A 10-bit Digital to Time Converter with a Phase Noise Filter

Master of Science Thesis

Tianyu Wang

A thesis presented for the degree of
Master of Science in EEMCS

August 20, 2023

A 10-bit Digital to Time Converter with a Phase Noise Filter

by

Tianyu Wang

to obtain the degree of Master of Science
in Electrical Engineering
at the Delft University of Technology,
Under the supervision of Dr. Sijun Du
to be defended publicly on Thursday August 31, 2023 at 9:30 AM.

Student number:	5399726
Project duration:	August 20, 2022 – August 31, 2023
Thesis committee:	Dr. Sijun Du, TU Delft, supervisor
	Dr. S. M. Alavi, TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Preface

This thesis proposes a new phase noise filter structure to filter the phase noise of the Digital to time Converter (DTC) for clock and data recovery (CDR) application. The thesis is under the instruction of Dr. Sijun Du at Delft University of Technology and Dr. Zhongkai Wang, who received Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA. The main finding of the thesis is a new filter structure that can filter not only the phase noise of Delay locked loop (DLL) but also that of phase interpolator (PI), while the currently charge-injection-based phase noise filter can only filter the DLL's phase noise. The thesis committee is formed by Dr. Sijun Du, supervisor, and Dr.S.M.Alavi from the Electronic Circuits and Architectures (ELCA) research group.

I want to give my sincere thanks to Dr. Sijun Du for his support and guidance on this thesis project. I'm also very grateful for the instructions from Dr. Zhongkai Wang, who shares the initial idea of the phase noise filter and teaches a large number of implementation details during the schematic and layout design.

Tianyu Wang
Delft, August 2023

Abstract

The growing number of users, devices, and connections leads to a growing demand for bandwidth. This speed communication is usually realized asynchronously, which means the data is exchanged between two servers while the accompanying sampling clock is not exchanged. A CDR system is required on the server that receives data to recover the sampling clock for the data. The DTC in the CDR system is the main jitter source of the recovered data. A low-jitter DTC is required to generate data of low-jitter performance, calling for the application of a phase noise filter. Currently, most phase noise filters are based on the charge injection technique, which can only filter the phase noise of the DLL-based DTC. This thesis presents a new phase noise filter, which can filter both the DLL and PI phase noise.

The proposed phase noise filter is inspired by the noise transfer function from the phase detector's input to the delay locked loop(DLL) output of a type-II DLL, which shows a first-order low-pass transfer function. The noise suppression pole frequency is adjustable and can be modified by changing the gain of each component in the circuit. In addition, by carefully placing the frequency of the LDO's pole, second-order noise filtering can be realized.

During design, a 10-bit DTC is constructed first and the proposed filter is placed behind the DTC to verify the effectiveness of the filter. The design achieves the post-layout level. The simulation results show that the DTC's phase noise drops from 1.099 ps_{rms} to 315.9 fs_{rms} with the filter. The area is 695 µm $\times$ 693.5 µm. The design consumes 42.3 mW with 1.8V supply in 180nm BCD technology.

Contents

1 Introduction

Modern Internet companies such as Google and Facebook store and provide data service through the use of “servers”. Thousands of their servers are connected to each other for data storage and exchange. In order to save cost, asynchronous data transmission is applied in most cases. The asynchronous means the accompanying clock of the data on the transmit side wouldn’t be sent to the receive side. Methods that can recover the clock from the data and then use this clock to re-time the data are required, calling for the application of Clock and Data Recovery (CDR) [1].

In this chapter, the concept of CDR is introduced first, whose main components are a digital-to-time converter (DTC) and the CDR logic. The design goal of the thesis is a low-jitter DTC. Thus, the introduction of the DTC structure comes in the next section. The concept of the jitter is introduced in the third section, which is a fundamental metric of the CDR system. The motivation and scope comes in the next section. Finally, the thesis outline is given.

1.1. Clock and Data Recovery

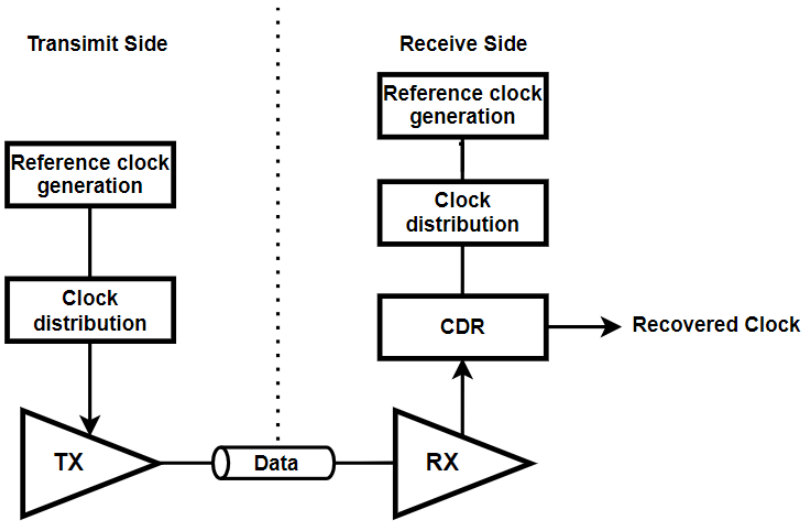

Figure 1.1: Block diagram of a typical CDR system

The function of the CDR is to generate a clock to sample a data stream without an accompanying clock. Fig 1.1 exhibits the block diagram of a typical CDR system. The reference clock is usually generated by a crystal oscillator(OSC), which outputs a stable but low-frequency clock. The phase-locked loop(PLL) aims to multiply the reference clock frequency. The clock distribution circuit distributes the multiplied clock to the components on chip that requires clocks(e.g. ADC, logic gates, DAC, DTC). The digital-to-time converter(DTC) is usually composed of a delay locked loop(DLL)/Phase locked loop(PLL)

and a Phase interpolator(PI). The data from the transmit side is sent to the receive side, which lacks the associated clock. The CDR system on the receiver side generates a recovering clock to re-sample the received data. The reference clocks from the crystal OSC on both the transmit and receive sides are assumed to have the same frequency. Thus, the reference clock on the receive side has the same frequency as that of the received data. The CDR logic circuit controls the DTC to generate the clock with a correct phase to sample the received data.

Fig1.2 shows the conceptual model of a CDR system. The input of the system is the data on the receiving side, which lacks the accompanying clock. The received data are sent to the CDR system to generate a recovered clock. The DTC outputs a clock, whose phase is controlled by the CDR logic and the frequency is the same as the clock on the transmit side. To correctly sample the data, the sampling point must be put away from the rising/falling edge transition. The recovered clock is utilized by the re-timer to re-sample the received data.

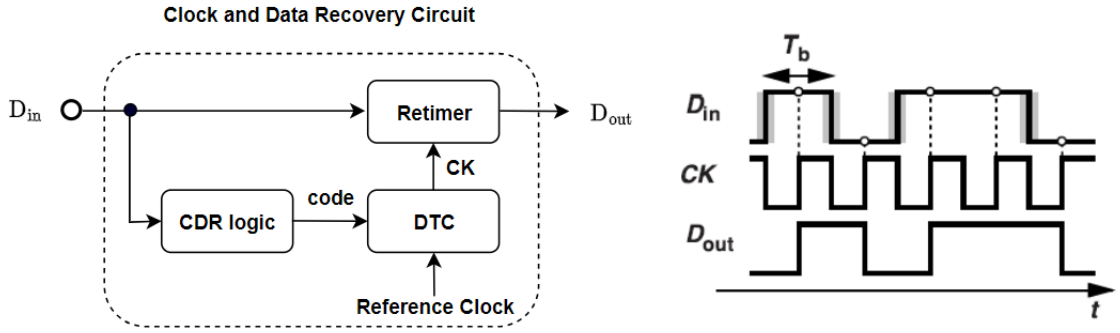

Figure 1.2: Conceptual model of a CDR system

1.2. Digital to time converter

As shown in Fig 1.2, the inputs of the DTC are the digital control codes and a reference clock. The output clock has the same frequency as the reference clock and the delay can be adjusted by the digital code. The delay is controlled by the CDR logic, which is adjusted such that the sampling point in Fig 1.2 is away from the rising/falling edge transition.

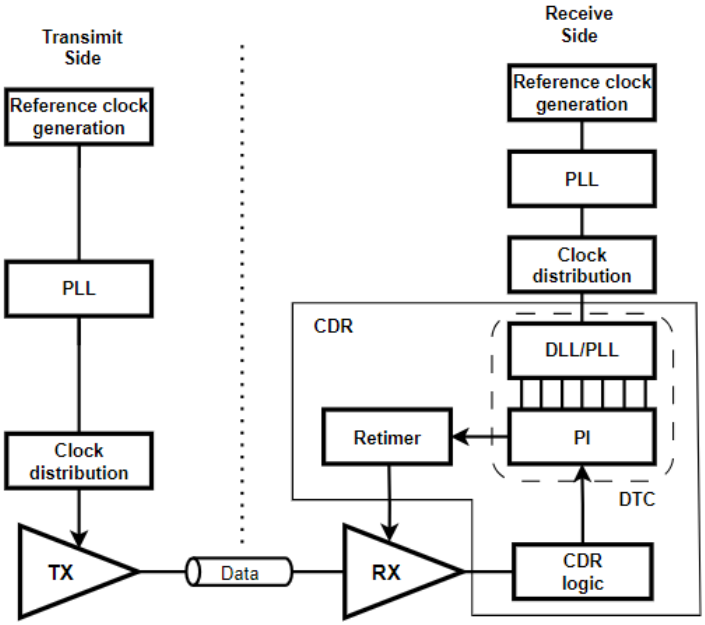

Figure 1.3: Implementation of DLL/PLL based DTC

There are multiple categories of DTC implementations. However, in this report, the two most significant and readily used implementations are introduced. Their implementations are shown in Fig 1.3. The function of the DLL and PLL is to generate multi phases. The main difference between the DLL and PLL is that the multiple phases are generated by a voltage controlled delay line(VCDL) in DLL while that is generated by a voltage controlled oscillator(OSC) in PLL. Consequently, there exists a jitter accumulation in the PLL-based implementation while the DLL-based implementation doesn't. The DLL also shows advantages in its design simplicity, robustness, and stability than the PLL. On the other hand, the PLL-based implementation exhibits an advantage in that the PLL's loop bandwidth can low-pass filter the incoming jitter[2]. Meanwhile, the DLL shows an all-pass phase characteristic, which means the input phase noise will not be filtered. The input jitter can also be amplified due to the finite bandwidth of the DLL delay line[3]. In a short summary, the DLL-based DTC is easy to construct but it will amplify the input phase noise at high frequency offset. The PLL-based DTC can provide noise filtering but the PI's phase noise is left unfiltered with high design complexity. A possible implementation is to utilize the DLL and PI structure, with a phase noise filter attached behind the PI.

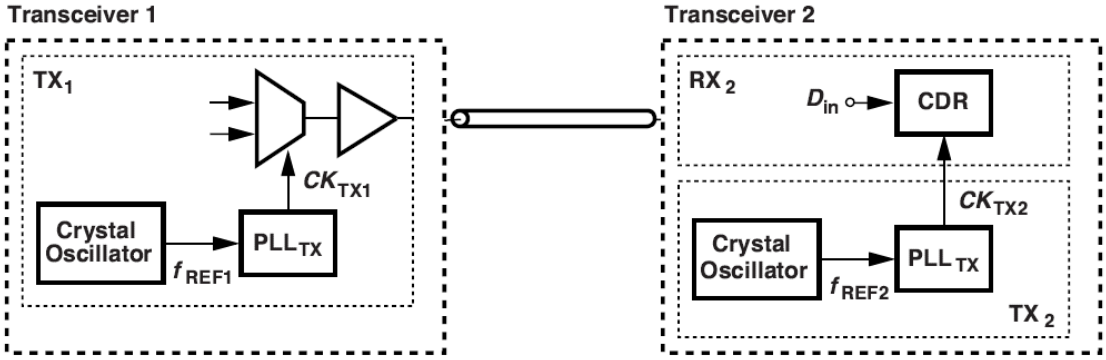

Figure 1.4: Problem of frequency mismatch between two transceivers[4]

The combination of DLL and Phase interpolator (PI) is utilized to achieve a high-resolution DTC. The DLL generates the initial multiple phases. The PI utilizes two adjacent DLL phases to generate more intermediate phases. The more stages the DLL has, the more power consumption will the DTC cost. Meanwhile, the PI's linearity increases with more DLL phases. The combination of DLL and PI ensures a power-efficient and area-efficient method to get a high-resolution and high-linearity DTC than DLL-only DTC or PI-only DTC[5].

The PI is usually implemented in a digital way to solve the plesiochronous operation problem. The plesiochronous operation problem refers to the frequency mismatch between the transmit and receive sides' reference clock, which is shown in Fig 1.4. The reference clock is generated by two crystal OSC, whose output frequencies can't be totally identical in reality. The post PLL multiplies the frequency of the crystal OSC's output clock. This frequency mismatch generates an unbound phase error. When the PI is realized in a digital way, it has the ability to introduce a monotonically increasing phase for CK1 that grows without saturation. Thus, it compensates for the unbound phase difference between the received data and the recovered clock and solves the issue [4].

1.3. Concept of Jitter and Phase noise

A fundamental metric of the CDR system is the jitter performance, which is defined as the dynamic deviation of the clock from an ideal edge. Fig 1.5 illustrates the influence of jitter on the ideal clock. The real clock is generated by the combination of the jitter and the ideal clock. The jitter causes a difference in the rising/falling edge of the clock. The lower 'absolute jitter' refers to the amplitude of the jitter. The amplitude of the jitter can be random or deterministic, resulting from the device noise such as thermal and flicker noise (random) or supply/substrate noise, channel induced jitter (deterministic).

The phase noise is defined as the ratio of the phase noise in a 1-Hz bandwidth at a specified frequency offset, f_m, to the oscillator signal amplitude at frequency f_0, which can be regarded as the frequency-domain representation of the jitter. Fig 1.6 illustrates an example of an OSC's phase noise spectrum. The area 'A4' represents the white phase noise region. The area 'A3' represents the flicker

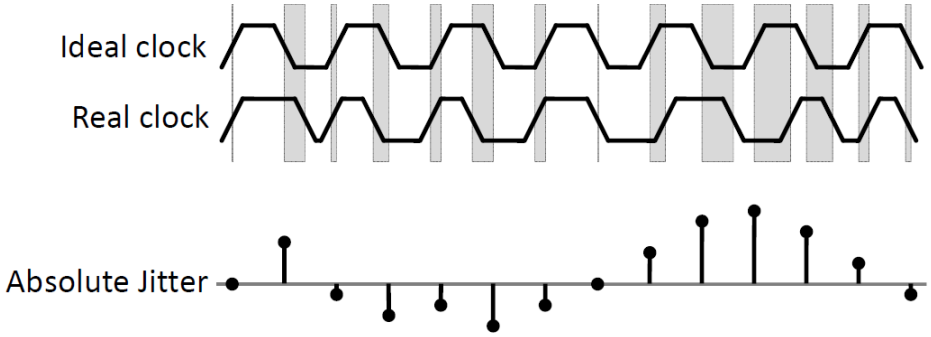

Figure 1.5: Illustration of the concept of jitter[6]

phase noise region(slope = -20dB/decade). The area 'A1' and 'A2' represents higher order phase noise region(slope = -40,-60dB/decade). Fig 1.6 also exhibits the conversion process from the phase noise to jitter. The jitter is the integration of phase noise in the frequency domain.

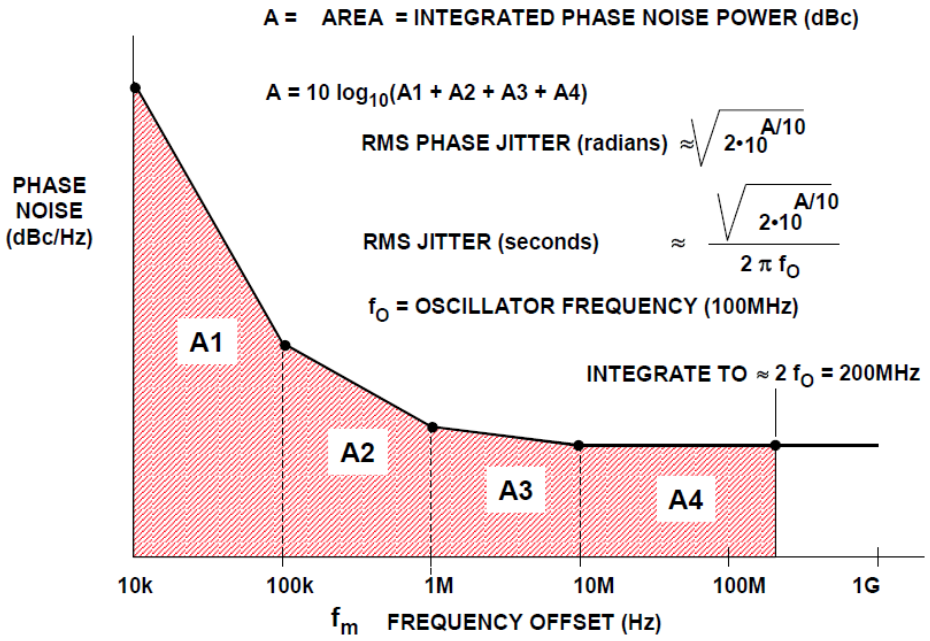

Figure 1.6: One example of the OSC Phase Noise in dBc/Hz vs. Frequency Offset[7]

1.3.1. Jitter source of the CDR's recovered data

A vital property of the CDR system is that the jitter of the received data will not be inherited by the recovered data. Fig 1.2 exhibits the waveform of a typical CDR system. The clock 'CK', which is the recovered clock of the CDR system, samples the data away from the rising/falling edge transition. Note that the jitter of the received data appears at its rising and falling edge. Thus, the jitter from the input data of the receiver side is masked. In this case, the jitter of the sampling clock becomes the major jitter source of the recovered data. To get low-jitter recovered data, a low-jitter DTC is required.

1.4. Motivation and Scope

The increasing data rate requirement for wireline communication leads to an increasingly important role of the clock and data recovery (CDR) circuits in maintaining low bit error rates. The DTC generates the re-sample clock in the CDR circuit, which becomes the main jitter source of the recovered data due to the jitter mask property of the CDR circuit. To get low-jitter recovered data, a low-jitter DTC is required, calling for the utilization of a phase noise filter.

This thesis proposes a new phase noise filter structure that can be attached to the DTC for phase noise suppression. To verify the effectiveness of the filter, a 10-bit DTC is constructed first, which composes of a 4-bit DLL and 6-bit PI. The DLL structure ensures a low design complexity and high robustness compared with the PLL-based structure. The PI is realized in a digital way to solve the plesiochronous operation problem. The proposed phase noise filter is placed behind the DTC for verification. The design is implemented in a 180-nm BCD process.

1.5. Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 introduces the related works for phase noise reduction and filtering. This chapter also reformulates the research questions in a form that complies with the state of the art in the field.

Chapter 3 presents the mechanism of the proposed design, including a 10-bit DTC(4-bit DLL and 6-bit pipelined PI) and the thesis's innovation, a phase noise filter.

Chapter 4 exhibits the implementation details of the proposed design. The design process of the main components in the DLL, PI and phase noise filters are introduced as well.

Chapter 5 shows the post-simulation results of the design, including the layout, phase noise, power, and linearity performance.

Chapter 6 discusses the effectiveness of the phase noise filter, along with the analysis on how to further reduce the phase noise components at high/low frequency offset.

Chapter 7 gives the conclusions and presents some suggestions for future improvement.

2

Related works

This chapter first introduces the related works on reducing phase noise, including the most basic technique, increasing power consumption, and state-of-the-art techniques. Finally, the research questions are reformulated in a form that complies with the state of the art in the field.

2.1. Related works to reduce the phase noise of the DTC

In this section, methods to reduce the phase noise of the DTC in three directions are introduced.

2.1.1. Trade power with phase noise

It's proved theoretically that the DTC's phase noise directly trades with power [4]. Thus, the most apparent method to reduce the phase noise is to increase the power consumption. A brief proof of this theory is given as follows:

1. Assume the output clock of one normal DTC can be expressed as:

$$\begin{aligned} V_{out}(t) &= V_0 cos[\omega_0 t + \phi_n(t)] \\ &= V_0 cos\omega_0 t cos\phi_n(t) - V_0 sin\omega_0 t sin\phi_n(t) \\ &\approx V_0 cos\omega_0 t - V_0 \phi_n(t) sin\omega_0 t \end{aligned} \tag{2.1}$$

2. Consider there are N identical DTCs with their output clocks added together. Their output can be expressed as:

$$V_{out}(t) = V_0 cos\omega 0t - V_0 \phi_n(t) sin\omega_0 t + ... V_0 cos\omega 0t - V_0 \phi_n(t) sin\omega_0 t \tag{2.2}$$

   where $\omega 0$ is the average oscillation frequency and $\phi_n 1$, ..., $\phi_n N$ denote the phase noise of the individual DTCs respectively. The output can then be expressed as:

$$V_{out}(t) = N V_0 cos\omega 0t - V_0(\phi_{n1} + ... + \phi_{nN}) sin\omega_0 t \tag{2.3}$$

3. The normalized phase noise can be expressed as $(\phi_{n1} + ... + \phi_{nN})/N$. The spectrum can be expressed as:

$$S_\phi(f) = \frac{S_{\phi n1} + ... + S_{\phi nN}}{N^2} \tag{2.4}$$

   because the phase noises of the N DTCs are uncorrelated. Since the DTCs are identical, their phase noise spectrum are equal and□

$$S_\phi(f) = \frac{S_{\phi n1}}{N} \tag{2.5}$$

   In other words, the overall phase noise is reduced by a factor of N at the cost of an N-fold increase in power dissipation. In a short summary, the simplest way to design a low-jitter DTC is to sacrifice power consumption, which is not desired as power is a vital metric for DTC as well.

