be around f_{ref} / N, where f_{ref} is the high frequency reference.

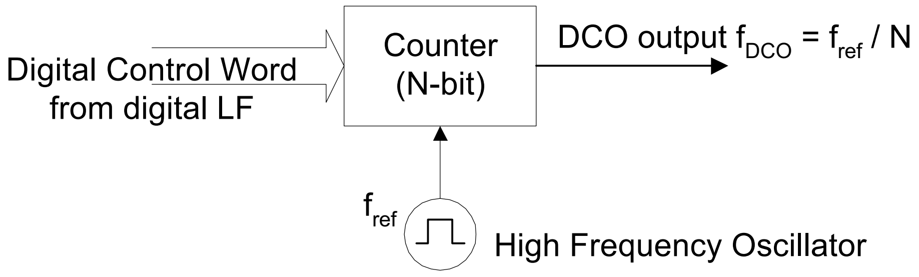

Digital Control Word from digital LF
Counter (N-bit)
DCO output $f_{DCO} = f_{ref} / N$
f_{ref}
High Frequency Oscillator

Figure 4.13. A counter-based DCO

Another popular DCO uses binary-weighted current sources and sinks to change the frequency [19] [20]. The control word can come from the shift register or the up-down counter. Figure 4.14 shows the basic idea of the DCO. The binary-weighted current sources and sinks can be implemented by carefully sizing the *P* and *N* MOSFETs. The control words of the currents, P_0 to P_n and N_0 to N_n come from the digital LF. In order to increase the resolution and suppress the output jitter, a large number of bits are needed and matched layout is critical to guarantee monotonic operations. A modified DCO includes a digital-to-analog converter (DAC) and current-controlled oscillator, where the bias current comes from the DAC. A simple current mirror can be used to convert the digital word to the proper current. The block diagram of this all digital implementation is shown in Figure 4.15.

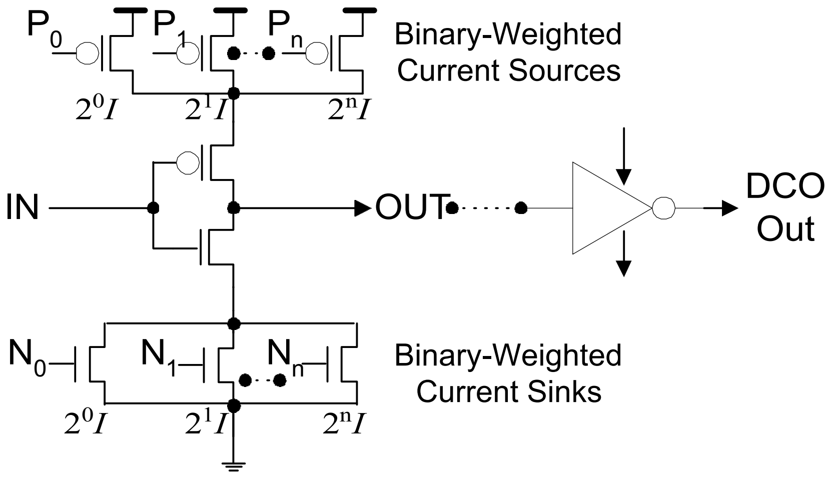


Figure 4.14. A binary-weighted DCO

The quantized error and resolution of the DCO are limitations of the locking characteristics of the all-digital PLL (ADPLL). Other implementation using a high-frequency reference limits the lock range of the ADPLL. Jitter accumulation of the DCO requires some phase maintenance architecture to suppress the static phase error [19].

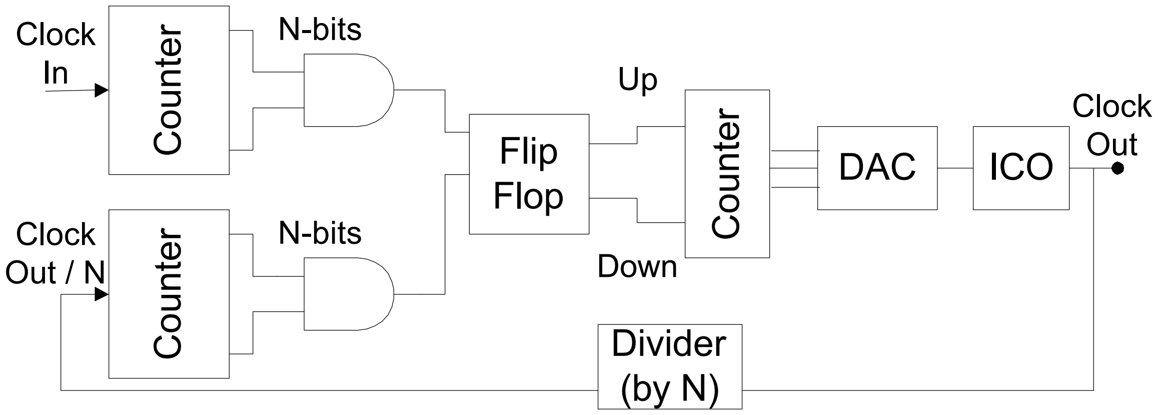


Figure 4.15. A counter-based frequency-locked loop (FLL)

IV. Digital-Controlled Delay Line (DCDL)

All digital techniques are widely used in delay-locked loops because it's simple and low cost compared to the analog implementation. A digital-controlled delay line (DCDL), instead of the DCO, can insert optimum delay between the input and output to minimize the phase difference. Since the delay is tuned by adding or removing delay taps, the jitter of this all-digital DLL is dependent on the basic delay tap (t_d). Assuming that noise from power supply and substrate is negligible, the cycle-to-cycle jitter is about $\pm\, t_d/2$ where the peak-to-peak jitter is $\pm\, t_d$. The basic delay element can be a basic inverter, which has delay typically about 50~100ps under the current CMOS technology.

To select the entry point of the DCDL, a selector, usually a bi-direction shift register, is used and controlled by the digital PD, which generates *shift-left* or *right* to increase or decrease the delay respectively. Figure 4.16 shows the block diagram of a DCDL. The basic delay tap can consist with two inverters and one TG (transmission gate). If the complementary input clock is available, an interleaved delay line can be used to simplify the basic delay tap to only one inverter and one TG (for selection) [24]. The lock range of the DCDL is based on how many delay taps are used.

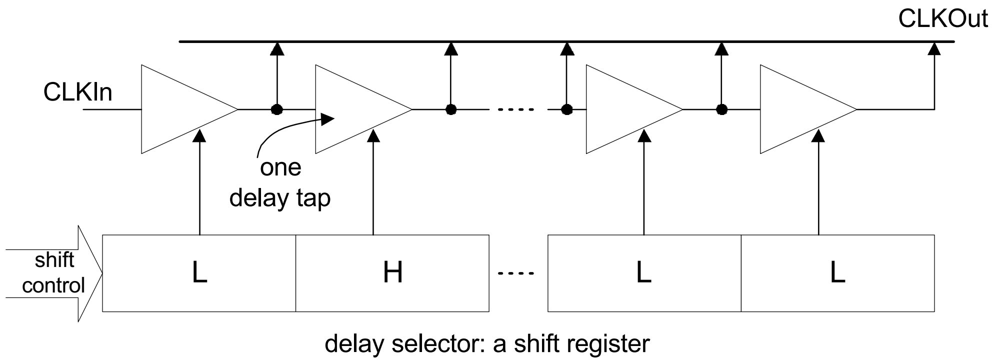


Figure 4.16. The block diagram of a DCDL

4.2.2. Design Criteria of All-Digital Loops

The order of all-digital loops is less than 1, which makes such loops unconditionally stable. If the noise issues are critical, digital filtering can be chosen for the equivalent function of analog LF. For the first-order system, high gain is desired to suppress static phase error and quickly pull in the loop. However, high-gain is same as wide bandwidth for the 1st order system, noise performance is degraded. Generally, two-loop architecture is required to achieve better locking performance for the ADPLL, first frequency locking, then phase maintenance. For the all-digital DLL (ADDLL), the additional loop can provide better resolution to get supreme jitter performance at the cost of large area and more power dissipation.

Typically, the pull-in range of all-digital loops is determined by the range of the DCO or DCDL. Longer delay line or more bits for the DCO can give wider range at the cost of large area, supply noise sensitive and more power dissipation. Within this range, the all-digital loops can always acquire lock. However, the lock time can vary a lot depending on different architecture. In general, a counter-based ADPLL may require a long time to lock because of the N-bit counter. A N-bit shift register used in the all digital LF will result in N additional clock cycles to perform an adjustment for the DCO or DCDL, whereas the loop may exempt from the false detection from the PD. During the frequency acquisition, initially aligned the reference with the DCO output (or the output from the divide-by-N) is helpful to pull in the loop without cycle slips. Initially setting the frequency of the DCO at the middle of the range by simply choosing the proper digital word may shorten the pull-in time (it is equivalent to set the VCO free running frequency to the middle of the VCO range, which is roughly because of drift and variations). For the ADDLL, the initial point should be set at the LSB (least significant bit) of the DCDL to avoid false lock. No drift exists in all-digital loops because the frequency or delay is set by the digital word. It requires much less time to re-acquire lock after system power-down or stand-by.

All-digital design is comparably simple and deterministic. Full custom design is not required and synthesis tools can be used to simplify the circuit design. For the high-performance applications, critical path and timing analysis are necessary to anticipate the locking characteristics. Layout and simulation are also simple compared to the analog loops.

By using the digital tuning, frequency granularity always exists and results in excessive jitter at the output. Interpolation is a widely used method to relieve this problem.

4.2.3. Delay Interpolation

Different ways can be used to achieve a delay which is less than a single inverter. It is also called a fine delay, compared to the coarse delay provided by the DCDL. The best achievable jitter performance is determined by the resolution of a fine delay tap.

I. Fast and Slow Paths

Shown in Figure 4.17, the delay is equal to the difference between two paths. The control signals, Q and Q^*, come from shift register. When $Q = 1$, the slow path is selected, and when $Q = 0$, the fast path is selected. Adjusting the size of inverters can achieve different resolution. With slightly different device size, the vernier delay (Δt) is set by the delay difference of the fast and slow paths. The interpolation range (T_{interp}) is given by

$$T_{\text{int}\,erp} = L \cdot \Delta t \tag{4.15}$$

where L is the number of taps. For each delay tap, four inverts and two TGs are used and large area is needed when wide interpolation range is desired. However, this architecture provides better tracking of process and temperature variations.

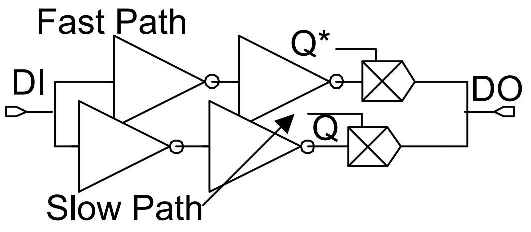


Figure 4.17. One delay tap for the interpolation using fast and slow paths

II. Adjust Loading

The delay can be adjusted by adding or removing the capacitive loading, shown in Figure 4.18. A pair of n-cap and p-cap is used as one delay tap to provide similar loading during both rising and falling edges [28] [26]. The capacitor is made up with a source-drain connected MOSFET which is operated at the strong inversion region. Two inverters at the each end will provide enough gain and isolation. The control word can come from the shift register. If the delay per tap is Δt, and $2L$ taps are used where initially L taps are ON and L taps are OFF, the tuning range is given by

$$T_{\mathrm{int}\,erp} = \pm L \cdot \Delta t \tag{4.16}$$

The size is small for this implementation. The number of taps is limited to provide reasonable transitions and noise immunity.

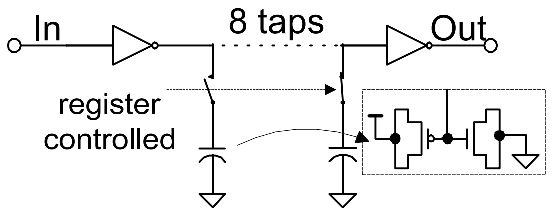


Figure 4.18. The register-controlled capacitors for the delay interpolation

III. Adjust Driving

An inverter matrix, shown in Figure 4.19, can offer various driving capabilities, and hence, different delay. The delay adjustment may not be linear in this case since it adjusts the slope of the output node. A delay interpolation can be realized by using two sets of the matrix for the input (A) and output (B) of one coarse delay tap. An intermediate phase can be created by mixing the ϕ_A and ϕ_B [30]. It is also called phase blending in [24].

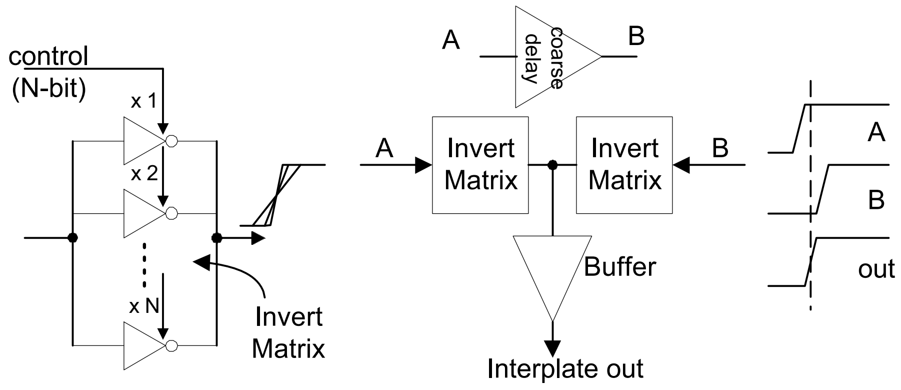


Figure 4.19. Delay interpolation by adjusting the driving

To design a robust all-digital loop, proper delay interpolation is a useful technique. Two practical designs based on the register-controlled mechanism, an ADDLL and an ADPLL are discussed in the following chapters, and some design criteria discussed in this chapter will be further examined in detail.

Chapter 5

An All-Digital DLL using the Symmetrical Delay Line

5.1. Synchronization in the Memory Interface

Synchronization is very important to minimize the timing skew and latency of the modern DRAM system. The state-of-the-art DRAM architecture should offer high sustainable bandwidth, low latency, low power, user upgradeability, and support for large hierarchical memory configurations. For the single data-rate (SDR) devices, such as extended data out (EDO) DRAM and synchronous DRAM, which use either rising or falling edge of a column address strobe (CAS) as the data trigger point for write and read operations, the peak bandwidth (BW_{peak}) is given by [56]

$$BW_{peak} = B \cdot f_{CLK} \qquad (5.1)$$

where B is the bus width of the memory system in Bytes, and f_{CLK} is the clock frequency in MHz. Expanding the bus width as wide as possible can increase the peak bandwidth at the cost of bigger board space. Increasing the clock frequency is another way while the output data strobe (DQS) should be synchronized to the data outputs (DQ outputs) to maximum the valid data access window. Since the clock driver has to drive all DRAMs in parallel, there are some limitations about the achievable clock frequency.

The more attractive memory interface architecture is to utilize both rising and falling edges of the clock for read/write accesses. This will increase the peak bandwidth by factor of two and is so-called a double-data-rate (DDR) memory interface. The timing chart for DDR DRAM is shown in Figure 5.1. The valid output data window can be widen by diminishing the undefined t_{DSDQ} and synchronizing both rising and falling edges of the DQS signal with the output data DQ. Many recently developed or developing memory architectures, such as DDR memories, DDR-SDRAM, SLDRAM (synclink DRAM) and Direct RAMBUS make use of this high-speed DDR memory interface.

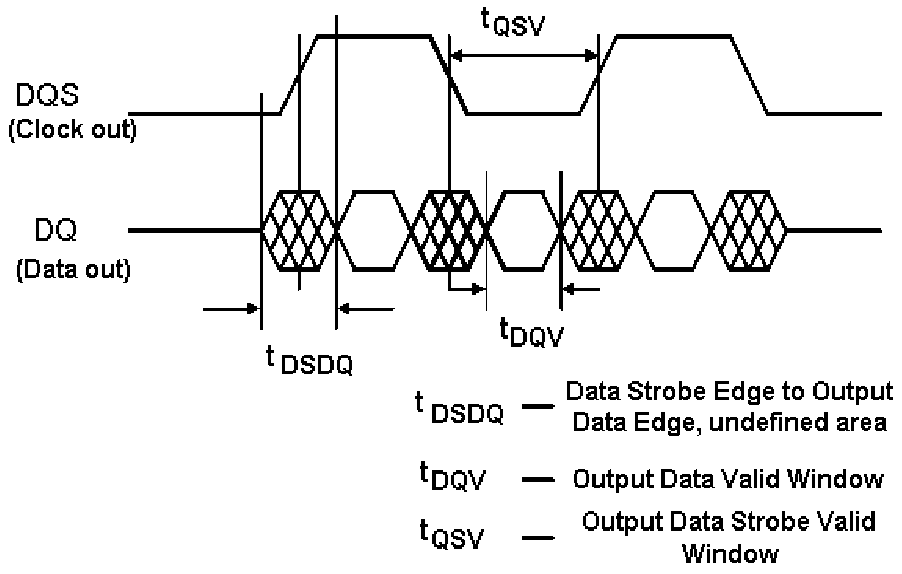

DQS
(Clock out)
DQ
(Data out)
t_{QSV}
t_{DSDQ}
t_{DQV}
t_{DSDQ} — Data Strobe Edge to Output Data Edge, undefined area
t_{DQV} — Output Data Valid Window
t_{QSV} — Output Data Strobe Valid Window

Figure 5.1. Data timing diagram for DDR DRAM

With variations in temperature and process shifts, a fixed delay inserted between DQ and DQS cannot provide stable operations. A feedback mechanism is needed to dynamically adjust the delay and provide optimum timing. DLL is well suited for this application and widely used in the high-speed memory interface to provide synchronization.

5.2. Register-Controlled Symmetrical Delay Line (RSDL)

A register-controlled delay-locked loop (RDLL) [21] [27] has been used to adjust the time difference between the output and input clock signals in SDRAM. Since the RDLL is an all-digital design, it provides robust operation over all process corners. Another solution to the timing constraints found in SDRAM was given in [23] with the synchronous mirror delay (SMD). Compared to RDLL, the SMD doesn't provide as tight of locking as the RDLL but has the advantage that the time to acquire lock between the input and output clocks is only two clock cycles. As the clock speeds used in DRAM continue to increase the skew becomes the dominating concern out weighing the disadvantage of the added time to acquire lock needed in an RDLL.

A modified register-controlled symmetrical delay line [22] is suitable for the DDR memory interface. Here, "symmetrical" means that the delay line has the same delay whether a HIGH to LOW or a LOW to HIGH logic signal is propagating along the line. No extra duty-cycle correction circuit is needed for this implementation. Symmetrical transitions also have additional benefits of jitter performance (referring to Chapter 3).

To understand the symmetrical delay line, a NAND gate based two-stage delay line is shown in Figure 5.2, instead of using a NAND + Inverter as the unit delay stage as was done in [21]. The problem when using a NAND + Inverter as the basic delay element is that the propagation delay through the unit delay resulting from a HIGH to LOW transition is not equal to the delay of a LOW to HIGH transition ($t_{PHL} \neq t_{PLH}$). Further, this delay varies from one run to another. Although the skew per stage seems tinny, for example 50ps between t_{PHL} and t_{PLH}, the total skew of the falling edges through 20 stages will be 1ns. Large number of delay stages/taps used in the delay line is required to achieve wide lock range. Because of this skew the NAND + Inverter delay element cannot be used in a DDR DRAM. In the modified symmetrical delay element, another NAND gate is used instead of an inverter (two NAND gates per delay tap). This scheme guarantees that $t_{PHL} = t_{PLH}$ independent of process variations since while one NAND switches from a HIGH to LOW, the other switches from LOW to HIGH. An added benefit of the two-NAND delay element is that two point-of-entry control signals are now available (*A* and *B* shown in Figure 5.2). Both are used by the shift register to solve the possible problem caused by the power-up ambiguity in the shift register.

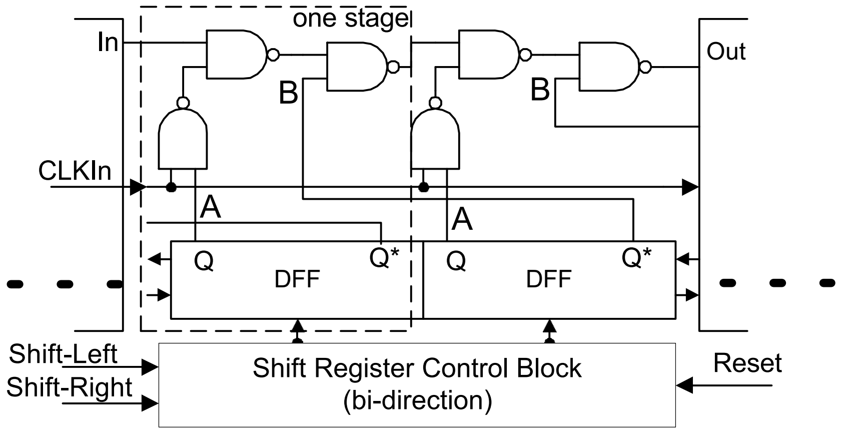


Figure 5.2. Symmetrical register-controlled delay line

The control mechanism of this RSDL can be shown in Figure 5.3. The input clock (*CLKIn*) is a common input to every delay stage. The shift register is used to select a different tap of the delay line (the point of entry for the input clock signal into the symmetrical delay line). The complementary outputs of each register cell are used to select the different tap: *Q* is connected directly to the input *A* of a delay element, and *Q** is connected to the previous stage of input *B*. From right to left, the first LOW to HIGH transition in the shift register sets the point of entry into the delay line. The input clock will pass through the tap with a HIGH logic state in the corresponding position of the shift register. Since the *Q** of this tap is equal to a LOW it will disable the previous stages; therefore, it doesn't matter what the previous states of the shift register are (shown as don't cares, *X*, in Figure 5.3). This control mechanism guarantees that only one path is selected. This scheme also eliminates power-up concerns since the selected tap is simply the first, from the right, LOW-HIGH transition in the shift register.

