MSc Thesis

ADPLL Design for WiMAX

Wenlong Jiang

September 18, 2011

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

Delft University of Technology

Printed in The Netherlands

DELFT UNIVERSITY OF TECHNOLOGY
FACULTY OF
ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"ADPLL Design for WiMAX"** by **Wenlong Jiang** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: *September 18, 2011*

Supervisor:

Dr. R. Bogdan Staszewski

Readers:

Dr. Ir. Wouter Serdijn

Dr. Ir. Nick van der Meijs

Ir. Frank Verwaal

Ir. Marcel van de Gevel

Dr. Xuefei Bai

Mr. Sejed Amir Reza Ahmadi Mehr

Abstract

The frequency synthesizer, which functions as a local oscillator, is a critical block in the transceiver. It needs to meet very stringent specifications and consume as less power as possible. Design of a traditional charge-pump PLL as the frequency synthesizer in the advanced CMOS technologies in the transceiver of advanced communication systems proves to be not an easy work and is becoming difficult due to the supply shrink. The ADPLL system, which defines every essential block with digital interface, proves to be an excellent alternative.

This thesis deals with the system level design of ADPLL for the WiMAX standard. The architecture of the ADPLL is presented, with the functional illustration for every building block, like DCO and TDC. The ADPLL system is modeled and described in Cadence using Verilog-AMS/Verilog. The performance of the system is analyzed in s-domain. Some advanced algorithms have been applied to the ADPLL system. The spur mechanism in the near-integer N cases is proposed and verified. The phase rotation algorithm and the FREF dithering algorithm have been adopted to effectively suppress these spurs. The top level issues of ADPLL are tackled, with emphasis on the test plan and the operation modes of the system. The behavior level simulation results of the system are presented and the performance summary is given.

The transistor level design of a basic DPA is presented. The layout for the important blocks is done and the practical concerns of the DPA design are discussed. The post-extraction simulation results are shown.

Acknowledgement

I am deeply grateful to all the people who in one way or another have helped me during my MSc project. Without the support of others, it would be impossible for me to reach this stage.

First and foremost I would like to express my sincere thanks to my MSc supervisor, Dr. Robert Bogdan Staszewski. It is really an honor to work under his supervision on the field of the ADPLL design. Thanks for his guidance in my MSc project work and other matters. I have learned a lot from his incomparable expertise on this field, his strict requirement on the design and his passion for the work. Once again, thank you for your patience.

Special thanks to my colleagues in Catena. I want to thank Ir. Frank Verwaal for the discussions on the system design and for his life wisdom. I also want to thank Ir. Marcel van de Gevel for the time-to-time help and proofreading of my thesis. The thanks also go to Koen van Hartingsveldt, Gerard Lassche, Frans Sessink, Floris van der Wilt, Iqbal Suhaib, Ernst Habekotte, Federico Bruccoleri, Bert Oude Essink, Aylin Donmez, Tom Fric and other members in the design team for the discussion on the circuit design and the chip creatioin; to Jerry Lit and Daniel Mitcan for the discussion on the digital design flow; to Nicole Walford and Ivaylo Bakalski for the help on the layout; to Atze van der Goot, Marcel van de Wiel, Frank van den Hout and other people in the CAD and IT support group; to Cynthia Thepass and Helma Timmermans-Piersma for the help on a lot of issues. I am especially grateful to Krass Maklev, Kave Kianush and Rien Geurtsen of Catena Microelectronics B.V for their generous support and great assistance during my MSc project.

I would like to express my gratitude to Popong Effendrik, Armin Tavakol and Xuefei Bai, who work with me in this design. We have spent wonderful time together for coffee, technical discussions and cultural discussions. We have a lot of fun and I have learned a lot from you. These memories and the friendship will be invaluable.

I am very thankful to the other members of my MSc defense committee for their insightful

questions and invaluable time, Dr. Wouter Serdjin, Dr. Nick van der Meijs and Mr. Sejed Amir Reza Ahmadi Mehr. Moreover, I want to thank Prof. John Long for his support as the chair of ELCA.

I would like to thank my friends here. These go to the PhD students in ELCA, especially Morteza Alavi, Wanghua Wu and Duan Zhao for the technical discussions and the help on other issues; go to Jianfeng Wu for the wonderful cooperation on the courses; and also go to Fan Guo, Jing Li, Xianli Ren, Ao Ba, Kezheng Ma, Xiaoqiang Zhang, Junfeng Jiang, Guanyu Yi, Jia Guo, Ting Zhou, Ting Yan, Zeng Zeng, etc. for the help and the fun during the two-year study on Microelectronics in TU Delft.

Lastly, I would like to thank my parents for all that they have done for me. Only with their support can I take the courage to study and succeed across globe. I owe my accomplishments to them.

Table of Contents

List of Figures

List of Tables

Acronyms

AB	Acquisition bank
ACW	Amplitude control word
ADC	Analog-to-digital converter
ADPLL	All-digital PLL
BB	Baseband
BIST	Built-in self test
BT	Bluetooth
CCW	Capacitance control word
DAC	Digital-to-analog converter
DCO	Digitally controlled oscillator
DE	Drain efficiency
DPA	Digitally-controlled power amplifier
FS	Frequency synthesizer
IP	Intellectual Property
LNA	Low noise amplifier
LO	Local oscillator
LPF	Low pass filter
MMD	Multi-modulus divider
NoB	Number of bits
NTW	Normalized tuning word
OFDM	Orthogonal Frequency Division Multiplexing

OTW	Oscillator tuning word
PA	Power amplifier
PA Driver	Power amplifier driver
Power added efficiency	PAE
PB	PVT bank
PFD	Phase/Frequency detector
PGA	Programmable gain amplifier
PLL	Phase Locked Loop
PVT	Process, voltage and temperature
RX	Receiver
SDR	Software-defined radio
SNR	Signal-to-noise ratio
SoC	System-On-Chip
TB	Tracking bank
TX	Transmitter
TDC	Time-to-digital converter
VCO	Voltage-controlled oscillator
ZPR	Zero-phase restart

Chapter 1

Introduction

1-1 Motivation

The continuous scaling down of CMOS technology has provided us far more superior computation power than ever before. With the technology node advancing from 90 nm to 65 nm to 45 nm (and now, we have microprocessor with 28 nm CMOS technology!), we can have one mobile terminal that possesses more functionalities than old-dated desktop computer even. This has greatly changed people's living and working style, leading to an era of smart phone, social networking, smart sensor, mobile business, etc. The explosive growth of applications and services call for telecommunication systems, i.e. transceivers with higher data throughput and safer data transmission. In high performance transceiver design, RF and Analog front end shows as the bottleneck due to that:

1. The voltage headroom for advanced CMOS technologies is limited because of reliability issues with thin oxide core transistors. The supply voltage decreases with technology advancement. This results in a drop of signal-to-noise ratio (SNR) and dynamic range in the voltage domain, which makes the design of the RF/Analog front end difficult.

2. In addition to what are provided in the standard digital process, RF and Analog circuits usually require some extra devices. Such requirements raise the cost of the product and prolong the development cycle as well.

3. The RF/Analog front end design takes much more effort than digital back end for migration to new technology. Designers need to tackle various issues like reduced supply voltage, degraded matching and lower intrinsic gain. Comparatively, the migration of digital logic is pretty easy as long as the design flow is established.

Therefore it's indispensable to borrow the power of Digital for Analog and RF circuits. The idea of digital assistance will bring benefits like easier calibration, built-in test, or even replacement of bulky traditional blocks with compact and flexible digital blocks.

One of the most challenging design tasks in mobile RF systems is the *frequency synthesizer* (FS), which is deployed as *local oscillator* (LO) both in the transmitter path and the receiver path. It needs to meet a set of very stringent specifications while still be low-area and low-power. The *Phase Locked Loop* (PLL) is the common architecture of frequency synthesizer for the high performance, low power wireless transceiver. The charge-pump PLL, as the most popular traditional technique for PLL, is analog intensive. It eats up significant area and power and is not so scalable to new technology (need re-design). In contrast, the *All-Digital PLL* (ADPLL) technology, which has been successfully applied to Bluetooth (BT) and GSM[1][3], has shown itself as a very potential candidate for the implementation of frequency synthesizer in more advanced communication standards, as:

1. *Integrable with digital process*: ADPLL technology has minimum analog and RF circuit content and doesn't need special devices for RF/Analog application, which reduces the cost of the chip. Moreover, it makes maximum use of the digital computation power to improve the performance of the system.

2. *Easy maintenance*: ADPLL has all the essential blocks with digital interface. Thus the Intellectual Property (IP) of ADPLL is easy to maintain while being migrated to latter technology nodes (or even not a CMOS technology, as long as the boolean algebra still holds).

3. *New modeling and simulation methodology*: ADPLL can have its top level modeled in an event-driven simulator or a time-driven simulator, like Matlab, VHDL and Verilog-AMS. This simulation methodology accelerates the simulation and can help identify the bottleneck of the system performance.

4. *Smaller area*: Since ADPLL has replaced the bulky analog blocks with digital logic, the area can be significantly reduced.

5. *Flexibility*: Since the control of ADPLL is fully digital, the parameters of ADPLL, like loop coefficients, target frequency and resolution can be easily modified according to our need. Complex algorithms are applicable to ADPLL even after the chip is taped out. The design cycle is greatly shortened, which stimulates the innovations, especially at the system level.

Thus there is a huge interest in the design of ADPLL for modern communication systems. However, nothing comes for free. To implement an ADPLL which fits into tight specifications, a comprehensive effort and innovation from various aspects like algorithm, architecture, circuit design and test has to be made. This thesis is dedicated to the design of ADPLL for WiMAX application, especially on the exploration of system level solutions.

In addition to that, the transistor level design of a *digitally-controlled power amplifier* (DPA), which is used for output buffer of ADPLL, will be presented. The design of this DPA is also a preparation for the ADPLL-based transmitter or transceiver design as an extension to our project.

1-2 Introduction to Frequency Synthesizer

1-2-1 Application of Frequency Synthesizer in Wireless Systems

The frequency synthesizer is an essential part in RF transceiver. As in both a *transmitter* (TX) and a *receiver* (RX), it is deployed as LO to perform frequency translation between *baseband* (BB) and RF.

Figure 1-1 shows a simplified direct-conversion transmitter.

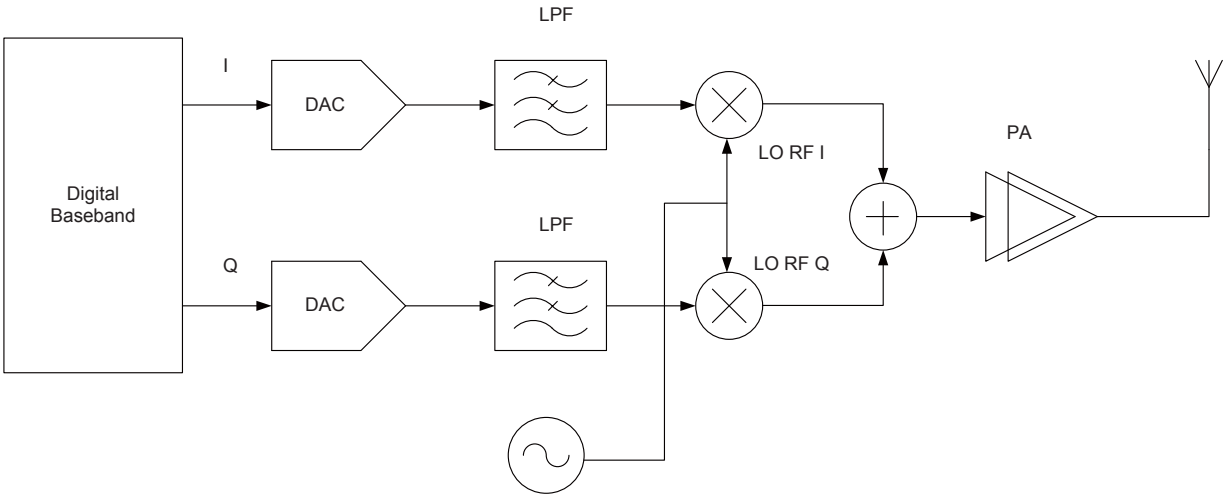

Figure 1-1: Architecture of direct conversion transmitter.

In a direct-conversion transmitter, the *in-phase* (I) and *quadrature-phase* (Q) pulse-shaped digital baseband signals are converted into analog signals via the digital-to-analog converter (DAC). Then the low pass filter (LPF) can filter out the alias in frequency domain due to sampling in DACs [1]. After that, the baseband analog signals are up-converted to RF frequency by a single-sideband modulator, which is usually implemented as a quadrature mixer as shown in Figure 1-1. A *power amplifier* (PA) acts as the last stage in the transmitter path to provide enough output power to the antenna.

Figure 1-2 shows a simplified direct-conversion receiver[2].

[1] Also out-of-channel noise is filtered. Usually that's not a big issue in transmitter.

[2] Direct-conversion receiver is often called zero-IF receiver, which means its *intermediate frequency* (IF) is ideally DC.

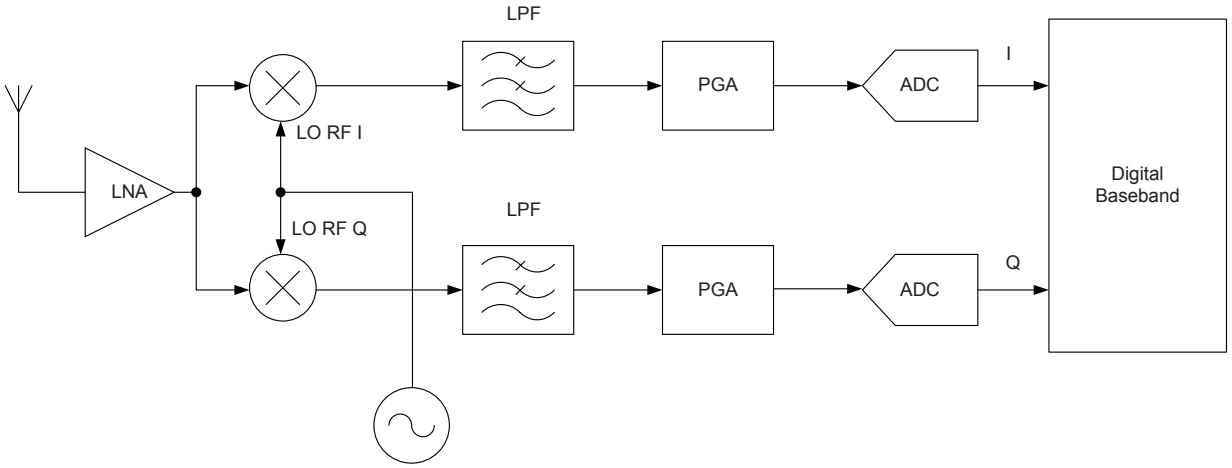

Figure 1-2: Architecture of direct conversion receiver.

The signal received from the antenna will first be amplified by a *low noise amplifier* (LNA)[3]. Then it's down-converted to baseband via a quadrature mixer. The following LPFs will filter unwanted frequency components (both interference and noise) and the programmable gain amplifier (PGA) can bring signal to the required level for the analog-to-digital converter (ADC). After being converted to the digital domain, the I and Q signals are fed into the digital baseband for further processing.

1-2-2 Common Metrics for Frequency Synthesizer

As a critical part of the RF transceiver, the frequency synthesizer has to meet specifications which vary for different applications. Yet there are some common metrics that are shared in common and are considered in our project. A tour of these metrics is the key to understanding the design, simulation and test presented in this thesis.

Since the frequency synthesizer is used to translate the signal frequency from BB to RF or from RF to BB, the frequency accuracy of its output is very important. When the frequency of the LO deviates from the desired value, for RX, after quadrature mixing, some unwanted frequency components will not be attenuated by LPF and some useful signal will be filtered away, which leads to degradation of SNR of the whole system. For advanced communication systems like WiMAX and WiFi, *Orthogonal Frequency Division Multiplexing* (OFDM) technology has been widely deployed. Then this frequency error will undermine the orthogonality of subcarriers, if the digital baseband processor cannot correct for this. Still the detail of this is more of an issue for telecommunication standards. In this project we get the specification of tolerated frequency error from customers.

[3]Sometimes we have a band-select filter in front of LNA. That depends on the tradeoff of noise and linearity, and always cost.

In older telecommunication standard like GSM and BT, the frequency range for the whole band is small. While for more recent communication standards, the range can be pretty wide. Some standards even have multi bands. Thus it's essential to have a good frequency planning at the frequency synthesizer level or the transceiver level to make sure LO can cover the whole frequency range.

As can be seen in Figure 1-1 and Figure 1-2, LO needs to provide the I/Q signal for quadrature mixer. The amplitude and phase mismatch of the I/Q signal would degrade the image rejection of quadrature mixer. Say $\Delta\theta$ and ΔG are respectively phase and amplitude mismatch of the I/Q signal, as follows:

$$I = A(1 + \Delta G)\cos(\omega_{\mathrm{LO}}t + \Delta\theta) \tag{1-1}$$

$$Q = A\sin(\omega_{\mathrm{LO}}t) \tag{1-2}$$

If we assume an RF signal above the LO which is $\cos((\omega_{\mathrm{LO}} + \omega_{\mathrm{IF}})t)$ and an image signal below the LO which is $\cos((\omega_{\mathrm{LO}} - \omega_{\mathrm{IF}})t)$, due to the mismatch of the I/Q signal, the image signal will generate some undesired output at the frequency of interest ω_{IF}. The magnitude of the image rejection, as the power ratio of the desired output to the undesired output, can be approximated as[4]

$$IRR \approx \frac{4}{\Delta\theta^2 + \Delta G^2} \tag{1-3}$$

Yet note that not all transceiver will need both I and Q signals. As in the traditional polar transmitter which is used for constant-envelope modulation, maybe just one-phase signal (single-ended or differential) is needed. However, for some transmitter architectures of advanced modulation, maybe phases more than just I and Q are needed.

The above metrics are mostly static, which means they are measured when the frequency synthesizer has settled to a certain frequency. In some modern communication systems, the frequency synthesizer is required to settle to another frequency in very short time. In that case the settling time and the settling dynamic of the frequency synthesizer are also very important.

The more complex specification for frequency synthesizer is phase noise. A detailed explanation for it is given below.

1-2-3 Explanation for Phase Noise

The output of a generic oscillator $v_o(t)$ with a sinusoidal wave shape and a nominal oscillation frequency f_o hertz is[5]:

$$v_o(t) = [A + a(t)]\cos[2\pi f_o(t) + \phi(t)] \tag{1-4}$$

Here A is the mean amplitude of the oscillator output, $a(t)$ is the zero-mean amplitude noise, and $\phi(t)$ contains all phase and frequency departures from the nominal oscillation

frequency f_o and phase $2\pi f_o t$. Phase disturbance $\phi(t)$ (in radians) includes the zero-mean phase noise, the initial phase, and the integrated effects of frequency offset and frequency drift.

For generic oscillators that contain an amplitude-control mechanism, amplitude fluctuations are greatly suppressed. Besides that, the signal is often converted into a square wave somewhere in the system, which also clips off amplitude noise. Thus the effects of phase noise far overshadow the effects of amplitude noise. This applies to our ADPLL design in which a DCO is followed by dividers, as can be seen in Chapter 2. We will ignore the effect of amplitude noise for the discussion below.

Since $v_o(t)$ as in Equation 1-4 is a random signal, we shall consider the Fourier transform of its autocorrelation function, which is the double-sided spectrum of $v_o(t)$. We convert this to the single-sided spectrum $W_{vo}(f)$, by multiplication of two. Ideally, in the absence of phase noise, the single-sided spectrum $W_{vo}(f)$ would be a single line located at $f = f_o$. Due to the phase noise, the spectrum would spread into the vicinity of f_o. The more noisy the signal is, the greater the spreading is, as can be seen in Figure 1-3. However, the total power of the signal, which equals the integral of $W_{vo}(f)$ over all frequencies $f = 0$ to ∞, is $A^2/2$ volts^2.

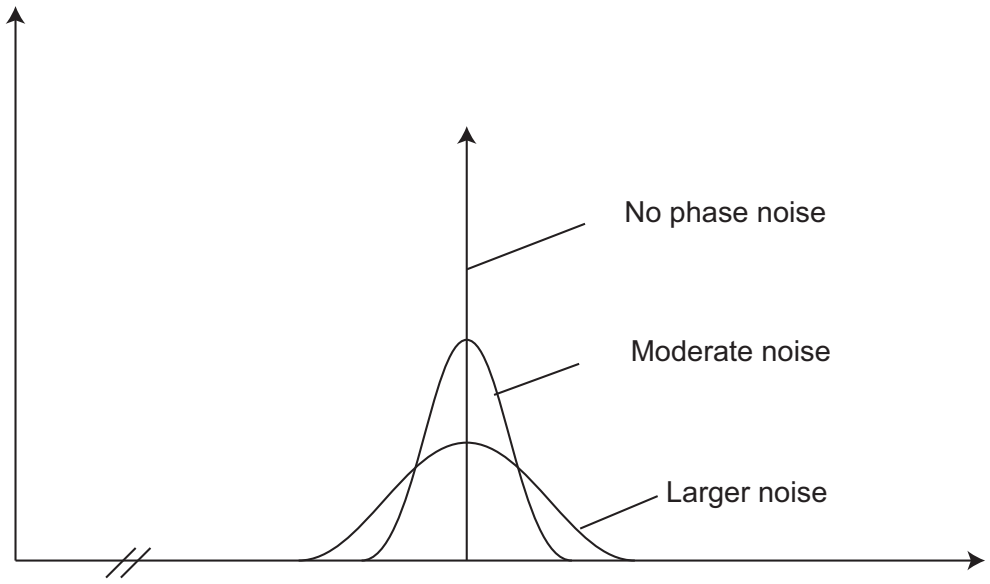

Figure 1-3: Theoretical spectrum $W_{vo}(f)$ of oscillator output $v_o(t)$.

The normalized version of $W_{vo}(f)$, $\mathcal{L}(\Delta f)$, is defined as

$$\mathcal{L}(\Delta f) = \frac{W_{vo}(f_o + \Delta f)}{A^2/2} \tag{1-5}$$

which means the noise power within a bandwidth of 1 Hz in a single sideband at a frequency offset of Δf from the center frequency f_o, relative to the total power. Usually we express $\mathcal{L}(\Delta f)$ as 10 $\log[\mathcal{L}(\Delta f)]$ dBc/Hz.

$W_{vo}(f)$ and thus $\mathcal{L}(\Delta f)$ can be measured via a RF spectrum analyzer. Figure 1-4 gives a simplified block diagram of one kind of spectrum analyzer. The signal with the frequency f_o is mixed with the signal of a swept local oscillator that has a frequency of f_{LO} . The frequency difference $f_o - f_{\rm LO}$ is applied to a bandpass filter that has the center frequency f_{IF} and the resolution bandwidth RBW. The output of this bandpass filter is fed into a square-law detector and then a lowpass smoothing filter with video bandwidth VBW. This smoothing-filter output either goes directly to a display that shows power or goes through a logarithmic converter to a display that shows power on a dB scale, which is $P_{\rm RF}(f)$. After scaled to RBW and normalized to the integral of the power spectrum, we can estimate $\mathcal{L}(\Delta f)$ from $P_{\rm RF}(f)$.

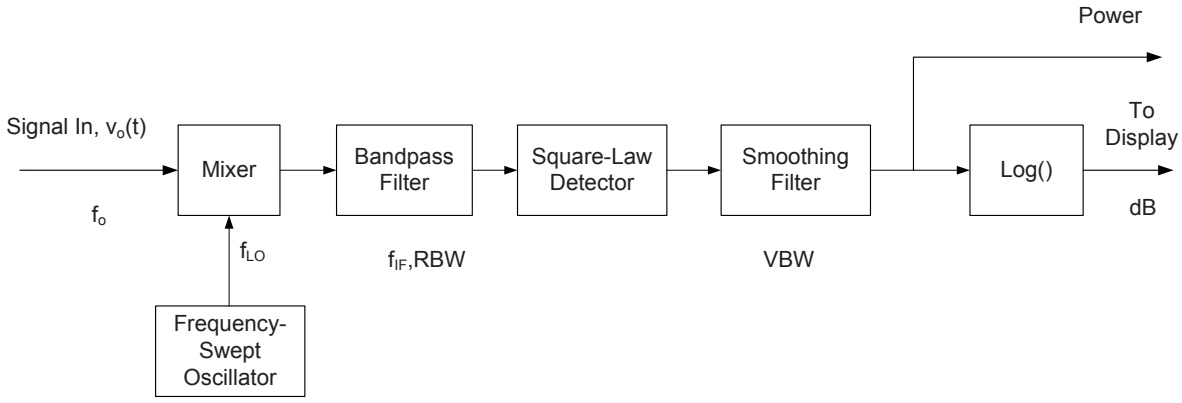

Figure 1-4: Simplified block diagram of a spectrum analyzer.

The problem with the RF spectrum analyzer is that it needs to handle the total power of signal while still can detect the weak sidebands due to spurs or phase noise. Such difficulty has led to the widespread use of $W_\phi(f)$, i.e. the low-pass, single-sided spectrum of the phase-noise modulation $\phi(t)$. A conceptual block diagram for the measurement of $W_\phi(f)$ is shown in Figure 1-5. The measurement instrument consists of a phase demodulator which reproduces a magnitude-scaled version of $\phi(t)$, a low-frequency spectrum analyzer to produce $W_\phi(f)$, and a logarithmic converter for the display purpose.

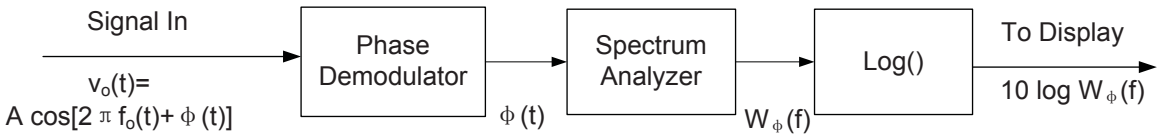

Figure 1-5: Block diagram of a generic phase-noise analyzer.

If the phase-noise amplitude is small enough, it can be shown that $\mathcal{L}(\Delta f) \approx W_\phi(f)/2$, so we can obtain $W_\phi(f)$ using a phase-noise analyzer and minus 3 dB to get the corresponding $\mathcal{L}(\Delta f)$. Since our specifications for phase noise are lower than -90 dBc/Hz, as can be seen in Table 1-1, this approximation works very well. In the system simulation shown in Chapter 5, $\mathcal{L}(\Delta f)$ is derived just in this way. We will touch that later.

1-2-4 The Impairment of Phase Noise in Frequency Synthesizer

The impairment due to phase noise of LO in a receiver is usually illustrated by the phenomenon called *reciprocal mixing*. This happens when a receiver receives two signals at its antenna, one is the small desired signal and one is an undesired large interference featuring a frequency close to that of desired signal. If the LO has a significant amount of phase noise in it, when two signals are mixed at the mixer, the noise of LO from frequency synthesizer is superimposed on both of the down-converted signals, as seen in Figure 1-6 [4]. Then at the output of the mixer, the small desired signal is corrupted by the LO's noise which has been down-converted by the large interference. This is named reciprocal mixing because the RF port of the mixer now acts like an LO port (the large interference becomes the LO signal) and the LO port has become the RF port (the so called 'RF signal' is actually the noisy LO signal).

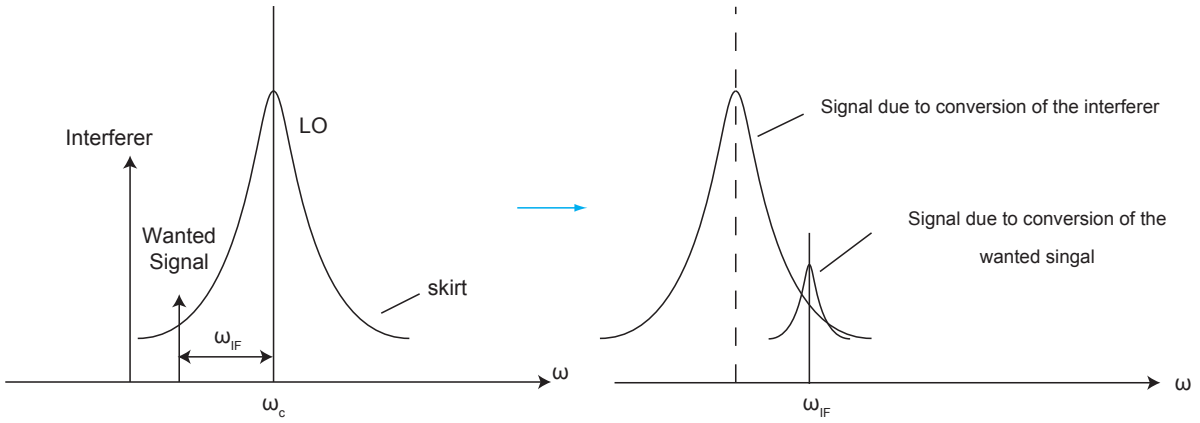

Figure 1-6: Effect of LO phase noise in a receiver (reciprocal mixing).

The real situation is more complicated as the specifications for most standards would include both single-tone interference and the modulated interfering signals. To have a good estimation of the requirement for LO's phase noise, comprehensive measures of theoretical calculation, simulation and field test are often taken.

In a transmitter, the phase noise of LO will spread the ideal spectrum of the output signal, causing spurious emissions. For a receiver that wants to detect a small desired signal from some transmitter, the tail of a large signal in nearby channel from another transmitter would significantly degrade SNR of the received signal[5], as shown in Fig 1-7. Thus communication standards usually specify stringent mask for spectrum of transmitter output.

In addition to the simple analysis above, for modern wireless systems that use OFDM, the bandwidth of one channel can be in the order of several MHz or even larger than 10

[4] It may seem strange to put the spectrums of deterministic signal and stochastic signal in one picture.
[5] Be aware that this is different from reciprocal mixing effect in the receiver.

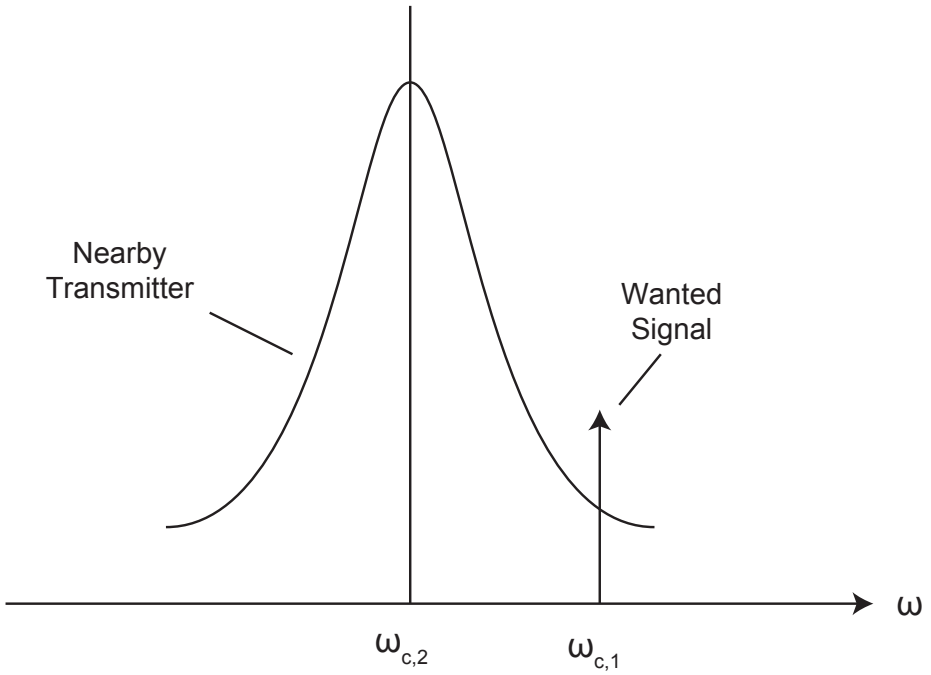

Figure 1-7: Effect of LO phase noise in a transmitter.

MHz (and sometimes, the bandwidth of the channel is variable, which strategy WiMAX has adopted). The channel is divided into several orthogonal subcarriers with bandwidth on the order of 100 kHz. In that case, the close-in phase noise would spread out every subcarrier, cause inter-subcarrier interference and jeopardize the orthogonality of subcarriers. The close-in spot noise of the frequency synthesizer at a certain frequency offset, which means $\mathcal{L}(\Delta f)$, will indicate the degradation resulted from the adjacent subcarriers. The integrated single-sided phase noise, $\int \mathcal{L}(f)df$, instructs the interference to any subcarrier introduced by the interaction of all other subcarriers and the LO phase noise. Here the boundaries of the integration are defined according to the bandwidth of the communication standard.

1-3 Introduction to ADPLL

1-3-1 Fraction-N Charge-pump PLL and Related Issues

Traditionally a great majority of frequency synthesizers for wireless applications are based on the charge-pump PLL topology. As shown in Fig. 1-8, the output clock of the *voltage-controlled oscillator* (VCO) is divided by N. The divided clock FDIV is compared with reference clock FREF. The phase error (actually, the time difference) of the edge of the two clocks will be detected by the *Phase/Frequency Detector* (PFD) and it will generate either an UP or a DOWN pulse proportional to the detected time difference. That UP or DOWN pulse will control the on/off of current source I_P and I_N. In the loop filter, this

current flow will be converted to a VCO tuning voltage which will control the frequency of FVCO and thus closing the loop. The frequency of FVCO shall be N times the frequency of FREF when the loop is stable and settled.

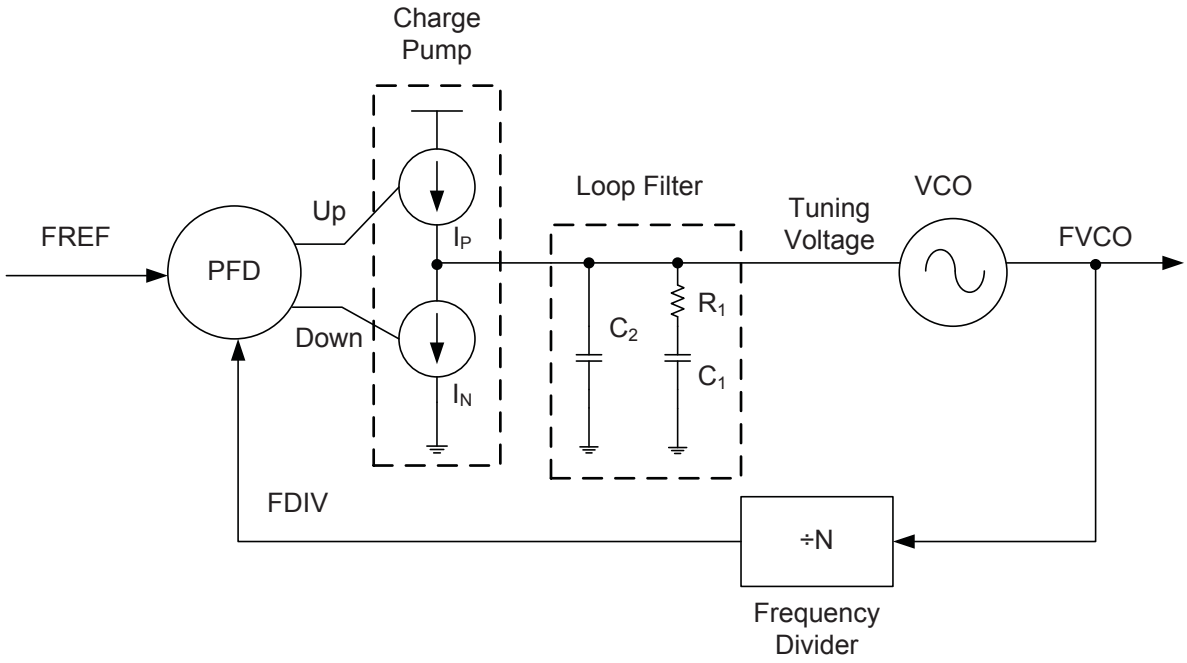

Figure 1-8: Simplified charge-pump PLL topology.

As we have mentioned above, a fine frequency resolution is desired in modern wireless systems. If N is just integer, which means the frequency resolution is FREF, a.k.a. Integer-N PLL, then the reference frequency is on the order of hundreds of Hz or even tens of Hz. However, the bandwidth of PLL is usually no more than one tenth of FREF for stability concern. In that case the settling of PLL is pretty slow and cannot correct for the drift of VCO in time. Besides, a very narrow bandwidth will result in a loop filter that takes too much area.

The Fractional-N PLL can alleviate the tradeoff between frequency resolution and bandwidth. The division ratio of frequency divider N can be fractional thus the resolution will be determined by the effective bits of fractional part of N. The most common method to generate a fractional division ratio is to use a *multi-modulus divider* (MMD), i.e. the division ratio of the divider would toggle between two integer values and the average effect results in a fractional value. This toggling behavior can incur spur or noise within the bandwidth. To remove them out of bandwidth, usually a *Sigma-Delta Modulator* (SDM) is to generate the command word for MMD.

The Fraction-N charge-pump PLL architecture, as shown in Fig. 1-9, has issues being integrated into current deep-submicron CMOS technology, such as the leakage of loop filter, distortion due to non-idealities of charge-pump, special mask needed from performance concern. There are some challenges brought up when used in modern wireless systems[6]:

1. Wideband wireless systems demand larger tuning range, which may results in a higher gain or a larger swing in VCO control. The high gain of VCO tuning voltage, a.k.a. K_{VCO}, will make PLL noise-sensitive. Thus PLL generates higher spur and noise due to the hostile environment in the System-On-Chip (SoC). Or a larger swing will incur non-linearity of VCO and needs more effort for calibration.

2. There is a trend to choose the polar transmitter architecture, which encompasses PLL and the power amplifier driver (PA driver) as the essential blocks, for complex communication system. To add a modulation to PLL, measures like either pre-distortion or two-point modulation are needed. However, these measures are all subject to the non-idealities in the analog domain, like variation with respect to *process, voltage and temperature* (PVT). The accuracy is limited. The performance degrades due to these non-idealities.

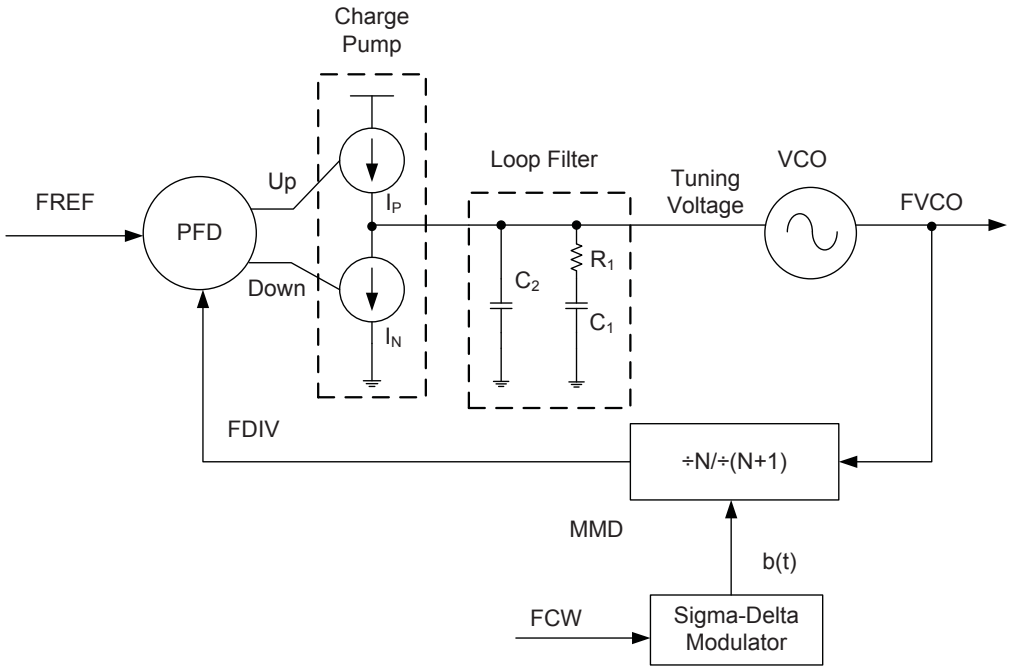

Figure 1-9: Fractional-N charge-pump PLL topology.

1-3-2 The Simplified ADPLL Schematic

ADPLL is a very promising candidate to replace charge-pump PLL in advanced CMOS process and modern wireless applications. Fig 1-10 is the simplified schematic of an ADPLL as in [1]. The DCO, an oscillator whose frequency is controlled by the digital tuning word rather than the tuning voltage, lies in the heart of the ADPLL. FCW, which is the abbreviation of *Frequency Command Word*, is the ratio of the desired frequency f_V

divided by the reference frequency f_R. Since this FCW is digital signal, the cost of finer frequency resolution is just more *number of bits* (NoB) for the fractional part of FCW. Ideally one FREF cycle shall contain FCW cycles of the desired frequency. Thus the reference phase accumulator in the figure gives the phase of the desired signal divided by 2π. The oscillator phase accumulator, which is actually a counter clocked at DCO positive edges, indicates the phase of DCO output in a resolution of 1 cycle (2π). The *time-to-digital converter* (TDC) can compensate for the fractional error of the oscillator phase accumulator. Then the phase detector, which is merely an arithmetic subtractor, produces faithfully the phase error PHE[k] between the desired signal and the feed-back signal from DCO. PHE[k] is fed to the digital loop filter and the loop filter generates the *normalized tuning word* (NTW). The *oscillator tuning word* (OTW) is obtained by multiplying NTW with the normalization ratio of f_R divided by $\hat{K}_{DCO}$ (an estimation of DCO gain, which is defined as the frequency change due to change of one LSB in the DCO tuning word). Therefore, we have a closed-loop ADPLL with full digital signals at the top level.

It shall be noted that for digital logic operations, a synchronous clock is needed. However, the output clock of DCO (CKV) is asynchronous with the reference clock (FREF). A retimer, which is drawn as a flip flop, is used to generate the clock in the digital domain (CKR). The idea is basically to oversample FREF with the high-frequency clock CKV. Therefore the edge of CKR is aligned with the edge of CKV. Also the digital logic is active only when TDC has finished the activity and been quiet.

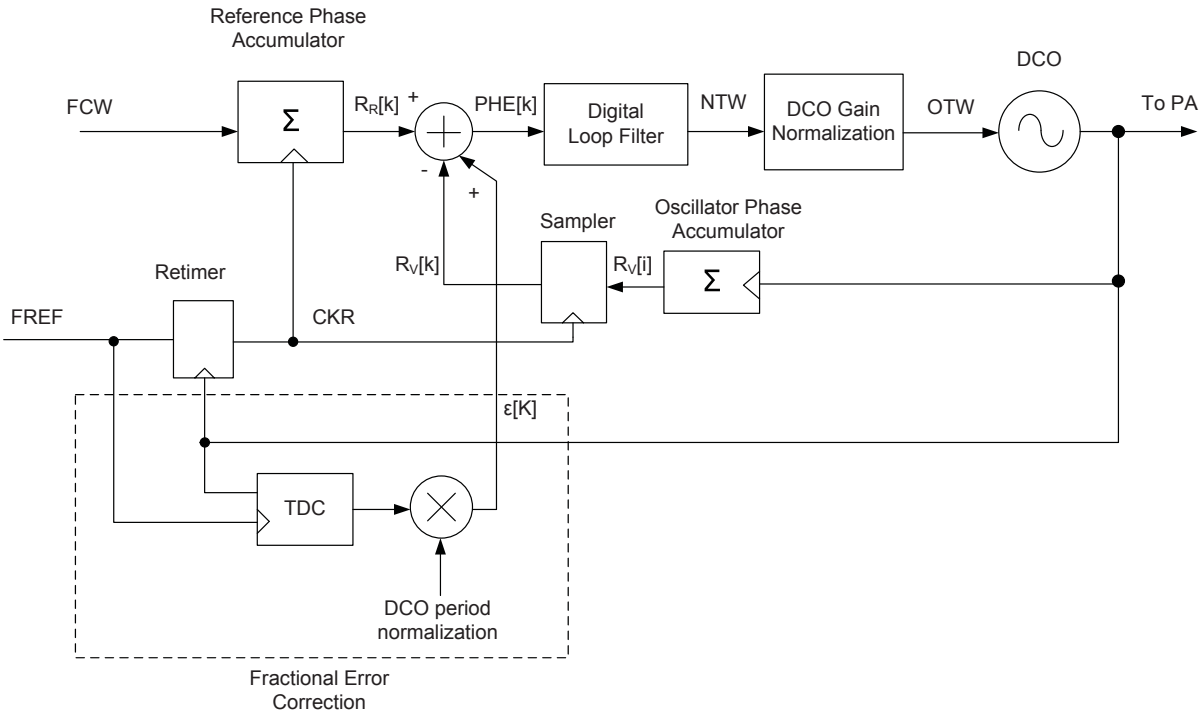

Figure 1-10: Simplified schematic of ADPLL in [1](modulation part is not drawn).

Through these efforts, all blocks in ADPLL can be seen as ASIC cells. Say for DCO the input is the digital tuning word(s) and the only information we care about in the DCO output is the timing of edge. All issues with analog-intensive blocks in the charge-pump PLL are avoided. The digital logic is highly immune to noise and if well calibrated, will not be hurt by the DCO non-linearity. The bandwidth of ADPLL is well defined compared to analog peers. What's more, the digital signals are easier to store and maintain than analog signals. For all these reasons, the ADPLL technology has advanced to so-called duty-cycled PLL [7] and RF *built-in self test* (BIST) [8], and even further to transmitter synthesized from standard cell library [9] and is expected to be the corner stone for *software-defined radio* (SDR) [10].

There are now two basic architectures for ADPLL. The first one resembles the traditional fractional-N charge pump PLL. A programmable frequency divider is used in the feedback path and the TDC simply replaces the combination of the phase detector and the charge pump. The phase difference is fed into the digital loop filter instead of the analog loop filter. Thus it's called digital $\Sigma\Delta$ fractional-N PLL. The second one, which is a simplified version of Fig 1-10, doesn't need programmable divider and truly works in phase domain. Therefore it's named divider-less ADPLL[6]. Both types have shown rather good performance in recent papers[11][12][13]. In this project we choose the second architecture because comparatively it consumes less power for the same performance in the literature and more techniques have been proposed for this architecture[14].

1-4 ADPLL for this WiMAX Project

1-4-1 Specification Requirement for the Whole System

The specifications of the frequency synthesizer from the customer for WiMAX standard are shown in Table 1-1.

Table 1-1: Specification for the WiMAX ADPLL System.

Parameter	Target
Frequency Bands	2.3-2.7,3.3-3.8 GHz
Frequency Step Size	25 Hz
Integrated SSB Noise (1 kHz - 10 MHz)	-39 dBc
Spot Noise @ 10 kHz	-90 dBc/Hz
Spot Noise @ 100 kHz	-95 dBc/Hz
Far-out Noise	-150 dBc/Hz

According to Table 1-1, this frequency synthesizer is to deliver two frequency bands: *High Band* (HB) which is from 3.3 to 3.8 GHz and *Low Band* (LB) which is from 2.3 to 2.7 GHz.

[6]Note actually some fixed-ratio divider like divide-by-2 or divide-by-4 divider can be introduced in the feedback path of this architecture to ease the requirement of phase accumulator as well as TDC.

Therefore a frequency plan shall be proposed for ADPLL to cover dual band. The frequency step size, i.e. the frequency resolution of ADPLL, shall be within 25 Hz. The phase noise specifications, including the integrated phase noise, spot close-in noise and far-out noise, shall be met.

In measurement, sometimes the RMS phase error rather than the integrated SSB phase noise is given. There is a simple formula for the conversion of these two values:

$$\text{RMS phase error (in degree)} = (\frac{180}{\pi})\sqrt{2\int \mathcal{L}(f)df} \tag{1-6}$$

and $\int \mathcal{L}(f)df$ is just integrated SSB noise. From Equation 1-6, we can know an integrated SSB phase noise of -39 dBc/Hz corresponds to a RMS phase error of 0.91°.

