FRACTIONAL-N DLL FOR CLOCK SYNCHRONIZATION

QIU LIN

NATIONAL UNIVERSITY OF SINGAPORE

2007

FRACTIONAL-N DLL FOR CLOCK SYNCHRONIZATION

QIU LIN

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

ACKNOWLEDGEMENTS

I would like to thank the National University of Singapore for the support they had given, which led to the success of this project.

I would also like to thank my project supervisor, Assistant Professor Heng Chun Huat, for his guidance on research and this thesis.

Besides, many thanks should be given to my colleagues in the Signal Processing and VLSI Design Laboratory, for their kind advice during these two years.

Last but not least, I would like to thank everyone who had helped, in one way or another, towards the completion of this project.

TABLE OF CONTENTS

LIST OF FIGUERS

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

CP	Charge pump
DLL	Delay-Locked Loop
FSM	Finite State Machine
LP	Low-pass Filter
PD	Phase Detector
PLL	Phase-Locked Loop
VCDL	Voltage Control Delay Line
MUX	Multiplexer
RTL	Register Transfer Level
f_{REF}	Reference frequency
$\Delta\Sigma$	Delta sigma
K_{PD}	The phase detector gain
I_{CP}	The charge pump current
K_{DL}	Gain of the delay line
ω_N	The loop bandwidth
V_C	The control voltage
$\hat{X}$	Fractional division ratio
g_m	Transconductance
r_{ds}	Output resistance

ABSTRACT

This thesis presents on the design of a fractional-N delay locked loop (DLL) circuit for clock synchronization in the transceiver system. A delta-sigma modulator is integrated into the DLL design to achieve low noise and low jitter performance. It is verified through the behavior model simulation that this DLL can provide fine phase resolution, wide operation range and low jitter performance. The whole architecture is implemented in 0.35 μm CMOS and is able to handle input clock frequency range from 10MHz~200MHz. This novel technique can also be used for a variety of applications which requires accurate timing delay or fine tuning resolution.

CHAPTER 1 INTRDUCTION

As the speed performance of the VLSI systems increases rapidly, more emphasis is placed on suppressing the clock skew and jitter. Phase-locked loop (PLL) and delay-locked loop (DLL) have been typically applied to microprocessors, memory interface or communication IC, where the external signal is required to be synchronized to the local reference clock. However, the clock skew and jitter reduces the resolution of clock synchronization, especially in high speed system. In this chapter, a novel DLL architecture is introduced for low noise and low jitter purpose.

1.1 Motivation of Fractional-N DLL

1.1.1 PLL and DLL Comparison

Many applications requires accurate phase placement of clock or data signal. Although phase shift can be done by simply delaying the signal, it is not robust to variations of processing, voltage or temperature. For more precise control, a feedback loop is used to lock the output phase with respect to an input reference signal, which is the essence of PLL. Recently, DLL has emerged as an alternative to the traditional oscillator-based PLL.

Compared with PLL, phase is the only state variable for DLL. With its first order loop characteristic, DLL has better stability and no cycle-to-cycle jitter accumulation.[1] The basic building blocks for both PLL and DLL are similar. This includes phase detector, charge pump and loop filter. The oscillator and divider in PLL are replaced with the variable-delay line, which are shown in Figure 1-1 and Figure 1-2, respectively. Obviously, DLL has simpler structure than PLL

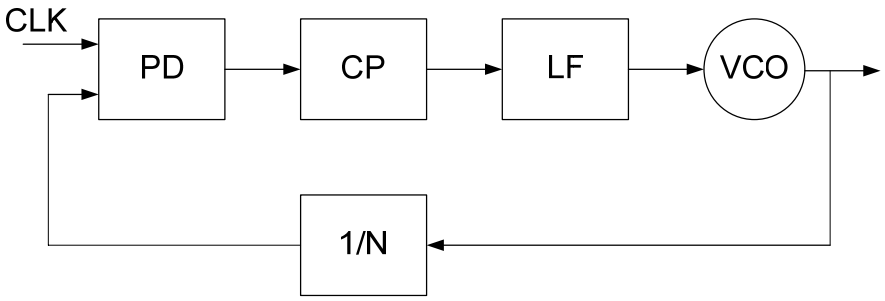

Figure 1-1: Block diagram of PLL

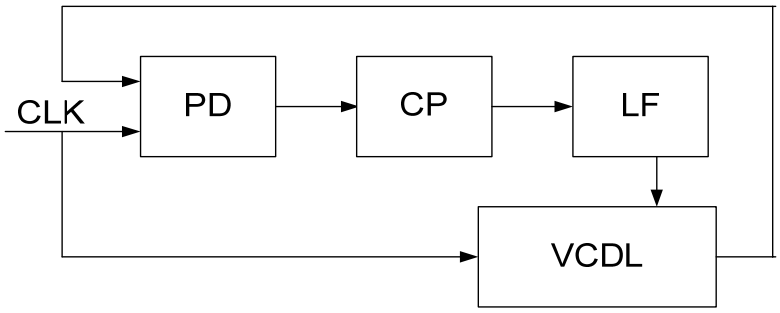

Figure 1-2: Block diagram of DLL

DLL is preferred for clock synchronization due to fine phase resolution and low jitter performance. Recently, it has been commonly employed in the ultra-wideband (UWB) system, because the signals consist of short pulses with typical pulse width around nanoseconds and the phase resolution is very critical.

1.1.2 Fractional-N PLL Technology

The development of fractional-N PLL becomes an important area of PLL study. As shown in Figure 1-1, the basic integer-N PLL has a limitation that the output frequency is fixed to integer multiples of the reference frequency, f_{ref} . In other word, finer tuning resolution can only be achieved with lower f_{ref} .[2] Moreover, the reference frequency also determines the loop bandwidth of PLL, which is approximately f_{ref} /10. Thus, reducing f_{ref} tends to decrease the maximum loop bandwidth, thereby increasing the settling time of PLL, the noise contributed by VCO, and so on.

To solve the problem caused by this fundamental tradeoff between bandwidth and tuning resolution in integer-N PLL, fractional-N PLL is created by incorporating additional digital circuitry, which allows accurate interpolation between integer multiples of reference frequency. In this way, tuning resolution depends on the design and algorithm of the digital circuitry, so considerable flexibility and programmability can be achieved.[3] The ΔΣ fractional-N PLL, illustrated in Figure 1-3, takes the idea one step further by randomizing the division ratio to eliminate the spurious tones. [4]

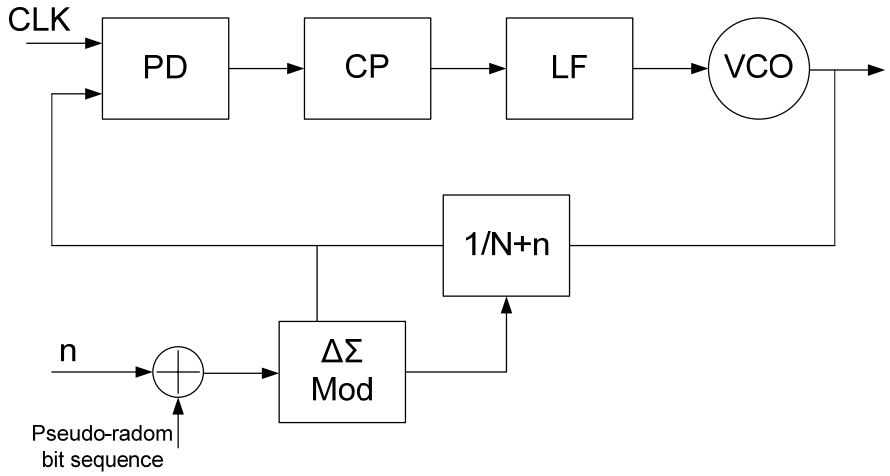

Figure 1-3: ΔΣ fractional-N PLL

The success of ΔΣ PLL has spurred our interest in incorporating similar idea to the DLL architecture to achieve the objective of fine tuning resolution and better jitter performance.

1.1.3 Motivation of Fractional-N DLL

The previous discussion indicates that the high speed transceiver design triggers study and research on precise clock synchronization, which targets low jitter performance. DLL is considered as a proper structure because of its inherited characteristic with no cycle-to-cycle jitter accumulation. In addition, to further reduce

phase noise while improve tuning resolution, ΔΣ fractional-N technology is applied to the DLL design in the form of using a delta-sigma modulator.

The proposed DLL block diagram is shown in Figure 1-4. A delta-sigma modulator and some digital control circuit (finite state machine) are added to realize the concept of fractional-N DLL.

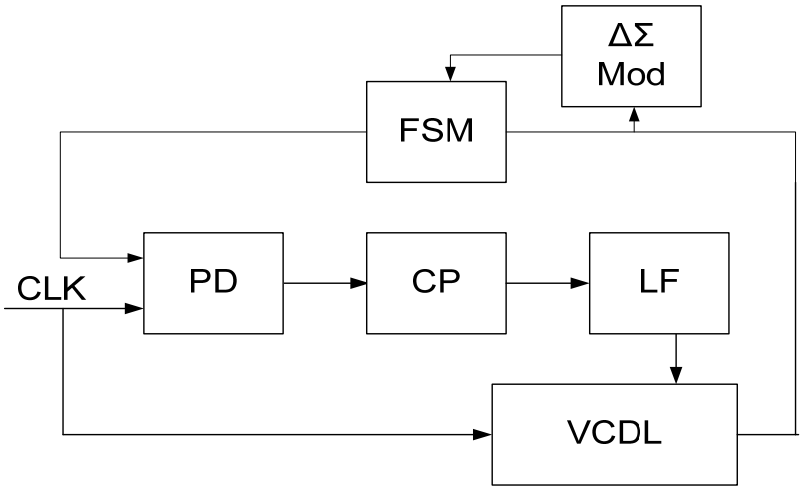

Figure 1-4: Block diagram of fractional-N DLL

The digital control circuit is implemented as finite state machine (FSM), which could be described in Verilog language and synthesized into digital circuits. The function of FSM and how it achieves programmable tuning resolution will be discussed in Chapter 3.

1.2 Thesis Contributions

The major contributions of this thesis include:

1. A novel approach to design DLL by incorporating a delta-sigma modulator to achieve low jitter performance.

2. A novel FSM algorithm for controlling the coarse tuning and the fine tuning. This simplifies the overall system architecture by sharing one delay line, and also helps achieving programmable tuning resolution.

3. A novel circuit design of adaptive loop filter with programmable bandwidth and phase margin.

1.3 Thesis Organization

The rest of this thesis is organized in the following manner.

Conventional DLL architecture and its development are reviewed and studied in Chapter 2. This is followed by detailed discussion on the proposed fractional-N DLL architecture in Chapter 3. Chapter 4 presents the CMOS implementation on both the analog and digital circuitry. The final conclusion is summarized in Chapter 5.

CHAPTER 2 EXSITING DLL ARCHITECTURES

For clock synchronization, DLL performs better stability and lower jitter performance than PLL in the applications with no frequency variation. However, conventional DLL suffers from the problem of limited operating frequency range. A few works have been proposed to solve this problem. This chapter begins with the basic analysis of conventional DLL, such as the operation principle, design parameters and the origin of its limitation. Two most common architectures, phase interpolation DLL and dual loop DLL, targeting wide frequency range, precise delay control and low jitter performance would then be discussed.

2.1 DLL Fundamental

To demonstrate the operation principle of DLL, the linearized model is shown in Figure 2-1. It is characterized by the phase detector gain K_{PD} (rad), charge pump current I_{CP} (A), loop filter transfer function $F(s)$, and the gain of delay line K_{DL} (rad/V). The input and output delay time are denoted as D_I and D_O, respectively. The input clock period is T_{REF}. [5]

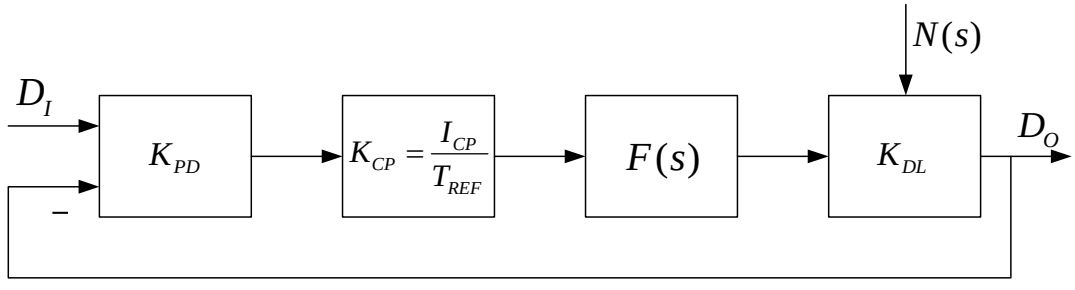

Figure 2-1: Linearized model of conventional DLL

Conventional DLL employs a single capacitor C_{LF} as the loop filter, which has the first-order transfer function

$$F(s) = \frac{1}{sC_{LF}}. \tag{2.1}$$

The close-loop behavior can be expressed by the following first-order transfer function, with low-pass characteristic.

$$H(s) = \frac{G(s)}{1+1/G(s)} = \frac{1}{1+s/\omega_N} \tag{2.2}$$

In this equation, $G(s) = K_{PD}K_{CP}F(s)K_{DL}$ is the open-loop transfer function, and ω_N is the bandwidth of close-loop response. The loop is unconditionally stable since there is only one pole which contributes to 90 degree of phase lag.

Assume a noise source introduced at the delay line output, so the noise transfer function is a high-pass response.

$$N(s) = \frac{1}{1+1/G(s)} = \frac{s/\omega_N}{1+s/\omega_N} \tag{2.3}$$

Compared with the signal transfer function which is low-pass response, the noise due to delay line is high-pass filtered.

It should be pointed out that the loop bandwidth ω_N tracks the operation frequency ω_{REF}, which is derived below.

$$\omega_N = \frac{K_{PD}K_{DL}I_{CP}}{2\pi C_{LF}} \cdot \omega_{REF} \tag{2.4}$$

The loop bandwidth is a critical design parameter. To achieve fast loop locking speed, ω_N should be set as high as possible. On the other hand, high frequency noise cannot

be suppressed if ω_N increases. The design guideline suggests a typical ratio of 1/10 for ω_N / ω_{REF}. [6]

Although conventional DLL is preferred for its unconditional stability, low phase error accumulation and fast locking time, it encounters operation range problem. [7] The architecture of conventional DLL is shown in Figure 2-2. Assume the reference clock (REF_CLK) origins from the first rising edge, and the output of delay line (DLL_CLK) is the output clock through delay line.

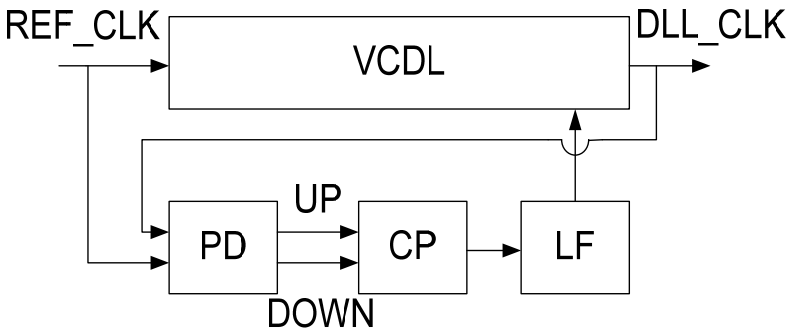

Figure 2-2: Conventional DLL architecture

The normal operation of conventional DLL is illustrated in Figure 2-3. The 1st rising edge of the reference clock (REF_CLK) and its delay version (DLL_CLK) are indicated by the circle. The rising edge of DLL_CLK is then compared with the 2nd rising edge of the REF_CLK, which is indicated by the dark upward arrow. Depending on whether there is a phase lag or phase lead, an UP or DOWN control signal is generated correspondingly. Once DLL is locked, the DLL_CLK is delayed by exact one clock period from the REF_CLK.

However, this normal operation is only guaranteed when the initial DLL_CLK appears within the dotted line regions. When the initial DLL_CLK is outside this

region, it is no longer locked by one clock period from the REF_CLK. The false locking problem is shown in Figure 2-4.