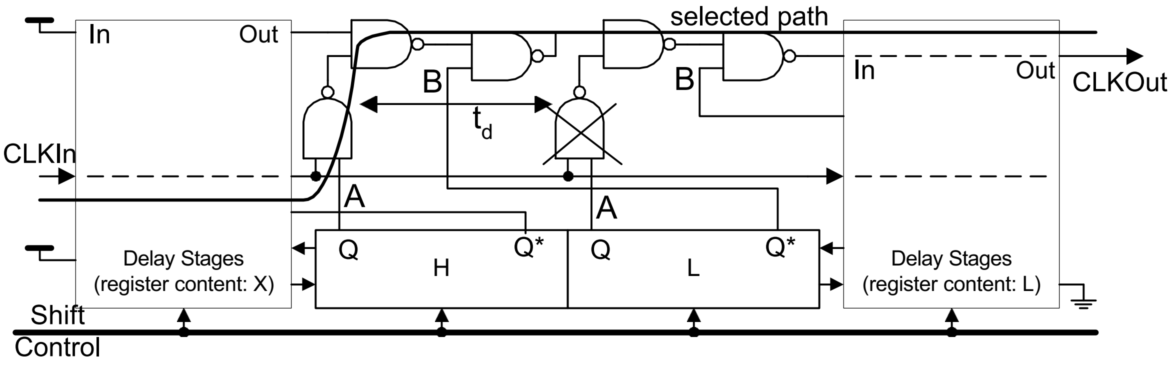


Figure 5.3. Delay Line and Shift Register

The effective tuning range of the RSDL is determined by the minimum delay per tap (t_d), which is the delay of two NAND gates, and the number of taps (N). If the control point *B* of the last tap is set HIGH, *CLKIn* will propagate through the last tap when the content of the shift register is all-HIGH. An all-LOW content means that the limit of the RSDL is reached and the RSDL will be reset to the initial condition. Therefore, the content of the shift register varies from 11…1 to 10 ..0 where the tuning range (R_{tune}) is given by

$$t_d \le R_{tune} \le (N-1)\cdot t_d \tag{5.2}$$

Since the basic resolution is determined by t_d, t_d cannot be too large and N is the only parameter to adjust the tuning range.

5.3. An All-Digital DLL using the RSDL

To design a register-controlled symmetrical DLL (RSDLL), the target specifications for this all digital implementation are listed below.

1) Robust operation eliminating the need for post-production tuning (something required in an analog implementation).
2) Operating frequency ranging from 145 MHz (290 Mbits/s/pin) to 250 MHz (500 Mbits/s/pin).
3) Tight synchronization (skew less than 5% of the cycle time) between the output clock and data on both rising and falling edges of the output clock.
4) Low skew between the input and output clocks (with low, < 5%, duty cycle distortion).
5) Power supply voltage operating range from 2.5 V to 3.5 V
6) Portable for ease of use in other processes.

5.3.1. System Considerations

The block diagram of this RSDLL is shown in Figure 5.4. From the system point of view, this all-digital DLL is a 0th-order system, where no integration is involved in this implementation. The operating frequency range is not only determined by the tuning range (R_{tune}) of the RSDL, but also determined by the input buffer and propagation delay in the clock path. The replica input buffer dummy delay in the feedback path, in Figure 5.4, is used to match the delay of the input clock buffer. In general, any component in the data path should be added equivalently at the clock path to tack the variations.

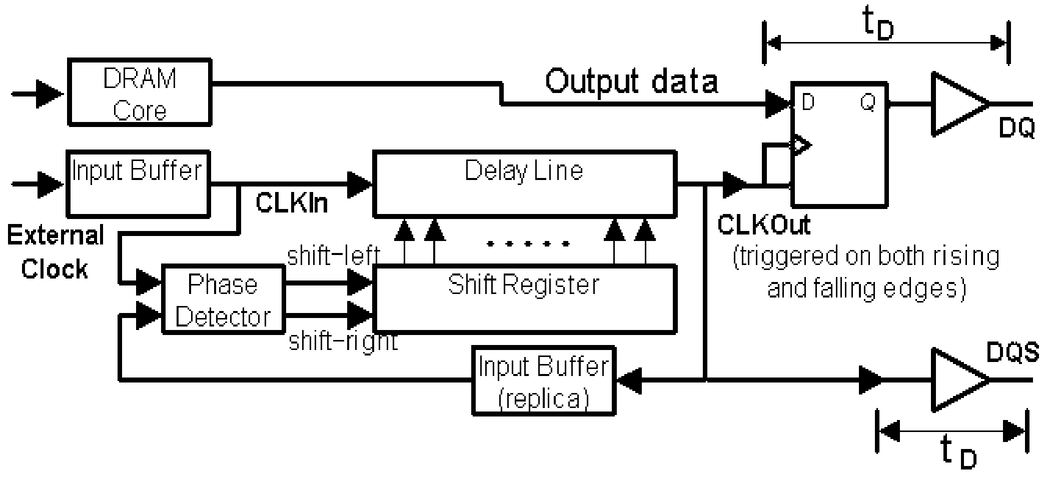


Figure 5.4. Block diagram of the RSDLL

Assuming that the DLL is locked when the delay between *External Clock* and *CLKOut* is equal to one clock period (T_{CLK}), and the delay except for the delay line is t_B, then the lock range is given by

$$\omega_L = 2\pi \cdot \frac{1}{R_{tune} + t_B} \tag{5.3}$$

With the help of Equation (5.2) and (5.3), the maximum achievable operating frequency is given by

$$f_{\max} = \frac{1}{t_d + t_B} \tag{5.4}$$

To achieve high-speed operation, the delay with the peripheral circuits is as important as the core DLL delay line. The lower range of the operating frequency is inversely proportional to the number of delay taps (*N*).

$$f_{\min} = \frac{1}{(N-1)t_d + t_B} \tag{5.5}$$

Adding more delay taps will increase the lock range of the RSDLL at the cost of increased layout area. Longer delay line is susceptible to supply variations. Decoupling capacitor or other layout techniques are needed to provide clean supply and substrate.

The lock time is also related to the number of delay taps used in the RSDL. In order to stabilize the operation of the shift register (SR) and give enough time for the content of the SR to update after a *Shift-Left* or *Right*, a divide-by-*M* (a SR or a counter) is used in the phase detector. As discussed in Chapter 4, this divide-by-*M* acts like a digital LF and will slow down the loop operation by factor of *M*. In the worst case, the RSDL moves from the initial state (first bit at right) all the way to the left, and the lock time is given by

$$T_{L,worst} = M \cdot (N-1) \cdot T_{CLK} \tag{5.6}$$

The phase detector (PD) design is directly determined the lock behavior of this all digital DLL. The PD is used to compare the relative timing of the edges of the input clock signal and the feedback clock signal, which comes through the delay line, controlled by the shift register. The outputs of the PD, *Shift-Right* and *Shift-Left*, are used to control the shift register. When the rising edge of the input clock is within the rising edges of the output clock and one unit-delay of the output clock, both outputs of the PD go to logic LOW and the loop is locked. The resolution of this RSDLL is determined by the size of a unit delay used in the delay line. The cycle-to-cycle jitter of this RSDLL is given by

$$\sigma_c = \pm \frac{t_d}{2} \tag{5.7}$$

Since the RSDLL changes the delay by adding or removing delay taps, the slope of the transitions will be the same and immune to noise. Good jitter performance is anticipated under this concern. Delay interpolation can further improve the jitter performance. For this design, single-loop architecture is used.

5.3.2. Circuit Design

Circuit design of this all-digital DLL is straightforward with only two major parts: all-digital phase detector and register-controlled delay line. A dedicated all-digital phase detector (ADPD) is designed to detect the phase error and ensure stable operation of the RSDLL.

When design the ADPD used in DLLs, two major features should be kept in mind: 1) only phase detection is needed and 2) the minimum adjustable range is limited by the resolution of the delay line. The minimum delay using RSDL is equal to the delay of two NAND gates. In order to track the variations over process and temperature corners, a dummy delay of one delay tap is used in the ADPD to provide the limitation. Phase error less than the basic delay (t_d) will be ignored and no shift signal is generated under this situation. Stable operation can be achieved without bouncing back and forth in the delay line.

The block diagram of the ADPD is shown in Figure 5.5. Two D flip-flops are used to catch the input difference. In order to stabilize the movement in the shift register, after making a decision, the phase detector will wait at least two clock cycles before making another decision. A divide-by-two using a simple D flip-flop is applied to select the shifting point. Figure 5.6 shows the waveforms associated with the different points of this PD. Every other decision, resulting from comparing the rising edges of the external clock and the feedback clock, was used with the *Select* signal. This will provide enough time for the shift register to operate and the output waveform to stabilize before another decision by the PD is implemented. The unwanted side effect of this delay is an increase in the lock time by the factor of two.

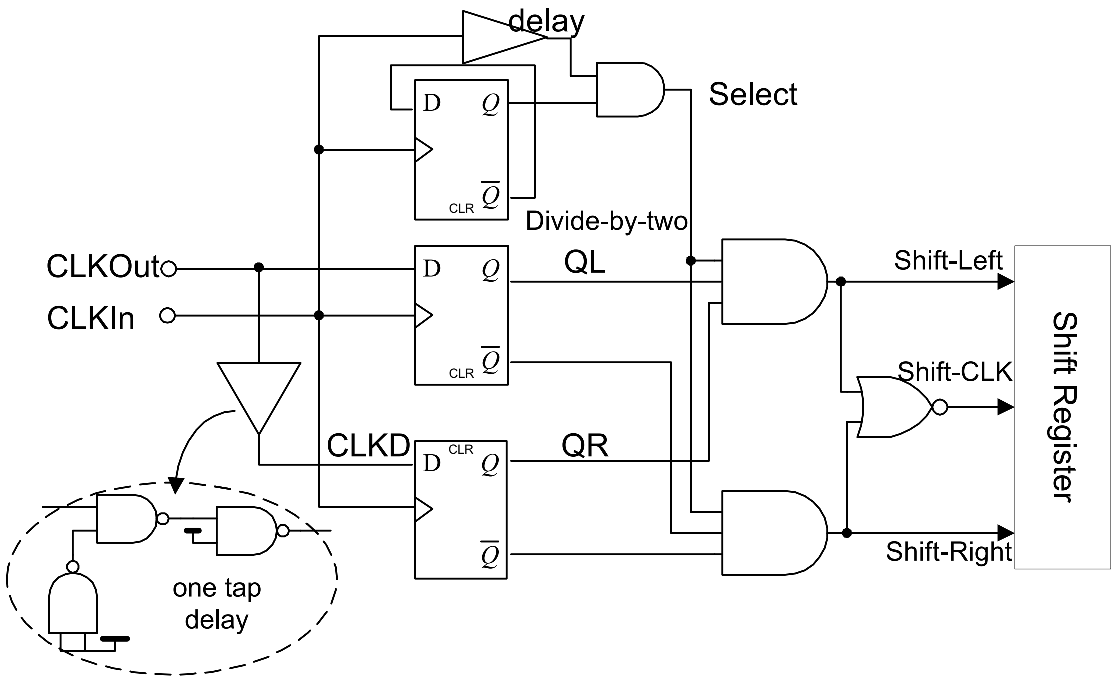


Figure 5.5. All-digital phase detector for the RSDLL

The shift register of the RSDL is clocked by combining the *Shift-Left* and *Shift-Right* signals (*Shift-CLK*). The power consumption will decrease when there are no *Shift-Left* or *Right* signals and the loop is locked. Like the PFD, this ADPD also provides three states: shift left, shift right and no change, as shown in Figure 5.6.

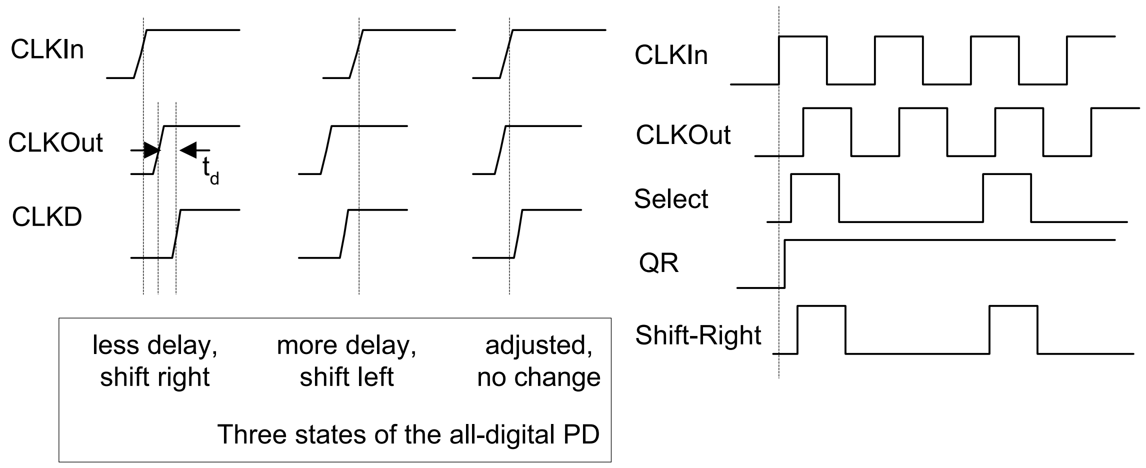


Figure 5.6. Operations of the all-digital PD

A jitter within the minimum resolution of the RSDL exists in this all-digital phase detector. Missing edges will cause error at the outputs, which may not be the case for the clock synchronization. Another concern with the phase detector design is the design of the flip-flops (FFs). To minimize the static phase error, very fast FFs should be used, ideally with zero setup time. A dynamic D-FF with single-phase clock can be a good candidate as shown in Figure 5.7. Also, the metastability of the flip-flops becomes a concern as the loop becomes locked. This together with possible noise contributions and the need to wait, as discussed above, before implementing a *Shift-Right* or *–Left* may increase the desirability of adding additional filtering in the phase detector. Some possibilities include increasing the divider ratio used in the phase detector or using a shift register in the phase detector to determine when a number, say four, *Shift-Rights* or *–Lefts* have occurred. For the present design a divide-by-two in the phase detector is used because of lock time requirements. For 48 delay taps, the maximum lock time for monotonic response of the PD is equal to 94 T_{CLK}, where T_{CLK} is the clock period, referring to Equation (5.6).

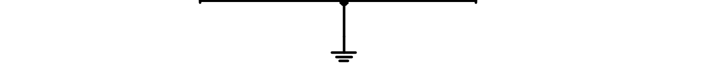


Figure 5.8. A dynamic D flip-flop with single-phase clock

5.3.3. Experimental Results

The RSDLL was fabricated in a 0.21-μm 4-poly double-metal CMOS technology (a DRAM process). A photograph of the RSDLL test scribe is shown in Figure 5.9. A 48-stage delay line is used with an operation frequency of 125MHz to 250MHz. The maximum operating frequency was limited by delays external to the DLL such as the input buffer and interconnect. There was no noticeable static phase error on either rising or falling edges. Figure 5.10 shows the resulting RMS jitter versus input frequency. One sigma of jitter (cycle –to-cycle jitter) over the 125MHz to 250MHz frequency range was below 50ps. The peak-to-peak jitter over this frequency range was below 150ps. In the lower frequency range, from Figure 5.10, jitter is larger than that in the upper frequency range. For the RSDLL, more delay taps are used to get longer delay at the low operating frequency. Longer the delay line is, the more it is susceptible to the power supply and substrate noise.

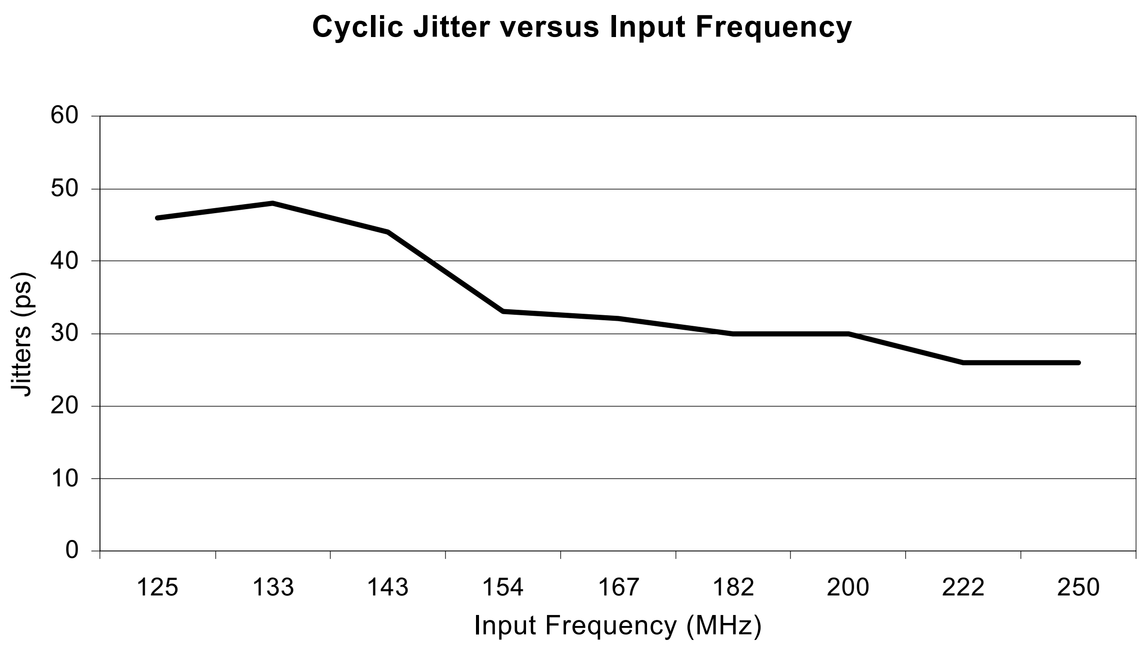


Figure 5.10. Measured RMS jitter versus input frequency

The measured delay per stage versus power supply voltage (VCC) and temperature is shown in Figure 5.11. Note that the 150ps typical delay of a unit delay element was very close to the rise and fall times on-chip of the clock signals and represents a practical minimum resolution of a DLL for use in a DDR DRAM fabricated in a 0.21 μm process. When the VCC is less than 2.5V, the delay per stage varies a lot, which gives the operating range from 2.5V to 4V. Under the extreme conditions, the delay of two-NAND gates can vary from 90ps (fast corner) to 220ps (slow corner). For the low-power operation, the resolution of this delay line may not meet the design requirement. Delay interpolation can be used to further improve the jitter performance for low supply voltages.

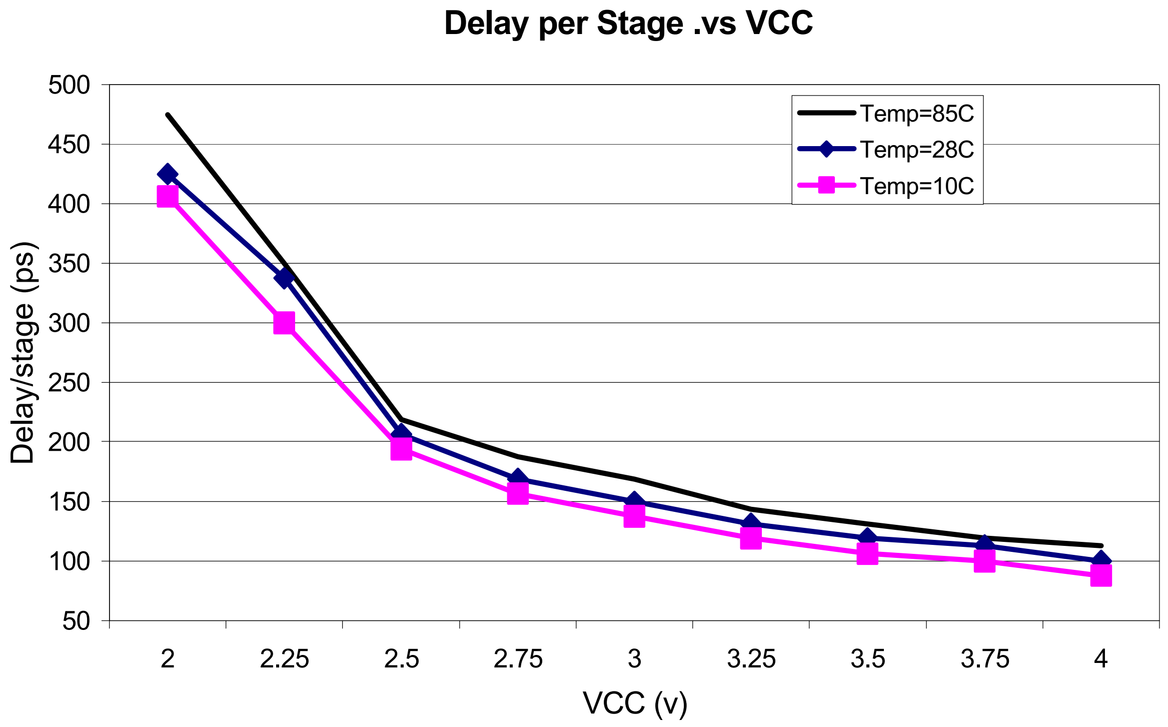


Figure 5.11. Measured delay per stage versus VCC and temperature

The power consumption (current draw of the DLL when VCC = 2.8 V) of the prototype RSDLL is illustrated in Figure 5.12. The power consumption was mainly determined by the dynamic power dissipation of the symmetrical delay line. The NAND delays, in this test chip, were implemented with 10μm/0.21μm NMOS and 20μm/0.21μm PMOS. By reducing the widths of both the NMOS and PMOS transistors, the power dissipation can be greatly reduced without a speed or resolution penalty (with the added benefit of reduced layout size).