1-4-2 Additional Requirement for this ADPLL Project

In ADPLL, the bulky analog loop filter in the charge-pump PLL is replaced by a digital loop filter, which results in a significant chip area save. Active area in recent ADPLL papers is within 0.5 mm^2[12] [13] [11]. In this project, the whole area of chip would likely be pad-limited due to extra pads for test and measurement. We also plan to add SRAM in the chip for the test. This adds to extra cost of chip area also. Still, we expect the active area for functional part to be within 0.5 mm^2.

Low power consumption is also an important consideration, especially for mobile applications. In this project, the goal of the total power consumption for the core part, excluding FREF slicer, DPA output buffer and Digital test blocks, is expected to be less than 10 mW.

Unlike the traditional charge pump PLL which shows a trade-off between bandwidth and settling time, ADPLL can control its bandwidth easily in the digital domain so that it can achieve both fast-settling and narrow bandwidth. We aim at 10-20 us settling time in this project to show the potential use in the frequency-hopping application.

1-5 Project Sketch

This ADPLL project is a cooperative project in which three MSc students are involved. My part lies in the system level work as well as the DPA design, as shown above. My colleague Popong Effendrik is mainly responsible for the design of TDC and Armin Tavakol is responsible for the implementation of the *digitally controlled oscillator* (DCO) with the related dividers. This project is conducted in Catena Microelectronics BV and supervised by Professor Robert Bogdan Staszewski. The whole effort is to explore how to use ADPLL for the WiMAX standard and converge to a feasible implementation.

1-6 Outline of the Thesis

This thesis focuses on the system level design of ADPLL for the WiMAX application, as well as a simple DPA transistor-level implementation as the output buffer of ADPLL. Chapter 2 shows the architecture of the ADPLL system and the building blocks. Chapter 3 presents the modeling and description of the whole system in Cadence using the Verilog-AMS/Verilog languages. Also the performance analysis of the system is given. In Chapter 4, some advanced algorithms for ADPLL are discussed to overcome the non-idealities of the system. Chapter 5 goes back to the top level of the system, with emphasis on the test plan, the operation mode and the simulation results for the system. Chapter 6 demonstrates a simple implementation of the DPA block for ADPLL. In the last chapter, some conclusions are drawn to summarize the contribution of this thesis and to present the future work. The appendix includes the important data not shown in the bulk part of the thesis, like the code samples, the register map, the figures showing the system performance, etc.

Chapter 2

ADPLL Architecture and Building Blocks

As most complicated systems go, the design of ADPLL needs comprehensive cooperation both at system level and circuit level. Especially when the design is to explore the possibility and do innovations, the methodology of either 'top down' or 'bottom up' will not fulfill the task. The design team has to go through multiple cycles of iteration and negotiation to make the system feasible. To present the work, we choose to first deliver the whole architecture of ADPLL and then dive into the essential blocks.

Our ADPLL is a derivative structure as presented in [3]. The difference with previous one shown in Fig 1-10 is that in this structure the feedback phase information is differentiated into frequency information. After that frequency difference with desired frequency is detected and then accumulated into phase error as input for digital loop filter. Theoretically the function is the same with the previous one except for the extra delay. However, with differentiation and accumulation separated, one can easily freeze the value of PHE signal or reset that to zero. This facilitates the implementation of some algorithm and can help avoid some undesired perturbations. We will return to that in Section 4-1 and 4-2.

2-1 Architecture of ADPLL

A 'bird view' of ADPLL in this project is shown in Fig 2-1. In this figure, our ADPLL system is clearly divided into two divisions: red blocks that demand custom design either due to the analog nature (like DCO) or due to that it is a very high speed logic (like incrementor in ADPLL feedback path); blue blocks that work at low frequency and are expected to be synthesized in digital flow. Therefore it is a mixed-signal system and only

with effort and cooperation from analog/RF designer, digital designer and system engineer can this chip be made possible.

The dash lines in this figure separate the whole chip into five parts. The most right part is DPA which acts as the output buffer of ADPLL. The black blocks above and below DPA block are virtual load intended for future integration into receiver, i.e. they are not contained in the design. The part next to DPA block has DCO and the related dividers/buffers. This part will receive the control from the loop and generate the desired signal for virtual receiver block / DPA block / feedback path in ADPLL. The left part which contains TDC and Retimer+Incrementor receives the feedback signal from DCO and feeds the raw phase information to digital loop logic. The blue part in the middle is the loop logic which processes the phase information from TDC/Incrementor and delivers the control word to the DCO block. The top part with SPI Slave, Sequencer and memory in it will specify the working mode of ADPLL and store snapshot of the system for test. With all these five parts in place, we demonstrate a controllable ADPLL that can close loop with buffers for input (output) signal to go in (out of) chip, in short, a system feasible in silicon.

This chapter is mainly dedicated to the first four parts in this figure. The SPI slave, sequencer and memory will be discussed in detail in Chapter 5. Some advanced algorithms have been applied to ADPLL system and will be discussed in Chapter 4.

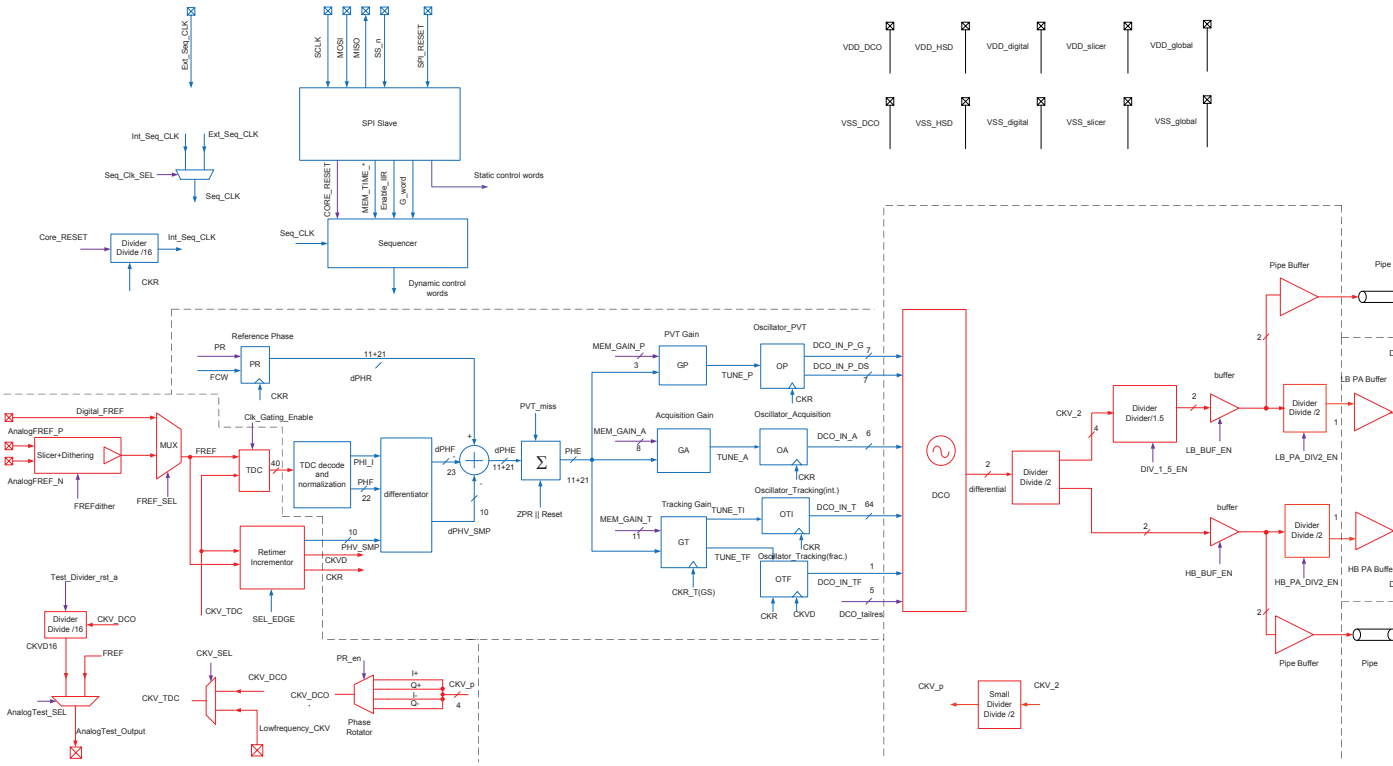

Figure 2-1: Bird view of ADPLL in this project.

2-2 DCO and Related Dividers/Buffers in ADPLL System

As mentioned in section 1-3, DCO has played an important role in ADPLL as the foundation to perform *digital-to-frequency conversion* (DFC). Despite the analog nature of an oscillator, it is encapsulated as an ASIC cell with digital I/O interface. The input tuning words OTW control the output frequency of DCO. The edge transition instances of DCO output signals will contain all the information we want, like frequency, phase noise and I/Q phase mismatch. Therefore, the modeling of DCO from the system point of view can be greatly simplified. The specifications and consideration for DCO from system level are also pretty straight forward.

It shall be noted that in ADPLL DCO need to connect to other blocks. DCO's outputs will be divided and fed back to TDC and high-speed counter in ADPLL. Then the loop can correct for DCO's imperfections such as frequency drift and frequency pushing. The outputs of DCO are also supplied to DPA and then go off-chip. Also the quadrature signals shall be generated for the potential quadrature mixers in receiver.

All these connections are shown in zoom-in bird view of ADPLL system for DCO block, as in Fig 2-2. To avoid the I/Q mismatch during signal propagation, differential signal with 2 times the desired frequency are needed for the local divide-by-2 quadrature signal generation within receiver. So DCO need to deliver signal with frequency of 4.6-5.4 GHz and 6.6-7.6 GHz. The ratio of highest frequency for HB and lowest frequency in LB is almost octave. Thus it's hard for oscillator to directly cover this range. The frequency plan in this project is to generate signal with frequency at 6.6-7.6 GHz using a divide-by-2 divider and generate signal with frequency at 4.6-5.4 GHz using a divide-by-3 divider. Therefore the oscillator only needs to cover frequency range of 13.2-16.2 GHz.

It shall be noted that to take frequency deviation of PVT corner into consideration, DCO core need to have about 1.5 GHz margin besides the frequency range specified above, i.e. 11.7-17.7 GHz. Still the only difference shown in system simulation is the deviation of central frequency and can be easily compensated by calibration. Thus the frequency deviation of DCO core is not taken into account into the system level simulation.

Fig 2-3 shows the schematic for frequency planning. The divide-by-3 function is implemented as a cascade of divide-by-2 and divide-by-1.5 so that divide-by-2 divider can be shared for HB and LB. Another divide-by-2 divider in the feedback path is adopted. Thus feedback CKV is just one fourth of output frequency of DCO core (3.3-4.05 GHz). The design difficulty of TDC and the high-speed incrementor is reduced.

The dividers and buffers in Fig 2-2 are pretty simple. In Fig 2-2, there are five control signals to turn on/off each divider or buffer. DIV_1_5_EN is to enable/disable the divide-by-1.5 divider for LB TX and RX. LB_BUF_EN(HB_BUF_EN) turns on and off the buffer for LB(HB) output. LB_PA_DIV2_EN and HB_PA_DIV2_EN will enable/disable the divide-by-2 divider for LB TX and HB TX. The redundancy of control signals provides both the flexibility and the way to test the power consumption of every divider/buffer.

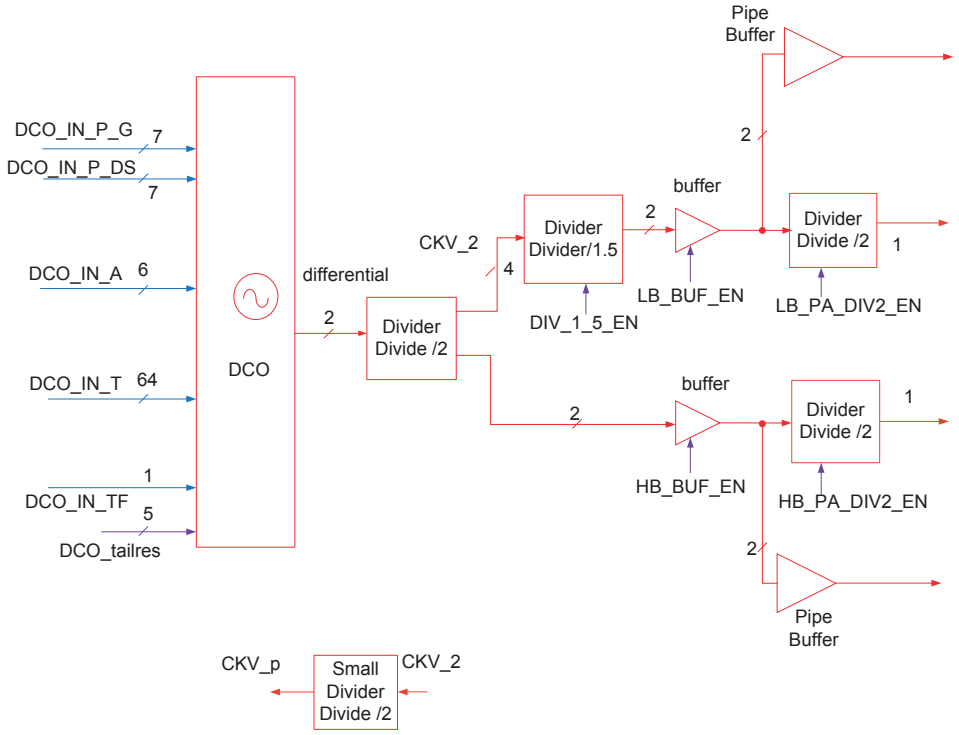

Figure 2-2: Zoom-in bird view for the DCO block.

The focus is the DCO core as in Fig 2-4, which is a push-pull oscillator. The cross-coupled transistors provide negative resistance needed in the oscillator. Compared with normal cross-couple based oscillator, an obvious difference is that some resistors are added to lower down the flicker noise within DCO. As we will see later, the flicker noise in DCO is up-converted to $1/f^3$ in phase noise spectrum and may hurt close-in phase noise performance of ADPLL. Reduction of this eases the effort to meet system specifications. The inductance value in LC tank of oscillator core is fixed. OTW will control the capacitance banks in LC tank. In that way, the frequency of the oscillator core is digitally controlled. The 5 bit binary DCOtailres signal at the tail of oscillator can specify the bias current of the core. The oscillator core is turned off when DCOtailres value is 0. Larger value of DCOtailres means more current bias and thus larger negative resistance from cross-coupled transistors for the oscillator. Then the amplitude of oscillator output signal is larger and the phase noise performance also improves. By modifying this signal, the situation that oscillator cannot start up will be prevented just by pumping in more current, and we can have control on the trade-off between performance and power consumption.

DCO in this project has three banks with different dynamic range and resolution in the frequency domain: PVT bank (PB), acquisition bank (AB) and tracking bank (TB). They together guarantee both the frequency range the DCO core need to cover and the resolution DCO is to deliver. The control words for PB and AB are separately 7 bits and 6 bits binary code. For TB, 64 bit thermometer-code control word is chosen to satisfy the matching

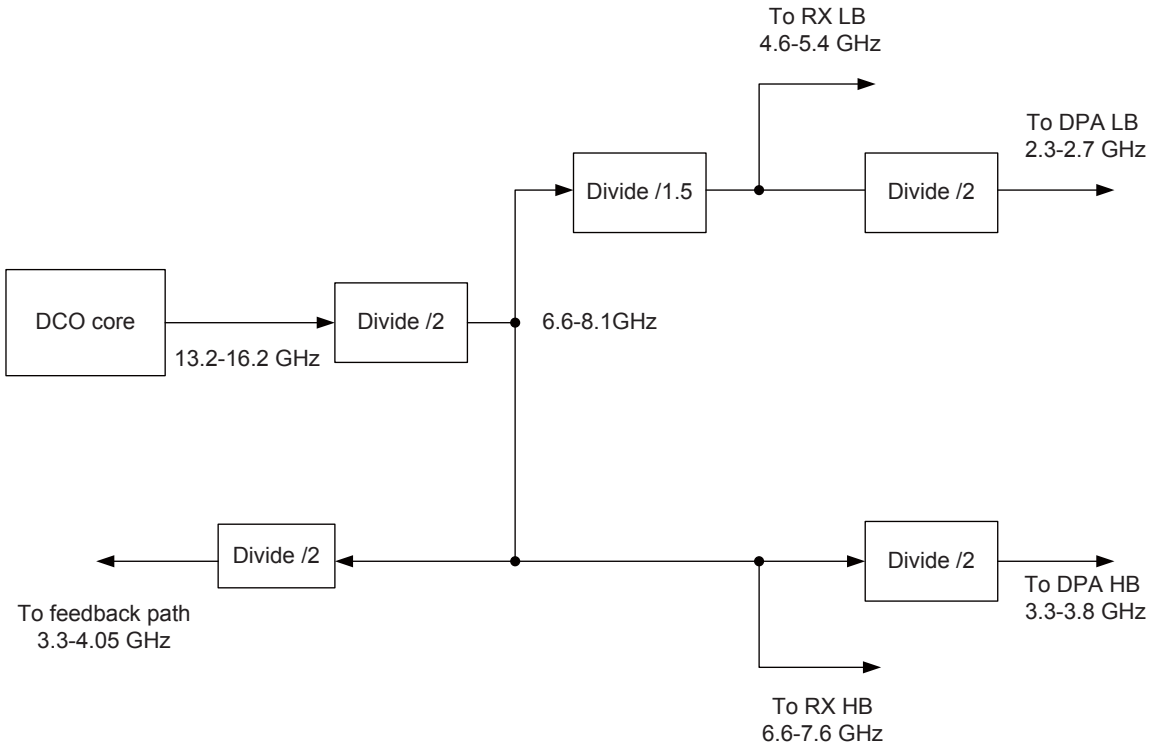

Figure 2-3: Frequency planning of ADPLL

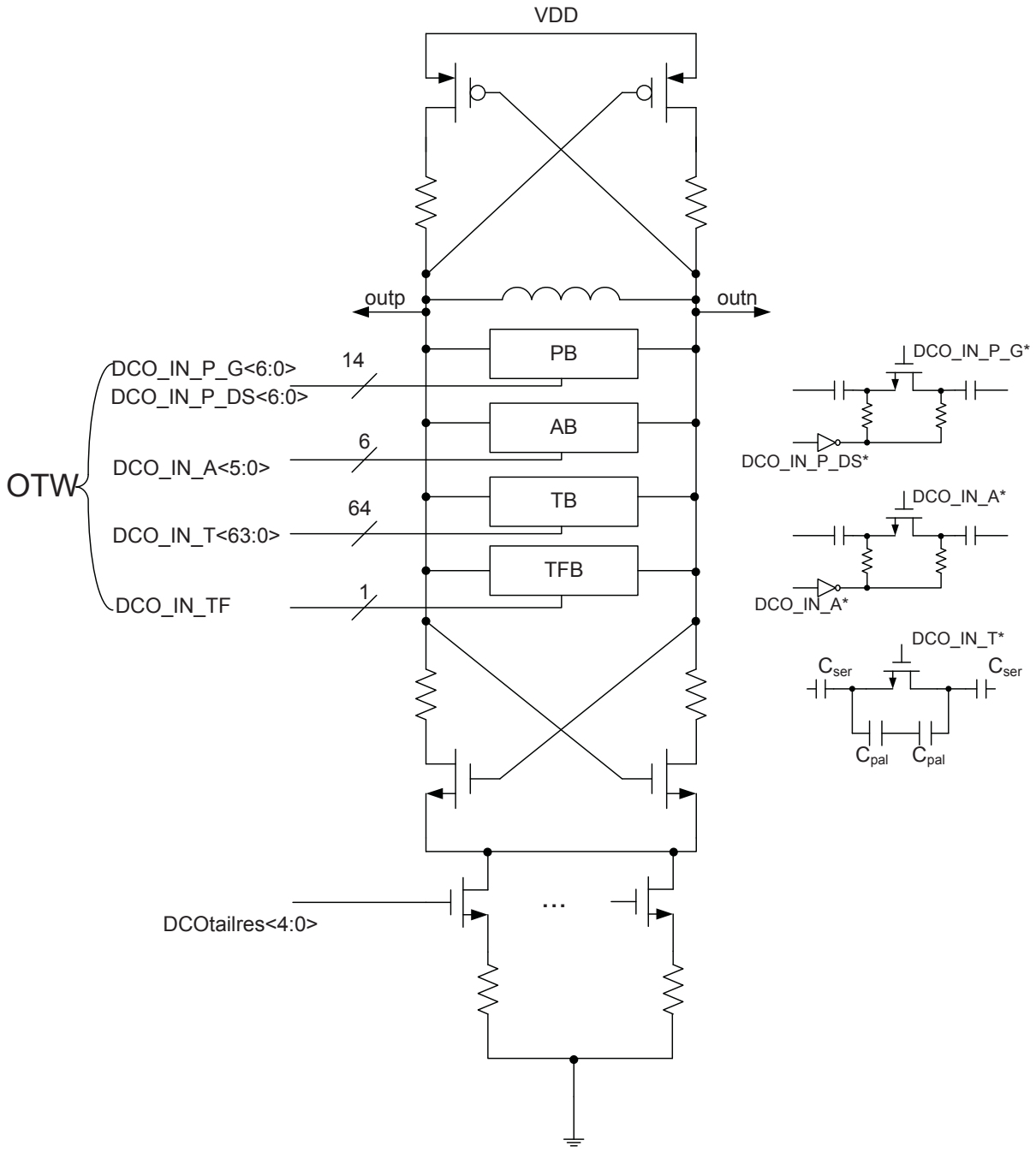

Figure 2-4: Simplified schematic for the DCO oscillator core.

requirement. As shown in Fig 2-4, in PB and AB, the input tuning words control NMOS switches to turn on/off capacitors. The resolution we can achieve via this method is limited by the smallest capacitance value the process can provide and the parasitic of the switches, which is not good enough. To achieve a finer resolution, in TB the switch is in parallel with relatively large capacitors (compared to the capacitor in series with the switch). When switch is turned on, the large capacitors are shorted and only series capacitors show up. When switch is turned off, what we see from the terminals are small capacitors in series with big capacitors. Say the capacitance value for the series capacitor is C_{ser} and the capacitance for the parallel capacitor is C_{pal}, the turn-on capacitance value for the cap unit is $\frac{C_{ser}}{2}$ while the turn-off value for the cap unit is $\frac{1}{2}\frac{C_{ser}C_{pal}}{C_{ser}+C_{pal}} = \frac{C_{ser}}{2}\frac{C_{pal}}{C_{ser}+C_{pal}}$. Thus capacitance change can be really fine by using huge parallel capacitors. However, here the analysis is simplified and for practical design, the effect from not-ideal switches (finite resistance both on and off) and parasitic capacitance kick in. The accurate values of capacitance change should be extracted via the transistor-level analysis and simulation.

For the nominal case, the capacitance change corresponding to on/off of LSB of PVT bank (ΔC_P) is 2.35 fF, the capacitance change corresponding to on/off of LSB of acquisition bank (ΔC_A) is 160 aF and the capacitance change corresponding to on/off of LSB of tracking bank (ΔC_T) is 10 aF. The central capacitance is set to be 361 fF and the inductance is 325 pH.

The change of capacitance will correspond to a change of frequency. For the total capacitance of C_{tot} and inductance value of L, the oscillation frequency is

$$f = \frac{1}{2\pi\sqrt{LC_{tot}}} \tag{2-1}$$

the derivative would give

$$\Delta f = -\frac{1}{2}\frac{1}{2\pi\sqrt{LC_{tot}}}\frac{\Delta C}{C_{tot}} = -f\frac{\Delta C}{2C_{tot}} = -2\pi^2 L f^3 \Delta C \tag{2-2}$$

As is shown the ratio of Δf and ΔC is proportional to f^3. For oscillator with narrow tuning range, f^3 will not change that much. However, in our case $(f_{MAX}/f_{MIN})^3 = (16.2/13.2)^3 = 1.85$. The variation of $\frac{\Delta f}{\Delta C}$ respect to frequency shall be taken into consideration. Table 2-1 shows the frequency change of DCO core for LSB in PB/AB/TB at the highest frequency, the lowest frequency and the central frequency. When we consider the frequency resolution of the feedback CKV, i.e. K_{dco}, it is one fourth of the values in this table.

The fractional part of tracking bank (TFB) utilizes the high-speed dithering to further increase the frequency resolution. This dithering is achieved by using a digital Sigma-Delta modulator with a high speed clock. The average of high-rate toggling 0/1 bit stream approaches low-speed fractional input. In our design, a 1st order Sigma-Delta modulator is adopted. As shown in Fig 2-5, it is just a clocked adder. The high speed clock is the CKVD8 signal in Fig 2-12, which is CKV divided by 8. The 5 bit fractional input

Table 2-1: Frequency resolution for PB/AB/TB in DCO core.

Frequency resolution	Lowest frequency 13.2 GHz	Central frequency 14.7 GHz	Highest frequency 16.2 GHz
PVT bank	34.67 MHz	47.89 MHz	64.10 MHz
acquisition bank	2361 kHz	3261 kHz	4364 kHz
tracking bank	148 kHz	204 kHz	273 kHz

corresponds to a frequency resolution of $\frac{1}{2^5}\frac{204\text{kHz}}{4} = 1.6\text{kHz}$ in the central frequency. This simple structure reduces the power compared to higher-order modulator and the matching requirement is lower as we only have 1 bit output. One may worry that the 1st order modulator cannot fully randomize the output stream and will result in too high spurs. Yet the input of this modulator is not constant due to the noise of DCO and TDC. And the clock of this modulator works at a rather high speed. So the far-out spur due to dithering is spread out and is sufficiently low, though still observable, as shown in Sec. As this modulator is expected to be synthesized, we will come back to it later.

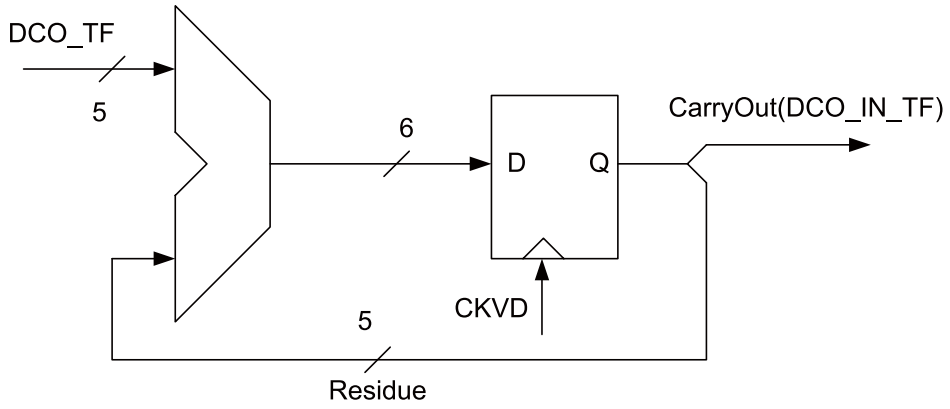

Figure 2-5: The 1st $\Sigma\Delta$ Modulator for fractional part of tracking bank.

The cooperation of PB, AB, TB and TFB provides both a sufficient frequency range and a sufficient frequency resolution. As ADPLL needs to settle to a certain frequency, it will traverse the modes of using PB (PVT mode), AB (acquisition mode), TB & TFB (tracking mode), as in Fig 2-6. The mode switchover is controlled by the sequencer discussed in Chapter 5.

2-3 TDC and Incrementor in ADPLL System

A zoom in version of the block at the left part of Fig 2-1 is shown in Fig 2-7. As is shown, TDC and Incrementor+Retimer lie at the heart of this block. They together provide the sampled feedback phase information for comparison in digital phase detector.

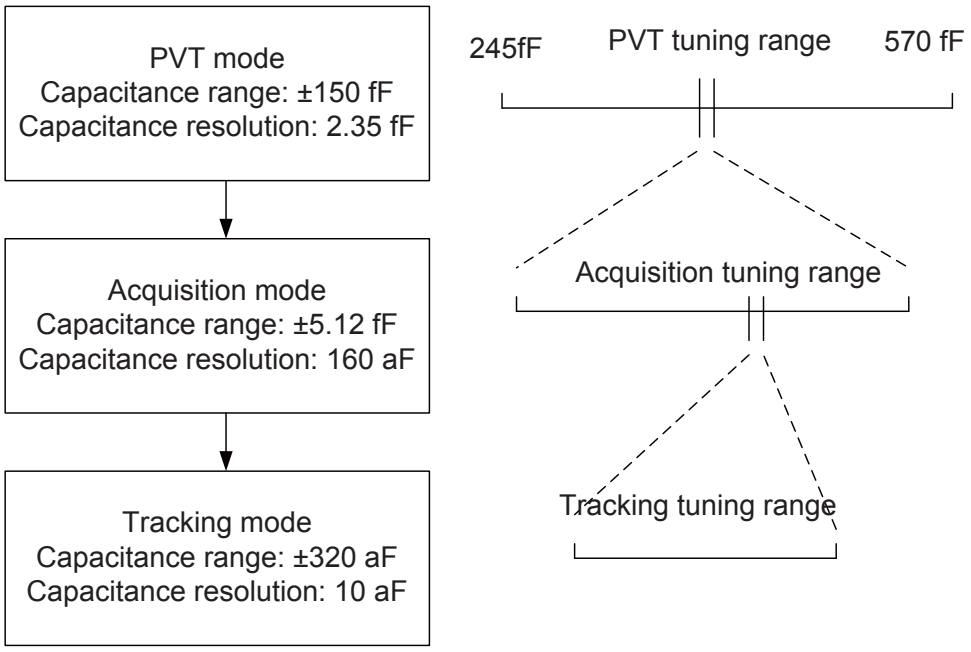

Figure 2-6: Flowchart of DCO operation modes.

A simplified timing diagram in Fig 2-8 illustrates how phase information can be measured via cooperation of the retimer, the incrementor and TDC. Say the reference clock is FREF and the feedback clock is CKV. Their positive edges are shown in the timing diagram. The function of retimer is just to clock FREF with CKV. Then the retimed output CKR features aligned edges with CKV and average frequency same with FREF. If the frequency of CKV is 2.25 times the frequency of FREF and we assume the phase starts from 0, the phase of CKV at FREF rising edges (measured in cycles) shall be sequentially 2.25, 4.5, 6.75, 9, 11.25, 13.5, 15.75, 18 and etc. The high-speed incrementor counts the cycle numbers of CKV and its output (PHV_SMP) is sampled by CKR. Then we have PHV_SMP with value of 3, 5, 7, 9, 12, 14, 16, 18 and etc. From this we can see that PHV_SMP is a quantized output of CKV phase with resolution of one cycle (2π). On the other hand, the timing difference between rising edges of FREF and CKR is measured in TDC and then normalized to one cycle of CKV to correct for the quantization error of PHV_SMP. These two combined together deliver exactly the feedback phase information. Fig 2-9 is a schematic of the quantization error from the incrementor and the correction from TDC with respect to phase (cycles of CKV) in the ideal situation.

There are other modules in Fig 2-7. The phase rotator receives quadrature feedback signals and executes the phase rotation algorithm to suppress spurs in near integer-N cases, which will be discussed in detail in Chapter 4. The signal CKV_SEL controls a multiplexer to choose whether the CKV for TDC is from DCO as in normal ADPLL operation or from off chip generated signal for TDC test. The feedback CKV from DCO is divided by 16. The signal AnalogTest_SEL chooses off-chip analog signal between reference clock FREF and a signal which is CKV divided by 16. If the divided-by-16 CKV is fed off-chip, we can check

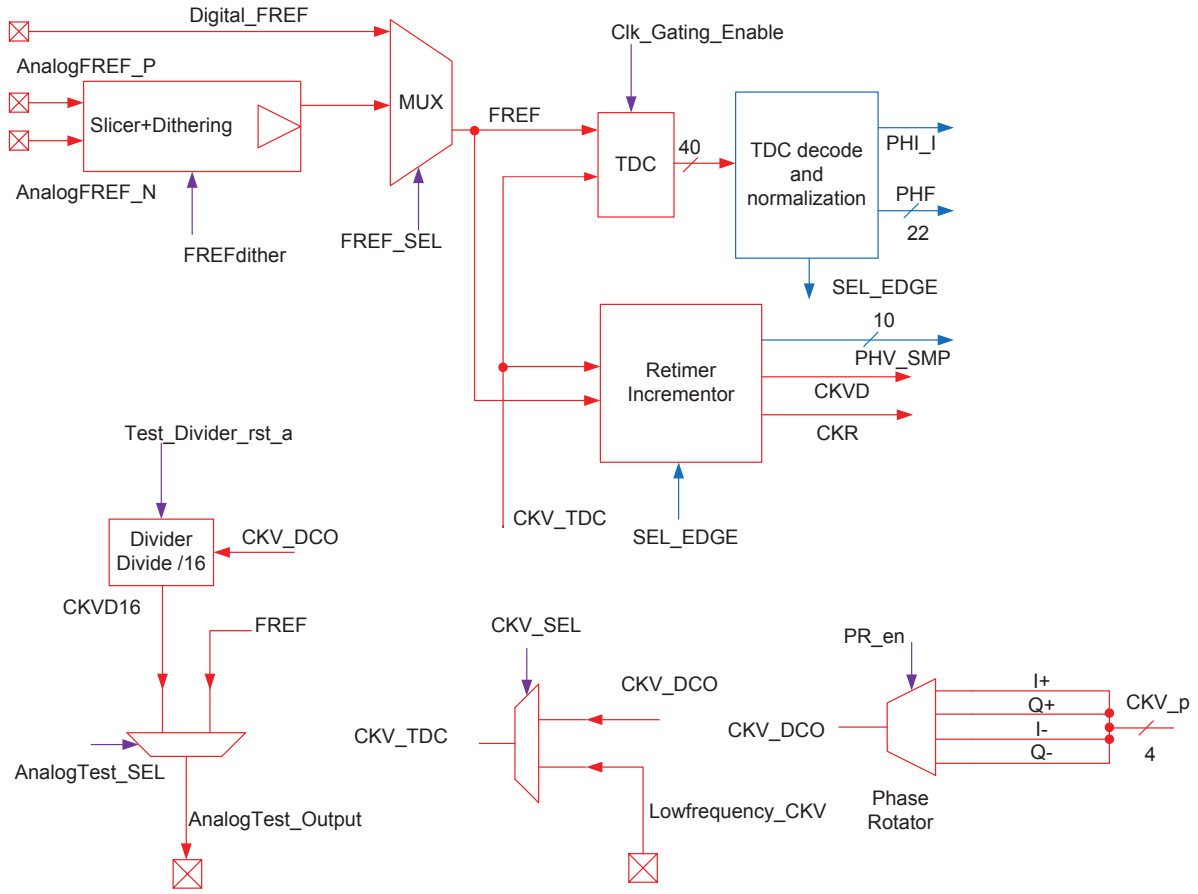

Figure 2-7: Zoom-in bird view for block with TDC and Retimer+Incrementor.

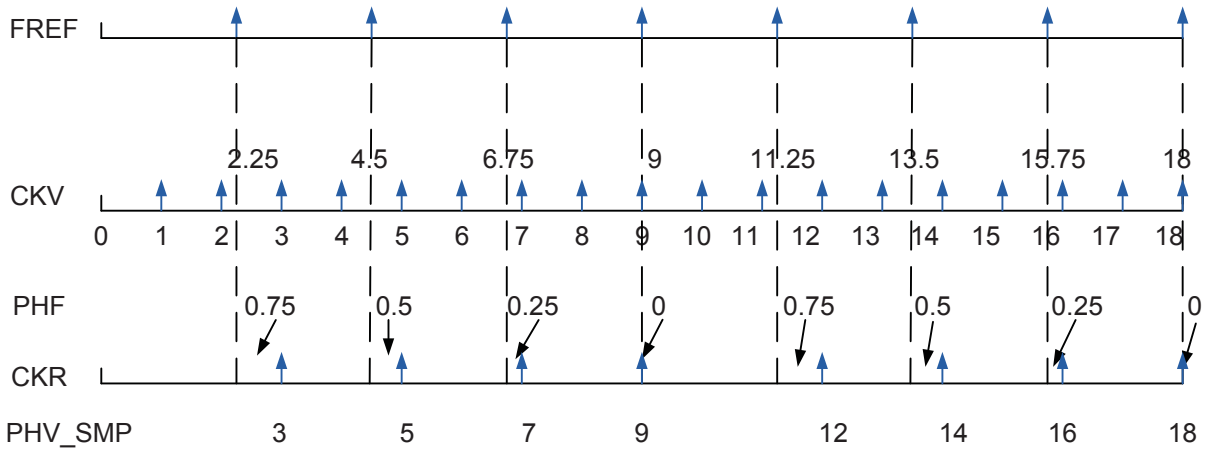

Figure 2-8: Simplified timing diagram on the working mechanism of retimer, incrementor and TDC.

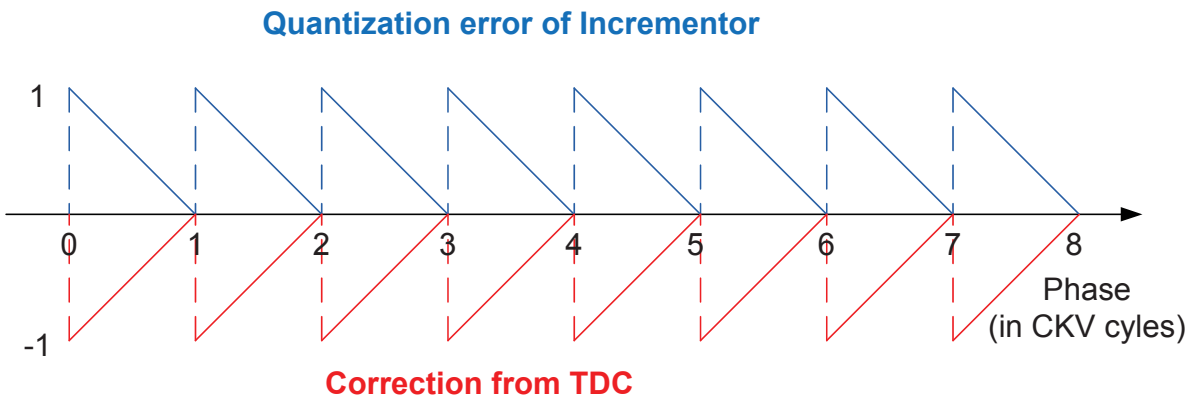

Figure 2-9: Schematic of incermentor's quantization error and TDC's correction in ideal situation.

the output frequency of ADPLL and the close-in phase noise without DPA turned on. If FREF is fed off-chip, we can check whether reference clock is working as we expected. They are discussed in detail in Chapter 5.

FREF slicer in Fig 2-7 receives the off-chip analog differential sinusoidal signal from crystal oscillator and then generates the square wave clock signal. The generated FREF signal must have a low enough phase noise so as not to degrade the ADPLL performance. In our design, the FREF signal will be dithered to suppress spurs in near integer-N situations and we will come back to this topic in Chapter 4. In case that the slicer cannot work properly, a lower quality signal digital_FREF is fed to the chip to make sure that at least ADPLL can still work. This digital_FREF signal is also utilized in the TDC test plan, as in Section 5-3-3. A multiplexer chooses which of these two FREF signals to be the FREF signal in TDC and the incrementor.

TDC and the high-speed incrementor are rather complex modules and are critical to the performance of the whole system. Therefore they are expected to be fully-custom designed. This thesis will treat them more from system-level view. The detailed transistor-level design is beyond the scope.

2-3-1 TDC in this ADPLL System

As is stated above, TDC in the general sense is to deliver the time difference between the edges of two signals. In the ADPLL system, more specifically, the time difference between the rising edge of FREF and CKR is needed. Because the edge of CKR and CKV are aligned, in the implementation, TDC measures the time difference between FREF and CKV edges.

Here we adopt the original pseudo-differential TDC topology composed by cell with inverters and flip flops as in Fig 2-10[15]. The inverter acts as a delay element with a resolution

of T_{inv} between 10.5 ps to 12.5 ps. So the CKV signal will be delayed by kT_{inv} after it propagates through k inverters. The flip flop is clocked by FREF and will sample the value of delayed CKV at the FREF rising edge[1]. Notice that after every inverter, the polarity of the signal (high or low) will be reversed. To make the output right, in Cell #1, the positive input of the flip flop is connected to the delayed version of CKV‘ and the negative input of flip flop is connected to the delayed version of CKV. While for Cell #2 the connection is reversed. Such a toggling connection goes on in the whole TDC chain.

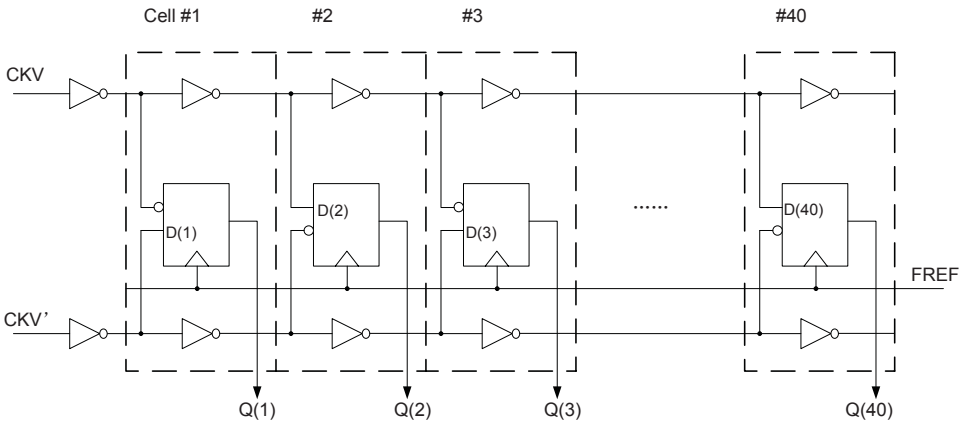

Figure 2-10: Pseudo-differential TDC architecture.

For the analysis of the TDC module, just look at the positive input of flip flops, it is equivalent to that CKV is delayed by buffers with delay of T_{inv} and then sampled by FREF. We name the positive input of kth flip flop as D[k] and then we have timing diagrams for TDC as in Fig 2-11. In Fig 2-11 (a), the last CKV edge before the FREF rising edge is falling edge while in fig 2-11 (b), the last CKV edge before the FREF rising edge is rising edge. In the traditional charge pump PLL, the case in (a) is called 'FREF leads CKV' while the case in (b) is called 'FREF lags CKV'. Also we assume that one CKV cycle T_{CKV} is exactly 8 T_{inv} to simplify the illustration (In reality, T_{CKV} is more than 20 T_{inv} and we will talk about that later.). For (a) the output of flip flops Q(1:9) is 001111000. The transition of $0 \rightarrow 1$ happens between Q(2) and Q(3). Then the timing difference between the FREF rising edge and the last CKV falling edge earlier than FREF rising edge, which is shown as T_F, is quantized as two inverter delays ($2T_{inv}$). Similarly, since the transition of $1 \rightarrow 0$ happens between Q(6) and Q(7), the timing difference between the FREF rising edge and the last CKV rising edge earlier than the FREF rising edge, which is shown as T_R, is quantized as $6T_{inv}$. For the case in (b), we can also conclude that T_R is $2T_{inv}$ and T_F is $6T_{inv}$.

Therefore the time difference between CKV edges and FREF edges are quantized in this TDC by the resolution of T_{inv}. Yet the desired information is actually the timing difference

[1]Here flip flop is simplified as an ideal sampler. Practical flip flop has setup time and hold time requirement. We will talk about that later.

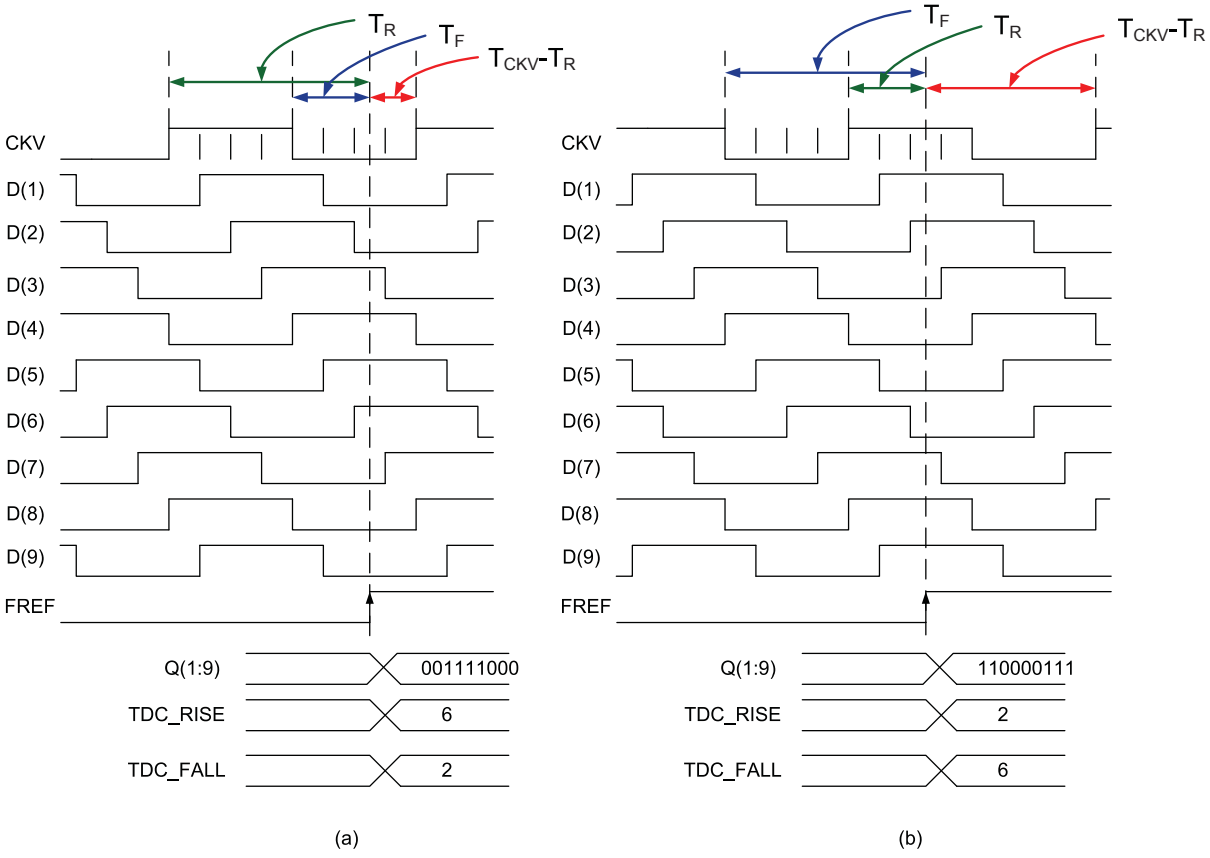

Figure 2-11: Timing diagram for TDC.

between the FREF rising edge and the first CKV rising edge no earlier than FREF rising edge, which is shown in Fig 2-11 as $T_{CKV} - T_R$. It will be normalized to T_{CKV}, transferred to phase domain as PHF in Fig 2-8 and combined with incrementor output for phase comparison in digital logic. Assume that T_R is quantized as kT_{inv}, we have the phase domain output as:

$$\frac{T_{CKV} - T_R}{T_{CKV}} = 1 - \frac{kT_{inv}}{T_{CKV}} = 1 - k\frac{T_{inv}}{T_{CKV}} \tag{2-3}$$

Thus it's important to estimate $\frac{T_{inv}}{T_{CKV}}$, which we call K_{tdc} with enough accuracy. In Fig 2-11, the quantized output for T_R and T_F are called TDC_RISE and TDC_FALL. Notice that the difference of T_R and T_F is half cycle of CKV, so the difference between TDC_RISE and TDC_FALL can be an estimation of $\frac{1}{2}\frac{T_{CKV}}{T_{inv}}$. We have

$$\frac{T_{CKV}}{T_{inv}} \approx 2(|\text{TDC\_FALL-TDC\_RISE}|) \tag{2-4}$$

The inverse is just K_{tdc}.

This estimation is rather coarse as it's limited by TDC resolution. A way to approach enough accuracy is to accumulate multiple cycles of the estimated value and then do averaging. We will return to this issue later. For now, one can just assume that this normalization is accurate enough.