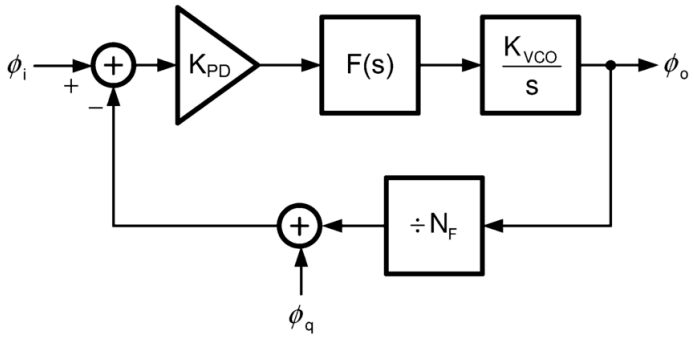

Figure 2.1: Phase domain model of a typical phase locked Loop[8]

2.1.2. PLL as a Phase noise filter

An alternative approach is to apply a phase noise filter. One implementation of the phase noise filter is a phase-locked loop (PLL). Fig 2.1 exhibits the phase domain model of a typical PLL. Its working principle is briefly introduced since it's not the focus of the thesis. 'K_{PD}' refers to the linearized gain of the phase detector. '$F(s)$' refers to the transfer function of the components, CP, and loop filter. 'K_{VCO}/s' refers to the linearized gain of the voltage controlled OSC(VCO). The output of the VCO is a clock with a frequency. The phase of a clock is the integration of the clock frequency, which is the reason why the VCO's linearized gain is integrated in the phase domain model. The transfer function from input to output can be expressed as:

$$S_\phi(f) = \frac{K_{PD}K_{VCO}F(s)/s}{1 + K_{PD}K_{VCO}F(s)/Ns} \tag{2.6}$$

The transfer function from the PLL's input to output shows a low-pass characteristic [9], which can be used to filter the high-frequency phase noise. The PLL can be post-connected to the DTC to realize the

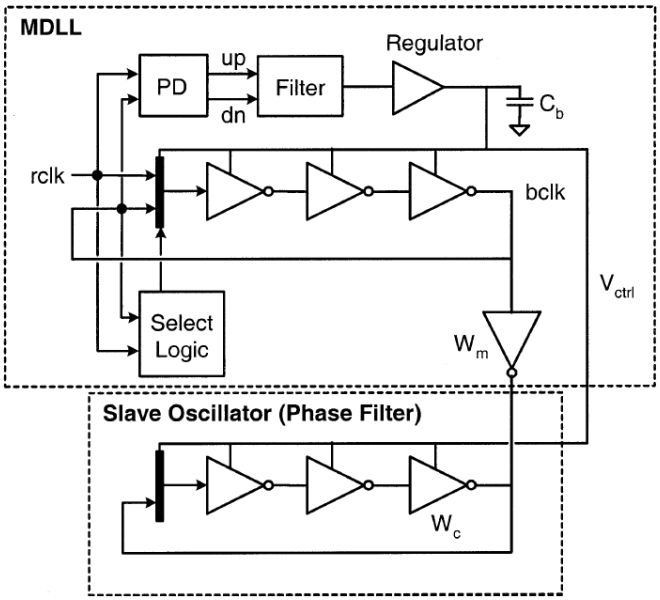

Figure 2.2: Multiplying DLL and the slave OSC as the phase noise filter

phase noise filtering. The inputs of the PLL's PD are the DTC's output clock and VCO's output clock after frequency division. The PLL's output clock becomes the filtered clock. The problem with the PLL phase noise filter is that the traditional second-order and third-order PLLs increase significant design complexity, calling for a simpler phase noise filter implementation.

2.1.3. Charge injection based phase noise filter

One phase noise filter implemented based on injection locking is exhibited in Fig 2.2 [10]. The clock without filtering is generated by the multiplying DLL(MDLL). The working principle of the MDLL is briefly introduced since it's not the focus of the thesis. In most of the periods, the MUX connects the first and end stages of the VCDL to form a ring OSC. In the rest period, the MUX connects the reference clock 'rclk' to the first stage the VCDL. The number of ring OSC periods determines the multiplying coefficient. The select logic counts the number of edges and determines when to inject into the ring oscillator. The PD, Filter, and regulator work similarly to the typical DLL, which corrects the phase difference between the reference clock and the VCDL's output clock.

The output clock of the multiplying DLL is charge-injected to the slave OSC for phase noise filtering. The slave OSC utilizes the same delay element as the MDLL and shares the same control voltage. Thus, the slave OSC runs the same frequency as the MDLL in the ideal case.

Fig 2.3 exhibits the phase domain model of the MDLL and slave OSC in the Z-domain. The injection strength is defined as $S_c = W_c/(W_c + W_m)$. In the frequency domain, the jitter transfer function from the slave OSC's input to output is given by:

$$\frac{\phi_{filt}}{\phi_{out}} = \frac{K_c}{z - (1 - K_c)} \tag{2.7}$$

where $K_c = 1 - (1 - S_c)^N$ and N is the multiplication factor of the MDLL. Equation2.7 indicates a first-order low-pass filter with a single pole located at:

$$p_0 = \frac{-ln(1 - K_c)}{T_i} \tag{2.8}$$

where T_i is the injection frequency from the MDLL to the slave OSC. The problem with this single-phase charge injection structure is that there exists a phase imbalance between the injected stage and non-injected stages. Consider the injected clock frequency is f_{inj} and the slave OSC's self-oscillation frequency is f_0, the frequency mismatch would cause a phase error between the injected and non-injected stages of the slave OSC, whose magnitude is proportional to $(f_0 - f_{ing})/f_0$[11]. As a result, the slave OSC needs to set the f_0 to a frequency different from f_{inj} to achieve a zero phase error among the slave OSC's output phases. To address this residual phase error issue, the multi-phase injection scheme is utilized [12].

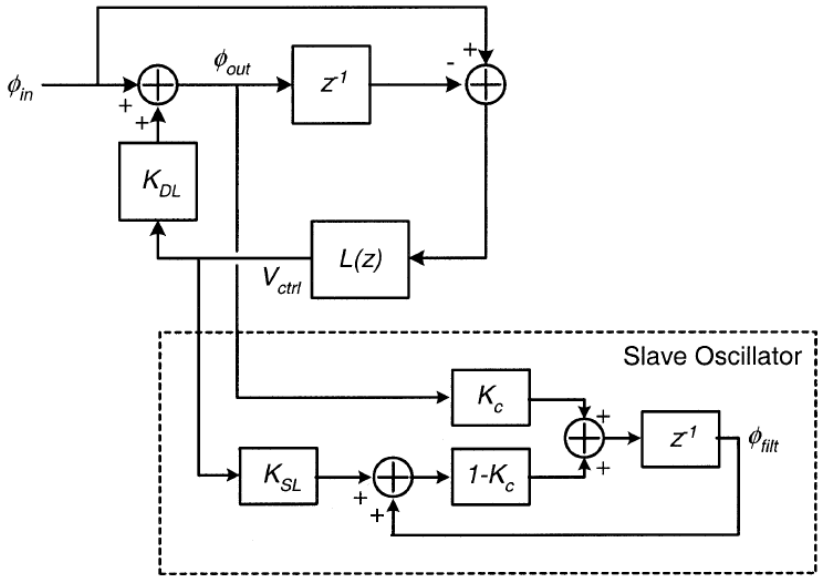

Figure 2.3: Phase domain model of the MDLL and the slave OSC

Multi-Phase charge injection

The OSC's phase symmetry is restored by multi-phase injection to all OSC stages. Fig 2.4 exhibits the implementation of the multi-phase charge injection. The first stage is a DLL with quadrature error correction(QEC). The mixers and the OTA form the quadrature error detection and correction path. The generated control voltage controls the delay time of the VCDL in the DLL and the oscillation frequency of the Ring OSC(ROSC) in the next stage. All the output phases of the QDLL are injected into all the stages of the ROSC. The inaccuracies of the QDLL's injection phases would lead to leads or lags in the ROSC phases. On the other hand, the feedback structure of the ROSC requires the total phase shift around all the OSC stages to be 2π. Consequently, the QDLL's phase inaccuracies can be suppressed.

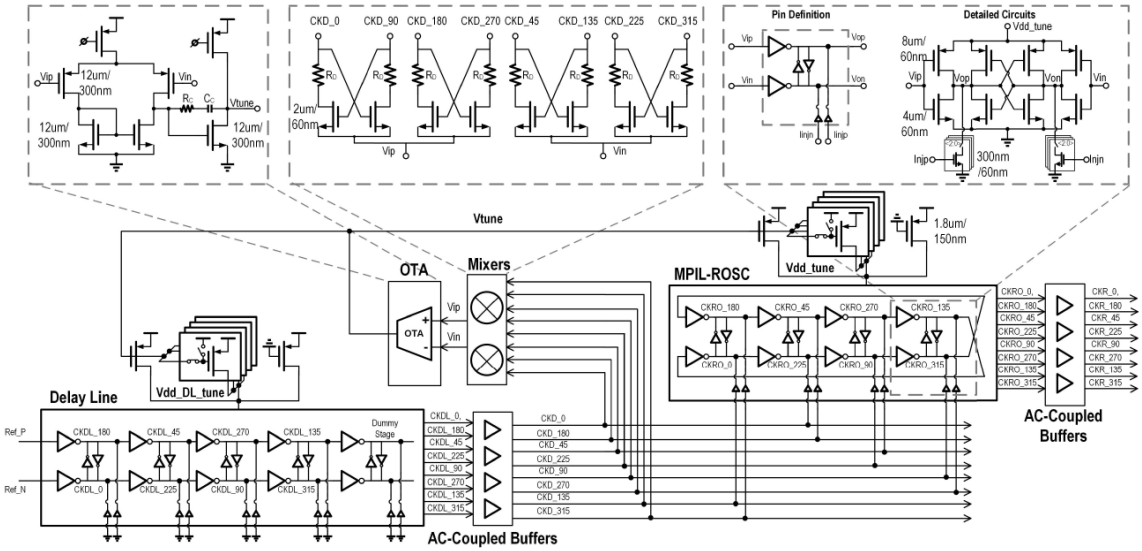

Figure 2.4: Implementation of the multi-phase charge injection using a QDLL and a ROSC

The phase domain analysis of the multi-phase charge injection is similar to that of the signal phase injection. The phase noise from the QDLL is first-order filtered. A multi-phase charge injection-ROSC is an ideal candidate for high-accuracy and low-jitter multi-phase clock generation. However, when it comes to the DTC design, this structure has the limitation that the phase noise of the phase interpolator(PI) can't be filtered.

2.2. Research Question

There are three methods to reduce the jitter of the DTC. The direct method to reduce phase noise is to increase the power consumption, which is not desired since power is another vital metric for the DTC design. The low-pass characteristics of the phase locked loop(PLL)'s transfer function from input to output can be utilized to form a phase noise filter. However, the traditional second-order and third-order PLLs increases significant design complexity due to stability issue. A multi-phase charge injection-ROSC is an ideal candidate for high-accuracy and low-jitter multi-phase clock generation. However, when it comes to the DTC design, this structure has the limitation that the phase noise of the phase interpolator(PI) can't be filtered. The only method to reduce PI's phase noise is to increase the power consumption.

In order to solve the issues of the above methods, a new phase noise filter is proposed in this thesis, featuring:

1. First-order phase noise filtering instead of sacrificing power.
2. Based on DLL instead of PLL to reduce design complexity and increase robustness.
3. Filter all the noise from the DTC, including DLL and PI

3

The proposed design

In this chapter, the mechanism of the proposed design is presented. Firstly, the overview of the design is exhibited. The design can be divided into two parts: a high-resolution DTC and the thesis's innovation point, a phase noise filter, whose working principles are introduced sequentially in this chapter.

3.1. Overview of the proposed design

The overview of the design is depicted in Fig 3.1. The DTC is composed of a 4-bit DLL. The DLL is realized with quadrature error correction instead of the traditional PFD-based structure to ensure DTC phase linearity. The 16:2 Multiplexer(MUX) selects two adjacent DLL phases into the PI. The pipelined PI extends the phase resolution to 10 bit. The output clock of the pipelined PI contains the correct phase information, which is controlled by the external CDR logic. The phase noise filter replicates the phase information of the PI's output clock to the output and generates the filtered clock with the reference clock.

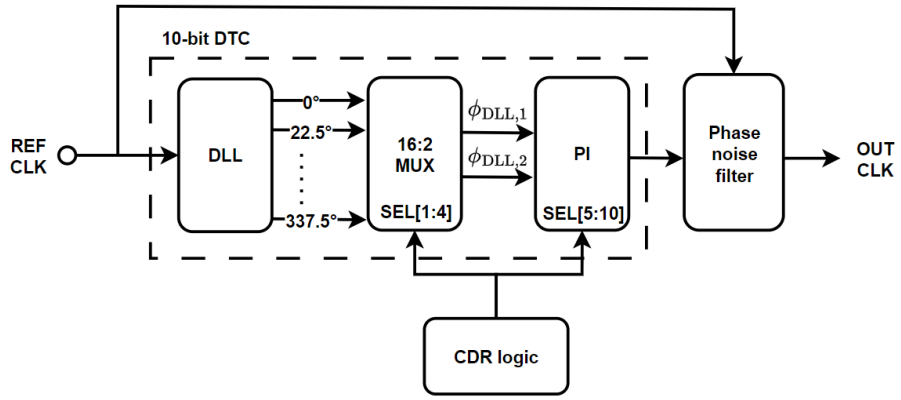

Figure 3.1: Overview of the proposed design

3.2. Phase noise filter

Fig 3.2 exhibits the implementation of the proposed phase noise filter, which is inspired by the typical type-2 DLL's noise transfer function(NTF) from the phase detector to the output[13].

The working principles of the phase noise filter are the same as the type-2 DLL as well. The phase frequency detector(PFD) detects the phase difference between the output clock of the DTC stage and the output clock of the voltage controlled delay line(VCDL). The charge pump(CP) converts the phase difference into a current. The loop filter(LF) integrates the phase difference error current to generate the control voltage for the VCDL. The low-drop amplifier(LDO) controls the supply voltage of the VCDL to modify its delay time and provides currents for the VCDL. The VCDL is composed of a cascade of

differential inverter stages. The negative feedback loop ensures the phase alignment between the PD's two input clocks. This property is utilized to replicate the phase information from the former DLL and PI stage to the output of the phase noise filter.

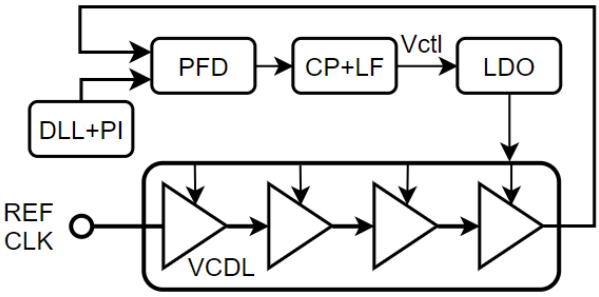

Figure 3.2: Implementation of a typical delay locked loop

Fig 3.3(b) shows the phase domain model of the phase noise filter. All the components of the filter are linearized and all the noise sources are included. The loop filter generates a pole at DC for integration. The LDO will also generate a pole, whose frequency is put higher than the loop bandwidth. Thus, the whole DLL can be regarded as a one-pole system, which is stable and robust.

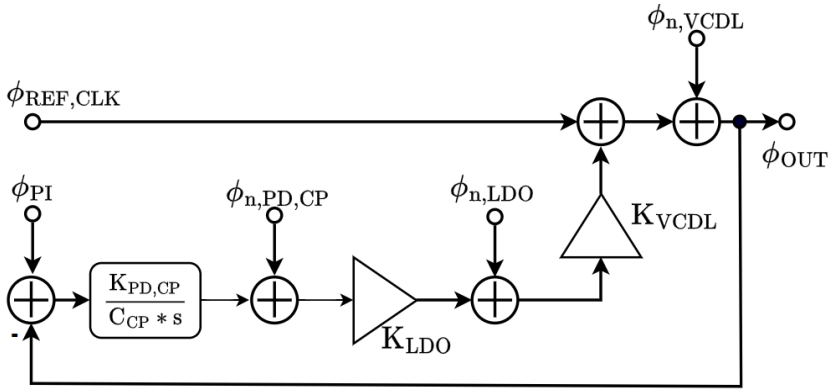

Figure 3.3: The phase domain model of the proposed phase noise filter

Noise transfer functions of the phase noise filter

Based on the phase domain model, the NTFs from each component to the output can be analyzed. The most important NTF is from the PFD to the output, which shows a low-pass characteristic:

$$\frac{\phi_{OUT}}{\phi_{PI,IN}} = \frac{K_{PD,CP}K_{LDO}K_{VCDL}}{sC_{LF} + K_{PD,CP}K_{LDO}K_{VCDL}} \tag{3.1}$$

This low-pass characteristic means that the phase noise from the former DLL and PI stages will be first-order low-pass filtered. The phase noise suppression effectiveness is determined by the pole frequency. For example, the reference clock of the proposed design is 1GHz. With this frequency, the jitter performance is usually calculated as the integration of phase noise from 1kHz to 100MHz. By placing the noise suppression pole at a low frequency(e.g. 10kHz), the jitter from the DLL and PI can be effectively reduced.

It can also be observed from the equation 3.1 that the pole frequency can be modified, which is related to the gain of the PD, CP, VCDL, and LDO. The magnitude of the capacitor is also relevant. During design, it was found that changing the gain of the PD, CP, and VCDL by several orders of magnitude is challenging due to their inherent design limitations. However, utilizing a large LF capacitor can be a more feasible approach.

Although a lower pole frequency leads to a better phase noise filtering effect, the pole frequency's lower bound is limited by the loop bandwidth of the DLL. The loop bandwidth is defined as the unity gain frequency of the DLL's open loop gain:

$$LoopGain = \frac{K_{PD,CP}K_{LDO}K_{VCDL}}{sC_{CP}} \tag{3.2}$$

It can be observed from equation3.2 and3.1 that the pole of the NTF corresponds to the unity gain frequency of DLL's loop gain. Consequently, by placing the pole at a low frequency, the loop bandwidth is effectively set to a low frequency as well. This property limits the application scenario of the proposed design. The NTFs from the other components to the output are analyzed as well. The NTF of the PFD, CP, and LF to the output shows similar low-pass characteristics as well:

$$\frac{\phi_{OUT}}{\phi_{PD,CP,LF}} = \frac{K_{LDO}K_{VCDL}}{sC_{CP} + K_{PD,CP}K_{LDO}K_{VCDL}} \tag{3.3}$$

This NTF shows the same noise suppression pole frequency as that of the PFD's NTF. This property means that the noise from the PFD and CP will be effectively reduced as well when a large loop filter capacitor is utilized to reduce PFD's phase noise. This reduces the design difficulty for the PFD, CP, and LF for low-jitter purposes. The NTFs from the VCDL and LDO to the output exhibit a high-pass instead of low-pass characteristic:

$$\frac{\phi_{OUT}}{\phi_{VCDL}} = \frac{s}{s + \frac{K_{PD,CP}K_{LDO}K_{VCDL}}{C_{CP}}} \tag{3.4}$$

$$\frac{\phi_{OUT}}{\phi_{LDO}} = \frac{sK_{VCDL}}{s + \frac{K_{PD,CP}K_{LDO}K_{VCDL}}{C_{CP}}} \tag{3.5}$$

Utilizing a large LF capacitor shifts the high pass poles of the NTFs to a lower frequency. This causes a larger integrated jitter in the band of interest (1kHz to 100MHz), which is not helpful in reducing the jitter from the LDO and VCDL. Consequently, the phase noise from these two components becomes the dominant jitter source of the proposed DTC and becomes the upper limit of the proposed design's jitter performance. It's crucial to design these two components for low-phase noise. Methods like increasing the power consumption of the VCDL, increasing the transconducance of the input pairs of the LDO's internal differential amplifier and lowering the loop bandwidth of the LDO should be applied.

3.3. Delay locked Loop

The major function of the first-stage DLL and second-stage PI is to provide the required 10-bit phase resolution. They can be designed to feature low power consumption and high phase noise. Their high phase noise can be filtered by the post phase noise filter. Thus, the design emphasis of the DLL and PI stage is the linearity of the output phases.

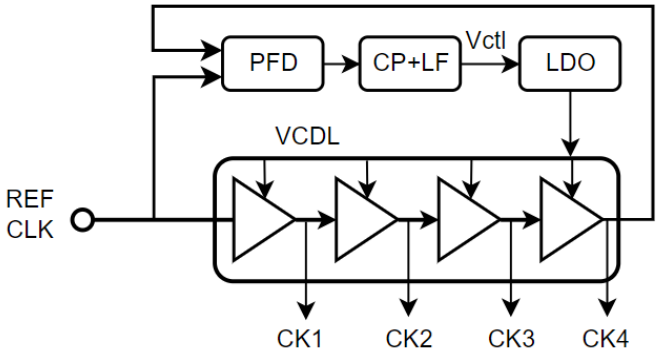

Figure 3.4: An implementation of a 2-bit DLL

The most challenging problem to design a high-linearity DLL is the static phase error. In this section, the causes of the static phase error are analyzed firstly. The quadrature error correction technique is then introduced to solve the static phase error problem.

3.3.1. Introduction to the Static phase error

The static phase error is defined as the phase difference between the PD's two input clocks when the loop reaches the stable state. Fig.3.4 exhibits a typical DLL with a 4-stage VCDL, which outputs 2-bit phases. When the DLL reaches the stable state, the PFD's two input clocks are aligned, which are the input and output clocks of the VCDL. In the ideal case, the VCDL's output clock is one period delayed version of its input clock. The VCDL has four identical delay stages, which consequently divide one period(2π) into four equally-spaced phases($\pi/2, \pi, 3\pi/2, 2\pi$). However, the PFD's two input clocks can't be perfectly aligned in the real situation, leading to the static phase error.