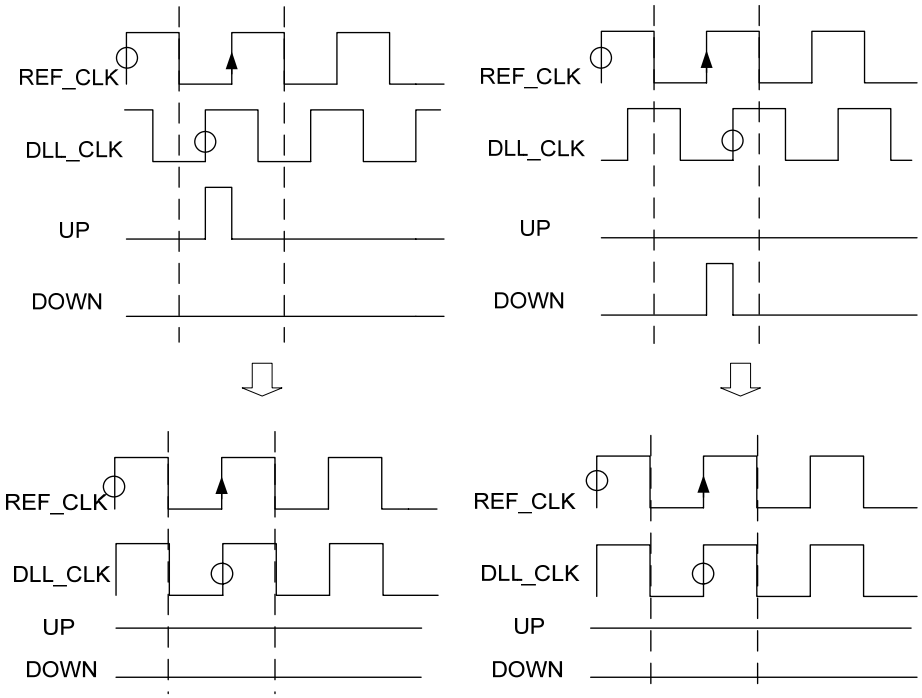

Figure 2-3: DLL locking range

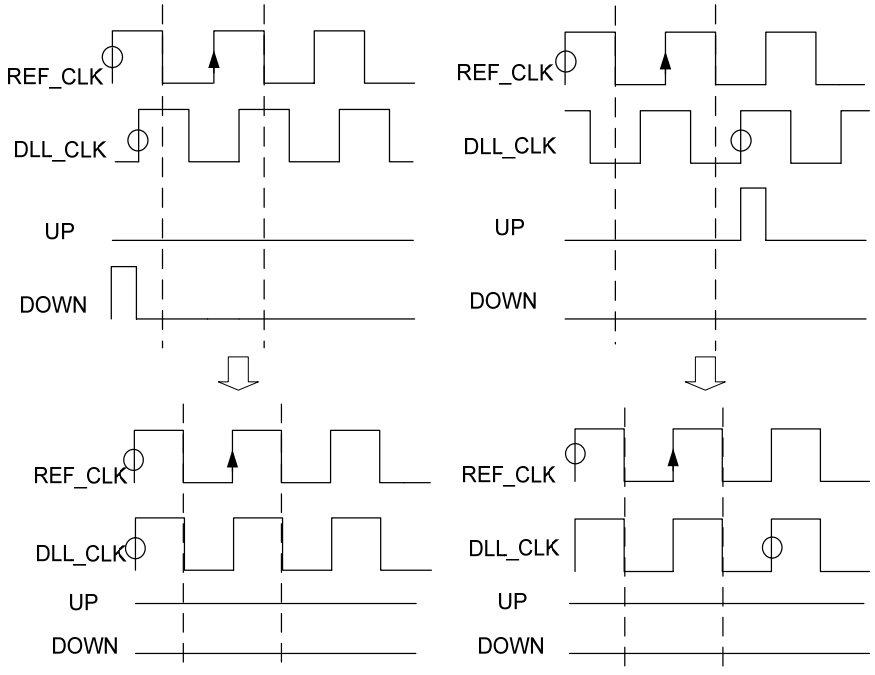

Figure 2-4: False locking problem

To avoid false locking problem, the delay range is found as follows.

$$\frac{1}{2}T_{CLK} < T_{DL,\min} < T_{CLK}$$

$$T_{CLK} < T_{DL,\max} < \frac{3}{2}T_{CLK} \tag{2.5}$$

Or, it can be expressed in terms of T_{CLK} ,

$$Max(T_{DL,\min}, \frac{2}{3}T_{DL,\max}) < T_{CLK} < Min(2T_{DL,\min}, T_{DL,\max}) \tag{2.6}$$

The above inequality indicates that the operating frequency can only be satisfied over a very limited range.

The application of DLL always depends on its basic structure. In the first type, the input clock REF_CLK is compared with the delayed version of itself, as shown in Figure 2-5. Usually, the output DLL_CLK is used for clock generation and frequency synthesis. [8]

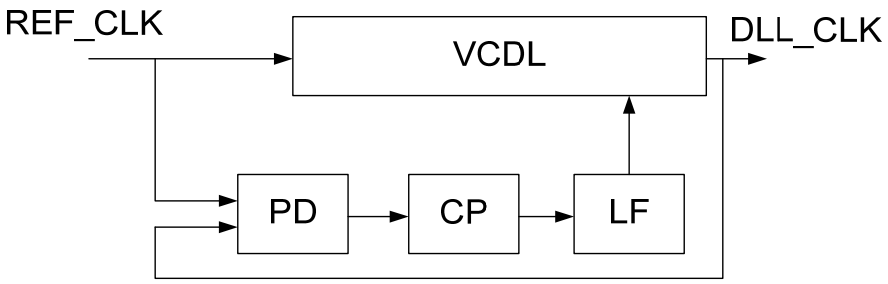

Figure 2-5: The first type of DLL

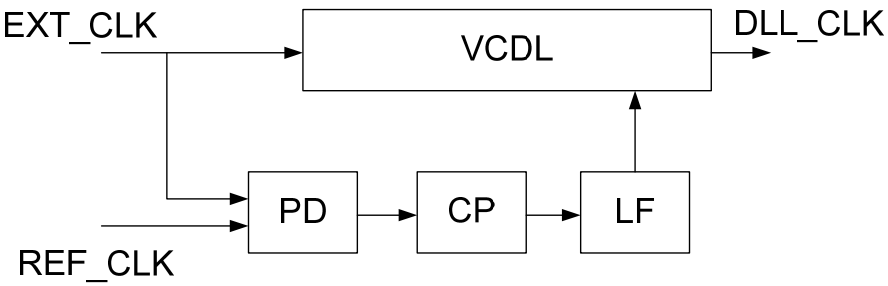

Figure 2-6: The second type of DLL

The second type of structure is shown in Figure 2-6, where the reference clock REF_CLK is compared with the delayed version of another external clock, EXT_CLK. There are often two input clocks, and the output clock, DLL_CLK, is used for clock synchronization or signal recovery. [9]

2.2 Phase Interpolation DLL

Phase interpolation DLL architecture, shown in Figure 2-7, is proposed to overcome the false locking problem and attain wide operation range. Instead of only using the clock phase at the end of the delay line, multiple coarse clock phases are tapped from the middle of the delay line. The phase interpolator then produces a clock phase which is interpolated from the tapped coarse clock phases.

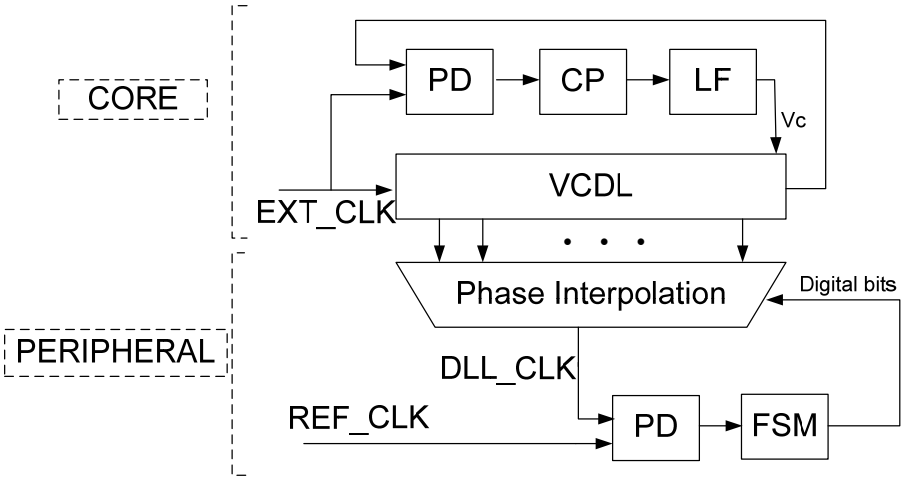

Figure 2-7: Phase interpolation DLL architecture

There are two loops commonly known as the core loop and the peripheral loop, The core loop is a conventional DLL which locks the delay line with one cycle period to generate accurate multiple phases. The peripheral loop generates the phase interpolated DLL_CLK to align with REF_CLK. The phase selection and interpolation are controlled by the finite state machine (FSM).

The linearized model is shown in Figure 2-8. [10] Each of the two loops is modeled as a single pole system, in which the input, output and error variables are represented in delay (in seconds), similar to the analysis in chapter 2.1. D_{EXT} and D_{REF} are the input clocks, while D_{OC} and D_O are the output clocks for the core loop and the peripheral loop, respectively. So D_N represents delay errors related to the supply and substrate noise. D_{EC} and D_E are delay errors seen by the phase detector in each loop.

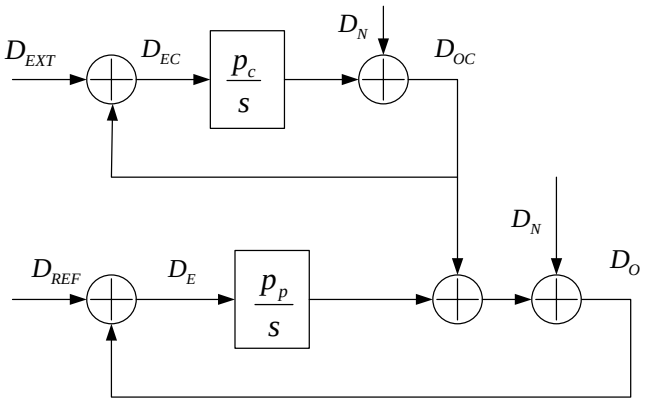

Figure 2-8: Linearized model of dual loop DLL

For the core loop, the input-to-output transfer function can be derived as

$$\frac{D_{OC}}{D_{EXT}} = \frac{1}{1 + s/p_c} \tag{2.7}$$

where p_c (in rad/s) is the pole of the core loop determined by the charge pump current, the phase detector and the delay line gain, as well as the loop filter capacitor. The noise-to-error transfer function is given as

$$\frac{D_{EC}}{D_N} = \frac{s/p_c}{1 + s/p_c} \tag{2.8}$$

This indicates that the delay errors (D_{EC}) related to supply and substrate noise (D_N) can be tracked up to the loop bandwidth.

For the peripheral loop, the input-to-output transfer function is similarly derived as

$$\frac{D_O}{D_{REF}} = \frac{1}{1 + s/p_p}, \tag{2.9}$$

To understand the effect of external clock disturbance (D_{EXT}) on the delay error of the peripheral loop (D_E), the following expression can be attained.

$$\frac{D_E}{D_{EXT}} = \frac{s/p_p}{(1 + s/p_c) \cdot (1 + s/p_p)} \tag{2.10}$$

The delay error D_E related to supply and substrate noise D_N is given as follows.

$$\frac{D_E}{D_N} = \frac{(1 + 2s/p_c) \cdot s/p_p}{(1 + s/p_c) \cdot (1 + s/p_p)} \tag{2.11}$$

This equation establishes that there is no overshoot in the dual loop step response when the peripheral loop bandwidth is less than half of the core loop bandwidth. [10]

The design of phase interpolator in [8] performs a weighted average of the input phases. As shown in Figure 2-9, ideally, the FSM controls multiplexers to select two input phases, both of which drive an integrator and their output currents join in to charge a capacitor. For example, when α = 1, the integrator output depends only on CLK0; when α = 0.5, the current is split equally between two integrators and the output phase is at the middle of CLK0 and CLK1. By varying α , the phase interpolator produces a programmable DLL_CLK.

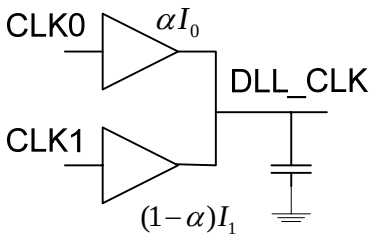

Figure 2-9: Design of phase interpolator

This phase interpolation architecture effectively solves the false locking problem of conventional DLL. Since the DLL_CLK is able to switch smoothly between clock phases tapped from the delay line, it can increase or decrease a wide range of delay to gradually track the reference clock. In other word, DLL covers a larger operation frequency range. Besides, programmable tuning resolution can be obtained by proper design of the FSM algorithm and the weighted factor and so on.

However, the digital phase interpolation limits the tuning resolution, because α could only be a finite number of fractions between 0 and 1. In addition, more delay cells are needed to avoid slew rate problem in the phase interpolation. Both issues impact the jitter performance, the area and the power consumption of phase interpolation DLL.

2.3 Dual Loop DLL

Dual loop DLL is developed to improve jitter performance. In dual loop DLL, the digital controlled phase interpolation is replaced with another analog mode DLL to achieve continuous phase adjustment.

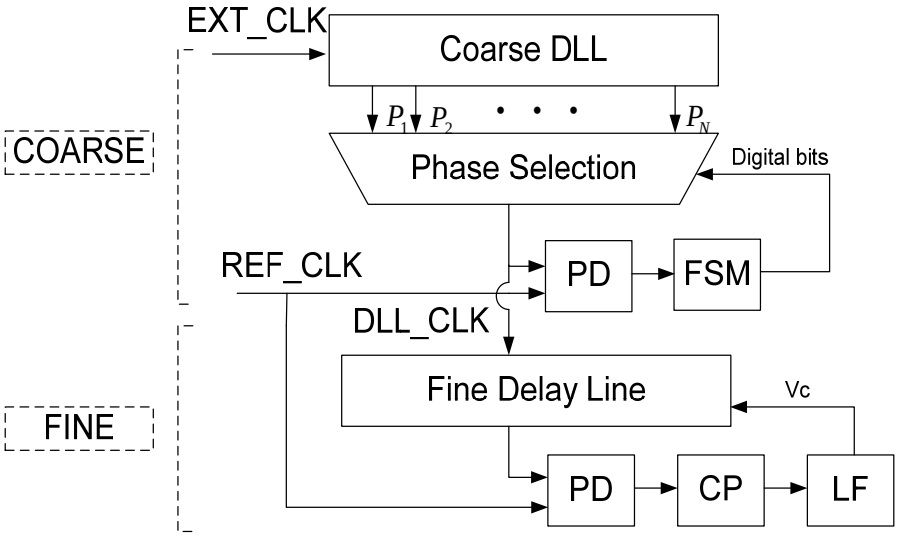

Figure 2-10: Dual loop DLL architecture

The architecture of dual loop DLL is displayed in Figure 2-10. [11] It involves two loops, the coarse loop and the fine loop, to synchronize the external clock (EXT_CLK) with the reference clock (REF_CLK). The coarse loop provides the closest clock phase (DLL_CLK) to the REF_CLK. The fine loop is only activated once the coarse loop has finished the phase selection. Then, the selected clock is aligned with the reference clock through the fine loop, which consists of anther analog DLL.

The phase selection is shown in Figure 2-11. For example, if the size of the coarse DLL is 10, the rising edges of multiple clocks (dashed lines) are equally spaced in one cycle of EXT_CLK. Thus, for the displayed phase error between EXT_CLK and REF_CLK, P_4 is selected as the DLL_CLK, which is the closest clock to REF_CLK.

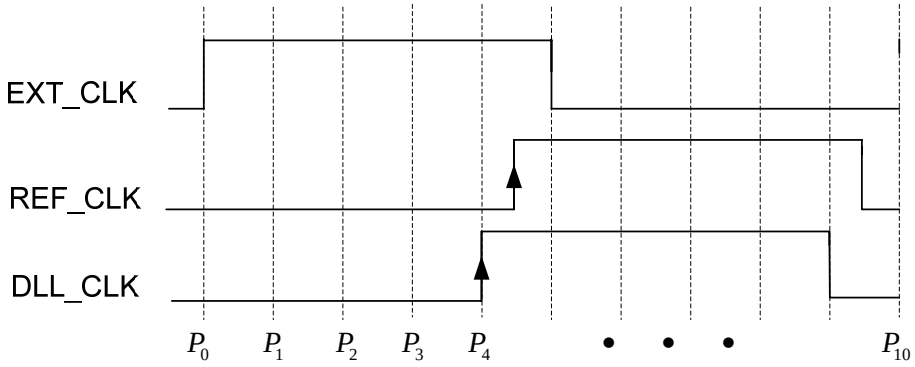

Figure 2-11: Phase selection of dual loop DLL

The step response of the coarse tuning and the fine tuning are shown in Figure 2-12. The left figure illustrates that the coarse tuning is digital mode, while the right figure illustrates that the fine tuning is analog mode.