To catch both the RISE edge and the FALL edge of CKV, TDC shall guarantee to cover one full cycle of CKV. The number of TDC cells L can be calculated as below:

$$L \geq \frac{max(T_{CKV})}{min(T_{inv})} \tag{2-5}$$

From Fig 2-3 we know that the lowest frequency of CKV is 3.3 GHz, corresponding to 300 ps for $max(T_{CKV})$. $min(T_{inv})$ is 10.5 ps from corner simulation result[15]. Then L shall be no less than 29. To save some margin, L in this design is chosen as 32, i.e. TDC_RISE and TDC_FALL will be 5 bits. To minimize the mismatch, 4 cells have been added before and after the effective cell chain. Therefore we have 40 cells in this TDC as in Fig 2-10.

2-3-2 Retimer and High-Speed Incrementor

As the retimer and incrementor is not implemented at the transistor level yet, for modeling purpose, we adopt the topology as in Figure 4.27 of [14]. The schematic is redrawn in Fig 2-12.

The retimed clock CKR is generated by clocking FREF with CKV, which can be done via an ideal flip flop as is shown in Fig 1-10. However, in reality the flip flop has metastability, i.e. when the rising edge of FREF and CKV are too close, it may take very long time for the flip flop to resolve the value of CKR. Then it is not proper to rely on CKR as the clock

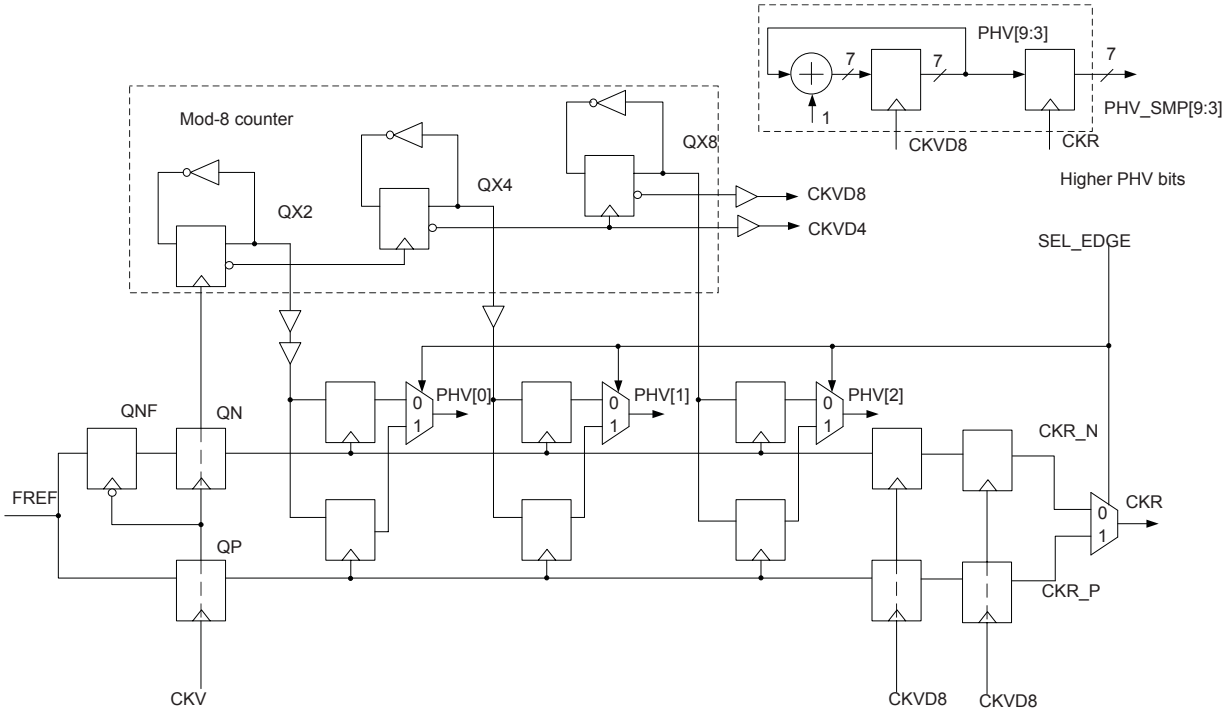

Figure 2-12: Schematic of proposed implementation of the retimer and incrementor.

for the system. The way to solve this problem is to sample FREF both at the CKV rising edge and the CKV falling edge, as shown in Fig 2-12, the output are named respectively QP and QNF. Since at least one of the two CKV edges are far away from the FREF rising edge, either QP or QNF will deliver the correct sampled value within a short time. QNF is sampled by the CKV rising edge and the output is QN. Then QN and QP are aligned with CKV.

The top part of Fig 2-12 is a mod-8 counter for the 3 least significant bits in PHV. As shown in the timing diagram of Fig 2-13, PHV[2:0] does increase by 1 at every CKV cycle. The signal CKVD8 can be used as clock in the synchronous counter for the higher bits of PHV. Therefore the higher bits can just work at lower frequency to reduce the power consumption.

PHV[2:0] are both sampled by QP and QN in Fig 2-12. For QP and QN, corresponding to the two situations as in Fig 2-11 (a) and Fig 2-11 (b), we have timing diagram shown in Fig 2-14(a) and Fig 2-14 (b). In this timing diagram QN lags QP for one CKV cycle in the case of Fig 2-11 (a) while QN and QP are aligned in the case of Fig 2-11 (b). Thus we need to compensate for the case (a) when SEL_EDGE is 1. One can observe that the difference between Fig 2-11 (a) and Fig 2-11 (b) is the value of Q(1). Q(1) is 0 in case (a) while and Q(1) is 1 in case (b). So we have the logic for PHV compensation as:

When SEL_EDGE is 0 and Q(1) is 0, PHV_SMP shall minus 1 for correct operation.

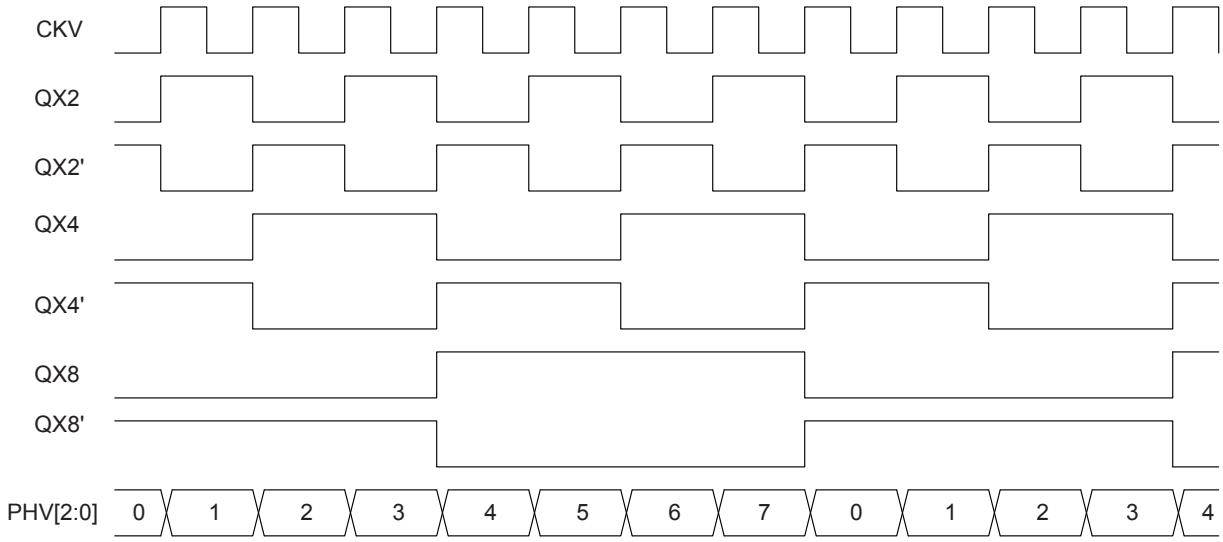

Figure 2-13: Timing diagram for signals in a mod-8 counter.

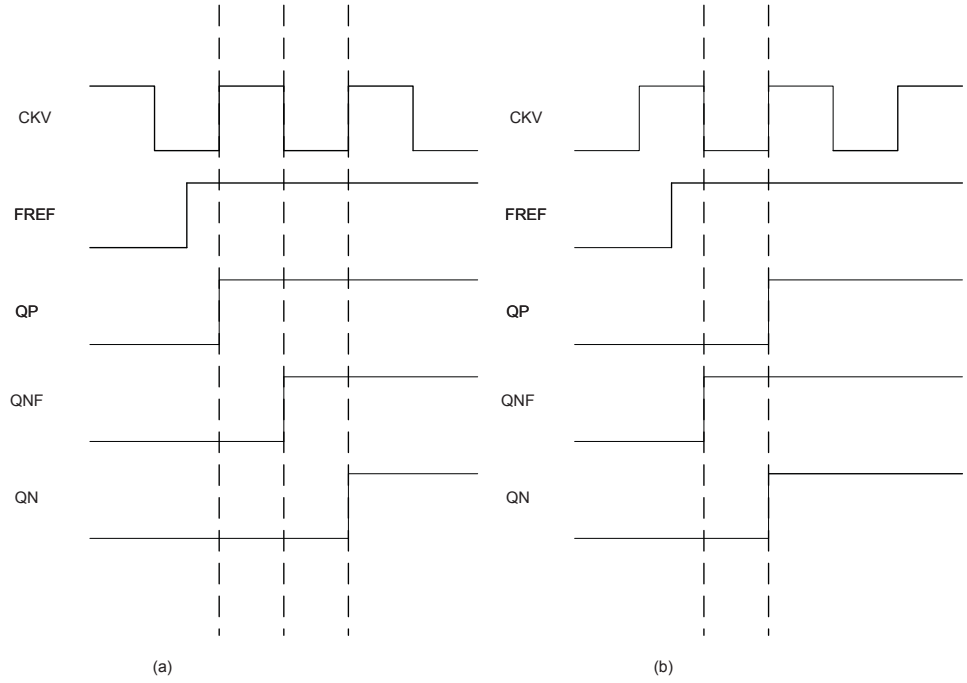

Figure 2-14: Timing diagram for QP and QN in the retimer.

2-3-3 Cooperation of TDC and Retimer+Incrementor

Now we have a functional TDC and a functional Retimer+Incrementor. It's indispensable to inspect their cooperation so that they can deliver the desired phase information. There are two important issues that are of concern:

1. The generation of SEL_EDGE signal.

   One may notice that for the operation of Retimer+Incrementor, there is an important signal SEL_EDGE. This is to choose CKR_P or CKR_N as the retimed clock CKR. As seen in Fig 2-7, this signal is generated from the TDC decoder logic by utilizing the redundancy of the TDC output. Assume that $T_{CKV} = 20T_{inv}$, which more resembles our design than the situation shown in Fig 2-11 with $T_{CKV} = 8T_{inv}$, if FREF rising edge and CKV rising edge are too close, we have timing diagram as in Fig 2-15. One can conclude that for both cases, the value of Q(2) to Q(8) are all 0. Actually as shown in the figure, Q(k) is the value of CKV that is kT_{inv} ahead of FREF sampling moment. Thus if Q(k) is 0 for k that is around $\frac{1}{4}\frac{T_{CKV}}{T_{inv}}$ (in Fig 2-15 the value is 5), it means that CKV falling edge is far away from FREF rising edge. Then SEL_EDGE can be made low to choose the falling edge. Also if Q(5) is 1, then the CKV rising edge is far away from the FREF rising edge and SEL_EDGE can be made high to choose the rising edge.

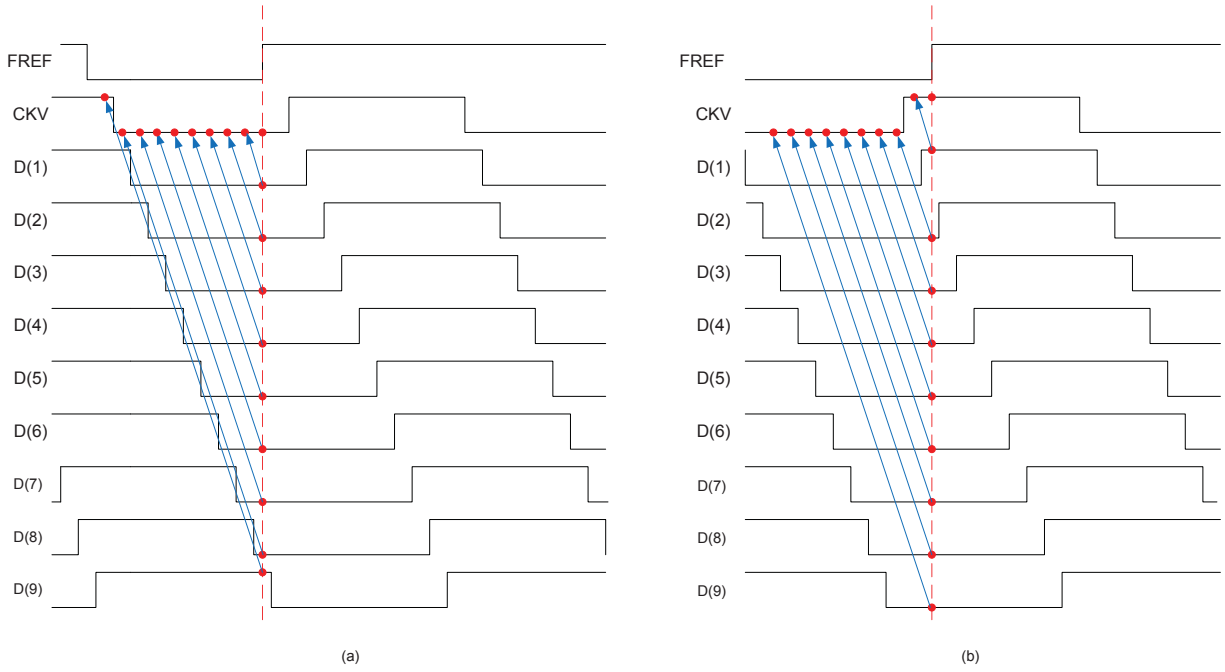

Figure 2-15: Timing diagram for SEL_EDGE signal.

In our design $\frac{T_{CKV}}{T_{inv}}$ can vary with desired frequency as well as TDC delay variation

caused by *process, voltage and temperature* (PVT). It may change between 20 and 30. Still, we can just choose Q[5] as the SEL_EDGE signal.

There is a doubt with the metastability of the SEL_EDGE signal itself. Notice that when SEL_EDGE is metastable, it means that the rising edge of D[5] is close to the rising edge of FREF, which has the hidden meaning that neither the rising edge nor the falling edge of CKV will be close to the rising edge of FREF. In that sense we can choose either CKR_P or CKR_N in Fig 2-12 as CKR for the digital domain operation. The only concern is that we shall make sure that SEL_EDGE is a stable 0 or 1 before the rising edge of CKR_P or CKR_N to avoid the possible glitch. That is achieved by inserting two columns of flip flops clocked at CKVD8 as in Fig 2-12. As CKVD8 has a maximum frequency of 500 MHz (4.05 GHz/8), this adds a minimum delay of 2 ns for the SEL_EDGE signal to resolve the metastability. From Figure 5-11 of [15] (redrawn here as Fig 2-16), the extrapolation of the curve gives a metastability window t_w of less than 3×10^{-26} s for the delay of 2 ns. The error probability for SEL_EDGE is

$$\frac{P_e}{1 \text{ second}} = \frac{t_w}{T_v} f_R = t_w \cdot f_v \cdot f_R < 3 \times 10^{-26} \times 4.05 \times 10^9 \times 40 \times 10^6 = 4.86 \times 10^{-9} \quad (2\text{-}6)$$

The mean time between failures is 6.48 years, which is just the inverse of the value above. Then it is almost impossible for the metastability of SEL_EDGE to propagate into the whole system. If we want higher reliability for operation of more than 5 years, another column of flip flops clocked at CKVD8 can be added.

2. The PHE spike problem. The PHE spike problem can be illustrated in an extreme situation. Say FREF rising edge and CKV rising edge are just at the same time, when we observe at that time, would incrementor increase by 1 or don't increase? Would TDC deliver a compensation value of 0 or 1?

   If the incrementor increases by 1 and the compensation of TDC is 1, then the accumulated output phase is correct. This also applies when the incrementor doesn't increase while TDC compensates 0. However, if the incrementor increases by 1 and the compensation of TDC is 0, the output phase is one cycle larger than the correct value. Then PHE will show a spike of -1. If the incrementor doesn't increase and the compensation of TDC is 1, the output phase is one cycle smaller than the correct value. PHE will show a spike of +1.

   The spike of ± 1 also happens when TDC path and Incrementor path has any time mismatch. This may be because of the CKV delay difference for the two modules due to layout imperfections or the setup time difference of flip flops in the two modules. Fig 2-17 shows the consequences if there is some time mismatch between two modules. Here we assume FREF signals of the two modules are exactly the same while CKV signals are not. [2] We name the CKV signal for TDC as CKV_TDC while the

[2] Actually the essential reason is the mismatch of time difference between FREF edge and CKV edge in the two modules.

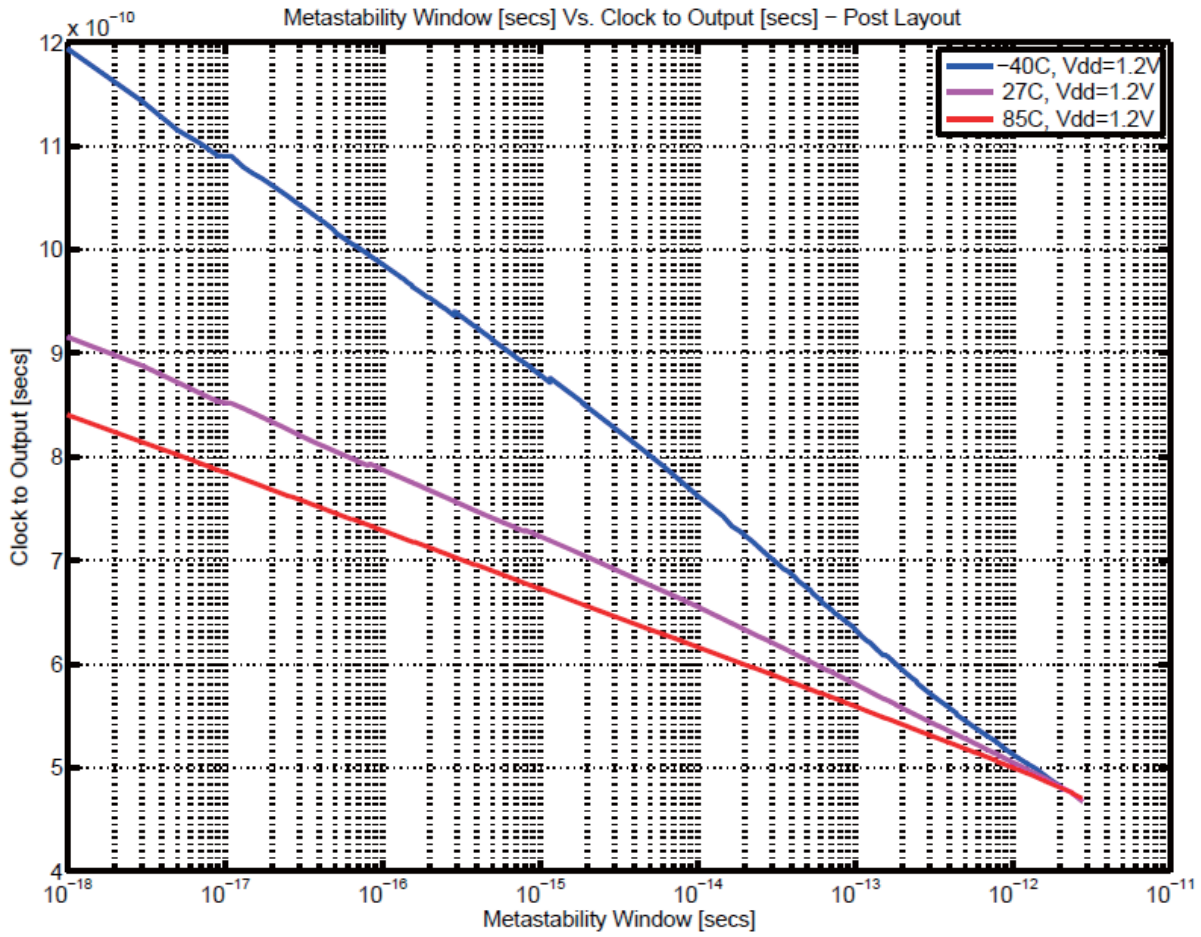

Figure 2-16: Metastability window of sense amplifier flip-flop.

CKV signal for Retimer+Incrementor as CKV_inc. In Fig 2-17 (a) CKV_inc lags CKV_TDC by a very small fraction of CKV cycle ϵ. As shown in the timing diagram, for most of the time, when the FREF rising edge is not between CKV_inc rising edge and CKV_TDC rising edge, the compensation due to TDC is smaller than quantization error of incrementor by ϵ. That kind of offset is of no significance to the operation of ADPLL. However, if FREF edge is between CKV_TDC and CKV_inc, the compensation due to TDC is larger than the quantization error of the incrementor by $1-\epsilon$. So comparatively, we have a PHE spike of +1. Similarly, for Fig 2-17 (b) CKV_TDC lags CKV_inc by just ϵT_{CKV}. For most of the time, the quantization error of incrementor is smaller than the TDC compensation by ϵ while if FREF rising edge is between CKV_inc rising edge and CKV_TDC rising edge, the quantization error of incrementor is larger than the TDC compensation by $1-\epsilon$. Then PHE spike of -1 can be expected.

It may be easier to explain this phenomenon in Fig 2-18. Unlike the ideal situation in Fig 2-9, Fig 2-18 (a) corresponds to Fig 2-17 (a) and Fig 2-18 (b) corresponds to Fig 2-17 (b). Here ϵ is chosen as 0.1 and one can easily see the periodical PHE spike as in deep green curve.

In previous works on ADPLL, this phenomenon is also mentioned [16, 17]. The method proposed in these papers is basically to detect this spike in the digital logic and then compensate for that. However, notice that an intentional frequency hopping can also cause a sudden change of PHE. Besides that, some unexpected change of temperature or supply voltage may also lead to such effect. To distinguish that from this spike, the detection algorithm has to be robust, which may add to the complexity of the logic and introduce extra delay in the loop.

For our design we have SEL_EDGE which will choose the CKV falling edge to retime FREF when the CKV rising edge and the FREF rising edge is too close. That means for the incrementor we are actually using the CKV rising edge that follows the falling edge. Then the quantization error can be larger than one CKV cycle. The consequences of time mismatch between TDC module and Retimer+Incrementor module is shown in Fig 2-19. Compared to Fig 2-17, one may notice that for case (a), when FREF is between rising edge of CKV_TDC and CKV_inc, the offset is the same as the normal situation. However, for case (b), we still have an offset of 1.

It is much more clear when we consider Fig 2-20. Case (a) and case (b) both have time that a spike of +1 happens (that is a spike of -1 for PHE). Now we can know this can happen and only happens when SEL_EDGE chooses CKV falling edge for retiming and FREF rising edge happens before CKV_TDC rising edge. One can find that it is equivalent to SEL_EDGE==0 and TDC_Q(1)==0.

Thus we have another signal PHI_I as the TDC decoder output in Fig 2-1. It is high when and only when SEL_EDGE is low and TDC_Q(1) is low. (Notice that here the input for the TDC decoder is from the effective TDC chain with 32 TDC delay cells. The dummy cells before and after the effective chain are not fed to decoder input.)

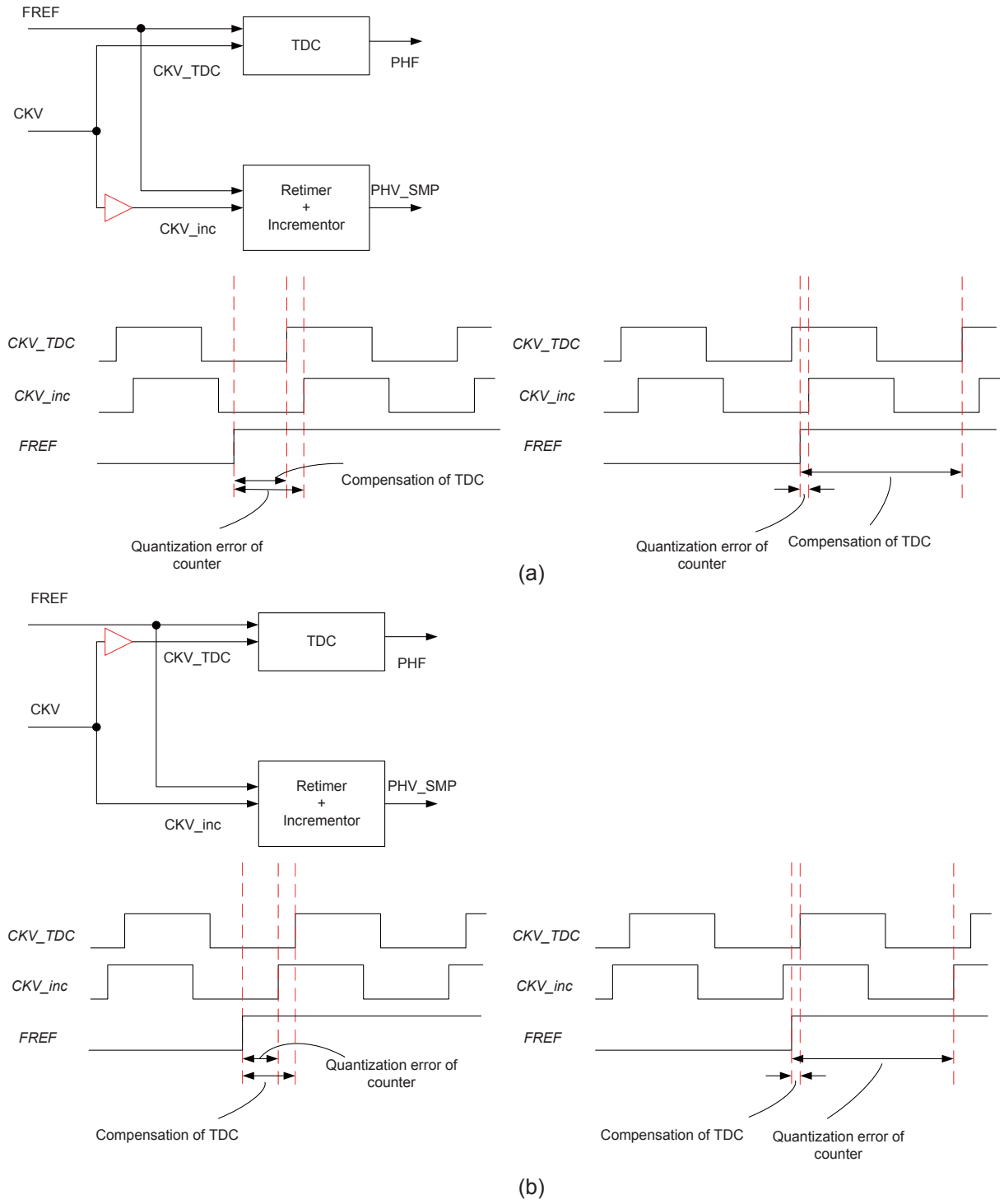

Figure 2-17: PHE spike due to mismatch between TDC path and incrementor path.

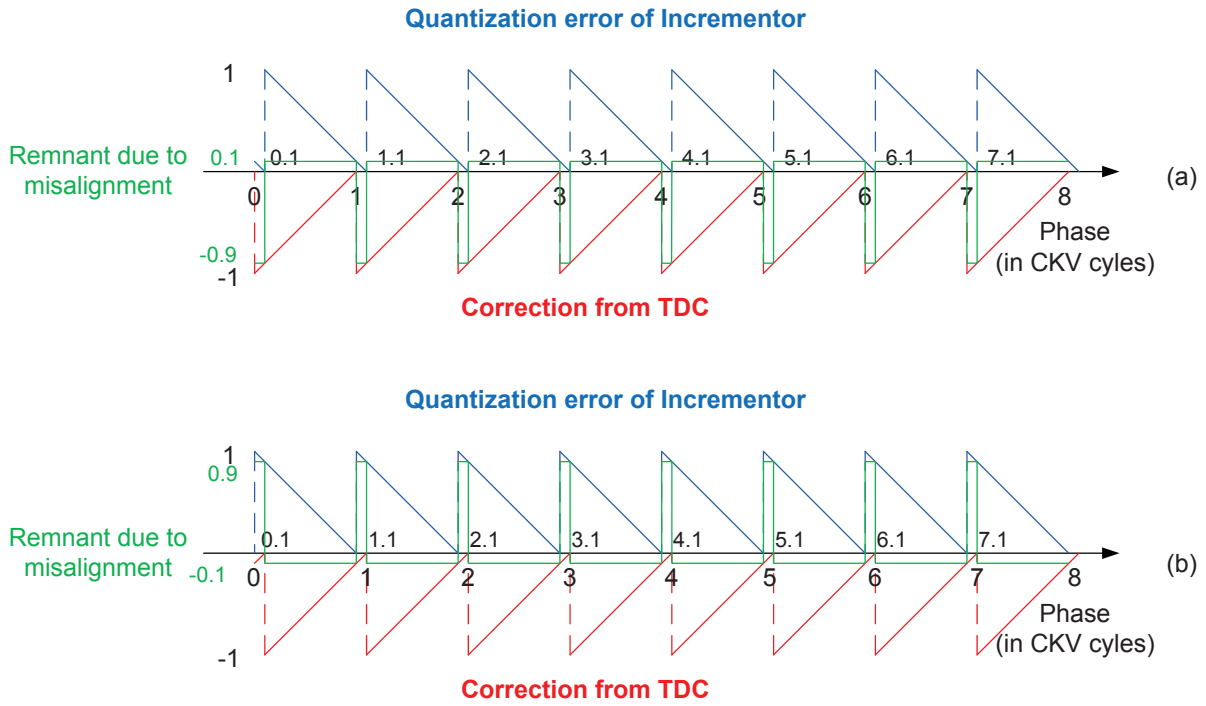

Figure 2-18: Generation of PHE spike.

That just corresponds to our previous assertion for PHV compensation. In that way the PHE spike is eliminated by only inspecting TDC output. The complex logic for spike detection is avoided and we can distinguish it from the sudden change of PHE due to frequency hopping or voltage/temperature change.

2-4 Digitally-Controlled RF Power Amplifier

In our ADPLL design, DPA works as the output buffer of ADPLL. It receives the ADPLL output and feeds them to off-chip measurement devices. Besides this purpose, we add some amplitude control signals to control the output power. This is to prepare for future implementation of the transmitter for WiMAX. There is no strict requirement for the resolution of this power control and also no hard limit for the efficiency of this DPA. However, the noise contributed by DPA shall be sufficient low to avoid overshadowing the phase noise of ADPLL output signal.

We have also a zoomed in bird view for DPA block as in Fig 2-21. The detailed design of DPA is given in Chapter 6.

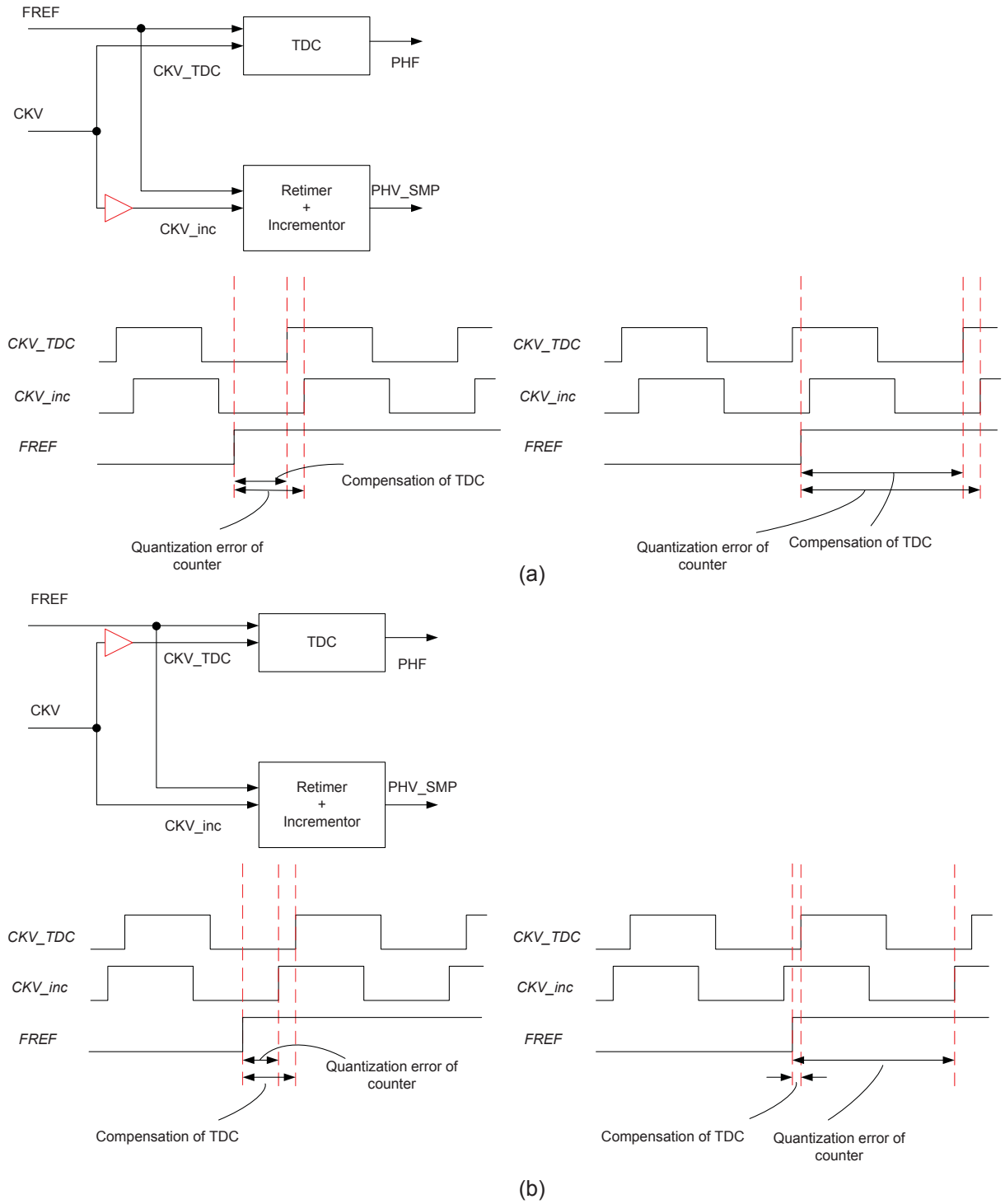

Figure 2-19: PHE Spike with SEL_EDGE signal in this design.

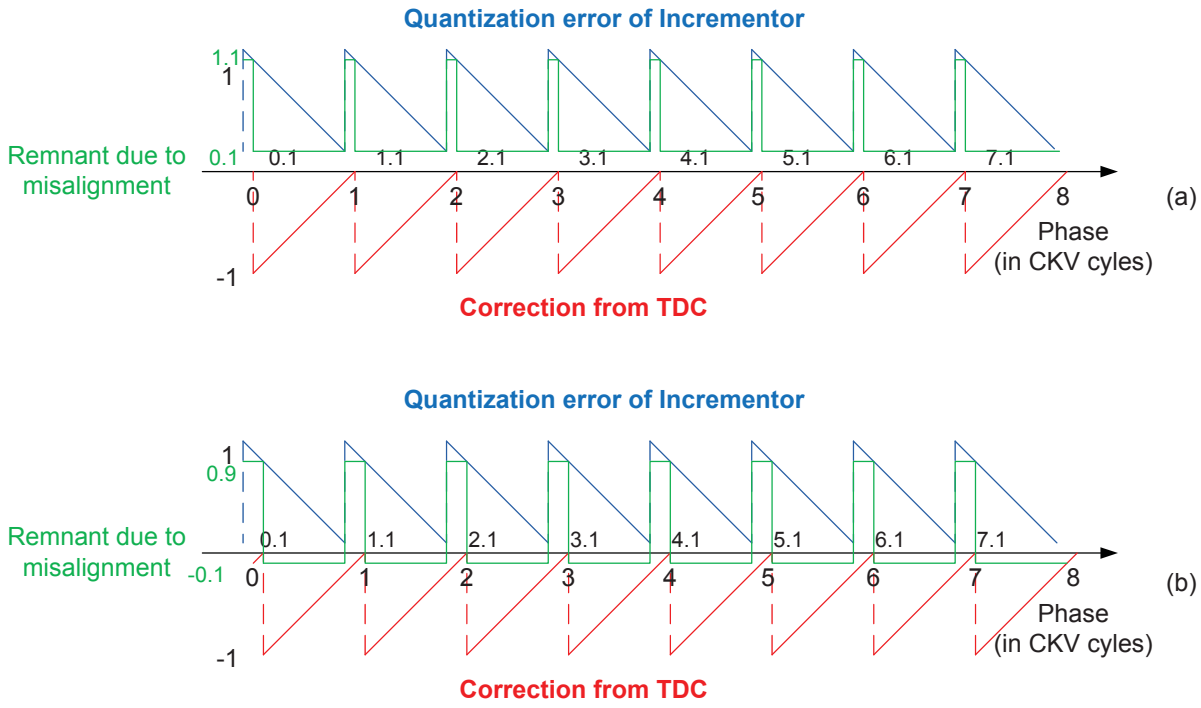

Figure 2-20: Generation of PHE spike in this design.

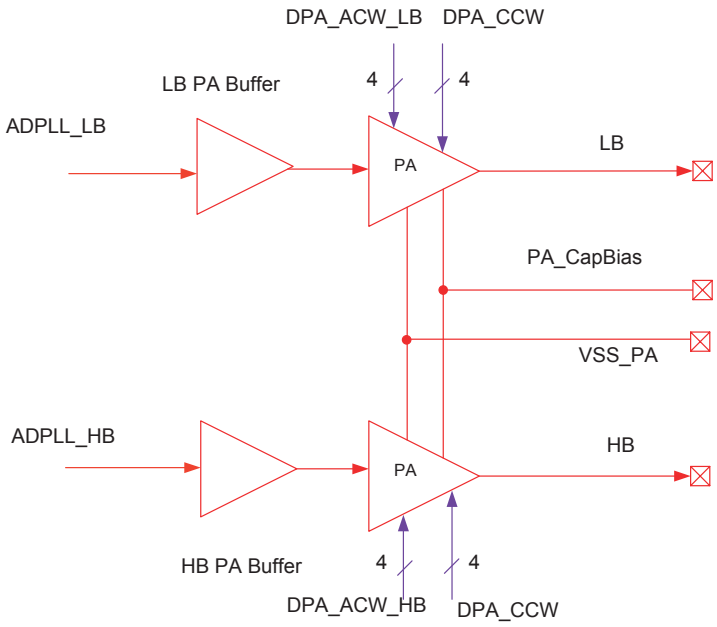

Figure 2-21: Zoom-in birdview of DPA.

2-5 Low Speed Digital Logic

As shown in Fig 2-22, outputs of TDC and the incrementor are fed to low speed digital logic. The logic first extracts the feedback phase information and does differentiation to obtain the feedback frequency. This is compared with desired frequency and the result is accumulated as phase error information PHE. PHE signal will be fed to individual loop filter for control of three capacitor banks (PB, AB and TB as in section 2-2) of DCO. Therefore DCO can be stabilized at desired frequency.

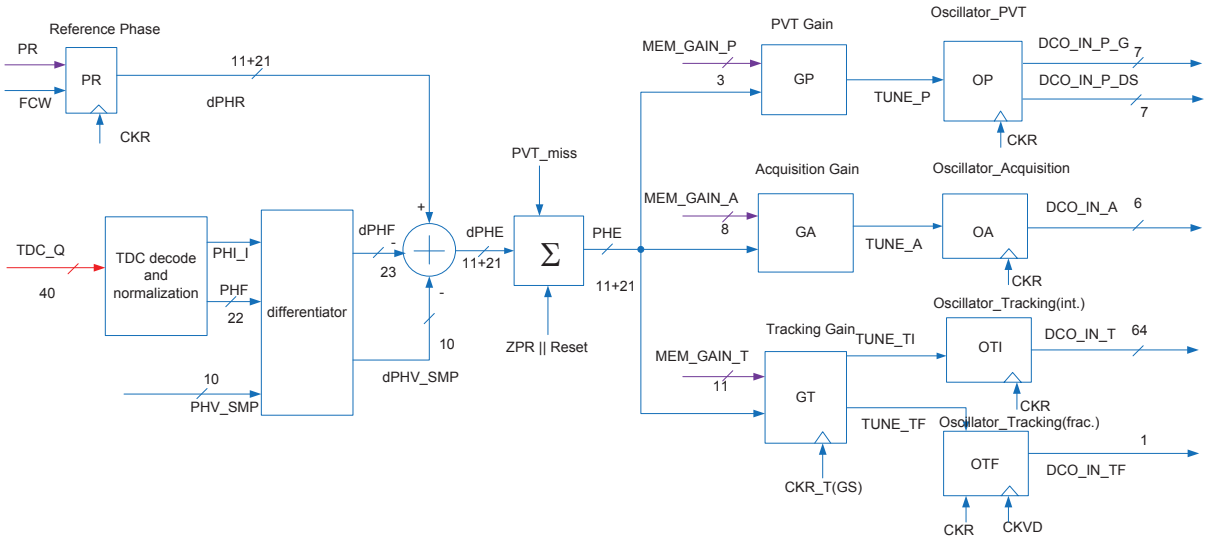

Figure 2-22: Zoom-in birdview of low speed digital logic block.

2-5-1 TDC Decoder and Normalization Block

Fig 2-23 shows the schematic of the TDC decoder and normalization block. The outputs of TDC core TDC_Q are Q(k) which are shown in Fig 2-10, where k is from 1 to 40. This information has to be decoded to TDC_RISE and TDC_FALL as in Fig 2-11. Notice that Q(1)–Q(4) and Q(37)–Q(40) are outputs of dummy cells and shall not be taken into account, thus the effective input has 32 bits. The decode logic is pretty straightforward: TDC_RISE corresponds to the first $1 \rightarrow 0$ transition and TDC_FALL corresponds to the first $0 \rightarrow 1$ transition. Fig 2-24 exemplifies how the TDC decode logic works.

For the low speed digital logic, we use CKR as its universal clock. As in Fig 2-12, CKR edge and FREF edge can be very close. On the other hand, the TDC output also changes when the FREF rising edge happens. It may lead to some issues if we want to sample the new TDC output and send that to the decoder at CKR rising edges.

There are two easy ways to work around that. One is to choose sampling TDC outputs at CKR falling edge. The other one is to implement TDC decoding logic as a latch. The

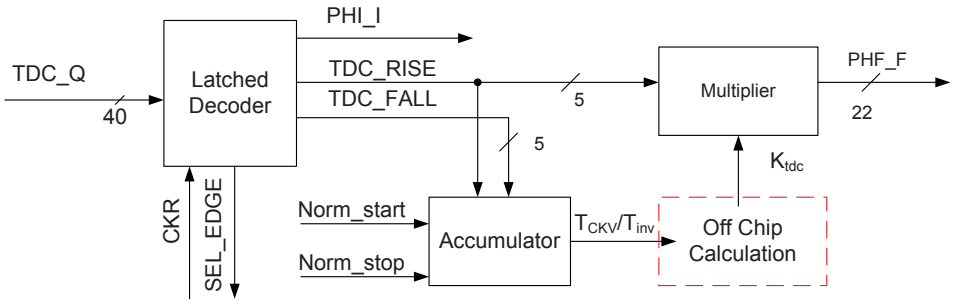

Figure 2-23: Schematic of the TDC decoder and normalization block.

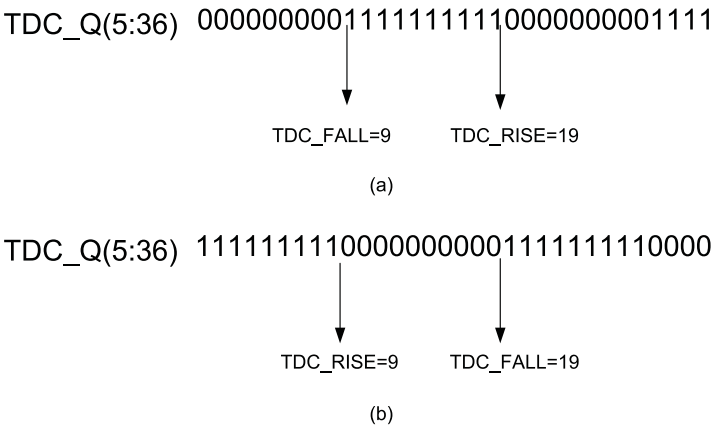

Figure 2-24: TDC decode logic illustration.

decoder always works as long as CKR is high. The benefit of this method is that, as we will see later, we have more time for the TDC normalization. With a latched decoder, the normalization of TDC_RISE can borrow some part of the time when CKR is high, as it doesn't take much time for the decoder to deliver the correct output. Therefore the timing constraint of normalization logic is relaxed and the power consumption can be reduced.

As mentioned in Equation 2-3, to normalize the time difference of edges indicated by TDC to the CKV cycle, we need to estimate K_{tdc} and multiply with k, i.e. TDC_RISE. Equation 2-4 shows that with the information of TDC_RISE and TDC_FALL, $\frac{T_{CKV}}{T_{inv}}$ can be estimated. To achieve an enough accuracy for this estimation, |TDC_FALL-TDC_RISE| is averaged over 256 cycles. Actually |TDC_FALL-TDC_RISE| is accumulated for 256 cycles and then shift by 7 bits concerning the proportional coefficient of 2 in Equation 2-4. The start and stop signal is provided by the sequencer, which we will discuss in Chapter 5.

However, division logic is needed to get K_{tdc} from $\frac{T_{CKV}}{T_{inv}}$. A generic divider can be quite complex and power hungry. Thus in this design we implement the division operation off-chip, i.e. the estimation for $\frac{T_{CKV}}{T_{inv}}$ is read out from register and then the calculated $\frac{T_{inv}}{T_{CKV}}$ is written into some register as the normalization coefficient. For the future ADPLL design with baseband, the *digital signal processor* (DSP) can be utilized to fulfill this division task.

Finally, a multiplier is needed to execute TDC_RISE $* K_{tdc}$ in real time. With a latched TDC decoder, we have actually more than half FREF cycle for this operation. Ideally, the output only has fractional bits. Yet chances are that the output may be more than 1 due to the inaccuracy of K_{tdc} and the fluctuation of TDC_RISE. Then one extra bit is needed for the output.

To exemplify how normalization and multiplication works in fixed-point logic, we assume that the summation of |TDC_FALL-TDC_RISE| over 256 cycles is 2740. Then this number is shifted by 7 bits and inversed to obtain $\frac{T_{inv}}{T_{CKV}}$. To provide enough resolution and cover dynamic range of K_{tdc}, $\frac{T_{inv}}{T_{CKV}}$ has LSB corresponding to 2^{-16} and MSB corresponding to 2^{-4}. The fixed-point representation for K_{tdc} is actually

$$\frac{1}{2740/2^7} \times 2^{16} = \frac{2^{23}}{2740} = 3062 \tag{2-7}$$

If TDC_RISE=5, as LSB of PHF is 2^{-21}, we have

$$\text{PHF} = \text{TDC\_RISE} * K_{tdc} \overset{fixed-point}{=} 5 * \frac{3062}{2^{16}} * 2^{21} = 5 * 3062 * 2^{21-16} = 489920 \tag{2-8}$$

which is about 0.234 for floating equivalent value.

2-5-2 Phase Detection Logic

The feedback phase information provided by PHV_SMP, PHF and PHI_I will be compared with the desired value in the phase detection logic. As mentioned above, actually the frequency error is detected and its accumulation gives the phase error.

As in Fig 2-22, we have two signals to control the accumulator. One signal, Reset||ZPR is to reset accumulator output to zero for reset and *zero phase restart* (ZPR) algorithm. We will discuss that in Section 4-1. Another signal, PVT_miss, is to freeze the accumulator output to avoid a wrong frequency detection result. This will be explained in Section 4-2.