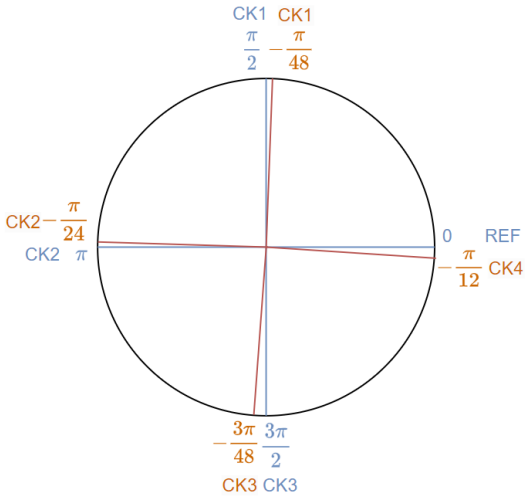

Figure 3.5: Output phases of a 2-bit DLL

The effect of static phase error on the linearity of DLL's output phases is illustrated in Fig.3.5. The blue curves refer to the phases without static phase error, which divides one period into four identical pieces. The brown curves show an output phase example with a static phase error. Considering the reference clock's phase is 0, the end VCDL stage outputs a '$-\frac{\pi}{12}$' phase. The end stage's phase error is conveyed to the intermediate stages. The phase error is equally divided as well, considering the four VCDL stages are identical. The end stage's static phase error will influence the linearity of all DLL's phases.

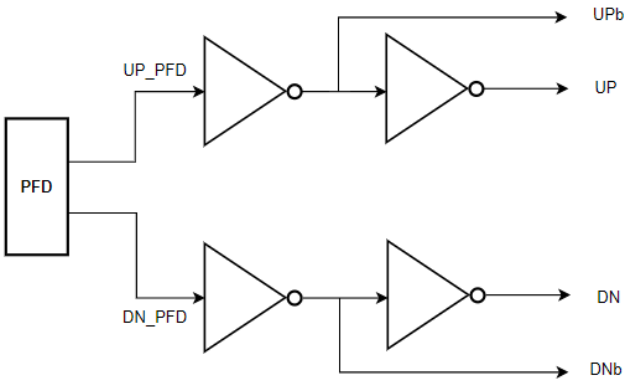

Figure 3.6: Interface between the PFD and the CP.

Causes of the static phase error

The causes of the static phase error are the non-idealities in the phase error detection path, including the PFD and CP. Their non-idealities are analyzed separately as follows:

Non-idealities in the PFD: The function of the PFD is to convert the phase error into a duty cycle error. Depicted in Fig.3.6, the duty cycle of the UPb and DN signals reveals how much time the charging

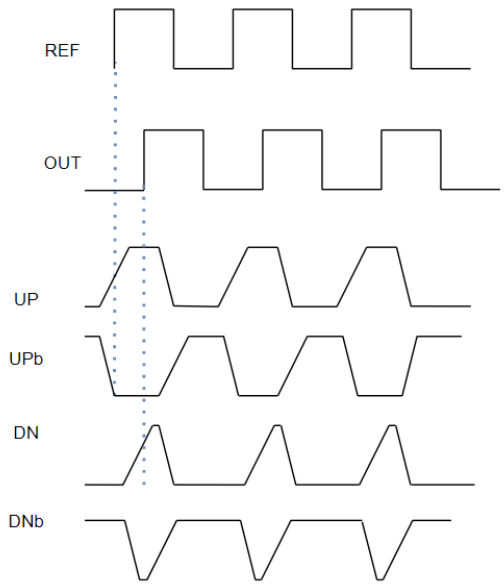

Figure 3.7: Timing diagram of a PFD

and discharging paths of the CP should be enabled. Through simulation, it's found that the phase error resulting from the PFD itself is neglectable(indicated by UP_PFD and DN_PFD in Fig 3.6). The main error source of the PFD is the interface between the PFD and CP. The working principle of the CP in the next stage requires the switching signal UPb for charging to be low voltage level active while the DN signal to be high voltage level active. The initial output signals of the PFD are both high-level active. The interface is used to convert the high-level-active signal into low-level-active. Due to the difference in PMOS and NMOS's mobility, mismatch, and different inverter loading, the duty cycles of the four output indicating signals will deviate from their ideal value. An example timing diagram of a PFD is shown in Fig.3.7, where the duty cycle deviation can be observed. The duty cycle deviation influences the zero phase error detection of the PFD.

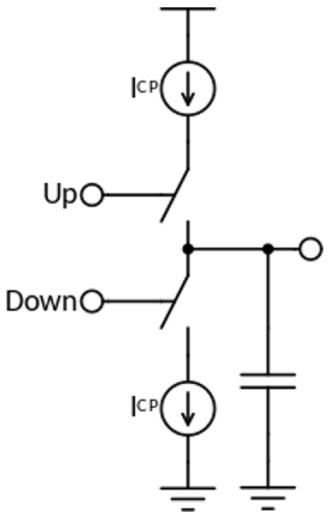

Figure 3.8: Conceptual model of a CP

Non-idealities in the CP and LF: The charge pump(CP) converts the duty cycle errors into a current. Fig 3.8 shows the conceptual model of a CP. The duty cycle of the signal UP and Down controls the charging and discharging time. For example, With a larger duty cycle of the UP signal, more charging currents will flow on the capacitor, which increases the control voltage and reduces the VCDL's output

clock's delay time. Fig.3.9 shows a schematic of a typical charge pump. The charging path is enabled by a PMOS, which is low voltage level active and requires UPb signal from the PFD-to-CP interface. The other two control signals, UP and DNb, from the interface stop the charging and discharging process. The differences in UP_B's falling time and DN's rising time, the coupling from the switch signals to the bias voltage will all lead to a worse static phase error.

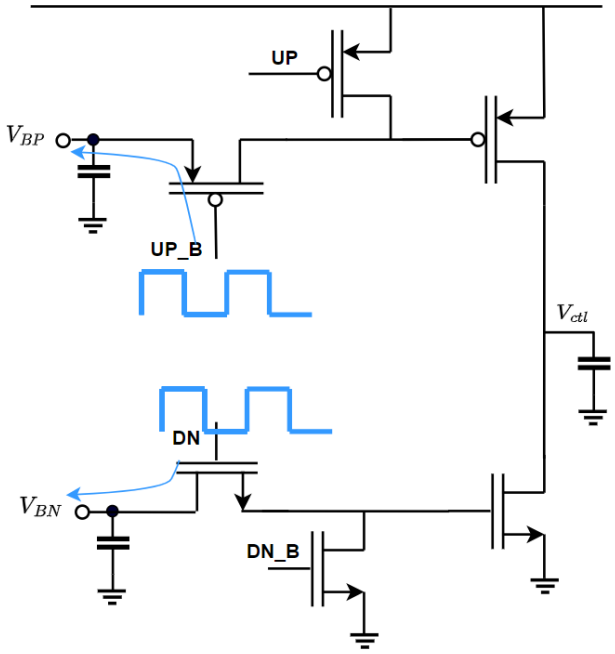

Figure 3.9: Schematic of a typical CP with signal coupling

These inherent non-idealities make it impossible to generate a zero static phase error, which calls for the application of quadrature error correction.

3.3.2. Quadrature error correction

The quadrature error correction(QEC) technique is utilized to solve the static phase error problem in the PFD-based DLL structure[14]. The reason for this improvement is that the QEC converts the phase error into a differential voltage level error instead of the duty cycle error. Thus, the static phase error is determined by the symmetry of the design instead of the mobility difference between PMOS and NMOS.

The 'quadrature' means a 90 ° phase difference. Fig 3.10 exhibits the block diagram of a typical DLL with quadrature error correction. The phase detector in this technique is the XOR logic gate, which detects the quadrature phase error between the input two clocks. Fig 3.11 shows the XOR's output signals with and without quadrature errors. In perfect quadrature conditions, the XOR output signals

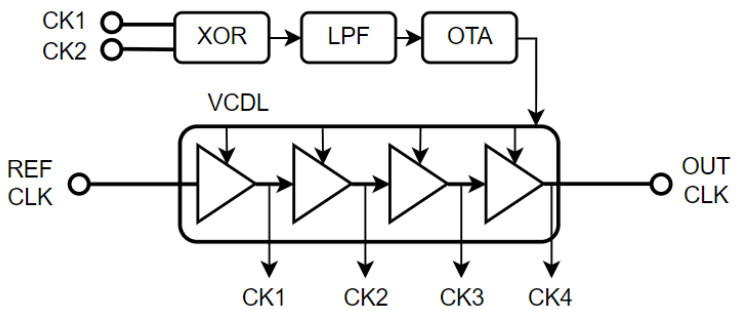

Figure 3.10: the block diagram of a typical DLL with quadrature error correction

have a 50 percent duty cycle and twice the frequency. When the quadrature error is larger than 90°, the XOR's output signals' duty cycle is larger than 50 percent, which means it has a higher average value than that in perfect quadrature condition, and vice versa. The following low-pass filter(LPF) is used to extract the average value information from the XOR gate. The operational transconductance amplifier(OTA) converts the output voltages of the LPF into a current and integrates the current.

The 'XOR+LPF+OTA' quadrature error detection path converts the quadrature into a duty cycle's average value, a differential voltage, and a current subsequently. The static phase error depends on the gain and the mismatch situation of each component, apart from the NMOS and PMOS's mobilities. Consequently, this technique ensures a small static phase error.

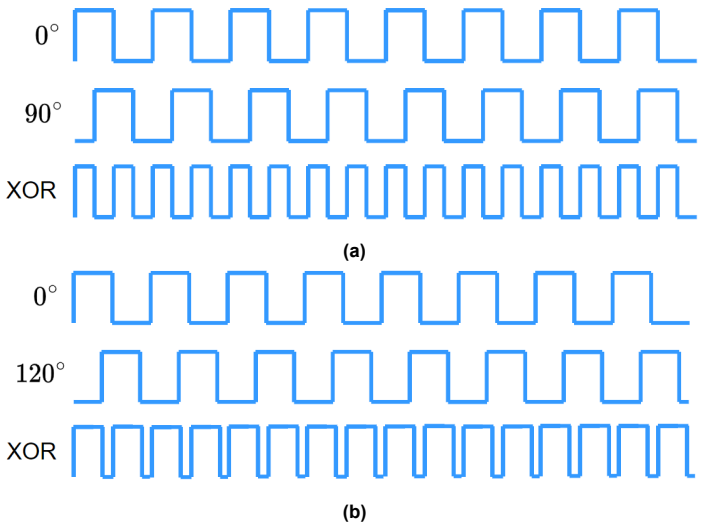

Figure 3.11: XOR output when the input clocks are(a) in perfect quadrature(b)the presence of quadrature errors

Phase domain model

The phase domain model of the DLL with QEC is depicted in Fig.3.12, where each component's linearized transfer function(TF) and noise source are exhibited. The VCDL is realized with the cross-coupling differential structure. Three main stages of the VCDL are shown in fig.3.12, including the first, end, and middle stages. Eight differential VCDL stages are used in the VCDL, which generates the required 16 phases(4-bit). The first stage and the middle stage generate the required quadrature phases, which are sent to the XOR logic gate for quadrature error detection.

The whole loop can be regarded as a one-pole system, which is similar to the traditional PFD-based DLL. However, it can be observed from fig.3.12 that there are three poles in the loop. Apart from the integration pole of the OTA at DC, the second pole is created due to the utilization of an active LPF. A Low-dropout (LDO) amplifier is utilized to adjust the delay of the VCDL, which creates the third pole. In order to maintain the loop as a one-pole system, the DC pole from the OTA is chosen as the main pole, which reduces the phase margin from 180° to 90°. On the other hand, the rest two poles' frequencies should be put higher than the DLL's loop bandwidth. The DLL's loop bandwidth is defined as the unity-gain frequency of its loop gain. The influence on the phase margin from the two poles of the LPF and LDO can be ignored if they are set at least 10 times higher than the DLL's loop bandwidth.

When it comes to the quantitative analysis, the loop gain of the DLL loop is exhibited below. The frequency when the loop gain reaches 0dB is the DLL's loop bandwidth.

$$LoopGain = \frac{gm_{OTA}K_{XOR}K_{LPF}K_{LDO}K_{VCDL}}{sC_{INT}} \tag{3.6}$$

The NTFs of the components, XOR, LPF, and the OTA can be combined into one equation. This is because the noise of the XOR and LPF can be transformed into the noise of the OTA by timing the linearized gain of XOR and LPF. The NTF from the three components to the output is given by the equation below, which shows a first-order low-pass characteristic:

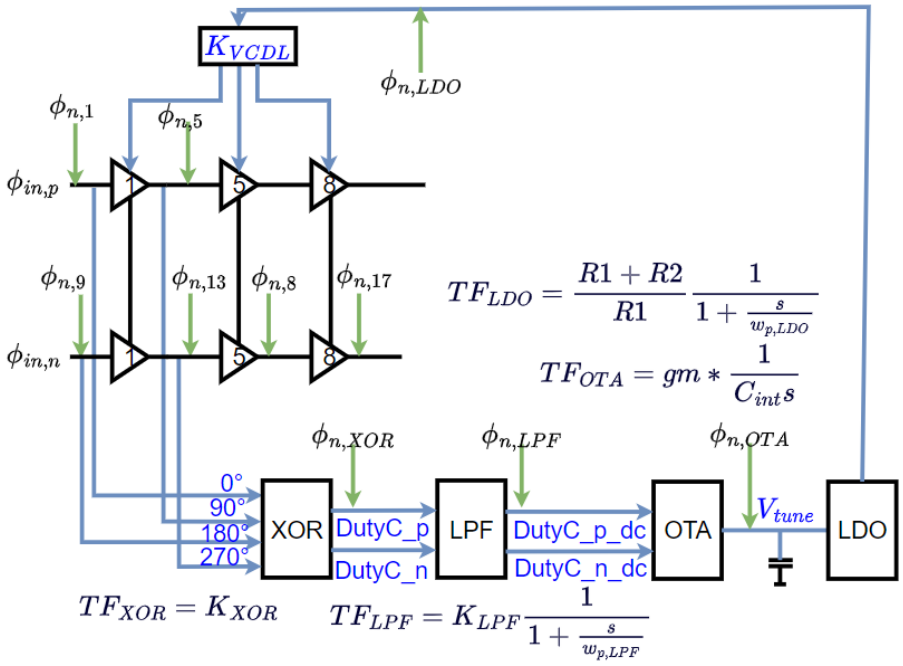

Figure 3.12: Phase domain model of the DLL with quadrature error correction

$$\frac{\phi_{CK_{90}}}{\phi_{n,XOR,LPF,OTA}} = \frac{K_{LDO}K_{VCDL}}{sC_{INT} + gm_{OTA}K_{XOR}K_{LPF}K_{LDO}K_{VCDL}} \tag{3.7}$$

Note that the NTF refers to the relationship between the XOR's two input clock, which is P1, P5, and their reverse phases. The actual phase output P16 is not contained in the feedback loop. Consequently, the NTF from the XOR to the actual output P16 is the NTF in equation3.7 plus the rms noise components of the rest delay line stages.

The NTF from the LDO to the output is given by the equation below, which shows a first-order high-pass characteristic:

$$\frac{\phi_{CK_{90}}}{\phi_{n,LDO}} = \frac{sC_{INT}K_{VCDL}}{sC_{INT} + gm_{OTA}K_{XOR}K_{LPF}K_{LDO}K_{VCDL}} \tag{3.8}$$

The NTF from the VCDL to the output is shown below, which shows first-order high-pass characteristics as well:

$$\frac{\phi_{CK_{90}}}{\phi_{n,VCDL}} = \frac{sC_{INT}}{sC_{INT} + gm_{OTA}K_{XOR}K_{LPF}K_{LDO}K_{VCDL}} \tag{3.9}$$

Power consumption consideration of the DLL-based DTC

The number of output phases in DLL-based DTC grows exponentially with the bit number. For example, if the 10-bit DTC is realized by DLL, the number of the delay line stages becomes 1024. The required power consumption and area are intolerable. To solve this issue, the phase interpolator(PI) is incorporated into the design, which provides a more power-and-area-efficient way to expand the phase resolution.

3.3.3. Phase noise consideration of the DLL stage

From the phase domain model of the DLL with QEC, it can be estimated that the low-frequency phase noise will be dominant by the XOR, LPF, and OTA while the high-frequency phase noise will be dominant by the LDO and VCDL. Based on the phase domain model of the phase noise filter, the phase noise from the DLL will be low-pass filtered. The application scenario of the CDR circuits usually requires the system to have a bandwidth of 20MHz. Thus, the noise suppression pole of the phase noise filter

is set to at least 20MHz. The bandwidth of the DLL with QEC is usually at least several hundred MHz. It can be estimated that the phase noise filter's phase noise from the DLL will be dominant by the low-frequency-offset phase noise from the XOR, LPF, and CP. During design, these components should be taken more attention to reduce their resulting phase noise.

3.4. Phase interpolator

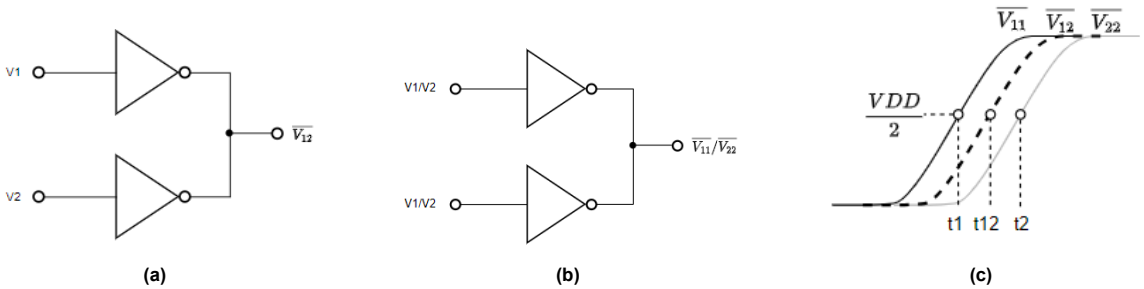

Figure 3.13: (a)PI in interpolation mode(b)PI in phase transfer mode(c)Output phases in interpolation mode and phase transfer mode

The phase interpolator(PI) is incorporated in the design to expand the 4-bit resolution of the DLL to 10 bits. The working principle of a basic PI is illustrated in Fig 3.13[15], which is realized by two inverters simply. Fig 3.13a exhibits the PI in the interpolation mode. In this mode, the two input phases of the inverters are different, whose output clock's phase $\overline{V12}$ is the intermediate phase between V1 and V2. It can also be observed from Fig 3.13a that the phase interpolation incurs a delay with respect to the original edges, V1 and V2. For this skew to be removed, V1 and V2 must experience the same delay, calling for the arrangement in Fig 3.13b. The two input clock's phases are the same in the phase transfer mode. Fig 3.13c illustrates the above process. The phase information of each output clock is detected by measuring the time that each clock reaches $VDD/2$.

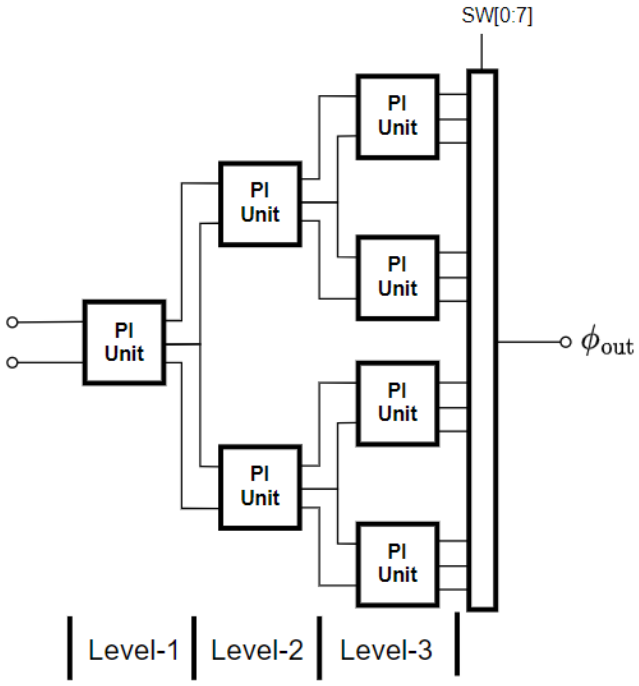

Figure 3.14: The block diagram of a typical DLL with quadrature error correction[16]

The PI units in Fig 3.13 are cascaded to expand the resolution with the PI structure. The problem with this traditional structure is that the number of PI units grows exponentially with the bit number. Fig

3.14 illustrates the configuration of a 3-bit PI. The 3-bit configuration requires 7 PI units and the N-bit configuration requires $2^N - 1$ PI units. Although the power consumption of the PI unit is smaller than that of the VCDL delay line unit, the exponential growth power consumption and the area of the PI structure are still not preferred. To reduce power consumption and area, the pipelined PI structure is applied.

3.4.1. Pipeline PI structure

The power consumption and area of the pipelined PI structure grows linearly with the bit number[17]. Fig 3.15 shows the structure of the pipelined PI from the design in [17]. Each stage extends one-bit resolution, which is composed of three PI units. The inputs of each stage are two phases to be interpolated. The output phases are two delayed versions of the two input phases and one interpolated phase. There is an extra multiplexer (MUX) stage between each PI stage, compared with the traditional PI structure. The MUX selects the desired two intermediate phases of the three phases from the former PI stage. For example, the first stage outputs $0°, 45°, 90°$. The output phases of the second stage would become $0°, 22.5°, 45°$ if $0°, 45°$ are chosen by the MUX stage. The output phases of the second stage would become $45°, 67.5°, 90°$ in the other case.

The pipelined PI structure's power consumption and area grow linearly with the bit number. Thereby the power consumption and area can be saved.