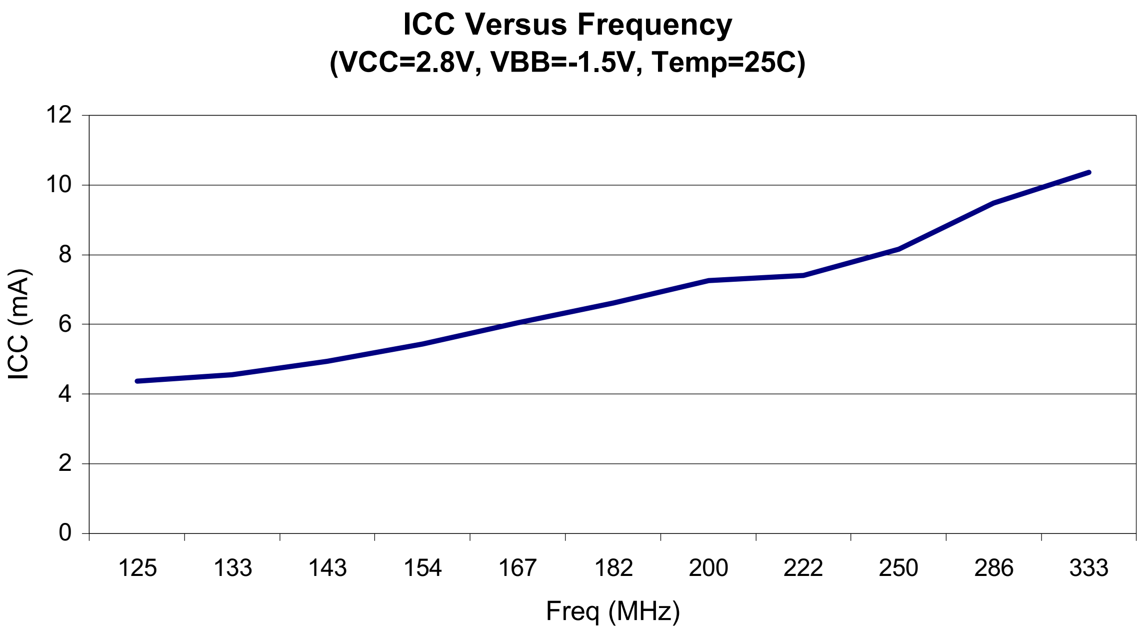


Figure 5.12. Measured ICC versus input frequency

The main features of this all-digital DLL are summarized in Table 5.1. The jitter performance is indeed comparable to the analog implementations [12] [18]. Experimental results verify that this RSDLL is stable against temperature, process and power supply variations. The simplicity and all-digital implementation make this DLL scalable and portable for other processes and applications.

Operating Range	125MHz – 250MHz
Power Supply Voltage (VCC)	2.5V – 4V
Number of Delay Stages	48
Unit Delay (Two NAND gates)	90ps – 220ps
RMS Jitter	< 50ps
Peak-to-Peak Jitter	< 150ps
Power Dissipation	20mW@ VCC=2.8V, 200MHz
Duty Cycle Distortion	< 5%
Process Technology	0.21-μm 4-poly double-metal CMOS

Table 5.1. Summary of the RSDLL test scribe characteristics

5.4. Phase Shifter and Register-Controlled Oscillator

Several symmetrical delay lines can be used to generate exact phase shift referred to the input clock signal. A block diagram of such application is shown in Figure 5.13.

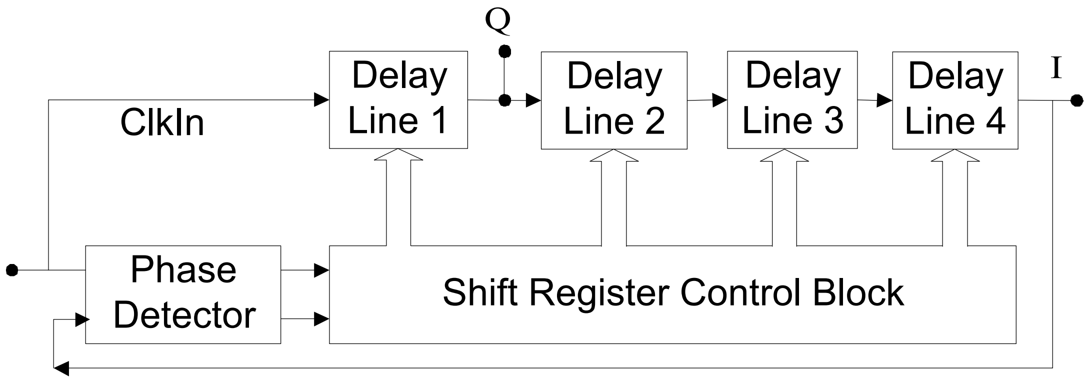


Figure 5.13. Block diagram of the RDLL with in-phase and quadrature phase shifter

Since the total delay through the four delay lines is one period after *ClkIn* when the loop is locked, the output of delay line 4 is in-phase (*I*) of *ClkIn* and output of delay line 1 is quadrature part (*Q*). This configuration is useful in communications that require both *I* and *Q* signals. Unlike the analog implementation, which using fixed number of delay stages and adjusting the delay per stage, the RSDL used for each delay line adjusts the number of delay taps to change the delay, while the delay per tap (t_d) is fixed. To generate arbitrary phase relative to the reference, a large number of delay lines are needed and the resolution is worse for this configuration. For example, to generate I and Q from the configuration shown in Figure 5.13, four taps are added or removed during each shifting, which gives the resolution of 4 x t_d. To generate equal interval, small phase shifts, analog implementation such as voltage-controlled delay line has to be used.

When feeding the clock output (*CLKOut*) to the clock input (*CLKIn*) in the RSDL, with proper phase shift, a register-controlled oscillator (RCO) is formed and the frequency is digitally controlled by the content of the shift register, shown in Figure 5.14.

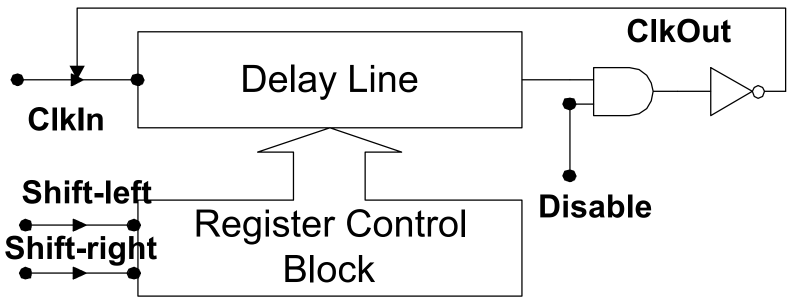


Figure 5.14. Block diagram of a register-controlled oscillator (RCO)

A *Shift-right* decreases the delay through the delay line and thus increases the oscillation frequency while a *Shift-left* increases the delay and thus decreases the frequency. Unlike the VCO which changes the frequency by tuning the delay per stage, the RCO varies the oscillating frequency by changing the number of delay taps. There is no slew rate limitation with this digital tuning at the cost of frequency granularity. With a typical delay of 150ps through one delay element, the period of this RCO is changed by 300ps. An all-digital PLL based on the RCO is discussed in the next chapter with proposed architecture to minimize this inherent frequency granularity associated with the digital tuning.

Chapter 6

An All-Digital PLL based on the Register-Controlled Oscillator

6.1. Design an All-Digital PLL (ADPLL)

Compared to the all-digital DLL, ADPLL is more challenge because of

1) A clock signal is generated by an oscillator rather than passing through a delay line;
2) Both phase and frequency need to be locked in contrast to only phase locking in DLLs;
3) Frequency granularity is more serious due to the jitter accumulation in oscillators.

In spite of these complexities, phase-locked loop is still desirable for some applications, such as clock recovery and frequency synthesis. For clock synchronization, the lock range of the ADPLL is primarily determined by the frequency range of the digital oscillator. With the benefits of all-digital implementation, the methods for developing an all-digital PLL are explored in this chapter.

From Chapter 4, the counter-based ADPLL requires a high-frequency clock reference and the lock time may be long. In some topologies, only the frequency is locked, which is acceptable for frequency synthesis application but not clock synchronization. When an N-bit counter is used as a digital-controlled oscillator (DCO), truly frequency synthesis can not be achieved. For the DAC (digital-to-analog converter) based implementation, the DCO uses binary weighted current sources and sinks which is controlled by digital words. The tuning method is sensitive to the noise from supply and substrate. Any mismatching of the binary weighted current sources and sinks will make the delay vary non-monotonically.

This paper describes an innovative all-digital PLL based on a register-controlled oscillator (RCO). A two-loop architecture with the fine phase adjustment of this ADPLL makes its performance comparable to analog implementations. Some target specifications for this RCO-based ADPLL are:

1. All-digital implementation with robust operation and good stability;
2. Fast locking (less than 100 clock cycles)
3. Tight locking (with phase jitter less than 50ps) and low, <5% duty cycle distortion;
4. A frequency granularity below 1%;
5. Wide operating range and no need for a high-frequency reference;
6. Both frequency and phase locking;

In order to realize above specs, frequency granularity and jitter accumulation need to be accommodated to ensure robust operation and small phase frequency variations. It is also desired to have a wide lock range as well as fast locking for this digital PLL.

6.2. Two-Loop Register-Controlled Oscillator (RCO)

Frequency granularity is critical to design an ADPLL and primarily related to the digital oscillator. Delay interpolation is the best method to minimize the frequency granularity in all-digital implementation. In this section, generalized formulas are derived to characterize the frequency granularity of the RCO. Two-loop architecture is proposed and analyzed to illustrate the significant improvement over the single-loop RCO.

6.2.1. Frequency Granularity of the RCO

A register-controlled symmetrical delay line (RSDL) is used to form the RCO in this design, which can give little duty-cycle distortion because of the symmetrical transitions. However, the following derivation is independent to what kind of the delay element is used.

Assume N taps are available in the delay line and the delay per tap is $t_{d,}$ which is treated as a constant for each tap. If the entry point of the delay line is n (the number of 0s in the shift register, which can also be considered the *control number* in the shift register), where $1 < n < N$-1, then the oscillating frequency of the RCO is given by

$$f_{RCO} = \frac{1}{2 \cdot n \cdot t_d} \tag{6.1}$$

If the phase detector makes no decision when the period is within $2nt_d \pm t_d$, then the frequency difference, referred to the nominal value f_{RCO}, can be expressed as

$$\Delta f_n = f_{RCO} - \frac{1}{2 \cdot n \cdot t_d \pm t_d} = \pm \frac{1}{t_d} \frac{1}{2n \cdot (2n \pm 1)} = \pm \frac{f_{RCO}}{2n \pm 1} \tag{6.2}$$

The frequency granularity can then be defined as

$$\eta_f = \frac{\Delta f_n}{f_{ref}} = \frac{1}{2n} \tag{6.3}$$

where f_{ref} is the reference frequency, and η_f is the frequency granularity. When f_{ref} is within the range of Δf_n, the RCO output frequency is f_{RCO}. It's interesting to notice that the frequency granularity has no direct relationship with delay per tap, but rather is inversely proportional to the control number (n). For lower oscillating frequency and large n, the frequency granularity is small, while the performance is degraded at high speed. For the single-loop architecture, to achieve $\eta_f < 1\%$, the n must be greater than 50 according to

Equation (6.3). This will dramatically limit the lock range of the single-loop ADPLL. Large number of delay taps used in the RCO can improve the frequency granularity and lock range at the cost of large area and poor noise performance (more noise sources are added in the RCO).

In reality, since the *CLKIn* is a common input to all delay taps, shown in Figure 6.1, a gain stage is needed to provide enough drive. Propagation delay is also related to the feedback path and the common input path. The delay associated with these can be called t_P. Without losing generality, we can further assume that t_P is related to t_d by

$$t_P = \lambda \cdot t_d \tag{6.4}$$

where λ is a factor greater than 1.

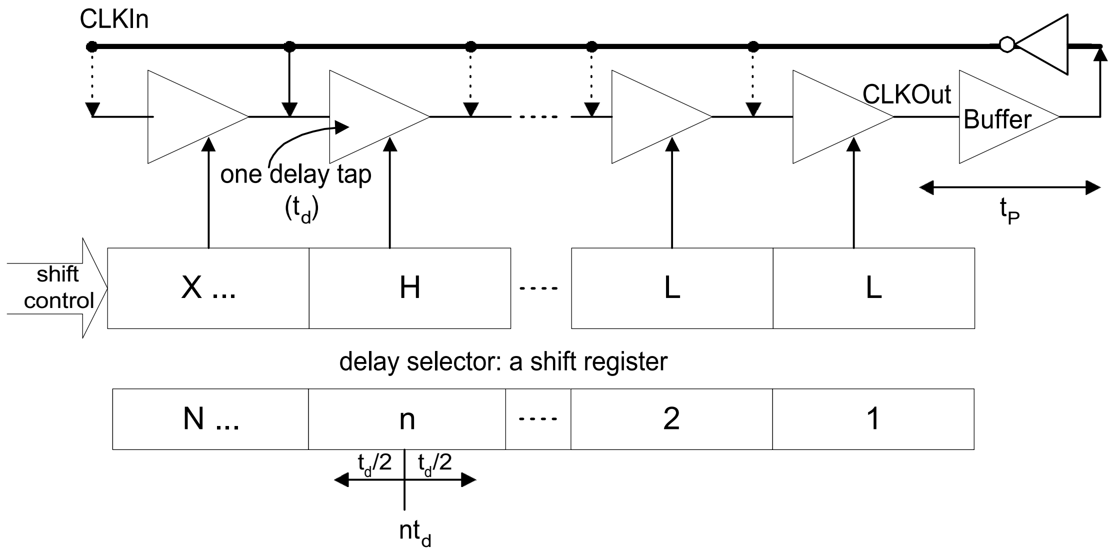


Figure 6.1. Single-loop register-controlled oscillator (RCO)

With this extra delay, Equation (6.1) to (6.3) can be rewritten to

$$f_{RCO} = \frac{1}{2 \cdot (n + \lambda) \cdot t_d} \tag{6.5}$$

$$\Delta f_n = \pm \frac{f_{RCO}}{2(n+\lambda) \pm 1} \tag{6.6}$$

$$\eta_f = \frac{\Delta f_n}{f_{ref}} = \frac{1}{2(n+\lambda)} \tag{6.7}$$

Assuming λ is 10, n is still large (40 in this case) to achieve 1% frequency granularity for single-loop architecture. In other words, it requires large extra delay (large λ) to achieve better frequency granularity at the cost of a limited maximum operating frequency of the single-loop RCO.

Another observation is the gain of this RCO is not linear according to Equation (6.1), if the gain is the slope of the curve shown in Figure 6.2 (set t_d= 200ps). The gain is given by

$$K_{RCO} = \frac{\partial f_{VCO}}{\partial n} = \frac{-1}{2t_d \cdot n^2} \tag{6.8}$$

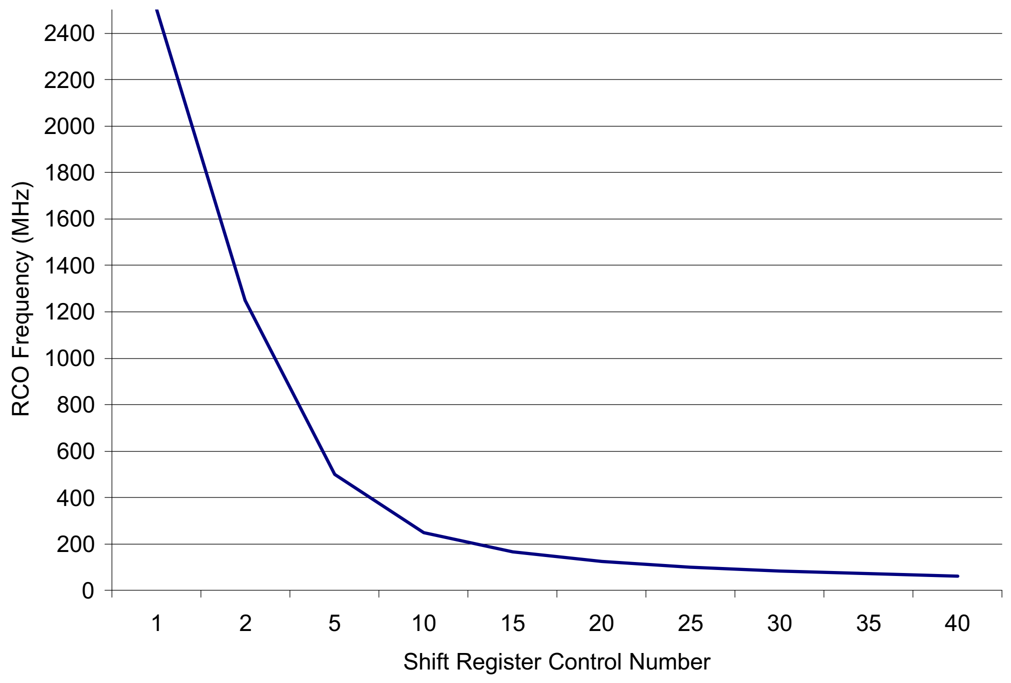


Figure 6.2. RCO frequency versus digital control number

With the extra delay t_P, the gain is inversely proportional to $(n+\lambda)^2$. The gain is much linear compared to Equation (6.8) and the RCO frequency versus control word is shown in Figure 6.3 with $\lambda = 10$, $t_d = 200$ps.

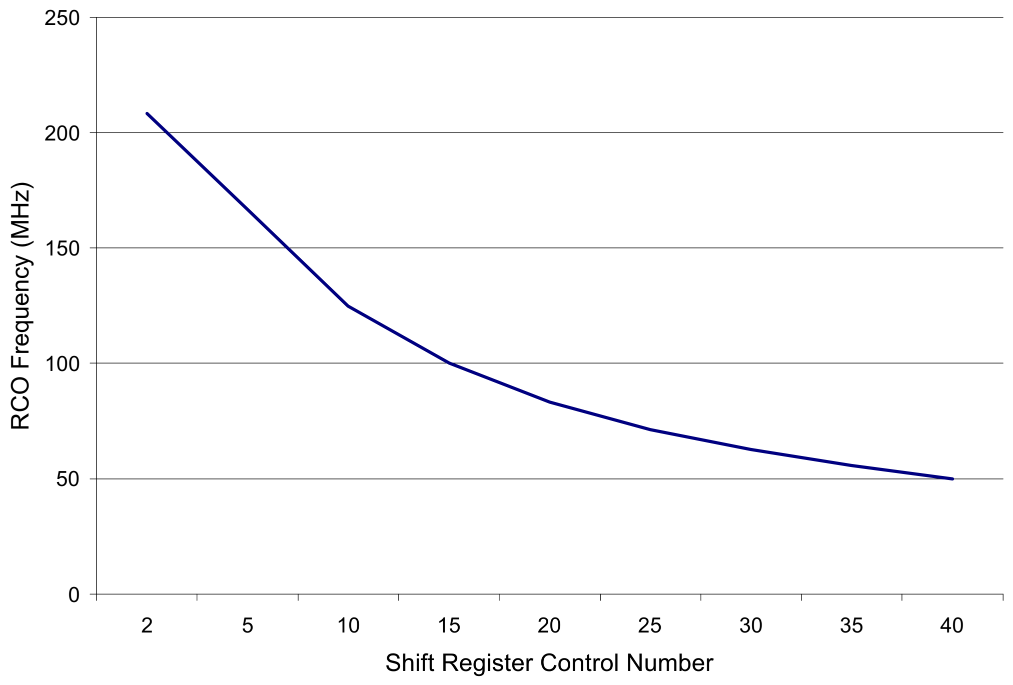


Figure 6.3. RCO frequency versus control word with extra delay

6.2.2. Two-Loop Architecture

In order to improve the frequency granularity and jitter performance of this RCO, a two-loop RCO is proposed and shown in Figure 6.4. A coarse loop is made up of the register-controlled delay line as discussed previous. Delay interpolation (adjusting loading) is used in the fine loop to interpolate a vernier delay within a coarse delay tap. Assuming the delay of a vernier tap in the fine loop is t_{vd} and effective number of delay taps is $2M$. Half of this fine delay line should cover the delay of $t_d/2$, where t_d is the delay per tap in the coarse delay line.

The control number used in this fine loop is equal to *m*, where $-M < m < M$. The oscillating frequency (f_{2RCO}) of this two-loop RCO is given by

$$f_{2RCO} = \frac{1}{2(nt_d + mt_{vd})} = \frac{1}{2t_{vd} \cdot (2nM + m)} \tag{6.9}$$

where $t_d = 2M \cdot t_{vd}$, and *n* is the control number of the coarse loop. Following the same procedure as used in the single-loop RCO, the reference frequency (f_{2ref}) is given by

$$f_{2ref} = \frac{1}{t_{vd}(4nM + 2m \pm 1)} \tag{6.10}$$

and the frequency difference when the two-loop RCO frequency is set by *n* for the coarse loop and *m* for the fine loop is

$$\Delta f_{n,m} = \pm \frac{1}{t_{vd}} \frac{1}{(4nM + 2m)(4nM + 2m \pm 1)} \tag{6.11}$$

The frequency granularity is determined by the ratio of $\Delta f_{n,m}$ and f_{2ref}.

$$\eta_{2f} = \frac{\Delta f_{n,m}}{f_{2ref}} = \frac{1}{4nM + 2m} \tag{6.12}$$

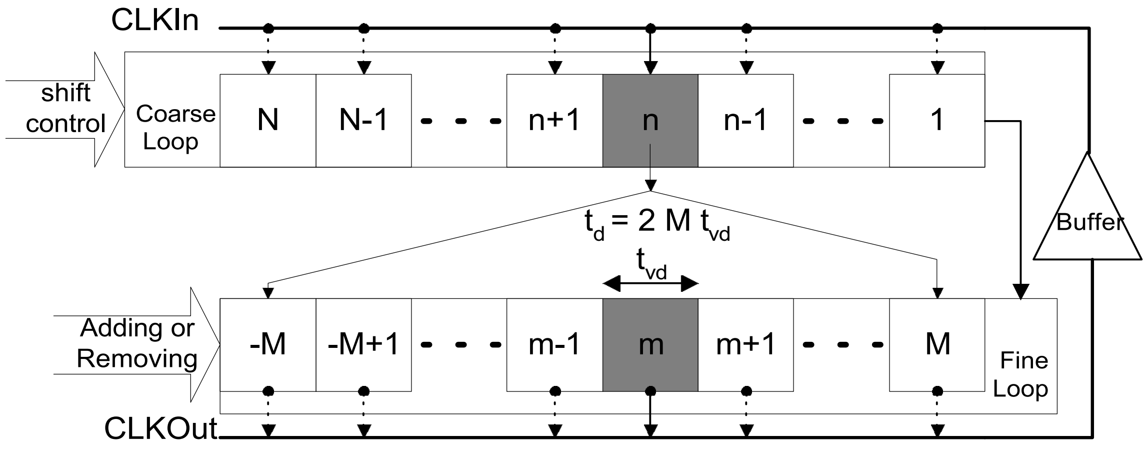


Figure 6.4. Two-loop register-controlled oscillator

When the extra delay (t_P) is considered, n' can be used instead of n in the above equations with the relationship of $n' = n + \lambda$, where λ is the factor relating the delay through the feedback and common input delays to the delay per tap, t_d. If fast and slow paths delay interpolation is applied, a total of $2M\,t_n$ (t_n is the nominal delay per stage of the fast path, and $t_n + t_{vd}$ for the slow path) will be added and the maximum oscillating frequency is limited because of this. For the third method of delay interpolation (adjusting driving, Chapter 4), a decoder is needed to provide the control word for the binary weighted inverters. Parallel-inverter matrix will also increase the loading for the previous stage. In this design, a register-controlled capacitor is used for delay interpolation and the fine loop.