This differentiation-then-accumulation way provides the flexibility to control the loop dynamic. However, an extra delay of 1 CKR cycles is introduced to the loop. Then the loop stability is hurt when loop gain is rather high. For TB operation, this is not a big issue because the loop gain is pretty small. However, for PB and AB, the proportional coefficient, α, shall not be made too large to make sure the loop is stable.

The 25 Hz frequency resolution and reference frequency of around 30 to 40 MHz set the fractional bit of FCW to be 21 bits. As for the integer part of FCW, sufficient number of bits is needed to avoid aliasing. Feedback frequency of 3.3-4.05 GHz corresponds to about 7 bits. Actually the integer part of FCW is 11 bits wide for an enough dynamic range margin[3].

This sets the word length of integer part and fractional part for signals in the digital logic. PHF has 22 bits as the extra bit is to avoid overflow. Its differentiated output dPHF has 23 bits to include the sign bit. The frequency detection result, dPHE, has 11 integer bits and 21 fractional bits. This also applies to the accumulated phase error result PHE.

2-5-3 Loop Filter for PVT and Acquisition Bank

In Fig 2-22, the phase error result from the accumulator is fed to three gain circuits separately for PB, AB and TB (GP, GA and GT). The output of gain circuit is sent to the oscillator control circuit (OP and OA correspond to GP and GA, OTI and OTF is for integer and fractional part of GT output respectively). Most of the system specifications are related to how the tracking bank loop filter is constructed, i.e. the algorithm in GT, OT and OTI. We will leave the discussion of these complex blocks to next subsection and start from the relatively simple loop filter for PB and AB.

In the gain circuits, GP and GA, the arithmetic shift and the multiplication are executed. The arithmetic shift is to implement proportional coefficients α_P and α_A in PVT and acquisition loop, as we will discuss in Section 3-4. The proportional coefficients are chosen as integer powers of 2, say $2^{-2}, 2^{-3}$, to simplify calculation. The output of the arithmetic shift, NTW, is just the ratio of the desired frequency change of DCO divided by reference frequency. From Table 2-1, every bank of DCO has its own frequency resolution (we call that $K_{dco,P}$, $K_{dco,A}$ and $K_{dco,T}$) and the resolution even changes with oscillating frequency. Then NTW has to be multiplied with the values $\frac{f_R}{\hat{K}_{dco,p}}$ and $\frac{f_R}{\hat{K}_{dco,p}}$ for DCO Gain Normalization in PVT bank and Aquisition bank, where $\hat{K}_{dco,p}$ and $\hat{K}_{dco,a}$ are respectively the estimation of CKV frequency resolution for PVT bank and acquisition bank.

[3]Say, if we want to use a reference clock with lower frequency. The loop shall still work properly.

The oscillator control circuits, OP and OA, are to control which mode in Fig 2-6 DCO is traversing. The descriptive schematic of OP and OA is shown in Fig 2-25. The topology of OP and OA block are almost the same, except for the PVT miss logic in OP block. We will ignore that for now and come back to this in Section 4-2.

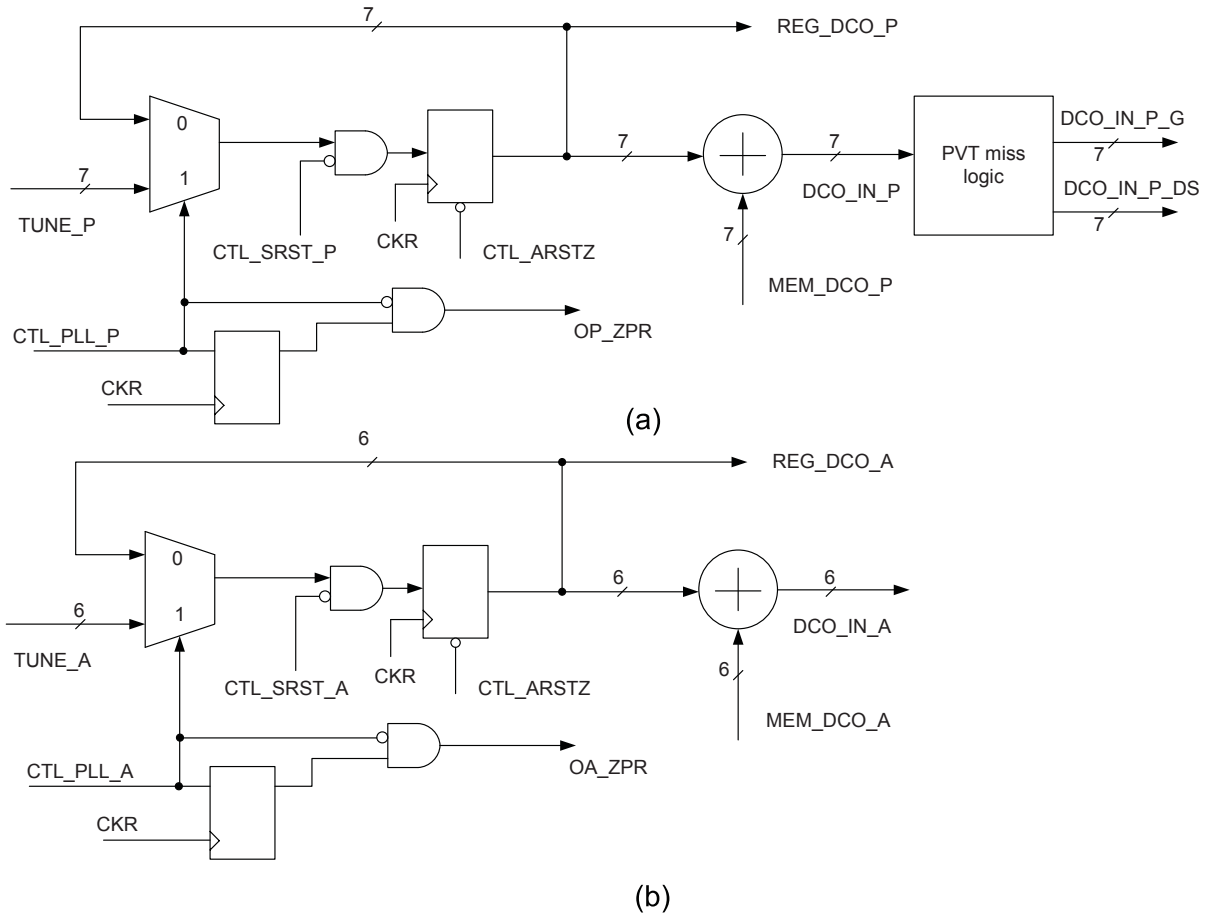

Figure 2-25: Schematic for OP and OA blocks.

The schematic of OP block shows quite a few control signals. CTL_PLL_P is to control whether PVT mode is active or already frozen (say, when DCO has advanced to acquisition or tracking mode). When CTL_PLL_P is high, the output of GP block TUNE_P is selected. When CTL_PLL_P is low, the tuning word at the last CKR cycle is again selected as the input, which means PVT tuning word is fixed and PVT mode is frozen. Thus CTL_PLL_P changes from high to low when we switch from PVT mode to acquisition mode. Then OP_ZPR will have a high pulse to activate the ZPR mechanism detailed in Section 4-1. CTL_SRST_P is a synchronous reset and CTL_ARSTZ is an asynchronous reset. MEM_DCO_P reads a start value for PVT bank tuning word to speed up the frequency settling process. The REG_DCO_P signal can be stored in some register for ADPLL test.

Notice that TUNE_P is a signed digital word and ranges from -2^6 to 2^6-1. However, the

number of capacitor unit turned on for PB ranges from 0 to $2^7 - 1$, which is an unsigned word. So we shall adds 2^6 to TUNE_P and that results in DCO_IN_P signal. This can be achieved by just reversing the signed bit (MSB) of TUNE_P. For example, TUNE_P is 0 and then DCO_IN_P is 64. That is the central capacitance value for DCO, as we expect.

2-5-4 Loop Filter for Tracking Bank

The effective processing of phase error signal in the loop filters for PVT and acquisition bank are only arithmetic shift and multiplication. No integration is included. This is called type-I PLL. The purpose of PVT mode and acquisition mode is to make sure that the CKV frequency can settle to the range of tracking bank within a rather short time, say a few microseconds. The type-I PLL is appropriate for the fast frequency settling. Nevertheless, for tracking bank, the phase noise and the settling time are both important considerations. As more constraints are added, the architecture of the loop filter for tracking bank is more complicated than that of PVT bank and acquisition bank.

Fig 2-26 is a high-level schematic for the GT block. The PHE signal has to pass through an IIR filter bank before the arithmetic shift operation for proportional gain. The IIR filter bank contains four IIR filters in series. Fig 2-27 shows the schematic of single IIR filter stage. The enable signal selects the output signal to be just the input signal or the IIR filter output so as to enable/disable this IIR filter. SRST will reset the filter output to be zero. The signal Lambda controls the coefficient of this IIR filter $2^{-\lambda}$.

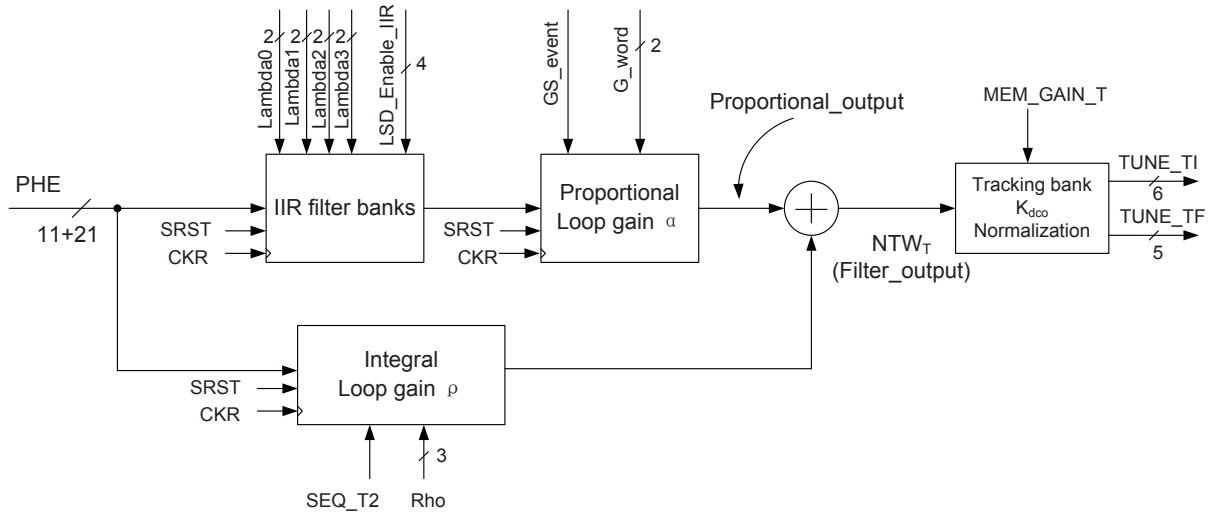

Figure 2-26: High-level schematic of the GT block.

The expression for y[k] when the IIR filter is enabled is

$$y[k] = (1 - 2^{-\lambda})y[k-1] + 2^{-\lambda}x[k] \tag{2-9}$$

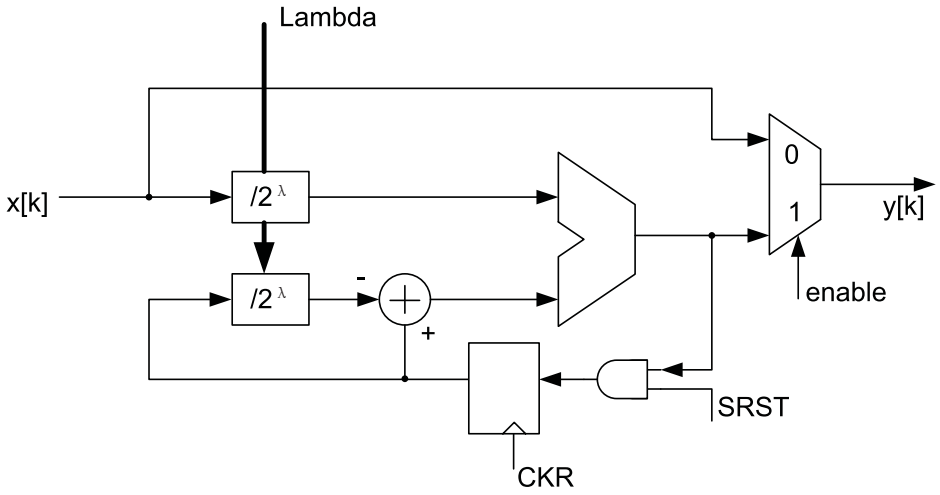

Figure 2-27: Schematic of a single IIR filter.

Similar to the analysis of Equation 3-10, we know that this filter has a corner frequency of $\frac{1}{2^{\lambda}2\pi} f_R$.

In Fig 2-26, the 4-bit LSD_Enable_IIR signal provides the enable signal to every IIR filter separately. The 2-bit Lambda0 signal is to choose λ of the first IIR filter from 0, 1, 2 and 3. The same function applies to Lambda1, Lambda2 and Lambda3. They together provide a great flexibility for the whole IIR filter bank.

The proportional coefficient α for tracking bank (α_T) is also realized as an arithmetic shift. α_T is of high importance to the phase noise of the ADPLL system. An intuitive observation is that when α_T is smaller, the effect of noise due to TDC and reference clock is reduced while the loop has weaker capability to correct for DCO frequency drift. Vice versa. α_T has to be chosen carefully for the tradeoff and some flexibility is desired. In this design α_T can range from 2^{-3} to 2^{-6} as controlled by 2-bit G_word signal.

We asserted above that larger α_T means stronger capability to correct for DCO frequency drift. It also means larger α_T can shorten the frequency settling time when ADPLL has entered the tracking mode. So the idea is to first adopt a larger α_T (say, 2^{-3} or 2^{-4}) and when the frequency is pretty close to desired frequency, α_T will be changed to smaller value (say, 2^{-5} or 2^{-6}). Therefore we can minimize the settling time while still maintain a good phase noise performance, which is not so easy to be done in traditional charge-pump PLL.

However, during the transition of α_T value, the control word for tracking bank shall not show an abrupt change, i.e. a smooth curve of NTW_T is desired. To achieve that, a gear-shifting circuit which is similar to the IIR filter is used. Its schematic is shown in Fig 2-28. Here we have a fixed arithmetic shift of 3 bits and an extra arithmetic shift which is controlled by 2-bit G_word signal. In total we can shift the input by 3 to 6 bits, corresponding to 2^{-3} to 2^{-6} as we mentioned before. Notice that the main difference of the gear-shifting circuit with the IIR filter is that the feedback flip flop is triggered by GS_event

signal rather than CKR signal. Thus the feedback value only changes when GS_event goes from low to high. In the sequencer we can synchronize the GS_event rising edge with the change of G_word. Then a smooth transition of α_T is achieved.

In the design practice, since it's easier for the tool to just use one clock domain for synthesis, one can use CKR as clock of flip flop. A detection logic for GS_event rising edge is to choose either to use the new flip flop output or the previous GS_FB signal (in Fig 2-28) as the updated value of GS_FB. As this change only needs some simple AND, NOT and MUX logic, the schematic is not drawn here.

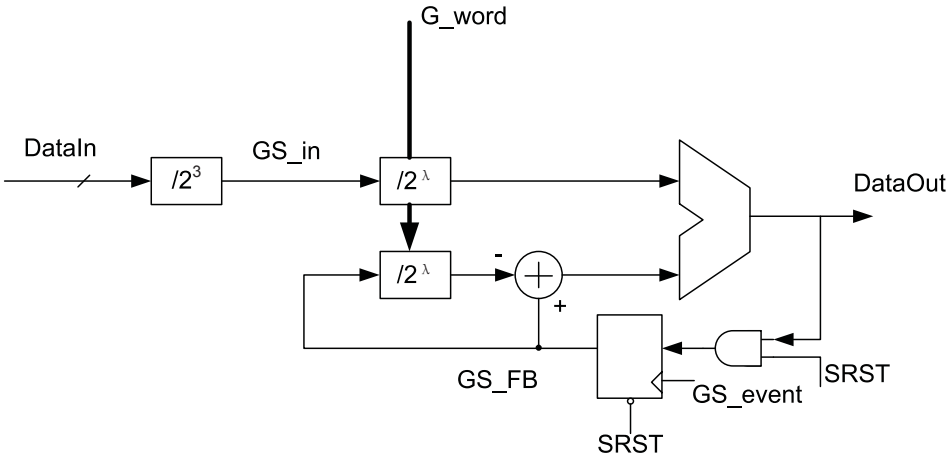

Figure 2-28: Schematic of proportional gain path for tracking bank.

Besides the proportional path, a path featuring integral loop gain is added in parallel. This path has an accumulator in it. When the path is enabled by the rising edge of SEQ_T2 signal, the ADPLL becomes a type-II PLL. We will see in Section 3-4 that type-II PLL can suppress in band phase noise contributed by up-converted flicker noise of DCO.

For a type-II PLL, the average value of PHE will be forced to zero for a fixed output frequency. However, before this integral path is activated, for the type-I PLL, in most situations, PHE has a non-zero average value. Thus it takes some time for PHE to settle to around zero at transition transient from type-I to type-II. To avoid this settling transient, before type transition starts, PHE is sampled as a residue for the integration. We call this residue $\phi[k_0]$. Then after integral path is turned on, the accumulated value is PHE$[k]-\phi[k_0]$ rather than PHE$[k]$.

Thus we need a residue latching circuit and an accumulator circuit for integration. The schematic is exactly the same as Fig 4.44 in [14] and is not redrawn here. For the integral loop gain ρ, similar to proportional gain α_T, it's implemented as an arithmetic shift to simplify the logic. The shift bit can range from 8 to 15, meaning that ρ can change from 2^{-8} to 2^{-15}. A fixed shift of 8 bits is firstly done. Then 3-bit Rho signal can select the extra shift bits(from 0 to 7). Therefore the value of ρ can also be variable. However, compared with α_T, ρ will not be changed during the procedure of frequency settling and no gear-shifting circuit is needed.

The proportional path and integral path output are summed together as NTW_T and that is multiplied with $\frac{f_R}{\hat{K}_{dco,T}}$ for DCO gain normalization of tracking bank, where $\hat{K}_{dco,T}$ is the estimation of the frequency resolution for CKV in tracking bank. The multiplier output, TUNE_T, is split into the integer part and the fractional part for tracking bank. TUNE_TI, the integer part, goes to the OTI block in Fig 2-22. OTI resembles OP and OA block with a few differences. Since the tracking mode is the main working mode of ADPLL, there is no ZPR generation circuit in OTI block. The 6-bit binary control signal is decoded to 64-bit thermometer code to control DCO TB, as shown in Fig 2-4. TUNE_TF, the fractional part, is first converted to an unsigned number DCO_TF and then fed to the 1st order $\Sigma\Delta$ modulator in Fig 2-5. Notice that since both TUNE_TI and TUNE_TF are converted to the unsigned value and stored for ADPLL test, we can merge OTI and OTF together as OT block and leave out high-speed modulator for DCO TFB as an individual block in the implementation stage, as shown in Fig 5-2.

Chapter 3

Modeling, Simulation and Analysis of ADPLL System

In Chapter 2 ADPLL system and the building blocks are presented to the detail. To assess the performance of the whole system, every block needs to be modeled or described at system level. Then a system-level simulation shall be done and data will be processed and analyzed. In this chapter, the modeling and simulation methodology of ADPLL is illustrated. Then we come up with model of critical blocks like DCO and TDC. After that, we go back to system analysis with insight from s-domain analysis and processed data from time-domain simulation.

3-1 Modeling and Simulation of ADPLL

Simulation of the whole PLL system is always a very challenging work. It's mostly due to the dynamic range of frequency in PLL. The oscillator is working at the frequency of several GHz or tens of GHz while the frequency accuracy requirement for the system is tens or hundreds of Hz. To get the response of the oscillator with enough accuracy, simulator like SPICE or SPECTRE will execute the simulation in step of *picosecond* (ps) or even *femtosecond* (fs). On the other hand, to characterize an accurate and smooth in-band phase noise the simulation needs to take several *milliseond* (ms). If we are to include the start-up transient for the LC tank as well as the decoupling network, and include parasitic from post-layout extraction, chances are the circuit will not converge and the server simulates for one month and then crashes.

So it's indispensable to have a sufficient abstraction for the detailed circuit and verify the whole PLL system with these abstractions. In the traditional charge-pump PLL design, this is called the behavioral level simulation. Various tools like Matlab, Verilog-A[18] or other

tools such as CppSim[19] are deployed for this job. In the ADPLL design the situation is different. As all essential blocks have digital interfaces, the *hardware description language* (HDL) can be used for the modeling of the whole loop for its great power of digital logic description. HDL with real-value modeling capability (allowing floating-point signals) is preferred. At the high abstraction level, using real-value signals is much more convenient and flexible than using fixed-point signals in simulation. Previous ADPLL designs have adopted VHDL as the simulation language[14]. However, ADPLL of this project will be the IP of Catena so it is nice to choose a language compatible with current design flow in the company. Since the company has chosen Verilog for digital logic description, here Verilog-AMS is chosen. Verilog-AMS is well compatible with Verilog and provides the real-value signal as type 'wreal'. Thus we can have very abstracted behavioral level real-valued model as well as almost near-synthesizable, fixed-point model in Verilog-AMS description. What's more, the Verilog-AMS/Verilog co-simulation in Cadence is seamless. Thus when NoB for signals has been decided, Verilog-AMS code of digital logic can be transferred to Verilog without special effort. Actually, the simulator of Verilog-AMS in Cadence can invoke SPECTRE for analog circuit simulation at the transistor level. However, to achieve that, analog-to-digital and digital-to-analog interface have to be set carefully or even specifically modeled. Also the simulation speed is slowed down. So, in the ADPLL system simulation within the scope of this thesis, analog blocks are simulated in the model of Verilog-AMS rather than SPECTRE.

Note that besides Verilog-AMS, Matlab is used at the very primary stage of system modeling in this project, as scripting in Matlab is easier. It can act as a fast way to verify the effectiveness of proposed algorithms. We will determine and verify the basic structure in Matlab and then shift to Verilog-AMS in Cadence, which is superior in digital logic description capability and closer to final implementation (can handle fixed-point calculation and can be easily synthesized).

The Verilog-AMS model in this project has 4 abstraction levels, like VHDL abstraction level in [14]. Level 1 code could use both wreal and integer type signals. The description is quite similar to the code in Matlab yet the system has much more clear division of each block due to the module based nature of Verilog-AMS. Level 2 code has fixed-point interface. The effective number of bits for signals in the simulation of level 1 is taken into consideration so that the truncation and overflow (rounding in the digital domain) are to be avoided in the normal case. Level 3 will be the fully synthesizable RTL code. As nowadays synthesis tools are very powerful, sometimes level 2 code can be just used for synthesis, i.e. level 2 and level 3 are just the same code. When level 3 code is done, timing constraint is specified and standard cell library is ready, synthesis tool will automatically[1] deliver a gate-level netlist with back annotated timing information. Thus a simulation with much more precise timing can be done. From this we can know that modeling of analog parts will just stop at level 2.

Despite the benefits offered by Verilog-AMS, it has some disadvantages, most of them

[1] With instructions, of course.

Table 3-1: Verilog-AMS abstraction levels.

Level1	Model with wreal and integer type signals. Behavioral model.
Level2	Modules with fixed-point interfaces. Truncation and rounding are considered.
Level3	Fully synthesizable RTL code.
Level4	Gate-level netlist with back annotated timing information.

from the old Verilog standard. There are two disadvantages that are worth notice in our modeling:

1. The word length of integer signal in Verilog-AMS is 32 bits. It may not be sufficient to cover the dynamic range of the frequency in PLL, say the ratio of the highest frequency in PLL model (usually pertained to signals in oscillator) and the lowest frequency in PLL model(can be frequency accuracy for PLL), or some internal signals in the digital logic. In that case, even for level 1 code, a fixed-point representation rather than integer number is needed.

2. The minimum time accuracy of Verilog-AMS simulation is 1 fs. One may argue that 1 fs is a very small time interval even for oscillator. However, notice that

$$\frac{\Delta f}{f} = \frac{\Delta t}{t} \tag{3-1}$$

   One can easily figure out that for 1 GHz signal, a deviation of 1 fs could lead to frequency deviation of 1 kHz. So this accuracy needs to be taken care of in the model.

Besides that, Verilog-AMS is not compatible with the up-to-date SystemVerilog language. Although there is an effort to drive the merge of SystemVerilog and Verilog-AMS, a.k.a. SystemVerilogAMS[20], it does take some time to get the support from main EDA vendors.

Table 3-2 is a list of pros and cons for Verilog-AMS language as used in modeling.

This thesis will not explain Verilog-AMS in detail. The modeling methodology for every block shown below can, in principle, be applicable to other languages. Still, there are some practical issues specifically with Verilog-AMS in this project and they will be discussed within certain context.

3-2 DCO Modeling

3-2-1 Time-Domain Modeling of DCO Phase Noise

We have made a default assertion that the phase noise of DCO is self-contained in the rising edge transition timing information of its output signals. Here we will show the DCO

Table 3-2: Pros and Cons of Verilog-AMS.

Pros	***Cons***
Fully compatible with Verilog	Incompatible with SystemVerilog yet
Support mixed-signal simulation with SPICE/SPECTRE as analog simulator	Need to define analog/digital interface carefully
Support co-simulation with VHDL, VHDL-AMS and etc.	
Event driven simulation mechanism to avoid unnecessary oversampling	Limited timing accuracy to 1 fs
Digital design flow friendly	

phase noise can be modeled in time domain and with that, we will go to a complete DCO model in Verilog-AMS given later.

Measurements have shown the continuous phase-noise spectra of oscillators can be approximated by[5]:

$$W_\Phi(f) \approx \frac{h_4}{f^4} + \frac{h_3}{f^3} + \frac{h_2}{f^2} + \frac{h_1}{f} + h_0 \qquad \text{rad}^2/\text{Hz} \tag{3-2}$$

where f is the frequency offset from the carrier and h_n (n=0,1,2,3,4) are certain coefficients.

From this equation, a complete asymptote of the phase noise shall contain five segment straight-lines with slope of -40 dB/decade, -30 dB/decade, -20 dB/decade, -10 dB/decade and 0 dB/decade in the log-log scale plot[2]. Yet for the oscillator in PLL, usually the terms of $\frac{h_3}{f^3}$, $\frac{h_2}{f^2}$ and h_0 are of concern. From the classical Leeson model[21], the first term is due to up-conversion of the flicker noise inside the oscillator and the second term is due to up-conversion of the white noise inside the oscillator. A simple illustration is that the noise will add to the time-domain uncertainty of the zero-crossing point (for differential output as in Fig 2-4, that would be the time $V(outp) = V(outn)$). However, as the oscillator have no memory about what happened in the previous cycle, these uncertainties would add up, which is just an integrator effect and adds to an additional $1/s$ transfer function in S domain[3]. This effect as reflected in the power spectrum is an extra -20 dB/dec slope.

The 0 dB/dec phase noise is mostly due to the contribution of dividers/buffers. In contrast with oscillator, dividers/buffers receive the input signal with clear timestamp of edge transition. Although the noise also causes time uncertainty for zero-crossing points, this doesn't accumulate. The close-in flicker noise is usually buried in the up-conversion phase

[2]We will use dB/dec in short of dB/decade. Notice another terminology of dB/octave, or in short dB/oct, is also widely used. The translation between the two slopes is very simple: -10 dB/dec = -3 dB/oct.

[3]One may argue that the integrator effect is kind of a memory. However, that means oscillator only has a memory about the edge of the last cycle yet no memory about what exactly happened in the last cycle.

noise caused by the oscillator and only the white noise shows up at offset frequency that's far enough.

Thus the flat 0 dB/decade phase noise is modeled as non-accumulative jitter, while the -20 dB/decade phase noise and -30 dB/decade are modeled as accumulative wander[14].

For the non-accumulative jitter of 0 dB/dec, assuming the input signal are ideal oscillating signal with kth positive edge exactly at kT_0, where T_0 is the ideal oscillation cycle time. Then the timestamp of divider/buffer output is:

$$t_j[k] = kT_0 + \Delta t_j[k] \tag{3-3}$$

$\Delta t_j[k]$ is a zero-mean random value with Gaussian distribution and the standard deviation is:

$$\sigma_{\Delta t_j} = \frac{T_0}{2\pi}\sqrt{\mathcal{L} f_0} \tag{3-4}$$

where f_0 is the ideal oscillation frequency and $\mathcal{L}$ is the phase noise as defined in subsection 1-2-3.

It shall be noted here for the 0 dB/dec phase noise, a.k.a. the noise floor, the magnitude of $\mathcal{L}$ depends on the sampling rate. When only the rising edge information is considered, as in our case, the noise is sampled at a rate of f_0 and folded within the frequency range of $-\frac{f_0}{2} \sim \frac{f_0}{2}$. Nevertheless, in the spectrum analyzer, both the rising edges and falling edges are captured. Then the noise is folded to the frequency range of $-f_0 \sim f_0$. Thus the magnitude of the noise floor shall be 3 dB lower than the case with only the rising edge information. The spectre simulator may also sample at both the rising edges and the falling edges, which needs to be verified. Therefore one needs to be aware of whether to add (or minus) 3 dB for the simulation result of spectre and the measurement result of the spectrum analyzer when executing the simulation.

Say the phase noise for the 0 dB/dec segment is $\mathcal{L} = -157$ dBc/Hz, the desired frequency is $f_0 = 3.65$ GHz, and the period is $T_0 = 274$ ps, $\sigma_{\Delta t_j}$ would be about 37 fs.

As $\Delta t_j[k]$ can be either positive or negative, the edge transition of the divider/buffer output may happen before the edge event of the input signal, which causes a difficulty for the modeling in event-driven simulator like VHDL/Verilog-AMS. Thus in this design $\Delta t_j[k]$ is precalculated and we merge that jitter into the period information as:

$$p[k] = T_0 + \Delta t_j[k] - \Delta t_j[k-1] \tag{3-5}$$

where p[k] is the time of period for the kth cycle. We will illustrate how to build up the DCO model using period information later.

For the accumulative wander corresponding to the -20 dB/dec segment in the phase noise profile, from previous discussion on its origin, one can infer that it shall be modeled as:

$$p[k] = T_0 + \Delta t_{ww}[k] \tag{3-6}$$

$\Delta t_{ww}[k]$ is also a zero-mean random value with Gaussian distribution and its standard deviation is:

$$\sigma_{\Delta t_{ww}} = \frac{\Delta f}{f_0}\sqrt{T_0}\sqrt{\mathcal{L}(\Delta f)} \tag{3-7}$$

Since the standard deviation is proportional to $\Delta f\sqrt{\mathcal{L}(\Delta f)}$, as long as the -20 dB/dec slope holds, it doesn't matter which frequency offset (Δf) you exactly choose in the phase noise profile for the calculation.

Say at the offset frequency $\Delta f = 10$ MHz, we have $\mathcal{L}(\Delta f) = -140$ dBc/Hz, and the desired frequency $f_0 = 3.65$ GHz, the period $T_0 = 274$ ps, the standard deviation $\sigma_{\Delta t_{ww}}$ is about 4.54 fs.

For the accumulative wander corresponding to the -30 dB/dec segment in the phase noise profile, we have a similar equation

$$p[k] = T_0 + \Delta t_{wf}[k] \tag{3-8}$$

the difference from the modeling of up-converted white noise is that $\Delta t_{wf}[k]$ is caused by the 1/f noise rather than the white noise.

The construction of the 1/f noise is achieved by passing a white noise through several first-order low-pass filters. As shown in Fig 3-1, when the envelope formed by the DC gain and the corner frequency of these filters, say the line composed by the points $(A_1, f_{c,1}), (A_2, f_{c,2}), (A_3, f_{c,3}), \ldots, (A_n, f_{c,n})$ has a slope of -10 dB/dec, the composite filter will approaches a response with -10 dB/dec.

When the ratio of corner frequency for the adjacent filter $r = \frac{f_{c,k+1}}{f_{c,k}}$ is 10, which is usually good enough to approximate this -10 dB/dec profile, the ratio of the DC gain would be

$$A = \frac{A_k}{A_{k+1}} = 10^{10dB/20} \tag{3-9}$$

Then the k$^{\text{th}}$ filter is modeled as a first-order IIR filter with the equation

$$y_k[i] = (1 - a_k)y_k[i-1] + a_k A^{-(k-1)}x[i] \tag{3-10}$$

This translated to z-domain, we have the transfer function

$$H(z) = \frac{a_k A^{-(k-1)} z}{z - (1 - a_k)} \tag{3-11}$$

The DC gain can be known as $A_k = A^{-(k-1)}$. For the corner frequency, as $z = e^{j2\pi f/f_s}$, where f_s is the sampling frequency, when $f \ll f_s$, we have $z \approx 1 + j2\pi f/f_s$. Then the denominator of equation 3-11 will be $a_k + j2\pi f/f_s$. For corner frequency of $f_{c,k}$, we have

$$a_k = 2\pi\frac{f_{c,k}}{f_s} \tag{3-12}$$

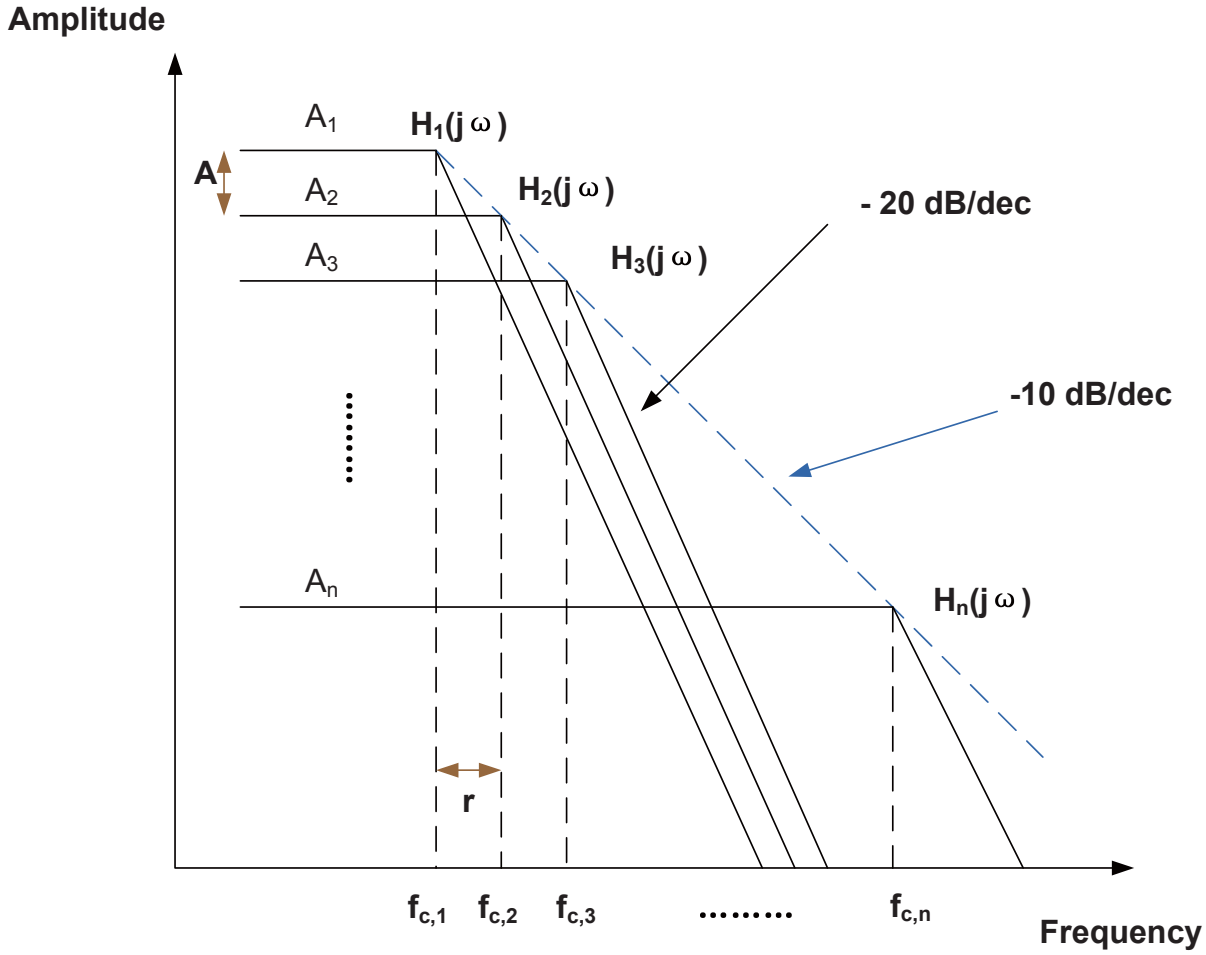

Figure 3-1: Composition of flicker noise using multiple low-pass filters (Log-Log Scale).

In this design, five filters with respective corner frequency of 100 Hz, 1 kHz, 10 kHz, 100 kHz and 1 MHz are chosen. From DCO simulation results, the up-converted flicker noise is overshadowed by the up-converted white noise when the offset frequency is beyond 1 MHz. Thus the highest corner frequency is just 1 MHz. All the filters are sampled at the same rate, which is 1/400 of DCO core oscillating frequency. Say if the DCO core works at 14 GHz, the sampling rate is 35 MHz. Then the combined output will be oversampled by the DCO core clock to avoid aliasing. This modeling procedure is illustrated in Fig 3-2. The anti-aliasing filter in the figure can be done with oversampling, or more specific, the zero-order hold interpolation.

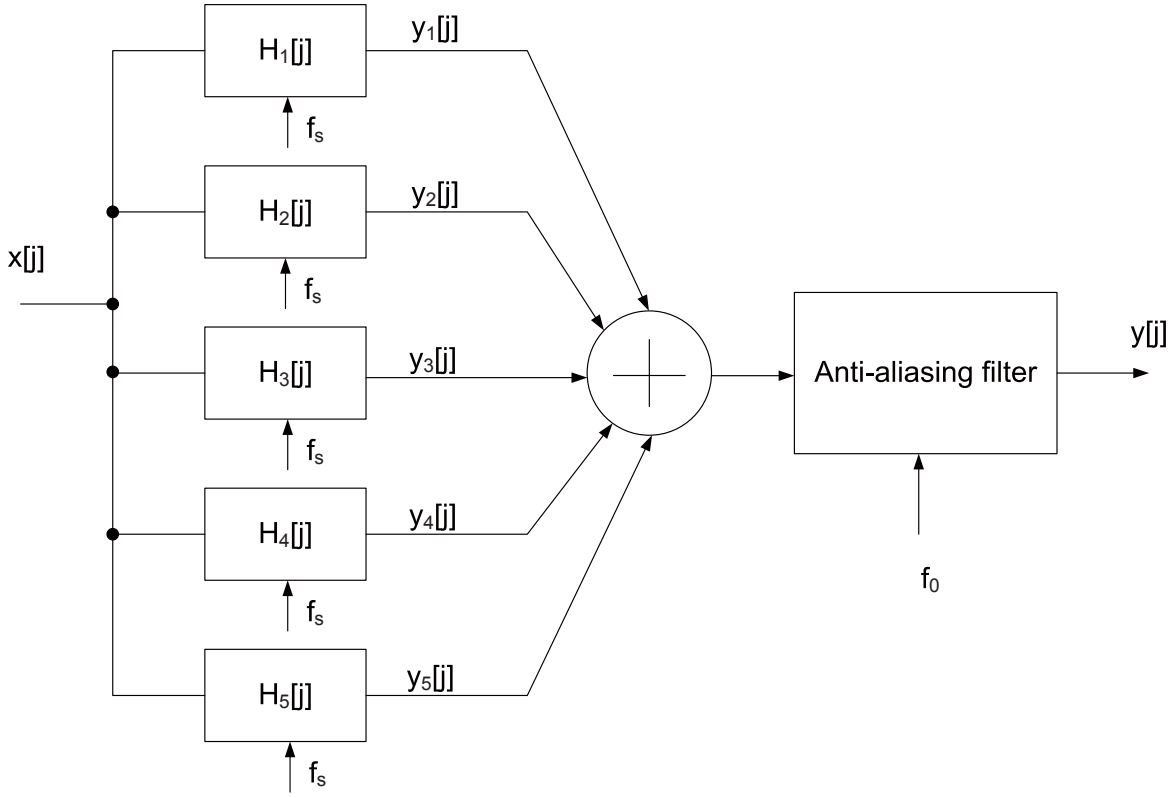

Figure 3-2: Construction of flicker noise with single sampling clock and oversampling.

Still we need to know the standard deviation of the input white noise x[j] for the parallel filter banks. In [14], an estimated value has been proposed as

$$\sigma_{\Delta T,1/f} = \frac{\Delta f_{c,1}}{f_0}\sqrt{T_0}\sqrt{2\mathcal{L}(\Delta\omega_{c,1})} \tag{3-13}$$

and some coefficient (5.5 dB) to compensate for the correlation is given. However this value is quite mysterious and cannot be used in our modeling. So a more straight forward method is applied here to obtain the desired compensation coefficient.

From the phase noise profile, one can easily observe the region where the up-converted flicker noise dominates and the region where the up-converted white noise dominates. The

extrapolation of the two segments intersect at a certain frequency f_{corner}. If we plot the power spectrum of $\Delta t_{ww}[k]$ and $\Delta t_{wf}[k]$, they shall also just intersect exactly at this frequency. As the standard deviation of $\Delta t_{ww}[k]$ ($\sigma_{\Delta t_{ww}}$) can be calculated from Equation 3-7, it is easy to construct a white noise with this standard deviation in Matlab. With another uncorrelated white noise as x[j] in Fig 3-2, a flicker noise can also be generated. Then we can adjust the standard deviation of x[j] ($\sigma_{x[j]}$) so that its power spectrum intersect the power spectrum of $\Delta t_{ww}[k]$ at f_{corner}. From that, we just fix the compensation coefficient needed. This method can achieve the accuracy of 1 dB, mainly constrained by the simulation time and the memory needed.

Fig 3-3 illustrates how the coefficient is estimated. Assume that the phase noise at 100 Hz frequency offset (the -30 dB/dec segment) is -12.3 dBc/Hz while at 10 MHz frequency offset (the -20 dB/dec segment) is -140.9 dBc/Hz, the schematic says the corner is about 79 kHz. The extropolation of -30 dB/dec segment and -20 dB/dec at this corner frequency is respectively -99.2 dBc/Hz and -98.9 dBc/Hz. These two values are rather close, which means the compensation coefficient is estimated very well. The flicker noise profile also reveals the sinc filtering effect due to the oversampling operation.

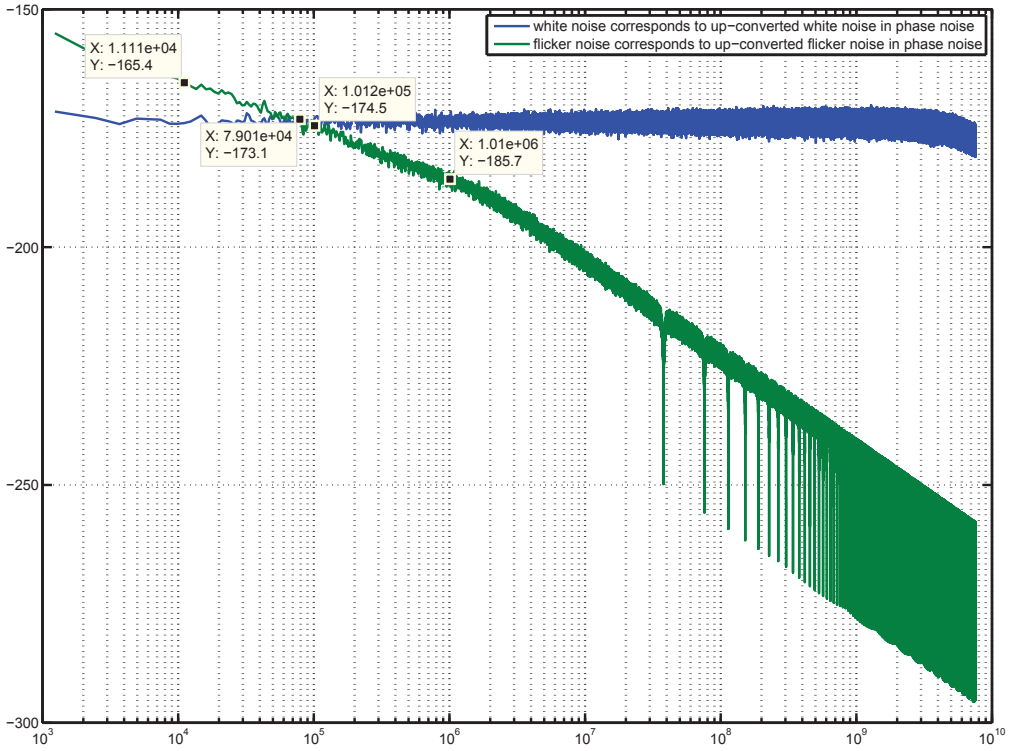

Figure 3-3: A schematic illustrating the estimation of flicker noise compensation coefficient.

3-2-2 The Effect of Ideal Divider

Quite often the phase noise of signals are not measured or simulated directly but done after the signal has been fed to some dividers. On one hand, as seen in Fig 2-2, dividers are usually deployed with the oscillator. On the other hand, in the measurement it's easier to analyze the phase noise of the divided signal as the frequency is lower and in simulation we need to consider the contribution of dividers to the phase noise of the whole block as well.

While in the modeling of DCO, it is convenient to have ideal dividers/buffers with an oscillator which contributes all the noise. We have seen that the period information of oscillator can model the 0 dB/dec, -20 dB/dec and -30 dB/dec phase noise. The modeling of ideal dividers and buffers are pretty trivial in the event-driven simulator. The required values ($\sigma_{\Delta t_j}$,$\sigma_{\Delta t_{ww}}$ and $\sigma_{x[j]}$) for the divided clock can be easily obtained via inspecting the measurement or simulation output. Thus the job is just to infer the corresponding values when we refer these noises to the oscillator core.

For the signals before and after ideal divider as in Fig 3-4, the non-accumulative jitter due to Δt_j and the accumulative wander due to Δt_{ww} are treated in [14] and we have

$$\sigma_{\Delta t_{j,0}} = \sigma_{\Delta t_{j,1}} \tag{3-14}$$

$$\sigma_{\Delta t_{ww,0}} = \frac{1}{\sqrt{N}}\sigma_{\Delta t_{ww,1}} \tag{3-15}$$

However, the accumulative wander due to Δt_{wf} is not mentioned in [14]. A good estimation is that

$$\sigma_{x[j],0} = \frac{1}{N}\sigma_{x[j],1} \tag{3-16}$$

That is because:

1. It's obvious from Equation 3-11 that $\Delta t_{ww}[k]$ and $\Delta t_{ww}[k+1]$ has a very strong correlation. Even without oversampling, some trials in Matlab show that Equation 3-16 is a good estimation.

2. Actually the flicker noise is constructed via the zero-order hold oversampling. Say the oversampling rate is 400, then after the ideal divide-by-4 divider, the only difference is that the oversampling rate has dropped to 100. Notice the bins in the zero-order hold window share the same value of Δt_{ww} so the accumulative wander of four cycles is just four times the wander value in one cycle. Thus Equation 3-16 holds.

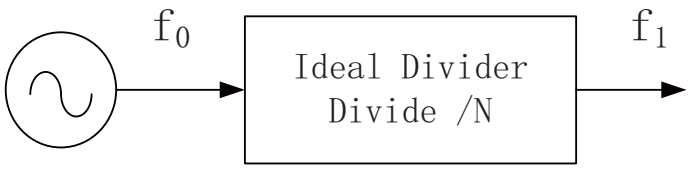

Figure 3-4: Oscillator with an ideal divider.

3-2-3 Essential Verilog-AMS Code for the DCO Modeling

The code of the DCO block is given in Section A-1. The instantaneous period is calculated at line 266. Ideally the period shall be $2\pi\sqrt{LC}$. The inductance value DCO_L is fixed and the instantaneous capacitance value DCO_Cinst are controlled by tuning words as modeled at line 262. tpp and pretpp model the non-accumulative jitter as Δt_j. tdev1 models accumulative wander corresponding to -20 dB/dec segment as Δt_{ww} and tdev2 models accumulative wander corresponding to -30 dB/dec segment as Δt_{wf}. The up-converted flicker noise Δt_{wf} is modeled from line 231 to line 248 exactly as we discuss before.