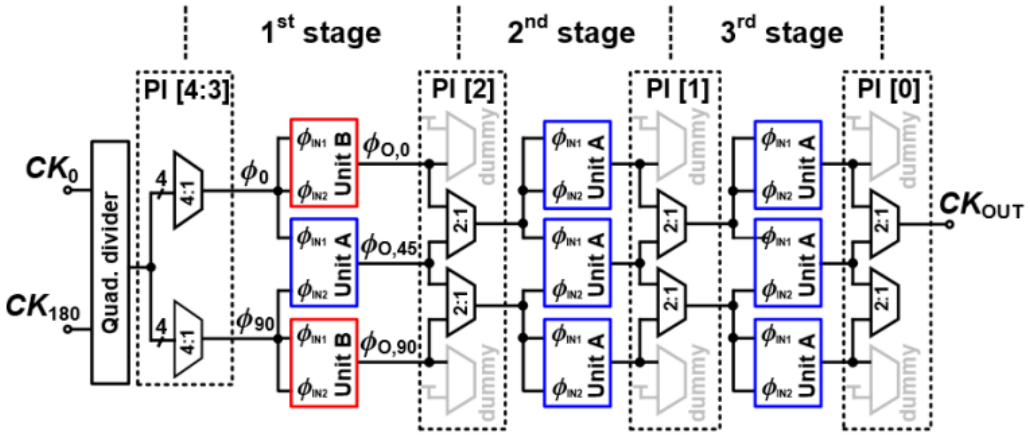

Figure 3.15: the block diagram of a typical DLL with quadrature error correction

3.4.2. Phase noise consideration of the PI stage

The PI structure is an open-loop system while the DLL with QEC is a closed-loop system. Thus, there's no phase model analysis of the PI stage. The phase noise of the PI will exhibit the following pattern: At low frequency offset, the phase noise is dominant by the flicker noise, which drops with a rate of $-10dBc/Hz$. At high frequency offset, the phase noise is dominant by the white noise, which shows a flat spectrum.

The only method to reduce the phase noise from the PI stage is to increase its power consumption by increasing the size of the transistors in the PI units and MUXs.

4

Circuit Implementation and Analysis

The implementation details of the proposed design go in this chapter. The construction of the main components in the DLL, PI, and phase noise filters are introduced sequentially.

4.1. Delay locked loop

The implementation of the DLL with QEC is shown in Fig 4.1, containing an 8-stage VCDL, a current-mode-logic(CML) XOR, a LPF, an OTA, and two LDO amplifiers. The input buffers and dummy XOR gates are not shown in the figure for clarity. The cross-coupled differential VCDL is utilized in the design to minimize the rise and fall time of the clock transition.

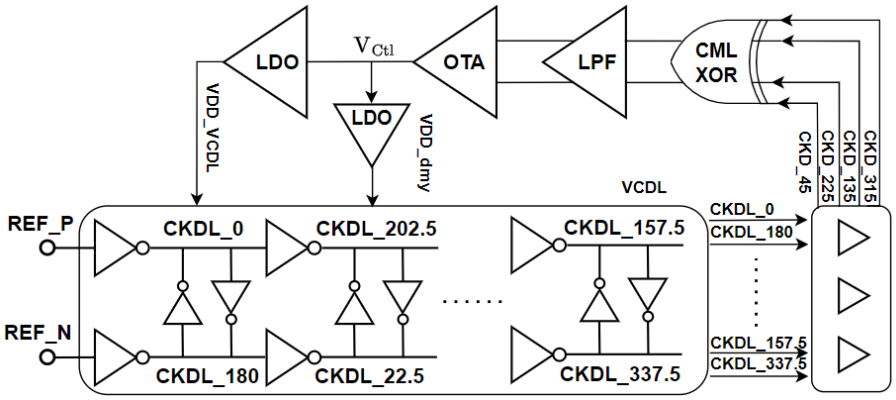

Figure 4.1: Implementation of the DLL with QEC

4.1.1. Procedures to design the DLL

The first step is to determine the number of VCDL unit stages to meet the delay time requirement of a 1GHz reference clock. The second step is to design the XOR, LPF, OTA, and LDO individually. The pole positions resulting from the LPF, OTA, and LDO should be figured out. The third step is to utilize the phase domain model in Fig 3.3. Each component's linearized gain can be derived by open-loop simulation. With the pole of the OTA at DC, the poles of the LPF and LDO need to be put higher than the loop bandwidth to ensure enough phase margin for loop stability. Utilizing the equation 3.2, the loop bandwidth can be calculated. The jitter performance of the DLL can be estimated by each component's noise times their NTFs. The static phase error can also be predicted by the open loop simulation of XOR, LPF, and OTA.

The rest of this section follows the procedures above to construct the first-stage DLL.

4.1.2. Determine the number of VCDL unit stages

The number of VCDL unit stages should consider three aspects: power, phase noise, and achievable delay range. The power refers to the size of the VCDL. The phase noise directly trades with the power consumption. Since the phase noise from the first-stage DLL will be filtered by the third stage, the attention on the power and phase noise aspects can be reduced.

The achievable delay range refers to the delay range from VCDL's input to the output. This delay range is modified by the VCDL's supply voltage provided by the LDO. The black curve in Fig 4.2 shows the simulation results of a VCDL unit stage's delay time by sweeping the control voltage. The control voltage starts from 0.7V, since the inverter will stop working if the control voltage is less than 0.7V. The points of interest in the figure are the delay time range of 'Vctl' from 0.7V to 1.3V. The delay time range of 8 delay unit stages is 0.66ns, which is less than the period of the reference 1GHz clock. Thus, the number of the delay unit is chosen as 16.

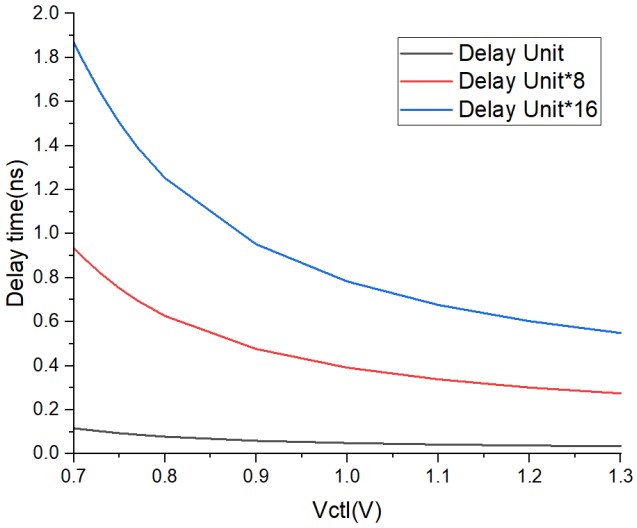

Figure 4.2: Simulation results of VCDL's delay time under different control voltage

Influence of VCDL's non-linear gain

The VCDL's gain refers to how much delay time that a VCDL can generate under different control voltages, which can be derived from Fig 4.2. The VCDL's gain with 16 stages is plotted in Fig 4.3a, which shows a non-linear characteristic. The gain of VCDL varies when the control voltage changes. This non-linear gain affects the loop gain. One example is illustrated in Fig 4.3b. The loop gain in Fig 4.3a shows a 2.5 times difference between its lowest and highest value. The gain of other components in the loop is fixed. The loop bandwidths show a 2.5 times difference as well.

4.1.3. Current-model Logic XOR logic gate

The CML-XOR gate serves as the phase detector for quadrature error detection. The reason to utilize CML-XOR logic gate instead of the normal XOR gate is that the normal XOR gate can't work properly at high speed. The XOR's output signals' frequency doubles the input frequency, which makes it challenging for the normal XOR to rise to VDD and fall to VSS.

Fig 4.4 exhibits the schematic of the CML-XOR gate. Note that it requires two quadrature phase inputs and their reverse phases. Thereby the phases, $CLK45°$, $CLK135°$ and their reverse phases are selected for quadrature error correction. Generating the reverse phases with inverters creates skew and worsen the quadrature error detection performance.

On resistance of the load PMOS determines the upper rail. The size of the input NMOS and the NMOS current sources determines the lower rail. The output swing is determined by the pull-up strength of the load PMOS and pull-down strength of the NMOS. A larger swing leads to a higher XOR detection gain at the cost of a higher dynamic power consumption. The PMOS and NMOS current sources have four identical pieces. This configuration ensures symmetry during the layout design. It's worth noting that any asymmetry will lead to a quadrature detection offset, which worsens the linearity performance.

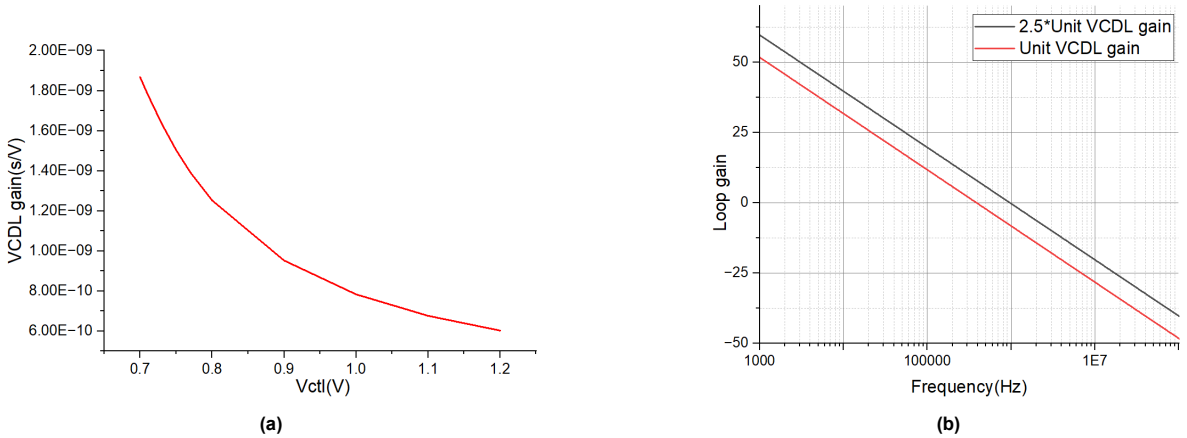

Figure 4.3: (a)Implementation of a typical delay locked loop(b)DLL's loop gain under two VCDL gain

The outputs of the XOR gate are two differential signals. The detected quadrature error information is indicated by the duty cycle of the output signals. The post LPF extracts the average value out of XOR's output signals. The quadrature error is zero when the average values of XOR_N and XOR_P are the same.

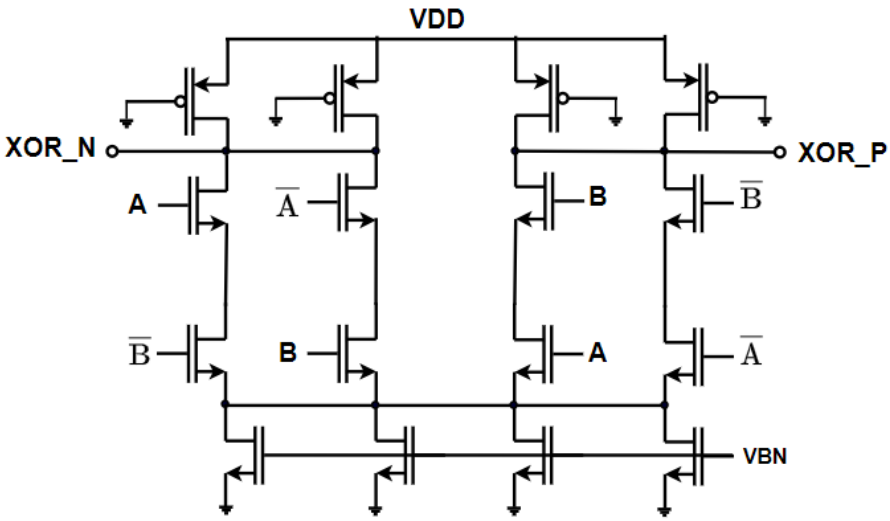

Figure 4.4: Schematic of the CML XOR in the proposed design

4.1.4. Active Low Pass Filter

The LPF behind the XOR gate extracts the average value out of XOR's output signals. The reason to use an active LPF is that the XOR's output signals' swing is between 'VDD' and 'VDD-0.2V'. Extracting the DC value of these two signals leads to the two input transistors of the OTA behind the LPF biased in the triode region. This reduces the gain of the OTA and the total loop gain. Common mode feedback is utilized in the active LPF to ensure the OTA's input transistors can be biased in a saturation region. This is also the reason why the differential XOR gate is used: the common mode feedback will not change the relative average value difference between XOR's output signals. Another advantage of utilizing an active LPF is that the active LPF has a non-0dB DC gain, which means the phase detection error coming from the XOR gate will be enlarged. This enlarged error enhances the operation of quadrature error correction.

Fig 4.5 shows the schematic of the Gm-C active LPF with common mode feedback(CMFB). The CMFB is realized by the left branch. The size of the current source in the left branch is proportionally smaller than that in the right branch. The other transistors are proportionally sized as well. The CMFB

ensures the output voltage level is around the input voltage 'VCM'. The value of the capacitors C1 and C2 is identical. AC simulations should be conducted to ensure the pole frequency of the active LPF is higher than the DLL's loop bandwidth.

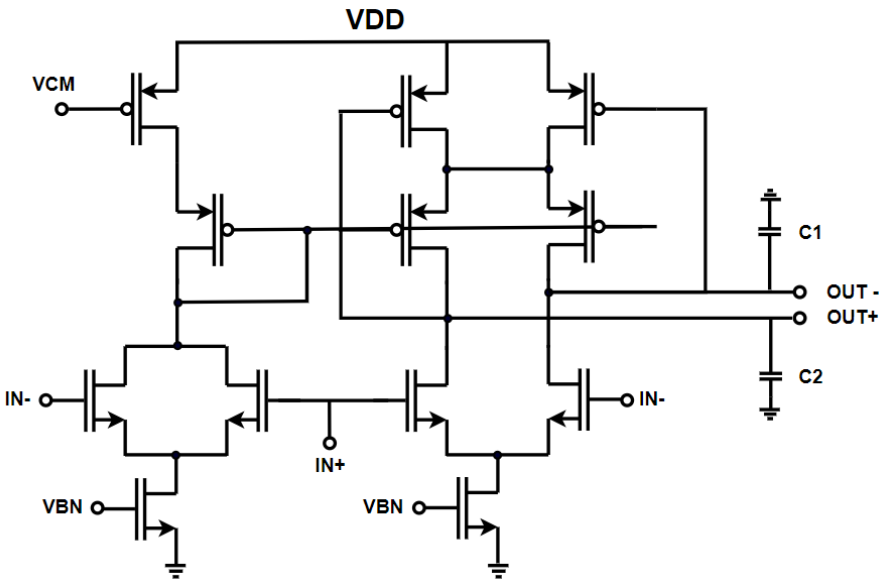

Figure 4.5: Schematic of the active LPF in the proposed design

4.1.5. Operational Transconductance Amplifier

The primary function of the OTA is to provide gain in the loop and converts the differential dc voltage difference of the LPF into a current. The integrated current generates the control voltage for the VCDL.

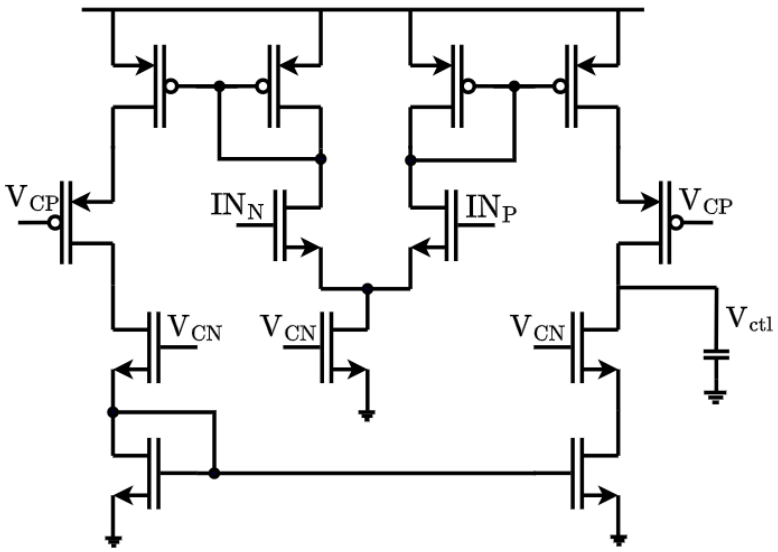

Figure 4.6: Schematic of the OTA in the proposed design

Fig 4.6 exhibits the schematic of the OTA. The cascode output stage ensures a high gain. The reason why the two-stage technique is not used is that this structure provides better symmetry. The capacitance of the capacitor for integration has a low limit. This is because the output resistance of the OTA isn't infinitely large, which makes the OTA's pole not at DC. The OTA's pole frequency should be low enough to ensure enough phase margin for the DLL.

4.1.6. Low Dropout Amplifier

The design of the LDO requires much attention as the LDO itself forms a control loop. The input voltage signal of the LDO is the integrated output voltage of the OTA. The function of the LDO is to provide the corresponding supply voltage level for the VCDL to adjust its delay time and provide currents for the VCDL to operate.

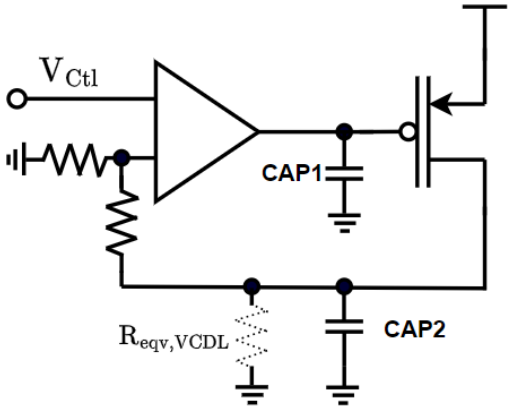

Figure 4.7: Schematic of the LDO in the proposed design

Fig 4.7 exhibits the schematic of a LDO. The loop contains an amplifier, a power PMOS transistor, two resistors, and two capacitors. The amplifier provides enough loop gain to ensure the output voltage is correctly amplified by the two resistors' ratio. The power PMOS supplies the currents for the VCDL. The two capacitors are utilized to adjust the LDO's loop bandwidth. 'CAP2' is also used to reduce the kickback from the VCDL. The dynamic currents drawn by the VCDL are visualized as an equivalent shunt resistor at the LDO's output node.

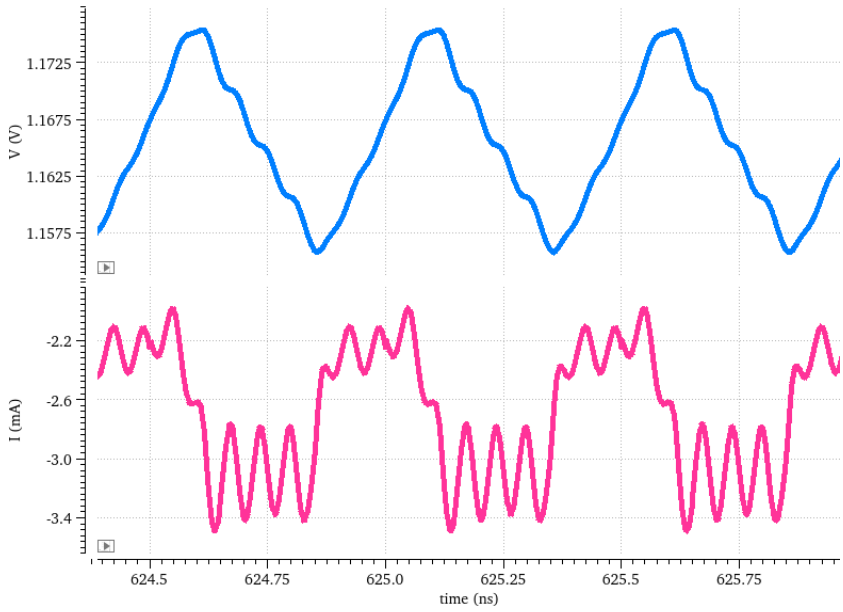

Figure 4.8: The LDO's output voltage and current profile with one LDO in closed-loop simulation

There are two poles in the LDO loop. The first pole locates at CAP1's node. Its frequency depends on the power of the PMOS's gate capacitor, the amplifier's output resistance, and CAP1. The second pole depends on CAP2, the VCDL's equivalent shunt resistance, the two resistors, and the PMOS' small signal on resistance. To ensure the LDO works properly, the frequencies of the two poles need to be arranged. One pole is selected as the main pole of the LDO loop, which needs to be put at a low frequency. The other pole needs to be put higher than the LDO's loop bandwidth to ensure

enough phase margin. Due to the small resistance of VCDL's equivalent shunt resistor(high power consumption), it requires an extremely large CAP2 to put the second pole at a low frequency. Thus, the pole at the CAP1 node is selected to be put at a low frequency. The high output resistance of the resistor makes it feasible to create a low-frequency pole without applying a large capacitor. The amplifier used in the LDO is of the same structure in Fig 4.6. Note that there exist poles at the current mirror nodes of the amplifier. The transistors of the current mirror need to be sized to put the poles at a higher frequency than the LDO's loop bandwidth.

4.1.7. The two LDO strategy

It can be observed from the whole DLL's circuit implementation in Fig 4.1 that two LDO amplifiers are utilized. These two LDO share the same control voltage and resistor ratio. The application of two LDO increases the power consumption due to the use of two amplifiers. However, it's necessary to apply two LDOs to ensure the output phases' linearity. One LDO supplies the currents for the input buffer stages of the VCDL and the dummy stages at the last of the VCDL.