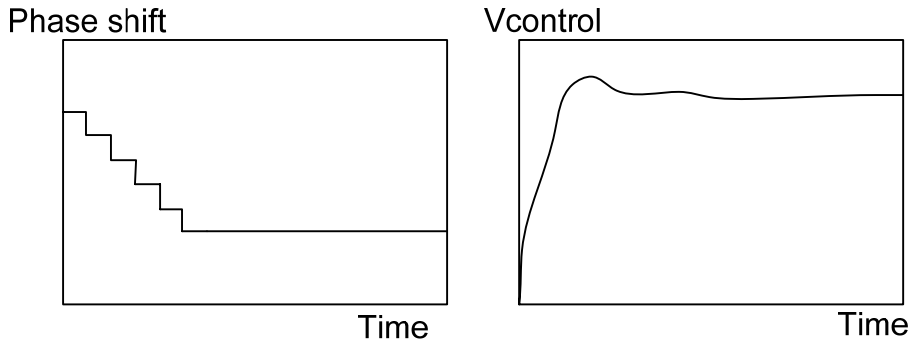

Figure 2-12: Coarse tuning and fine tuning

The dual loop DLL overcomes the finite phase problem of the phase interpolation DLL. It also significantly reduces the settling time. However, the additional delay line in the fine loop deteriorates the phase jitter, and it costs area and power penalty too.

In next chapter, the proposed architecture will be introduced to reduce the above limitation.

CHAPTER 3 FRACTIONAL-N DLL ARCHITECTURE

This chapter presents the proposed fractional-N DLL architecture. The system is designed and proven through behavioral simulation using Matlab/Simulink.

3.1 System Architecture

The fractional-N DLL employs a delta-sigma modulator to improve the jitter performance. Compared with the exiting DLL architectures, it doesn't introduce the phase interpolator or the additional delay line. The structure is very similar to the dual loop DLL, as illustrated in Figure 3-1.

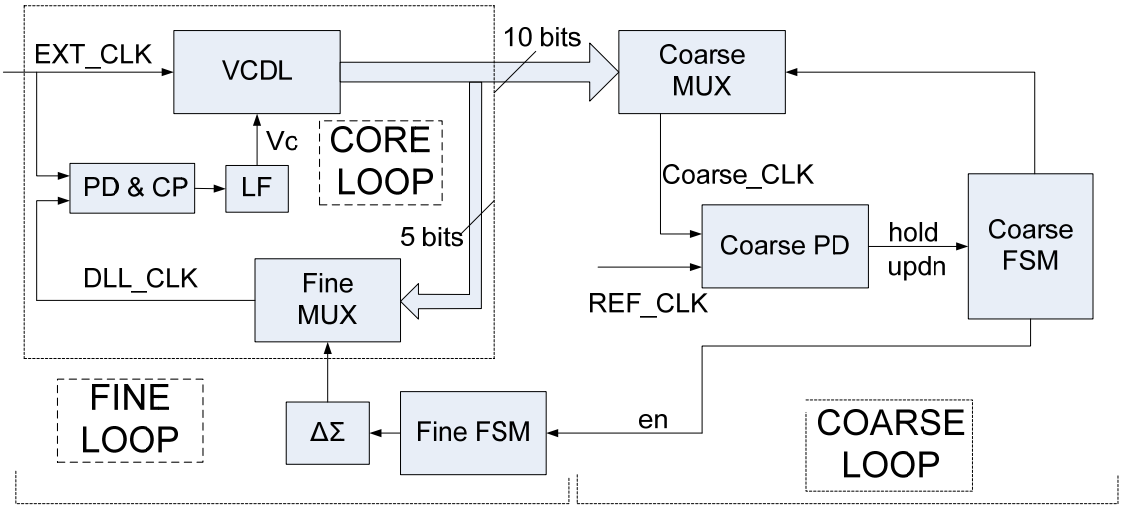

Figure 3-1: System architecture of fractional-N DLL

Initially, the core DLL is running at some pre-defined delay interval and generating multiple clock phases for the coarse loop. The coarse loop will select the clock phase that is closest to the reference clock (REF_CLK). During this period, the finite state machine and the $\Delta\sum$ modulator in the fine loop are disabled. Once the coarse loop acquires the closest clock phase (Coarse_CLK), it will enable the fine loop. This is realized by the digital block, Coarse FSM, to hold the state of Coarse_CLK selection, and send out logic control signal. The fine loop will then start adjusting the

initial pre-defined delay interval through the delta-sigma modulator. This is based on the principle that the $\Delta\sum$ modulator controls the feedback signal, DLL_CLK, from the VCDL, thus the effective number of delay cells. Another digital block, Fine FSM is designed to update the input of the $\Delta\sum$ modulator, then the DLL_CLK selection as well. It will stop when the Coarse_CLK, aligns with the REF_CLK, being locked with new delay interval. So the Coarse_CLK is the final output clock.

The main advantage of this proposed DLL is that the selected clock phase is derived directly from the core delay line without additional phase interpolator or second delay line. This saves extra power consumption and chip area. Besides, the jitter accumulation mostly comes from the delay line, which can be reduced by sharing the delay line for both the coarse loop and the fine loop. In the next few sections, each block will be discussed in detail.

3.2 Sub-block Models

The main sub-blocks could be divided into the following categories: voltage control delay line (VCDL) and other coarse loop components, delta-sigma ($\Delta\sum$) modulator and other fine loop components. Design specifications are described for each sub-block, with behavioral models built to verify their functions.

3.2.1 Voltage Control Delay Line

VCDL is the essential part of all DLL structures. It consists of a chain of delay cells which has one cycle of phase shift from the input to output.

The consideration about the size of delay line is the tradeoff between tuning resolution and power consumption. The phase interpolation based architecture in [12] uses up to 128 taps, which means 128 delay cells in the delay line. Later works ([11],

[13]) count on finite state machine to control fine tuning resolution, and hence significantly reduce the size of delay line. This work moderately chooses 13 delay cells and 10 taps of the delay line, as shown in Figure 3-2. The delay is the same for each cell, which is controlled by the voltage from the loop filter.

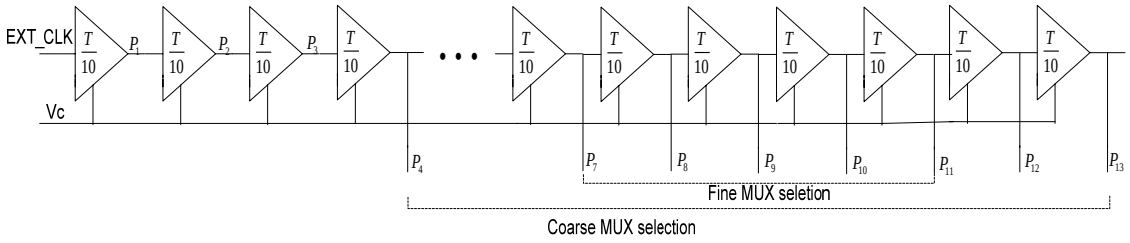

Figure 3-2: Design of the VCDL

The relationship between control voltage (V_C) and delay time for each cell (T_D) is designed as follows:

$$T_D = -\frac{T_{REF}}{20} \cdot V_C + \frac{T_{REF}}{10} \tag{3.1}$$

where $\frac{T_{REF}}{10}$ is the nominal delay, and T_D can change through V_C which ranges from -1 to 1.

Initially, all the delay cells are delayed by $\frac{T_{REF}}{10}$, and the output of the 10th delay cell, P_{10}, is feedback to compare with the delay line input (P_0) to form a conventional DLL. This will pre-define each delay cell with a delay time of $\frac{T_{REF}}{10}$ for the coarse loop adjustment.

Although 10 delay cells are needed to cover one clock period, $P_4 \sim P_{13}$ are designed for coarse phase selection rather than $P_1 \sim P_{10}$. This will relax the performance

requirement of the delay cell and also simplify the fine loop control. The design trade-off is explained with the two worst case considerations.

For the first scenario, where $P_1 \sim P_{10}$ is chosen for coarse phase selection, the worst case occurs when P_1 is selected as the coarse phase. In order for the fine loop to cover the whole tuning range ($\frac{T_{REF}}{10}$), the delay needs to be varied from $\frac{T_{REF}}{10}$ to $\frac{2T_{REF}}{10}$, which is 100% delay variation. In addition, the amount of delay cells required in the core loop will reduce from 10 to 5 through fine loop tuning. Both of the above conditions pose stringent design requirements on the delay cell and the fine loop control.

For the second scenario, where $P_4 \sim P_{13}$ is chosen for coarse phase selection, the worse case happens when P_4 is selected. Now, the delay cell only needs to vary from $\frac{T_{REF}}{10}$ to $1.25 \times \frac{T_{REF}}{10}$ in order to cover the whole tuning interval, which is only 25% delay variation. Furthermore, the size of delay line after fine tuning will reduce from 10 to 8, which costs less drastic change.

Similarly, the $P_9 \sim P_{18}$ scenario is also analyzed. Table 3-1 compares the performance of different scenarios, and the $P_4 \sim P_{13}$ scenario is the best option for our design.

Table 3-1: Comparison of different coarse phase selection scenarios

Coarse phase selection	Amount of delay cells in the core loop	Total length of VCDL	Delay variation
$P_1 \sim P_{10}$	5~10	10	100%
$P_4 \sim P_{13}$	8~10	13	25%
$P_9 \sim P_{18}$	9~10	18	11.1%

3.2.2 Coarse Loop Control

The coarse loop detector (PD) compares the VCDL output (Coarse_CLK) one by one with the reference clock, until the closest clock phase (leading the REF_CLK by less than $\frac{T_{REF}}{10}$) is found. Figure 3-3 is the design of the coarse loop phase detector (PD). The delay cells are duplicated from the VCDL. The output signals are *updn* and *hold*.

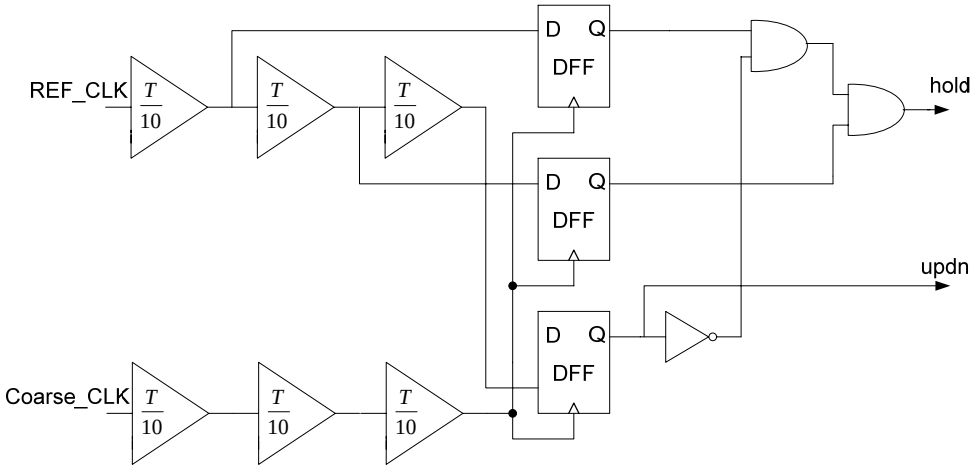

Figure 3-3: Design of the coarse loop PD

If the rising edge of the Coarse_CLK leads that of the REF_CLK, the *updn* signal is set to high; otherwise, it is set to low. The timing relationship of the Coarse_CLK and the REF_CLK and the corresponding signals can be categorized into 10 regions, illustrated as A~J in Figure 3-4 and Table 3-2. When the Coarse_CLK locates in region E, which means the Coarse_CLK leads the REF_CLK by less than $\frac{T_{REF}}{10}$, the *hold* signal goes to high. This indicates that the closest clock phase to the REF_CLK has been selected.

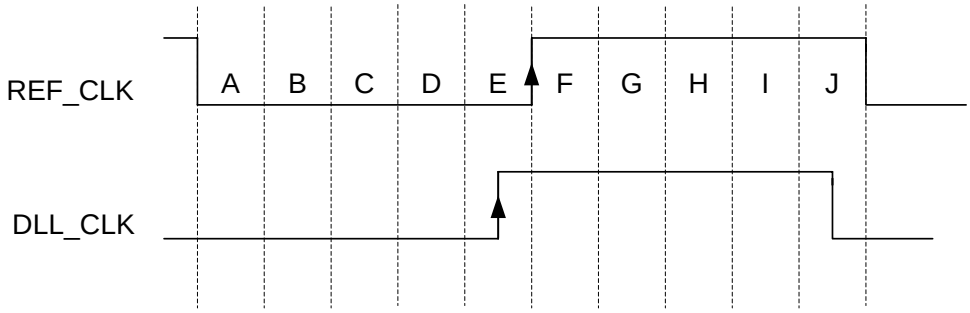

Figure 3-4: Timing of the coarse loop PD

Table 3-2: The coarse loop PD output signals

region	*updn*	*hold*
A	0	0
B	0	0
C	0	0
D	0	0
E	0	1
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0

The function of the coarse loop FSM is to dispose the *updn* and *hold* signal. The *updn* signal will control the multiplexer for coarse phase selection; whereas the *hold* signal will freeze the coarse loop and kick start the fine loop. The whole operation is best illustrated with the flow chart shown in Figure 3-5. The variable *mux* controls the coarse loop multiplexer. The *en* signal will be sent to enable the fine FSM.

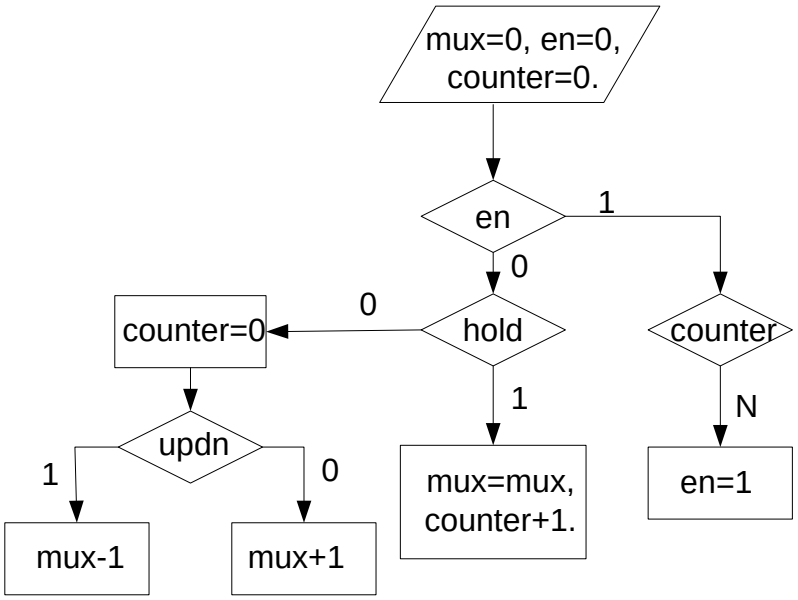

Figure 3-5: Flow chart of the coarse loop FSM

3.2.3 Delta-sigma Modulator

This work employs a $\Delta\Sigma$ modulator to generate an average clock (DLL_CLK) for phase comparison in the core DLL, with the similar idea of fractional-N PLLs. [14] Higher order $\Delta\Sigma$ modulator offers better noise shaping but suffers from stability issue. So, a second order digital $\Delta\Sigma$ modulator is designed, with the discrete model in Figure 3-6. [15] Integer arithmetic is developed with a 2-bit quantizer, and its output switches among four taps from the VCDL. The quantizer threshold, input and output range, quantized level, will be discussed later on, together with the fine loop design.