At line 268 and line 282, for every half cycle the differential output DCOcorep and DCOcoren of the DCO core toggle between high and low, i.e. a representation of DCO core oscillation in the digital domain The divide-by-2 division is modeled from line 295 to line 302. The divide-by-3 division is modeled in line 276 to 280 and line 290 to 294. The divide-by-4 quadrature output Div4_Ip, Div4_Qp, Div4_In and Div4_Qn are assigned as the feedback CKV to phase rotator input from line 153 to 156. Line 222-228 model the enable signals for dividers and buffers.

We shall notice that the ideal period $2\pi\sqrt{LC}$ at line 262 has an infinite resolution. However, as we mentioned in Section 3-1, Verilog-AMS has a limited time resolution of 1 fs. So the delivered frequency in the DCO model may deviate from the frequency in the real situation by several kHz. This results in a fake violation of the frequency resolution specification of 25 Hz in Table 1-1. The solution here is to model the jitter and wanders of DCO with the resolution of 1 attosecond (as). The jitter and wanders are to perturb the ideal period with the unit of 1 as. Then the resolution of the average delivered frequency in the model is not limited by the 1 fs unit from Verilog-AMS.

Note such solution has some premises:

1. The rms value of the jitter and the wanders shall be sufficient larger than 1 fs. If the oscillator is too clean, the time domain noise from the jitter and the wanders are not enough to perturb the ideal period. Then the frequency resolution of the delivered frequency is still not well characterized. Here in our design the jitter from the noise floor is 37 fs, which is far enough for this purpose.

2. The resolution of 1 as can still leads to a fake frequency deviation of a few Hz. So

if the requirement of the frequency resolution is even finer than that, we shall assign higher resolution to the jitter and the wanders.

3-3 TDC Modeling

To describe TDC as in the whole system, we need to first determine what factors in TDC performance are important and cannot be ignored.

Let's first consider the time-domain noise of TDC[15]. Just similar to ADC, which has the quantization noise in the voltage domain, TDC has quantization noise in the time domain. The noise power is

$$\sigma_q^2 = \frac{T_{inv}^2}{12} \tag{3-17}$$

Besides that, we have the jitter due to noise in the inverter and comparator. As stated from the transistor-level simulation of TDC[15], the rms jitter is no more than 500 fs. Compared to $min(T_{inv}) = 10.5$ps, the jitter adds less than 0.25 dB to the whole time-domain noise. So such jitter can be ignored in the modeling and we only need to care about the quantization effect of TDC.

Then the mismatch between the delay of every cell is considered, which is mainly caused by the layout imperfections. That leads to the non-linearity of TDC transfer curve, just like INL and DNL in ADC. This non-linearity causes the spurs in ADPLL output spectrum, which will be shown in Chapter 4. Therefore it's important to model the TDC non-linearity in the model.

These considerations result in the Verilog-AMS code for the model of TDC presented in section A-2. The TDC is decomposed into 40 cells. As in listing A.2, the cell is just an inverter with a specified delay and an ideal flip flop. In listing A.3, 40 cells are connected. The delays cells are normalized to the average TDC delay and the normalized ratios are shown in the lines with 'localparam'. Then by modifying the value of ratio in those lines, the mismatch can mimic the transistor-level simulation result.

3-4 From System Specifications to ADPLL Implementation

3-4-1 Noise and Error Sources In ADPLL

As we look at the whole ADPLL system, the digital logic will not give rise to any noise or error as long as the estimation of K_{tdc} and K_{dco} is sufficiently accurate. There are mainly four sources for the phase noise of ADPLL:

1. The uncorrected DCO phase noise. As we will see in s-domain analysis, ADPLL can correct for the phase noise and the frequency drift of DCO that is within the bandwidth. For the phase noise at rather far-away frequency offset region, ADPLL cannot track and correct for that. Therefore it is still inherited in ADPLL output.

   For the description of the DCO phase noise, three parameters are needed, as discussed in section 2-2: the phase noise of one frequency offset located in the -30 dB/dec segment, the phase noise of another frequency offset located in the -20 dB/dec segment and the value of the white noise. In s-domain, the three segments are extrapolated and added together as an estimation of DCO phase noise profile. Say we have the DCO phase noise simulation result with -72.3 dBc/Hz at 10 kHz offset, -98.9 dBc/Hz at 100 kHz offset, -140.9 dBc/Hz at 10 MHz offset and -157 dBc/Hz for the noise floor. In the modeling we assume that the DCO phase noise profile is fully in the -30 dB/dec segment for 10 kHz offset and fully in the -20 dB/dec segment for 10 MHz offset. Then the extrapolated s-domain result for the DCO phase noise is shown in Fig 3-5. One can see that it differs from the simulation result at 10 kHz offset and 100 kHz offset for less than 1dB. Thus the assumption is not 100% true yet the deviation is quite small and can be ignored.

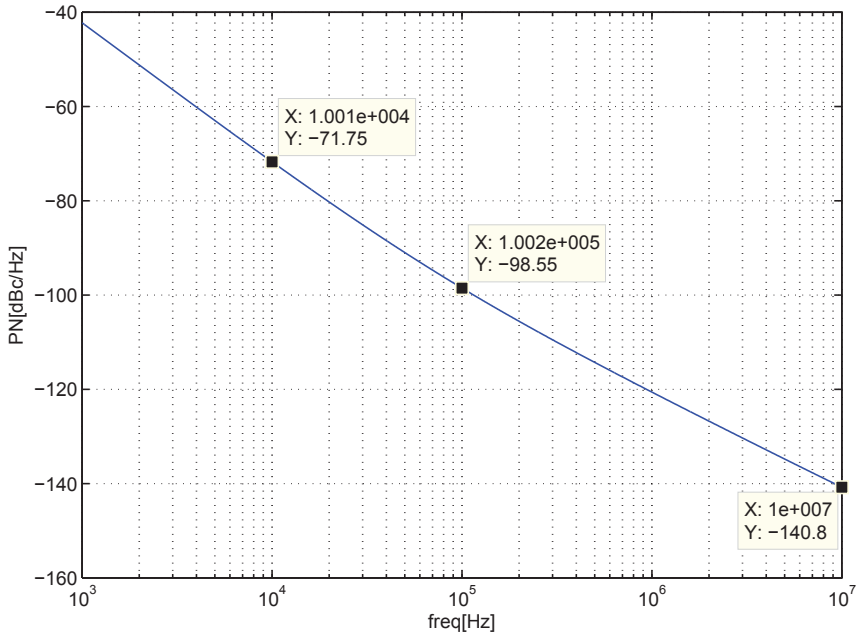

Figure 3-5: S-domain simulation result for DCO phase noise profile.

2. The inaccuracy of reference clock. In the previous discussion we assume that the reference clock FREF is a very accurate frequency source and doesn't show the phase noise itself. The truth is that for commercial products, cost factor has to be taken into consideration and the reference clock source, e.g. crystal oscillator, will always have a

not-so-low phase noise. This also adds to the in-band phase noise of ADPLL, with a multiplicative ratio of FCW since the desired phase is FCW times the reference phase. The situation will become severe for the high performance application, especially in the OFDM system, as it requires a very low in-band phase noise.

The dominating component in the phase noise profile of the reference clock is the white noise. Therefore the phase noise of the reference clock is pretty easy to model and only needs one parameter *RMS jitter*. Say we have 1.5 ps RMS jitter and the reference clock is 35 MHz, then we have -145.1 dBc/Hz noise floor for reference clock. With FCW equal to 100, a resonable value in our design, the noise floor would be -105.1 dBc/Hz.

3. The finite resolution of TDC. In Fig 2-9, the compensation of TDC completely matches the quantization error of incrementor. However, that only happens when we have infinitesimal resolution for TDC. In our design, TDC can have a resolution of between 10.5–12.5 ps. Thus the phase error for digital logic is hurt by the TDC resolution.

   If the input for TDC is fully random, then the error from the quantization can be simplified as a quantization noise, like the case in ADC. The phase resolution corresponding to TDC resolution is $2\pi\frac{T_{inv}}{T_{CKV}}$. Then the quantization noise in phase domain is

$$\sigma_\Phi^2 = \frac{(2\pi)^2}{12}(\frac{T_{inv}}{T_{CKV}})^2 \tag{3-18}$$

   The quantization noise is sampled by reference frequency, thus the phase noise contribution due to TDC resolution is[4]

$$\mathcal{L} = \frac{(2\pi)^2}{12}(\frac{T_{inv}}{T_{CKV}})^2\frac{1}{f_R} \tag{3-19}$$

   For our design, the worst case occurs when T_{inv} is the maximal value (12.5 ps), T_{CKV} is the minimal value (1/4.05 GHz) and f_R is the low bound of reference frequency range (30 MHz), the TDC phase noise is -95.5 dBc/Hz.

4. The finite frequency resolution of the DCO capacitor bank. The digital loop filter can, in theory, give an infinite-length NTW word as the DCO tuning word. However, DCO capacitor banks have only limited resolution. Therefore some information of NTW will be lost as we transfer that to OTW, which may add to the phase noise of the whole system.

   The effect of the finite frequency resolution in the tracking bank is of interest since ADPLL normally works in the tracking mode. The resolution in the tracking mode is defined by the $\Sigma\Delta$ modulator as in Fig 2-5. This has been discussed in [14]. The finite fractional bits of tracking bank and the shaped quantization noise from the

[4]For the single-sided spectrum of the phase W_ϕ we have $W_\phi = \frac{\sigma_\Phi^2}{f_R/2}$. Since $\mathcal{L} = \frac{W_\phi}{2}$, we have $\mathcal{L} = \frac{\sigma_\Phi^2}{f_R}$.

modulator both contribute to the phase noise. The equivalent phase noise at the DCO output caused by the finite resolution and the ΣΔ modulator is

$$\mathcal{L}(\Delta f) = \frac{1}{12}(\frac{K_{dco}}{2^{W_F}\Delta f})^2\frac{1}{f_R}(\text{sinc}\frac{\Delta f}{f_R})^2 + \frac{1}{12}(\frac{K_{dco}}{\Delta f})^2\frac{1}{f_{\Sigma\Delta}}(2\sin\frac{\pi\Delta f}{f_{\Sigma\Delta}})^{2n}(\text{sinc}\frac{\Delta f}{f_{\Sigma\Delta}})^2 \quad (3\text{-}20)$$

where W_F is the fractional bit of the modulator input, $f_{\Sigma\Delta}$ is the working frequency of the sigma-delta modulator, n is the order of modulator and K_{dco} is the frequency resolution of the tracking bank.

In Equation 3-20, the first term represents the quantization error due to the finite fractional bits of tracking bank. The second term represents the ΣΔ noise, i.e. the noise shaped quantization error due to the modulator. The sinc function in both terms represents the effect of the zero order hold. In the first term the value is clocked by CKR and in the second term the value is clocked by CKVD. So the divisors of sinc fuction in two terms differ. The sin function in the seond term represents the noise shaping of the n$^{\text{th}}$ order ΣΔ modulation.

IN our design W_F=5 and n=1. K_{dco} and $f_{\Sigma\Delta}$ both vary with the DCO working frequency. For the frequency offset region we care about, $\Delta f \ll f_{\Sigma\Delta}$, then the part of Equation 3-20 at the right hand side of '+' symbol can by estimated as

$$\frac{1}{12}(\frac{K_{dco}}{\Delta f})^2\frac{1}{f_{\Sigma\Delta}}(2\sin\frac{\pi\Delta f}{f_{\Sigma\Delta}})^2(\text{sinc}\frac{\Delta f}{f_{\Sigma\Delta}})^2 \approx \frac{1}{12}(\frac{K_{dco}}{\Delta f})^2\frac{1}{f_{\Sigma\Delta}}(2\frac{\pi\Delta f}{f_{\Sigma\Delta}})^2 = \frac{\pi^2}{3}\frac{K_{dco}^2}{f_{\Sigma\Delta}^3} \quad (3\text{-}21)$$

As we know, K_{dco} is proportional to f^3, thus the value in the above equation reaches its largest value for the highest CKV frequency. Table 2-1 says at the highest frequency $K_{dco} = \frac{273\text{kHz}}{4} \approx 68\text{kHz}$, $f_{\Sigma\Delta} = \frac{4050\text{MHz}}{8} \approx 500\text{MHz}$. Then the result of the above equation is -159 dBc/Hz, which is fairly enough for our specification.

For the evaluation of the second term of Equation 3-20, as this part shows a -20 dB/dec profile, it would be easier to compare that with the DCO phase noise. We choose K_{dco} to be 68 kHz and f_R as 30 MHz.The phase noise incurred at the 100 kHz frequency offset is -119 dBc/Hz and at the 10 kHz frequency offset is -99 dBc/Hz. That's much smaller than the contribution from the DCO phase noise.

Fig 3-6 plots the phase noise spectrum due to the frequency resolution of the DCO tracking bank. The red line corresponds to the first term in Equation 3-20 and the blue line represents the second term in Equation 3-20. The green line is the sum of the two terms. One can see that for the in-band and transition-band frequency region (Δf from 10 kHz to 10 MHz), the green line is much lower compared to the DCO phase noise profile as in Fig 3-5. For the far-out frequency region (Δf larger than 10 MHz), the green line is rather lower than the specification of -150 dBc/Hz.

To give a summarization, the frequency resolution of DCO and the 1st ΣΔ modulation will not degrade the performance of ADPLL. In the discussion below, we will ignore the influence of this factor.

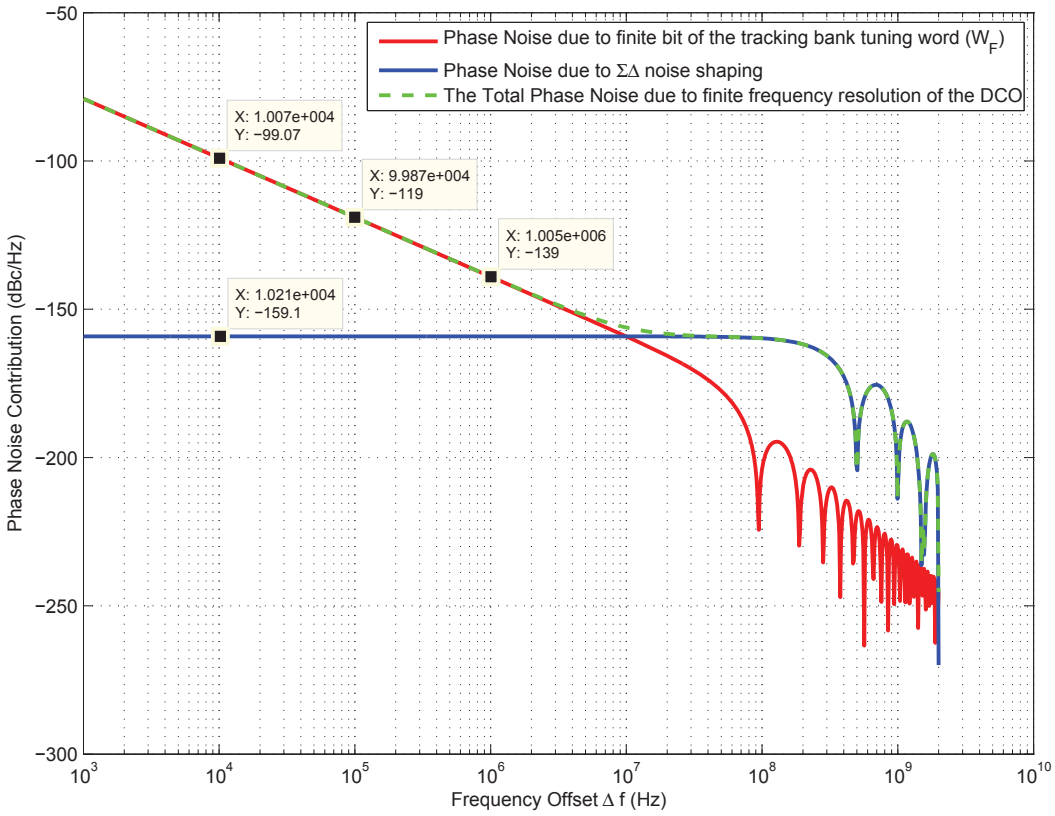

Figure 3-6: Phase noise spectrum due to the frequency resolution of the DCO tracking bank.

3-4-2 S-domain Analysis for the ADPLL System

When we look back at Table 1-1, the frequency range is covered in the analog domain by a careful frequency planning and in the digital domain by enough integer bits for the FCW signal. The requirement for the frequency resolution is satisfied via enough fractional bits for FCW signal. To gain an insight on how ADPLL can satisfy the phase noise requirement, the s-domain analysis is performed for a quick estimation.

For the phase noise, we are mostly concerned with the main working status of ADPLL, which means when ADPLL is in tracking mode and has switched to a type-II PLL. A s-domain diagram for ADPLL as in Fig 4.45 of [14] is redrawn in Fig 3-7.

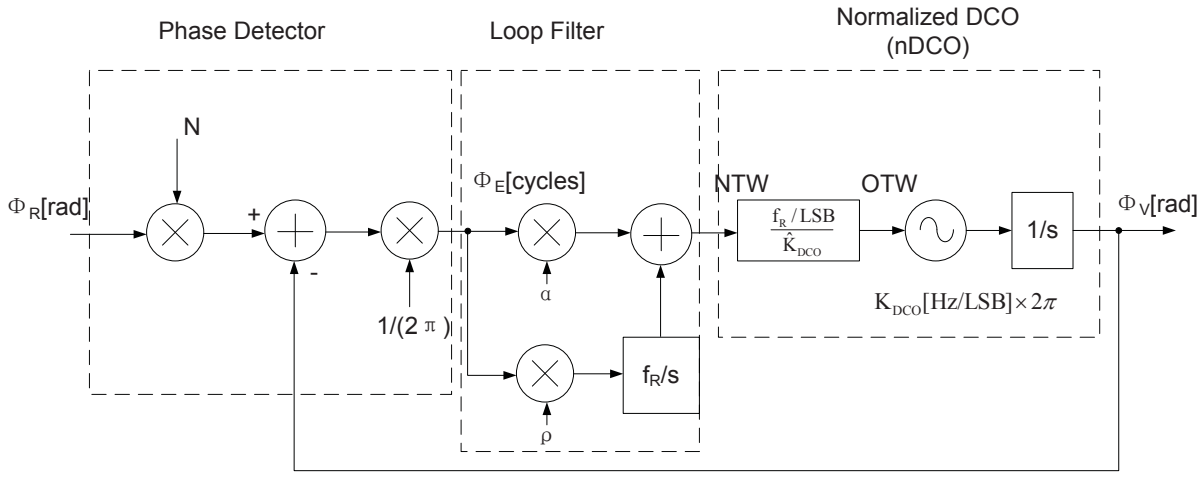

Figure 3-7: S-domain model for the type-II ADPLL.

The DCO features a gain of $K_{DCO} \times 2\pi$ as

$$\phi_V = \int 2\pi f dt = \int 2\pi (f_0 + OTW \times K_{DCO}) dt \tag{3-22}$$

The normalization circuits in the low speed digital logic block are included in Normalized DCO (nDCO) part in s-domain schematic. The input of normalization circuit is NTW, which is the sum of the proportional path and the integral path. We estimate K_{DCO} as $\hat{K}_{DCO}$ and in the normalization circuit, NTW is multiplied with $\frac{f_R}{\hat{K}_{DCO}}$.

For the integral path, we have the expression of accumulator:

$$y[k] = y[k-1] + \rho x[k] \tag{3-23}$$

In z-domain, we have

$$\frac{Y(z)}{X(z)} = \frac{\rho z^{-1}}{1 - z^{-1}} = \frac{1}{z - 1} \tag{3-24}$$

As $z \approx 1 + j2\pi f/f_R = 1 + s/f_R$, a simple calculation gives that the s-domain transfer function for the accumulator is f_R/s. [5]

For the phase detector, the desired phase information and the feedback information is compared and their difference is normalized to the CKV cycle. The desired phase information is FCW times the phase of reference clock Φ_R. Here we use N instead of FCW for more general analysis. The normalization of the phase difference is just a division by 2π.

Assuming we have an accurate enough estimation of K_{DCO}, i.e. $\hat{K}_{DCO} = K_{DCO}$, the open loop transfer function for ADPLL is

$$H_{ol}(s) = (\alpha + \frac{\rho f_R}{s})\frac{f_R}{s} = \frac{\alpha f_R s + \rho f_R^2}{s^2} \tag{3-25}$$

The closed-loop transfer function of the oscillator phase nosie is

$$H_{cl,v}(s) = \frac{s^2}{s^2 + \alpha f_R s + \rho f_R^2} \tag{3-26}$$

The closed-loop transfer function of the reference phase noise is

$$H_{cl,FREF}(s) = N\frac{\alpha f_R s + \rho f_R^2}{s^2 + \alpha f_R s + \rho f_R^2} \tag{3-27}$$

The closed-loop transfer function of the TDC phase noise is

$$H_{cl,TDC}(s) = \frac{\alpha f_R s + \rho f_R^2}{s^2 + \alpha f_R s + \rho f_R^2} \tag{3-28}$$

From [5, 14], we know that the loop gain $K = \alpha f_R$[6], the natrual frquency $\omega_n = \sqrt{\rho} f_R$ and the damping factor $\zeta = \frac{1}{2}\frac{\alpha}{\sqrt{\rho}}$. Say the reference frequency is 35 MHz, α is 2^{-6} and $\rho = 2^{-15}$, we have K=546.9 k rad/s=87.0 kHz, ω_n=193.3 k rad/s=30.8 kHz, ζ=1.414.

Here we have the TDC transfer function and the DCO transfer function with f_R of 33.8688 MHz. This particular reference frequency is chosen as some commercial available crystal oscillator just operates at that frequency. The TDC transfer function is plotted in Fig 3-8. It's almost flat when the frequency offset is smaller than loop gain K and then rolls down in an asymptotes with a slope of -20 dB/dec. The DCO transfer function is plotted in Fig 3-9. It's almost flat for the frequency offset higher than 200 kHz. For the close-in offset frequency, the phase noise rolls up at an asymptotes of 40 dB/dec.

The 40 dB/dec slope of DCO transfer curve with the -30 dB/dec up-converter flicker noise profile in DCO together show a 10 dB/dec slope, which means the up-converted flicker

[5] An intuitive explanation is that the accumulation rate is f_R. Then we have a coefficient of f_R with 1/s for the integration.

[6] In [5] this loop gain concept is mentioned. That can be a more reasonable estimation for the bandwidth of type II PLL.

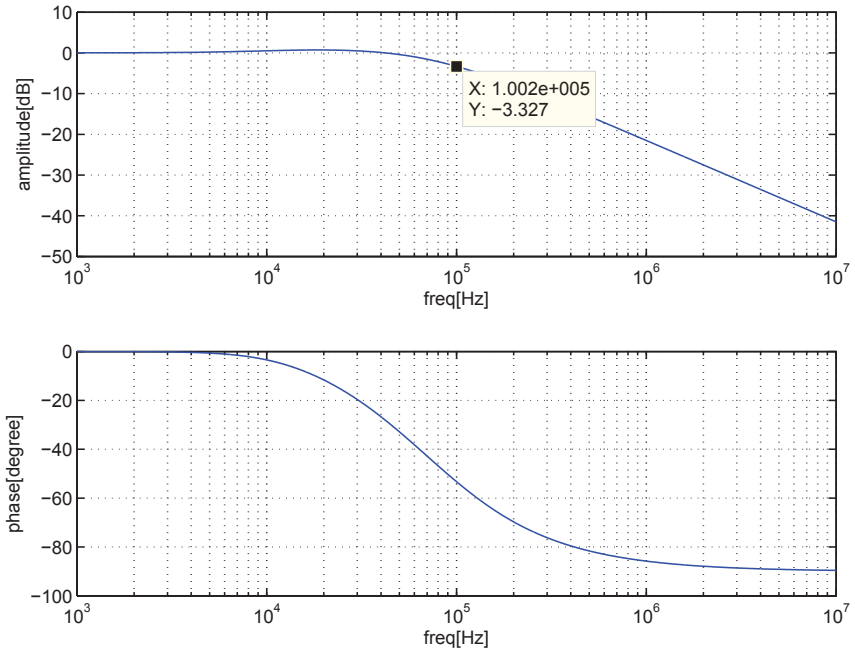

Figure 3-8: TDC Transfer function for type II PLL ($\alpha = 2^{-6}, \rho = 2^{-15}$).

noise in DCO is suppressed in the type-II PLL. The contribution of DCO to ADPLL phase noise in type-II ADPLL is shown in Fig 3-10. One can clearly see the 10 dB/dec rolling up for the close-in frequency offset region and the -20 dB/dec rolling down for the far-out frequency offset region, which is just a up-converted white noise profile in the DCO phase noise.

Then the intrinsic tradeoff of all PLL system shows up if phase noise specification is to be satisfied. When we have larger α and ρ, the bandwidth of PLL is larger and the DCO phase noise is more suppressed. Thus the design effort for DCO is relaxed. However, at the same time, the flat frequency offset region for TDC and the reference phase noise transfer curve extends, which indicates that we need less noise contribution from TDC and the reference to meet the requirement for the in-band phase noise and the integrated phase noise. This adds to the design difficulty of TDC, a FREF slicer and also adds to the cost of a high-quality reference.

In the design practice, both TDC and DCO take a lot of effort and are power hungry. It's essential to select optimal coefficients for ADPLL so as to ease design pressure for the two blocks. The traditional rule of thumb is to select the offset frequency where (reference phase noise+TDC phase noise) and DCO phase noise meet as the PLL bandwidth. Here we have the worst case TDC phase noise of about -95.5 dBc/Hz, which is very close to the phase noise requirement of -95 dBc/Hz at 100 kHz frequency offset. What's more, to avoid the violation of the integrated phase noise specification, the bandwidth cannot be

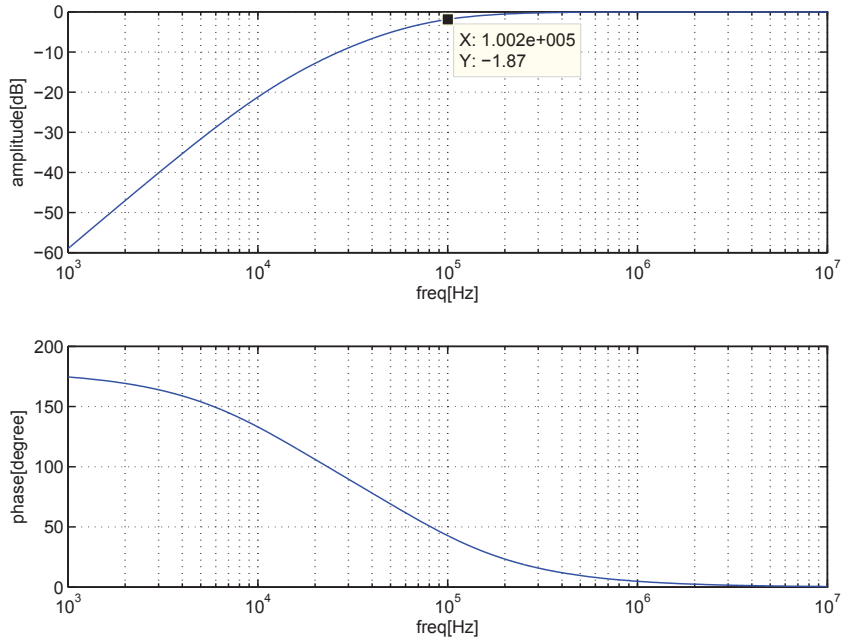

Figure 3-9: The DCO Transfer function for the type II PLL ($\alpha = 2^{-6}, \rho = 2^{-15}$).

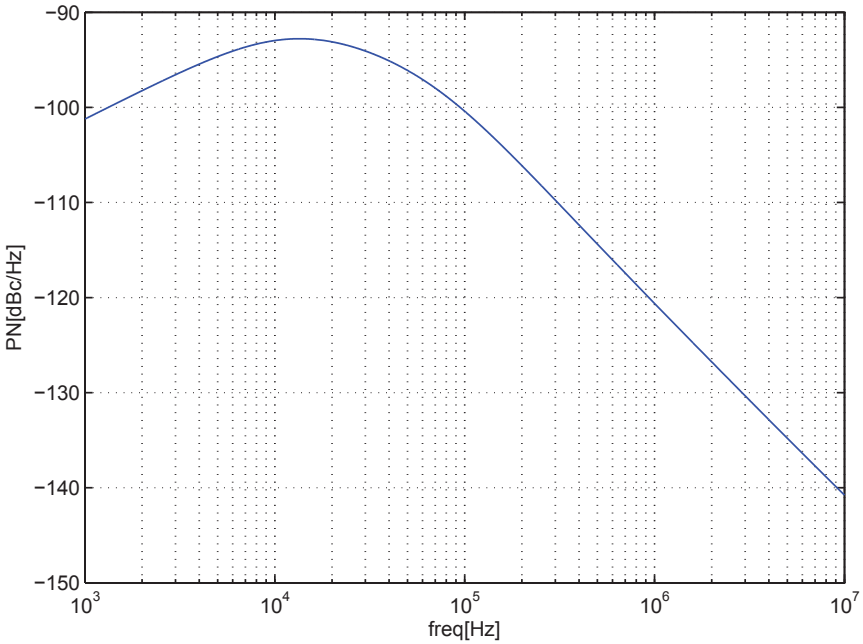

Figure 3-10: The DCO contribution to the ADPLL phase noise in the type-II PLL.

too large. For the worst case of the TDC resolution (slow corner, high temperature and low voltage), the bandwidth is expected to be less than 100 kHz.

Besides the bandwidth, for type-II PLL, we have another value which is the damping factor ζ. For our case, a over-damped type-II PLL ($\zeta \geq 1$) is preferred since it avoids a significant gain peaking [7]. For ζ=1.414, as we look at Fig 3-8 and Fig 3-9, there is no peaking for DCO transfer function and only less than 1dB gain peaking for TDC transfer function. This is especially useful for our case as we are actually stressed by the phase noise due to not-so-fine TDC resolution. One can see that at the 100 kHz frequency offset where most tough spec lies, TDC and DCO transfer function both show a little bit attenuation.

Up till now, we have left the IIR filter bank in ADPLL logic untouched. For an IIR filter as shown in Fig 2-27, we have its s-domain equation [14]

$$H_{iir}(s) = \frac{1 + s/f_R}{1 + s/(2^{-\lambda} f_R)} \tag{3-29}$$

The intuition tells us that the contribution to the far-out phase noise from TDC and reference clock would be reduced. However, as the DCO far-out phase noise is already uncorrected, it's hard to see the influence of the filter bank to the DCO phase noise contribution in the ADPLL system. Here s-domain simulation is done and we get the TDC transfer function in Fig 3-11 and the DCO transfer function in Fig 3-12. Here first and second filter in the bank are turned on. Lambda0 is 1 and Lambda1 is 2.

Comparison between Fig 3-8 and Fig 3-11 shows that IIR filters do add attenuation to the far-out phase noise due to TDC and the reference clock. Comparison between Fig 3-9 and Fig 3-12 reveals that the DCO transfer function has almost no change, expect a very small peaking at out-of-band frequency region. Thus the total far-out noise can be reduced without degrading the close-in phase noise performance. With IIR filters in the IIR filter bank all turned on and λ all chosen as 3, the attenuation in TDC transfer function is much more obvious yet with significant peaking for DCO transfer function. In our design, with only Lambda0=1 and Lambda1=2, the extra filtering is already enough.

Through s-domain analysis and simulation, one can have a rough estimation of DCO phase noise requirement. The specification for DCO design is shown below

Phase noise specification	Low frequency (3300MHz)	High frequency (3800MHz)
@10 kHz offset (dBc/Hz)	-71.7	-72.3
@100 kHz offset (dBc/Hz)	-98.0	-98.9
@10 MHz offset (dBc/Hz)	-140.2	-140.9

Table 3-3: DCO phase noise specification.

[7] The over-damped type-II PLL is sometimes called the 1st order type-II PLL as it features minimum gain peaking and suppression of DCO close-in phase noise.

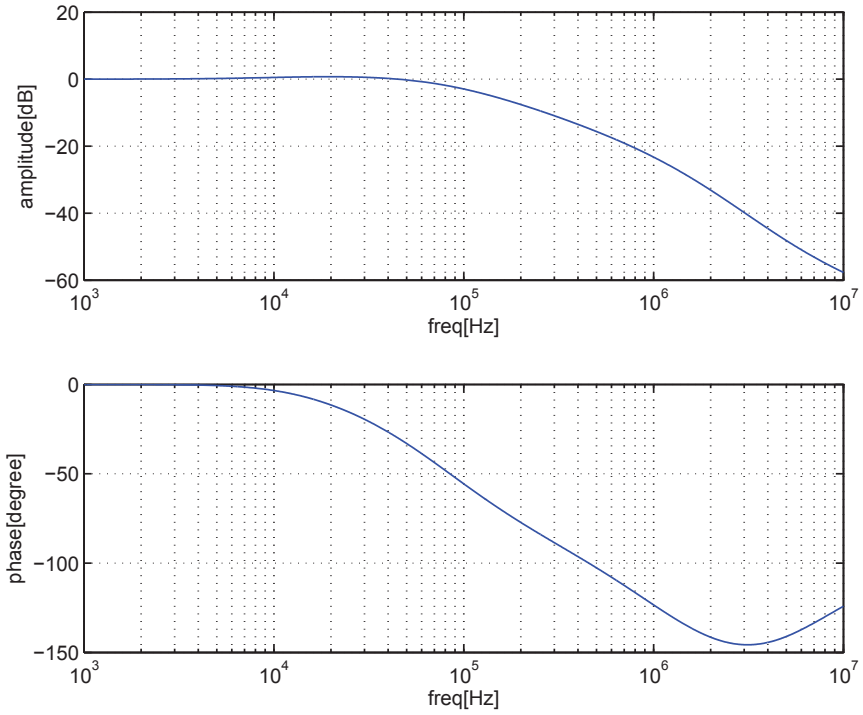

Figure 3-11: TDC transfer function for type II PLL with IIR filter bank

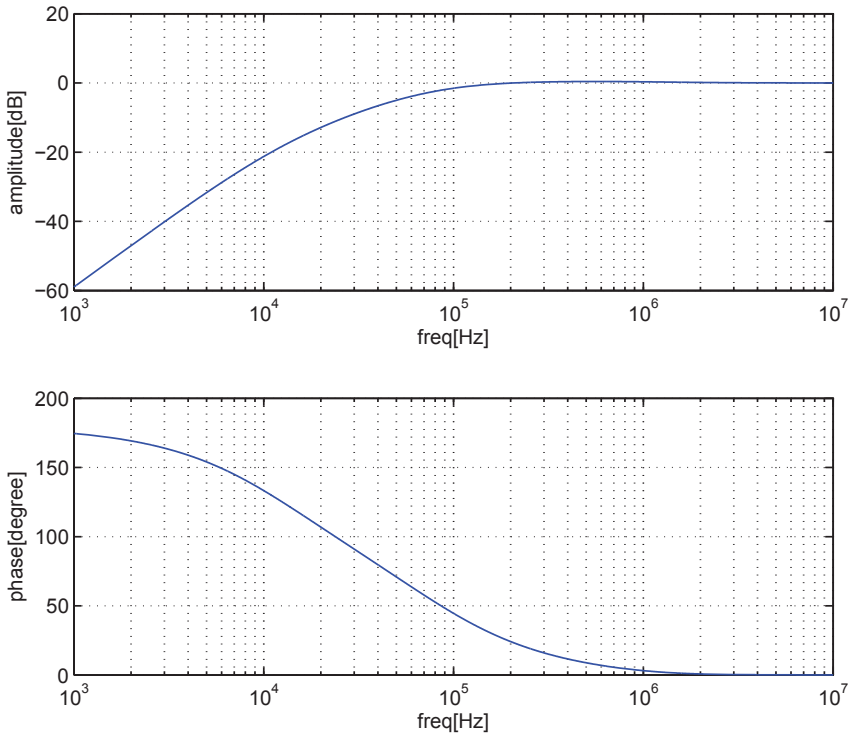

Figure 3-12: DCO transfer function for type II PLL with IIR filter bank

Notice in the s-domain simulation the highest frequency is 3.8 GHz rather than 4.05 GHz, as the frequency range of 3.8-4.05 GHz is only intended for the LB output. Since the HB outputs and the feedback CKV signal are DCO core outputs divided by 4 while LB outputs are DCO core outputs divided by 6, the effective division ratio of the LB output compared to the CKV signal is 1.5. Thus the LB output phase noise is 3.5 dB lower than HB output phase noise. There is no need to worry about the LB output phase noise as long as the requirement for the HB output phase noise is met.

The ADPLL phase noise is simulated in s-domain with the value in Table 3-3. The reference clock is 33.8688 MHz and TDC resolution is the worst case value 12.5 ps. RMS jitter for reference clock is 1.5 ps. One can see that in Fig 3-13 and Fig 3-14 for the high frequency and the low frequency the ADPLL phase noise complies with the specification.

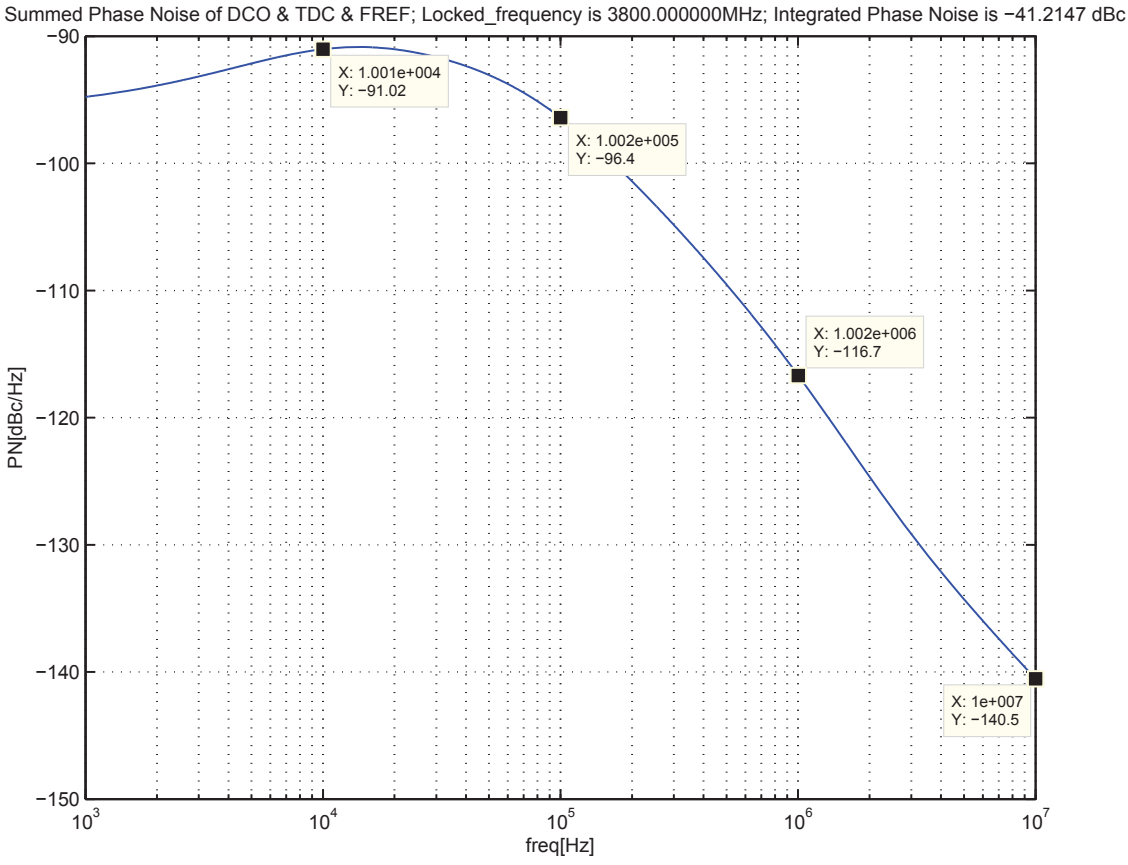

Figure 3-13: S-domain result for ADPLL PN (f_v=3800 MHz).

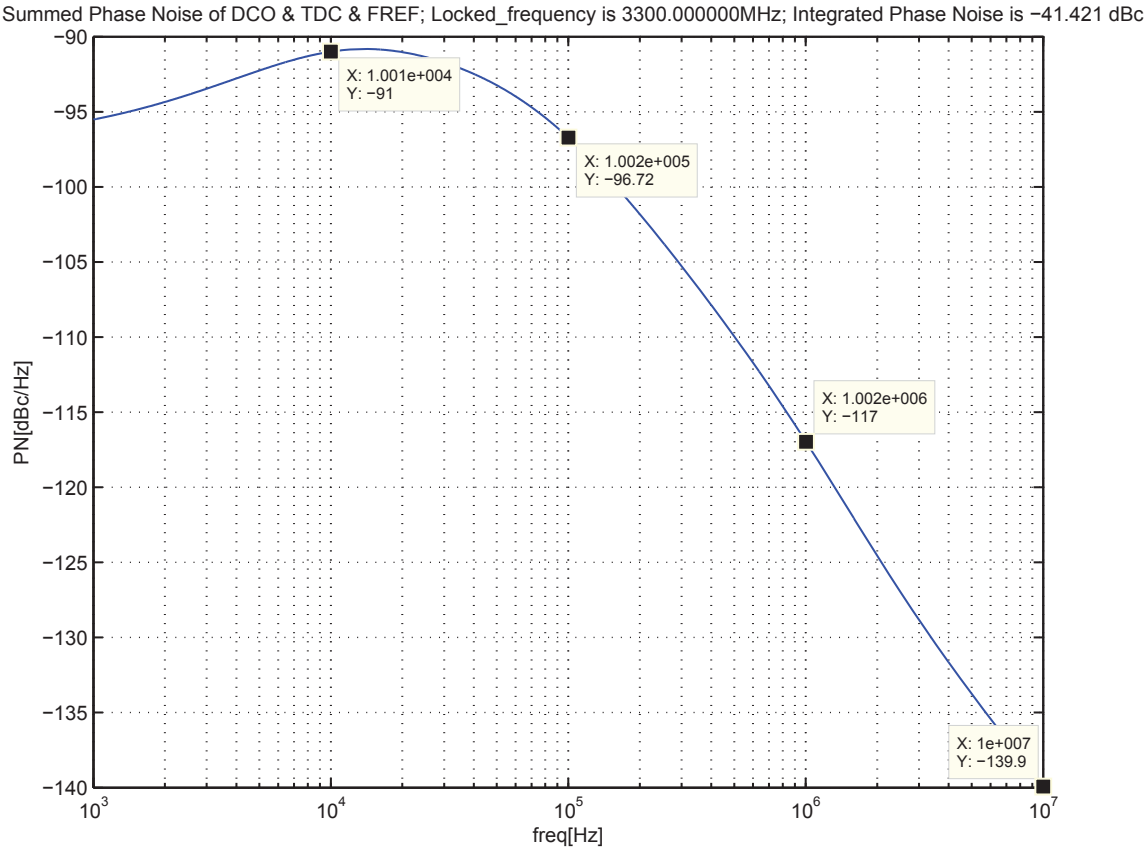

Figure 3-14: S-domain result for ADPLL PN (f_v=3300 MHz).

Chapter 4

Advanced Algorithm for ADPLL

4-1 Zero Phase Restart

When ADPLL goes from the PVT mode to the acquisition mode, CTL_PLL_P becomes low and the control word for PB is frozen. Thus in the view of the acquisition bank, DCO has a new central frequency determined together by the original DCO frequency and the frozen PB control word. Notice that in the PVT mode, the phase error information is obtained on the condition that NTW_A is 0 and NTW_T is 0. If MEM_DCO_A (as in Fig 2-25) and MEM_DCO_T are zero, then the acquisition bank and the tracking bank are fixed to the middle capacitance value (half capacitance units turned on and half turned off) in PVT mode. To make sure that the DCO frequency settles smoothly, PHE shall be reset to zero once ADPLL switches to the acquisition mode. In this way NTW_A and NTW_T are set to 0 at the beginning of the acquisition mode. The DCO output frequency is consistent before and after the mode switchover. Similarly, the PHE value shall also be reset during the switchover from the acquisition mode to the tracking mode.

We call this reset at the mode switchovers as *zero phase restart* (ZPR). The implementation is rather straightforward with the differentiation-then-accumulation phase detection logic as in Section 2-5. One only needs to set the Reset||ZPR signal (as in Fig 2-22) high to reset the accumulator at the mode switchover event. At Fig 2-25, the detection logic of the mode switchover event is presented. Once CTL_PLL_P goes from high to low, the previous registered value is high and inversion of the current value is also high. Then OP_ZPR becomes 1. At the next CKR cycle, OP_ZPR will becomes low again as the registered CTL_PLL_P becomes low. So a positive pulse of OP_ZPR would indicate that ADPLL is switching from PVT mode to acquisition mode.

Similarly, OA_ZPR signal sends a positive pulse when ADPLL is going from acquisition mode to tracking mode. We feed the system reset signal, OP_ZPR and OA_ZPR signal

to an OR gate and its output, Reset||ZPR will reset the PHE signal in the occasions we expect.

Fig 4-1 shows the transient with the ZPR mechanism. One can easily see the OP_ZPR and OA_ZPR pulse at the mode switchover. Then PHE signal is reset to zero. In this figure the pulse of signal OP_ZPR is wider than OA_ZPR. That is because in the implementation the falling edge of CTL_PLL_P is detected and then OP_ZPR pulse is stretched to last for several CKR cycles. Then the DCO can settles to the frequency indicated by the frozen word for PVT bank before it starts the acquisition mode. The stability is improved. As in the switchover from the acquisition mode to the tracking mode we don't see the stability problem, one CKR cycle is enough for the OA_ZPR pulse.

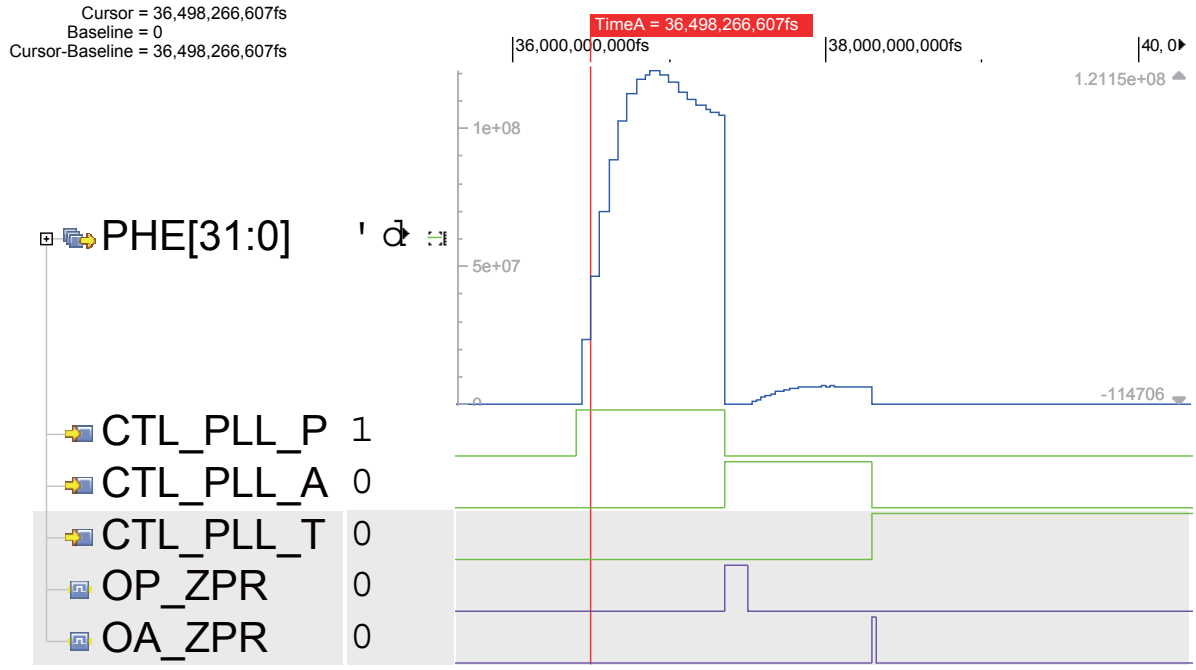

Figure 4-1: Transient for zero phase restart.