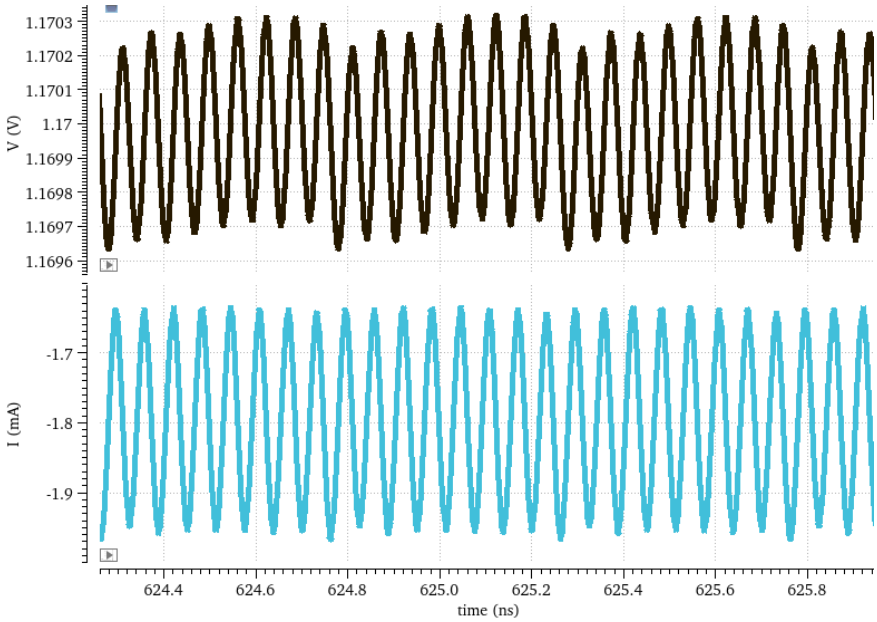

Figure 4.9: The LDO's output voltage and current profile with two LDOs in closed-loop simulation

The main reason to apply the two LDO strategies is to reduce the influence of VCDL's kickback effect. Fig 4.8 illustrates the effect of VCDL's kickback. The results are derived from the DLL's closed-loop simulation with only one LDO. The upper curve shows the output voltage of the LDO. The lower curve shows the LDO's output current profile. Based on the phase domain model, the delay time of the VCDL is determined by the LDO's output voltage level times the VCDL's linearized gain. The output voltage level of the LDO shows a 20mV peak-to-peak waveform, which results from the unbalanced current profile. The rise and fall transition of the VCDL's each phase draws current from the LDO's output capacitor. The power transistor of the LDO charges current to the capacitor. This process incurs the wavy shape of LDO's output current profile. However, the large current gap can be observed every 500ps, which results from the input buffer stages and output dummy stages. The input buffer stages include a self-biased inverter stage and two VCDL unit delay stages. The self-bias inverter stage is used to convert the inverter's supply voltage level from VDD to the LDO's output voltage level. The two VCDL unit delay stages ensure the clock transition shape is similar to the shape in the VCDL. The 16 phases of the VCDL ranges are equally spaced between $0°$ and $337.5°$. The phases of the input buffers can be regarded as $315°$, $292.5°$. The phases of the output dummy stage can be regarded as $360°$ or $0°$. More currents will be drawn from the LDO when these phases are reached, which creates the current gap in the current profile.

The two LDO strategy ensures a balanced current profile by utilizing an independent LDO for the VCDL. Fig 4.9 exhibits the simulation results with the two-LDO strategy. The output voltage level of the

LDO shows a 0.625mV peak-to-peak waveform, which is greatly reduced from the 20mV peak-to-peak waveform. Note that the other LDO's output voltage still has a large peak-to-peak waveform, which is verified by the simulation to have a tolerable influence on the VCDL's output phase linearity. There still exists a small voltage gap every 500ps, which results from the static phase error problem. When the DLL is in perfect quadrature condition, the voltage gap will disappear.

4.1.8. Method to predict jitter performance

To simulate the jitter performance of the whole DLL, a closed loop 'pss+pnoise' simulation in the virtuoso tool needs to be conducted. The closed loop simulation is time-consuming, since the loop requires a time corresponding to the loop bandwidth to settle at the stable state. The long-time simulation makes it inefficient to debug or get simulation results under different corners. A more efficient method to estimate the DLL's jitter performance is to utilize the phase domain model by each component's noise times the square of their noise transfer functions(NTF).

Fig 4.10 shows such procedures. The power spectral density(PSD) of the noise current is derived from the 'pss+pnoise' simulation. The NTF is derived from the phase domain model. Note that the NTF in section3.2 considers only one pole from the OTA. However, to get a better match between the simulation results and the calculated results. The poles from the LDO and LPF are also considered. The calculated equivalent phase noise of the DLL loop, contributed only by the noise from XOR,LPF, and OTA, shows a similar curve with the simulation results. This method is utilized in the first stage and third stage of design. In the third stage design, the closed loop simulation becomes infeasible due to the low loop bandwidth.

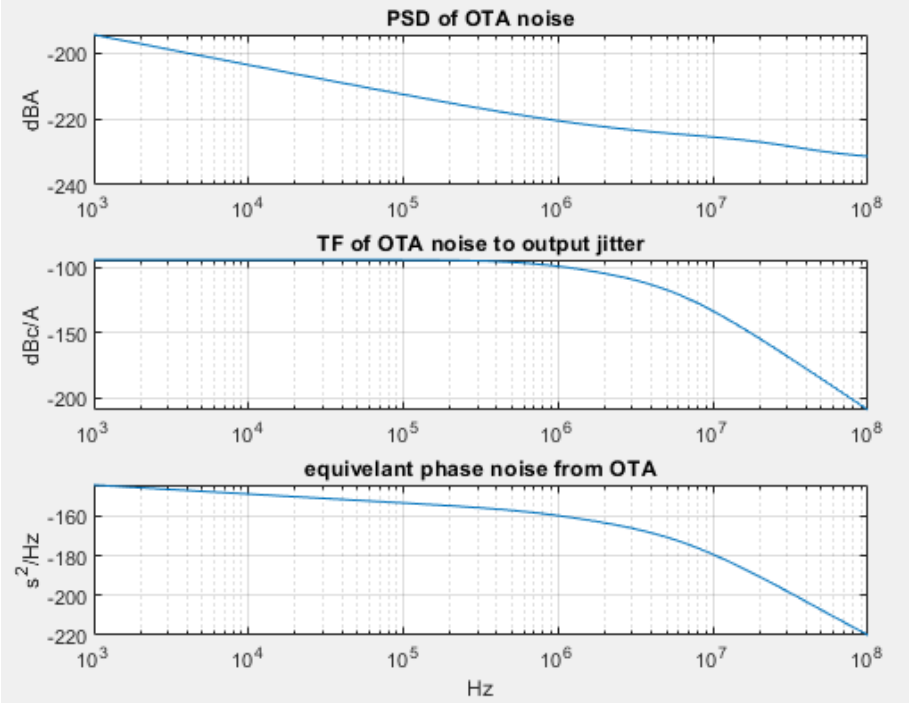

Figure 4.10: Phase Noise contribution of the XOR,LPF,and OTA based on the phase domain model

4.1.9. Method to predict static phase error

The DLL goes into the stable state when the OTA generates a zero integrated current. Thus, open loop simulation of XOR, LPF, and OTA can be conducted to predict the static phase error condition. The simulation set-up is exhibited in Fig 4.11. The inputs of the XOR gate are set such to emulate the clock delay of $45°$, $135°$, $225°$, $315°$. The variable 'Delay' is set as a range centering the ideal delay time of 250ps. The output node of the OTA is connected to a dc voltage source, whose value is set as the control voltage level in closed-loop simulation. The output waveforms to be measured are the input currents of the dc voltage source.

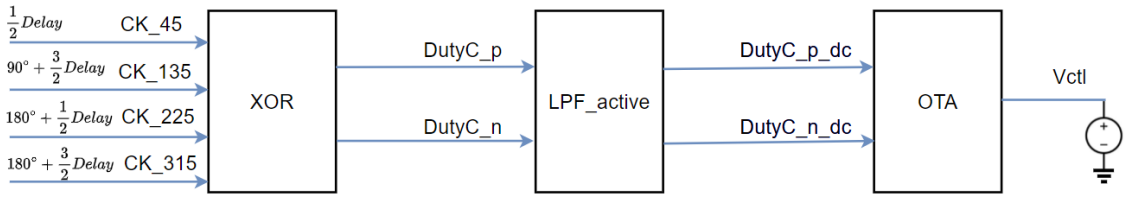

Figure 4.11: Simulation set up for open loop static phase error estimation

Fig 4.12 exhibits the pre-simulation results. The OTA's output currents show a linear relationship with the input clocks' delay offset. The zero current point of the Y-axis, OTA's output average current, refers to the 250.4ps input delay offset, which means a 0.8ps static phase error. The reason why the static phase error is doubled from the input delay offset of 0.4ps is that the QEC corrects quadrature phases, while the VCDL contains two quadrature phases from input to output. The static phase error results of the closed-loop simulation match this open-loop simulation results. This simulation strategy saves time since the closed-loop simulation needs a long time to achieve the stable state.

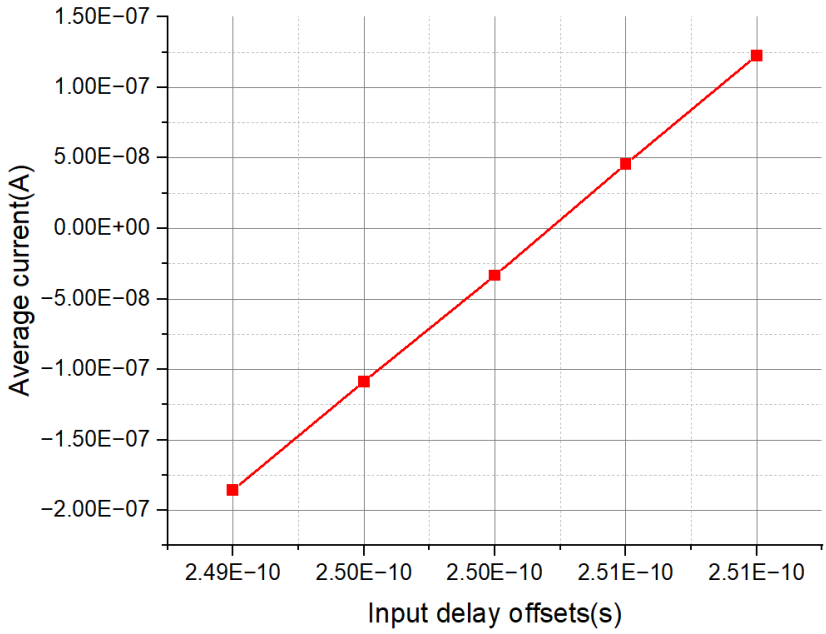

Figure 4.12: Simulation results of the open loop static phase error estimation

4.1.10. Issues that worsen the static phase error problem

A zero static phase error requires a fully-symmetry of the design. Several factors will worsen the QEC effectiveness, which is analyzed in this subsection:

Wiring of the XOR logic gate for phase detection

The XOR gate serves as the phase detector in the first-stage DLL. Any asymmetry in the XOR gate leads to a phase detection offset, which worsens the static phase error problem. During pre-simulation, the signals can be connected by the same wire name. The loading condition of the signal paths for the XOR's four inputs is the same. However, in the layout aspect, the loading condition is changed.

Fig 4.13 exhibits the wiring of the input/output paths in the layout level of the XOR design. The blue line represents 'metal1'. The yellow line represents 'metal2'. This wiring strategy shows less phase detection error compared with other wiring strategies. However, it can be observed that the wiring length of the four input signals is not the same. Each signal path has different resistance and parasitic capacitance conditions. These differences worsen the static phase error problem.

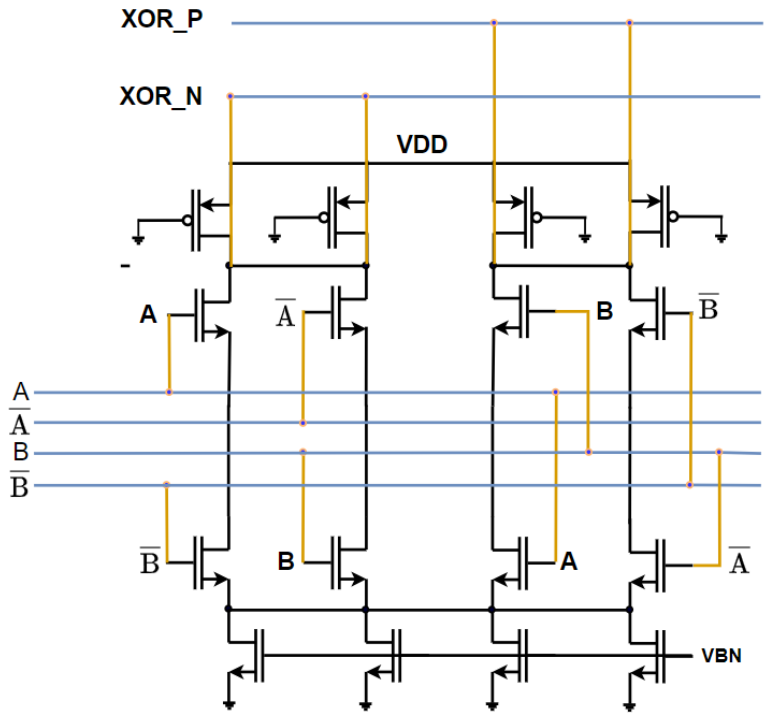

Figure 4.13: The wiring strategy of the signal paths in the XOR's layout design

Wiring between the XOR logic gate and the VCDL

The XOR gate of the quadrature error detection(QEC) detects 4 out of 16 phases of the VCDL. Three dummy XOR gates are required to ensure the 16 output phases of the VCDL have the same loading condition. The wiring between the 4 XOR gates and the VCDL becomes an issue that worsens the static phase error problem. Fig 4.14 exhibits the wiring strategy between the XOR gates and the VCDL. The wiring ensures an almost equal path resistance. However, the parasitic capacitance can't be made equal, leading to the offset in the quadrature error detection.

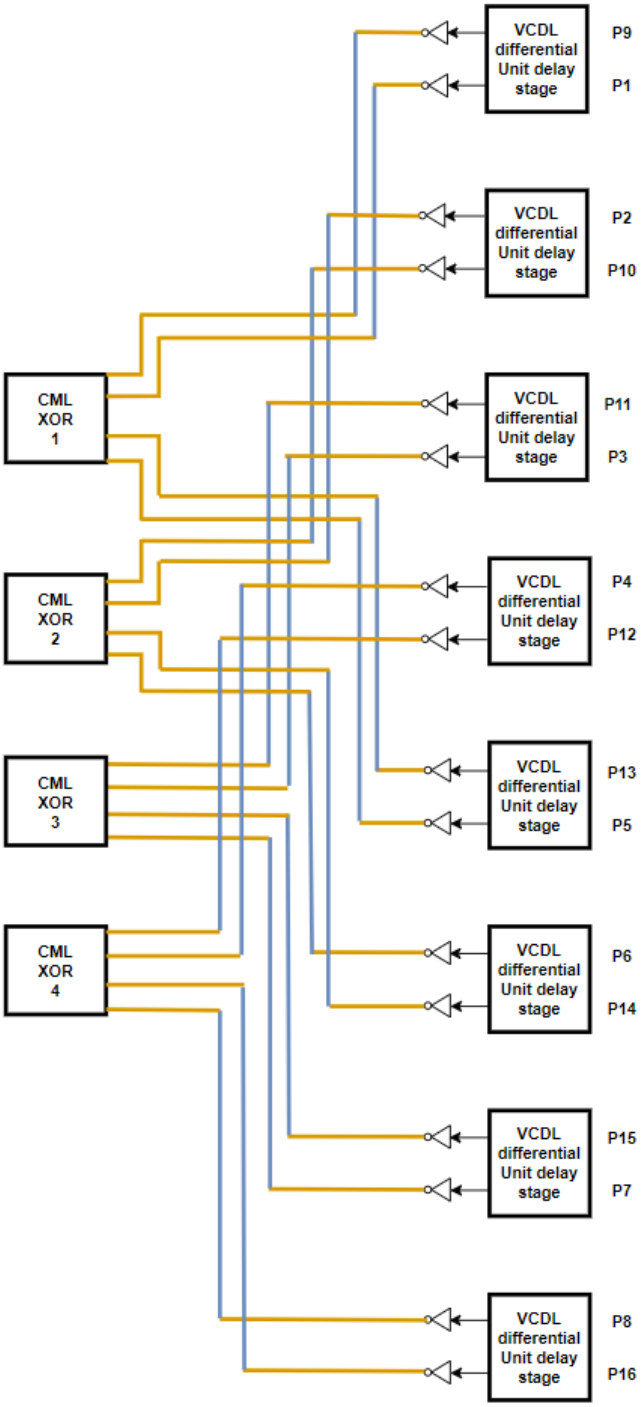

Figure 4.14: The wiring strategy of the signal paths between the four XOR gates and the VCDL

4.2. Interface between the DLL and PI

The number of the first stage DLL's output phases is 16. Two adjacent phases of the 16 phases will be sent to the next stage PI. An interface between the DLL and PI is required to select the two phases for the PI, which is realized by two 16:1 multiplexers(MUX). In this section, the architecture of the 16:2 MUX is introduced. The conventional transmission gate-based MUX has a cross-talk problem that worsens the phase linearity. The MUX structure that can solve the issue is then proposed.

4.2.1. Implementation of the two 16:1 MUX

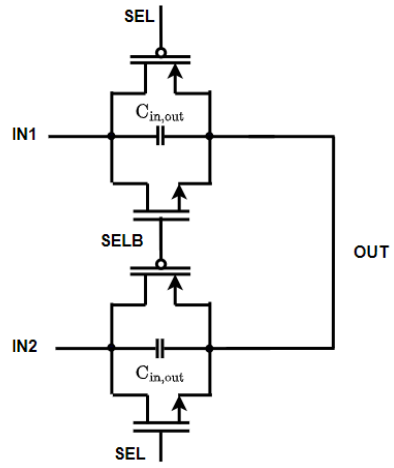

Figure 4.15: Schematic of the conventional MUX

The demonstration of the 16:2 MUXs would take up too much space. To demonstrate the overview clearly, the overview of an interface with two 8:1 MUXs instead of two 16:1 MUXs is depicted in Fig 4.18. The 8:1 MUX is constructed by three cascade stages of the 2:1 MUXs. The number of 2:1 MUX units in each stage is binary distributed. The different colors of the MUX in one stage indicate the difference in their selecting signals in Fig 4.18. The wiring of two colors represents a different metal used in the layout aspect design. Their length should be carefully arranged to ensure an equal path resistance since the unit resistance of metal one and metal two is different.

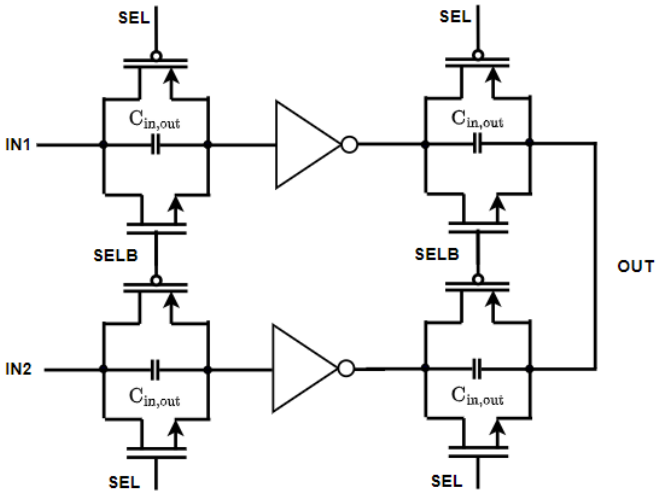

Figure 4.16: schematic of the MUX with internal buffers

Conventionally, an 8:2 MUX has three stages and requires six selecting signals. However, it can be observed from Fig 4.18 that eight selecting signals are applied in the design. Two extra selecting signal

is utilized at the first stage to ensure an equal loading for the VCDL of the first stage. Table4.1 exhibits the digital control codes for the two 8:1 MUXs. The 1-to-4 codes refer to one 8:1 MUX that outputs the first phase of the DLL. The 5-to-8 codes output the second phase. The '0' and '1' refers to the selected upper or lower path. Note that the first stage MUXs' input is connected directly to the VCDL of the first stage DLL. Thus, the loading of the VCDL would become unequal when the digital code 1 and 5 are the same. The red-emphasized code refers to the special cases where the loading for the VCDL wouldn't be equal without the extra control signal. In other cases, the fourth and eighth digital control codes are equal to the first and fifth codes. In special cases where Phase 4 and 5 or Phase 1 and 8 are required, the MUXs need to select the lower or upper paths at the same time, which creates unequal loading for the VCDL.

Selected phase 1	Selected phase 2	Selecting code 1 to 4	Selecting code 5 to 8
Phase 1	Phase 2	0110	1011
Phase 2	Phase 3	1011	0100
Phase 3	Phase 4	0100	1001
Phase 4	Phase 5	1000	0111
Phase 5	Phase 6	1111	0010
Phase 6	Phase 7	0010	1101
Phase 7	Phase 8	1101	0000
Phase 8	Phase 1	1000	0111

Table 4.1: Digital control code for the two 8:1 MUXs

The actual digital control code implementation for the two 16:1 MUXs follows the same logic. It would have four stages and 10 selecting signals.

4.2.2. Implementation of a 2:1 MUX

Fig 4.15 exhibits the schematic of a basic 2:1 MUX, which is basically two transmission gates. The selecting signal enables the passing of one of the two signal paths. This simple structure encounters a linearity problem due to the cross-talk capacitor between the input and output nodes. The size of the transistors are 2um/180nm(PMOS) and 1um/180nm(NMOS). With this size, the capacitance of the cross-talk capacitor is 0.2aF. The capacitance is very small but still leads to an intolerable linearity error(larger than 1LSB).