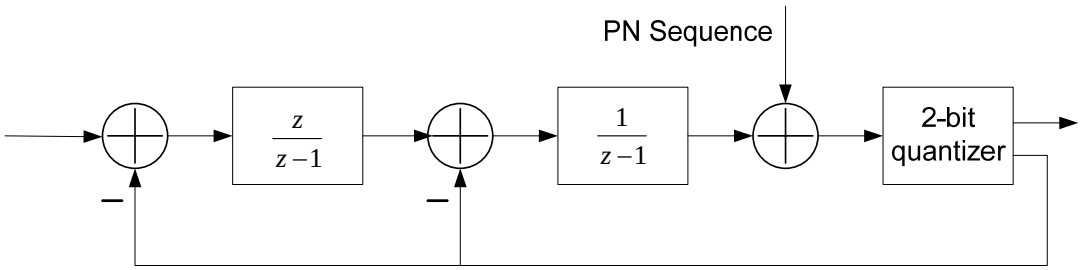

Figure 3-6: Discrete model of $\Delta\Sigma$ modulator

Based on the model, the following transfer functions can be obtained,

$$\frac{Y(z)}{X(z)} = z^{-1} \tag{3.2}$$

$$\frac{Y(z)}{Q(z)} = (1 - z^{-1})^2 \tag{3.3}$$

Simulation results verify that the output of the ΔΣ modulator is an array of integers, as shown in Figure 3-7. The data array is collected and analyzed, with the average value very close to the input value. However, it is observed that the waveform is repeated in some pattern, which converts to spurious tones and degrades jitter performance. This can be seen from the spikes in spectrum waveform of frequency response. (Figure 3-8)

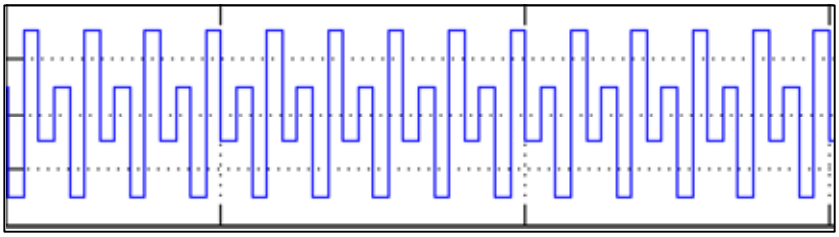

Figure 3-7: Waveform of ΔΣ output

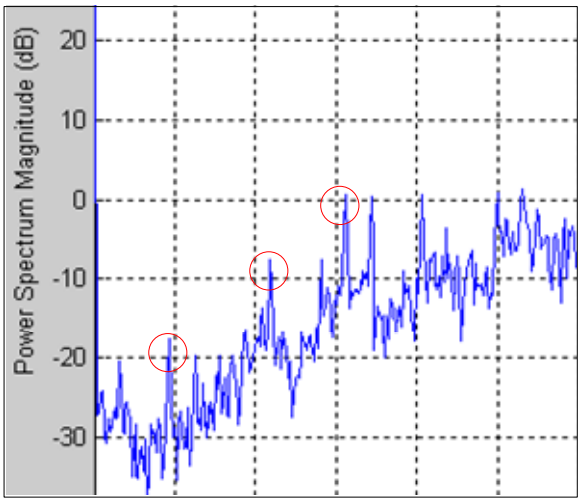

Figure 3-8: Frequency response of ΔΣ output

Dithering topology is introduced to solve this problem.[16] In this work, a pseudo random sequence is added to the quantizer to provide better randomness. The

gain needs to be carefully adjusted to produce sufficient dithering without affecting the desired average value. Figure 3-9 shows the dithered output frequency spectrum. Figure 3-10 is the corresponding output of the ΔΣ modulator with dithering. The dither amount can be characterized by the gain and the length of the pseudo random sequence. In this figure, a 22-bit shift register is placed in feedback connection to generate pseudo random sequence, with the gain of one unit of quantization level

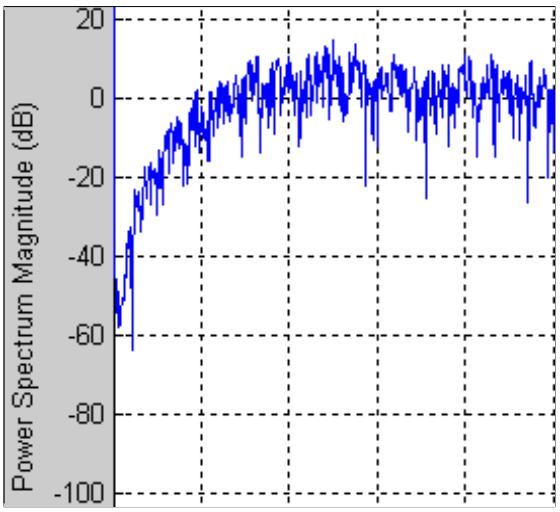

Figure 3-9: Frequency response of dithered ΔΣ output

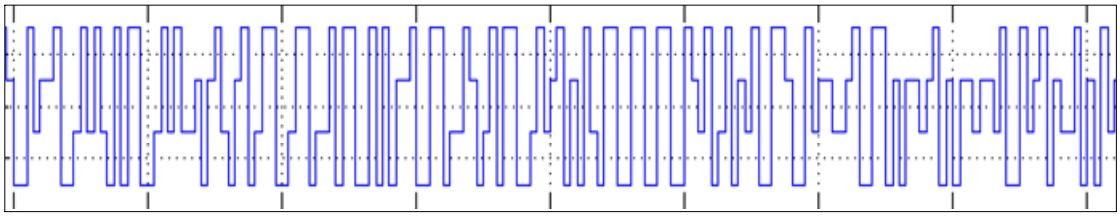

Figure 3-10: Waveform of dithered ΔΣ output

The ΔΣ modulator output controls the fine loop MUX to switch among $P_7 \sim P_{11}$. Because the quantizer is 2-bit, it requires a control signal to change the switching group between $P_7 \sim P_{10}$ and $P_8 \sim P_{11}$. Similarly, the trigger clock for ΔΣ modulator also needs to be changed for each switching group. P_7 is used for the switching group $P_8 \sim P_{11}$, whereas P_6 is used for the switching group $P_7 \sim P_{10}$.

3.2.4 Fine Loop Control

After the coarse loop selects the correct Coarse_CLK, the fine loop is enabled to synchronize the Coarse_CLK with the REF_CLK. This is achieved by changing the input of ΔΣ modulator, which in turns modify the delay of the Coarse_CLK.

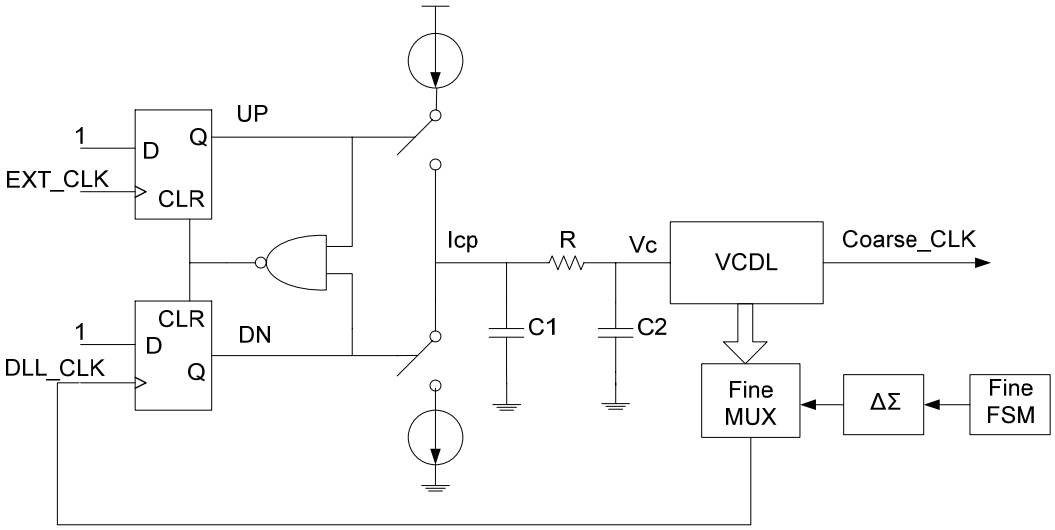

Figure 3-11: Fine loop DLL

Figure 3-11 shows the structure of the fine loop. The whole loop functions very similarly to the conventional DLL, but with one critical difference. The ΔΣ modulator controls the MUX to generate an average feedback clock, DLL_CLK. The feedback clock may not be exactly one clock period delayed. However, on the average, this DLL_CLK will have exactly one clock interval delay.

The fine loop phase detector is a traditional phase frequency detector (PFD). [17] But this work requires higher order low-pass filter to suppress the high frequency noise introduced by the ΔΣ modulator. In Figure 3-11, a 2nd order loop filter is designed with two capacitors and one resistor, and the transfer function is

$$F(s) = \frac{V_C}{I_{CP}} = \frac{1}{s^2 C_1 C_2 R + s(C_1 + C_2)}. \tag{3.4}$$

This in turns give rise to the following close loop transfer function,

$$H(s) = \frac{D_O}{D_I} = \frac{F_{REF} I_{CP} K_{DL}}{s^2 C_1 C_2 R + s(C_1 + C_2) + F_{REF} I_{CP} K_{DL}}. \quad (3.5)$$

The above equation can be further simplified into

$$H(s) = \frac{k / C_1 C_2 R}{(s + k / C_1)(s + 1/C_2 R)}, \quad (3.6)$$

where k is $F_{REF} I_{CP} K_{DL}$, and the two poles are respectively designed as $\omega_{P1} = k / C_1$ and $\omega_{P2} = 1/C_2 R$.

By assuming $\omega_{P2} >> \omega_{P1}$, which means the loop parameters, bandwidth (BW) and phase margin (PM) can be estimated as follows,

$$\text{BW} = \omega_{P1} = k / C_1 \quad (3.7)$$

$$\text{PM} = 90° - \tan^{-1}(\frac{\omega_{P1}}{\omega_{P2}}). \quad (3.8)$$

For the stability and settling time concern, it is required that the BW to be approximately $\frac{1}{10}$ of the reference frequency, and the PM to be around 65° ($\omega_{P2} \approx 2.2\,\omega_{P1}$) to avoid overshooting problem. So, $\omega_{P1} \approx 20$MHz and $\omega_{P2} \approx 45$MHz at 200MHz are estimated in behavior simulation.

The fine loop FSM realizes the following working principle to adjust the DLL_CLK towards the REF_CLK. It increases DLL_CLK delay in small steps. The maximum delay interval for the fine loop ($\frac{T_{REF}}{10}$) is further divided by N, so the tuning resolution becomes $\frac{T_{REF}}{10 \times N}$, where N is programmable depending on the operating clock frequency.

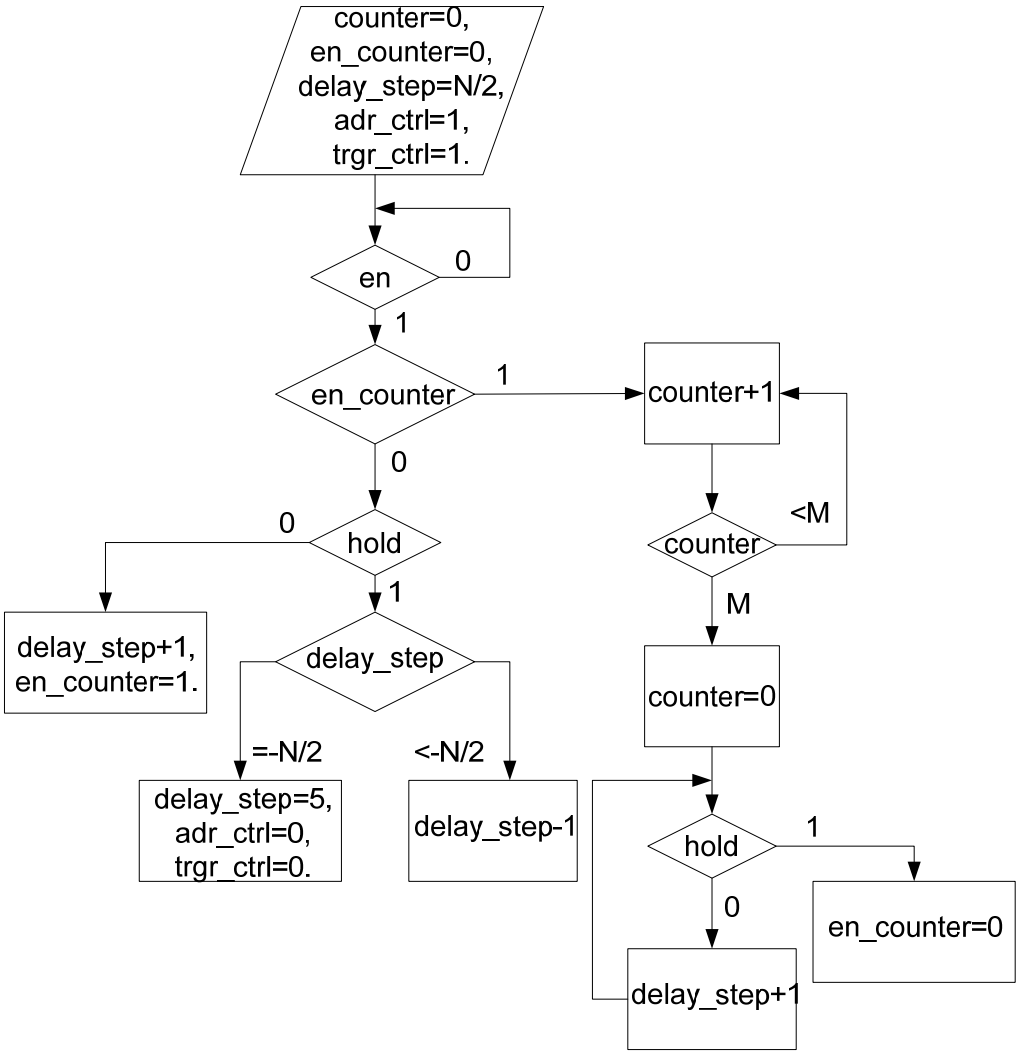

Figure 3-12: Flow chart of the fine loop FSM

As illustrated in the flow chart of the fine loop FSM (Figure 3-11), the variable *delay_step* represents this tuning resolution, within the range from $\frac{N}{2}$ to $-\frac{N}{2}$. It is updated according to the *hold* signal. Once the *hold* signal jumps down, which means the Coarse_CLK becomes lag to the REF_CLK, the state of the *delay_step* stays fixed. If the *delay_step* variable reaches $-\frac{N}{2}$ without the *hold* signal being low, the switching group needs to be changed by the signal, *adr_ctrl* and *trgr_ctrl.* After the fine tuning is finished, FSM starts an internal timer to check the *hold* signal every M clock cycles. This is used to release the fine tuning in case that the synchronized clock becomes

misaligned due to environmental variation. This waiting period can be estimated by assuming that the synchronized clock drifts away by 100ppm. For example, it takes at least 100 clock cycles for the locked clock edges to differ by 0.0001 at 200MHz.

The tuning resolution of the fine loop FSM also determines the input and output range of the ΔΣ modulator. The input comes from the *delay_step* variable in the fine loop FSM, which are digits from $\frac{N}{2}$ to $-\frac{N}{2}$. It will be scaled to 0~3 through the quatnizer, and then added with the *adr_ctr*l signal (0 or 1) to control the fine loop phase selection among $P_7 \sim P_{11}$. Figure 3-13 indicates the design of the 2-bit quantizer, of which the threshold values and the quantized feedback values are also programmable.

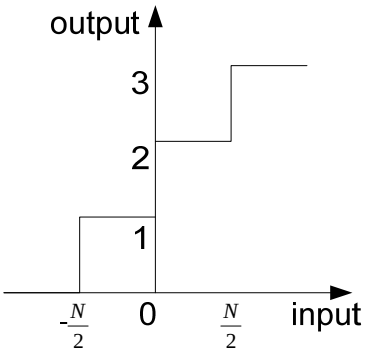

Figure 3-13: Design of the quantizer in ΔΣ modulator

3.3 System Integration and Simulation

The whole system behavior can be best illustrated through an example shown below. In general, assuming the REF_CLK is initially lagging the EXT_CLK by $\Delta D + \frac{T_{REF}}{10} \cdot N$, where $\frac{T_{REF}}{10}$ is the delay of each delay cell, the coarse loop will select P_N as the Coarse_CLK. This will reduce the phase difference between the final output

clock and the REF_CLK down to ΔD. The fine loop is then activated to modify the delay of each cell from $\frac{T_{REF}}{10}$ to $\frac{T_{REF}}{X}$, in order to absorb this difference of ΔD. This is achieved through the delta-sigma modulator, which is able to generate fractional division ratio X with fine delay resolution. The total delay needed for clock synchronization is best described by the following equation,

$$\Delta D + \frac{T_{REF}}{10} \cdot N = \frac{T_{REF}}{X} \cdot N = (-\frac{T_{REF}}{20} \cdot V_C + \frac{T_{REF}}{10}) \cdot N \tag{3.9}$$

where the total delay can also be expressed in terms of the control voltage, V_C.