Right now, two design parameters, n and M, can be set to achieve proper frequency granularity and lock range. For example, from Equation (6.12), to get $\eta_{2f} \le 1\%$, if n is chosen to reach a maximum available frequency ($n = 1$, $\lambda = 10$), and $m = M$, then the value of M can be calculated as $4nM + 2m = 100 \Rightarrow M > 2$ and $t_{vd} = t_d/4$. For $t_d = 200$ps, the maximum frequency $f_{2RCO,\,max} \approx 230$MHz, compared to the single-loop RCO, where under the same situation, $\lambda = 10$, $n = 40$ to ensure $\eta_f \le 1\%$, and $f_{RCO,\,max} \approx 50$MHz.

The frequency range of the two-loop RCO is given by

$$\frac{f_d}{2N + 2\lambda + 1} \le f_{2RCO} \le \frac{f_d}{2\lambda + 1} \qquad (6.13)$$

where $f_d = 1/t_d$, and the control number of the coarse loop varies from 1 to N-1. For a fixed entry point of the coarse loop (n is constant), the gain of two-loop RCO is given by taking the derivative of m, which is the effective number of the fine loop.

$$K_{2RCO} = \frac{-1}{t_{vd}(2nM + m)^2} \qquad (6.14)$$

The gain will be more linear compared to the single-loop RCO because of the $2nM$ in the denominator of Equation (6.14). With a two-loop architecture, the frequency granularity is taken care of by the fine loop and the coarse loop determines the operating range. Wide lock range and tight locking are possible with this configuration.

6.2.3. Circuit Design

Register-controlled delay line is the main part of the two-loop RCO. Symmetrical delay line discussed in Chapter 5 is used in the coarse loop, and register-controlled capacitor, examined in Chapter 4 is used in the fine loop. In order to achieve wide lock range, 48 ($N-1 = 48$) delay taps are used in the coarse loop and 16 ($M = 8$) fine taps are used in the fine loop. For a typical $t_d = 150$ps, t_{vd} is chosen to be greater than 10ps to get an effective tuning of 160ps for the delay interpolation.

A bi-direction shift register is used to control both delay lines. A master-slave D flip-flop with *Set* and *Reset* is used as the stored element for the shift register, shown in Figure 6.5. Instead of using an extra TG (transmission gate) at the feedback path for each latch, a feedback invert with resistive loading (long L device) in both sink and source paths can provide simple and stable operation for each latch. Passing transistors, *MN* and *MP* in Figure 6.5, are used for *Reset* and *Set* respectively. Since large number of shift bits is used in the RCO, such configuration can save area and still provide stable operation.

The outputs of the phase detector, *Shift-Right* and *Shift-Left* should be mutual exclusive (not occurring at the same time) to guarantee proper operations. Falling edges of *Shift-Right* and *Shift-Left* are used to clock the DFF. *Set-up* and *Hold* requirements of the DFF can be easily satisfied with this arrangement.

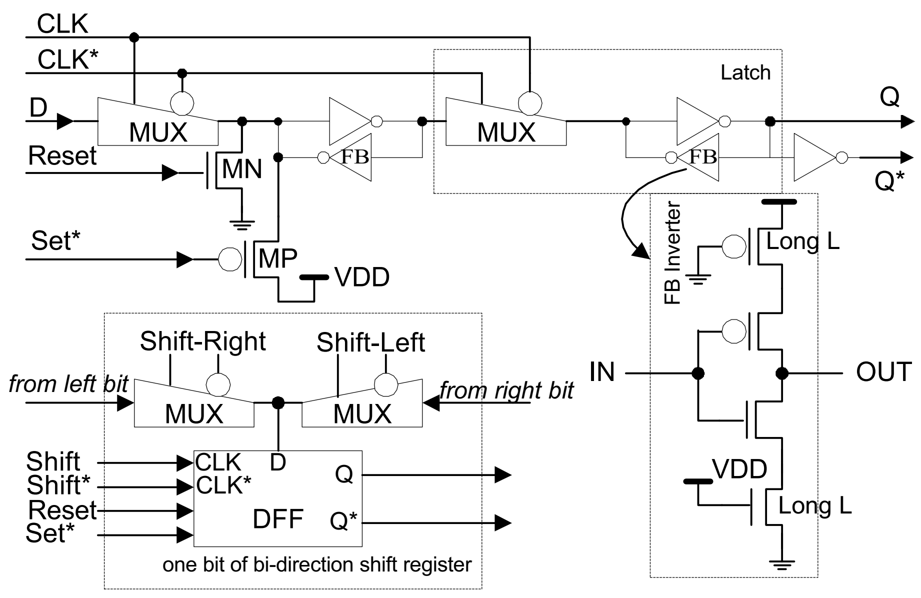


Figure 6.5. DFF, FB inverter and one bit of the bi-direction shift register

If the ADPLL is used primarily for high-speed applications, less delay stages have smaller λ (extra delay factor) and the number of stages can be determined using Equation (6.13). Small delay per tap of the fine delay line can provide better jitter performance at the cost of large number of taps and longer lock time. Layout of such delay lines is simple because of the regularity of the delay elements. Decoupling capacitors between the power buses can effectively limit the power supply and substrate noise.

6.3. Design the RCO-Based ADPLL

As shown in Figure 6.6, the RCO-based ADPLL consists with two loops to improve overall system resolution. When both loops acquire lock, the frequency granularity is small, as discussed above. Even for this small variation, a large phase error can exist because of jitter accumulation in the oscillator. A fine phase adjustment is proposed to make the ADPLL stay locked unless a large phase error is detected. Three features of this two-loop ADPLL are list below.

- *Coarse Loop*, which using register-controlled symmetrical delay line as described in Chapter 5 to provide coarse frequency selection and the resolution is within ±150ps.
- *Fine Loop*, which using register-controlled capacitors to give roughly 15ps/tap adjustment and interpolation. The resolution after the fine loop interpolation is within ±25ps and phase error < 50ps.
- *Fine Phase Adjustment* (FPA). Because of the nature of an oscillator, even a small variation of the RCO frequency will accumulate and cause a large phase error after cycles. A dynamic fine phase adjustment is proposed to compensate this and keep the phase error below 50ps. This fine phase adjustment can also suppress the phase noise caused by power supply noise, process and temperature variations.

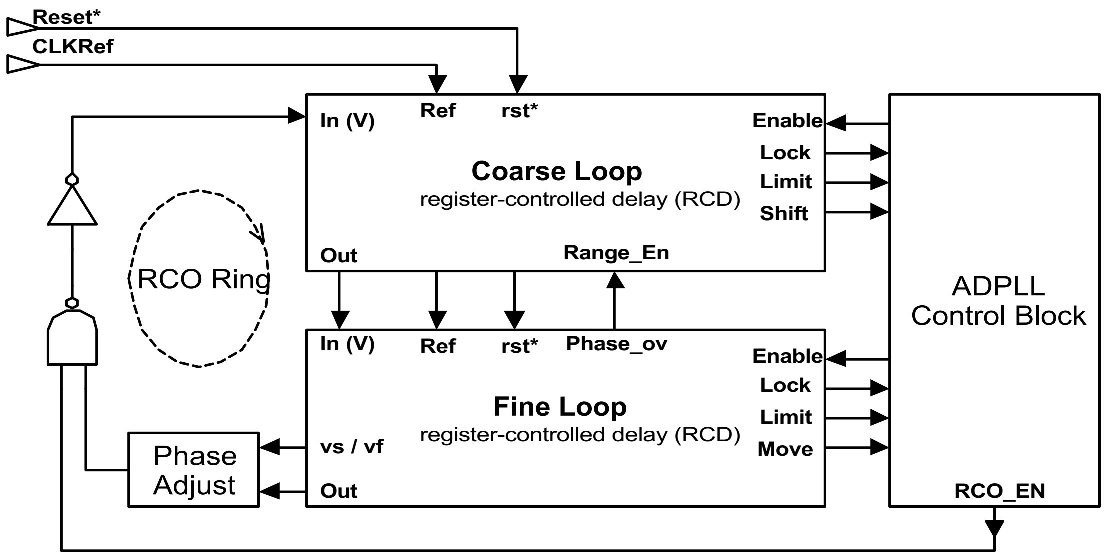


Figure 6.6. Block diagram of the two-loop RCO-based ADPLL

An ADPLL control block is used to select different loops, either coarse or fine. The typical VCO-based PLL uses frequency to adjust the phase. The traditional PLL is at least a second-order system, and the pull-in process is highly dependent on the loop parameters such as damping factor and nature frequency (Chapter 3). In order to acquire robust operation with this ADPLL along with fast lock-in process, an RCO enable signal is generated by the ADPLL control block to either enable or disable adjusting the RCO ring. With this control signal, zero initial phase error between the reference and RCO and fast lock can be achieved without bouncing back and force between the coarse and fine loops.

6.3.1. Hierarchy Pull-In

For the all-digital phase detector, no distinguish exists between large phase error and small one. If each time the delay shifts only one tap to tune the oscillating frequency, it may require a long time for the loop to acquire lock when a long delay line is used. The technique used to acquire fast lock is the hierarchy pull-in process, as illustrated in Figure 6.7.

Since a long delay line is used for wide lock range in the coarse loop, how to quickly and efficiently locate the inserting point in the RCD is important. In this design, a hierarchy configuration is used with a segment as the top level. One segment consists of several coarse delay taps. The coarse loop pull-in process is started with a segment selection. The initial entry point is reset to be in the middle of coarse delay line. After the proper segment is located, less shifts are needed for the coarse loop to find the proper coarse tap. When the coarse loop is locked, the control block passes the operation to the fine loop and there is no action in the coarse loop unless a large phase error is detected (such as a frequency-step or the limitation of the fine loop is reached). When the fine loop is locked, the shift registers for both loops are locked and any small phase error detected by the fine loop phase detector will start the fine phase adjustment to dynamically correct the phase error. A monitor block, a variable phase-error detector, watches for any large phase error (which can be adjusted according to different specifications) and resumes the two-loop correction if needed.

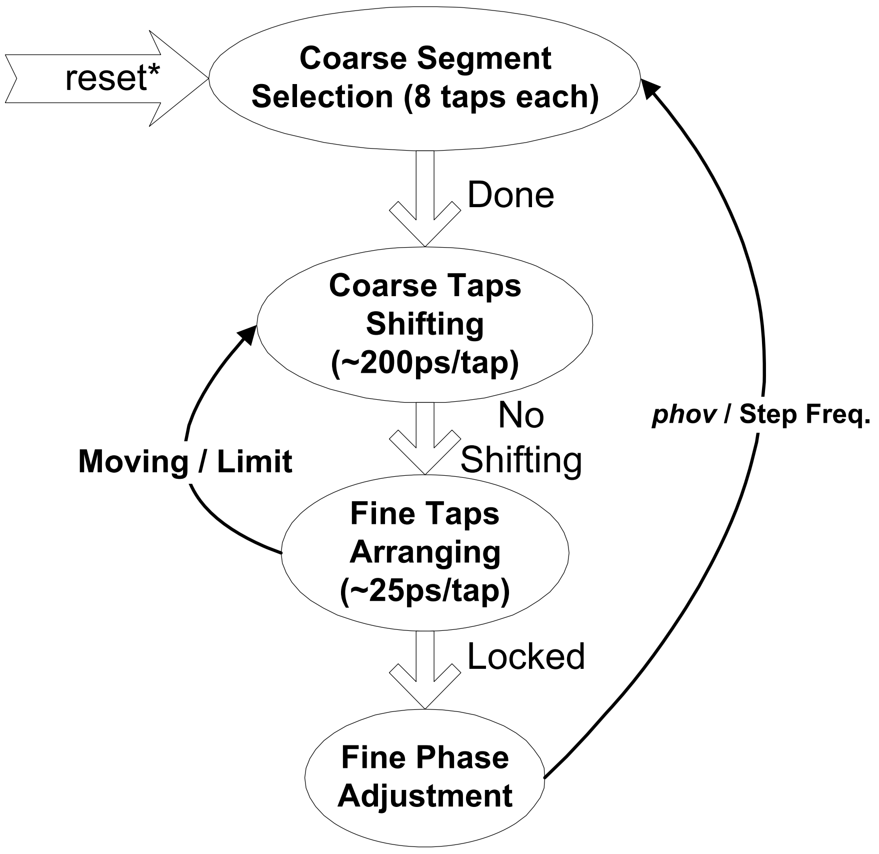

reset*
Coarse Segment Selection (8 taps each)
Done
Coarse Taps Shifting (~200ps/tap)
No Shifting
Moving / Limit
Fine Taps Arranging (~25ps/tap)
Locked
Fine Phase Adjustment
phov / Step Freq.

Figure 6.7. Block diagram of the Hierarchy Pull-in Process

The lock time of this hierarchy pull-in process can be analyzed as followed. Assuming the coarse delay line has *N* taps, with *F* segments and *L* taps per segment. This gives the relationship of

$$N = F \cdot L \tag{6.15}$$

For each shift from the phase detector, it needs *Y* clock cycles for the coarse to start another comparison. In the worst case, the time required to lock the coarse loop is given by

$$t_{cl,\max} = Y \cdot (L + \frac{F}{2}) \cdot T_{CLK} \tag{6.16}$$

where T_{CLK} is the reference clock period and $t_{cl,max}$ is the maximum lock time. The searching algorithm is illustrated in Figure 6.8. From the initial segment (generally in the middle of the delay line), the PD will determine which direction the segment goes, left or right. The segment entry point (*f*) can be located when the shift is bouncing back and force (*Shift-Right*, *-Left* and *-Right*). Then a proper tap is located within that segment when the limit of coarse delay is reached. At most *L* shifts are needed to lock the loop. The lock time will increase to *FL/(L+F/2)* times $t_{cl,max}$ if no searching algorithm is applied. For example, assuming $N = 48$, $F = 6$ and $L = 8$. Initially, the entry point is in the middle of the delay line. The worst-case lock time is $11T_c$ with hierarchy pull-in, compared to $24T_c$ without it, where $T_c = YT_{CLK}$. The lock time for the fine loop will be calculated after the locking mechanism is analyzed in the following sections.

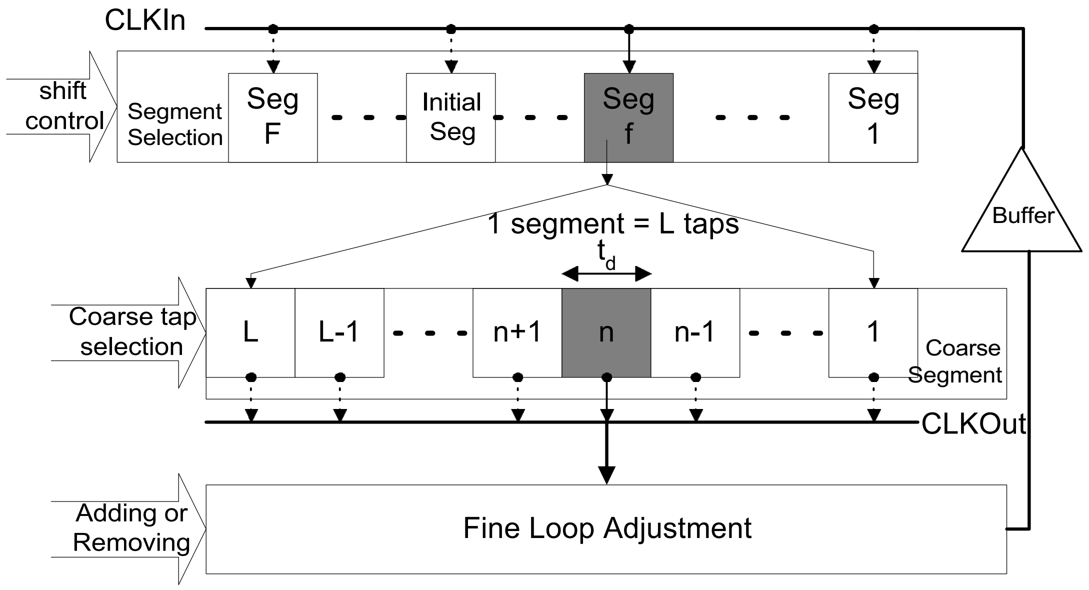


Figure 6.8. Searching algorithm used in the two-loop ADPLL

The simulation results of this hierarchy pull-in are shown in Figures 6.9, 6.10 and 6.11. In Figure 6.9, six segments are used for the coarse delay line, with 8 taps for each segment. Frequency granularity is worse when less delay taps are used. When the segment selection code is equal to 111111, the last segment is selected which gives frequency range from 270MHz to 540MHz. The gain is largest in this range. Eight different delay taps in this segment can smooth the slope and give approximately ±30MHz frequency variation when the RCO frequency is oscillating at about 480MHz, shown in Figure 6.10. For single-loop RCO, the frequency granularity is 6.25%, which is poor. With two-loop architecture, the delay interpolation will make the gain almost constant in a specific range, as shown in Figure 6.11. The initial frequency is 430MHz, correspond to the Tap 2 in Figure 6.10. The frequency variation after interpolation with a fine loop is about ±2.5MHz @ 455MHz, which gives $\eta_{2f} \approx$ 0.55%.

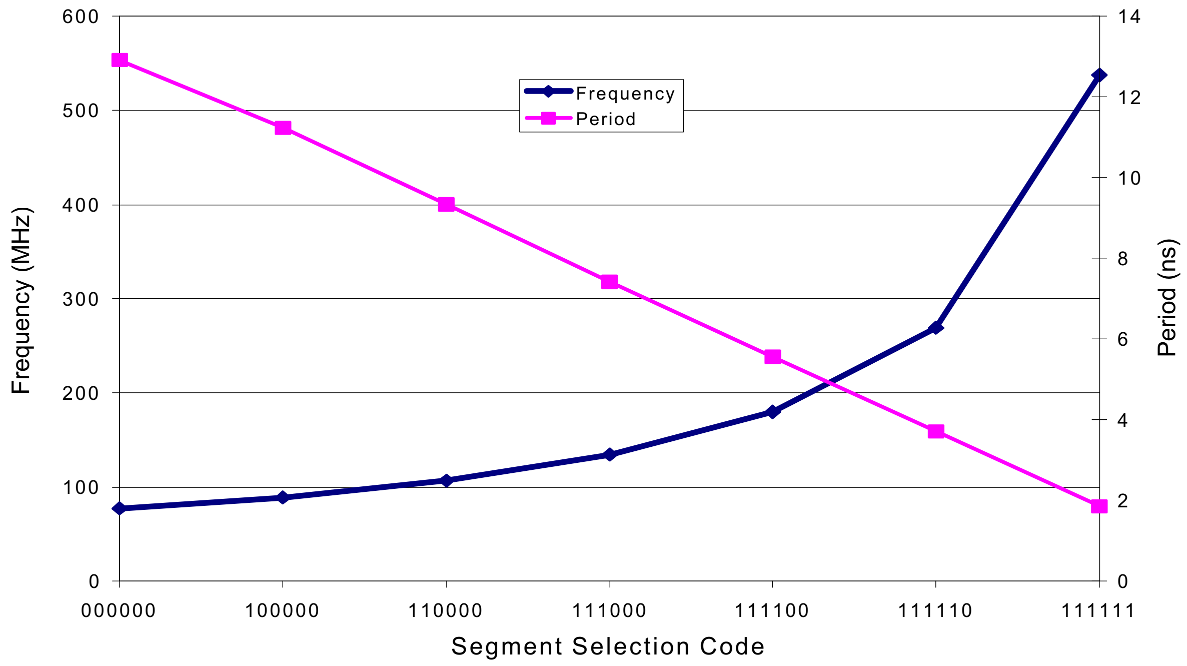


Figure 6.9. Frequency locking range of the coarse loop (segment selection)