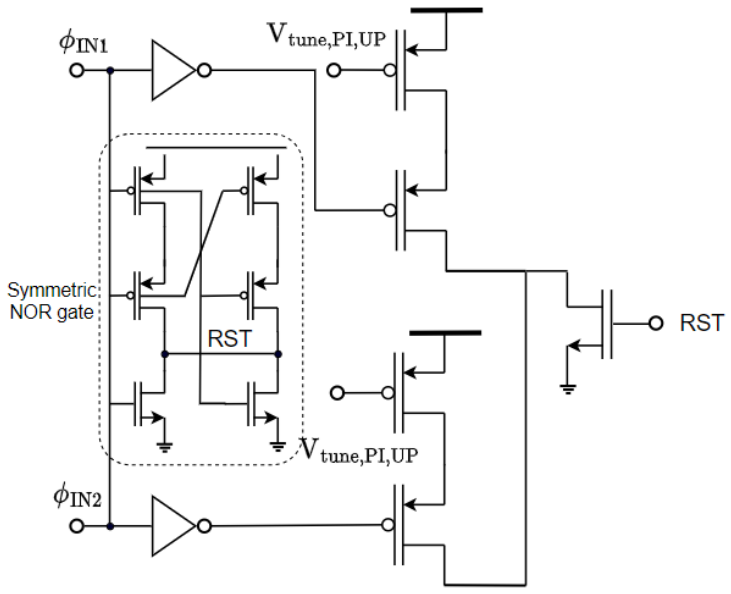

Figure 4.17: Schematic of the PI unit A

There are two methods to address this issue. The first method is to proportionally increase the width and length of the transistors, which leads to a longer distance between the input and output node to reduce the cross-talk capacitor. From the simulation, the cross-talk capacitor of the doubled-size transistors still has an intolerable linearity problem. A possible solution is to add a shield between the input and output nodes at the layout level. The implement ion of the shield refers to a metal bar connected to VDD or VSS that is placed between the input and output node(on the gate of the transistor at the layout level). This method increases the parasitic capacitance for the MUX signal path but reduces the influence of cross-talk. Due to the minimum width requirement of metal one and the minimum space between two metal ones, the size of the transistor can't be set as the minimum size to pass the design rule check(DRC).

The problem with the proportionally increased size method is that the drive capacity isn't increased. From the simulation, the rise and fall edge transition of the clock signal can't be realized properly due to the limited drive capacity of the MUX. To solve this issue, one inverter buffer is applied in the MUX, which provides enough drive capacity for the clock signal to pass.

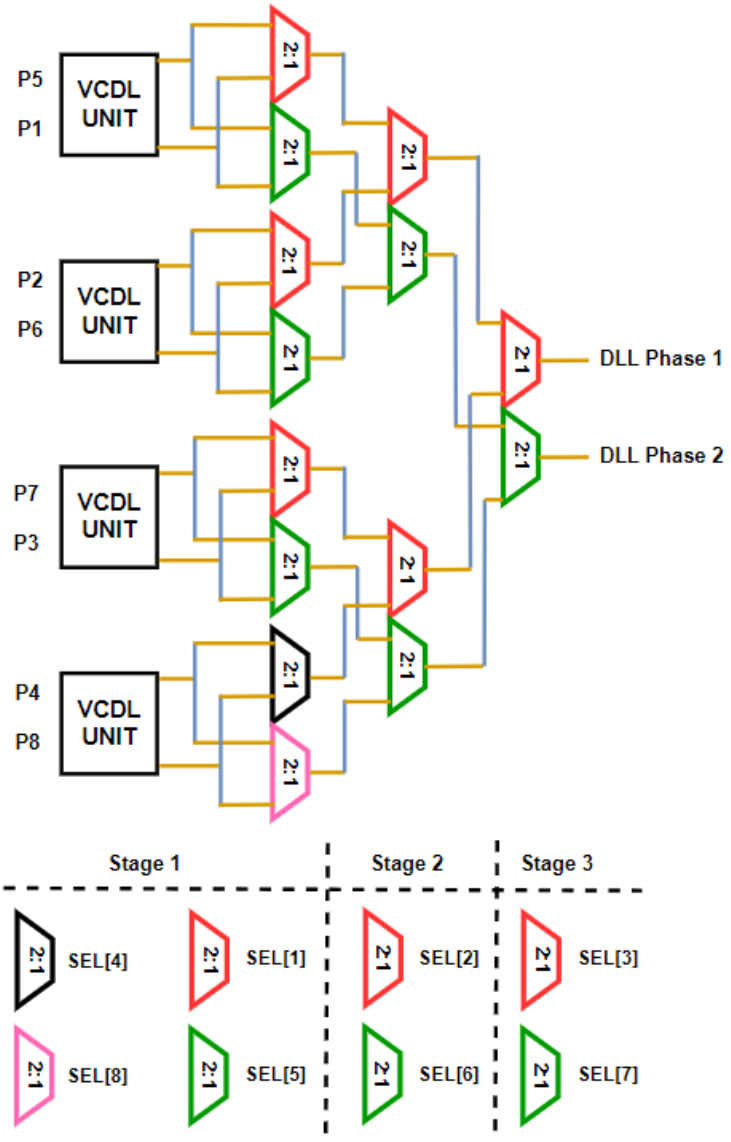

Figure 4.18: Overview of the two 8:1 MUXs

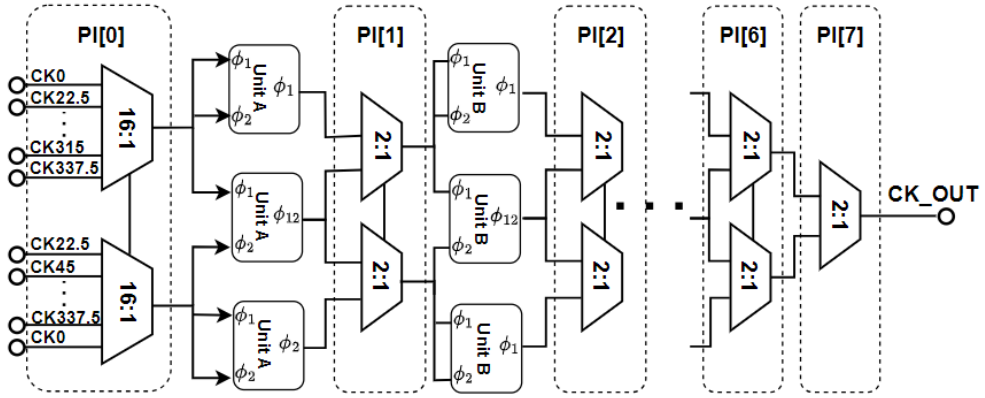

Figure 4.19: Overview of the 6-bit Phase Interpolator

4.3. Phase Interpolator

The job of the phase interpolator stage is to expand the 4-bit phase resolution of the DLL to 10 bit. In this section, the overview of the phase interpolator is introduced first. Then comes the circuit implementation of the two PI units. Thirdly, the implementation of the MUX stages is introduced as well. Finally, the methods to estimate the overall linearity performance of the PI are explained.

4.3.1. Overview of the 6-bit phase interpolator

The traditional PI structure has the problem that its power and area grow exponentially by 2 with the bit number. For example, a six-bit phase interpolator would require 64 PI units. In order to save the power consumption and area, the pipelined PI structure is applied, whose power and area grow linearly with the bit number.

PI Phase	Digital codes	PI Phase	Digital codes	PI Phase	Digital codes
1	000000	23	010110	45	101100
2	000001	24	010111	46	101101
3	000010	25	011000	47	101110
4	000011	26	011001	48	101111
5	000100	27	011010	49	110000
6	000101	28	011011	50	110001
7	000110	29	011100	51	110010
8	000111	30	011101	52	110011
9	001000	31	011110	53	110100
10	001001	32	011111	54	110101
11	001010	33	100000	55	110110
12	001011	34	100001	56	110111
13	001100	35	100010	57	111000
14	001101	36	100011	58	111001
15	001110	37	100100	59	111010
16	001111	38	100101	60	111011
17	010000	39	100110	61	111100
18	010001	40	100111	62	111101
19	010010	41	101000	63	111110
20	010011	42	101001	64	111111
21	010100	43	101010		
22	010101	44	101011		

Table 4.2: Digital control code for the two 8:1 MUXs

Fig 4.19 exhibits the overview of the pipelined PI. The input is the 16:2 MUX, which selects two

adjacent phases from the DLL. Every two path represents the period($360°$) for the PI. Each stage is composed of PI units of one type. The PI unit A stage passes the phase delay information from the rising edge to the falling edge. The PI unit B stage does the opposite job. The upper and lower PI unit aims to transfer the original phase information of the two input clocks. The middle PI unit aims to generate an intermediate phase between the two input clocks by phase interpolation. The 2:1 MUXs select two of the three clocks to the next PI stage. For example, when the 16:2 MUX inputs $0°$ and $22.5°$, the first 2:1 MUX stage can be selected to output $0°$ and $11.25°$ or $11.25°$ and $25°$.

The 2:1 MUXs use a similar structure to the one in the interface between the DLL and PI stages. However, the 2:1 MUX stages between each PI stage will not reverse the edge phase. Each PI unit stage can provide a one-bit resolution. The PI unit A and B stages are placed in an alternative order. 3 unit-A stages and 3 unit-B stages are required to realize the 6-bit resolution. Table.4.2 shows the digital control codes for the 6-bit PI. The number '0' enables the upper path of the two 2:1 MUX at each stage. The number '1' enables the lower path. Note that the last phase '64' is not necessary. This is because of the phase rotation property of the PI, which means the last phase of the PI's period equals the first phase of the PI's next period from the DLL.

4.3.2. Implementation of PI unit

There are two types of PI units applied in the design. Fig 4.17 exhibits the schematic of the PI unit A. The two input clocks are firstly sent to two inverters and a NOR logic gate. When the phase information is carried on the rising edge, the information is transferred to the falling edge after the inverter, which triggers the post-pull-up PMOS. The pull-up PMOS transforms the falling edge information into the rising edge. The symmetric NOR gate is applied, which ensures equal loading for the two input clocks. The output of the NOR gate will become high to reset the PI's output voltage from high to low when the two input clocks are both low.

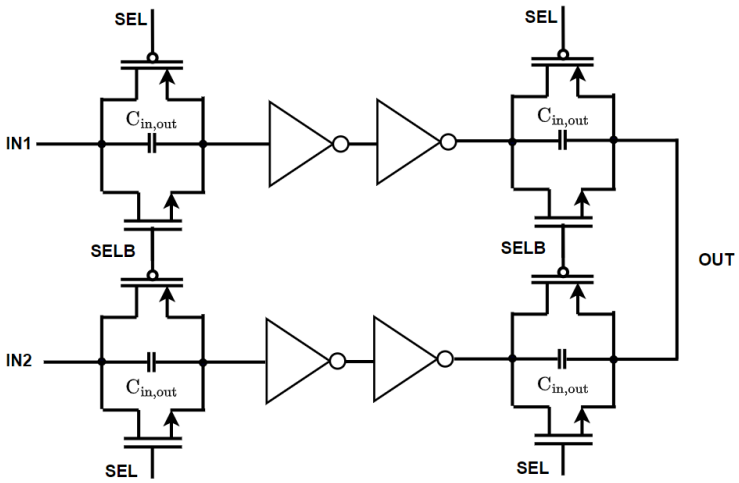

Figure 4.20: Schematic of the 2:1 MUX in the PI stage

It is worth noting that when the pull-up PMOS and reset NMOS are turned on at the same time, a short current path would occur between VDD and VSS. The exact point is when the input clock is transformed from low to high. The NOR gate would output a high voltage when the two input clocks are both low. When the delay time of the NOR gate is longer than the inverter's delay time, the short current path would occur. In order to avoid this short current path, the delay time of the inverter is designed to be slightly larger than the delay time of the NOR logic gate. The '$V_{tune,PI,UP}$' is the control voltage of PI unit A, which can adjust the pull-up strength and the phase interpolation point. Thus, the phase linearity of the whole PI stage can be adjusted.

The PI unit B does the opposite job of unit A, which converts the phase information on the falling edge to the rising edge.Fig 4.22 exhibits the schematic of the PI unit B. To eliminate the short circuit current path, the delay time of the inverters is designed slightly longer than that of the NAND gate. The output of the NAND gate will become high to reset the PI's output voltage from low to high when the two input clocks are both high.

4.3.3. Implementation of the 2:1 MUX

In order to avoid the influence of the coupling capacitor from the input to output nodes, the 2:1 MUX applied in the PI stage has a similar structure to that in the Interface stage. Fig 4.20 exhibits the schematic of the 2:1 MUX utilized in the PI stage. The number of inverters is chosen as two instead of one. Thus, the phase information on the rising or falling edge won't be reversed.

The design will become more power efficient if the number of inverters between two PI unit stages is one. In this case, just one type of PI unit is required.

Linearity problem due to cross-talk capacitors

In this subsection, the influence of the cross-talk capacitors on the linearity performance of the PI is introduced. One simulation example result is exhibited in Fig 4.21. The x-axis is the digital control code and the y-axis is the PI's output clock's cross time of 0.9V. The simulation is conducted when applying the simple transmission gate MUX. Only the first 9 digital codes are simulated but the linearity problem is clear enough to observe. Ideally, the curve should be a straight line. Due to the cross-talk capacitors, the output cross time is not even monotonous. The linearity problem is solved by applying the 2:1MUX structure shown in Fig 4.20.

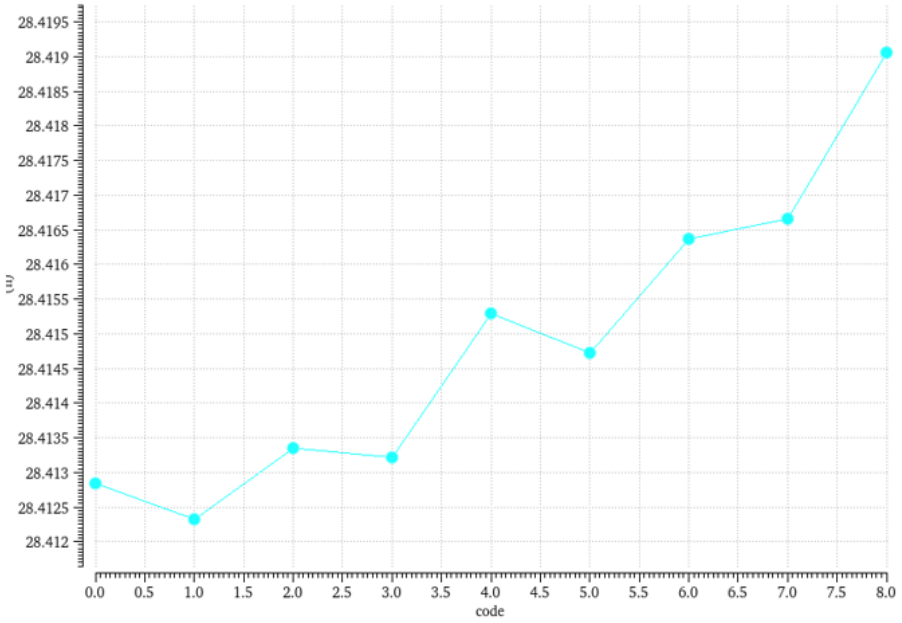

Figure 4.21: Simulation results of the PI's linearity when applying the simple transmission gate MUX

4.3.4. Methods to estimate the overall linearity performance of the PI

The simulation of the PI's linearity requires a sweep of the PI code from 1 to 64. The simulation of one code doesn't take much time compared with the first DLL simulation. This is because the PI is an open-loop system while the DLL is a close loop system. However, the sweep of all the digital codes still takes a long time.

Thanks to the structure of the pipelined PI, the overall linearity performance can be estimated by observing the phase information of a few points. The output phases of the first PI stage are P1, P32, and P64. The tuning of its control voltage will not influence the phase of P1 and P64 but influence the phase of P32. With a higher '$V_{tune,PI,UP}$', the pull-up strength becomes weaker, which leads to a longer time that the output voltage reaches the phase interpolation threshold point. Thus, the phase of P32 will become more close to P64. With a lower '$V_{tune,PI,UP}$', the phase of P32 will become closer to P1. The linearity of the first PI stage affects the linearity of the third and fifth stage, since they are all constructed by PI unit A. On the other hand, the linearity of the second, fourth, and sixth PI stages can be estimated by the phase information of P16. By simulating 3 points, P1, P16, and P32, the overall linearity performance can be estimated.

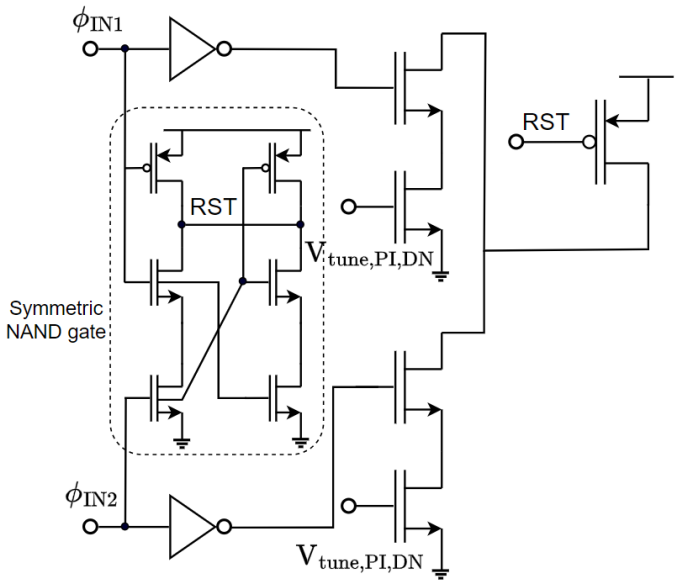

Figure 4.22: Schematic of the PI unit B

Debug Linearity problem

The debugging of PI's linearity problem becomes challenging in the layout level. The virtuoso calibre PEX tool extracts all the parasitic capacitance and resistance. The PI's performance can be fine in schematic level simulation while failing at the layout level. The debugging of the PI follows the below procedures:

- Using PEX tool to extract 'R' and 'C+CC' parasitic. Simulating the results respectively to where the error comes from, resistor or capacitor.
- Assume the problem comes from the parasitic capacitor.
- Add parasitic capacitors between two nodes in the schematic(e.g. IN-OUT; IN-VDD; OUT-VSS)
- Simulating in the schematic level to locate the problem

4.4. Phase noise filter

The phase noise filter is a typical type-2 DLL. In this section, the overview is first introduced. Then comes the implementation of each component.

4.4.1. Overview of the Phase noise filter

Fig.4.23 exhibits the overview of the phase noise filter. The phase noise filter has two input clocks: the output clock from the DLL and PI stages; the reference clock. The reference clock is delayed by the VCDL. The delayed output clock is sent to the PFD to compare the phase difference with the output clock from the DLL and PI stages. The CP and LF convert the detected phase error into an error voltage signal. The LDO adjusts the supply voltage of the VCDL to modify the delay time.

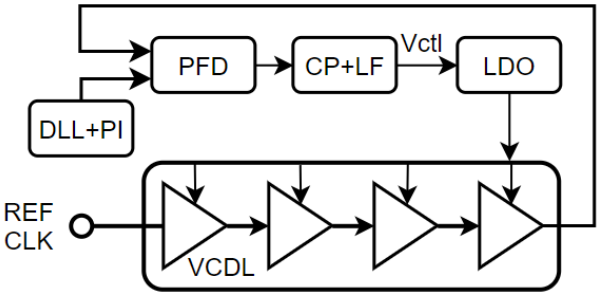

Figure 4.23: Overview of the phase noise filter

4.4.2. Voltage controlled Delay Line

The implementation of the VCDL is similar to that in the QEC-DLL stage. The main difference is that there is no output buffer for the VCDL except for the last stage, while each stage has an output buffer in the QEC-DLL. This is because the DLL with QEC requires an equal loading for each stage to ensure the 16 output phases' linearity. However, the DLL in the filter stage outputs only one phase.

The number of the VCDL stages is chosen as 10, which follows a similar logic introduced in section4.1.2. The delay time of the VCDL should cover 1ns when sweeping the control voltage from 0.7V to 1.2V. When the control voltage is lower than 0.7V, the inverter wouldn't work properly. When the control voltage is larger than 1.2V, the power PMOS transistor wouldn't work in the saturation region.

4.4.3. Phase frequency Detector

The phase frequency detector(PFD) applies the classic TSPC logic based structure, which inherits the high-speed PFD design in [4]. The working principle is explained by the timing diagram shown in Fig.4.25. This PFD detects the rising edge of the two input clock signals. Initially, both A and B are low while the UP and Down signal are high. Thus, the Reset signal generated by the NOR gate is low. The node X is charged to VDD. The rising edge of A occurs earlier than that of B and stops the charging to the node X. The UP signal changes from high to low due to the turn-on of M1. The rising edge of B does a similar job and causes the reset signal to become high.

The PFD converts the phase difference between the two clocks into a duty cycle error. A is the reference clock and B is the output delayed clock of the VCDL. The UP signal in Fig.4.25 shows a larger duty cycle, which will cause the charging time in the CP longer than the discharging time. Thus, the control voltage will rise and decrease the delay time of the VCDL to align the rising edge of A and B.

Noise consideration

The phase noise resulting from the PFD at low frequency is dominant by the flicker noise of the PFD while the dominant phase noise source at high frequency is the PFD's white noise. From the phase domain model, the phase noise resulting from the PFD will be low pass filtered by the loop bandwidth. Thus, the resulting phase noise at high frequency can be filtered effectively. The low-frequency phase noise requires more attention. The flicker noise can be decreased by proportionally increasing the width and length of the transistors. The white noise can be decreased by increasing the power consumption.

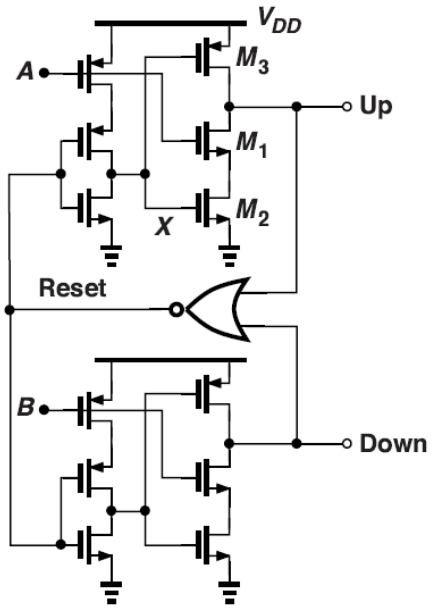

Figure 4.24: Schematic of the Phase frequency detector in the phase noise filter

Dead zone problem

The dead zone problem refers to the time that UP and Down are both low, which is indicated by 'T_{RST}' in Fig.4.25. Ideally, the reset signal should be high immediately. However, the delay time for the NOR logic is unavoidable. This dead zone time will influence the CP in the next stage, which will cause the transistor of charging and discharging to turn on at the same time. The mismatch between the charging and discharging currents will worsen the static phase error problem. However, the static phase error will not become a problem in this design as long as the phase error is constant.