The number of steps required to acquire synchronization in the fine loop depends on the operating clock frequency. For 200MHz clock frequency, 10 steps are needed with each step finely adjusting the clock phase by 50ps. When the fine loop is first activated, the input to the delta-sigma modulator (the *delay_step* variable) is first reset to 5 and reduced by 1 subsequently. The phase difference between the output clock and the REF_CLK is examined after every adjustment of the delta-sigma input to determine whether there is any phase change. If the phase difference changes from phase lagging to phase leading, clock synchronization is achieved and the subsequent adjustment is stopped. Otherwise, the fine loop continues its operation.

The required fractional division ratio X determines the expected value of the *delay_step* variable. However, due to the finite value of the *delay_step,* it will result in quantized fractional division ratio $\hat{X}$. The relationship between the input variable *delay_step*, the fractional division ratio (X), and the quantized fractional division ratio ($\hat{X}$) is shown as follows,

$$X \approx P_B + 0.5 + \frac{delay\_step}{10} = \hat{X} \tag{3.10}$$

where P_B is 8 if the 4 feedback phases in the DLL is $P_7 \sim P_{10}$, and P_B is 9 if the 4 feedback phases in the DLL is $P_8 \sim P_{11}$.

Table 3-3 shows the desired values of the different variables for different initial phase difference with operating clock frequency of 200MHz.

Table 3-3: Testing cases of the system

Initial delay error (ns)	Coarse selection P_N (N)	Required division ratio X	Quantized division ratio $\hat{X}$	Final state of the *delay_step* variable	Vc
2.25	4	8.88	9	4	-0.25
2.75	5	9.09	9.1	-4	-0.2
3.25	6	9.23	9.3	-2	-0.17
3.75	7	9.33	9.4	-1	-0.14
4.25	8	9.41	9.5	0	-0.12
4.75	9	9.47	9.5	1	-0.11
0.25	10	9.52	9.6	1	-0.1
0.75	11	9.56	9.6	0	-0.09
1.25	12	9.6	9.6	0	-0.08
1.75	13	9.63	9.7	0	-0.07

As an example, if the initial delay error is set to 3.75ns, the system simulation shows that P_7 is chosen as the Coarse_CLK. This is verified by the *mux* variable shown in Figure 3-14. It should be pointed out that the *mux* value of 0~9 corresponds to clock phase of $P_4 \sim P_{13}$ respectively.

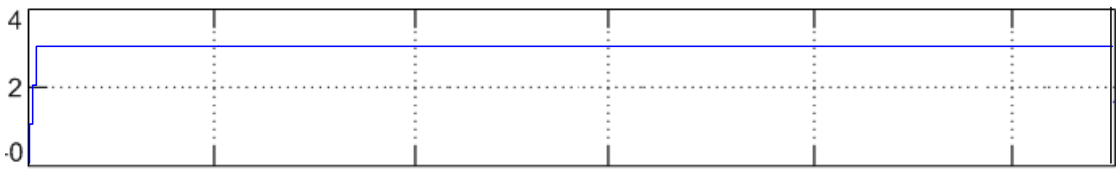

Figure 3-14: Simulation of coarse loop

The operation of the fine loop is verified through the simulated *delay_step* and Vc, as illustrated in Figure 3-15. It was shown that and the *delay_step* variable and Vc

lock to -1 and -0.13 respectively once clock synchronization is achieved. In addition, the simulated $\hat{X}$ can be obtained by averaging out the output of the delta-sigma modulator, and is found to be 9.43 from the simulation. This is comparable to the estimated $\hat{X}$ and Vc of 9.4 and -0.14 respectively from Table 3-3 .

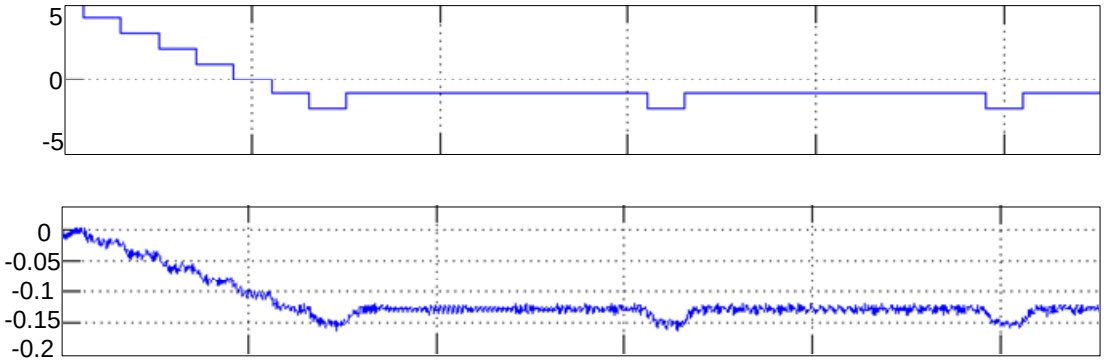

Figure 3-15: Simulation of fine loop

The various clock phases before and after clock synchronization are displayed in Figure 3-16 and Figure 3-17 respectively. It is demonstrated that the desired output clock P_7 locked to the REF_CLK after clock synchronization.

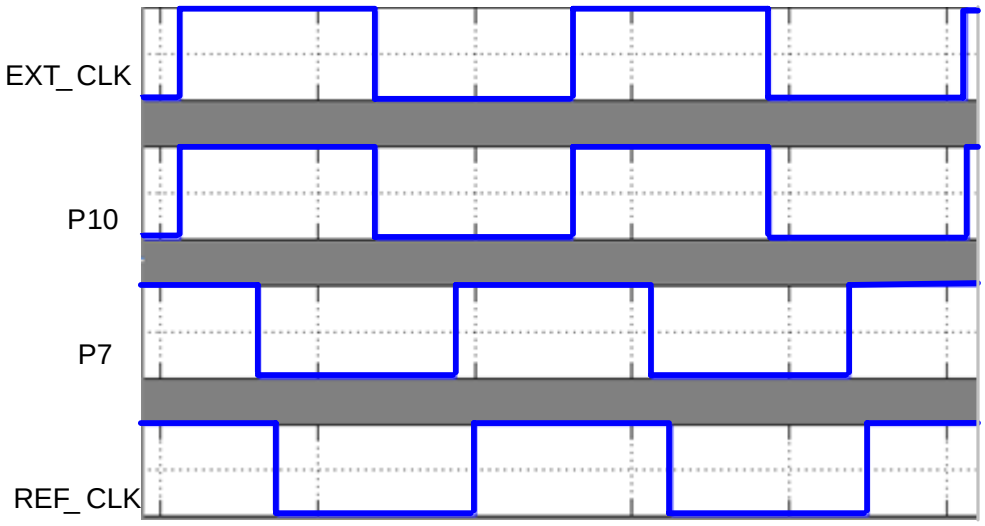

Figure 3-16: DLL clock before synchronization

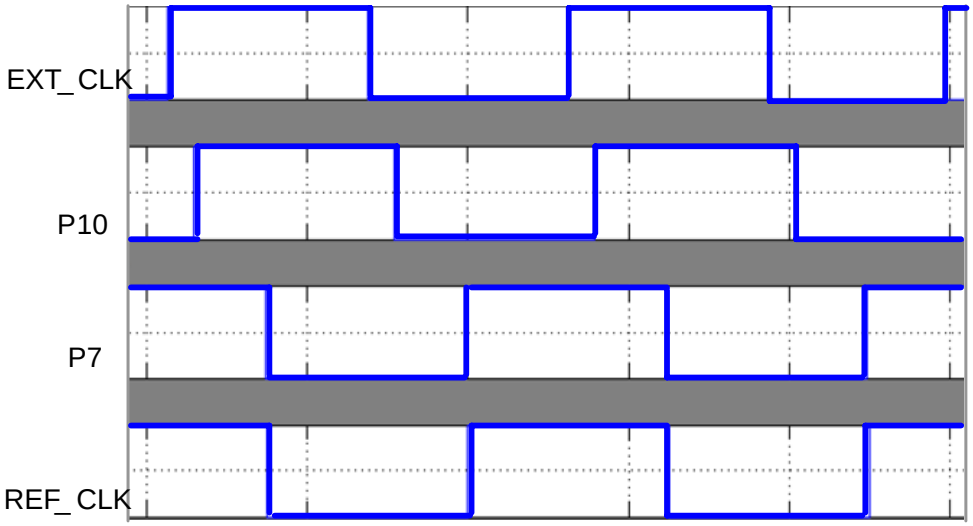

Figure 3-17: DLL clock after synchronization

In next chapter, the CMOS implementation of this fractional-N DLL architecture will be discussed in detail.

CHAPTER 4 CMOS IMPLEMENTATION

This fractional-N DLL is implemented in 0.35 μm CMOS technology. The implementation includes both analog circuit design and digital synthesis. The operating frequency range covers from 10MHz to 200MHz, with 3V power supply.

4.1 Analog Circuit Design

4.1.1 Delay Cell

Voltage controlled delay line consists of cascoded delay cells, which is one of the most critical blocks in DLL system, because the performance of the VCDL considerably affects the stability and jitter performance. Unlike in all-digital DLL system where basic inverters are simply used as the delay cell, analog DLLs employ many different configurations for their delay cells. Generally, it can be divided into single-ended type and differential type. For example, Figure 4-1 shows a digital RC delay cell [18] and a current starved single-ended delay cell [19], respectively. But differential delay cells are more widely used because of their inherent advantages, such as better immunity to common-mode noise and improved spectral purity.

Typically, the differential delay cell involves a source coupled differential pair with two load elements and a biasing tail current source. An ideal tail current source would be highly immune to static supply noise and cost small voltage headroom. The load element determines the relationship between delay time, control voltage, the range of output swing, and so on. It can be implemented either in a diode-connected configuration or in a triode-connected configuration, as shown in Figure 4-2. The diode loads have simple structure but consume voltage headroom, so the output

voltage swing is limited. In addition, the load transistors work in saturation region, so the output DC voltage is not controllable. The triode loads work in linear region, thus provide wider output voltage swing. However, the drawback is the difficulty to maintain the load transistors in linear region.

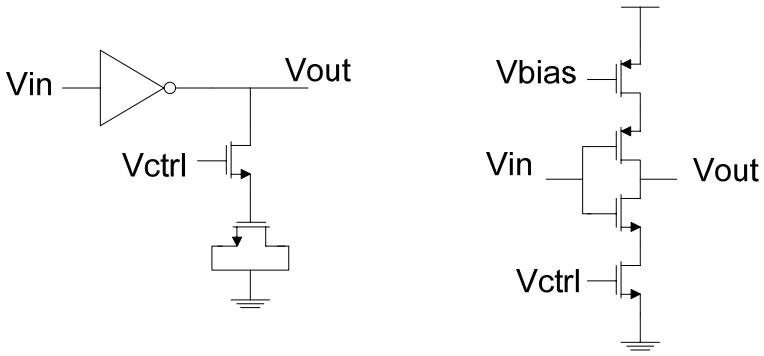

Figure 4-1: Other VCDL delay cells

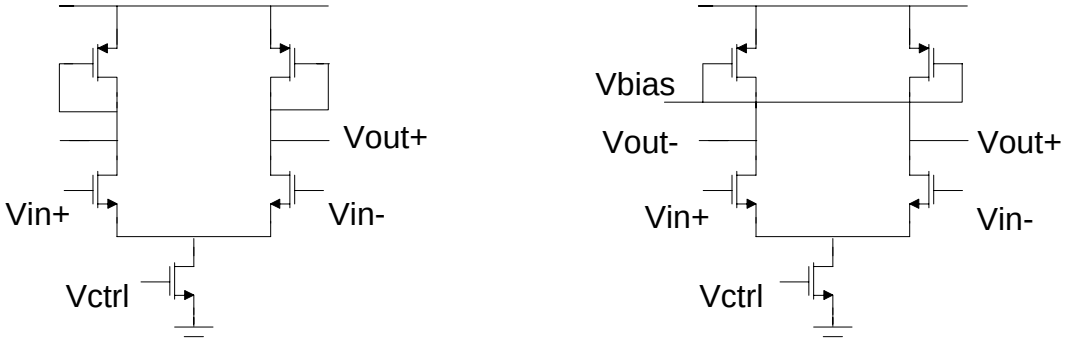

Figure 4-2: Differential delay cell with diode loads and triode loads

The schematic of differential delay cells in this work is a source coupled pair with symmetric loads, which is an improvement of the above two configurations. In Figure 4-3, the load element is comprised of an equal-sized PMOS pair in shunt, one being diode-connected (M3) and the other one being active load (M4) controlled by V_{BP}. The delay is proportional to its RC time constant, of which the effective load resistance can be adjusted by V_{BP}. It has been proved that this symmetrical load can provide good control over the cell delay as well as high dynamic supply noise rejection. The simple NMOS current source is biased by V_{BN}, which is less susceptible to static supply and substrate noise. [6]

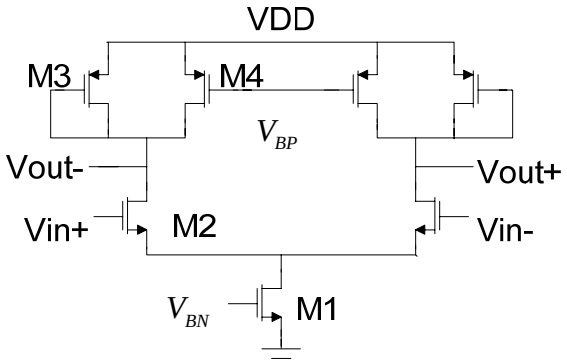

Figure 4-3: Differential delay cell with symmetric loads

The load resistance can be calculated as

$$R_L = \frac{1}{g_{m3}} \| r_{ds2} \| r_{ds4} \approx \frac{1}{g_{m3}}, \tag{4.1}$$

where $g_{m3} = u_p C_{ox} (W/L)_P (V_{DD} - V_{BP} - |V_{TP}|)$.

Thus, the relationship of delay time for each stage with respect to V_{BP} is

$$T_D = R_L C_L = \frac{1}{g_{mp}} \cdot C_L = \frac{C_L}{u_p C_{ox} (W/L)_P (V_{DD} - V_{BP} - |V_{TP}|)}. \tag{4.2}$$

Notice that C_L is the effective load capacitance which combines all the parasitic capacitance of M2, M3 and M4, also including gate capacitance from other load components, such as the next delay stage and multiplexer.

Another consideration is the gain of this differential pair. Since the VCDL is a cascode chain of delay cells, the gain of each stage must satisfy A>1, to guarantee no signal loss. However, if the gain is too high, the thermal noise from previous stage is also amplified, and hence the timing error. Therefore, the gain is kept as small as possible with a sufficient safety margin, including some process variation tolerance.

It is considerably designed that A=1.5, which can be derived as

$$A = g_{mn} \cdot \frac{1}{g_{mp}} = \frac{\sqrt{u_n C_{ox} (W/L)_N I_{DN}}}{\sqrt{u_p C_{ox} (W/L)_P I_{DP}}}. \tag{4.3}$$

where, $I_{DN} = 2 I_{DP}$. When the differential pair is fully switched, the output voltage swing for each cell is

$$V_{SW} = 2(V_{DD} - V_{BP}) \tag{4.4}$$

The delay time is simulated at 200MHz in typical case, as shown in Figure 4-4. By adjusting V_{BP}, the delay time $\frac{T_{REF}}{10}$ (500ps) is achieved at V_{BP} =1.8V. The output voltage swing (peak to peak) is 2.35V.