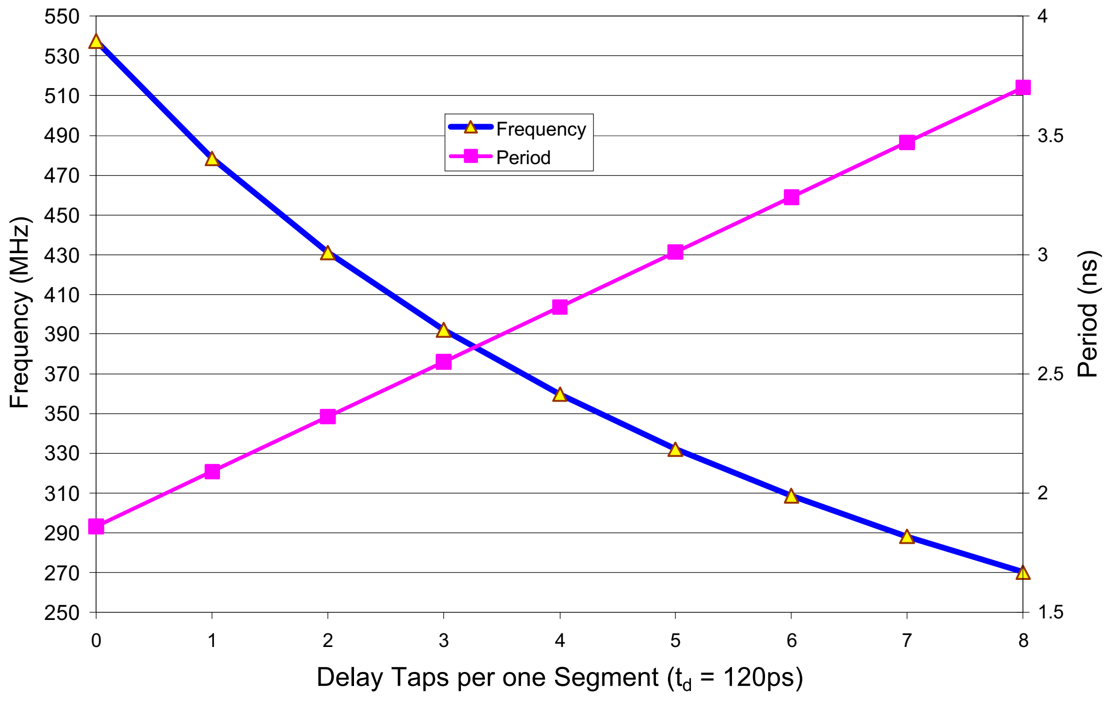


Figure 6.10. Shifting the delay taps in one segment of the coarse loop

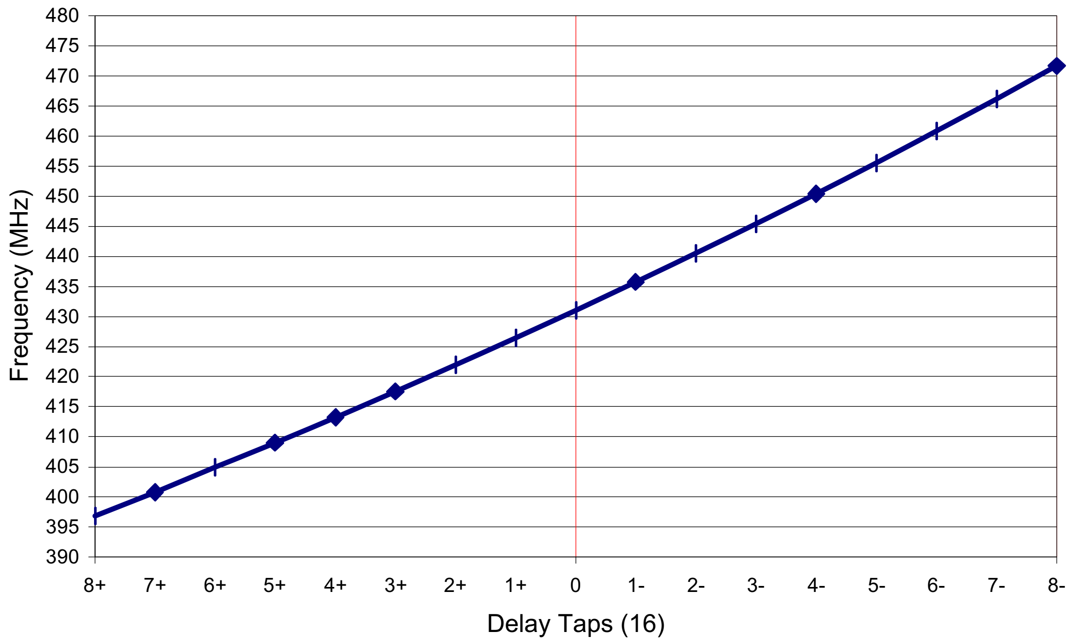


Figure 6.11. Enhance the ADPLL resolution – the fine loop adjustment

6.3.2. Coarse Loop

The block diagram of the coarse loop is shown in Figure 6.12. A coarse delay segment selector is used to locate the inserting segment of the coarse delay line. The segment selector is basically a 6-bit shift register (for $F = 6$ and $L = 8$). The 6-bit segment code (like a thermometer code, is a series of ones followed by a series of zeros) that can be used to set/reset the content of the 8 taps RCD and set the proper entry point. The segment code is initially set to 111000 and the entry point is in the middle of the delay line. A limit detector watches for a delay limitation in the RCD. If the frequency limitation is reached, a *Limit* signal is generated and no action will take place until the frequency is back in the proper range.

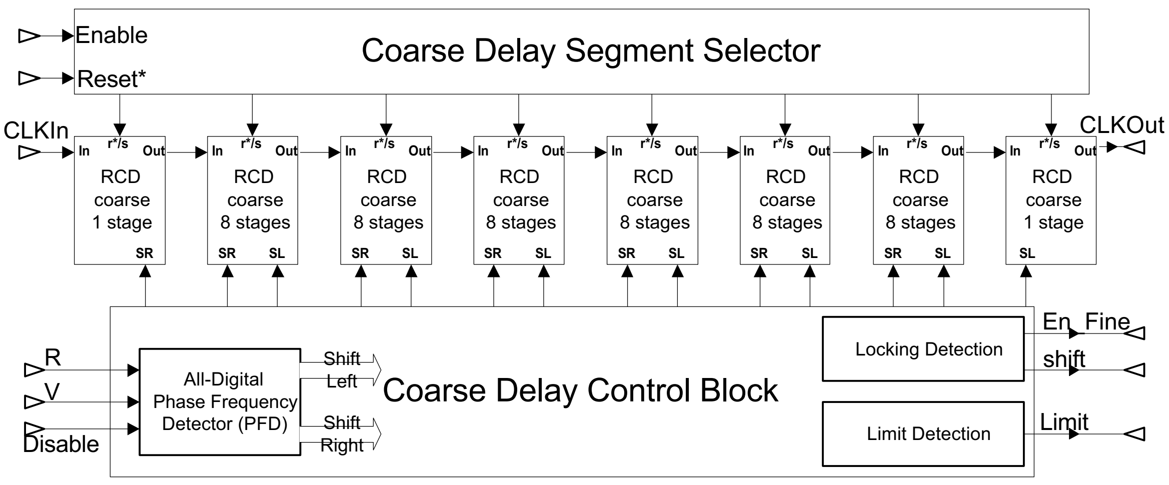


Figure 6.12. *Two-loop ADPLL:* the coarse loop

An all-digital phase frequency detector (ADPFD) is proposed to generate proper shift signals, which control the movement of the RCD (more on this later). Since the resolution of the coarse loop is within ±150ps, the ADPFD cannot be too sensitive to the small phase or frequency error. With the initial zero phase error, a simple 3-bit shift register is used to indicate whether the loop is locked or not. With initially setting to 000, the locking detector declares *Lock* and passes the control to the fine loop (*En_Fine*) after the content of the register is 111. Since the ADPFD makes decisions on the rising edges, the locking-detector can tolerant phase error within ±150ps, which is the minimum resolution of the coarse loop at worst cases. A bouncing detector is used to prevent bouncing back and forth in the delay line when the boundary is reached. In this case, the fine loop is activated even though the lock detector doesn't declare *Lock*.

The block diagram for the coarse loop control logic is illustrated in Figure 6.13. The operation starts at *Segment Enable*. The *Frequency and Lock Detection* will determine the operation flow, either *Segment Selection*, *Tap shifting* or *Limit Handling*. After the proper shifting, the *RCO_En* is asserted to start another comparison. The procedure will continue until the coarse loop is locked and *En_Fine* is generated to start fine loop operations. During this time, the coarse loop is in the idle condition, where no shifting signal is generated.

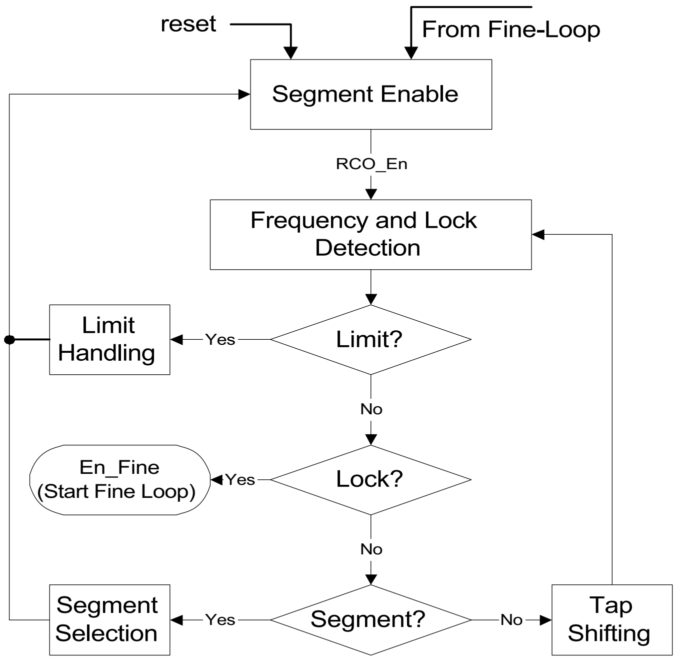


Figure 6.13. Block Diagram of the control logic of the coarse loop

6.3.3. All-Digital PFD

Since the ADPLL can be used as a frequency synthesizer, the phase detector in the coarse loop needs to provide both phase and frequency detection, shown in Figure 6.14. In a classic three-state phase frequency detector (PFD) the *Up/Down* signals drive the charge pump to create a proper control signal for the VCO. In the all-digital design, the narrow pulse generated by the PFD (when close to locking) is not suitable for the control signal of the shift registers. A pulse generator is inserted to convert the narrow width *Up/Down* signal into well-defined shift signals. Some features of this ADPFD are listed below.

- No distinction is made between large and small phase difference (no integration will occur as in the typical charge-pump PLL).
- When the phase difference is less than the minimum resolution of a coarse tap ($\pm t_d$), no shift signal is generated. With the third falling-edge of clock signals, if no shifting, the coarse loop declares locking and passes the control to the fine loop.
- The lock detect circuit is always enabled before the entire ADPLL is locked to watch for any large phase difference according to the results from the ADPFD.
- With zero initial phase error, the ADPFD acts like a frequency detector to pull in the frequency quickly for the coarse loop.

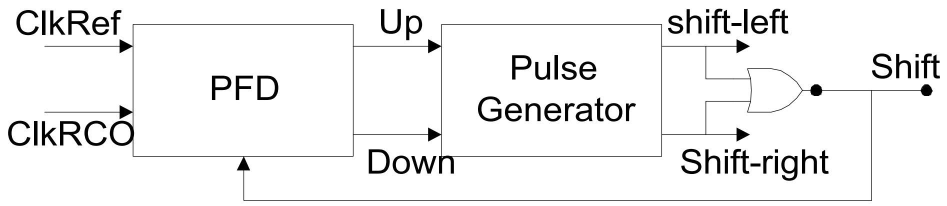


Figure 6.14. An all-digital phase frequency detector (ADPFD)

A zero initial phase error is implemented to acquire fast locking without cycle slips. A signal (*RCO_En*) is used to disable RCO when the register-controlled delay (RCD) line is shifting. The *RCO_En* is synchronized with the reference clock signal. A matched dummy delay is put on the reference signal path to get the zero initial phase error between the *CLKRef* and *CLKRCO*. The phase detector can make right decision every time using this scheme, and fast locking can be achieved. The operation of the ADPFD along with the zero initial phase error can be illustrated with Figure 6.15.

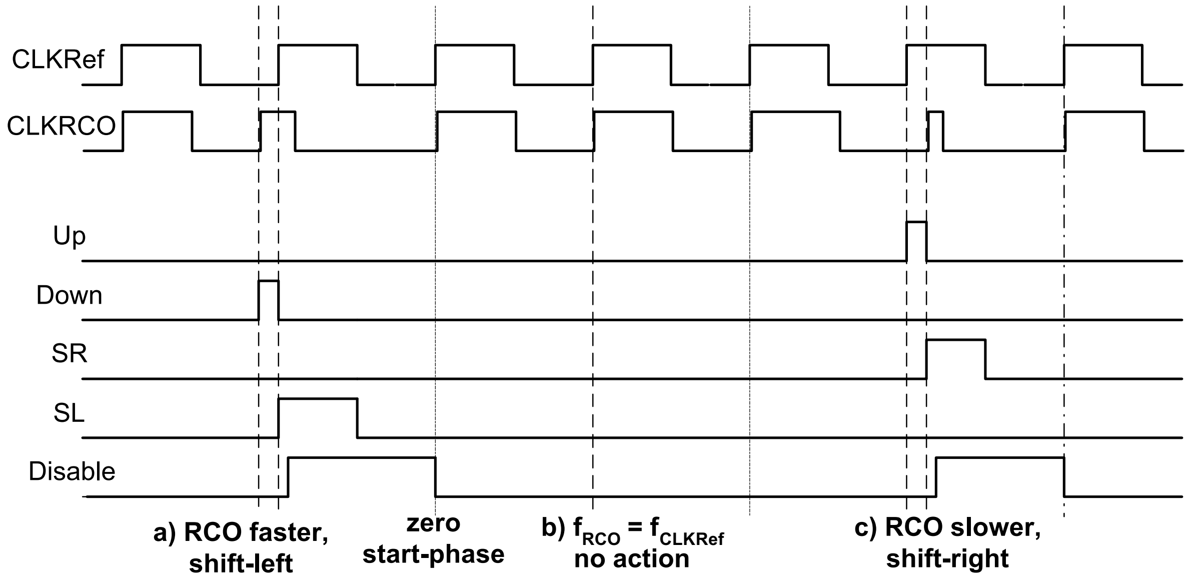

CLKRef
CLKRCO
Up
Down
SR
SL
Disable
a) RCO faster, shift-left
zero start-phase
b) $f_{RCO} = f_{CLKRef}$ no action
c) RCO slower, shift-right

Figure 6.15. The operations for the ADPFD and the coarse loop of the ADPLL

The ADPFD makes a decision at the following rising edges of *CLKRef* and *CLKRCO* since with zero phase error at the start point (*Disable* is LOW), no *Up/Down* signal will be generated at the first rising edges. In the case a), the RCO is faster compared to the reference, and the PFD asserts *Down* with the pulse generator of the ADPFD converting this signal into the *shift-left (SL)* signal. The *shift* control signal is active at the falling edges of *SR/SL*, while the RCO disable signal is asserted to ensure proper shifting of the delay line.

The RCO is re-enabled at the next rising edge of the reference clock, and starts another phase frequency comparison. In case b), the phase frequency differences are small with the condition of coarse delay taps, no shifting will take place even if some narrow *Up* and *Down* signals can still be generated by the PFD (which are filtered by the pulse generator). In this case, the coarse loop can declare it is locked and pass the control to the fine loop. Case c) is opposite to case a). The *shift-right* will be asserted and decrease the RCO frequency.

A modified PFD circuit in Figure 6.16 is used to detect frequency error for the coarse loop. Without dead-zone consideration for the all-digital PLL, a static phase error can be traded with no *Up* and *Down* signals simultaneously. A *Disable** is used to guarantee that no *Up/Down* at the first comparison edges (zero initial phase error).

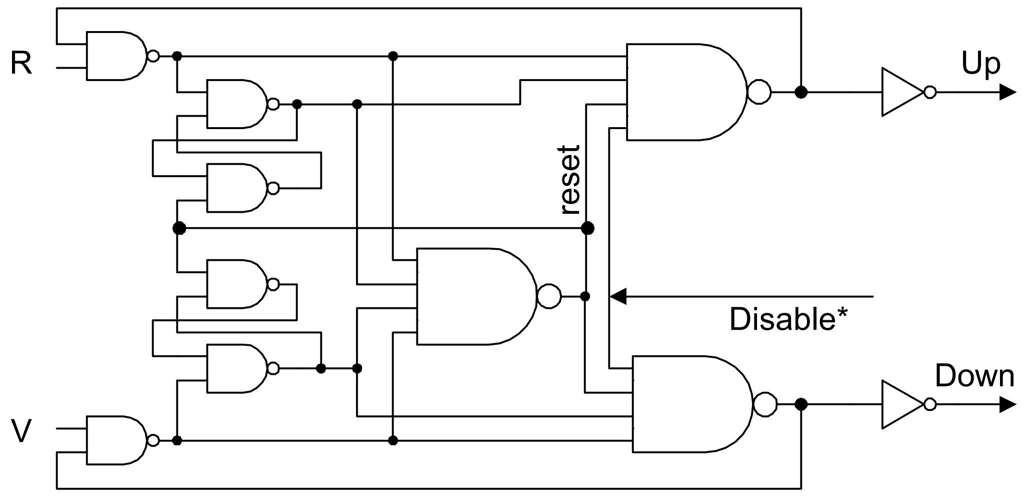


Figure 6.16. A modified phase frequency detector

A pulse generator is used to convert *Up/Down* from the PFD to well-defined shift signals. A simplified circuit diagram is shown in Figure 6.17. Two DFFs are used to catch the *Up/Down* respectively. The *R* and *V* are delay version of the PFD inputs.

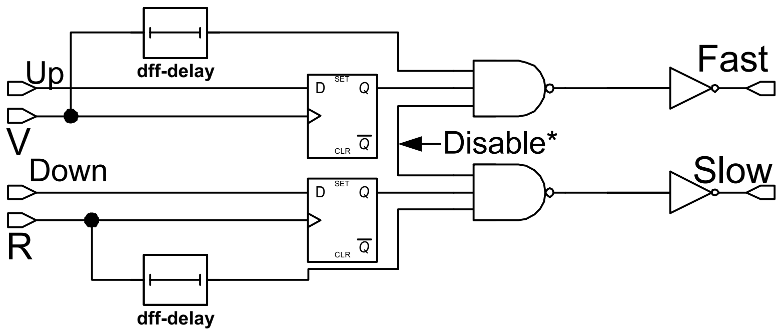


Figure 6.17. Circuit diagram of the pulse generator

For the coarse loop, the delay can be one coarse delay, which is used to track the variations. If the pulse width (*Up/Down*) is less than one coarse delay, it will be filtered out by the pulse generator. Notice that *V* is used to clock *Up*, and *R* is used to clock *Down*. The output of the DFF is reshaped to form the proper *Fast / Slow* signal, which in the coarse loop, the *shift-right* or *-left.*

6.3.4. Fine Loop

Figure 6.18 shows how the fine loop is used in the two-loop ADPLL. Sixteen fine delay taps, which are controlled by a 16-bit shift register, are used to interpolate a small delay to improve the resolution of the overall performance.

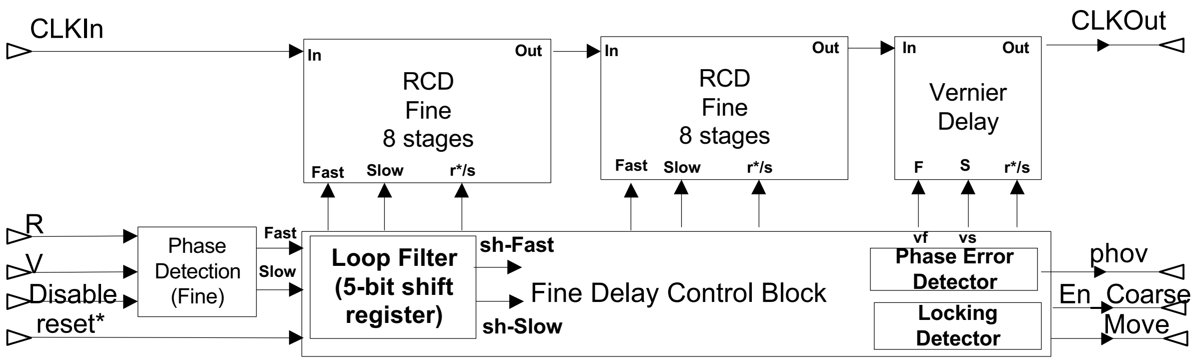


Figure 6.18. *Two-loop architecture:* the fine loop

The register-controlled capacitor (RCC) is used for the fine delay line. There are two seg-ments used in the fine loop, and the movement is interleaved between the two segments, shown in Figure 6.19. The interleaved structure can offer better balance between the two segments. Each fine delay segment has 8 taps, and initially four of them are ON, the others OFF. Each segment can offer at most a 4-tap adjustment in either way (add / remove) according to the shift signals. One inverter at the either end can provide gain and isolation between the two segments. The slope of transitions will not be degraded because of this arrangement. The RCC can offer 10-20ps (t_{vd}) resolution under worst-case process and temperature corners, which will change the oscillating period by 20-40ps.

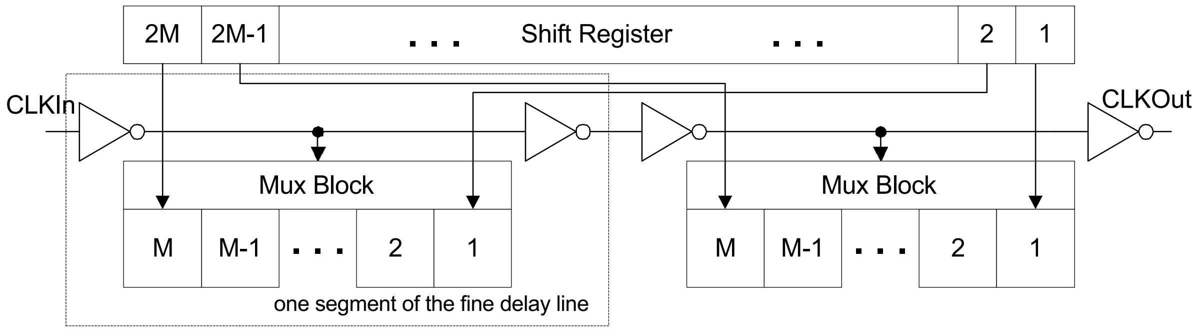


Figure 6.19. Interleaved fine delay line with control blocks

A dedicated phase detector is used to discriminate less than 50ps phase difference. A shift-register type digital loop filter is used to stabilize the shift operation. A fine phase adjustment is proposed to handle jitter circulation in the RCO and compensate for small varia-tions caused by power supply, substrate noise, process and temperature.