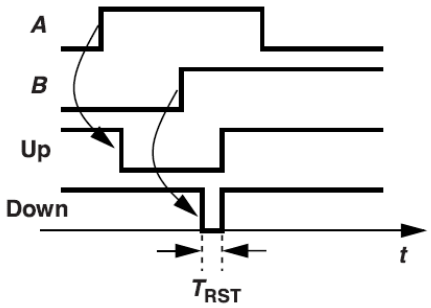

Figure 4.25: Timing diagram of the Phase frequency detector

4.4.4. Interface between the PFD and CP

The duty cycle difference between the Up and Down signals will adjust the charging and discharging time of the CP in the next stage. The charging process is controlled by PMOS while the discharging process is controlled by NMOS. An interface is required to convert the low-active duty cycle of the Down signal into the high-active duty cycle.

Fig.4.26 exhibits the schematic of the interface. Due to the structure of the CP in the next stage, the control signals and their reverse signals are generated. The UPb signal turns on the charging transistor. The DN turns on the discharging transistor. Note that there exists a skew between the four control signals. This skew can be reduced by adding a transmission gate in the signal path. However, the transmission gate would change the output signals' rising and falling patterns, which increases the static phase error. Although in this design, the magnitude of the static phase error is not important. The transmission gate is still not added to save power and area.

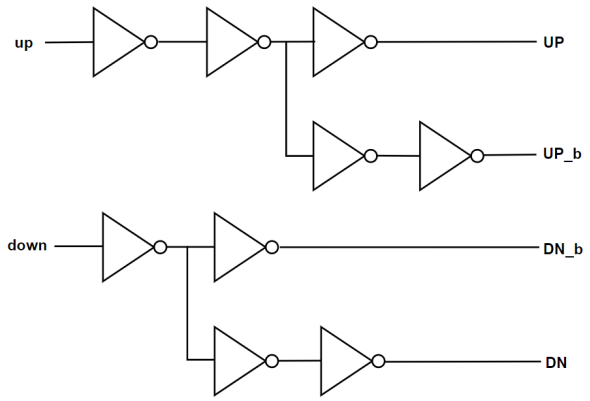

Figure 4.26: Interface between the PFD and CP

4.4.5. Gate-switch cascode Charge Pump and loop filter

The function of the CP is to convert the duty cycle error from the PFD into a current. When the phase difference is zero ideally, the current becomes zero. This current is integrated into the loop filter to generate the control voltage for the VCDL. The loop filter in this DLL design is a simple capacitor. Fig.4.27 exhibits the schematic of the CP in the proposed phase noise filter. The design is modified from the gate-switch CP in [4]. The modification is the cascode output to ensure a high output impedance. During the operation of the phase noise filter, the output control voltage of the CP changes corresponding to the digital control codes. The control voltage determines the bias situation of the output transistors and changes the static phase error. To ensure a constant static phase error, the gate-switched cascode CP is utilized.

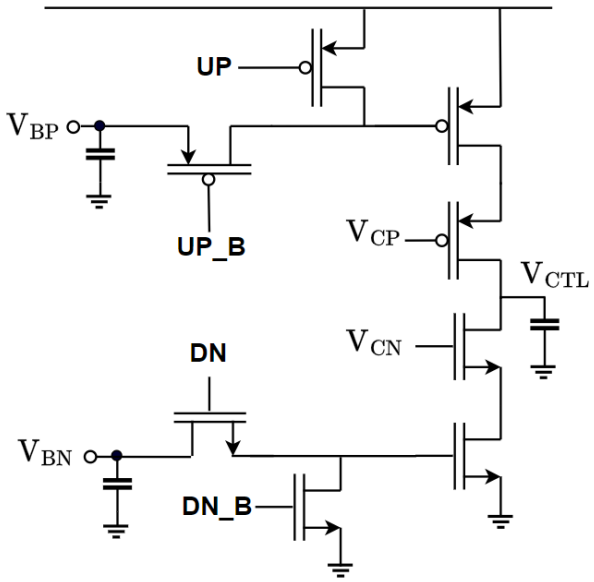

Figure 4.27: Schematic of the charge pump in the phase noise filter

There are three types of CP introduced in [4], including source-switched, drain-switched, and gate-switched. The reason to choose the gate-switched structure is the high output impedance that it offers. It has been tested by simulation that the output impedance of the two-transistor cascode structure is not enough to ensure a constant static phase error for the source-switched and drain-switched structures. This is because the output transistors will be in the triode region during the rising and falling edge of the UP and DOWN signals.

4.4.6. Low Dropout Amplifier

The LDO applied in the phase noise filter has the same structure as that in the first stage DLL with QEC. The main difference is the two capacitors' value. From the phase domain model, there should exist only one pole within the loop bandwidth to ensure enough phase margin. This main pole is selected as the loop filter's pole at DC. Thus, the pole of the LDO needs to be put higher than the loop bandwidth. Although the DLL is viewed as a linear system during analysis, the nonlinearity of the VCDL's gain makes the loop a nonlinear system. The VCDL's linearized gain at the lowest VCDL control voltage is three times larger than the gain at the highest VCDL control voltage. As a result, the loop bandwidth will change three times when the control voltage varies from its lowest to the highest value. The lowest bandwidth is 160kHz. Thus, the bandwidth of the LDO is set at 1MHz to ensure the DLL's stability.

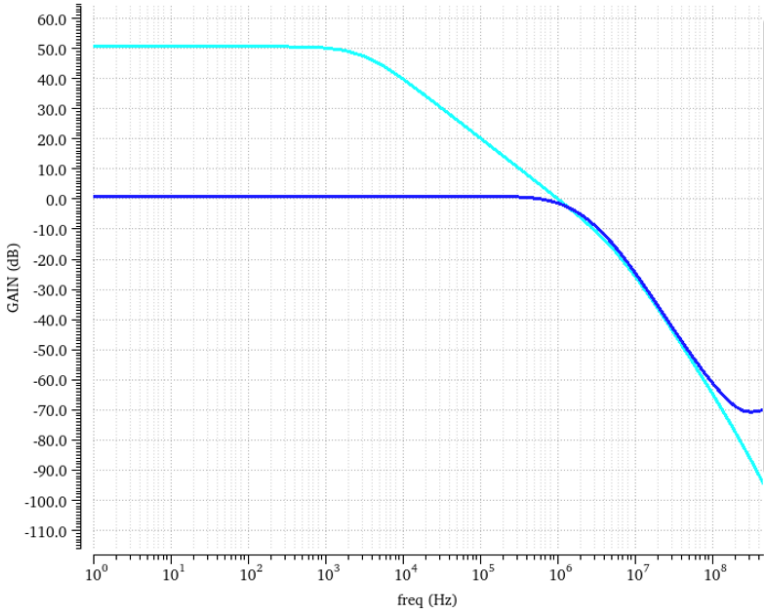

Figure 4.28: Loop gain and AC response of the LDO

Fig.4.28 shows the two pole frequencies in the LDO of the last stage DLL. The blue curve is the loop gain of the LDO. The purple curve is the AC response of the LDO. The first pole of the LDO's loop gain locates at several thousand Hz. The second pole locates at several mega Hz. This configuration ensures that the whole LDO acts as a stable block with a DC gain determined by the resistor ratio and a pole locates at 1MHz. However, two large capacitors (201pF and 203pF) are utilized to ensure these two poles' frequencies. The large capacitors take up a large amount of area of the final layout.

It's worth noting that the DLL would still be stable if the LDO's bandwidth is set at tens of mega Hz. In this case, the value of the two capacitors needs not to be so large. The reason to utilize two large capacitors is that it can reduce the noise bandwidth of the LDO and the VCDL, since the phase noise from the VCDL and LDO dominates the whole design's phase noise at high frequencies.

5 Results

The work stops at the layout and post-layout simulation level. Firstly, the layout view is given. Secondly, the output spectrum of the phase noise filter is exhibited. Thirdly, the power breakdown for each component is exhibited. Fourthly, the simulation results of two design trade-off relationships are exhibited. Fifthly, the linearity performance is exhibited. Finally, the results of this work are compared with a work with a similar structure.

5.1. Layout

Fig.5.1 shows the layout view of the design. The total area is 940 µm × 955 µm. The total active area is 695 µm × 693.5 µm. The number of active I/O pins is 28, including 7 digital control pins for the PI, 10 digital control pins for the DLL, and 6 analog voltage control pins for the PI.

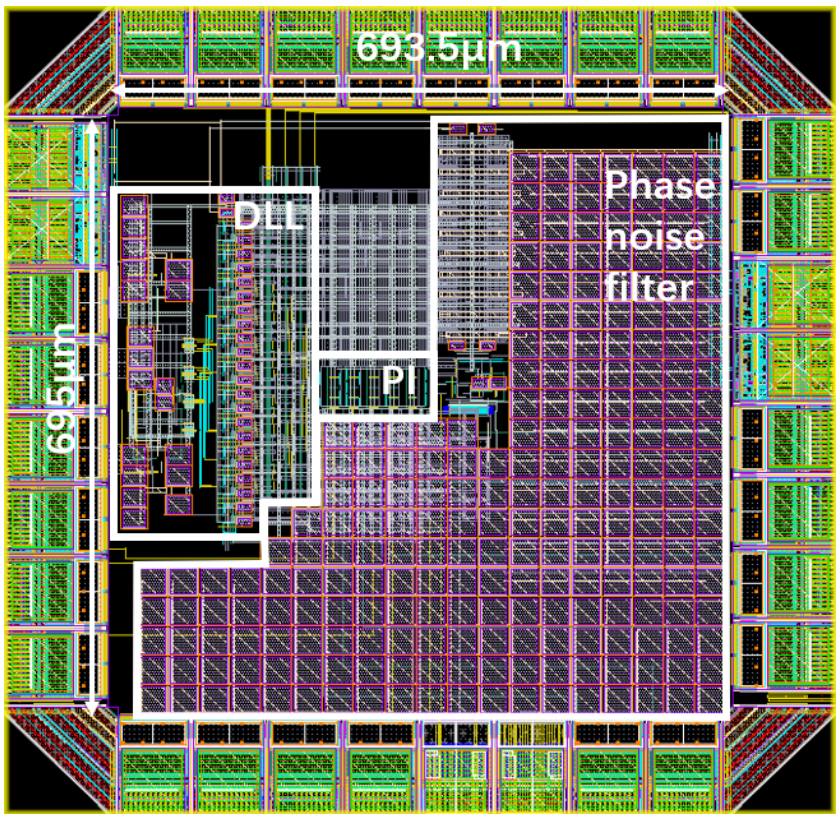

Figure 5.1: Overview of the phase noise filter

To control the pull-up and pull-down strength of the PI unit A and B in the pipelined PI stage, two

analog control voltages are required. However, there are six analog control voltages used in the design. The reason to utilize six control voltages is to compensate for the influence of mismatch. Thus, six PI unit stages are assigned with six independent control voltages. A large area is consumed by the two 200pF capacitors of the two LDO in the phase noise filter. These large capacitors ensure a low noise performance for the VCDL and LDO. The loop filter capacitor of the phase noise filter is externally connected, which makes it possible to adjust the noise suppression pole of the filter.

5.2. Phase noise spectrum of the phase noise filter

Simulation Set-up

The simulation results are derived when the noise suppression pole of the phase noise filter is put at 160kHz, which requires a 100pF loop filter capacitor. Two 1GHz reference clocks are input to the system with a 1.8V power supply. The corner is set at TT 27°C. The virtuoso PEX tool is utilized to extract the 'R+C+CC' parasitic. The transient simulation is conducted first to get the time when the DLL loop reaches the stable state(Although the bandwidth of the system is known, the time for initialization is unknown). This time is then utilized in the 'Pss+Pnoise' simulation to get the phase noise spectrum of the phase noise filter's output clock.

The design has three stages, including the DLL, PI, and the phase noise filter. The simulation of the whole system is time-consuming, since the DLL and the phase noise filter are both closed-loop systems and requires time to reach the stable state. In order to save simulation time, the unfiltered phase noise from the DLL and PI and the filtered phase noise from the phase noise filter are simulated separately. The phase noise spectrum from the DLL and PI is simulated first. The results are input to the phase noise filter by the 'noise file' property of the 'Vsin' voltage source in analoglib. Two inverters are placed behind the 'Vsin' voltage source to convert the sine wave into the square wave.

There are 1024 (10-bit) results to be derived for the phase noise filter. However, to show the effectiveness of the phase noise filter, the simulation results of one situation are enough.

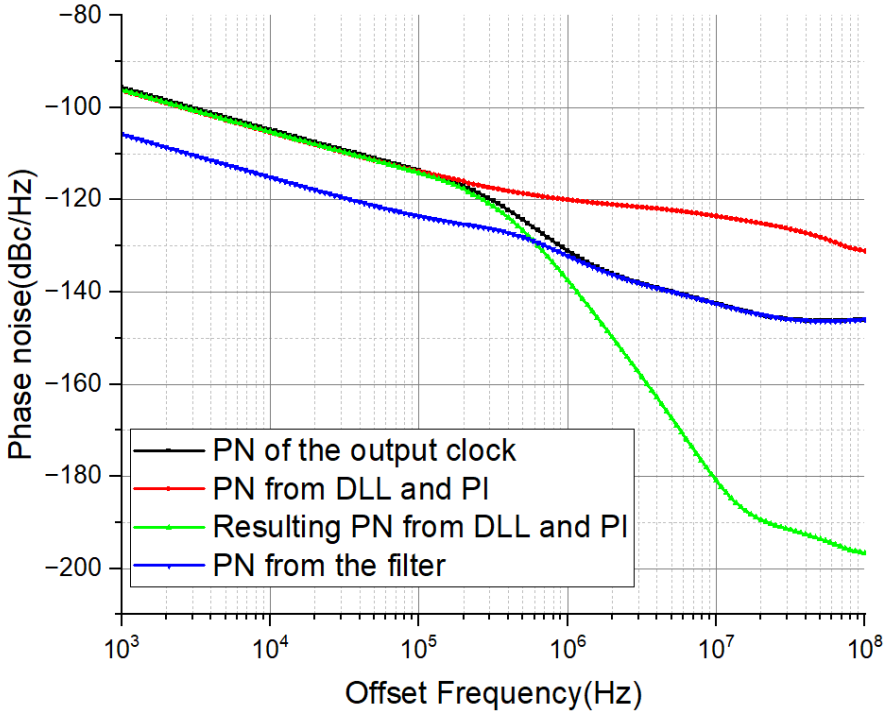

Figure 5.2: Output spectrum of the phase noise filter

Simulation results

Fig.5.2 exhibits the post-simulation results of the phase noise filter in the worst case, where the VCDL's control voltage is lowest and the phase noise is in the worst situation. It's worth noting that the loop bandwidth of the phase noise filter would also change corresponding to the digital control code. With a lower VCDL control voltage, the filter's bandwidth is higher, corresponding to a higher noise suppression pole frequency. The 'PN from DLL and PI' curve is derived from the simulation results of the DLL and PI stage without the phase noise filter stage. The 'PN from the filter' curve refers to the output phase noise when only enabling the phase noise from the phase noise filter. The 'PN of the output clock' curve is the output phase noise spectrum when all the noise sources are enabled. The 'Resulting PN from DLL and PI' curve is the output phase noise spectrum of the phase noise filter when only turning on the phase noise from DLL and PI.

There are two noise sources of the output clock, which are the input clock's noise(from DLL and PI) and the phase noise filter. By comparing the 'PN from DLL and PI' and the 'Resulting PN from DLL and PI' curves, it can be observed that the phase noise from the DLL and PI drops at the rate of -40dBc/Hz from the 160kHz. The integrated jitter(from 1kHz to 100MHz) of the unfiltered DLL and PI's output clock is 1.099 ps_{rms}. The integrated jitter drops to 315.9 fs_{rms} after filtering. The integrated jitter resulting only from the phase noise filter is 188 fs_{rms}. The accumulated jitter from 1kHz to 100MHz is 368 fs_{rms}.

5.3. Power Breakdown

Simulation set up

The power breakdown analysis becomes infeasible after extracting the parasitic with the PEX tool. Thus, the following power breakdown analysis is derived from the pre-simulation. The corner is TT 27°C. The simulation results are derived when the DLL reaches the stable state.

The power breakdown of the design is divided into two parts. This is because the power consumption of the VCDL in the phase noise filter would change based on the digital control codes. The power breakdown of the first and second stages, DLL and PI, is analyzed together while the power breakdown of the phase noise filter is analyzed independently.

5.3.1. DLL and PI

Fig.5.3 shows the power break down of the DLL and PI. The total power consumption of DLL and PI in the post-simulation level is 27.9 mW. The 'VCDL buffer' item refers to the power consumption of the ac-coupling buffer and inverter buffer in the VCDL, whose power supply is VDD. The power consumption of the VCDL delay stages(including dummy stages) is contained in the power consumption of the 'LDO main' and 'LDO buff'.

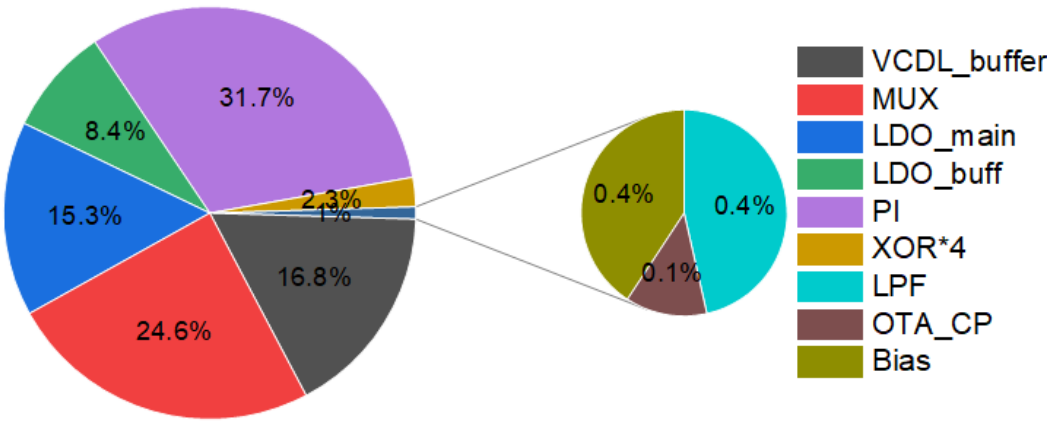

Figure 5.3: Power Breakdown of the DLL and PI

5.3.2. Phase noise filter

The power breakdown of the phase noise filter depends on the chosen digital control code. The digital control code decides the delay time of the VCDL by changing the control voltage. The different control voltage of the VCDL corresponds to different power consumption. The power consumption of the phase noise filter ranges from 14,4mW to 20.7mW (post-simulation results)depending on the chosen digital control code. Table.5.1 is one example of pre-simulation results of the phase noise filter's power breakdown.

Components	Current(uA)
PFD	408
LDO	8600
CP	167
Bias	33
VCDL Buffer	1011

Table 5.1: Digital control code for the two 8:1 MUXs

5.4. Power consumption of the DLL and PI versus the Phase noise filter's bandwidth

The main design goal of the thesis project is to verify the effectiveness of the phase noise filter, while the jitter requirement is left unspecified. The final output jitter of the filter is 368 fs_{rms} while the total power consumption is 42.3mW, where the DLL and PI contribute 315.9 fs_{rms}.

The high-frequency-offset phase noise component of the DLL and PI is filtered by the post phase noise filter, while the low-frequency-offset phase noise is left unfiltered. The low-frequency-offset phase noise of the DLL and PI can only be reduced by increasing their power consumption. The aim of this simulation is to figure out the relationship between the power consumption of the DLL and PI and the final phase noise.

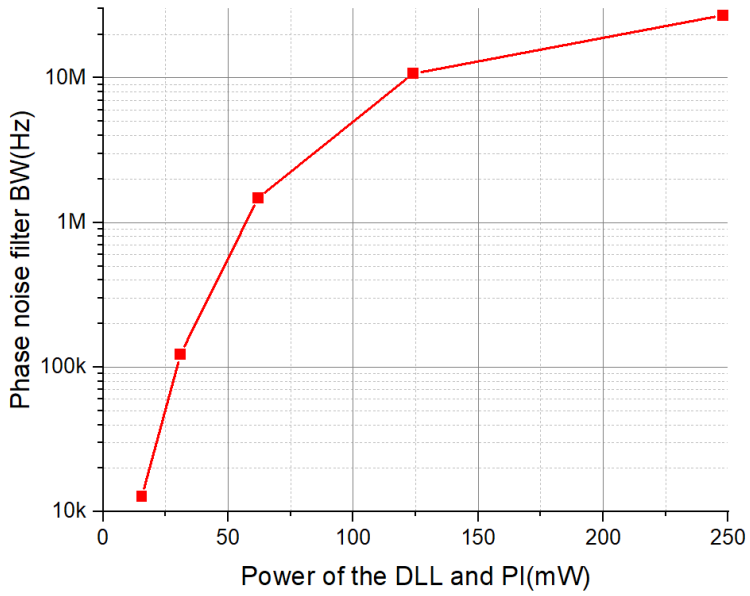

Figure 5.4: Power consumption of the DLL and PI stages versus the phase noise filter's bandwidth

Simulation Setup

The simulation is conducted at the schematic level. This is because the increase of the DLL and PI's power consumption is realized simply by increasing the number of the symbol(e.g. By changing the name in virtuoso to 'i195<1:10>', ten DLL and PI units are parallel-connected, which consumes ten times power consumption). Another reason to simulate at the schematic level is that the power consumption of the DLL and PI can't be extracted from the total system's power consumption after extracting the parasitic with the PEX tool. The selected numbers of the units are 1,2,4,8,16. The phase noise spectrum of these five situations is recorded and sent to Matlab. Matlab applies the transfer function of the phase noise filter on this spectrum, whose noise suppression pole frequency(which is also the phase noise filter's bandwidth) can be adjusted. The pole frequencies are adjusted such that the final jitter maintains a rms jitter of $266fs_{rms}$ under different DLL and PI power situations.