4-2 PVT Miss Mechanism

In Fig 2-4 there are two tuning words for PVT bank: DCO_IN_P_G and DCO_IN_P_DS, one connected to the gate of the switch in the capacitor units and one connected to the drain and source of the switch in the capacitor units via inverters. They are used to prevent the choking of the DCO core.

The capacitance of the PVT bank plays a significant part in the total capacitance in DCO. When drastic capacitor switching events happen in PVT mode, e.g. the toggling event of

the tuning word between 0111111 to 1000000, chances are that the oscillator may pause working for a short time before it resumes the oscillation and settles to the new frequency. During this transient, the divider will not capture the expected edge. The output of TDC and incrementor also fails to deliver the correct phase information.

The transistor level simulation reveals that DCO works fine when the switches are turned off. Yet the choking does happen for the case when switches are turned on. A valid solution obtained from the transistor level design exploration is to pull up DCO_IN_P_DS signal before pulling up DCO_IN_P_G. Fig 4-2 gives a digital implementation for this purpose. One can see that for the rising edge of DCO_IN_P, DCO_IN_P_G rises one CKR cycle later while DCO_IN_P_DS has no delay; for the falling edge of DCO_IN_P, both DCO_IN_P_G and DCO_IN_P_DS are delayed. Since DCO_IN_P_G is to turn on/off the switch, this circuit introduces one extra delay to the PVT mode of ADPLL.

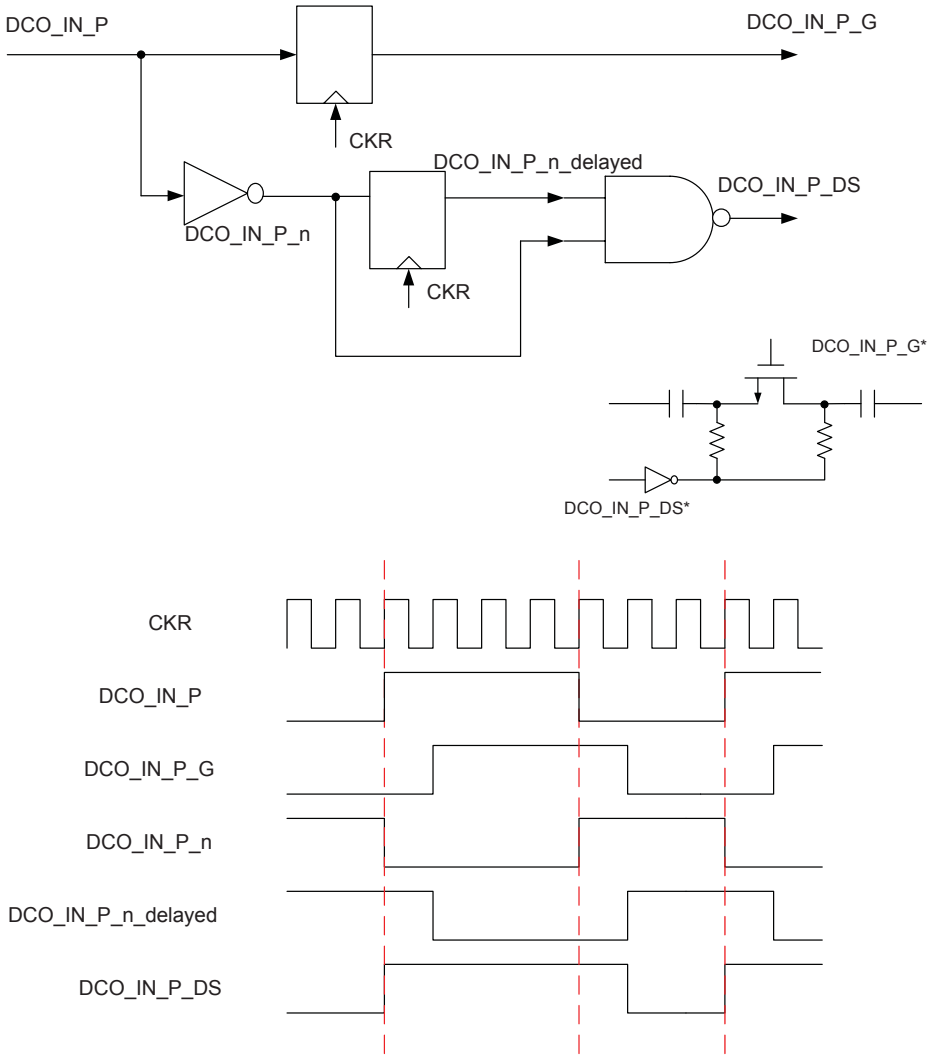

Figure 4-2: Logic for PVT tuning word generation and the timing diagram.

Still one can easily see that when DCO_IN_P keeps toggling between 1 and 0, DCO_IN_P_DS will be always high, i.e. the drain and source of switches are always biased at voltage of logic 0 (VSS), which is undesired. We avoid this situation by the mechanism asserted below:

If DCO_IN_P changes at one CKR rising edge, it shall keep this value for the next CKR rising edge. In the other word, once changed, DCO_IN_P shall be constant for at least two cycles.

To achieve this, PHE needs to be frozen at some certain time, which means dPHE is missed at these moments. We call this the PVT miss mechanism. It can be done by observing TUNE_P, the input of OP block. Ideally, TUNE_P changes one cycle ahead of DCO_IN_P. Thus when the logic detects the change of TUNE_P while ADPLL is in the PVT mode[1], PVT_miss signal will be made high to freeze the accumulator output PHE. Fig 4-3 shows the signal waveforms when PVT miss algorithm is activated. TUNE_P will hold a certain value for at least two cycles before it changes and when it changes the value, PVT_miss signal sends a high pulse to freeze the PHE signal. The timing diagram for DCO_P, DCO_P_G and DCO_P_DS is also shown in this figure.

4-3 Spur Suppression Techniques for ADPLL

4-3-1 Spurious Tone Issue of ADPLL

For the previous s-domain analysis in Section 3-4, we have made two assumptions: 1. Every TDC cells will show the same delay. There is no mismatch between TDC cells, i.e. TDC has zero DNL and INL. 2. TDC input is fully random. Therefore the quantization error can be approximated as the quantization noise as in Equation 3-19.

Nevertheless, that is not a true image. For the static ADPLL without any modulation, when the frequency is locked, the feedback phase increases linearly. TDC input (timing difference) will follow a saw tooth trajectory with modulo of one CKV cycle. This gives rise to the periodicity of PHF and PHE, which causes spurs at the ADPLL output spectrum.

This spur issue has been treated in a universal way in [13], where the quantization effect is viewed as a subset of TDC's nonlinearity. It's asserted that when FCW=(N+α), where $|\alpha < 1|$, the spur may show up at frequency offsets which are integer times $|\alpha|$FREF, depending on the Fourier coefficients of the nonlinearity for the input-output transfer curve. As the loop filter attenuates the frequency component of PHE that are not within the PLL bandwidth, the spur performance is worst when $|\alpha| \ll 1$, which is also called near-integer N case.

[1]The detection logic just compares TUNE_P of the previous cycle and TUNE_P of the current cycle, i.e. TUNE_P[k]==TUNE_P[k-1].

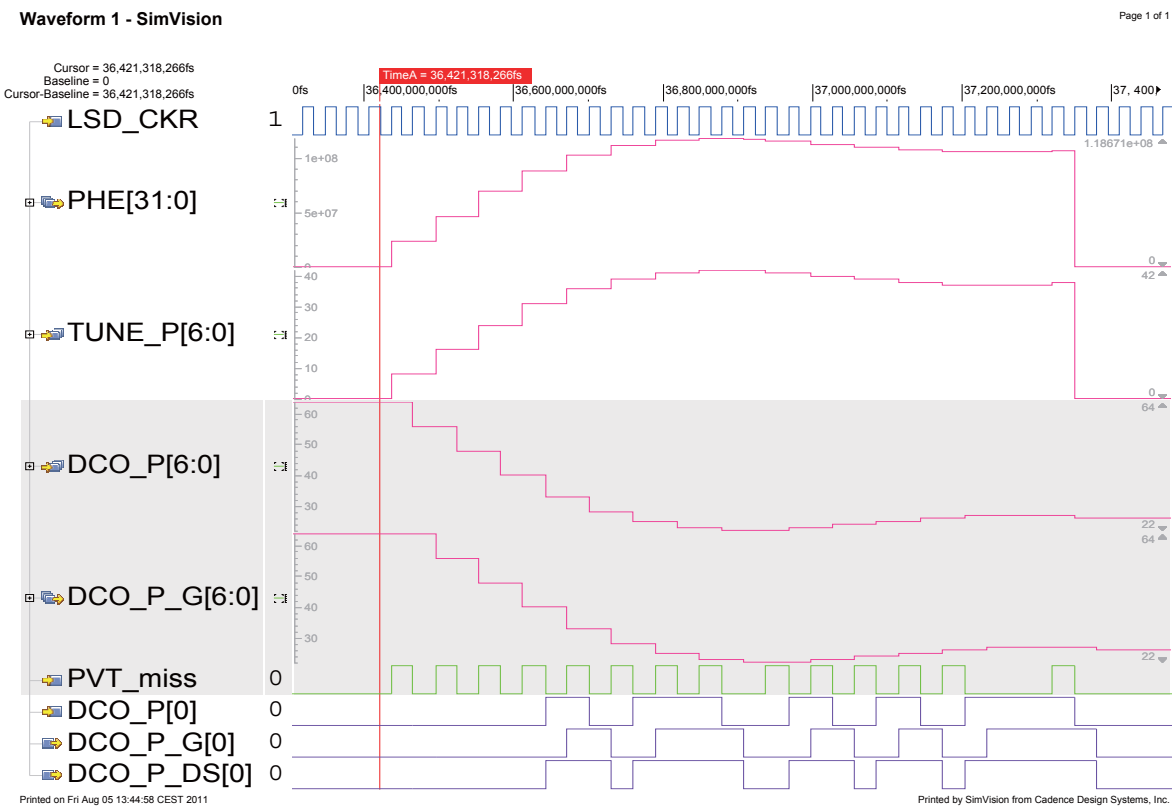

Figure 4-3: Transient signal waveforms for the PVT miss algorithm.

However, the time-domain simulation result suggests that it's better to consider the TDC mismatch and the quantization effect separately, as they raise significant spurs at different frequency offsets.

For the TDC mismatch effect, the spurs would mostly reside at the frequency offset of k*|α|FREF, where k is from 0 to 5. For the quantization effect, when $|\alpha| \ll 1$, the timing difference between FREF and CKV rising edges would drift only a little bit every FREF cycle. Then the time error indicated by the phase error signal PHE increases in a saw tooth way with the modulo of one TDC cell delay, as in Fig 4-4. The timing difference of the CKV edge and the FREF edge sampled by CKR differs from the previous sampling value by $\Delta t\_cycle = |\frac{N}{CKV} - \frac{1}{FREF}| = |\frac{N*FREF-CKV}{CKV*FREF}| = \frac{|\alpha|}{CKV}$. Then it takes $\frac{T_{inv}*CKV}{|\alpha|}$ cycles of CKR sampling to sweep one TDC cell delay, which corresponds to the period of $T_{cycle} = \frac{T_{inv}*CKV}{|\alpha|FREF}$. We could expect spur at harmonics of

$$f_{q,spur} = \frac{|\alpha|FREF}{T_{inv} * CKV} \tag{4-1}$$

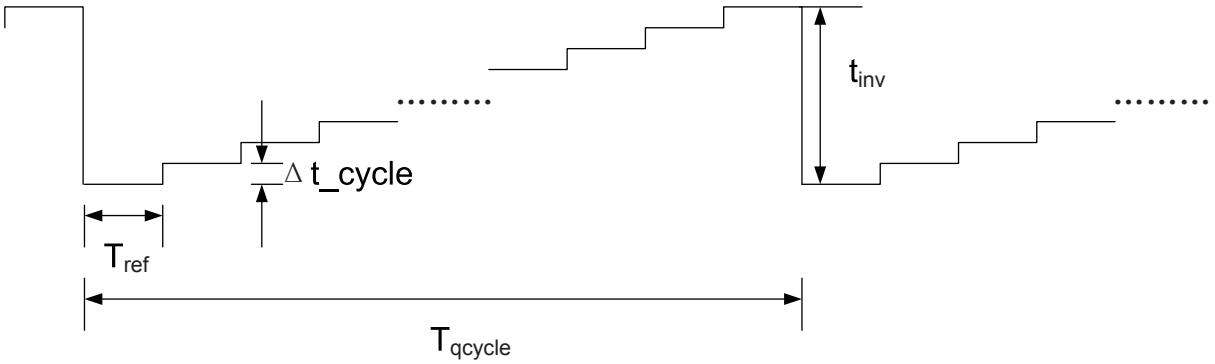

Figure 4-4: Timing diagram for spurs due to the TDC quantization effect.

For example, when $f_R = 33.8688$ MHz and $f_v = 3793.3156$ MHz, we have N=112.0002953. The desired frequency deviates only 10 kHz from the integer-N frequency. The TDC resolution is 12.5ps. The mismatch of TDC is modeled in the system as in Section 3-3.

We shall expect the spurs at frequency offsets which are integer times 10 kHz due to the TDC mismatch and which are integer times

$$\frac{10\text{kHz}}{12.5\text{ps} \times 3793.3156\text{MHz}} = 210.9\text{kHz} \tag{4-2}$$

due to the TDC quantization effect.

Here in Fig 4-5 the phase noise from the time-domain simulation of CKV with 3793.3156 MHz and FREF with 33.8688 MHz is given. There are two strong spurs located at the 10 kHz offset and the 210 kHz offset, as well as some weaker spurs at harmonics of these two frequencies, as expected from the analysis above. The transient of PHE signal is given in

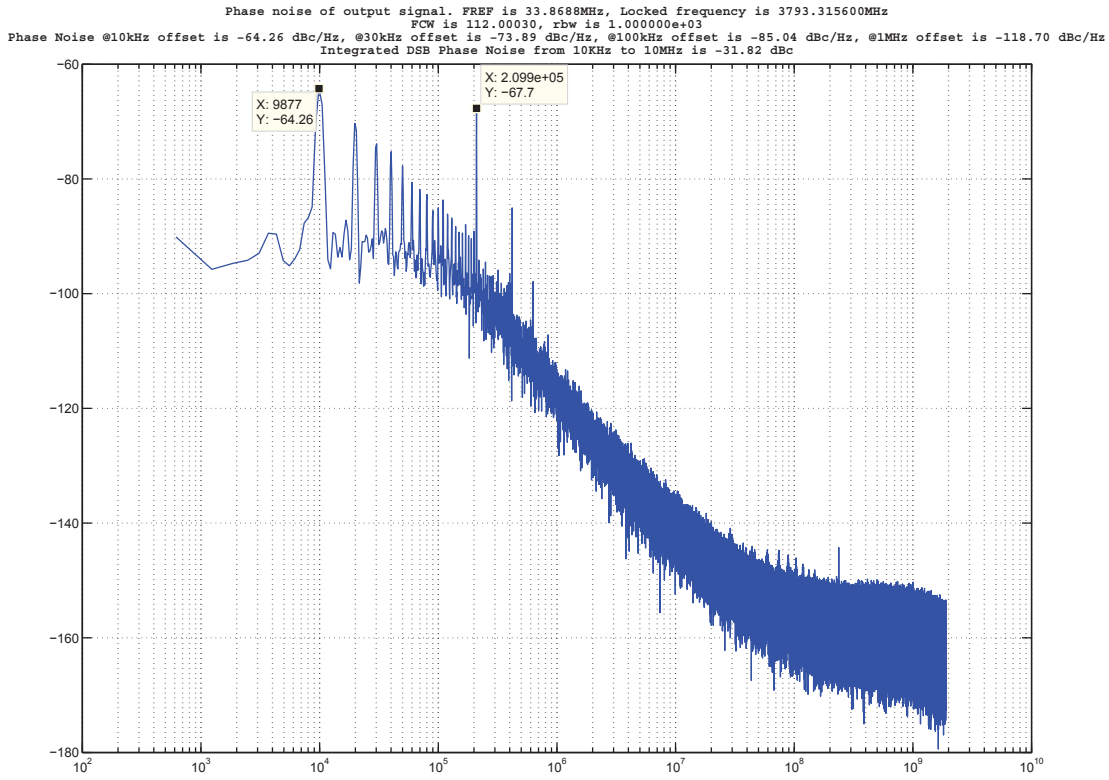

Figure 4-5: ADPLL spectrum for CKV frequency of 3793.3156 MHz, RBW=1 kHz.

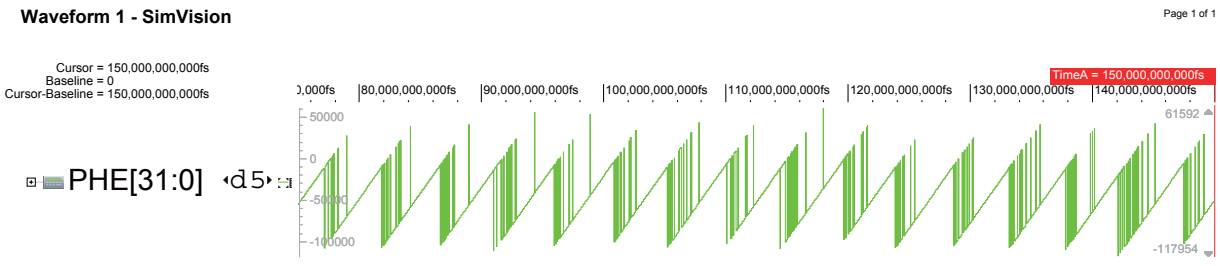

Figure 4-6: PHE transient for CKV frequency of 3793.3156 MHz.

Fig 4-6. The signal shows a period of about 5 us, which corresponds to the spurs due to the TDC quantization effect.

An obvious solution to solve this spur issue is to improve the TDC resolution, which consumes more power and increases the design complexity[16]. Thus other ways to work around this issue have been proposed, e.g. calibration or dithering[22, 13, 23]. In our design, the algorithms of phase rotation and FREF dithering have been adopted for the spur suppression. These two algorithms are chosen because they are simple yet effective for our case.

4-3-2 Phase Rotation Algorithm

As shown in Fig 2-2, the feedback clock is generated by the divided-by-2 divider, which means that the four clock phases are naturally available. As stated in [24], the feedback frequency can be offseted $-f_R/4$ via the phase rotation algorithm.

Fig 4-7 is a timing diagram similar to Fig.5 in [24], where the output of phase rotator will shift to the next phase every FREF cycle. To exemplify how this phase rotation algorithm works, we assume FCW=3.0. In Fig 4-7 (a), the phase will rotate in the sequences of $Ip \rightarrow Qp \rightarrow In \rightarrow Qn \rightarrow Ip$. When the transitions happen at the time indicated by solid line, the phase increment seen by the incrementor and TDC, as normalized to CKV cycle, will be 2.75. This is equivalent to a frequency translation of $-f_R/4$. In Fig 4-7 (b), the phase will rotate in the sequence of $Ip \rightarrow Qn \rightarrow In \rightarrow Qp \rightarrow Ip$ and the phase increment seen by the incrementor and TDC, as normalized to CKV cycle, will be 3.25. This is equivalent to the frequency translation of $+f_R/4$.

This algorithm aims at breaking the near-integer N condition for FCW, as for Fig 4-7 (a) and (b) we shall add -0.25 and 0.25 to the original FCW value. The new FCW value will not be close to an integer any more.

For Fig 4-7 (a), notice that if the phase transition of $Ip \rightarrow Qp$ happens at the time indicated by dashed lines, either leads or lags the solid line, the incrementor would see one extra rising edge. Then the phase error would deviates from the expected value by 1. One can either ignore this error value in the integer part of PHE or correct for this error, yet with the same disadvantage we have stated for the PHE spike due to the mismatch between the TDC path and the incrementor path as in Section 2-3. In another way, [24] proposes to detect the falling edge of CKR as to start up phase rotation. As for the transition of $Ip \rightarrow Qp$, when we gate the rotation event start-up with the rising edge of Qp, the timing for transition is set to be in the regime of solid line and the possibility of an extra rising edge is eliminated. However, for this method, the next phase is involved and some extra logic is needed.

In contrast to the potential problem in Fig 4-7 (a), Fig 4-7 (b) will not give rise to this extra edge issue and is adopted in our design. The phase rotation is performed by a multiplexer in Fig 2-7, which is controlled via a modulo-4 counter clocked at CKR falling edges. To

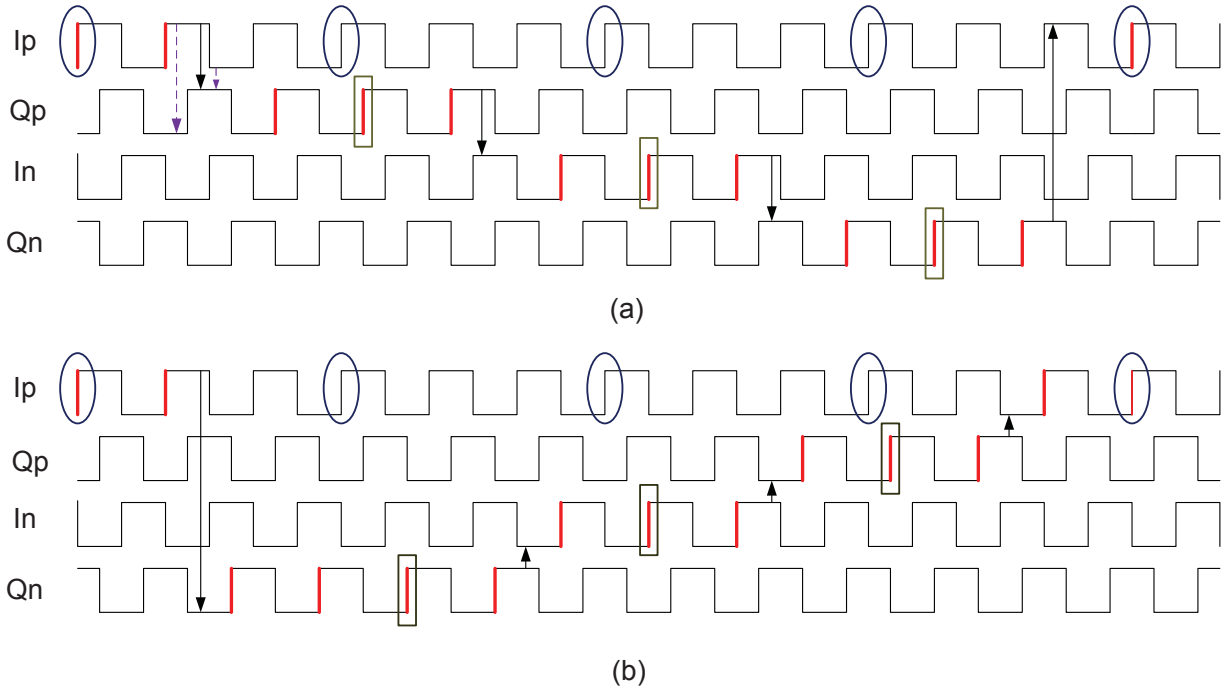

Figure 4-7: Timing diagram for phase rotation.

avoid the possible glitch in the modulo-4 counter logic, the Gray code counter shall be used. At the same time we shall compensate for the equivalent frequency translation in the phase detection logic, which is achieved in PR module shown in Fig 2-22. When PR signal is high, dPHR is the fixed-point value of $(\frac{f_v}{f_R} + 0.25)$ rather than just FCW.

Fig 4-8 shows the ADPLL spectrum for CKV=3793.3156 MHz but now with the phase rotation activated. RBW is 1 kHz. Compared to Fig 4-5, the spurs have been significantly suppressed, especially for these located at the frequency offset of 10 kHz, 20 kHz and 30 kHz. However, we still see significant spurs at around 40 kHz and 200 kHz. This phenomenon is not hard to explain. For TDC, it sees an equivalent FCW as $(\mathrm{N}+0.25+\alpha)$. Due to subsampling, the 4th order harmonic component of $(0.25+\alpha)$ falls in the bandwidth. That gives rise to the spur at 40 kHz. For the spur at 200 kHz, it is still due to the TDC quantization. However, the frequency location is subtle. In the spectrum with RBW=10 kHz as Fig 4-9, spurs are still noticeable, which means the improvement brought by only phase rotation is not that satisfactory.

4-3-3 FREF Dithering Algorithm

In [23, 25, 26] dithering has been used to reduce the spur in ill-behaved near-integer N situation. Its purpose is to introduce randomized delay of up to several T_{inv} to FREF so as to avoid the slow slope in Fig 4-4. In that way the spurs due to TDC quantization are suppressed.

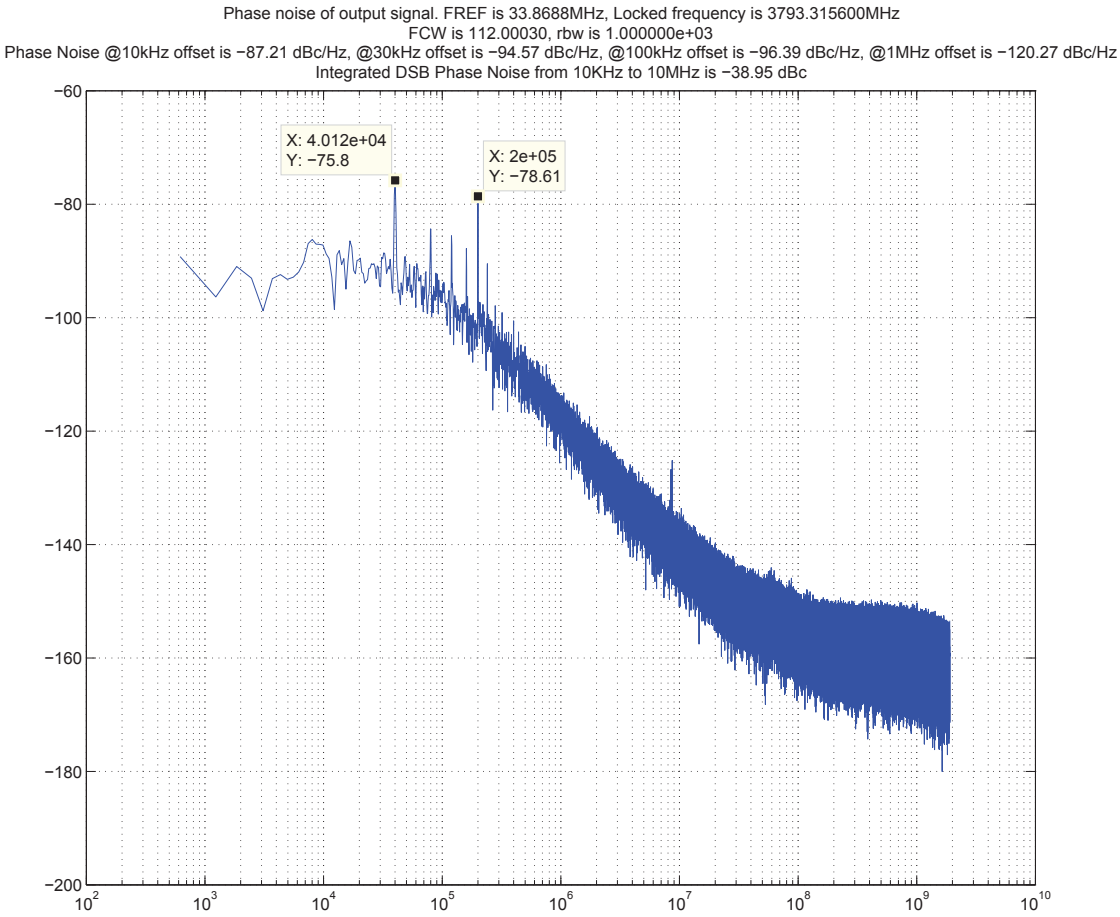

Figure 4-8: ADPLL spectrum for CKV frequency of 3793.3156 MHz with phase rotation algorithm activated, RBW=1 kHz.

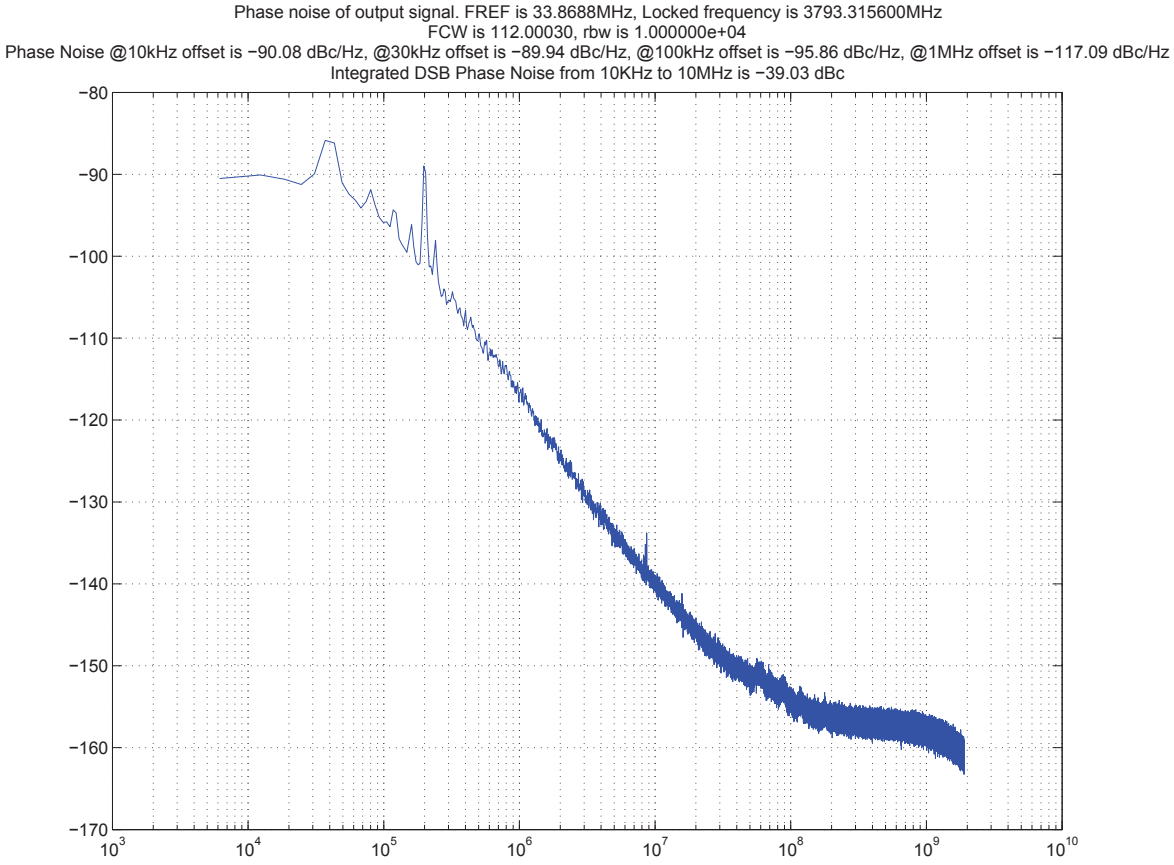

Figure 4-9: ADPLL spectrum for CKV frequency of 3793.3156 MHz with phase rotation algorithm activated, RBW=10 kHz.

In our design, the randomized ΣΔ dithering technique proposed in [25] has been adopted. The pseudo-random signal is used as the input of a 3rd order Sigma-Delta modulator shown in Fig 4-10, which just consists of three adders and several flip flops clocked at the negative edge of FREF. The sum value of d1–d7, ranging from 0 to 7, indicates how long the delay shall be added to the FREF. The delay resolution in the system simulation is assumed to be 9ps to guarantee the delay range spans several TDC cell delay. For example, if d1–d7 is 0011001, their sum is 3 and the delay for the FREF is 27 ps.

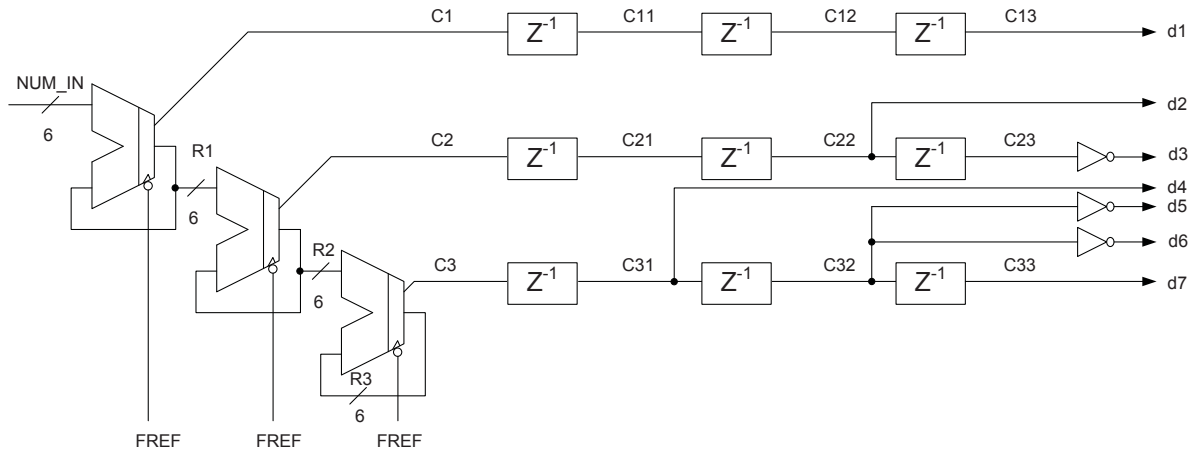

Figure 4-10: Schematic of ΣΔ modulator for FREF dithering.

The input of the modulator, NUM_IN, is randomized to avoid possible tones in the modulator output[2]. Similar to the analysis in Section 3-4-1, this random input adds to the phase noise of TDC and thus will degrade the close-in phase noise of ADPLL. To reduce this effect, the distribution range of NUM_IN is limited here. Rather than having a uniform distribution in [0,63] the 6 bit NUM_IN is uniformly distributed in [13,44]. Through this the noise contribution from FREF dithering is lowered down by 6 dB and the modulator output is still free from tones.

In the schematic of Fig 2-7 the FREF dithering is absorbed into the slicer. In real silicon, that can be done either in a FREF slicer as in [26] or in an individual DTC in [23], which is not implemented yet. More accurate model extracted from the transistor-level simulation shall be substituted into the system simulation to determine the performance.

The spectrum of ADPLL for CKV=3793.3156 MHz with FREF dithering activated only is shown in Fig 4-11. RBW is 1 kHz. The spurs due to the TDC quantization are well suppressed yet we still can see spurs at 10 kHz / 20 kHz / 30 kHz / 40 kHz, which is what we expect as the FREF dithering does not help much on the mismatch of the whole TDC. In the ADPLL spectrum with RBW of 10 kHz in Fig 4-12, the inband noise is very high due to the spurs.

[2]One can use LFSR to create pseudo-random number

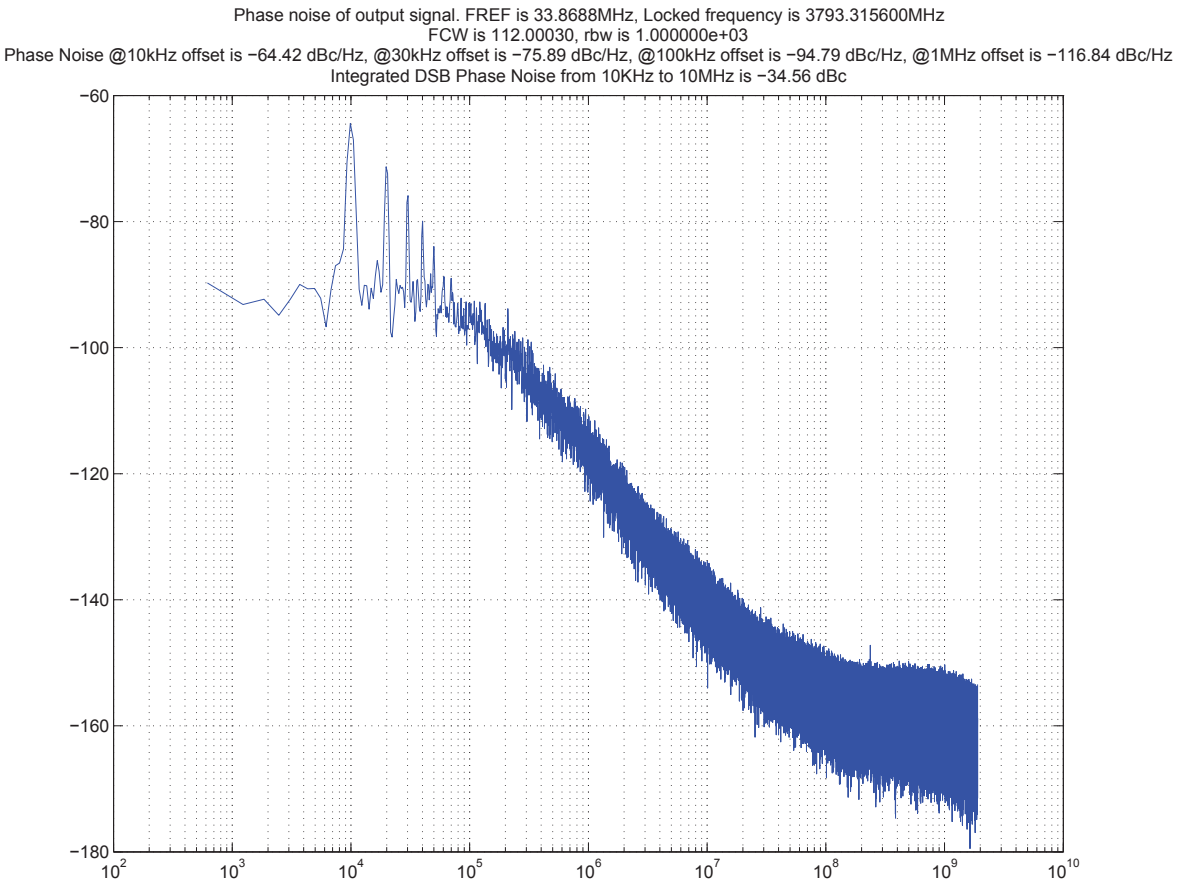

Figure 4-11: ADPLL spectrum for CKV frequency of 3793.3156 MHz with FREF dithering Algorithm Activated, RBW=1 KHz.

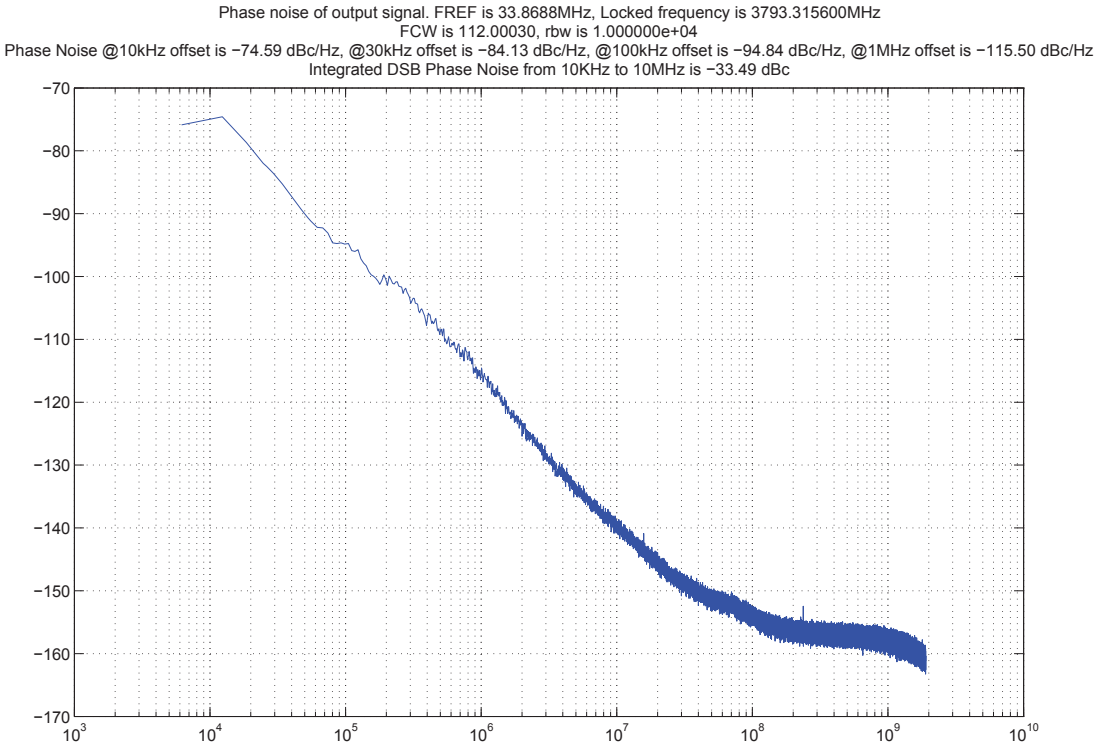

Figure 4-12: ADPLL spectrum for CKV frequency of 3793.3156 MHz with FREF dithering algorithm activated, RBW=10 kHz.

4-3-4 Spur Suppression with phase rotation and FREF dithering

From Fig 4-8 and Fig 4-11, one can conclude that the phase rotation is more effective against spurs due to the TDC cell mismatch and FREF dithering is more effective against spurs due to the TDC quantization effect. With both algorithm activated, the spurs shall be sufficiently suppressed. Fig 4-13 and Fig 4-14 just present the spectrum for RBW=1 kHz and RBW=10 kHz when both algorithm are turned on. All the spurs are effectively suppressed and the only noticeable spur in Fig 4-13 is at 40 kHz offset. From Fig 4-14 we can see it doesn't degrade the system's performance much and with even the worst TDC resolution, the spectrum fits the specification requirement.

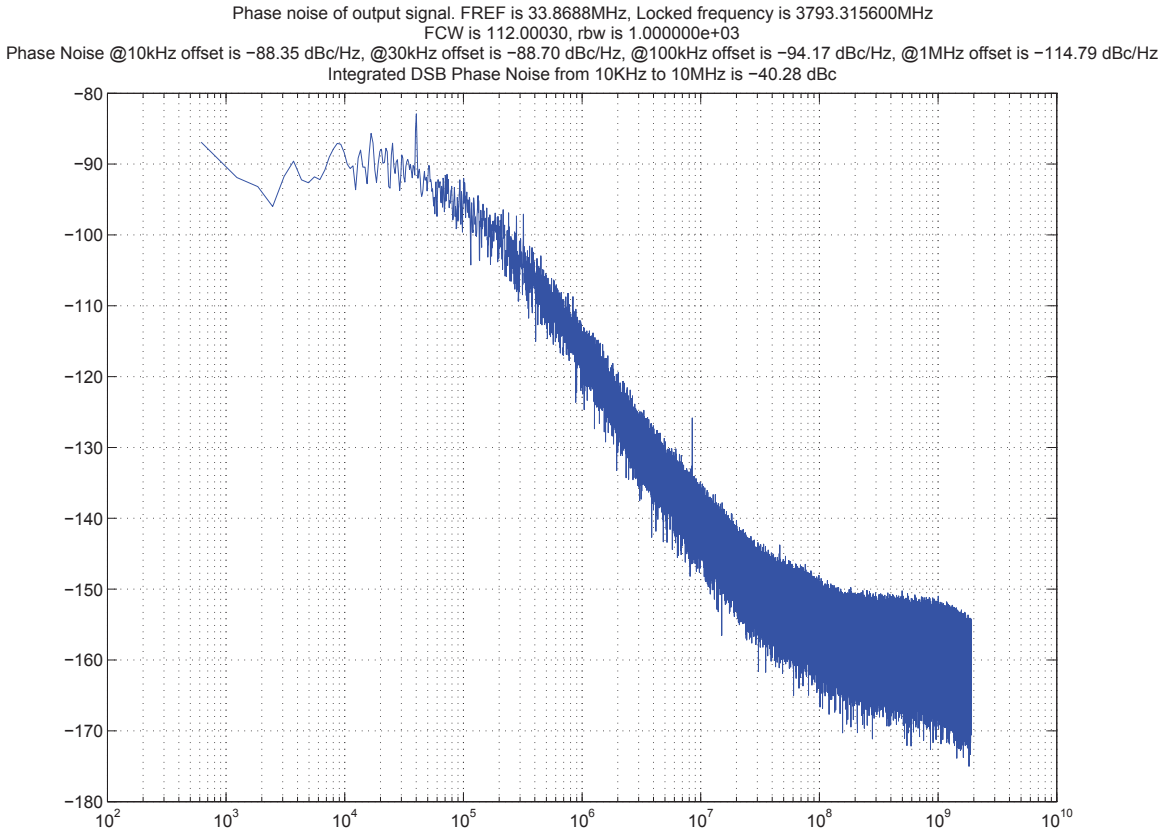

Figure 4-13: ADPLL spectrum for CKV frequency of 3793.3156 MHz, Phase rotation and FREF dithering algorithm activated, RBW=1 kHz.

The PHE transient with the two algorithms activated is shown in Fig 4-15. The PHE signal is well randomized compared to Fig 4-6, which illustrates why spurs are well suppressed.

What's more, the channel center frequency step size for the WiMAX application is 250 kHz or 100 kHz[27]. Our ADPLL loop bandwidth is within 100 kHz. Thus for every integer N, only one or two channels featuring a FCW value close enough to the integer may give rise

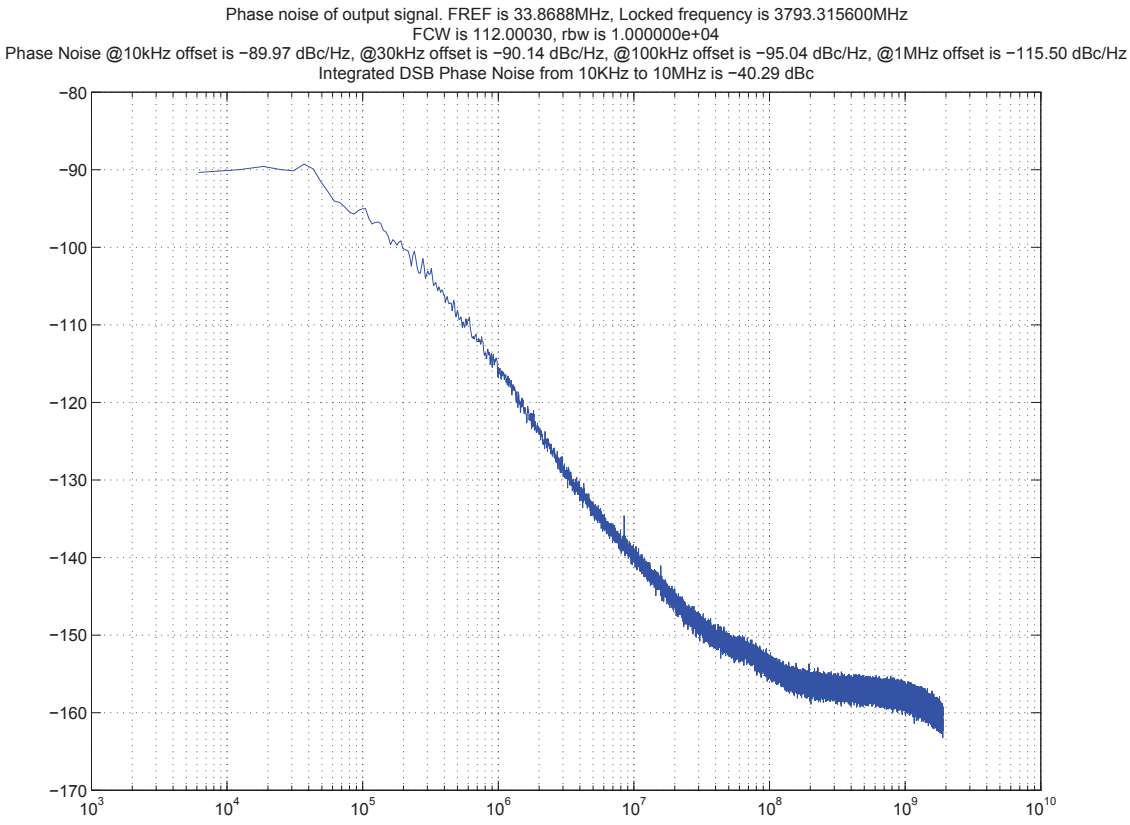

Figure 4-14: ADPLL spectrum for CKV frequency of 3793.3156 MHz phase rotation and FREF dithering algorithm activated, RBW=10 kHz.