The control logic of this fine loop is shown in Figure 6.20. When the fine loop is activated by *En_Fine* signal from the coarse loop, the *Phase and Lock Detection* will indicate the next operation of the fine loop, either *Arranging Taps* (adding or removing), or starting

Fine Phase Adjustment when *Lock* is declared. An H-bit counter is used as digital loop filter to stabilize the opera-tion. If large phase error is detected or after adding or removing one delay tap, the fine loop will pass the control back to the coarse loop. In this case, a large phase error caused by the frequency stepping can be quickly corrected by the coarse loop.

To detect a small phase error for the fine loop, a two-way arbiter, Figure 6.21, is used as a part of the fine PD to perform phase detection and generate the mutually exclusive *Up/Down* signals. The RS-latch will provide good phase discrimination, down to 10ps. The output of this arbiter is like a cross-coupled inverter, where this configuration can guarantee that *Down* and *Up* will not be asserted simultaneously.

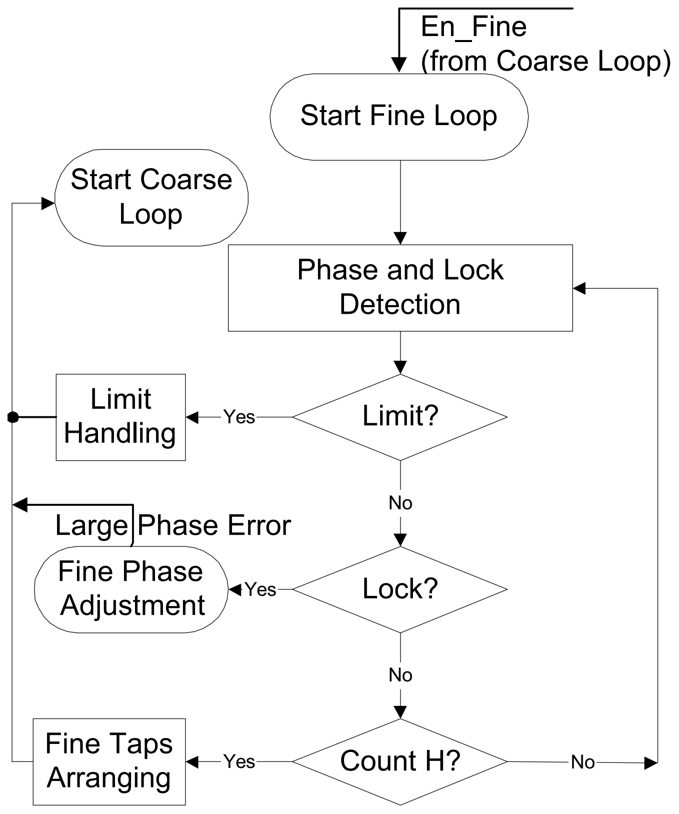


Figure 6.20. Block diagram of the fine loop operations

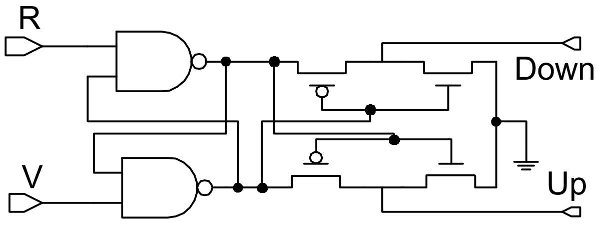


Figure 6.21. The two-way arbiter as part of the fine PD

Not like the PFD in which narrow pulses are generated when the loop is close to lock, the outputs of this arbiter have well-defined logic level. Another pulse generator is used to set the proper timing and filter out any decision which is made when the phase error is less than 50ps. The timing diagram of the fine PD is shown in Figure 6.22. The *R* AND *V* with delay is used to clock the pulse generator (rising edges) and *Fast* is the output of the pulse generator (Down is filtered out and *Slow* is inactive). The net delay is determined by the phase error specification, in this design, less than 50ps. Without such delay, this PD may be too sensitive to small phase jitter, which is unavoidable in the digital implementation.

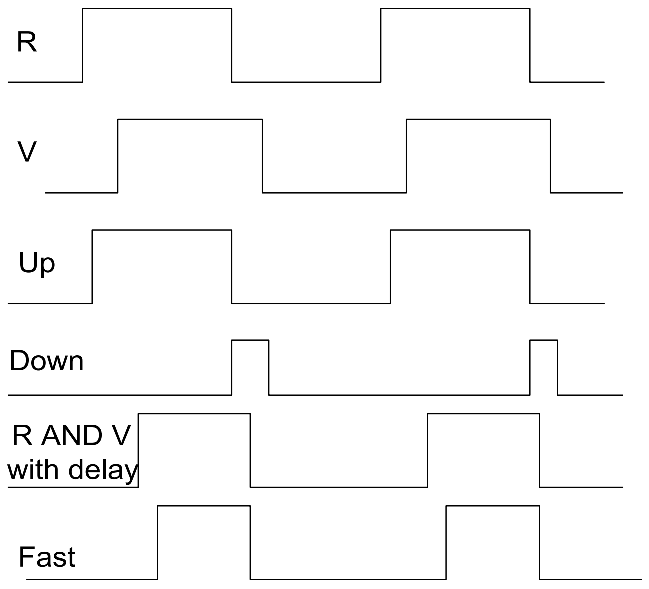


Figure 6.22. Operations of the fine PD

A variable phase error detector (VPD), in Figure 6.23, is used to detect large phase errors where “large” is defined by the delay element used. Since the fine loop can only correct for less than $\pm t_d$ (delay of one coarse tap) timing error, the *phov** generated by the VPD can indicate that a step in frequency or large phase error is occurring and pass the control back to the coarse loop. The *S / F*, coming from the fine PD, is used to select the leading signal, either *R / V*. After the specified delay (e.g. 300ps), this selected signal is used to clock the two DFFs to catch the difference. If both outputs of DFF are HIGH or LOW, no large phase error exists; otherwise, the VPD declares *phov*(active LOW)*. The retimed clock is inverted to provide a harmonic detection (activated at the falling edges).

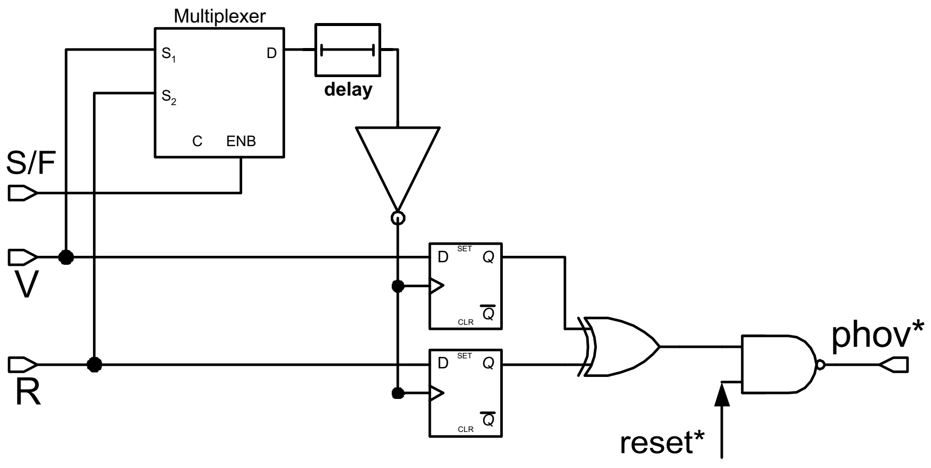


Figure 6.23. A variable phase error detector

6.3.5. Locking Mechanism in the Fine Loop

Unlike the coarse loop in which the frequency difference may be large initially, the fine loop is used to interpolate vernier delay to finely adjust the frequency and pull in the phase. The fine PD is mainly used to detect a phase difference. A 5-bit shift register, as a digital loop filter, is used to filter out any occasionally generated pulses, and stabilize the loop operation at the cost of longer lock time. When a large phase error is detected because of jitter accumulation, a proper *Move* signal is activated for the fine loop. In this way, a fast lock for the fine loop is still possible even though a 5-bit shift register is used. If no delay adjusting for two consecutive cycles after the fine loop is activated, the fine loop declares *Lock* and enters the *Fine Phase Adjustment*. The lock mechanism for the fine loop is shown in Figure 6.24. The *Fast / Slow* is generated by the fine PD according to the phase difference between the reference (*CLKRef*) and RCO output (*CLKRCO*). *Sh_Fast* and *Sh_Slow* are control signals after the 5-bit shift register. If large phase error is detected, *phov** (active LOW) is asserted to generate *Sh_Fast/Slow* regardless what the content of 5-bit shift register is.

In the worst case, after the coarse loop, the RCO clock period is t_d away from the reference clock period. Total of *M* delay taps need to be added or removed to interpolate and lock the fine loop. Since each fine loop comparison starts with coarse loop operation, three more clock cycles are added to calculate the lock time of the fine loop. Since the lock-in process of the fine loop is not linear, the average number of clock cycles for each fine phase comparison is equal to *MOD*(*H*/2), where *H* is the length of the digital loop filter, and *MOD* is a function in which *MOD*(5/2) returns 3. The lock time of the fine loop can be approximated as

$$t_{fl,\max} = M \cdot [3 + MOD(\frac{H}{2})] \cdot T_{CLK} \qquad (6.17)$$

For $M = 8$ and $H = 5$, the maximum lock time for the fine loop is approximately $48T_{CLK}$.

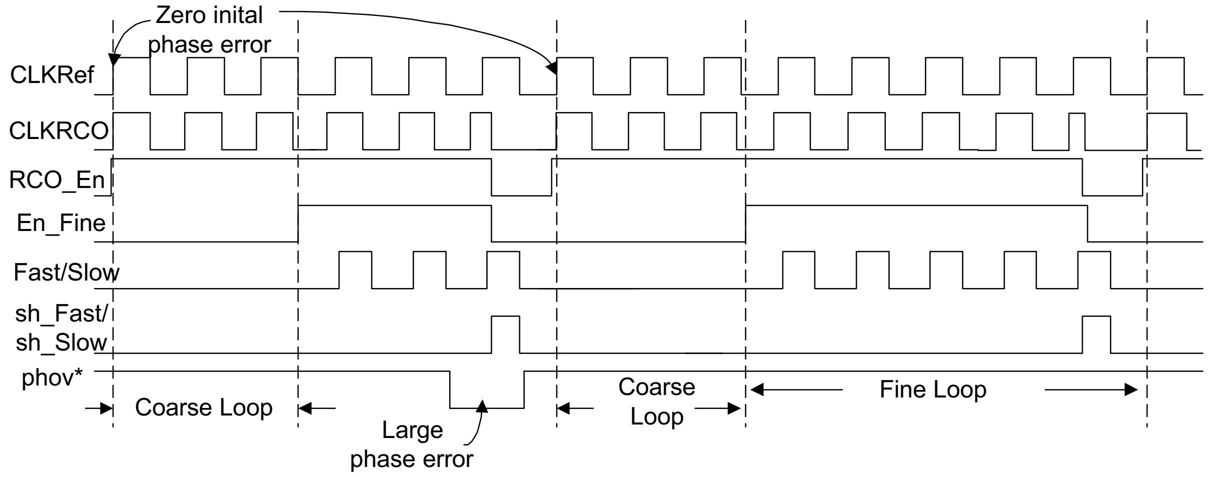


Figure 6.24. Locking characteristics of the fine loop

When the fine loop is locked, the phase error is small (generally less than 50ps). Such tiny error will accumulate in the RCO and result in large phase errors after several cycles. In order to stabilize the two-loop ADPLL operation, a fine phase adjustment using vernier delay elements is proposed to continuously adjust the phase and keep the phase error below 50ps. The operation of such a fine phase adjustment is shown in Figure 6.25. When no *Slow/Fast* is generated in two cycles, *phase_en* is asserted to indicate the loop is locked. In the worst case, the period difference between the RCO and reference is less than 25ps, which is the resolution of the fine delay tap. This small variation will accumulate quickly in the RCO ring and cause a large phase error, which may lead to the loop unlocking. Two vernier delay elements, which are made up of the same fine delay elements (different size), are used to perform the fine phase adjustment according to the *Slow/Fast* coming from the fine PD. Initially, one delay tap is ON and the other is OFF (here, ON / OFF means connect / disconnect from the RCO ring respectively). When *Slow* is asserted, the *vs* is activated to turn both taps ON. When *Fast* is asserted, the *vf* is activated to turn both taps OFF. If the PD indicates no move, the two vernier delay taps are back to the initial condition. This process will continue until a large phase error is detected, or *sh-Slow/Fast* is generated. The fine phase adjustment will pass the control back to the coarse loop to correct for a large phase error.

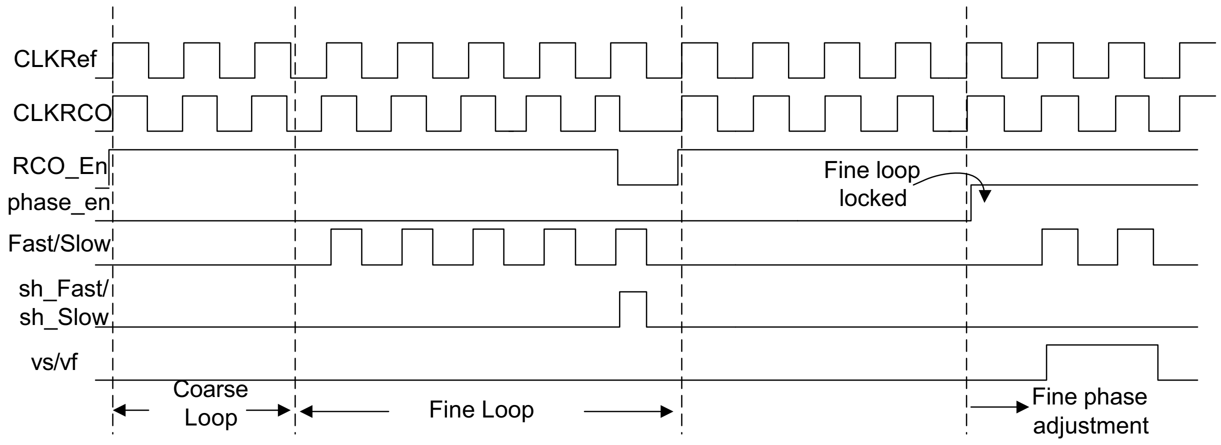


Figure 6.25. Locking Mechanism: the fine phase adjustment

6.4. Loop Dynamics of the RCO-Based ADPLL

Since the pull-in process of the RCO-based ADPLL is nonlinear, it's challenging to analyze the loop dynamics quantitatively. With this all-digital implementation, the lock characteristics are mainly determined by the frequency granularity and locking mechanism. In this section, the responses to phase and frequency steps are examined qualitatively.

6.4.1. Step-in Phase Response

Assuming the ADPLL is locked before applying a step in phase ($\Delta\phi$) in the reference clock, the ADPLL is in the fine phase adjustment state. For small $\Delta\phi$, the two vernier delay taps may be capable of correcting the phase error within five consecutive *Fast/Slow* pulses. If not, the 5-bit shift register will generate *sh_Fast/Slow* and terminate the fine phase adjustment. A normal hierarchy pull-in process is ignited to correct such a phase error. If $\Delta\phi$ is large, such a phase step can be captured by the variable phase detector (used to detect large phase error) and generate *phov** to bring the loop back into the hierarchy pull-in process. With the zero initial phase error the phase frequency detection is started. Such a phase step will disappear and the loop should fall in lock quickly. Generally, the coarse loop will not be affected by such a phase step. The fine loop may settle within five cycles after the reference is stable. The fine phase adjustment is resumed to stabilize the operation.

If the loop is in a transient state when the phase step is applied, such perturbation will have little effect on the hierarchy pull-in process because the *RCO_En* signal synchronizes the phase frequency comparison each time to the reference clock. If $\Delta\phi$ occurs after the initial rising edges within the coarse loop, the ADPFD may make a wrong decision. The error can be corrected in the following comparisons. In the fine loop operation, the variable phase detector will catch such variation and pass the control to the coarse loop.

Therefore, for the step-in phase response, it can be concluded that no static phase error will exist and no anomalous behavior will be seen in this two-loop RCO-based ADPLL.

6.4.2. Step-in Frequency Response

When the frequency is stepping during the pull-in process, the coarse loop will act first to correct the phase/frequency error if suitable, then the fine loop will start the normal hierarchy pull-in process. Since the tuning range of the fine loop is narrow (only $\pm t_d$ for the oscillating period), it's necessary to use the coarse loop first to correct for large frequency differences. Compared to the fine loop using a digital loop filter, the pull-in process for the coarse loop is much faster with segment selection. It's even possible to correct a large frequency step (only limited by the lock range of the coarse loop) with no overshoot or saturation occurring as in the analog PLL. The pull-in time can be estimated using Equation (6.16), and the lock-in time for the fine loop can be evaluated using Equation (6.17).

If the frequency is stepping during the fine phase adjustment (both loops are locked), the *phov** generated by the variable phase detector, or *Move* (either *sh_Fast* or *sh_Slow*) signal will terminate the locking condition and push the loop back into the pull-in process. Simulated timing waveforms shown in the next page illustrate the step frequency response of this ADPLL. First, the ADPLL tried to lock at 400MHz. In the first 65ns, the coarse loop acquired lock with the segment selection from 10ns to 45ns and tap shifting between 50ns to 60ns. A *Phov** was generated at 85ns, which indicated a large phase error occurring. The fine loop acquired lock around 120ns, when the *Phase_en* was asserted. The fine phase adjustment was proceeded after that. The ADPLL locks within $50T_{CLK}$, phase error less than 50ps, and frequency granularity is below 0.5%.

The frequency stepped at 300ns from 400MHz to 200MHz. The *phov** generated by the VPD at 320ns terminated the fine phase adjustment (*phase_en* went LOW) and pushed the loop back to the hierarchy pull-in process. The two *Shift_Left*s selected different segment of the coarse loop, and the following *Shift_Right*s tuned the frequency by shifting the delay taps. At 350ns, the coarse loop locked again and the fine loop comparison began at 400ns. *Phov** asserted at 415ns to indicate a large phase error. The fine loop acquired lock at 490ns, and the fine phase adjustment started to finely tune the phase according to output from the fine PD. From the timing diagram, the ADPLL reacquired lock after 190ns. The dynamic response took 38 clock cycles for both loops in the locking condition.

All the simulation results illustrate in this chapter uses 0.18μm CMOS technology. From the simulation diagram of the step-in frequency response, the proposed ADPLL has a unique locking mechanism: hierarchy pull-in with fine phase adjustment. Fast locking can be achieved even with a large frequency step.

6.4.3. Limit Handling

In a traditional PLL when the input frequency reaches the oscillator's range, the gain will drop dramatically and the oscillator may cease to work. In the RCO, for the fine loop, it can be designed to cover one coarse delay tap or more under various conditions. If a limitation is reached, because of the step-in phase or noise, it is simple to reset the PLL for another operation. Since the coarse loop essentially determines the operating range, it requires more attention. For the register-controlled delay line, the MSB (most significant bit, N-1) and LSB (least significant bit, 1) of the N-bit bi-direction shift register determine the limit of the coarse loop and lock range, referring to Equation (6.13). A limit detector is used to watch these two bits to change. When the RCO range is reached, a *Limit* signal is generated by the limit detector and the RCO frequency is locked without any change unless the frequency is back in the proper range. During this period, no shift signal is allowed and the shift register is locked without any movement to save the power. The simulation results for this limit handling are shown in Figure 6.26 with a reference clock at 70MHz. When the input reference steps from 70MHz to 100MHz, the ADPLL resumes operations and pulls in the loops accordingly in Figure 6.27. In Figure 6.26 and 6.27, *CV* is the RCO signal, and *CR* is the reference. *S_Right* and *_Left* are shift signals generated by the ADPFD. *OSC_En* is the

RCO enable signal, and *Limit_* (active LOW) is the signal which indicates that RCO boundary is reached.

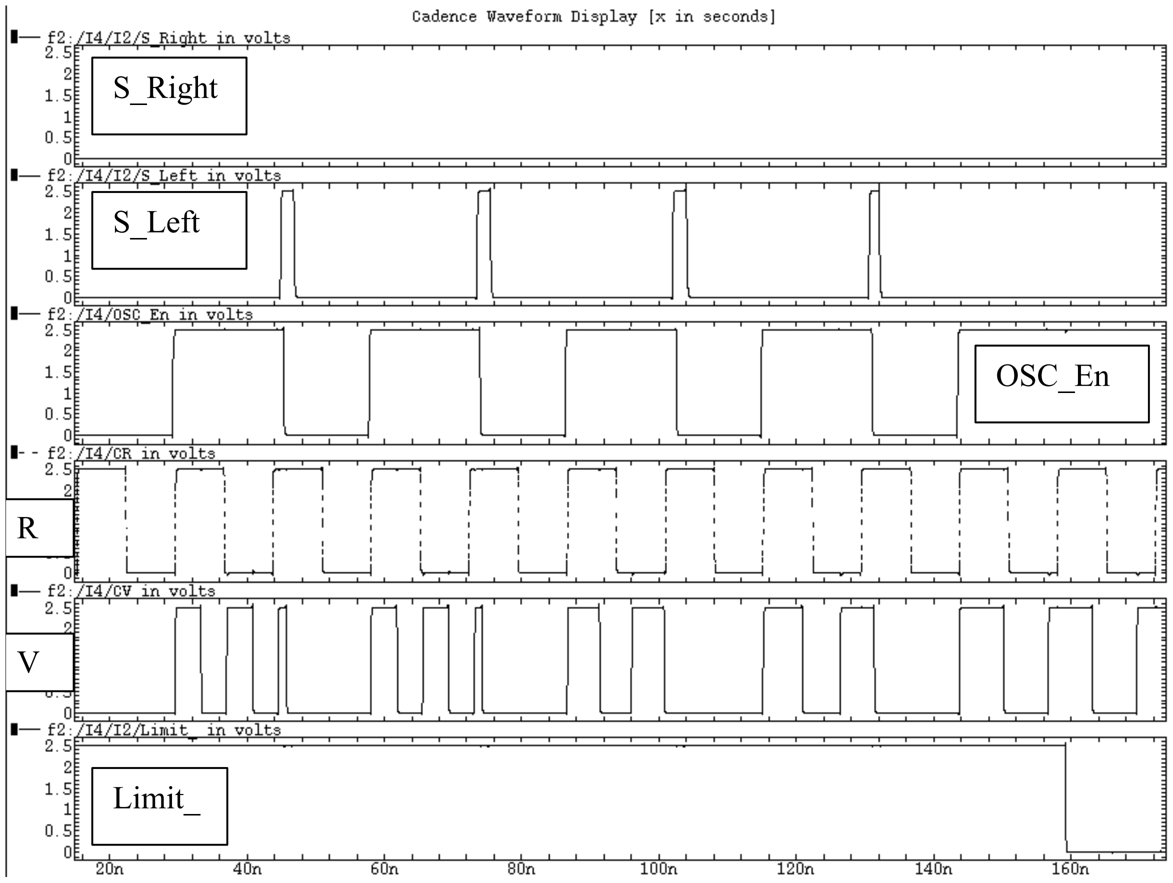


Figure 6.26. Simulation results of the limit handling

After the 4-consecutive *S_Left* (shift left for the segment selector) pulse in Figure 6.26, the *Limit_* was asserted at about 160ns to indicate that the limitation of the coarse delay line is reached and no further action could take place. The delay entry point was at N^{th} tap, which was the most left position of the coarse delay line. The RCO continued oscillating at this frequency until the reference frequency was back in the proper range, shown in Figure 6.27. *Limit_* went back to HIGH at 200ns when the frequency stepped to 100MHz. A typical hierarchy pull-in process took place and the ADPLL locked at about 490ns (*phase_en* went HIGH).