Simulation Results

Fig.5.4 shows the relationship between the phase noise filter's bandwidth and the power consumption of the DLL and PI stages. Each point in fig.5.4 indicates how wide the filter bandwidth should be under a given DLL and PI power consumption to achieve a rms jitter of $266fs_{rms}$.

5.4.1. Bandwidth of the phase noise filter versus the final output jitter

The phase noise filter's bandwidth influence the noise suppression effectiveness of the phase noise at high-frequency offset. The noise suppression pole frequencies can be adjusted by modifying the loop bandwidth of the phase noise filter, which is realized by changing the magnitude of the loop filter's capacitance.

Simulation Setup

The simulation is conducted at the post-simulation level. The phase noise spectrum of the DLL and PI(the unit number is 1) is input to the phase noise filter by utilizing the noise file property of the 'vsin' voltage source. The corner is TT 27°C. The capacitance of the loop filter is chosen such that the loop filter's bandwidths are 1.6k,16k,160k,1.6M, and 16M respectively.

Simulation Results

Fig.5.5 exhibits the relationship between the bandwidth of the phase noise filter and the resulting rms jitter.

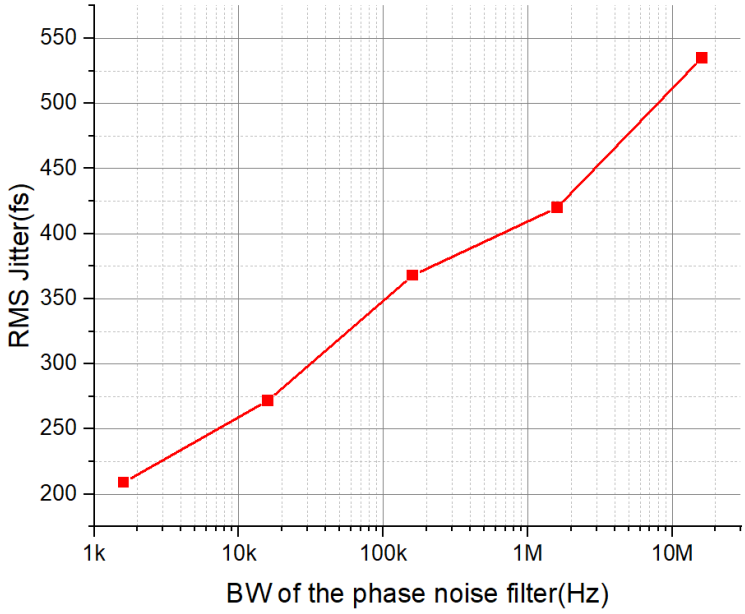

Figure 5.5: relationship between the bandwidth of the phase noise filter and the resulting rms jitter

5.5. Linearity performance

Due to the low loop bandwidth of the phase noise filter, simulating the integral non-linearity(INL) and differential non-Linearity(DNL) performance of the DTC's all digital codes is time-consuming. An alternative approach is to simulate the INL and DNL performance of the DLL and PI's output clock, whose delay time information will be replicated by the phase noise filter. Fig.5.6 shows the post-simulated INL and DNL performance of the output clock produced by the DLL and PI. The INL_{PP} is 2.75 LSB and the DNL_{PP} is 3.42 LSB. The INL and DNL performance will worsen by 0.1 LSB due to the static phase error offset.

The linearity of the first stage DLL with QEC determines the performance of 16 codes 0,64,128,..., 1024. It can be observed from the figure that the main linearity problem comes from the DLL, which results from the asymmetry in the layout design. The linearity performance of the rest digital codes is determined by the PI. It can be observed that the INL and DNL error of the PI is less than 1 LSB, thanks to the two analog control voltages.

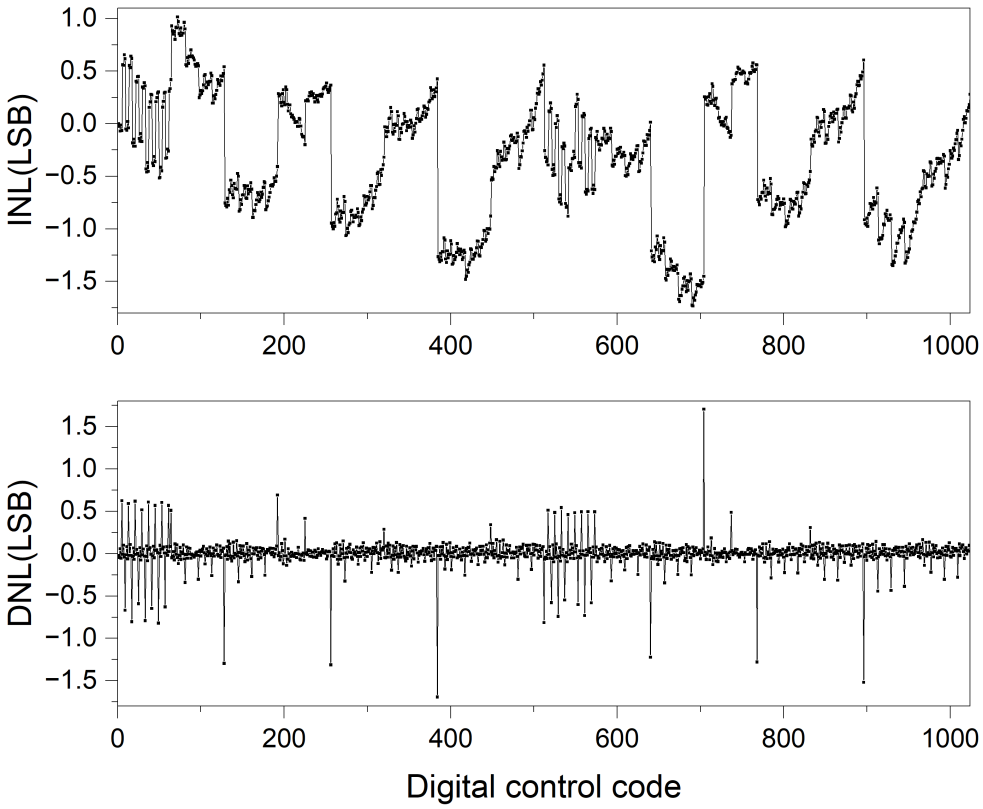

Figure 5.6: INL and DNL performance of the phase noise filter

6 Discussion

In this chapter, the effectiveness of the phase noise filter is examined first, based on the simulation results. Secondly, methods to further improve the jitter performance of the design are discussed, based on the simulation results and phase domain model. Finally, the results of the design are compared with another phase noise filer based on a different working principle. The advantages and the limitation of the proposed design are then discussed.

6.1. Effectiveness of the phase noise filter

The effectiveness of the phase noise filter is proved by the 'PN from DLL and PI' and the 'Resulting PN from DLL and PI' curves in fig.5.2. The phase noise from the DLL and PI drops at the rate of -40dBc/Hz from 160kHz, which is the frequency of the noise suppression pole. Based on the phase domain model, the filter should provide first-order low-pass filtering. The first pole is provided by the loop bandwidth 160kHz. The second pole is provided by the LDO bandwidth 1MHz.

6.2. Methods to improve jitter performance

From the phase domain model and the simulation results, the final output phase noise at high frequencies is dominant by the VCDL and LDO, which is high-pass filtered by the feedback loop. The phase noise at low frequencies is dominant by the PD and CP, since their phase noise is low-pass filtered.

The final output jitter of the filter is 368 fs_{rms} with 42.3mW power consumption. This jitter performance can be improved by taking methods to reduce low-frequency and high-frequency phase noise.

6.2.1. Reduce Low-frequency-offset phase noise

From the simulation results, the dominant low-frequency offset phase noise source is the PN from DLL and PI. Based on the phase domain model, the PN from the PD and LPF in the filter is another source.

To reduce these low-frequency offset phase noise, the most feasible approach is to increase their power consumption, since the power consumption directly trades with the phase noise. From the simulation results, the DLL and PI contribute 315.9 fs_{rms} to the final jitter of 368 fs_{rms}. The high-frequency-offset noise components from the DLL and PI are low-pass filtered while the low-frequency-offset noise components are left unfiltered. The increase in the DLL and PI's power consumption is unavoidable if a lower jitter performance is required. This method is illustrated in fig.5.4.

From the phase domain model, the low-frequency-offset phase noise can be filtered with a low-frequency-offset noise suppression pole frequency. However, it requires a large capacitor to put the noise suppression pole at a low frequency, which takes much area. The loop bandwidth would drop corresponding to the pole frequency as well, which will limit the application scenario of the phase noise filter. Thus, this method is not considered.

6.2.2. Reduce High-frequency-offset phase noise

The high-frequency-offset phase noise sources are the VCDL and LDO in the filter, which can't be low-pass filtered by the phase noise filter. The most feasible approach is to increase their power consumption(e.g. increase the sizing of the transistors)

The second method is to decrease the bandwidth of the LDO. The modification of the LDO bandwidth is not helpful in improving the noise transfer function from the LDO to the output, but is helpful in reducing the noise bandwidth of the LDO itself. Note that there's a lower bound for the LDO bandwidth to ensure the loop stability. The LDO bandwidth can be realized by increasing the two capacitors in the LDO. The increased value of the load capacitor decreases the second pole frequency. The other capacitor needs to be increased as well to ensure enough phase margin.

Although the high-frequency-offset phase noise components from other sources will be low-pass filtered, their high-frequency-offset phase noise may still contribute much jitter if the noise suppression pole frequency is relatively high. In this case, the pole frequency can be decreased by increasing the capacitance of the loop filter. This method is illustrated in fig.5.5.

6.3. State of the art comparison

The results are compared with a similar structure with a 10-bit DTC with a charge-injection-based phase noise filter [12]. This work shows a larger jitter and higher power consumption, along with a larger area. However, the performance of this design is limited by the 180nm process. For example, with a higher reference clock frequency under more advanced technology, the number of VCDL delay stages in the DLL can be reduced, which saves area and power. The jitter integration range can also increase from 1kHz-100MHz to 10kHz-1GHz. Thus, the noise suppression pole frequency can be increased, which leads to a higher loop bandwidth and a smaller loop filter capacitor.

Table 6.1: PERFORMANCE SUMMARY

Metrics	This work	[12] JSSC'22
Architecture	QDLL+PI+DLL	QDLL+MPILOSC+PI
Resolution(bits)	10	10
DLL Resolution	4	3
PI Resolution	6	7
Technology	180nm	65nm
Reference Freq.(GHz)	1	7
Power(mW)	42.3	22.71
Supply(V)	1.8	1.2
Integrated Jitter(fs_{rms})	368(1kHz-100MHz)	106 (10kHz-1GHz)
INL_{max}	1.73LSB	$<$2.84LSB in DLL($<1°$) $<$1.12LSB in PI
Area(mm^2)	0.482	0.0432
Bandwidth(Hz)	160k	20M
Filter DLL noise	✓	✓
Filter PI noise	✓	✗

The reasons for the disparity and the theoretical advantages of the design are given in the following subsections.

6.3.1. Advantages of the design

The most outstanding advantage of the proposed phase noise filter is that it can filter the phase noise from both the DLL and the PI, while the other phase noise filter can only filter the DLL's phase noise. Last but not least, the high-frequency offset phase noise is dominant by the phase noise from a ring OSC in [12] while the dominant high-frequency-offset phase noise source is a VCDL in this design. The jitter generated by the VCDL is less than that generated by the ring OSC under the same power consumption condition[18]. This is because the phase noise will circulate in the ring OSC, which means the phase noise will be integrated, while the VCDL doesn't have this problem.

6.3.2. Limitations of the design

The first stage of the two designs applies a similar structure, DLL with quadrature error correction. The asymmetry in the layout design becomes the main limitation of the total design's linearity performance.

The second stage of the proposed design applies the PI with pipelined structure while the other design applies the traditional PI structure. The pipelined PI shows less power, area, and a better linearity performance but requires two extra analog control voltages.

To adjust the noise suppression pole frequency, a large capacitor may be required for the loop filter, which takes much area. However, the noise suppression pole frequency is adjusted by charge injection strength in [12].

7 Conclusion

In this chapter, the main contributions of the proposed design are summarized. It also provides some recommendations for future research.

7.1. Summary of Main Contributions

This thesis proposes the design of a 10-bit DTC with a new architecture of phase noise filter. The DTC is realized by a 4-bit DLL with QEC and a 6-bit pipelined PI. The type-2 DLL's noise transfer function from the PFD to the output is utilized to form the phase noise filter, which shows first-order low-pass characteristics. The loop bandwidth of the LDO can also be utilized to provide a second pole and forms a second-order low-pass characteristic. The design is conducted in a 180-nm BCD process, where the schematic and layout design is finished. The post-layout simulation results show that the final output jitter of the filter is 368 fs_{rms} while the total power consumption is 42.3mW, with the noise suppression pole frequency at 160kHz. The jitter from the DLL and PI is reduced from 1.099ps_{rms} to 315.9 fs_{rms}. The area is 695 µm $\times$ 693.5 µm. The design consumes 42.3 mW with 1.8V supply in 180nm BCD technology.

Compared with the charge-injection-based phase noise filter, the proposed design shows a theoretical advantage in that the PI's phase noise can be filtered. The phase domain model and the simulation results also reveal a design framework for further jitter improvement. The low-frequency-offset phase noise sources are the DLL, PI, and the PD and LPF in the phase noise filter, whose phase noise can be reduced by increasing power consumption. The high-frequency-offset phase noise sources are the VCDL and LDO in the phase noise filter, which can be reduced by increasing power consumption and reducing noise suppression pole frequency.

7.2. Recommendation for future work

7.2.1. More advanced technology

The design applies the 180nm technology, which shows a larger parasitic capacitance and higher supply voltage compared with the more advanced technology. These properties all lead to higher power consumption.

The large parasitic capacitance also determines the upper limit of the clock frequency. The highest applicable clock frequency in 180nm is about $2 \sim 3$ GHz. The applied clock frequency in the proposed design is 1GHz, considering the XOR gate would work at twice the reference clock frequency. The number of the VCDL stages in the DLL is determined by the period of 1GHz. 16 VCDL stages are applied in the proposed design to cover the 1ns delay range. However, with a higher reference clock frequency, less number of VCDL stages can be utilized. For example, the VCDL stage number is 8 with a 7GHz reference frequency in [12]. The reduction of the VCDL stages is desired in the DTC design, since the phase resolution expansion by PI is more power and area efficient than that by the DLL. Last but not least, the jitter is accumulated throughout the VCDL. With a smaller number of VCDL stages, less jitter would be accumulated at the end stage.

One problem would occur with a more advanced technology, which is the reduced output impedance

of the gate-switch cascode CP in the phase noise filter. The reduced impedance would affect the charging and discharging current offset when the control voltage varies. This would influence the phase information copy from the DLL and PI to the phase noise filter. However, the control voltage range to cover a period of the reference clock would also reduce, which loosens the output impedance requirements.

7.2.2. Layout design symmetry

From the INL and DNL simulation results, the maximum INL error resulting from the DLL is larger than 1LSB while the maximum INL error resulting from the PI is less than 1LSB. The major linearity errors result from the DLL's static phase error problem. The pre-simulation of the DLL shows almost zero static phase error while the post-layout simulation shows a large INL error due to the asymmetry in layout. During the transformation from the schematic to the layout, the simulation shows the following factors lead to the degradation of the static phase error problem:

- The unequal resistance and unsymmetrical parasitic capacitance conditions of the four input paths of the XOR gate. The equal resistance can be realized by ensuring the same length of each wiring path. However, the symmetrical parasitic capacitance is difficult to realize. The feasible solution is to reduce the parasitic capacitance as small as possible between the signal paths.
- The unequal resistance and unsymmetrical parasitic capacitance conditions of the sixteen signal paths from the sixteen output buffers of the VCDL to the four inputs of the XOR gate.
- Unequal loading of the VCDL due to the wiring of the independent MUX (the Mux which is controlled by an independent selecting signal) at the last stage of the VCDL.
- The resistance from the power supply (VDD and VSS) to the transistors of the VCDL and the MUXs. The path length from VDD to various stages of the VCDL and MUX is different, leading to unequal path resistance. The actual supply voltage on each VCDL stage and MUX will then deviate from the ideal 1.8V.

7.2.3. MUX stage of the pipelined PI

The MUX stage of the pipelined PI in this design is realized by inserting two inverters between the two transmission gates. The two inverters reverse the falling/rising edge transition direction and require two types of PI units to ensure the phase can be conveyed from rising to falling edge or falling to rising edge.

This MUX structure can be improved by inserting only one inverter between the two transmission gates. Although the MUX will then reverse the phase direction, the PI unit will reverse the direction as well. Consequently, one type of PI unit is required and they can share the same analog control voltage, which can reduce one I/O pin.

Bibliography

[1] J. Liang, A. Sheikholeslami, H. Tamura, Y. Ogata, and H. Yamaguchi, "6.7 a 28gb/s digital cdr with adaptive loop gain for optimum jitter tolerance," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 122–123. DOI: 10.1109/ISSCC.2017.7870291.

[2] M.-J. Lee, W. Dally, T. Greer, *et al.*, "Jitter transfer characteristics of delay-locked loops - theories and design techniques," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 4, pp. 614–621, 2003. DOI: 10.1109/JSSC.2003.809519.

[3] G. Balamurugan and N. Shanbhag, "Modeling and mitigation of jitter in multigbps source-synchronous i/o links," in *Proceedings 21st International Conference on Computer Design*, 2003, pp. 254–260. DOI: 10.1109/ICCD.2003.1240903.

[4] B. Razavi, *Design of CMOS Phase-Locked Loops: From Circuit Level to Architecture Level*. Cambridge University Press, 2020, ISBN: 9781108494540. [Online]. Available: https://books.google.nl/books?id=Sm%5C_CDwAAQBAJ.

[5] Y. Yoon, H. Park, and C. Kim, "A dll-based quadrature clock generator with a 3-stage quad delay unit using the sub-range phase interpolator for low-jitter and high-phase accuracy dram applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2342–2346, 2020. DOI: 10.1109/TCSII.2020.2976983.

[6] M. Mansuri, "Clocking, clock distribution, and clock management in wireline and wireless subsystems," *International Solid-State Circuits Conference*, 2021.

[7] W. Kester. "Converting oscillator phase noise to time jitter." (2008), [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf.

[8] C.-Y. Yang, C.-H. Chang, and W.-G. Wong, "A $\Delta-\Sigma$ pll-based spread-spectrum clock generator with a ditherless fractional topology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 1, pp. 51–59, 2009. DOI: 10.1109/TCSI.2008.926975.

[9] R. J. Ferreira, R. E. Araújo, and J. Peças Lopes, "A comparative analysis and implementation of various pll techniques applied to single-phase grids," in *Proceedings of the 2011 3rd International Youth Conference on Energetics (IYCE)*, 2011, pp. 1–8.

[10] P. Kinget, R. Melville, D. Long, and V. Gopinathan, "An injection-locking scheme for precision quadrature generation," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 7, pp. 845–851, 2002. DOI: 10.1109/JSSC.2002.1015681.

[11] M. Raj, S. Saeedi, and A. Emami, "22.3 a 4-to-11ghz injection-locked quarter-rate clocking for an adaptive 153fj/b optical receiver in 28nm fdsoi cmos," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3. DOI: 10.1109/ISSCC.2015.7063097.

[12] Z. Wang, Y. Zhang, Y. Onizuka, and P. R. Kinget, "Multi-phase clock generation for phase interpolation with a multi-phase, injection-locked ring oscillator and a quadrature dll," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 6, pp. 1776–1787, 2022. DOI: 10.1109/JSSC.2021.3124486.

[13] Y. Li and G. W. Roberts, "Design of high-order type-ii delay-locked loops with a fast-settling-zero-overshoot step response and large jitter-rejection capabilities," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1805–1818, 2018. DOI: 10.1109/TCSI.2017.2771141.

[14] B. D. Kale and U. Ghodeswar, "Design of quadrature error correction loop for communication," in *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 626–632. DOI: 10.1109/ICCES45898.2019.9002200.

[15] A. AbdelHadi, M. Allam, and S. Ibrahim, "A high-linearity 6-ghz phase interpolator in 28-nm cmos technology," in *2017 Japan-Africa Conference on Electronics, Communications and Computers (JAC-ECC)*, 2017, pp. 41–44. DOI: 10.1109/JEC-ECC.2017.8305774.

[16] A. T. Narayanan, M. Katsuragi, K. Kimura, *et al.*, “A fractional-n sub-sampling pll using a pipelined phase-interpolator with a fom of −246db,” in *ESSCIRC Conference 2015 - 41st European Solid-State Circuits Conference (ESSCIRC)*, 2015, pp. 380–383. DOI: 10.1109/ESSCIRC.2015.7313907.

[17] Y.-T. Chen, P.-J. Peng, and H.-W. Lin, “A 12–14.5-ghz 10.2-mw −249-db fom fractional-n sub-sampling pll with a high-linearity phase interpolator in 40-nm cmos,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 5, pp. 634–643, 2022. DOI: 10.1109/TVLSI.2022.3160327.

[18] A. Abidi, “Phase noise and jitter in cmos ring oscillators,” *IEEE Journal of Solid-State Circuits*, vol. 41, no. 8, pp. 1803–1816, 2006. DOI: 10.1109/JSSC.2006.876206.