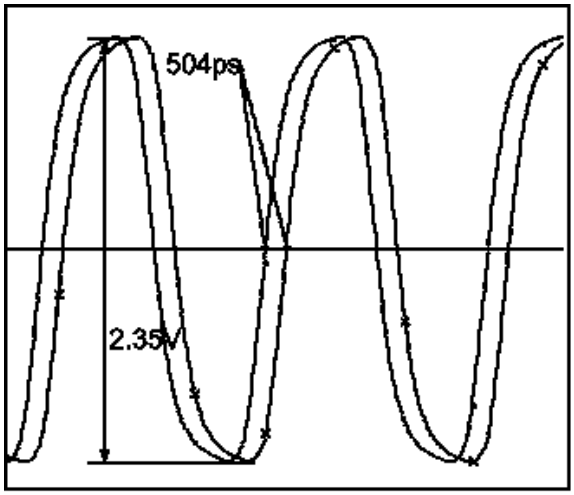

Figure 4-4: Simulation of delay cell at 200MHz

Since the relationship of T_D and V_{BP} is nonlinear, simulation results are shown in Table 4-1. For different operating frequency, the related nominal delay ($T_D = \frac{T_{REF}}{10}$) can be achieved by varying V_{BP}. In Figure 4-5, T_D increases dramatically when V_{BP} goes close to $V_{DD} - |V_{TP}|$.

Table 4-1: Simulation results of T_D and V_{BP}

F_{CLK} (MHz)	Nominal T_D (ns)	V_{BP} (V)
10	10.4	2.235
50	1.98	2.1
100	1.02	2
200	0.5	1.8
400	0.25	1.48

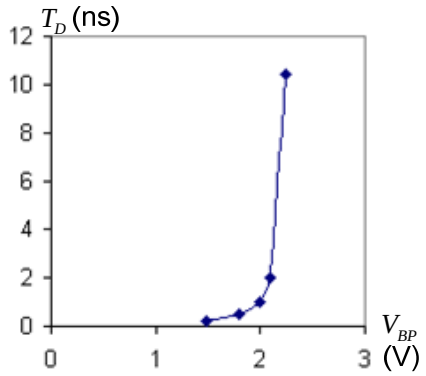

Figure 4-5: Nonlinear relationship of V_{BP} vs. T_D

Since V_{BP} is the loop filter output voltage, its variation will transfer to the jitter noise of the delay cell. The minimum delay tuning resolution should be larger than this clock jitter. From the system simulation, the peak-peak variation of V_{BP} is 0.02V. So the peak-peak jitter which per delay cell can tolerate is around 20ps (multiplied by the delay gain, 1ns/V)

4.1.2 Replica Bias

Replica bias technique is employed to provide the control voltage V_{BP} and the bias current for the delay cells. As shown in Figure 4-6, the main structure is two duplicated circuits of half the delay cell, one as replica stage and the other as V_{CTRL}

buffer stage. A differential op-amp keeps tracking the voltage at the load element to V_{CTRL} with feedback, which keeps the PMOS pair (M1) in saturation region. Meanwhile, the output of op-amp set biasing voltage V_{BN} to NMOS current source (M3) so that the V_{CTRL} buffer stage and all the following delay cells have the same operation condition. Rather than directly using V_{CTRL} for delay cells, the control voltage is buffered as V_{BP} to isolate replica stage from the load effect by delay cells. Another advantage is that this simple current source is independent of the supply voltage and hence has high static supply noise rejection.

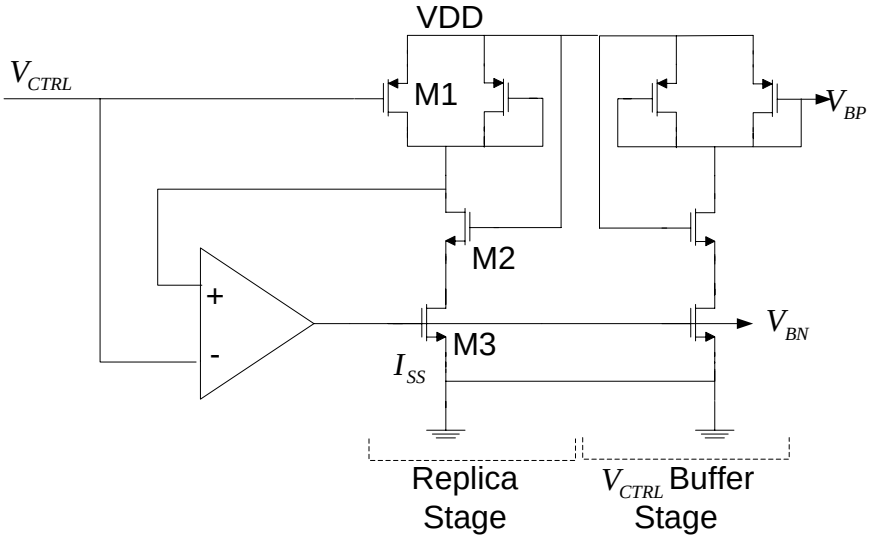

Figure 4-6: Schematic of replica bias

To achieve wide tuning range of V_{CTRL}, a wide-swing differential op-amp is used to extend the allowable input voltage range.[20] It uses two complementary input differential amplifier stages in parallel, to ensure sufficient gain when the input DC voltage is either too high or too low. The minimum gain is designed to be 40dB for offset consideration.

Practically, for a specific reference frequency, V_{CTRL} should cover certain range to fulfill the required delay, $\frac{T_{REF}}{X}$, where X is 8~10 as calculated before. Taking F_{REF} = 200MHz as an example, the desired delay time for each stage is 500ps~625ps. From Equation(4.2), V_{CTRL} have to reach 1.8V~1.91V. The control voltage may need minor adjustment because the load effective capacitance is not a constant for the whole frequency range, which is finally simulated in Table 4-2. The table also shows the current that flows through the replica stage, which is mirrored to all the delay cells. Theoretically, the current satisfies the following equation as long as the PMOS load (M1) stays in saturation region.

$$I_{SS} = 2I_{DP} = 2u_p C_{ox}(\frac{W}{L})_3(V_{DD} - V_{CTRL} - |V_{TP}|)^2 \quad (4.5)$$

When V_{CTRL} gets quite close to $V_{DD} - |V_{TP}|$, the PMOS load will fall into sub-threshold region, consuming very little current.

Table 4-2: Simulation results of V_{CTRL} range

F_{REF} (MHz)	N	T_D (ns)	V_{CTRL} (V)	I_{SS} (μA)
400	10	0.25	1.47	546
	8	0.31	1.62	374
200	10	0.49	1.8	198
	8	0.63	1.9	127
100	10	1.02	2	74.4
	8	1.22	2.03	60
50	10	1.98	2.1	32.6
	8	2.55	2.14	21.3
10	10	10.5	2.233	5.09
	8	12.5	2.242	4.3

In addition, 20% design margin is included for the delay cell for process variation consideration.

4.1.3 Phase Detector

In the fine loop, phase detector compares the phase error of the REF_CLK and the DLL_CLK, therefore produces pulse signal to control the charge pump. The conventional structure, as modeled in behavior level, has the limitation of initial condition.[21] For both input clocks, the rising edge generates a short pulse (UP or DN), and the difference between their pulse widths determines the leading or lagging relationship. However, in DLL system, the first rising edge of the DLL_CLK is feedback to compare with the second rising edge of the REF_CLK, as shown in Figure 4-7. To avoid that PD falsely compares the DLL_CLK with the original rising edge of the REF_CLK, as illustrated in Figure 4-8, a modified PD structure is introduced with a start-up circuit.

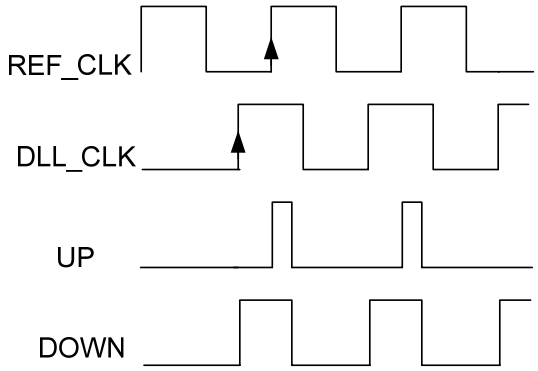

Figure 4-7: Desired timing of PD in DLL

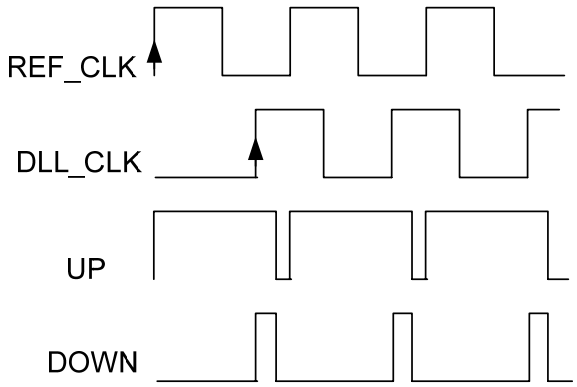

Figure 4-8: False comparison of conventional PD

In Figure 4-9, an additional D flip-flop is placed at the first stage of a conventional phase detector, and this configuration functions correctly for any initial condition. When the START signal is low, the PD is disabled and the RDY signal is low too. Once the START signal goes high, the PD is enabled. At the 1st rising edge of the REF_CLK, the RDY signal will be latched to high. It should be pointed out that the whole PD works similarly to the conventional PD once the RDY signal is high. Therefore, this start-up structure successfully hides the 1st rising edge of the REF_CLK for false comparison.

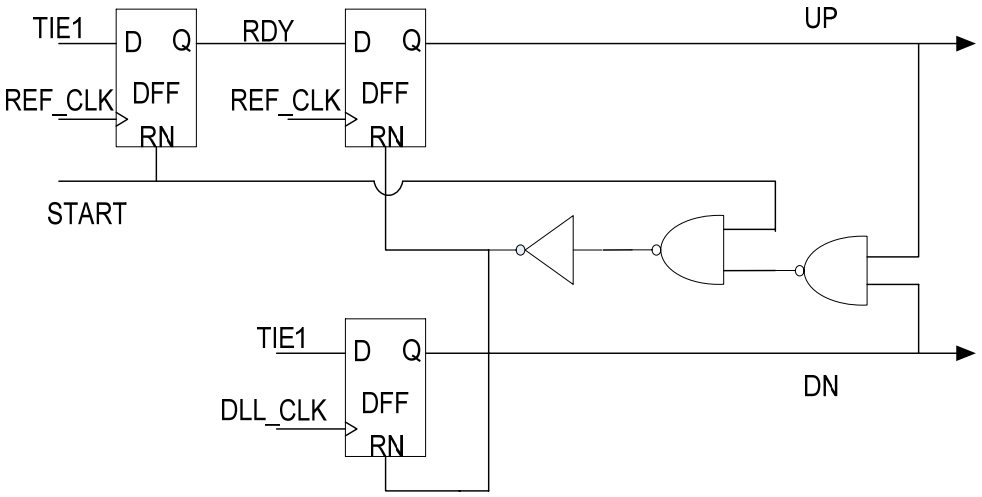

Figure 4-9: Schematic of PD with start-up function

Figure 4-10 shows the simulation result when pre-defined delay error is set between the REF_CLK and the DLL_CLK.

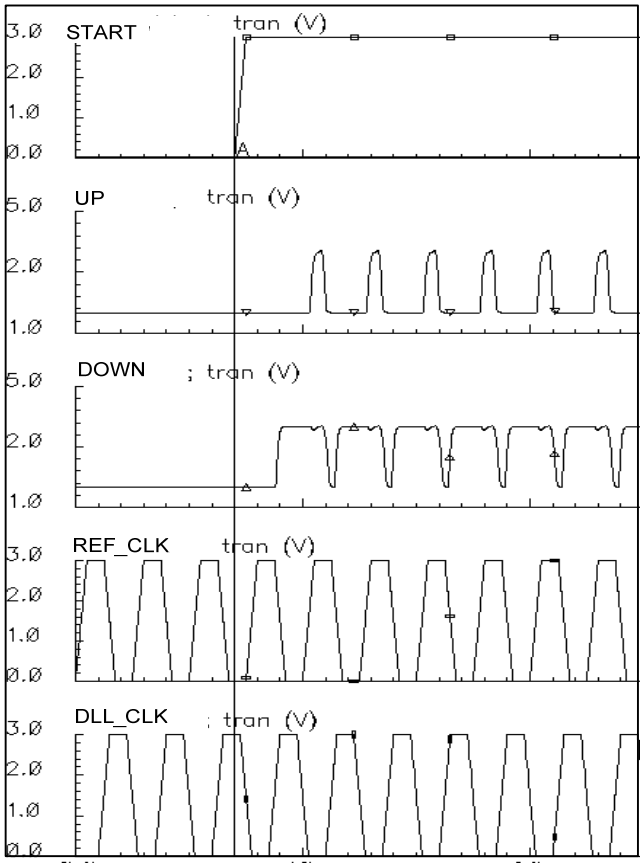

Figure 4-10: Simulation of PD with start-up function

When the REF_CLK and the DLL_CLK are in phase, the pulse width of UP and DN are the same, which is also the shortest pulse that UP or DN could have. This pulse width is designed to be around $\frac{T_{REF}}{5}$ to avoid deadzone problem.

4.1.4 Charge Pump

Charge pump is the block following phase detector to convert the phase difference to current. Figure 4-11 illustrates the schematic of the charge pump. The UP and DN signal from the phase detector switch on or off the charge pump current. When either UP or DN is high, I_{CP} is charging or discharging the loop filter, and hence V_{CTRL} is increasing or decreasing, correspondingly.

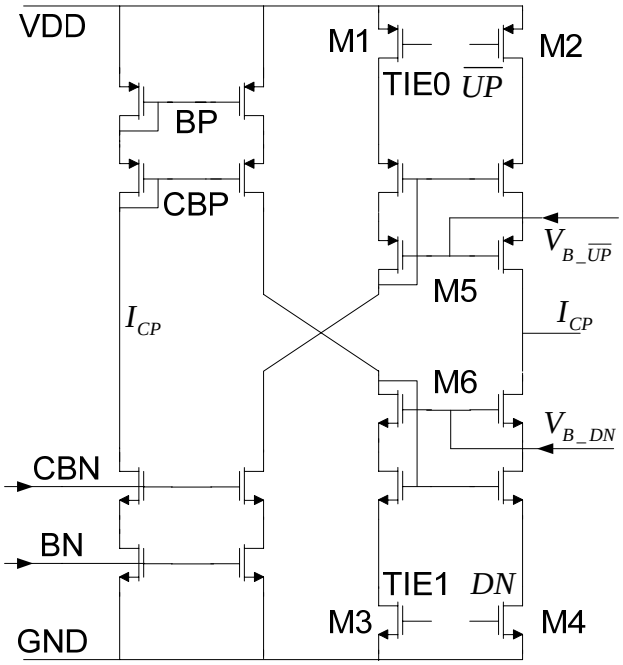

Figure 4-11: Schematic of charge pump

Since V_{CTRL} covers a very wide range, high swing current mirrors are used in the schematic (Figure 4-11). The current I_{CP} is controlled by MOS transistor switches, M2 and M4. They are placed away from the charge pump output such that the switching won't cause severe glitches to the output. In this configuration, the voltage swing at CP output is from $2V_{DSAT6}+V_{DS4}$ to $V_{DD}-(2V_{DSAT5}+V_{DS2})$.To minimize the headroom, the size of M2 and M4 should be large.

To achieve the highest possible swing and to track the process variation, the bias voltage of $V_{B\_\overline{UP}}$ and V_{B_DN} are generated by the cascode biasing circuit as shown in Figure 4-12. In addition, dummy switches are used to provide better matching.

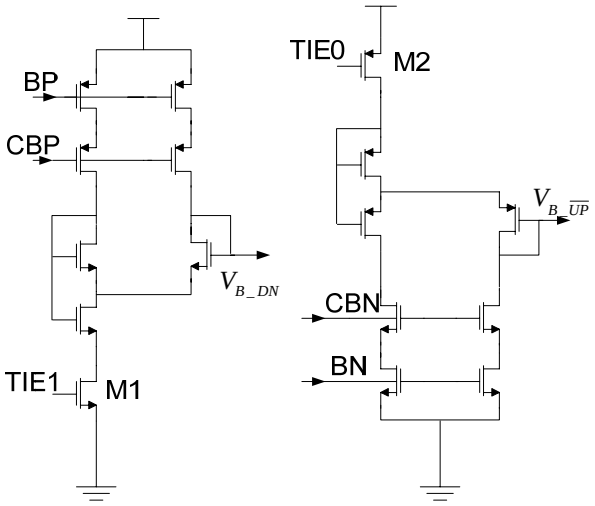

Figure 4-12: Schematic of cascode biasing

The charge pump current I_{CP} is also mirrored and scaled from replica bias I_{SS}, but it is programmable to achieve adaptive bandwidth for wide operation range, which is designed according to $\frac{BW}{F_{REF}} = \frac{I_{CP}K_{DL}}{C_1} = \frac{1}{10}$. In order to maintain this ratio, $I_{CP}K_{DL}$ is required to be constant so that the bandwidth of this DLL system goes adaptive with different reference frequency. An obvious choice is to make I_{CP} inversely proportional to K_{DL}. So the inverse relationship between I_{SS} and K_{DL} are desired to achieve the above objective. However, this relationship is far from ideal, and some proper scaling needs to be introduced.