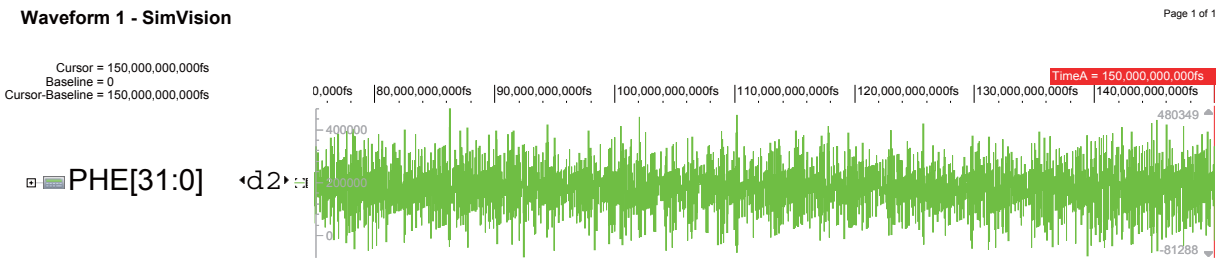

Figure 4-15: PHE Transient for CKV frequency of 3793.3156M (with phase rotation and FREF dithering on).

to the spurs. Phase rotation and FREF dithering will be applied to these channels. For most cases, the two algorithms can be deactivated and we have less in-band phase noise as well as simpler loop topology.

Chapter 5

The ADPLL Top Level

In previous chapters, the ADPLL architecture and algorithms have been introduced, with the insight into building blocks, simulation methodology, performance analysis and advanced algorithms. To forge these fancy ideas into a feasible solution (finally a working chip) takes significant effort at the top level of the system. In this chapter, we present the top-level schematic and then deal with various issues of the ADPLL top level. Extensive top level simulations are also performed to make sure the system works as we desire.

5-1 Top-Level Schematic of ADPLL System

The top level of ADPLL is divided into three parts: *analog core* (ACORE), *digital core* (DCORE) and AnalogIO. The ACORE part contains all the blocks to be handcrafted, i.e. their schematic and layout are designed manually. The DCORE part has all the blocks to be synthesized automatically. The AnalogIO part contains the IO rings for ACORE, which is not covered in the scope of this thesis and is ignored here.

Fig 5-1 is a schematic of ACORE and Fig 5-2 is a schematic of DCORE. Due to the limited space, the ACORE/DCORE interfaces are simplified here. Their interface signals are listed exhaustively in Table B-1 and Table B-2 as a reference.

In Fig 5-1, the Phase Rotator block is the implementation of the phase rotation algorithm as discussed in Section 4-3. It contains a simple Gray counter and a multiplexer with the I/Q signals from DCO (Ip,Qp,In and Qn). For its interface, PR_en is to enable/disable the phase rotation algorithm and PR_rst_a will reset the output of the phase rotator to the default input signal (Qn). The CKV test module block has a divide-by-16 divider and a multiplexer for the feedback CKV which are shown at the bottom left corner of Fig 2-7. CKV_SEL is to choose the feedback CKV signal between the Phase Rotator output and

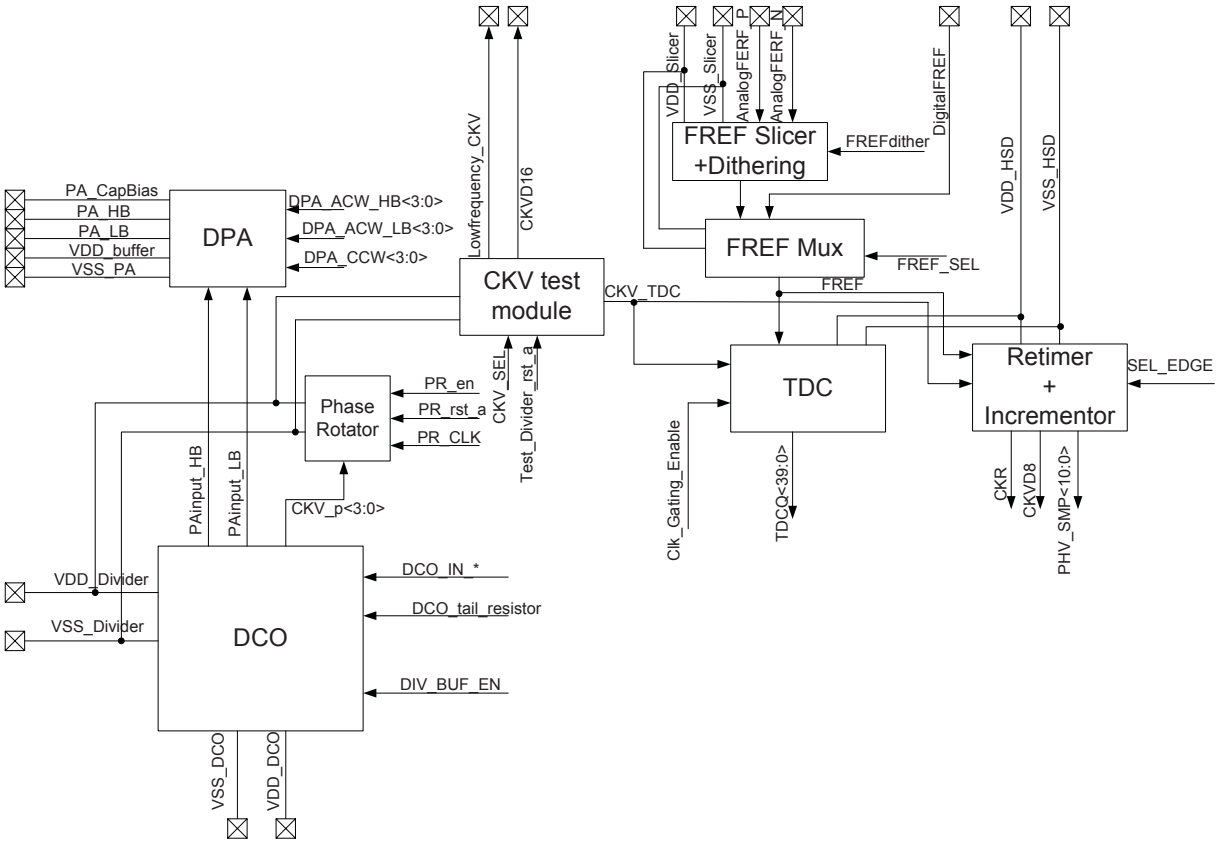

Figure 5-1: Top Level View of ACORE in ADPLL.

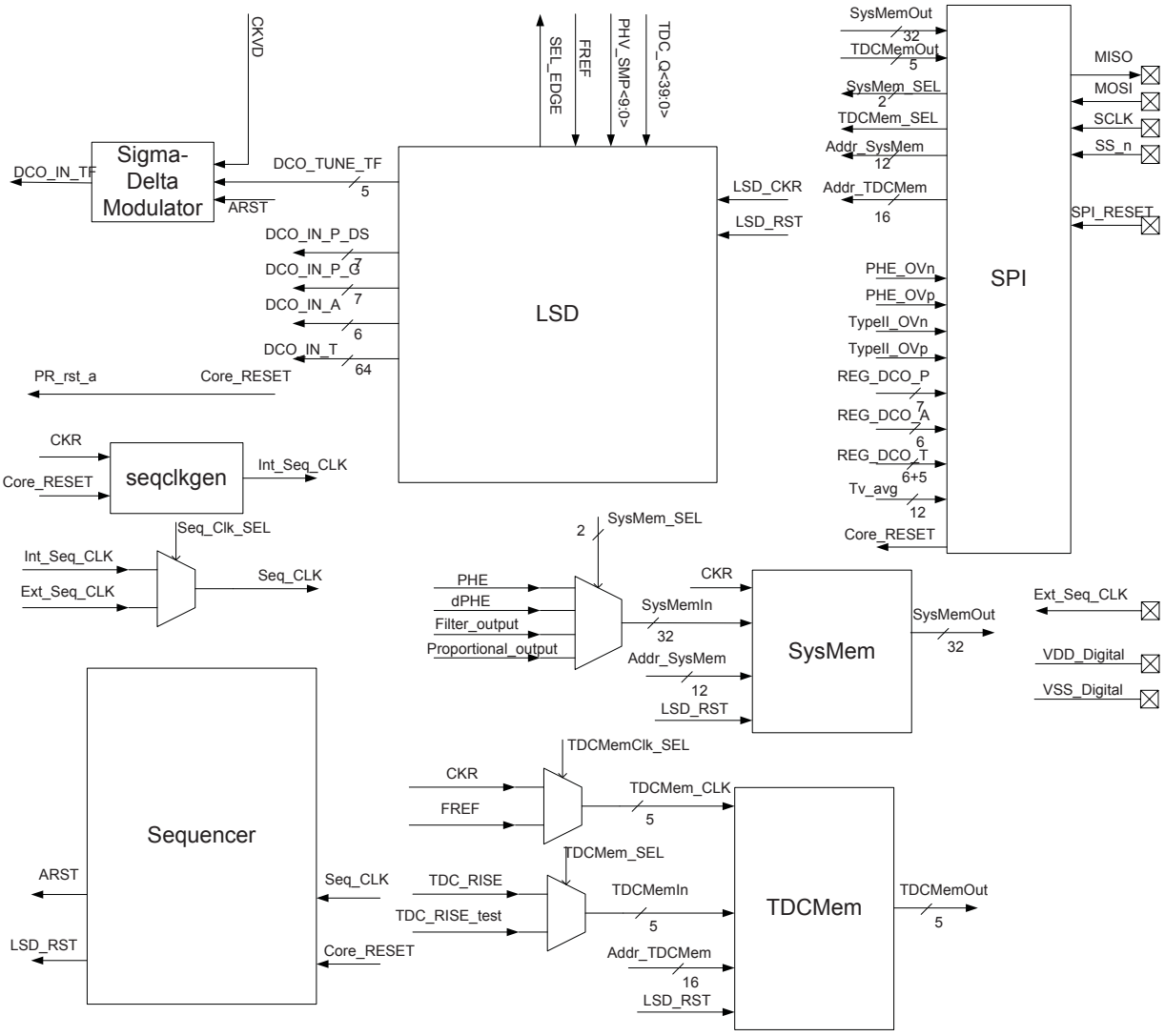

Figure 5-2: Top Level View of DCORE in ADPLL.

the off-chip input signal Lowfrequency_CKV. All other blocks in the ACORE have been already introduced. Table 5-1 lists the supplies and grounds in the ACORE block. All VSS signals are 0 V and all VDD signals are 1.2V. The PA_CapBias voltage is 0.6 V. The idea of using so many supplies/grounds is to make sure sensitive blocks can have a clean supply for the performance requirement (like the DCO core) and to be able to measure the power consumption of every block separately. However, since the transistor-level design of the blocks in ACORE is not finished and the decoupling capacitors are not added, here in Table 5-1 the specifications of the tolerable voltage drop and the tolerable supply noise are not given.

Supply/Ground	Blocks/Modules connected
VDD_DCO, VSS_DCO	The DCO core in the DCO block
VDD_Divider, VSS_Divider	The buffers and dividers in the DCO block; Phase Rotator
VDD_buffer	The buffer stages in the DPA block
VSS_PA	The PA stages and the buffer stages in the DPA block
PA_CapBias	The capacitor banks in the DPA block
VDD_Slicer, VSS_Slicer	The FREF slicer + FREF dithering implementation
VDD_HSD, VSS_HSD	The TDC block and the Retimer+Incrementor block

Table 5-1: Supply and ground signal list in the ACORE.

In the DCORE schematic of Fig 5-2, the low speed digital logic as in Section 2-5 is encapsulated as the LSD block. Its interface signals are listed in Table B-3. The fractional part of the tracking bank tuning word goes to the Sigma-Delta modulator which is clocked by the high-speed clock CKVD. Besides these two blocks, DCORE has the sequencer block, the SPI block, the SysMem block and the TDCMem block. The last two blocks are mainly memories for test and we will refer to them later. All digital blocks have a supply of 1.1 V.

The sequencer block provides signals to trigger the events in the LSD block after the ADPLL loop starts up, like mode switchover, Ktdc normalization, gear-shifting, transition from type-I PLL to type-II PLL and etc. In the ADPLL normal working status, the CKR signal is valid. As shown in Fig 5-2, it is divided by 16 in the seqclkgen block and the output signal Int_Seq_CLK, which has a frequency of around 2 MHz, will be chosen as the sequencer's clock (Seq_CLK) by the signal Seq_Clk_SEL in the sequencer clock multiplexer. The sequencer has a 7 bit counter which counts the positive edge of the sequencer clock after start-up. This counter reserves value 0 for the ADPLL open loop operation and starts from value 1. We will refer to that later. When the counter value reaches a certain value specified by a certain input of sequencer from the SPI block, some output signal(s) will be activated/deactivated to trigger a planned event. For example, the sequencer input MEM_TIME_A2T signal is 5. Then when the counter value is 5, which is about 2 us after ADPLL starts up, the sequencer decides to tell the LSD block to switch from the acquisition mode to the tracking mode. At that time, the CTL_PLL_A signal as in Fig 2-25 goes from high to low and the CTL_PLL_T signal goes from low to high. The CTL_SRST_T signal

becomes low to disable the synchronous reset of the OT block. ARST will also be low to enable the operation of the Sigma-Delta modulator. When the counter value reaches the full scale, i.e. 127, it maintains this value rather than wraps around. Via this, the sequencer has a time resolution of around 0.5 us and a time span of around 60 us. The interface signals of the sequencer are listed in table B-4.

It's worth notice that only when the DCO block starts oscillation and the retimer works can we have a valid CKR. To provide a clock for the sequencer when these conditions are not satisfied, say before ADPLL starts up, here in Fig 5-2 an off-chip signal Ext_Seq_CLK is provided as a clock. The signal Seq_Clk_SEL from the SPI block is made high to select this external clock when CKR is not ready. If a valid CKR is delivered, the signal Seq_Clk_SEL is made low to select Int_Seq_CLK as its clock. In this way the sequencer is always with a valid clock and can control the operation in the LSD block.

The SPI block is an implementation of the slave device in the *Serial Peripheral Interface* (SPI) Bus[28, 29] with some registers. The SPI Bus is chosen as it is simple, flexible and can have a high data transmission rate (can work at MHz or tens of MHz). The SPI block is clocked by the external clock SCLK. Before the start-up of ADPLL, the SPI block will read the desired information from the off-chip controller via the signal line MOSI and then save the information to registers. The registers specify the select/enable/reset/timing signals to the sequencer block or the LSD block or the blocks in the ACORE part. Therefore the working mode and the status of ADPLL can be flexible. During the transient of ADPLL operation, some signals within the ADPLL system will be saved to registers or memories. The SPI block can read the saved signals and send them to the external controller via the signal line MISO, either for monitoring purpose or for test purpose. Its importance will be revealed as we go through this chapter. The interface signal list of the SPI block is given in Table B-5. One extra input signal SPI_RESET from pad is used to reset the registers associated with the SPI block. Then the ADPLL is working in the default mode.

5-2 Back-Up Modules in the ADPLL System

In our ADPLL design, some back-up modules are introduced to make the system more robust, which helps reduce the risk of this prototype design.

For the ADPLL input, as seen in the ACORE schematic, except for the normal analog FREF signal from the FREF slicer we also have Digital_FREF signal generated off chip. This Digital_FREF can be noisier than Analog_FREF signal and when it's chosen as FREF for ADPLL loop, the in-band phase noise of the system may degrade. Nevertheless, if the off-chip crystal oscillator or FREF slicer doesn't work normally, Digital_FREF is still available and can be chosen as FREF for the system by pulling down FREF_SEL, which is the control signal of FREF multiplexer. In that way the system can still work.

For the ADPLL output, a divided-by-16 divider in the CKV test module block in the ACORE schematic is used for the feedback CKV. The frequency of divided output is no

more than 250 MHz and it shall go off-chip without special effort[1]. If the output frequency is still too high to go off-chip, the division ratio of the divider can be made larger than 16, say 32 or 64. A spectrum analyzer can be connected to this off-chip signal and read out the frequency of CKV. The jitter of this divider only has a significant influence on the far-out noise. Thus we can estimate of the close-in phase noise performance for the CKV signal via this divider output. In this way some useful information about the ADPLL output will be retrieved even when DPA is not functional or when DPA adds some strong interference to the system.

5-3 Test Plan for ADPLL

The test plan is an essential part to a successful design on silicon. Under the constraints of limited pads, one shall find a way to diagnose the whole system, to gather enough information for the system evaluation, for system debug, and even as clues on how to improve the system.

As the ADPLL features digital I/O for every essential block, some digital features are included in its test plan. Registers and memories are deployed here to take advantage of the high logic/memory density in 40 nm CMOS process. Then the SPI block is utilized to read/write the contents of these registers and memories.

5-3-1 System Snapshot with Digital Logic

For ADPLL, the compared phase/frequency signal and the DCO capacitor bank control signals are digital. These signals render the insight on the system status and facilitate the system debug, if that is needed.

For example, if there are significant spurs at the output spectrum, one can look at the transient of the PHE signal to see whether it has a similar shape as in 4-6. If so, we can apply the phase rotation and FREF dithering algorithms to suppress these spurs. Also, one can look at the PHE or dPHE signal to make sure there is no PHE spike event due to the mismatch of the TDC path and the incrementor path. Moreover, by observing the tuning word for tracking bank, NTW_T as in Fig 2-26, one can know whether gear-shifting circuits / IIR filters / the type-I to type-II transition circuitry work as we expect or not.

Compared to analog peers, probing into digital signals will not perturb the normal working of system. They can be accessed either out-of-chip in a real time way or processed by the BIST circuitry which is often used in SoC or stored in the memory as a snapshot. Here in our project, most digital signals feature long word length (32 for PHE and dPHE). A real-time monitoring will take a lot of pins, increase the chip area and also increase the

[1]For example, *current mode logic* (CML) is often used for really high speed signal at on-chip to off-chip interface.

test cost. Also this chip has no DSP for digital processing and BIST introduces too much overhead on complexity. Therefore we prefer to save a snapshot of the ADPLL system in memory and then read it out slowly via the SPI block.

The signal chosen for the system snapshot is chosen from PHE, dPHE, Filter_output and Proportional_out. The latter two signals are show in Fig 2-26. The 2-bit control signal, SysMem_SEL, select which signal to be forwarded to the memory for system snapshot (SysMem). All these signals are updated per CKR cycle. Thus the clock for the memory of SysMem is also CKR. The SysMem block can store 4096 cycles of one signal, i.e. the address signal for SysMem, Addr_SysMem, is 12 bit. As the frequency of CKR is about 30-40 MHz, the snapshot has the transient of the chosen signal for a duration of more than 100 us. That is sufficient to cover the frequency settling transient of ADPLL and is larger than the time range controlled by the sequencer.[2] Even spurs located at pretty low frequency offset (on the order of several tens of kHz) can be observed if PHE is the stored signal.

The content in SysMem gives the detail information about the system working status. After the memory is filled up by the chosen signal, the external SPI controller tells the SPI block the address of the word in SysMem to be read out, which is the signal Addr_SysMem. This word is selected, read in by the SPI block to some register and then sent to the external controller via the signal line MISO. This reading operation will take several cycles of SCLK. So to get the whole content of SysMem it may take several minutes. Thus it would be nice to know about some brief system information before we spend time on reading out SysMem content. In our design some registers are added to serve this purpose.

When ADPLL has settled, the tuning word of capacitor banks (PB, AB and TB) can be saved in registers. The sequencer will write REG_DCO_P and REG_DCO_A as in Fig 2-25 (also there is a REG_DCO_T signal with 6 bit integer part and 5 bit fractional part) to registers at the time specified by the signal MEM_TIME_REGDCO. These words can be read out and used as MEM_DCO_P and MEM_DCO_A next time when ADPLL is to output a frequency close to current working frequency. In this way the frequency settling transient is shortened and loop stability is also improved.

There are some overflow registers as the outputs of the LSD block (PHE_OVp, PHE_OVn, typeII_OVp, typeII_OVn). The accumulation operations from dPHE to PHE as in Fig 2-22 and in the integral path of tracking bank loop filter as in Fig 2-26 are monitored. When these accumulation operations cause a overflow, either in positive direction or negative direction, the corresponding register will record this event (the output goes from low to high). In this way we have a brief estimation on the status of the ADPLL system. For example, when PHE_OVp is high, most probably the ADPLL system is locked to a frequency lower than the desired frequency.

The values in these registers are directly connected the SPI block. So the off-chip controller can specify the SPI block to read these values out after ADPLL has settled. That operation

[2]If the SysMem clock is too area consuming for the chip, one can reduce the address signal to 11 bits. That almost satisfies our requirement to monitor the system start up transient.

is much faster than the SysMem reading out procedure as it only needs several cycles of SCLK.

5-3-2 DCO Open-Loop Test

The DCO block lies at the heart of the ADPLL system and its performance is critical to the whole system. Thus it is desirable that DCO can be test alone. To achieve that, we need to be able to open the ADPLL loop and configure the DCO tuning words. These are rather straight forward in an all-digital environment. The timing of the LSD block is specified via the SPI block and controlled via the sequencer outputs. As mentioned in Section 5-1, the timing of the sequencer is relied on an internal counter, which starts from 1 rather than 0. When we are to open the ADPLL loop, all signals with timing information from SPI are set to 0. Thus the counter output can never be equal to any of the timing values specified by the SPI block. The OP, OA and OT blocks are always reset and the outputs of loop filter are nullified. Yet the tuning words of DCO can be changed as desired via the SPI output MEM_DCO_P, MEM_DCO_A and MEM_DCO_T. As in Fig 2-25, if the loop filter outputs are nullified, these signals will determine the DCO tuning words alone. The tuning word can be modified on the fly to sweep the DCO frequency curve. One can refer to the measurement set-up at Page 58 of [30]. There the SPI is controlled by the FPGA, which is connected to the PC. The PA output signal goes to the spectrum analyzer and its output is also fed to the PC via the GPIO bus. First the PC delivers the tuning word information to the SPI via FPGA. The DCO receives these new tuning words and settles to the new frequency. The spectrum analyzer has the phase noise and the frequency information and then sends it to the PC. In that way, the DCO test is done automatically. Also the DCO_tail_resistor can be modified to measure the DCO phase noise performance with respect to the bias current.

Actually in our ADPLL the DCO open-loop test would be performed before the measurement of ADPLL closed-loop performance. As we mentioned in Section 2-5, in the blocks of GP/GA/GT the signals will be multiplied with some coefficient for the Kdco normalization. Then we need to have Kdco in advance, as can be easily obtained here. Say, here we fix the values of MEM_DCO_P and MEM_DCO_A and then measure the output frequency of the ADPLL as the integer part of MEM_DCO_T goes from 0 to 63 (the fractional part MEM_DCO_T is set to 0). Thereafter we can extrapolate the slope for the curve of frequency versus tuning word. Say the slope gives a slope of $\hat{K}_{dco}$ = -51.3 kHz/LSB and the reference clock frequency is f_R = 33.8688 MHz. Then $\frac{f_R}{\hat{K}_{dco}}$ is about 660.21. In the ADPLL closed-loop operation, this information is provided via the signal MEM_GAIN_T, which only has 11 bits for integer part. Thus MEM_GAIN_T will be set to 660 when the ADPLL is to settle to the frequency within the range of this curve.

5-3-3 TDC Test Plan

TDC is another essential block in the ADPLL system. From Section 3-4 we know that its resolution has a significant contribution to the in-band phase noise of the ADPLL system. From Section 4-3, we know the TDC mismatch and the quantization effect are the reasons for spurs in the ADPLL output spectrum. Therefore, it would be nice if the TDC resolution and its mismatch can be measured.

The TDC resolution can be obtained during the normalization operation of the ADPLL system. In Section 2-5, we accumulate |TDC_FALL-TDC_RISE| and then do averaging to estimate K_{tdc}. Notice that K_{tdc} is just the fixed-point representation of the estimation for $\frac{T_{inv}}{T_{CKV}}$, as the desired frequency is known, the TDC resolution T_{inv} is also known. For example, say the desired frequency is 3800 MHz and the accumulation output for TDC normalization is 2740, then we know $\frac{T_{inv}}{T_{CKV}} \approx \frac{2^7}{2740}$. We have the TDC resolution $T_{inv} = \frac{1}{3800\text{ MHz}}\frac{2^7}{2740} \approx 12.29$ ps. This estimation achieves an accuracy of 2%.

To characterize the TDC mismatch, one needs to deliver the TDC transfer function as in Fig 5-3 [15, 31]. An easy way is the density test method, which is done in the ADPLL closed-loop operation. When we specify a desired frequency which is not a near-integer N case, the word TDC_RISE as in Fig 2-23 will sweep the TDC codes. If there is no mismatch between every TDC cell, the probability of TDC_RISE for every TDC code is the same. Then all TDC codes shall have almost the same density in the histogram of the TDC_RISE output. Nevertheless, when the mismatch kicks in, the density of the TDC codes shall differ. The density ratio between these codes reflects the mismatch of the TDC cells. For example, the average density of the TDC code is 25000 and the density of code 5 is 27500. Say the TDC normalization operation indicates that the TDC resolution is 12.5 ps. Then we estimate the delay of the fifth cell in the effective TDC chain (the 9th cell in the whole TDC chain) as $\frac{27500}{25000} \times 12.5\text{ ps} = 13.75$ ps.

For this density test method, enough cycles of TDC_RISE word shall be recorded for the histogram statistics. Here the TDC_RISE signal is stored into the TDCMem block for 65536 CKR cycles, i.e. the address signal for TDCMem (Addr_TDCMem) is 16 bits. Then the content of TDCMem is read out from the SPI block. The TDC_RISE value for the first 5536 cycles are thrown away to make sure we use the TDC_RISE value only when the ADPLL has settled to the desired frequency. This measurement is done for 9 times and then we do the density test for all the TDC_RISE words stored in these measurements. The histogram, as in Fig 5-4, has an accuracy of around 2% when checked against the TDC modeling in list A-2.

This density test method is very convenient as it can be done on the fly. However, it has several disadvantages. Such test is based on the premise that ADPLL functions correctly in the closed-loop mode, which means other modules like DCO, Retimer+Incrementor and the digital logic shall be all OK. Another reason is a lot of measurements need to be done for a very accurate characterization of the TDC mismatch, say with 1%. That adds to the test cost. The most important reason is, as we look at Fig 5-4, the histogram only tells us

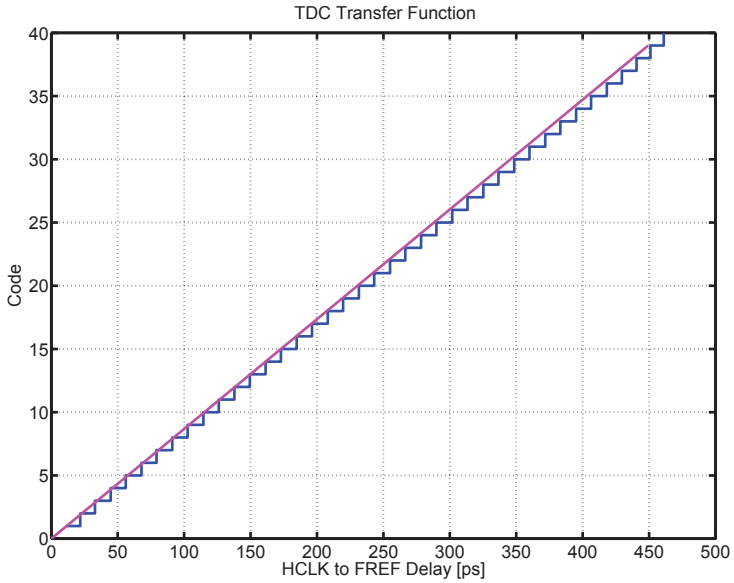

Figure 5-3: TDC transfer function.

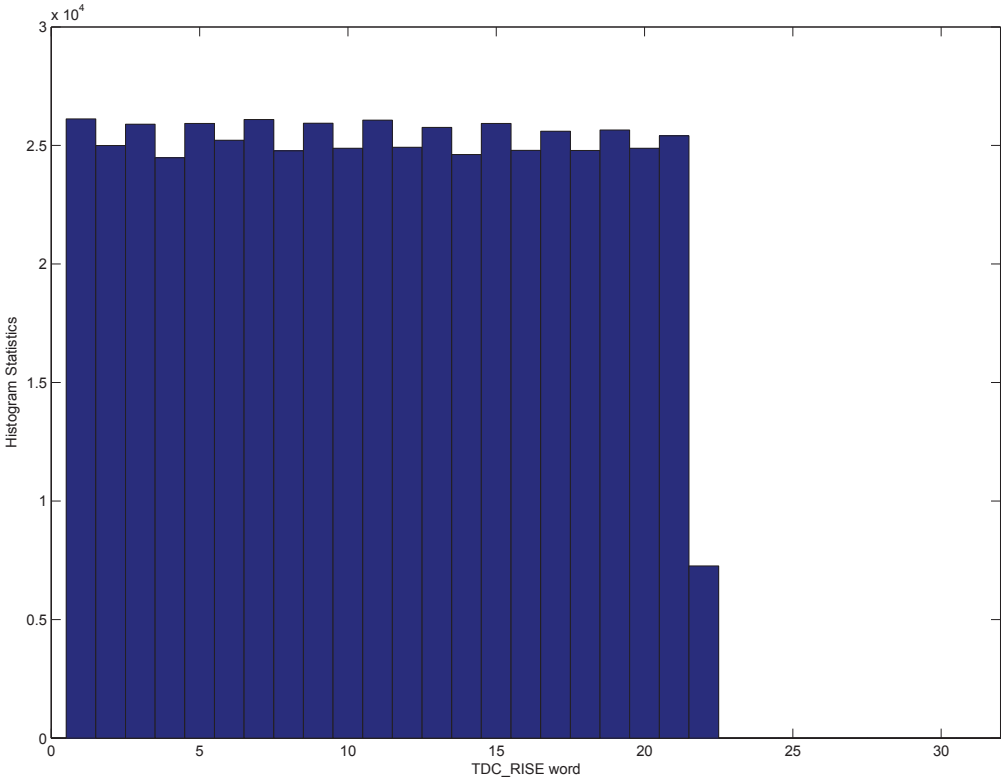

Figure 5-4: The TDC closed loop test (CKV frequency of 3800 MHz).

the mismatch from code 1 to code 21, which is cell #5 to cell #25 in the whole TDC chain as shown in Fig 2-10. Since the delay of the effective TDC chain, from cell #5 to cell #36, is longer than one CKV cycle, the closed-loop operation cannot cover all the TDC codes. Therefore some open-loop test method is needed to solve the above problems.

The open loop test set-up is shown in Fig 5-5. In the figure the pins of DigitalFREF and Lowfrequency_CKV are utilized. So we shall let the multiplexer for FREF to choose DigitalFREF rather than AnalogFREF and let the multiplexer in the CKV test module to choose Lowfrequency_CKV rather than the feedback CKV. Thus the ADPLL loop is open. The FREF_pulse signal is a low frequency clock signal. In the simulation it is set to be 33.8688 MHz yet can be another frequency. The Ext_CLK signal is a clock signal with the frequency higher than the FREF_pulse signal yet still much lower than the feedback CKV signal. Here we choose its frequency to be 100 MHz.

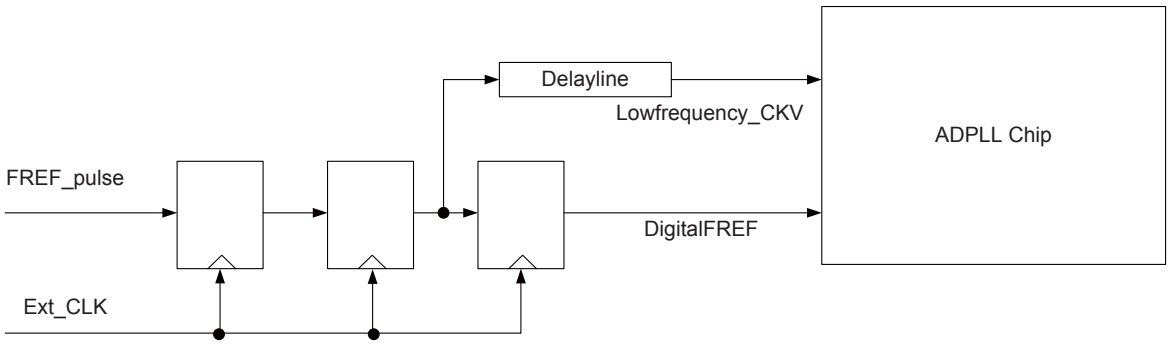

Figure 5-5: TDC open-loop static test.

In this test structure, FREF_pulse is sampled by the Ext_CLK signal. Thus the rising edge of DigitalFREF signal lags the rising edge of the output of the second flip flop by one Ext_CLK cycle T_{ext}. On the other hand, the rising edge of Lowfrequency_CKV lags the rising edge of the output of the second flip flop by the delay of the delay line τ. When $T_{ext} = \tau$, the rising edge of DigitalFREF and Lowfrequency_CLK just occur at the same time. So when the delay line is fine tuned so that τ is a little bit smaller than T_{ext}, say 200 ps, the rising edge of Lowfrequency_CKV leads that of DigitalFREF by 200 ps. This timing difference will be detected by the TDC and from Q(5) to Q(36) a transition from 1 to 0 will be detected.

Nevertheless, such transition cannot be processed properly by the normal TDC decoding logic as in Fig 2-23. The reason is that now FREF and CKV have the same frequency and the oversampling circuit in Fig 2-12 will not function. Thus there is no valid CKR[3] and the latch in Fig 2-23 doesn't work. The way to route such problem is to use negative edge of FREF to clock the TDC decoding logic rather than latching it. The decode output will

[3]Thus at this time the sequencer shall be clocked by the Ext_Seq_CLK signal.

be sent to TDCMem which is clocked by the rising edge of FREF, which is the reason of using a multiplexer between CKR and FREF for the clock of TDCMem.

As is noticed, assuming that the DigitalFREF and Lowfrequency_CKV signals have no jitter, once the frequency of Ext_CKV f_{ext} and the delay of the delay line are fixed, the timing relationship of the rising edge of these two signals are also fixed. The TDC decoded output shall also be unchanged. So it is called a static test. However, since all signals have jitter and the delay line has a variation, the TDC decoded output changes. We save the TDC decoded output for enough cycles (65536 cycles would be far enough) and then choose the code with the highest density as the averaged TDC output, which is called majority voting algorithm.

To sweep the timing difference between the two rising edges for characterizing the TDC transfer function, the frequency of Ext_CKV will increase/decrease in a uniform step. Then T_{ext} changes and finally the averaged TDC output also changes. Here f_{ext} goes in a step of 500 Hz. As $\frac{\Delta f}{f} = \frac{\Delta t}{t}$, when f_{ext} is around 100 MHz, the time step of $T_e xt$ corresponding to the frequency step is 50 fs, which is less than 0.5% of one TDC cell delay. Then as we sweep f_{ext}, transition events of the averaged TDC output would indicate the delay of a certain TDC cell and thus the resolution as well as the mismatch of TDC.

For example, in our simulation, from Fig D-1 and Fig D-2, one can know that the TDC average output goes from 17 to 16 as f_{ext} goes from (100 MHz-203*0.5 kHz) to (100 MHz-202*0.5 kHz). From Fig D-3 and Fig D-4, one can know that the TDC average output goes from 16 to 15 as f_{ext} goes from (100 MHz+38*0.5 kHz) to (100 MHz+39*0.5 kHz). Then the delay of cell #20 can be estimated as $\frac{1}{100-202*0.5*0.001} - \frac{1}{100+39*0.5*0.001}$us = 12.060 ps. The model of TDC in list A-2 says the delay of cell #20 is 0.968*12.5 ps=12.1 ps. The accuracy is around 0.3%. Similarly, we have the TDC average output transition from 15 to 14 when f_{ext} goes from (100 MHz+288*0.5 kHz) to (100 MHz+289*0.5 kHz). Then the delay of cell #21 is estimated as 12.480 ps. The estimation accuracy is 0.16%.

To summarize, this open loop static test method can achieve an accuracy of 1% for the TDC transfer function characterization. Then we can get a very accurate estimation of the resolution and mismatch of TDC. It shall be noted that we can modify the frequency step of the measurement to either increase the accuracy or reduce the measurement time.

5-4 Operation Modes of ADPLL

In Chapter 2 and Chapter 4, we have already touched into the operation of the ADPLL system, mainly with an architecture view. For example, for the frequency settling transient, the ADPLL system needs to go through the PVT mode, the acquisition mode and the tracking mode. It needs to perform ZPR, gear-shifting and transition from the type-I PLL to the type-II PLL at the certain time. Here we illustrate the operation modes of ADPLL from a top level view. The system is treated as a chip and the insight is given on making

the system an entity with the building blocks, the algorithms and the test plans mentioned in this thesis.

5-4-1 ADPLL Start-Up

Before the ADPLL system starts up, the SPI_RESET signal is high to reset the registers in the SPI block. The registers contain all the information to control the ADPLL system operation, as listed in Table C-1. Thus the whole system are reset: the DCOtailres signal in Fig 2-4 is 0 so that DCO doesn't oscillate; the sequencer chooses Ext_Seq_CLK as its clock with its counter reset to 1; the operation of the LSD block is reset by the control signals from the sequencer; the SysMem block and the TDCMem block are also reset and no valid signals are dumped into the memory.

After that silent period, the SPI_RESET signal is set low. Then via the MOSI signal line, the external controller writes the values of the registers in the SPI block to start up the system. Firstly, the needed analog modules in ACORE will be wake up. DCOtailres is set non-zero to turn on the DCO core and deliver a certain bias current for it. The phase rotator is either enabled or disabled via PR_en in Fig 5-1. This also applies to the FREF dithering, which is controlled by the FREFdither signal. The certain signals for controlling the dividers/buffers in the DCO block and the tuning words of DPA are set as we desire. For example, if the LB output of ADPLL is needed, we shall enable DIV_1_5_EN, LB_BUF_EN and LB_PA_DIV2_EN in Fig 2-2 as well as DPA_ACW_LB signal in Fig 2-1. Yet we can also disable DPA_ACW_LB signal and choose the CKVD16 signal as in Fig 5-1 to measure the ADPLL output frequency without turning on DPA. What's more, if one is to perform the open-loop static test of TDC, the SPI block just needs to write the certain bits of the register ckvfrefsel as in Table C-1 to FREF_SEL and CKV_SEL signals low. In that way, the Lowfrequency_CKV signal and the DigitalFREF signal are chosen, ready for the TDC open-loop test.

After the SPI block invokes the analog modules needed, the registers of TIME* (TIMEP, TIMEP2A, TIMEKtdc, etc.) will be loaded with values that specify the timing of the loop logic to the sequencer. The values of these registers are sent to the corresponding inputs of the sequencer, like MEM_TIME_P, MEM_TIME_P2A, MEM_TIME_Ktdc_Norm, etc. Then the internal reset signal Core_RESET as in Fig 5-2 is made low to bring the counter of the sequencer to work. Then the loop logic is enabled and the whole system starts up.

5-4-2 ADPLL Test Mode

From previous discussions one can know that the ADPLL loop needs to be open for the test of DCO or TDC. And that is achieved by setting all the values of TIME* registers to 0 before Core_RESET signal is made low. Then the sequencer will not enable the PVT/acquisition/tracking path in the digital logic. The phase accumulator as in Fig 2-22

is reset by the Reset||ZPR signal. Therefore the digital logic is reset and the ADPLL loop is open. If the MEM_DCO* signals (MEM_DCO_P, MEM_DCO_A and MEM_DCO_T) are configured via the corresponding registers MEMDCO* in the SPI register map and the ADPLL output is monitored, the DCO open loop test is just performed. If the FREF_SEL signal chooses DigitalFREF and the CKV_SEL signal chooses Lowfrequency_CKV, the TDC open loop test can be done with the test set up as in Fig 5-5.

As in the ADPLL normal operation, during the settling transient, the system will traverse the PVT mode, the acquisition mode and the tracking mode. In the tracking mode, the gear-shifting event is triggered and ADPLL changes from type-I PLL to type-II PLL. It helps if we can stop the system at a certain phase rather than let it go through all the events. For example, when the ADPLL system stays as a type-I PLL, we will see how the up-converted flicker noise of DCO hurts the in-band phase noise performance of the whole system. Or we can enable/disable the gear-shifting event and compare the phase noise performance for the two cases. This purpose can be served by setting some TIME* registers to 0. Say we set TIME / TIMEP2A / TIMEKtdc / TIMEA2T as non-zero values while TIMEGS / TIMELambda / TIMEType are set to 0, the gear-shifting mechanism is disabled and ADPLL stays as a type-I PLL.

5-4-3 ADPLL Closed-Loop Mode

When all TIME* register are configured as non-zero values, ADPLL works in a closed-loop way and will traverse all the events to the final normal working status (tracking mode, Gear-shifting triggered, IIR filter banks enabled, type-II PLL). Yet some preparations shall be done for a correct simulation or measurement. Thus we have a simulation flow for the ADPLL normal operation.

The simulation flow for the ADPLL normal operation is shown in Fig 5-6. We have mentioned above that K_{dco} can be measured in the ADPLL open-loop mode. In the simulation, it is done in another way. When the desired frequency is fixed, the frequency resolution K_{dco} can be calculated as the result of equation (2-2) divided by 4. Then we can know the coefficients for DCO Gain Normalization $\frac{f_R}{\hat{K}_{dco,p}}$, $\frac{f_R}{\hat{K}_{dco,a}}$ and $\frac{f_R}{\hat{K}_{dco,t}}$ with a certain reference clock. For example, if the desired frequency is 3550 MHz, the frequency resolution is

$$\hat{K}_{dco,t} = -\frac{1}{4}2\pi^2 L f^3 \Delta C = -\frac{1}{4}2\pi^2 \times 325\text{pH} \times (3550\text{MHz} \times 4)^3 \times 10\text{ aF} = 45.9\text{ kHz} \quad (5\text{-}1)$$

then for reference clock at 33.8688 MHz, we have $\frac{f_R}{\hat{K}_{dco,t}} = \frac{33.8688\text{ MHz}}{45.9\text{ kHz}} \approx 737.9$. Thus during ADPLL start-up transient, the value of MEM_GAIN_T will be written as 738. These values from the calculation will be written into the corresponding registers MEMGAIN*.

Still, K_{tdc} is also needed. As stated in Section 5-3-3, that is done by the TDC normalization circuitry, which is triggered when the ADPLL system has entered the tracking mode. This

normalization takes 256 FREF cycles, which is around 8 us. Thus before the long simulation for characterzing the ADPLL phase noise, a short simulation shall be performance for the TDC normalization. Here a simulation for 150 us is done. When the simulation ends, the SPI block reads the value from the Tv_avg register which is the sum value of |TDC_FALL-TDC_RISE| for 256 cyels. Then we do a calculation like Equation 2-7. The result is save to the registers Ktdc_0 and Ktdc_1. Then we have an accurate K_{tdc} and do a rather long simulation of 5 ms for the ADPLL phase noise performance.

The timestamp information of the CKV rising edges in the simulation are stored in a file. After the simulation end, Matlab just reads the timestamp information and do a first order polynomial curve fitting for these timestamps. Note the very first part of the timestamp information is thrown away as at that time the system has settled to a frequency. The slope of the fitting is just the average period of CKV. The CKV frequency is know and compared with the desired frequency to get the frequency deviation. The deviation of every timestamp point from the fitted polynomial indicates an instantaneous time error. The error divided by the average period times 2π delivers the phase error. Then we can do FFT to get the spectrum of phase error as $W_\phi(f)$ in Section 1-2-3. Then the phase noise spectrum $\mathcal{L}(f)$ is just that minus 3 dB.

5-5 The ADPLL Top Level Simulation

With the top level issues wrapped up in a nutshell, the ADPLL system is simulated and the performance is extracted from the timestamp information as stated in last section. Fig 5-7 shows the ADPLL top level test bench. The reference clock signal AnalogFREF_P and AnalogFREF_N are modeled in a similar way as the CKV signal in the DCO model. One difference is that the oscillating frequency is at f_R rather than f_v. The other difference is that we only model the white noise floor for the reference signals. Digital_FREF and Lowfrequency_CKV can be modeled in the way of Fig 5-5 for the TDC open-loop test. For the ADPLL normal operation we don't care how they are modeled. The Ext_Seq_CLK models a clock signal with the frequency of several kHz. The jitter performance of it is of no importance as it only provides the clock for the counter before the frequency synthesizer starts up. The SPI_RESET signal is high for the first 10 us of the simulation to reset the SPI block and then becomes low. After that the SPI master writes the control words to the SPI block via the MOSI signal line to configure the operation modes of the ADPLL system. The example code of the SPI master is in Section A-3. The edges of ADPLL output signals, including CKVD16, DPA_HB and DPA_LB, can be detected and stored in some file for the post-processing.