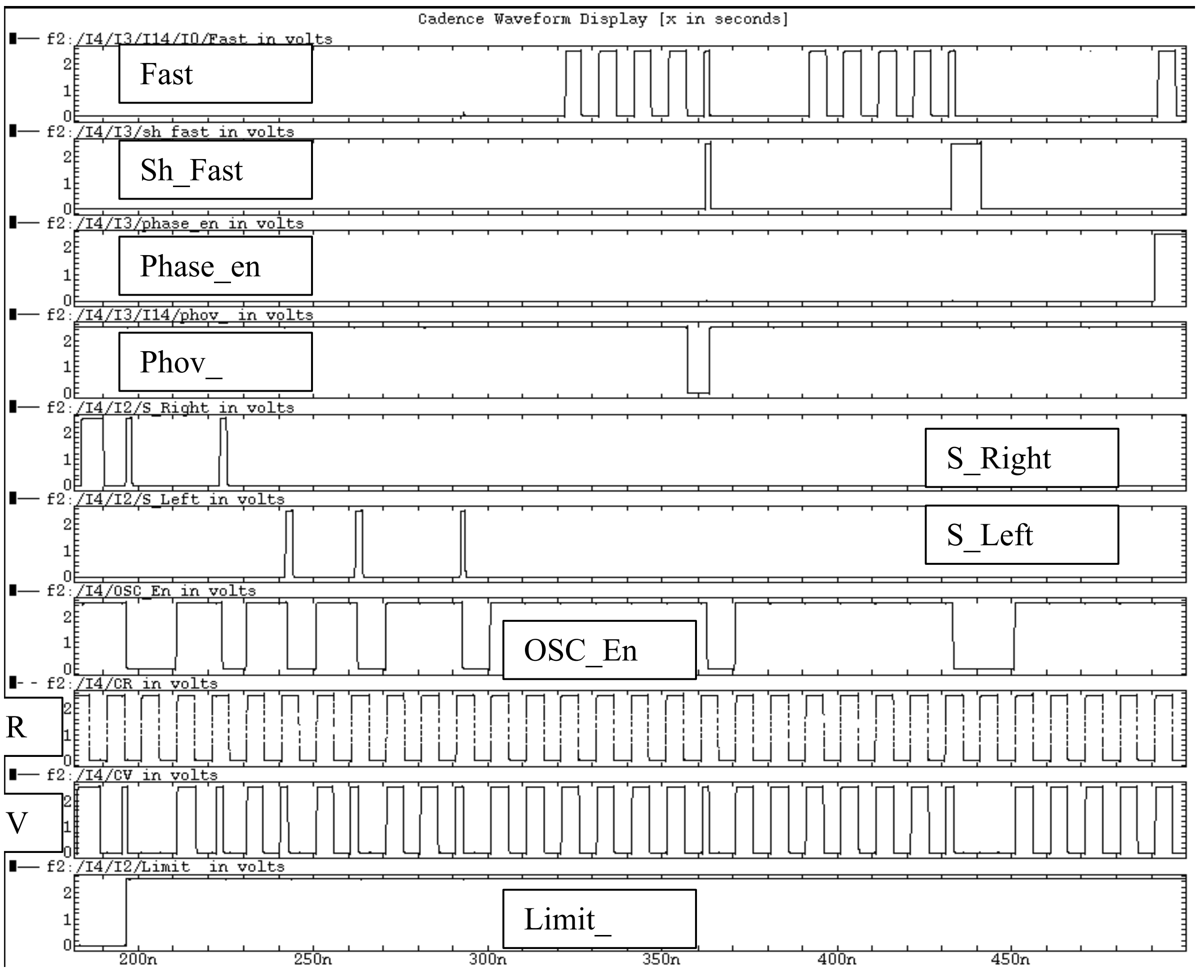
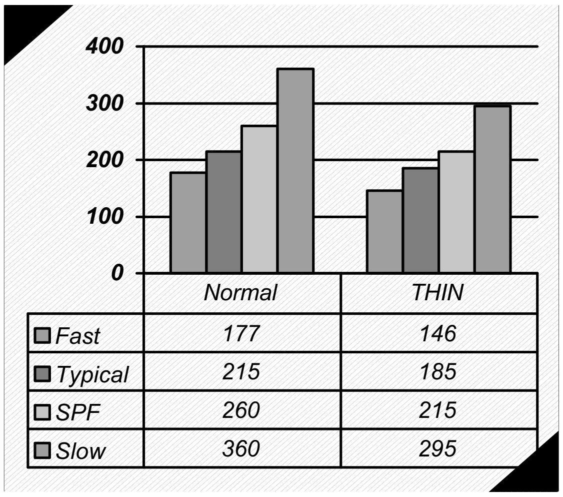


Figure 6.27. Simulation results for the limit handling back to the normal operation

6.4.4. Locking Characteristics with Variations

In the previous discussion and derivations, the coarse delay per tap (t_d) is assumed to be constant. In fact, process, voltage and temperature (PVT) variations may make this delay variable. Another factor is the parasitic capacitance with each delay element (parasitic resistance is also taken into concern if long routs are used). A process technology, which using thin gate oxide, can effectively decrease the threshold voltage of the MOSFETs and decrease the delay per stage of the two NAND gates. The simulation results under different concerns are shown in Figure 6.28.

- Typical: 2.5V, TT, 85C, w/o PRC
- SPF: 2.5V, TT, 85C, w/ PRC
- Fast: 2.75V, FF, 0C, w/ PRC
- Slow: 2.25V, SS, 110C, w/ PRC
- Normal gates versus THIN gates (PRC: parasitic resistance and capacitance; TT: typical NMOS and typical PMOS; FF is the fast corner and SS the slow corner.)



	Normal	THIN
▨ Fast	177	146
■ Typical	215	185
□ SPF	260	215
▨ Slow	360	295

Figure 6.28. Coarse delay per tap ($2t_d$ in ps) under different conditions

To design the two-loop RCO, such variations need to be considered. For the coarse loop, a t_d' can be used instead of t_d to calculate the operating range with the following relationship.

$$t_d' = \alpha \cdot t_d \tag{6.18}$$

where α is a process dependent parameter, and t_d is the delay per tap under typical conditions. For 0.18μm CMOS technology, α is in the range of 0.823 ~ 1.674. It's a good idea to form a single RCO and simulate its operation under different conditions to determine the value of α. The lock range of the coarse loop can be estimated by

$$\frac{f_d}{\alpha_{\min}(2N+2\lambda+1)} \le f_{2RCO} \le \frac{f_d}{\alpha_{\max}(2\lambda+1)} \tag{6.19}$$

where $f_d = 1/\,t_d$, α_{min} and α_{max} are the minimum and maximum values of α respectively, N is the number of coarse delay taps, and λ is the factor related to the extra delay. Table 6.1

shows the relationship between the RCO lock range and different parameters. For large N, λ is big because of the extra driving needed for a long delay line. For high-speed applications, the N should be small. The ADPFD should also be able to track such variations and provide correct decisions.

PVT FACTOR (α)	DELAY PER TAP (t_d, ps)	DELAY FACTOR (λ)	NUMBER OF DELAY TAPS (N)	MIN. RCO FREQUENCY (MHz)	MAX. RCO FREQUENCY (MHz)
0.8	96	5	24	**176.5**	947
		10	48	**89**	496
1	120	5	24	141	758
		10	48	71	397
1.6	192	5	24	88	**473**
		10	48	44.5	**248**

Table 6.1. RCO lock range versus different parameters

For the fine loop, the delay per tap (t_{vd}) also changes with PVT variations. Since the fine delay line should cover at least $\pm t_d$ to ensure proper interpolation, with variations, the relationship can be expressed as

$$t_d = \frac{\beta}{\alpha} \cdot 2M \cdot t_{vd} \tag{6.20}$$

where β is the factor related to the variations, and $2M$ is the length of the fine delay line. Generally, the fine delay is less sensitive to variations than the coarse delay tap, which results in $\beta < \alpha$. For the fast and slow paths interpolation, β is about equal to 1, which infers a better tracking. In this case, M and t_{vd} should be chosen to cover the worst case situation (α is maximum) and jitter performance is degraded. For the register-controlled capacitor, if t_{vd} is made too small, large M is required which may lead to sensitivity to noise. When designing the delay interpolations, tradeoffs have to be made to accommodate frequency granularity, jitter, lock time, area, and power dissipation.

6.5. Experimental Results

The RCO-based ADPLL was fabricated in a 0.18-μm 4-poly double metal CMOS technology. Figure 6.29 shows the chip layout and Figure 6.30 is the microphotograph of the actual die. A 48-stage variable coarse delay line is used with an overall tuning range of 65MHz to 385MHz. A 16-tap fine delay line is put on the left side of the test scribe to provide fine-tuning. The ADPLL control logic, along with the phase detectors and fine phase adjustment circuit, are laid out in the middle. Since the size of the test chip is limited by the dimension of the memory scribe, a longer delay is expected because of the interconnection (λ factor in Equation 6.5). The total size of the test circuit, including the bond pads (power supply, test pins etc.), is 3300 x 90 μm^2.

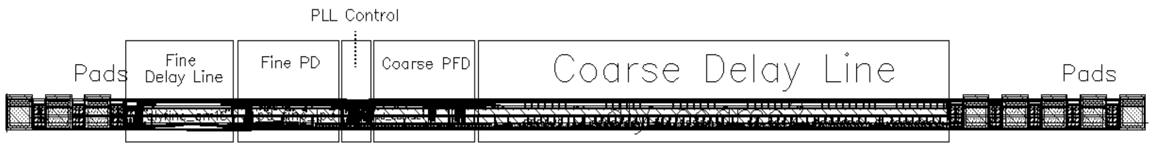


Figure 6.29. ADPLL chip layout

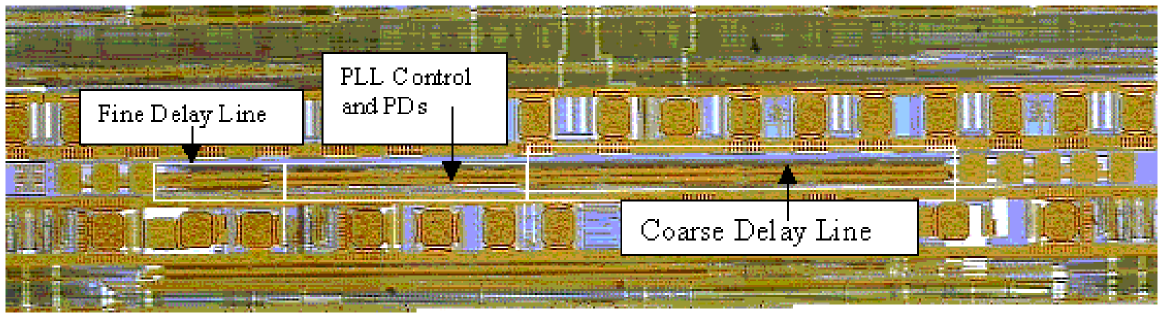


Figure 6.30. Microphotograph of the RCO-base ADPLL

A Tektronix TDS 784D four-channel digital oscilloscope was used to capture the signals. The reference clock was generated by Hewlett-Packard 8131A 500MHz Pulse Generator. Figure 6.31 shows the resulting RMS jitter when the input reference frequency is 200MHz and VCC=2.5V. One sigma of jitter (cycle –to-cycle jitter) is 30ps. The peak-to-peak jitter is about 210ps. When the clock frequency was jumped to 333MHz, the measured RMS jitter is about 25ps and 150ps peak-to-peak. At lower frequency, 100MHz, the measured RMS jitter is about 60ps, and 300ps peak-to-peak. The jitter performance is degraded when a large number of delay taps are used. Table 6.2 summarizes the measurement results

.

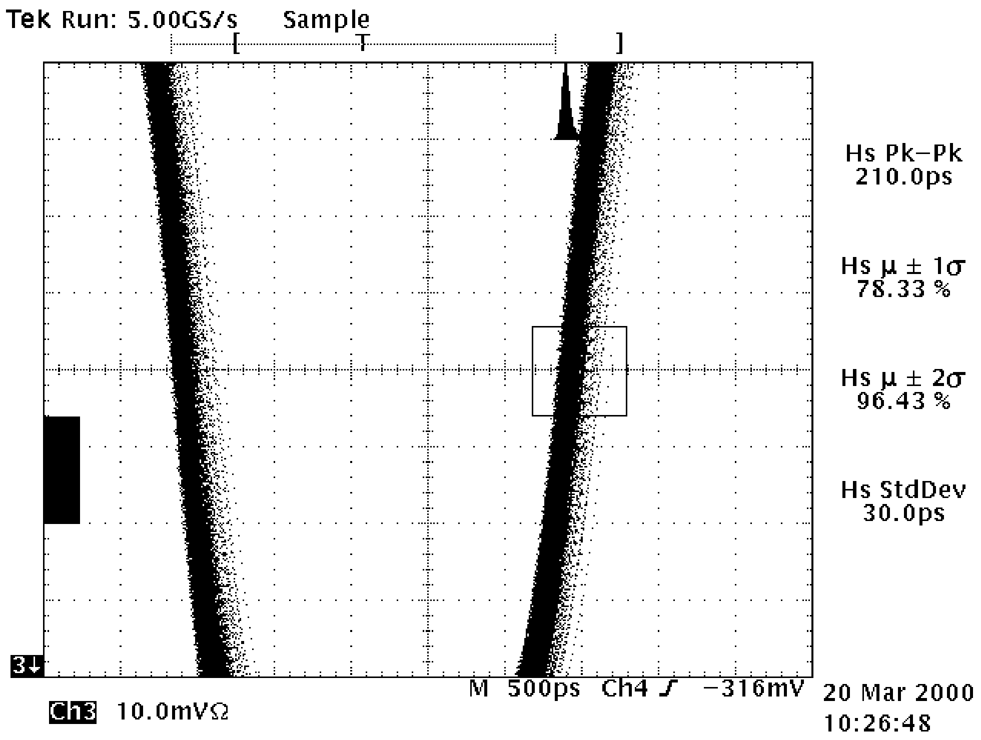


Figure 6.31. Measured clock jitter (@200MHz, VCC=2.5V, room temperature)

The RCO output signal and internal reference are shown in Figure 6.32, where the output was traced by the channel 3 (the upper curve). The lower curve is the internal reference clock at 200MHz. Both signals were triggered by the external clock (from the channel 4). The duty cycle distortion is less than 2% at this frequency.

Process Technology	0.18 µm, 4-poly, 2-metal CMOS	
Chip Size	3300 x 90 μm^2	
Transistor Count	4200	
Lock Range	65MHz – 385MHz	VCC=2.5V, 25C
Lock Time	Less than 80 cycles	
Supply Voltage Range	1.4V – 3.3V	
RMS Δ Jitter	30ps @200MHz	VCC=2.5V, 25C
Peak-to-Peak Jitter	~ 210ps @200MHz	VCC=2.5V, 25C
Duty Cycle Distortion	Less than 2% @ 200MHz	VCC=2.5V, 25C
Power Dissipation	5.9mW @ 200MHz	VCC=2.5V, 25C

Table 6.2. Summary of main features of the RCO-based ADPLL

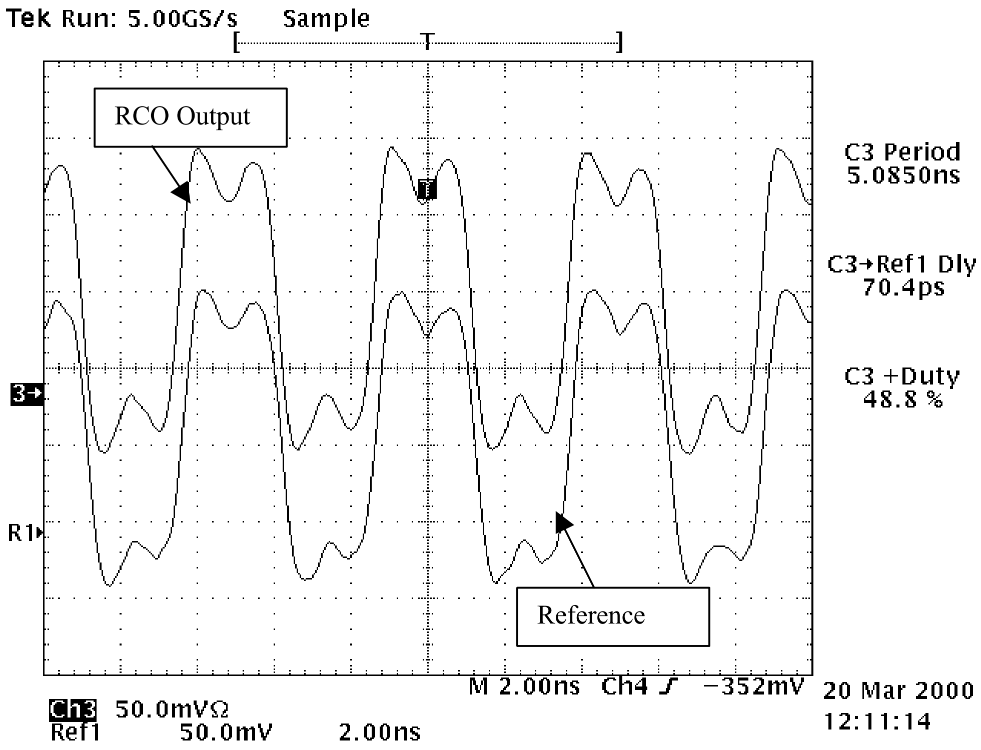


Figure 6.32. Measured results when the reference clock is at 200MHz and VCC=2.5V

Wide supply voltage range is another feature of this all-digital implementation. Measured RCO output clock period and frequency are illustrated in Figure 6.33 when the supply voltage is swept from 1.4V to 3.3V. The period reflects the lower end of the lock range, where most of the coarse delay taps are involved (the worst-case situation). Figure 6.34 and 6.35 show the measured output clock signals when VCC is equal to 3.3V and 1.4V respectively. The femto probe used to capture the signal has 20:1 attenuation, which gives the right scale for the power supply voltage. Figure 6.36 demonstrates that the generated clock signal has low duty-cycle distortion when reference clock is at 100MHz, which results from the equal propagation delay for the delay taps used in the RCO.