With the scaling factor α, $I_{CP} = \alpha I_{SS}$ results in the dotted curve in Figure 4-13. For the operating frequency range of 10MHz~400MHz, I_{CP} is designed to increase discretely with 4 different values of α to keep the ratio close to 0.1. The modified relationship is $I_{CP} = \alpha \cdot S \cdot I_{SS}$, shown as the solid line in Figure 4-13, which displays better adaptive feature,

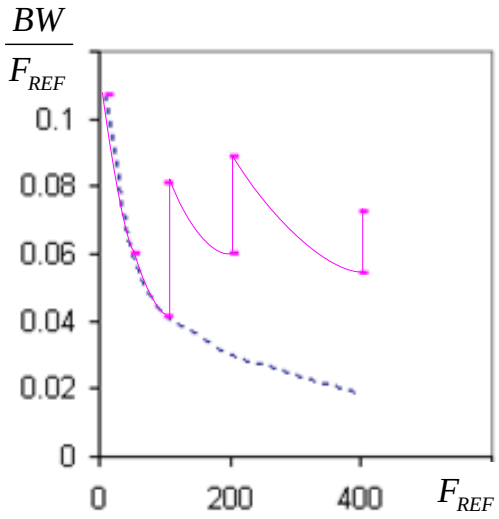

Figure 4-13: Simulation of adaptive bandwidth

This programmable current generation circuit is implemented as shown in Figure 4-14. The current comes from V_{BN} in replica bias, and is scaled by the factor α. The factor S is tunable through the digital switches S0~S2.

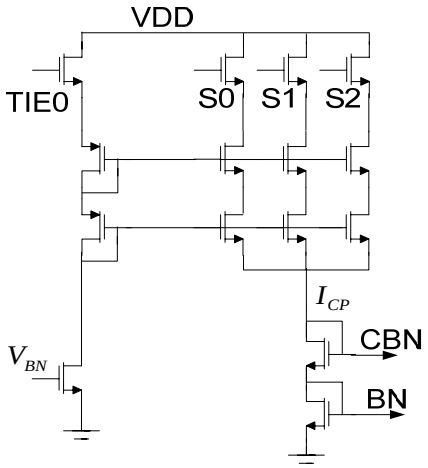

Figure 4-14: Schematic of charge pump current generator

The relationships between F_{REF} and I_{CP} are tabulated in Table 4-3. The scaling factor α is simulated to be 0.2, in order to satisfy $\frac{BW}{F_{REF}} = \frac{1}{10}$. The charge pump current has mismatch in the UP and DN branch, but within acceptable error range.

Table 4-3: Comparison of programmable I_{CP}

F_{REF} (MHz)	I_{SS} (μA)	S($S_2S_1S_0$)	Desired I_{CP} (μA)	Simulated I_{CP} (μA)	
				I (UP)	I (DN)
400	546	$3(011) \rightarrow 4(100)$	413	416.1	413.6
200	198	$2(010) \rightarrow 3(011)$	119	118.8	119.5
100	74.4	$1(001) \rightarrow 2(010)$	30	29.7	29.9
50	32.6	1(001)	6.5	6.56	6.52
10	5.09	1(001)	1	1.09	1.10

4.1.5 Adaptive Filter

The loop filter is designed with one resistor and two capacitors, as shown in Figure 3-11. Also, the system level analysis points out that the value of C_1, C_2 and R affects the two poles in the frequency domain. From the adaptive bandwidth property discussed before, C_1 is constant and chosen to be 5pF. On the other hand, the second pole is also desired to be adaptive with different reference frequency ($\omega_{P2} \propto F_{REF}$).

From last section, it is shown that ω_{P1} can be made adaptive, by exploiting the relationship between I_{SS} and K_{DL}. Similar idea is adopted to adaptively tune ω_{P2} location according to the operating frequency. As the operating frequency gets slower, K_{DL} increases whereas I_{SS} decreases. In order to reduce $\omega_{P2} = \frac{1}{RC_2}$, either R or C_2 can be increased accordingly. In this work, C_2 is kept constant and MOS resistor is employed to vary R. It is found that MOS resistance is inversely proportional to the square root of the current. Hence, by making the MOS resistor tracks the I_{SS}, the goal of tuning ω_{P2} for different operating frequency will be achieved.

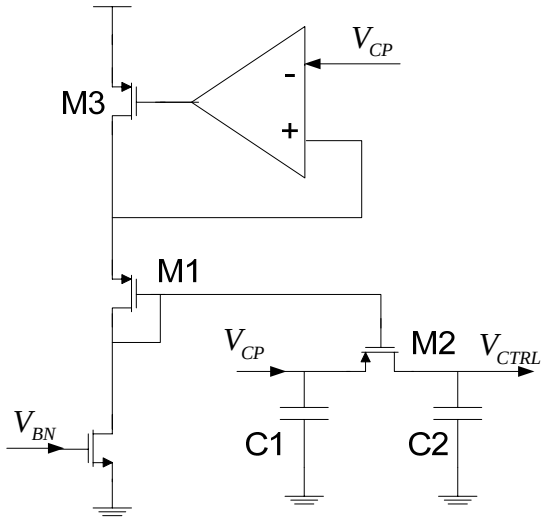

Figure 4-15: Schematic of adaptive filter

The circuit implementation of such adaptive filter with variable active resistor is constructed in Figure 4-15. A PMOS transistor, M2, acts as the resistor, and its gate is tied with the gate of diode-connected M1. The drain of M2 is connected to the output of charge pump, while the drain of M1 is also kept to V_{CP} through the feedback of an op-amp. The gain of op-amp is at least 40dB in order to maintain V_{CP} error smaller than 0.01. In this way, M1 and M2 have the same gate-drain voltage.

Thus, the variable resistor is calculated as:

$$R = \frac{1}{u_p C_{ox} (W/L)_2 (V_{GS2} - |V_{TP}|)} \tag{4.6}$$

where $V_{GS2} = V_{GS1}$, and V_{GS1} tracks the current I_{SS}. The I_{D1} is chosen to be $0.02\,I_{SS}$ to achieve reasonable R and C_2.

In Table 4-4, the variable resistor is simulated to compare with the calculated value. When the filter current drops at low reference frequency, the resistance is quite different from the expectation due to the sub-threshold operation of M1. However, the phase margin doesn't degrade significantly.

Table 4-4: Comparison of variable resistor in adaptive filter

F_{REF} (MHz)	Filter current (μA)	Designed R ($k\Omega$)	Simulated R ($k\Omega$)	PM($^\circ$)
400	10.8	7	7.2	69
200	3.96	11.4	12.6	72
100	1.49	18.6	29.2	67
50	0.65	28	56.8	77
10	0.1	72	308	63

4.1.6 Other Analog Blocks

The coarse loop PD is implemented in a straightforward manner as described in Figure 3-3, and its operation is verified by simulation.

Two multiplexers are required for phase selection. Since the MUX has differential input and output, it can be implemented using the basic delay cell in Figure 4-4, except that M1 is not connected to replica bias but works as a switch. The switches are controlled by the digital bits from FSM. According to system block diagram in Figure 3-1, a 10-to-1 MUX is connected with the 10 bits output variable (*mux[9:0]*) of the coarse loop FSM, and a 5-to-1 MUX is connected with the 5 bits output variable (*delay_step[4:0]*) of the fine loop FSM.

4.2 Digital Synthesis

4.2.1 Finite State Machine

The coarse loop FSM and fine loop FSM are synthesized into one block. They are coded with two individual modules, FSM_COARSE and FSM_FINE, which are instantiated in the top level module, FSM, as shown in Figure 4-16.

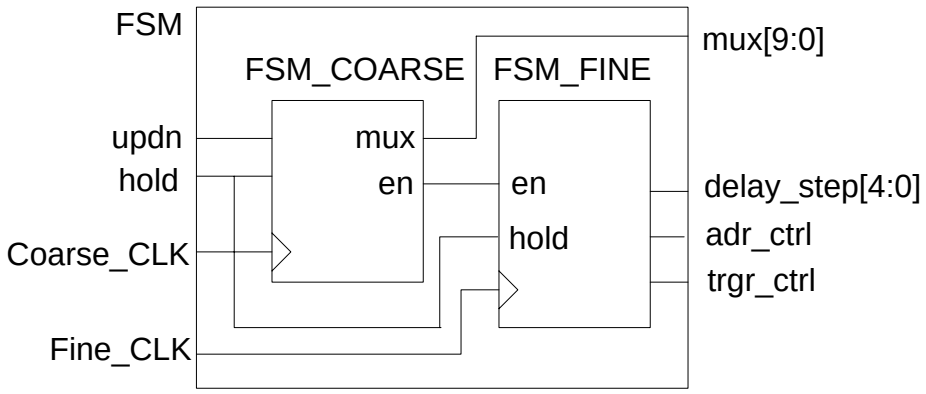

Figure 4-16: Schematic of FSM

The input ports are *updn* and *hold* signals from the coarse phase detector, which tells FSM the phase relationship between the Coarse_CLK and the REF_CLK. The output ports are *mux[9:0]* from FSM_COARSE and *delay_step[4:0]*, *adr_ctrl*, *trgr_ctrl* from FSM_FINE. It has been clearly discussed before about the connection of each port and their functions. Note that the Fine_CLK is much slower than the Coarse_CLK because every state update of the FINE_FSM needs much longer time for the whole loop to settle down.

The RTL synthesis process is constrained to meet the target in Table 4-5. The coarse loop clock period is 5ns and the fine loop clock period is 50ns. The output port is set to drive a maximum 0.5pF load. For setup time, 40% of the clock period is chosen as the timing budget. The synthesized layout is shown in Figure 4-17.

Table 4-5: Synthesis constraints for the FSM

Clock period	FSM_COARSE: 5ns	FSM_FINE: 50ns
Clock skew	0.5ns	
Driving cell	“BUF2”	
Output load	0.5pF (maximum)	
Setup timing	FSM_COARSE: 2ns	FSM_FINE: 20ns
Hold timing	FSM_COARSE: 0.3ns	FSM_FINE: 0.3ns

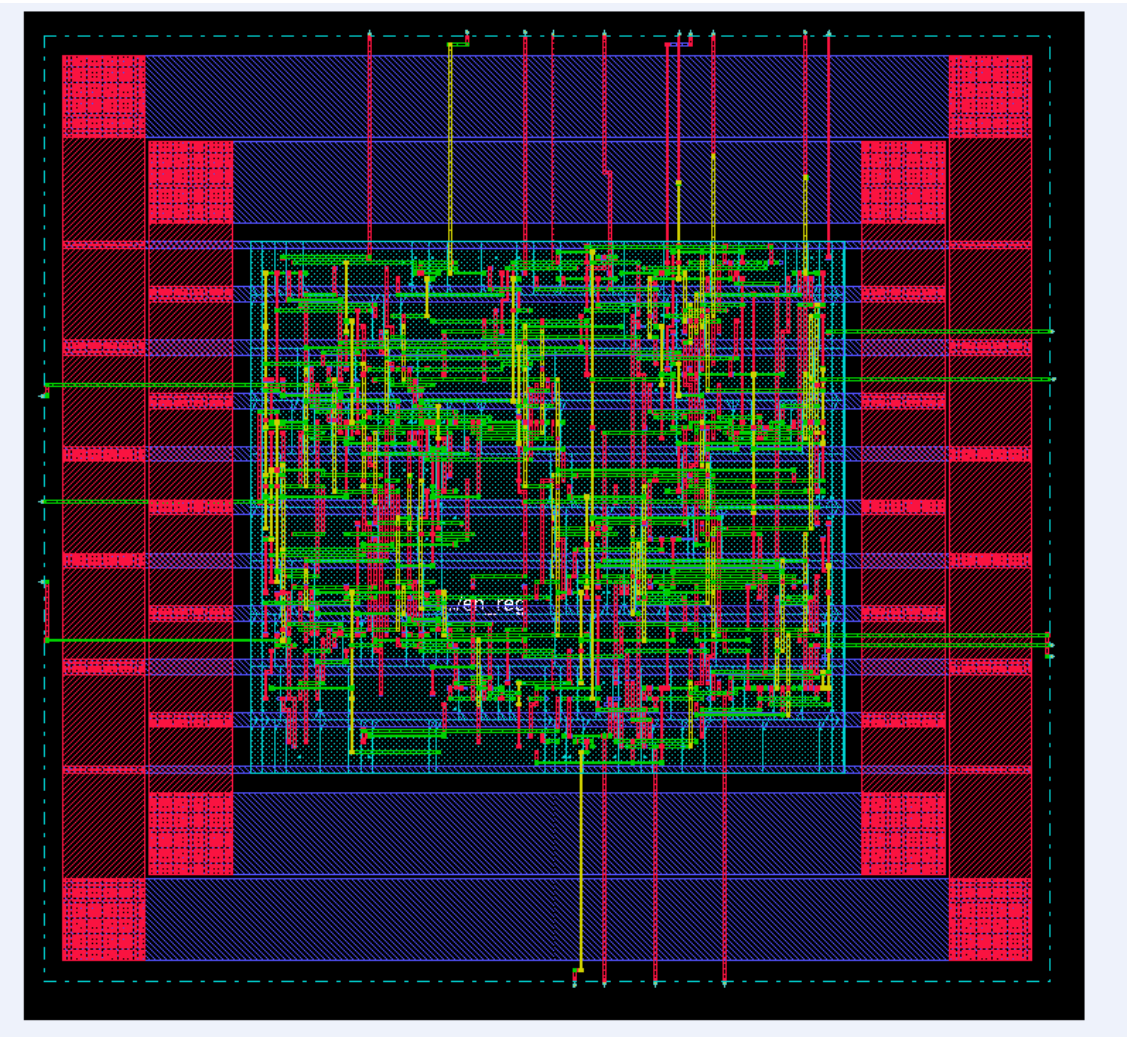

Figure 4-17: View of FSM layout

Pre-layout and post-layout simulations are compared to verify the function of the FSM. The following algorithm is tested to verify the design, with the waveform of simulation results illustrated in Figure 4-18.

Step 1: Reset the FSM block. All the variables are set to initial states in both FSM_COASE and FSM_FINE. The coarse loop starts to work immediately, and the shift register, *mux*, continuously move left or right according to the *updn* signal.

Step 2: Set the *hold* signal to be high so that the coarse loop is frozen. After a while, which is controlled by the FSM_COARSE internal counter, the *en* signal goes to high and enable the fine loop.

Step 3: The fine loop starts to work. The 4-bit variable, *delay_step*, changes from 0101(5) to 1011(-5). Once it reaches 1011, the *delay_step* is reset to 0101. At the same time, the *adr_ctrl* and *trgr_ctrl* signals change from 1 to 0.

Step 4: When the fine tuning continues on until *hold* = 0 again, the FSM_FINE starts the internal timer, so the *delay_step* state is fixed until the end of one timer cycle. Then, the *hold* signal is checked again and the *delay_step* state is adjusted accordingly.

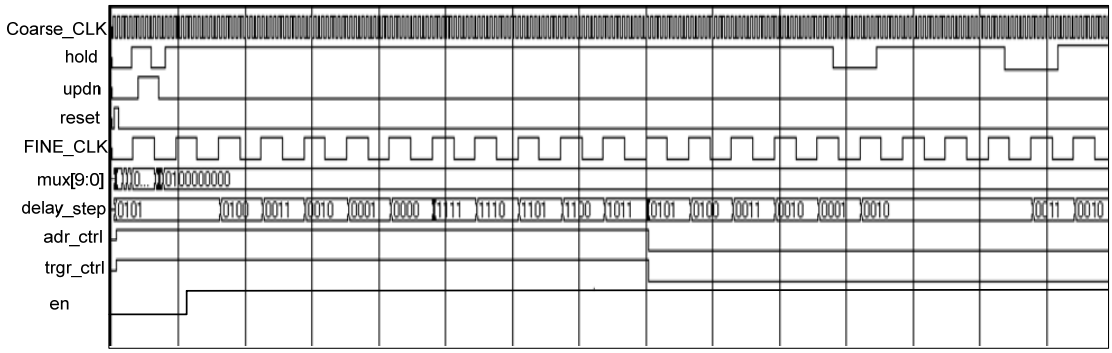

Figure 4-18: Simulation of FSM

4.2.2 Delta-sigma Modulator

From the structure of the delta-sigma modulator, it can be synthesized by instantiating the modules of adders, accumulators, PN generator and quantizer, as shown in Figure 4-19. ADD1 and ADD2 are combinational logic blocks, while ACCUM1, ACCUM2, PN GEN and QUANTIZER are sequential logic blocks. The clock is triggered by the FSM output signal, *trgr_ctrl*, which selects one of the delay line output clocks. The input ports are *mod_in* and *adr_ctrl* from the FSM output, *delay_step* and *adr_ctrl*, respectively. The output port is *mod_out[4:0]*, which connects to the 5-bit fine loop MUX for the feedback clock selection.

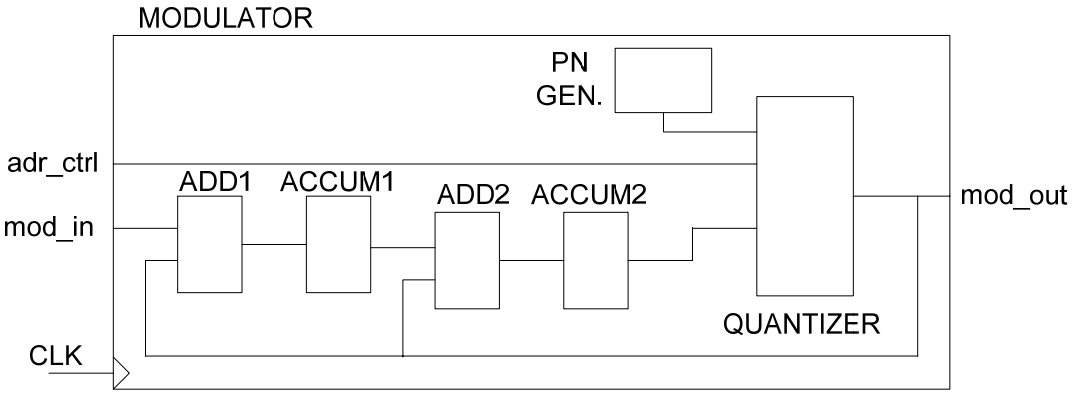

Figure 4-19: Schematic of modulator

Under the top level, it should be pointed out that both ACCUM1 and ACCUM2 are integrators, but their functions are slightly different, according to their discrete-time models in Figure 3-6. The PN sequence is simply generated from the LSB of a 24-bit shift register, which has certain configuration to achieve the maximum sequence length. The input of QUANTIZER takes the effect of dithering and the control signal, *adr_ctrl*.

The RTL synthesis constraints are similar to that for FSM block. Since the clock period is also 5ns, the timing requirement, such as setup and hold time, is set up similarly. There is minor violation in worst input delay (setup time), but it can be eliminated after clock tree generation in the place&route step, which produces the layout as shown in Figure 4-20.

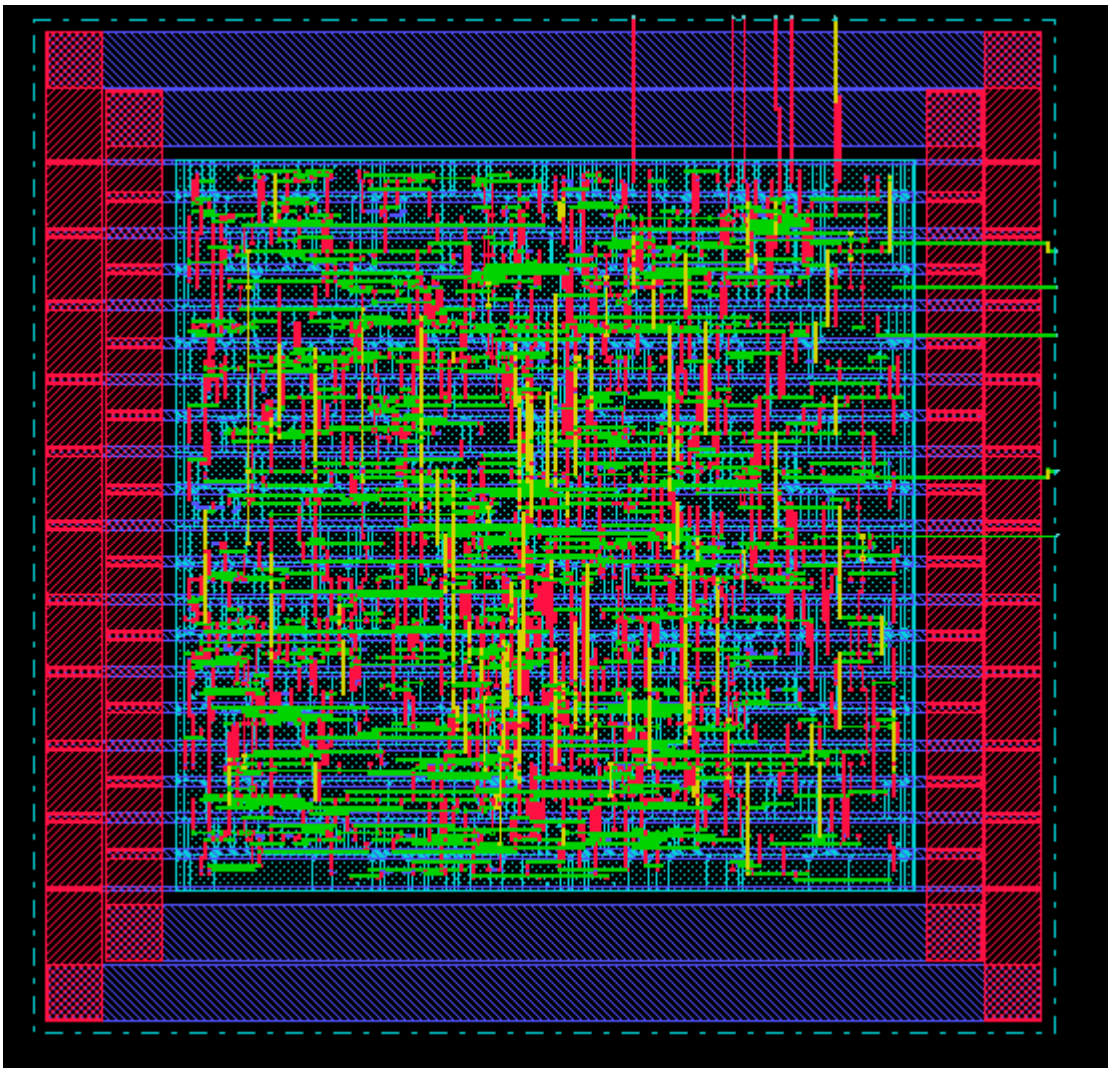

Figure 4-20: View of modulator layout

The testing algorithm and corresponding waveform of simulation results are shown in Figure 4-21.

Step 1. Initial values are set up. The output variable, *mod_out,* is initialized to select the DLL_CLK to be the 10th clock from the delay line. The input data, *mod_in,* is set to 5(0101).

Step 2. After the *reset* signal goes high, the modulator starts to work. The PN sequence is comprised of 0 and 1 in a random order. It can be seen that the *mod_out* state varies randomly to control the DLL_CLK selection.

Step 3. Collect the data array from the *mod_out* sequence and take average. For this example, the mean value of the *mod_out* sequence is 4.93, which is very close to the *mod_in* value.

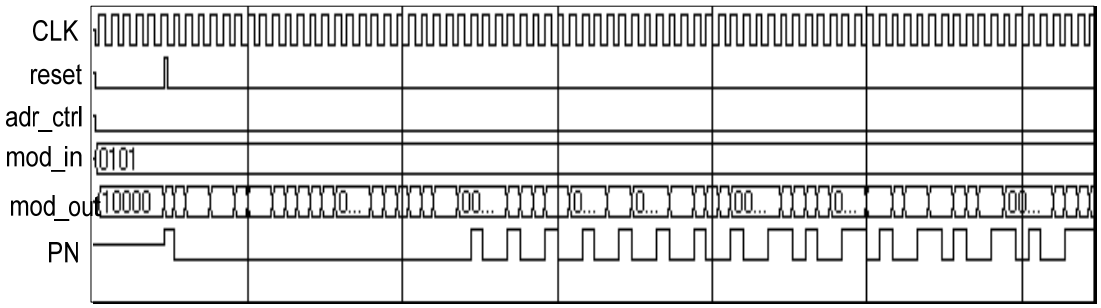

Figure 4-21: Simulation of modulator

Other test cases of different *mod_in* and *adr-ctrl* values are also simulated to verify the function of this delta-sigma modulator.

CHAPTER 5 CONCLUSION

This fractional-N DLL for clock synchronization is proposed in this project. The main benefits are wide operation range, infinite phase shift and low jitter performance. It borrows the idea from fractional-N PLL to reduce noise and jitter. The architecture is based on dual loop DLL: the coarse loop selects the closest clock to the reference clock and the fine loop further aligns these two clocks. But a delta-sigma modulator is applied to the fine loop to randomize the feedback clock. This technique has the feature of noise shaping, so the noise is shifted to higher frequency band, and can be filtered out by the loop filter. Moreover, the operation algorithm avoids extra delay line in the fine loop, which saves power and area compared to previous work.

Behavior model is built to verify the functions. CMOS implementation for analog circuit and digital synthesis has been discussed. For wide operating frequency range, the adaptive loop filter has been proposed. The finite state machine is designed to share the delay line for both coarse phase selection and fine tuning. Table 5-1 gives the performance comparison among the previous work and this proposed DLL.

Table 5-1: Performance comparison

	[6]	[10]	[7]	[23]	[13]	This work
Process	0.5um	0.8um	0.35um	0.35um	0.18um	0.35um
Power supply	3.3V	3.3V	3.3V	3.3V	1.8V	3V
Frequency range	0.0025M-400MHz	0.08M-400MHz	62.5-250MHz	6M-130MHz	60M-760MHz	10M-200MHz
Jitter (pk-pk)	262ps@250MHz	68ps@250MHz	44ps@250MHz	24ps@130MHz	39ps@760MHz	40ps@200MHz

Phase resolution	100ps	40ps	<40ps	X	X	<50ps
Lock time	X	X	X	~1130 cycles	X	~400 cycle
Power	>28.4 mW	102mW	41.6mW	132mW	63mW	~120mW
Area	1.18mm^2	0.8 mm^2	0.2 mm^2	0.45mm^2	0.19mm^2	N.A.

Noted in the table, the maximum operation frequency is 200MHz, which is limited by the digital synthesized blocks. From Chapter 4, the analog blocks work properly up to 400MHz. So, high speed performance could be realized if the digital blocks are customer designed. Besides, the phase resolution will be further improved by increasing the fine tuning steps. It can also be modified to achieve programmable tuning resolution.

The future work is to verify the pre-layout simulation with all frequency range and variation of corners, supply voltage and temperature. Then, this work will be fabricated in 0.35um CMOS technology.

REFERENCES

[1] Behzad Razavi, *Phase-locking in high-performance systems: from devices to architectures*, Wiley-IEEE Press, New Jersey, 2003.

[2] P. M. Gardner, "Charge-pump phase-lock loops", *IEEE Transactions on Communications*, vol. COM-28, pp.1849-1858, Nov. 1980.

[3] B. Miller, B. Conley, "A multiple modulator fractional divider", *Annual IEEE Symposium on Frequency Control*, vol. 44, pp. 578-583, Jun. 1991.

[4] T. A. Riley, M. A. Copeland and T. Kwasniewski, "Delta-sigma modulation in fractional-N frequency synthesis", *IEEE Journal of Solid-State Circuits*, vol. 28, No. 5, pp. 553-559, May. 1993.

[5] Cheng Jia, "A delay-locked loop for multiple clock phase/delays generation", Ph. D Thesis, Georgia Institute of Technology, Dec. 2005

[6] John G. Maneatis, "Low-jitter process-independent DLL and PLL based on self-biased techniques", *IEEE Journal of Solid-State Circuits*, vol. 31, No. 11, pp.1723-1725, Nov. 1996.

[7] Yongsam Moon *et al.*, "An all-analog multiphase delay-locked loop using a replica delay line for wide-range operation and low-jitter performance", *IEEE Journal of Solid-State Circuits*, vol. 35, No. 3, pp.1021-1027, Mar. 2000.

[8] A. Efendovich, Y. Afek, C. Sella and Z. Bikowsky, "Multifrequency zero-jitter delay-locked loop", *IEEE Journal of Solid-State Circuits*, vol. 29, No. 1, pp.67-70, Jan. 1994.

[9] W. Garlepp *et al.*, "A portable digital DLL for high-speed CMOS interface circuits", *IEEE Journal of Solid-State Circuits*, vol. 34, No. 5, pp. 632-635, May. 1999.

[10] Stefanos Sidiropoulos and M. A. Horowitz, “A semidigital dual delay-locked loop”, *IEEE Journal of Solid-State Circuits*, vol. 32, No. 11, pp. 1683-1692, Nov. 1997.

[11] Y. S. Song and J. K. Kang, “A delay locked loop circuit with mixed-mode tuning”, *The First IEEE Asia Pacific Conference on ASICs*, pp. 347-350, Aug. 1999.

[12] A. Efendovich, Y, Afek, *et al.*, “High resolution multi-frequency digital phase locked loop”, *IEEE International Symposium on Circuits and Systems*, pp. 1128-1131, May. 1993.

[13] S. J. Bae, *et al.*, “A VCDL-based 60-760MHz dual-loop DLL with infinite phase-shift capability and adaptive-bandwidth scheme”, *IEEE Journal of Solid-State Circuits*, vol. 40, No. 5, pp.1119-1129, May, 2005.

[14] M. H. Perrott, M. D. Trott and C. G. Sodini, “A modeling approach for Σ-Δ fractional-N frequency synthesizers allowing straightforward noise analysis”, *IEEE Journal of Solid-State Circuits*, vol. 37, No. 8, pp. 1028-1034, Aug. 2002.

[15] C. H. Heng and B. S. Song, “A 1.8-GHz CMOS fractional-N frequency synthesizer with randomized multiphase VCO”, *IEEE Journal of Solid-State Circuits*, vol. 38, No. 6, pp. 848-852, Jun. 2003.

[16] Richard Schreier, *et al.*, *Delta-Sigma Data Converters: Theory, Design, and Simu-lation*, pp.107-110, IEEE Press, 1997.

[17] R. Jocob Baker, *CMOS: Circuit Design, Layout, and Simulation*, pp. 379-382, IEEE Press, 1998.

[18] S. Liu, J. H. Lee and H. W. Tsao, “Low-power clock-deskew buffer for high-speed digital circuits”, *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 554-558, Apr. 1999.

[19] I. A. Young, J. K. Greason, “A PLL clock generator with 5 to 110MHz of lock range for microprocessors”, *IEEE Journal of Solid-State Circuits,* vol. 27, pp. 1599-1607, Nov. 1992.

[20] W. Sansen, *et al.*, “A high-dynamic-range CMOS Op Amp with low-distortion output structure”, *IEEE Journal of Solid-State Circuits,* vol. 22. No. 6, pp. 1683-1687, Dec. 1989.

[21] J. Christiansen, “An integrated high resolution CMOS timing generator based on an array of delay locked loops”, *IEEE Journal of Solid-State Circuits,* vol. 31, No. 7, pp. 952-960, Jul. 1996.

[22] C. H. Kim, *et al.*, “A 64-Mbit, 640-MByte/s bidirectional data strobed, double-data-rate SDRAM with a 40-mW DLL for a 256-MByte Memory System”, *IEEE Journal of Solid-State Circuits,* vol. 33, No. 11, pp. 1703-1710, Nov. 1998.

[23] H. H. Chang, *et al.*, “ A Wide-Range Delay-Locked Loop With a Fixed Latency of One Clock Cycle”, *IEEE Journal of Solid-State Circuits,* vol. 37, No. 8, pp. 1021-1028, Aug. 2002