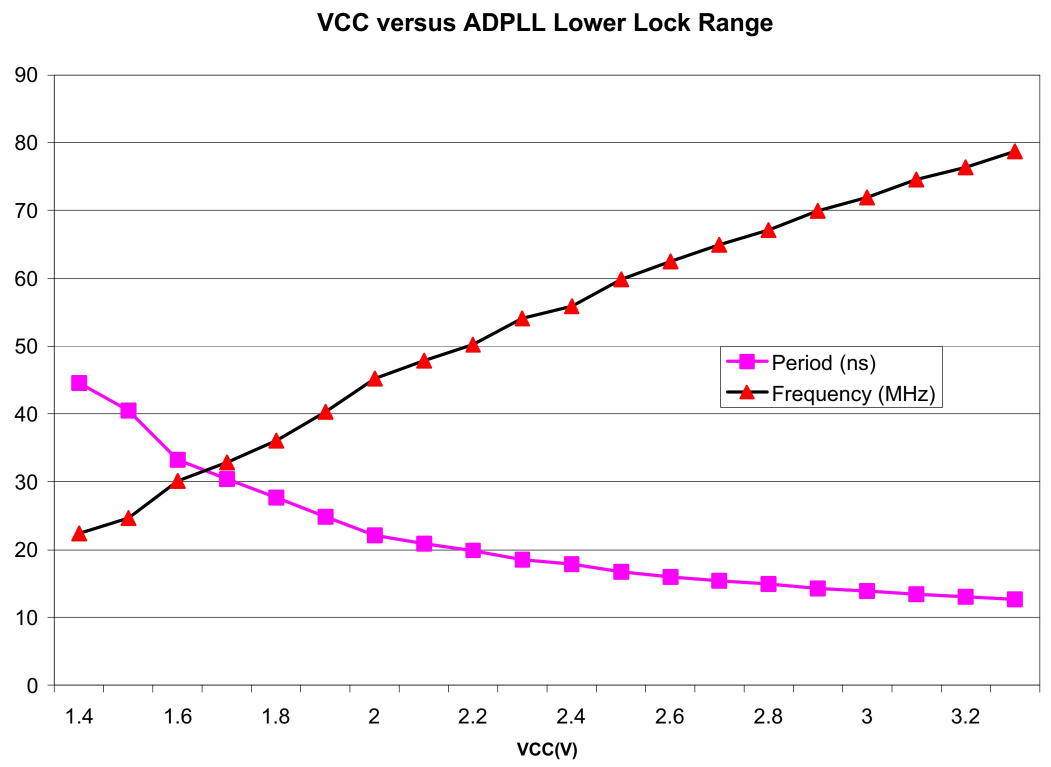


Figure 6.33. Measured output clock period and frequency (lower end) when VCC is swept from 1.4V to 3.3V (room temperature)

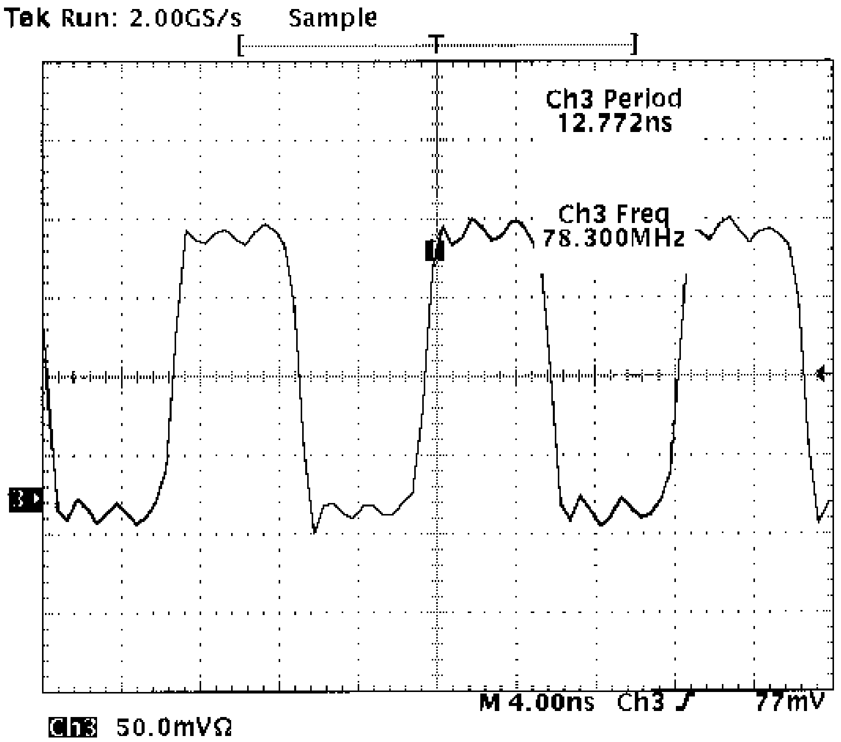


Figure 6.34. Measured output clock (lower end) when VCC=3.3, 25C

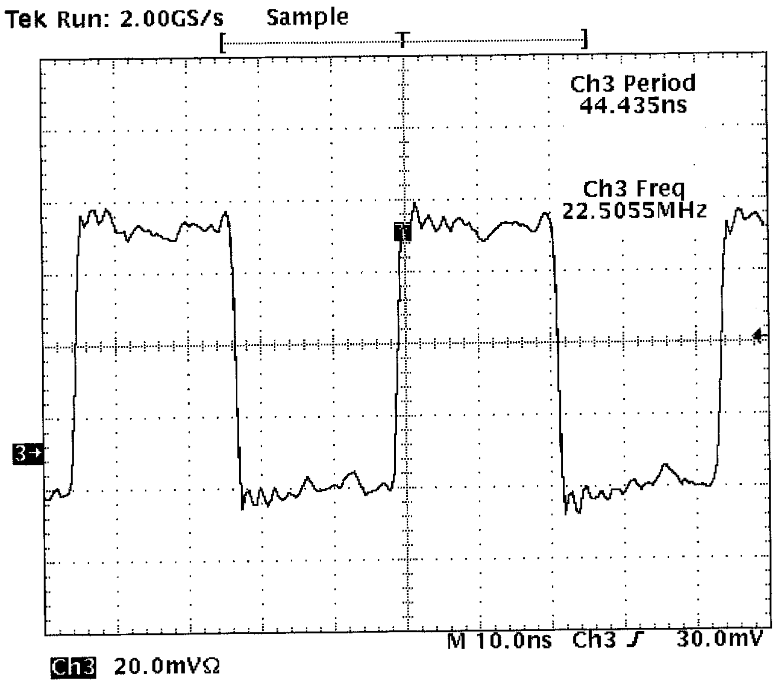


Figure 6.35. Measured output clock (lower end) when VCC=1.4V, 25C

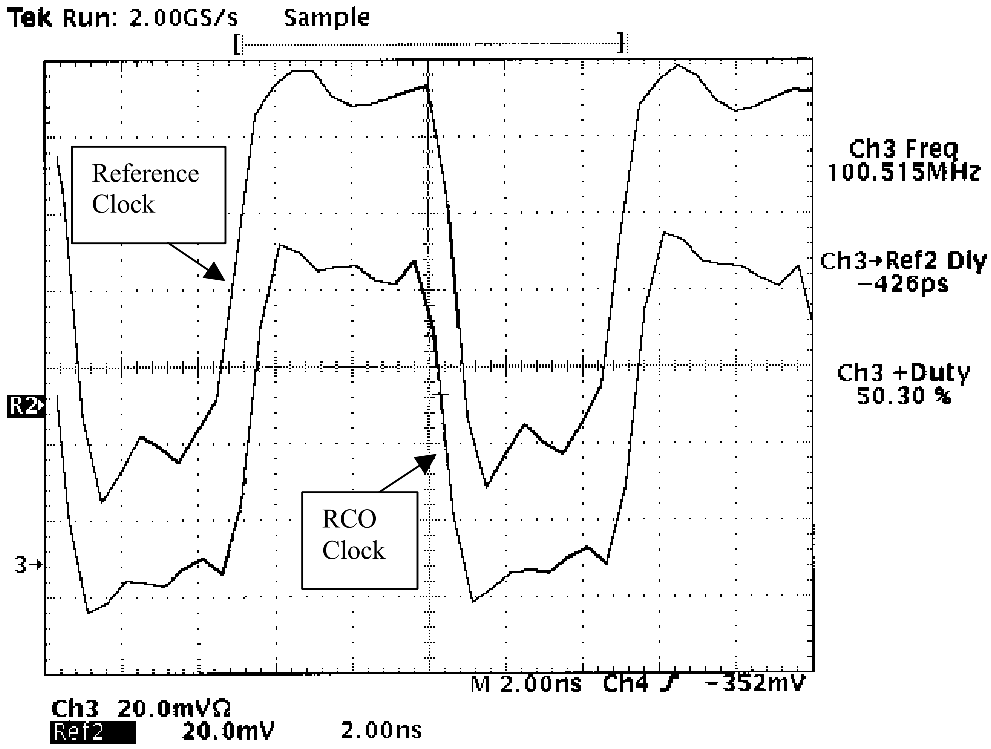


Figure 6.36. Generated RCO clock versus internal reference clock @100MHz, VCC=2.5V, room temperature

The experimental results of this RCO-based ADPLL are compared to the DCO-based ADPLL [19] and self-biased charged-pump PLL [15], and main features, including lock characteristics and operating conditions, are shown in Table 6.3.

Features	Self-biased CPPLL	DCO-based ADPLL	RCO-based ADPLL
Frequency Lock Range	2.5KHz – 550MHz @ VCC=3.3V	50MHz – 550MHz @VCC=3.3V	65MHz – 385MHz @VCC=2.5V
Lower Limit of VCC	2.1V	0.7V	1.4V
Peak-to-Peak Jitter	~384ps@250MHz*	~125ps@200MHz	~210ps@200MHz
RMS Jitter	~144ps@250MHz*	NA	~30ps@200MHz
Lock Time	NA	~50 cycles	~80 cycles
Chip Size	0.4mm^2	1100mil^2	0.3mm^2
Technology	0.5μm N-well CMOS	0.5μm TLM CMOS	0.18μm CMOS
Methodology	Analog	All-digital	All-digital

* Jitter was measured with 500mV of 1MHz square wave supply noise.

Table 6.3. Comparison between existing approaches and the RCO-based ADPLL

Chapter 7

Conclusion

This dissertation has discussed the basic design techniques used for high speed phase-locked and delay-locked loops. The innovative contributions of this work include:

(1) Develop an all-digital PLL utilizing a register-control oscillator with a comparable jitter performance as an analog implementation;

(2) The dual-loop architecture increases the capture range of this ADPLL and fast lock is possible with a proposed hierarchy pull-in process;

(3) Low duty-cycle distortion and frequency granularity of this ADPLL makes it attractive for many applications.

(4) As an entirely digital implementation, it is portable and scalable for other process and low supply voltage operation.

Although a special loop stabilized mechanism is used to get better jitter performance for this RCO-based ADPLL, it is still desired to used delay-locked loops for some timing critical applications. Some future developments of this RCO-based ADPLL include (1) How to implement an all-digital solution for poor input jitter applications? Higher-order digital loop filters may be necessary to suppress the external jitter and recover the clock/data. (2) The zero initial phase error is important especially for the fine loop to acquire and stay locked. More elaborated locking mechanism is needed if that is not the case. (3) In order to cover wide lock range, the delay line may not be settled down before next phase frequency comparison for the coarse loop. To achieve robust performance, delays should be added at the cost of longer lock time.

Another contribution of this dissertation is to develop a register-controlled symmetrical DLL, which has features of low duty-cycle distortion, robust operation over process, voltage and temperature. With additional delay interpolations, the skew could be minimized according to the resolution of fine delay taps.

As the feature size of CMOS process technology is shrunk down toward 0.1μm and clock frequency jumps up beyond 1GHz, new techniques need to be pursued to achieve robust performance over process, supply voltage and temperature variations. The major contribution of this dissertation shows one way to implement complicated feedback circuits, phase-locked and delay-locked loops, using all-digital means. The concepts of these designs have been proven on silicon using different process technologies, 0.21μm and 0.18μm CMOS.

Besides two practical all-digital designs, the concepts and design criteria of PLLs and DLLs, and the comparison between all-digital and analog implementations illustrated in this dissertation will provide useful information and lead to further investigation and development in these areas.

Bibliography

[1] F. M. Gardner, *Phaselock Techniques*, Second Edition. John Wiley & Sons, Inc., ISBN 0-471-04294-3, 1979.

[2] R. E. Best, *Phase-Locked Loops, Theory, Design and Applications,* Third Edition. McGraw-Hill, ISBN 0-07-006051-7, 1995.

[3] B. Razavi, *Monolithic Phase-locked Loops and Clock Recovery Circuits, Theory and Design*. IEEE Press, ISBN 0-7803-1149-3, 1996.

[4] F. M. Gardner, "Charge-pump phase-lock loops", IEEE Tran. Commun., vol. COM-28, no. 11, pp. 1849-1858, Nov. 1980.

[5] M. Pamel, "Analysis of a charge pump PLLs", IEEE Trans. Commun., vol. 42, no. 7, pp. 2490-2498, July 1994.

[6] D. K. Jeong, G. Borriello, D. Hodges, R. H. Katz, "Design of PLL-Based Clock Generation Circuits", IEEE J. of Solid-State Circuits, vol. SC-22, pp. 255-261, April 1987.

[7] M. Johnson, E. Hudson, "A Variable Delay Line PLL for CPU-Coprocessor Synchronization", IEEE J. of Solid-State Circuits, vol. SC-23, pp. 1218-1223, October 1988.

[8] I. A. Young, J. Greason, K. Wong, "A PLL clock Generator with 5 to 110 MHz of Lock Range for Microprocessors", IEEE J. of Solid-State Circuits, vol. SC-27, pp. 1599-1607, Nov. 1992.

[9] J. Alvarez, H. Sanchez, G. Gerosa, R. Countryman, "A wide-Bandwidth Low-Voltage PLL for PowerPC^{TM} Microprocessors", IEEE J. of Solid-State Circuits, vol. SC-30, pp. 383-391, April 1995.

[10] D. Mijuskovic, M. Bayer, T. Chomicz, N. Garg, F. James, P. McEntarfer, J. Porter, "Cell-Based Fully Integrated CMOS Frequency Synthesizers", IEEE J. of Solid-State Circuits, vol. SC-29, pp. 271-279, March 1994.

[11] I. Novof, J. Austin, R. Kelkar, D. Strayer, S. Wyatt, "Fully Integrated CMOS Phase-Locked Loop with 15 to 240 MHz Locking Range and ±50 ps Jitter", IEEE J. of Solid-State Circuits, vol. 30, pp. 1259-1266, Nov. 1995.

[12] S. Tanoi, T. Tanabe, K. Takahashi, S. Miyamoto, M. Uesugi, “A 250-622 MHz Deskew and Jitter-Suppressed Clock Buffer Using Two-Loop architecture”, IEEE J. of Solid-State Circuits, vol. 31, pp. 487-493, April 1996.

[13] J. Christiansen, “An Integrated High Resolution CMOS Timing Generator Based on an Array of Delay Locked Loops”, IEEE J. of Solid-State Circuits, vol. 31, pp. 952-957, July 1996.

[14] F. M. Gardner, “Frequency Granularity in Digital Phaselock Loops”, IEEE Tran. Commun., vol. 44, No. 6, pp. 749-758, June 1996.

[15] J. Maneatis, “Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques”, IEEE J. of Solid-State Circuits, vol. 31, No. 11, pp. 1723-1732, November 1996.

[16] V. Kaenel, D. Aebischer, C. Piguet, E. Dijkstra, “A 320 MHz, 1.5mW @ 1.35V CMOS PLL for Microprocessor Clock Generation”, IEEE J. of Solid-State Circuits, vol. 31, No. 11, pp. 1715-1721, Nov. 1996.

[17] H. Yang, L. Lee, R. Co, “A Low Jitter 0.3-165 MHz CMOS PLL Frequency Synthesizer for 3V / 5V Operation”, IEEE J. of Solid-State Circuits, vol. 32, pp. 582-586, April 1997.

[18] S. Sidiropoulos, M. Horowitz, “A Semidigital Dual Delay-Locked Loop”, IEEE J. of Solid-State Circuits, vol. 32, pp. 1683-1692, Nov. 1997.

[19] J. Dunning, G. Garcia, J. Lundberg, E. Nuckolls, “An all-digital PLL with 50-cycle lock time suitable for high-performance microprocessors”, IEEE J. of Sold-State Circuits, vol. 30, pp. 412-422, April 1995.

[20] J. S. Chiang, K. Y. Chen, “A 3.3V All Digital Phase-Locked Loop with Small DCO Hardware and Fast Phase Lock”, 1998 Symposium on VLSI Circuits Digest of Technical Papers.

[21] A. Hatakeyama, et al., “A 256-Mb SDRAM Using a Register-Controlled Digital DLL”, IEEE J. Solid-State Circuits, vol. 32, pp. 1728-1732, Nov. 1997.

[22] F. Lin, J. Miller, A. Schoenfeld, M. Ma, R. J. Baker, “A Register-Controlled Symmetrical DLL for Double-Data-Rate DRAM”, IEEE J. Solid-State Circuits, vol. 34, No. 4, pp. 565-568, April 1999

[23] T. Saeki, et al., “A 2.5-ns Clock Access, 250-MHz, 256-Mb SDRAM with Synchronous Mirror Delay”, IEEE J. Solid-State Circuits, vol. 31, pp. 1656-1665, Nov. 1996.

[24] B. Garlepp, et al., “A Portable Digital DLL for High-Speed CMOS Interface Circuits”, IEEE J. Solid-State Circuits, vol. 34, No. 5, pp. 632-642, May 1999.

[25] C. H. Kim, et al., “A 64-Mbit, 640-Mbyte/s Bidirectional Data Strobed, Double-Data-Rate SDRAM with a 40-mW DLL for a 256-Mbyte Memory System”, IEEE J. Solid-State Circuits, vol. 33, pp. 1703-1708, Nov. 1998

[26] G. Geannopoulos, X. Dai, “An adaptive digital deskewing circuit for clock distribution networks”, ISSCC digest of technical papers, pp. 400-401, Feb. 1998.

[27] S. Eto, *et al.*, “A 1Gb SDRAM with Ground Level Precharged Bit Line and Nonboosted 2.1-V Word Line”, IEEE J. Solid-State Circuits, vol. 33, pp. 1697-1701, Nov. 1998.

[28] M. Bazes, R. Ashuri, E. Knoll, “An Interpolating Clock Synthesizer”, IEEE J. Solid-State Circuits, vol. 31, pp. 1295-1301, Sep. 1996.

[29] D. L. Chen, “A Power and Area Efficient CMOS Clock/Data Recovery Circuit for High-Speed Serial Interfaces”, IEEE J. Solid-State Circuits, vol. 31, pp. 1170-1175, Aug. 1996.

[30] M. Combes, K. Dioury, A. Greiner, “A Portable Clock Multiplier Generator Using Digital CMOS Standard Cells”, IEEE J. Solid-State Circuits, vol. 31, pp. 958-965, July 1996.

[31] G. C. Moyer, et al., “The Delay Vernier Pattern Generation Technique”, IEEE J. Solid-State Circuits, vol. 32, pp. 551-562, April 1997.

[32] S. Y. Sun, “An Analog PLL-Based Clock and Data Recovery Circuit with High Input Jitter Tolerance”, IEEE J. Solid-State Circuits, vol. SC-24, pp. 325-330, April 1989.

[33] B. Kim, D. N. Helman, P. R. Gray, “A 30-MHz Hybrid Analog / Digital Clock Recovery Circuit in 2-μm CMOS”, IEEE J. Solid-State Circuits, vol. SC-25, pp. 1385-1394, Dec 1990.

[34] T. H. Lee, J. F. Bulzacchelli, “A 155-MHz Clock Recovery Delay- and Phase-Locked Loop”, IEEE J. Solid-State Circuits, vol. SC-27, pp. 1736-1746, Dec 1992.

[35] N. Ishihara, Y. Akazawa, “A Monolithic 156 Mb/s Clock and Data Recovery PLL Circuit Using the Sample-and-Hold Technique”, IEEE J. Solid-State Circuits, vol. SC-29, pp. 1566-1571, Dec 1994.

[36] T. H. Hu, P. R. Gray, “a Monolithic 480 Mb/s Parallel AGC / Decision / Clock - Recovery Circuit in 1.2-μm CMOS”, IEEE J. Solid-State Circuits, vol. SC-28, pp. 1314-11320, Dec 1993.

[37] B. Kim, T. C. Weigandt, P. R. Gray, "PLL/DLL System Noise Analysis for Low Jitter Clock Synthesizer Design", Proc. of ISCAS, June 1994.

[38] A. Abidi, R. G. Meyer, "Noise in Relaxation Oscillators", IEEE J. Solid-State Circuits, vol. SC-18, pp. 794-802, Dec 1983.

[39] T. C. Weigandt, B. Kim, P. R. Gray, "Analysis of Timing Jitter in CMOS Ring Oscillators", Proc. of ISCAS, June 1994.

[40] A. Hajimiri, S. Limotyrakis, T. Lee, "Jitter and Phase Noise in Ring Oscillators", IEEE J. Solid-State Circuits, vol. 34, pp. 790-804, June 1999.

[41] A. Hajimiri, T. Lee, "A General Theory of Phase Noise in Electrical Oscillators", IEEE J. Solid-State Circuits, vol. 33, pp. 179-193, Feb. 1998.

[42] B. Razavi, "A Study of Phase Noise in CMOS Oscillators", IEEE J. Solid-State Circuits, vol. 31, pp. 331-343, March 1996.

[43] J. McNeill, "Jitter in Ring Oscillators", IEEE J. Solid-State Circuits, vol. 32, pp. 870-879, June 1997.

[44] M. Soyuer, R. G. Meyer, "High-Frequency Phase-Locked Loops in Monolithic Bipolar Technology", IEEE J. Solid-State Circuits, vol. SC-24, pp. 787-795, June 1989.

[45] A. W. Buchwald, K. W. Martin, A. K. Oki, K. W. Kobayashi, "A 6-GHz Integrated Phase-Locked Loop Using AlGaAs/GaAs Heterojunction Bipolar Transistors", IEEE J. Solid-State Circuits, vol. SC-27, pp. 1752-1762, Dec. 1997.

[46] B. Razavi, J. J. Sung, "A 6 GHz 60 mW BiCMOS Phase-Locked Loop", IEEE J. Solid-State Circuits, vol. SC-29, pp. 1560-1565, Dec. 1994.

[47] M. Soyuer, "A Monolithic 2.3 -Gb/s 100-mW Clock and Data Recovery Circuit in Silicon Bipolar Technology", IEEE J. Solid-State Circuits, vol. SC-28, pp. 1310-1313, Dec. 1993.

[48] S. K. Enam, A. A. Abidi, "NMOS IC's for clock and Data Regeneration in Gigabit-per-second Optical-fiber Receivers", IEEE J. Solid-State Circuits, vol. SC-27, pp. 1763-1774, Dec 1992.

[49] H. Ransijn, P. O'Connor, "A PLL-based 2.5-Gb/s GaAs Clock and Data Regenerator IC", IEEE J. Solid-State Circuits, vol. SC-26, pp. 1337-1344, Oct. 1991.

[50] A. Pottbacker and U. Langmann, “an 8 GHz Silicon Bipolar Clock-Recovery and Data-Regenerator IC”, IEEE J. Solid-State Circuits, vol. SC-29, pp. 1572-1576, Dec. 1994.

[51] C. A. Sharpe, “A 3-state phase detector can improve your next PLL design”, EDN Magazine, Sep 20, 1976.

[52] C. R. Hogge, “A self Correcting Clock Recovery Circuit”, IEEE J. of Lightwave Technology, vol. LT-3, pp. 1312-1314, Dec. 1985.

[53] P. H. Lewis and W. E. Weingarten, “A Comparison of Second, Third, and Fourth Order Phase-Locked Loops”, IEEE Trans., AES-3, pp. 720-727, July 1967.

[54] L. W. Couch II, *Modern Communication Systems, Principles and Applications*, Prentice Hall, ISBN: 0-02-325286-3, 1995.

[55] R. J. Baker, H. W. Li, D. E. Boyce, *CMOS Circuit Design, Layout, and Simulation*, IEEE Press, ISBN: 0-7803-3416-7, 1998.

[56] H. Ikeda, H. Inukai, “High-Speed DRAM Architecture Development”, IEEE J. of Solid-State Circuits, vol. 34, pp. 685-692, May 1999.