Here we simulate for different cases: the desired frequency is at high frequency (around 3800 MHz), middle frequency(around 3550 MHz) or low frequency (around 3300 MHz); the TDC resolution under slow corner (T_{inv} is around 12.5 ps), under typical corner (T_{inv} is around 11.5 ps) or under fast corner (T_{inv} is around 10.5 ps); the reference clock frequency f_R is 33.8688 MHz or 40 MHz. The near-integer N cases for the two reference frequencies

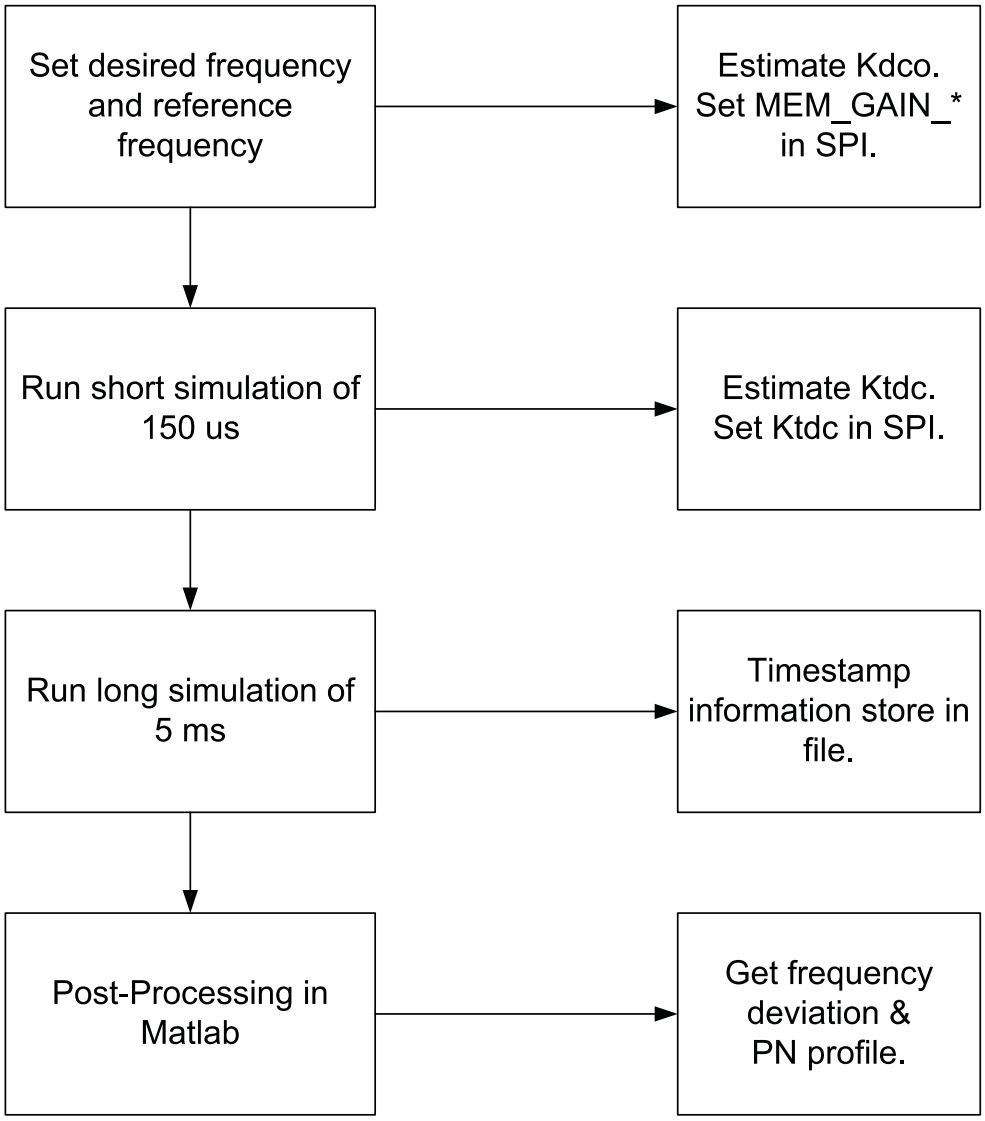

Figure 5-6: Simulation flow for the ADPLL normal operation

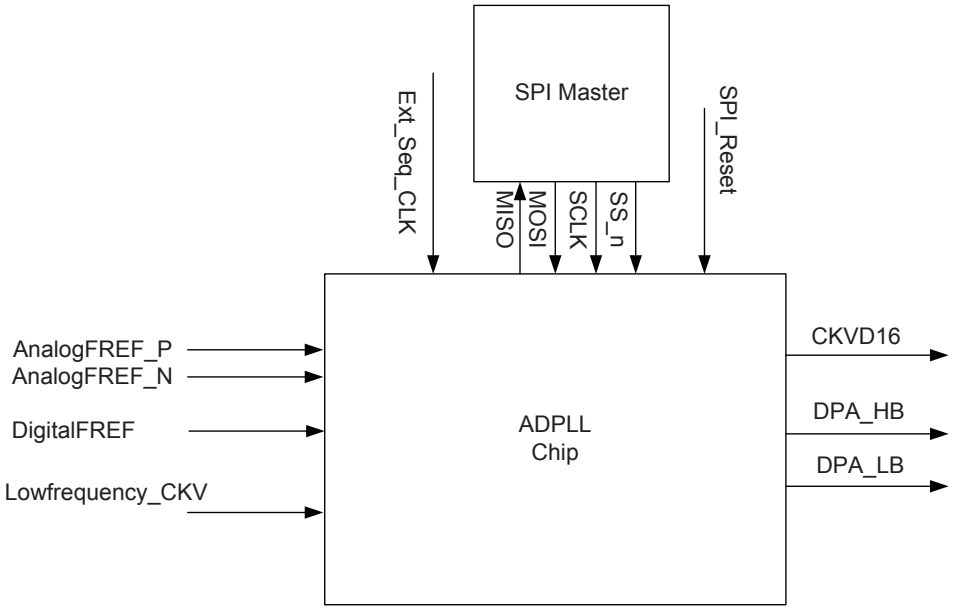

Figure 5-7: ADPLL top level test bench

are also simulated. f_v is 3556.214 MHz for the reference frequency of 33.8666 MHz and 3560.01 MHz for the reference frequency of 40 MHz. The phase rotation algorithm and the FREF dithering algorithm are enabled for the simulation of near integer-N cases. The phase noise profiles from the simulations are shown from Fig E-1 to Fig E-24. In these plots the resolution bandwidth RBW is 10 kHz. That is fine enough for the phase noise at frequency offset no less than 100 kHz. The spectrums with RBW of 5 kHz give the phase noise performance at frequency offsets closer than 100 kHz, which are not shown due to the limited space of this thesis.

The phase noise performance is summarized in Table 5-2.

Specification	Best performance	Worst performance
Frequency deviation	5.57 Hz	14.36 Hz
Spot noise @ 10 kHz	-96.06 dBc/Hz	-90.96 dBc/Hz
Spot noise @ 30 kHz	-95.08 dBc/Hz	-91.22 dBc/Hz
Spot noise @ 100 kHz	-98.84 dBc/Hz	-94.45 dBc/Hz
Spot noise @ 1 MHz	-120.97 dBc/Hz	-112.15 dBc/Hz
Integrated SSB Noise (1 kHz - 10 MHz)	-43.61 dBc	-40.13 dBc

Table 5-2: The ADPLL phase noise performance summary

From this table it is clear that the frequency step size specification and the integrated SSB noise specification as listed in Table 1-1 are met. The phase noise specification for the 10 kHz frequency offset is also met. The phase noise specification for the 100 kHz frequency offset violates the specification just by 0.55 dB in the worst case.

The frequency settling transient of ADPLL is also investigated. The timestamp information of CKV at the very first part of the simulation when ADPLL is settling to the desired frequency is processed by Matlab. The instantaneous period of CKV is compared with the desired period. The difference dT is plotted versus the time, which indicates how ADPLL settles to the specified frequency. However, as the CKV signal is noisy in time domain, the noise on the timestamp information may mask the frequency settling behavior when the CKV frequency get close to the desired frequency. Thus we also do averaging for the period of 100 CKV cycles. The CKV period is label as $T_v[k]$, k is just the index. Then we have the averaged sequence $T_{v,avg}[k] = (T_v[k]+T_v[k+1]+\cdots+T_v[k+98]+T_v[k+99])/100$. The difference of the average period value with the desired period value would filter out the noise and reveal the detail of the frequency settling.

Fig 5-8 and Fig 5-9 show the ADPLL settling transient in the PVT mode/acquisition mode and in the tracking mode separately. The upper subfigure in each plot is the difference of the instantaneous period and the desired period. The bottom subfigure in each plot is the difference of the average period and the desired period. The frequency settling begins when there is significant change of dT, say 36.48 us in Fig 5-8. After about 1 us, ADPLL switches from the PVT mode to the acquisition mode. The ZPR algorithm is triggered. We know that the reset pulse of OP_ZPR lasts several cycles. That's the reason why the dT value is almost fixed at 37.41 us. When OP_ZPR goes low, the dT value continues to approach zero and after around another 1 us, ADPLL enters the tracking mode.

In the bottom subfigure of Fig 5-9, one can see that when ADPLL switches to the tracking mode, the period deviation dT approaches zero quickly and then jumps up and down around zero before gear-shifting, which means CKV is quite noisy at that time. After we trigger the gear-shifting event and change ADPLL to a type-II PLL, the variation of dT is smaller and dT slowly settles to zero. From this figure we know at about 49 us ADPLL just settles to the desired frequency. The time for frequency settling is around 13 us.

We have also simulated the settling transient for low frequency (f_v is 3300 MHz), high frequency (f_v is 4050 MHz) and the near integer-N case (f_v is 3556.214 MHz) as from Fig E-25 to Fig E-30. The settling behavior is similar and the settling time in all cases is within 15 us.

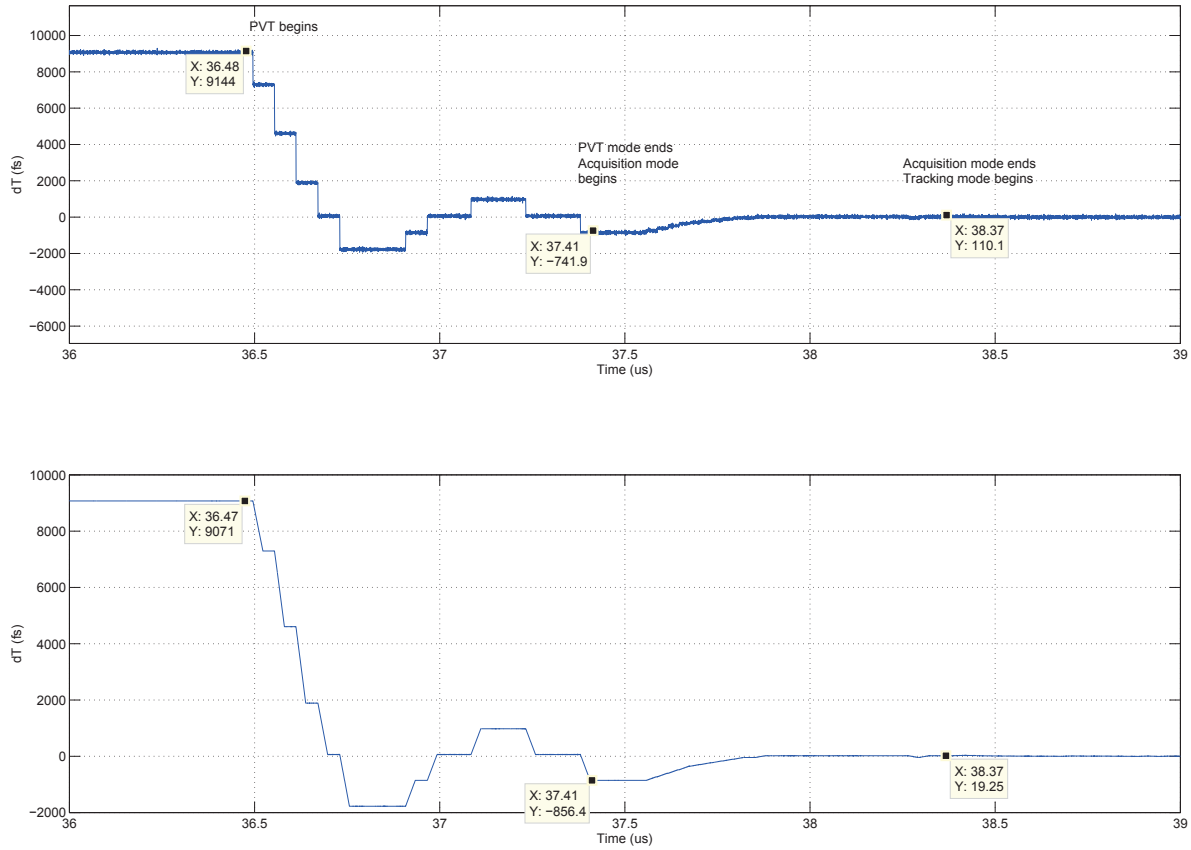

Figure 5-8: ADPLL settling transient in PVT mode and acquisiton mode($f_v = 3800$ MHz)

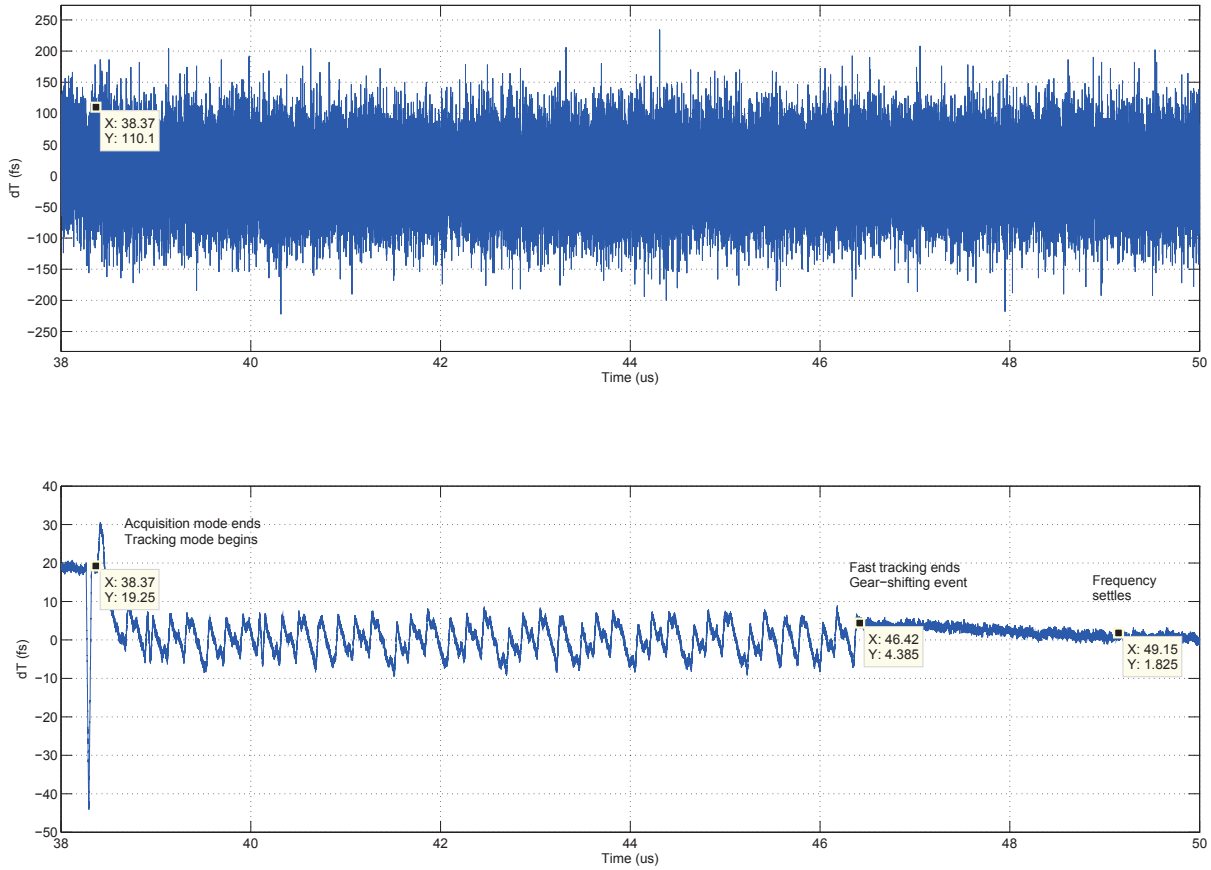

Figure 5-9: ADPLL settling transient in tracking mode($f_v = 3800$ MHz)

Chapter 6

DPA Design for ADPLL

Design of DPA in our ADPLL project serves two purposes: 1. Function as a low noise output buffer for ADPLL; 2. Explore the possibility of the WiMAX transmitter architecture with ADPLL and DPA.

Thus unlike the core blocks (TDC, DCO) of ADPLL, the specification of DPA is not very tight. One important consideration is that it should not degrade the far-out noise specification of DPA (-150dBc/Hz). This, as we will see in the simulation result, is not a tough requirement. To simplify the design effort, we have chosen the single-ended DPA with a topology similar to Fig 6-4.

The idea of this DPA topology is rooted in the traditional Class-E PA. We will discuss from that and converge to our design.

6-1 Introdution to the Class-E PA

This DPA design in the ADPLL project is based on the basic Class E PA concept, which is proposed by Sokal in 1975[32]. The architecture of Class E PA is shown in Fig 6-1.

The transistor in Class E PA functions mainly as a switch. Therefore we prefer the input signal to be a rectangular wave so as to turn the switch fully on and off. The RF choke works more like a constant DC current source. So basically the current would charge the capacitor C_x when the switch is off and the voltage at the drain of transistor would go up. When the switch is on, the capacitor is discharged and the voltage would decrease to zero. Therefore, the DC current from RF choke is converted to the RF current.

L_o and C_o form a tank resonating at the fundamental frequency of the input signal to make sure the harmonic component in the current flowing to the load is almost filtered out. L_x,

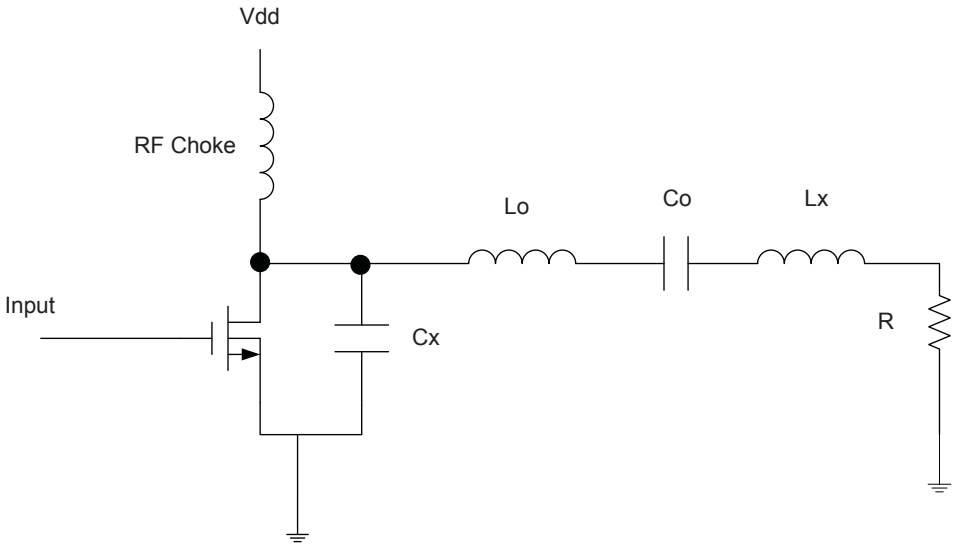

Figure 6-1: Topology of the Class E PA.

the residue inductance, together with C_x will define the waveform of drain voltage and current. There are three objectives for the waveform of drain voltage and current[33]:

1. The rise of the voltage across the transistor at turn-off should be delayed until after the transistor is off.
2. The drain voltage should be brought back to zero at the time of transistor turn-on.
3. The slope of drain voltage should be zero at the time of turn on.

When these conditions are met, at anytime, either the drain current or the drain voltage of the switch transistor will be zero to make sure there is no power dissipated on the transistor. (A typical waveform of drain voltage and current of the Class E PA is shown in Fig 6-2.) If all passive components are lossless, the Class E PA can achieve the ideal drain efficiency of 100%. So we call that 'optimum class E'. An amplifier which cannot meet these three criteria is called 'sub optimum class E'.

6-2 Introduction to the DPA Topology

The DPA concept proposed in [34, 35] is a near Class-E Amplifier. The DPA schematics of these two papers are shown below as Fig 6-3 and Fig 6-4. Via the digital control bits (or called *amplitude control word*, ACW), we can control AND gates to pass or stop the input oscillating signals, thus to enable/disable the switch transistor array. Therefore the output power will be digital controllable.

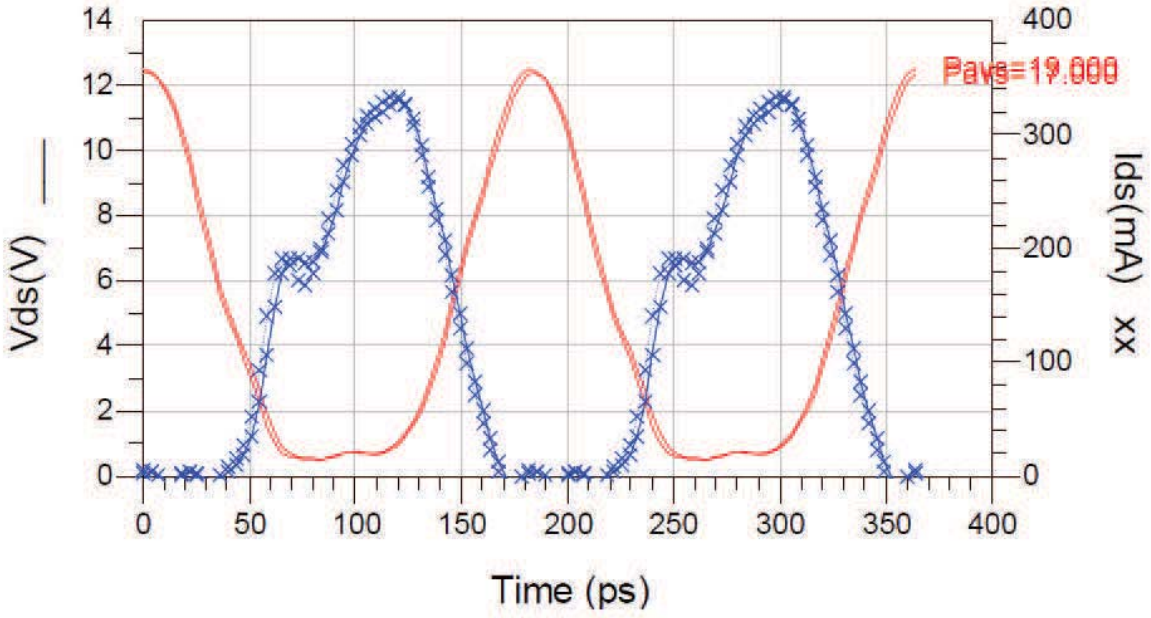

Figure 6-2: Typical drain voltage and current waveform of the Class E PA[2].

Actually, this DPA architecture deviates from the Class E PA in that the switch transistor array does show finite resistance rather than zero when turned on. If ACW is large, i.e. most of the AND gates will pass the input oscillating signal, the drain voltage and current waveform ressembles those of a typical Class E PA. When ACW is small, the resistance of switch is so large that the charge build up in the parallel capacitor cannot be discharged to near zero when the switch is on. Therefore, the amplitude of voltage swing at the drain of transistor is smaller and the output power is smaller. The switch will also dissipate some portion of power since the voltage is not zero when it's on. Then the drain efficiency of DPA will be lower.

6-3 DPA Schematic Overview

The top-level schematic for DPA is shown in Fig 6-5. In the schematic there is a module named catip_adpll_dpaswitcharray which functions as a resistance-controllable switch array, a module named catip_adpll_dpacaparray which is to tune the parallel capacitor value in the classical DPA toplogy, and a buffer in front of catip_adpll_dpaswitcharray.

The schematic of DPA switch array is shown in Fig 6-6. The Signal_input signal is just the output of the previous buffer as in Fig 6-5. It is fed to an array of DPA switch unit modules, which are binary controlled by the ACW signal. The schematic of switch unit module is given in Fig 6-7. By using the NAND gate and the inverter stage, an AND

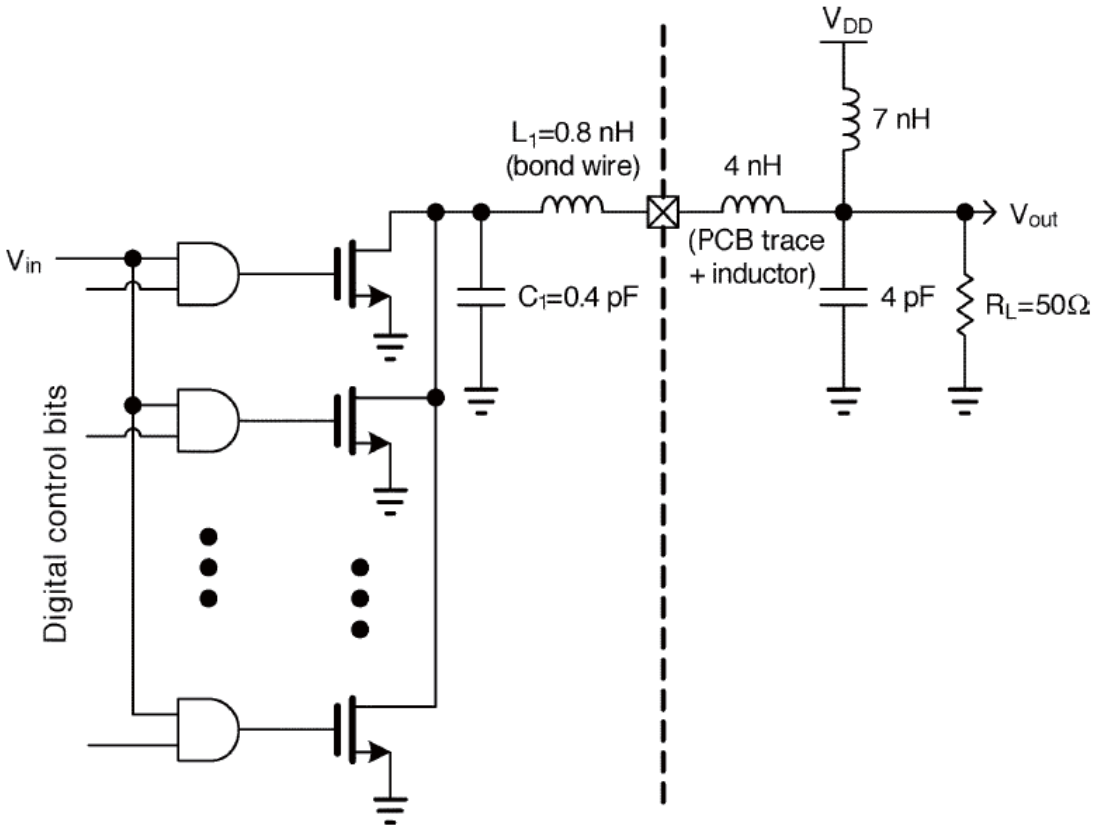

Figure 6-3: Schematic of DPA for BT transmitter.

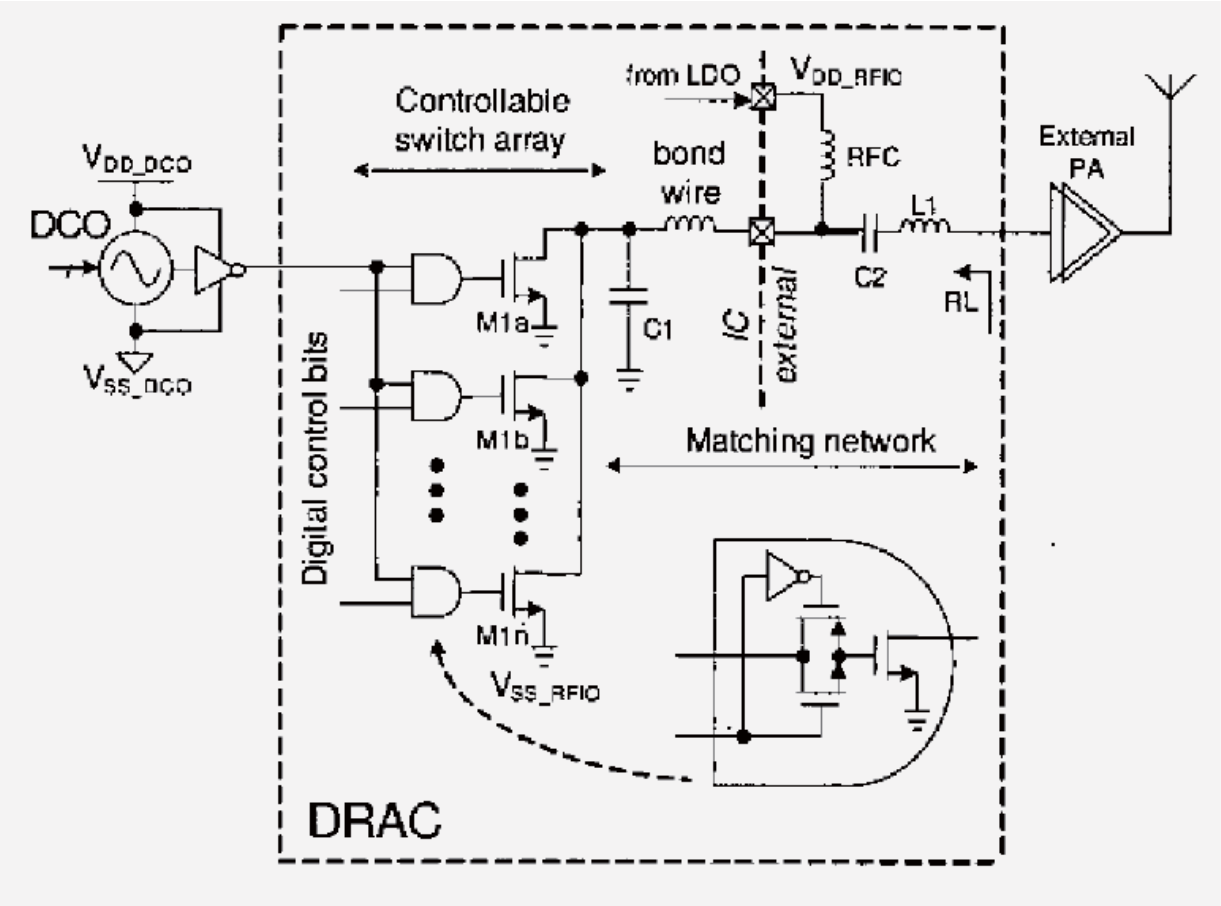

Figure 6-4: Schematic of DPA for GSM/GPRS/EDGE transmitter.

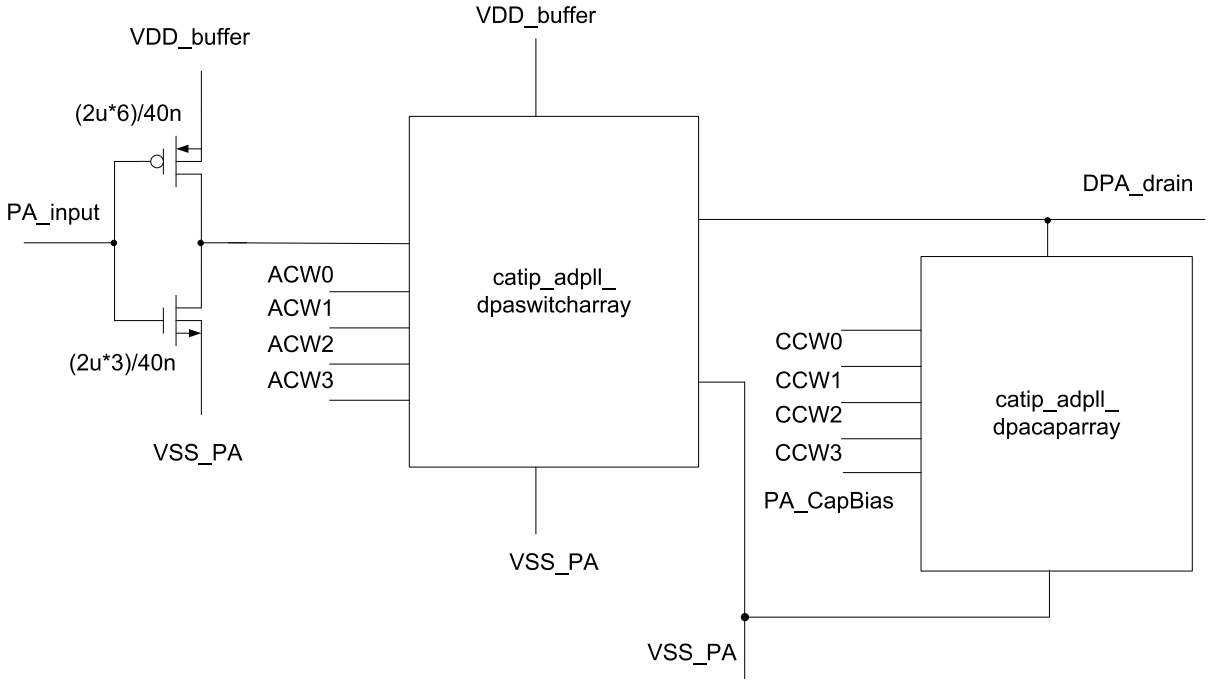

Figure 6-5: Top level schematic of DPA

function of ACW and Signal_input is realized. These two stages also function as a buffer for the DPA switch stage. The scaling guarantees that the rising time and falling time of signal is within 20-30ps, which is about 10% of the cycle of the input periodical signal. Thus the immunity of DPA to the supply noise is improved.

The NAND gate is just a basic CMOS logic NAND gate. Compared to a transmission-gate AND (or NAND) gate, the CMOS logic is regenerative and has driving strength. Thus the buffer chain length for the switch stage is reduced. Also the requirement of the far-out noise for WiMAX is not as tough as GSM. The noise of a CMOS NAND gate is acceptable.

The module of DPA cap array is a switchable-capacitor array that is binary controlled by the capacitance control word (CCW). The schematic of one switchable-capacitor unit is shown in Fig 6-8. The Capcontrol signal is a certain bit of the CCW word (CCW0–CCW3) and it controls a NMOS switch between the 50 fF MOM capacitor and the ground. The capacitor unit is connected to the output signal at the drain of the switch transistors. Thus by changing the CCW value, we can tune the capacitance value parallel to the switch transistors, which is C_x in Fig 6-1. By this we can adapt the DPA block to different working frequency and different package parasitic values. The resistor Rbias provides a bias voltage for the drain of the switch. Thus this node will not be floating and the switch is fully turned off when the Capcontrol signal is low.

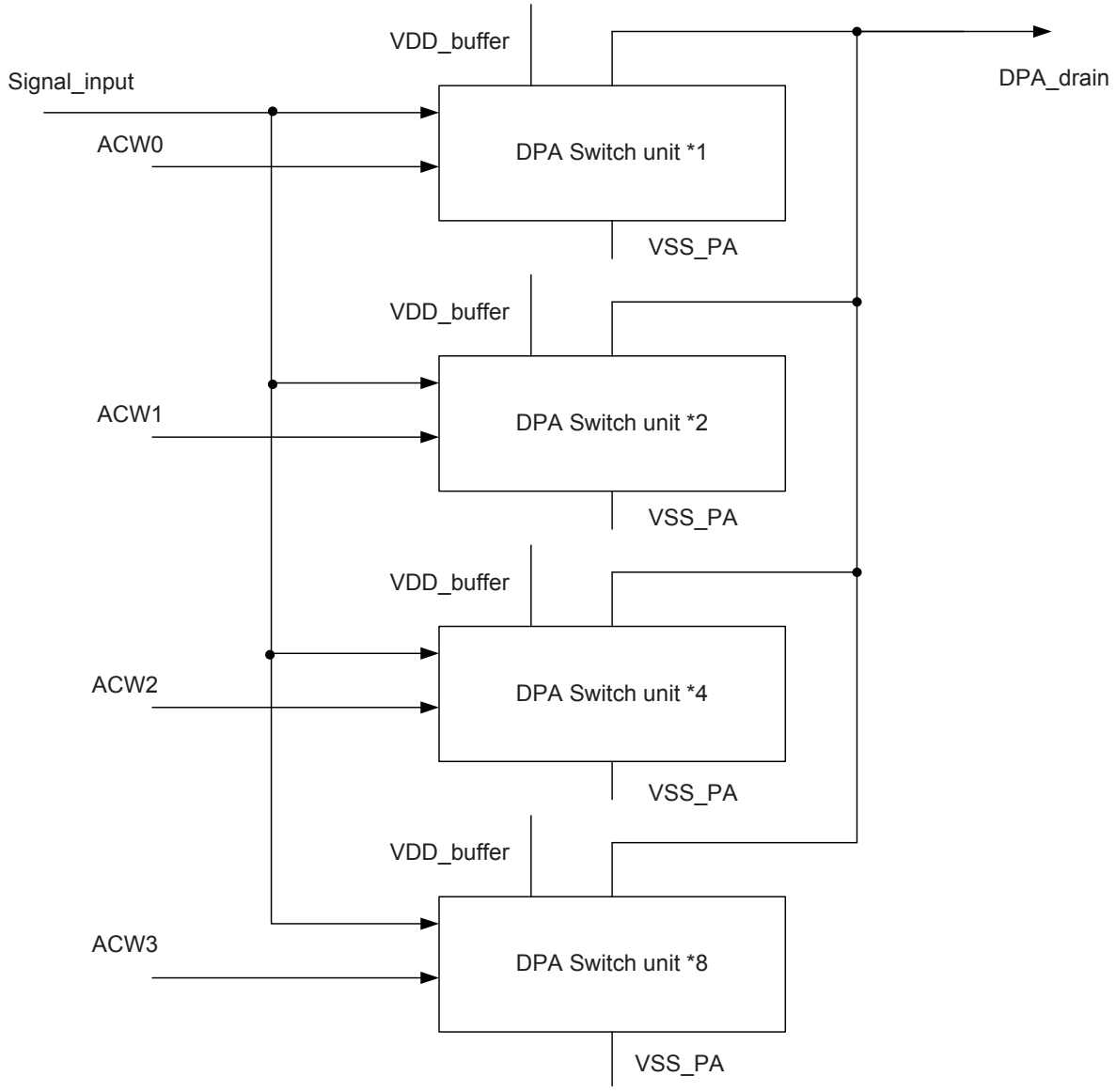

Figure 6-6: Schematic of DPA switch array.

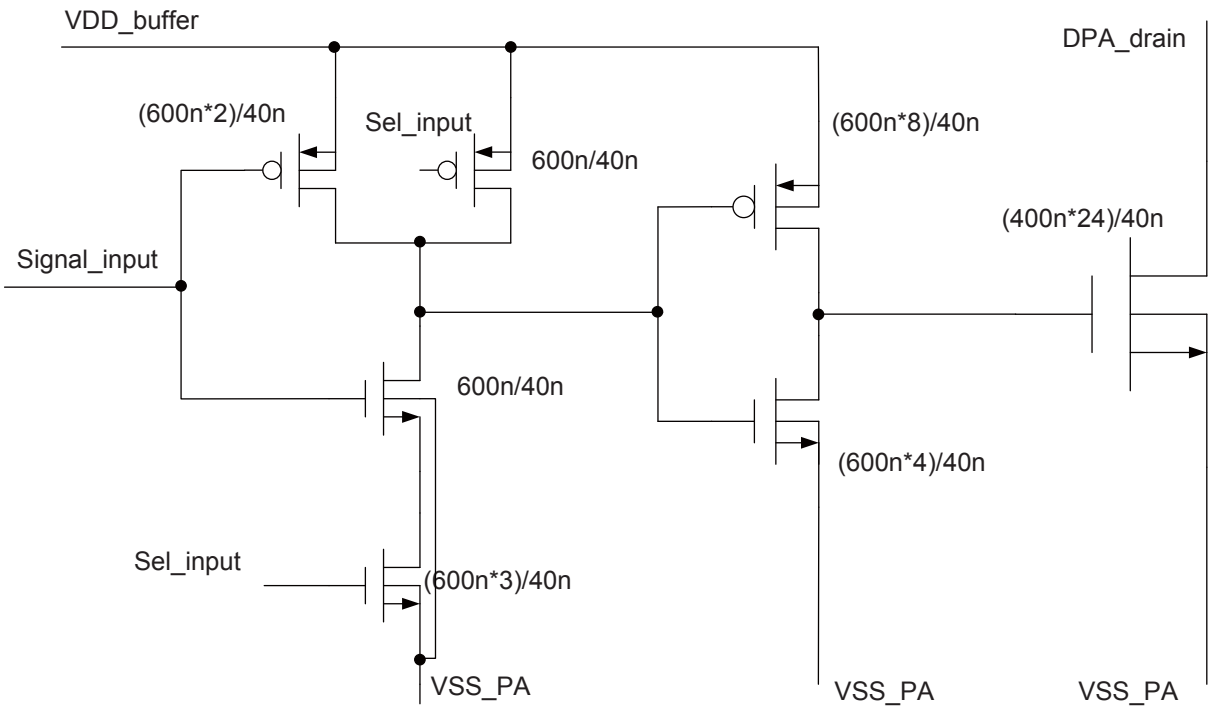

Figure 6-7: Schematic of DPA switch unit.

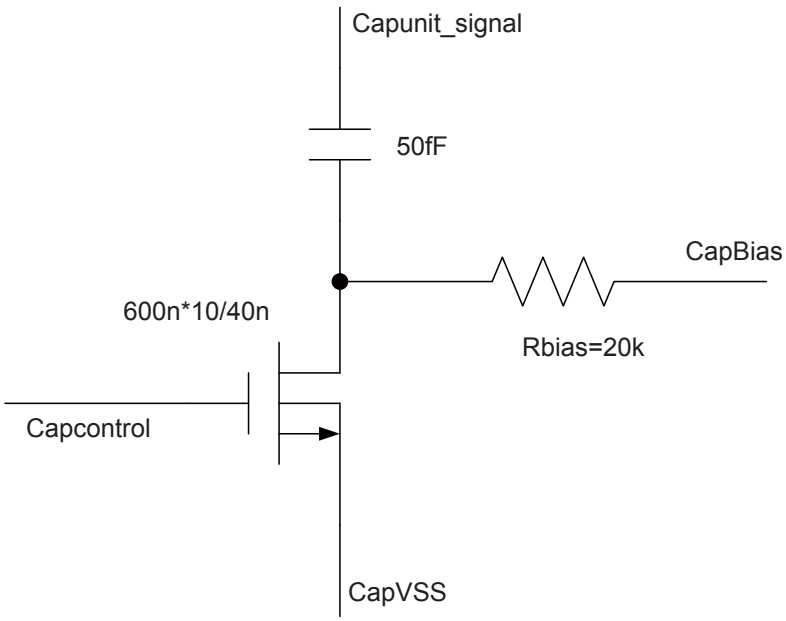

Figure 6-8: Schematic of a switchable capacitor unit in DPA

6-4 Practical Concerns for the DPA Design

DPA functions as the output buffer for ADPLL. So it will interface with package parasitic like bondwire / bondpad / lead frame. It also uses off-chip components as the RF choke and the matching network, which is of course, never ideal. Besides that, DPA does deliver a significant amount of power. Thus the reliability needs to be taken into consideration.

Schematics of the test benches for DPA HB (3.3-3.8GHz) and LB (2.3-2.7GHz) mimicking the practical situation are shown in Fig 6-9 and Fig 6-10.

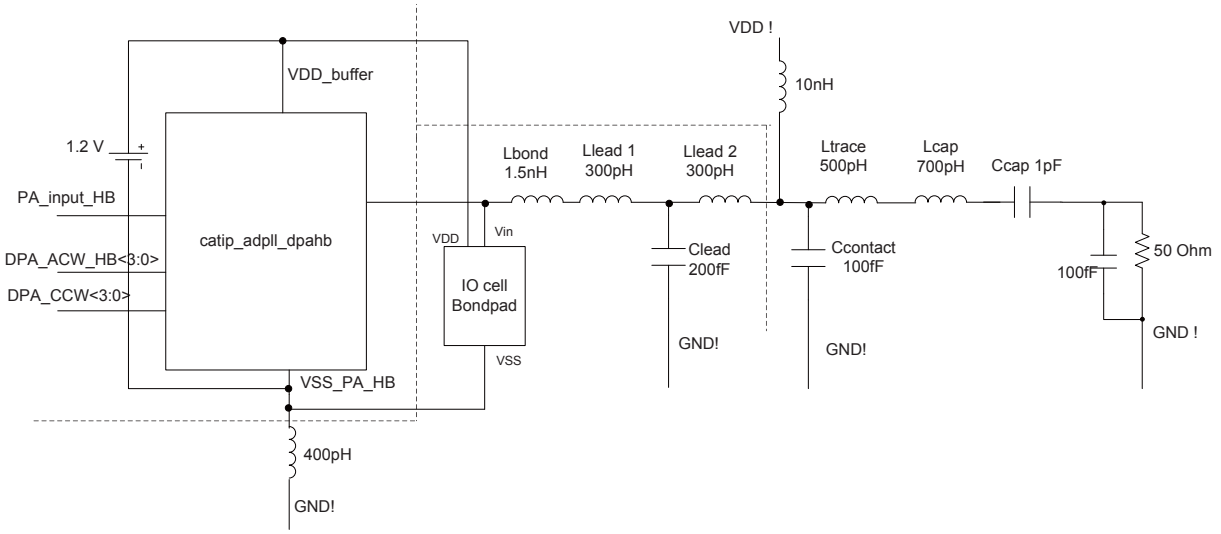

Figure 6-9: Schematic of the test bench for DPA HB part

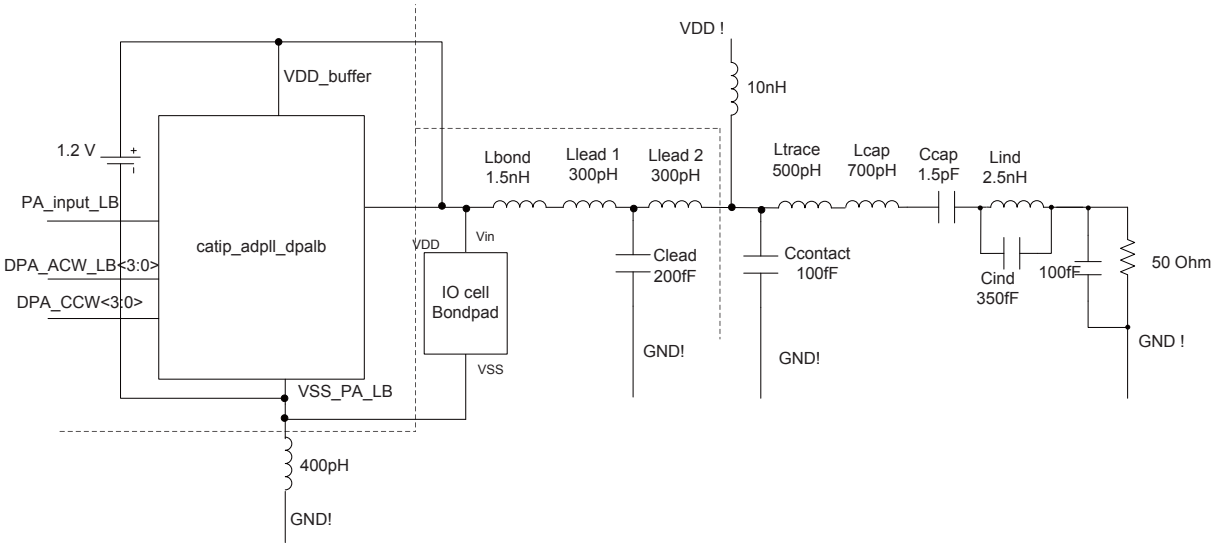

Figure 6-10: Schematic of the test bench for DPA LB part

In the test benches, the parasitic introduced by the IO cell and the bondpad are extracted out. We assume the bondwire inductance value to by a typical value of 1.5 nH. The parasitic of the lead frame is modeled by $Llead1$, $Llead2$ and $Clead$. The inductance value for the RF choke is 10 nH. The inductance due to PCB trace is 500 pH. The parasitic capacitance for the lead frame contact is 100 fF. For DPA HB, we just use a off-chip capacitor component for matching network. The capacitance value $Ccap$ and the parasitic inductance value $Lcap$ are 1 pF and 700 pH respectively. For DPA LB, an off-chip capacitor and an off-chip inductor are used. The capacitance value $Ccap$ and the parasitic inductance value $Lcap$ are 1.5 pF and 700 pH respectively while the inductance value $Lind$ and the parasitic capacitance value $Lcap$ are 2.5 nH and 350 fF respectively.

There some are issues to consider:

1) Not ideal ground. The ground on a chip can never be ideal. VSS of every block will be connected to the off-chip ground via some bondwire. When there is a current spike going through the bondwire, which is modeled as an inductor, the potential of the on-chip ground will shift. This is especially worse for singled-ended, high frequency, high current component, i.e. our DPA.

So here the ground inductance is minimized by choosing two parallel down-bonding bondwire to the die paddle. The estimation is about 400 pH. The ground bounce peak voltage for HB and LB are about 200mV.

2) Only low-Q tank. High-Q tank assumption of the Class E PA is important to make sure that the output waveform is purely sinusoidal at fundamental frequency. However, due to the bondwire-bondpad-leadframe network, the Q value of our DPA cannot be very high. Besides that, the band of WiMAX is wider compared to BT and GSM. With a very high Q tank, the output power would vary too much versus frequency. So in this design, the on-off chip interface and off-chip components form a low-Q matching network. The output waveform also deviates a lot from the ideal sine wave.

As we see in Fig 6-9 and Fig 6-10, the parasitic of off-chip components are also modeled by checking the SRF of inductor and capacitor. For the HB case, the optimized inductance value for an off-chip inductor is rather small. Thus the matching network doesn't have an off-chip inductor and the performance would not degrade too much.

3) Transistor reliability. For traditional Class E PA, the drain voltage can go to over 3 times VDD. The nominal working voltage for core transistor is 1.1V. From [35], voltage swing at the drain must be controlled by the matching network to satisfy that the equivalent dc voltage on the drain resulting from one RF cycle is smaller than 2 times VDD. In this design, two measures are adopted for reliving the stress on switch transistors:

a) For DPA switch stage, use a separate supply voltage of 0.6V. The drain voltage is, to first order estimation, proportional to supply voltage connected to RF choke. So low supply voltage will result in low peak value of drain voltage. b) Intentionally increase the capacitance value of the capacitor parallel with switch transistors. Drain voltage goes up in one RF cycle when switch transistors are off and current charges parallel capacitor. So

using larger capacitor value will damp the drain voltage waveform, which will reduce the DPA efficiency.

The simulated peak voltage between drain and source for switch transistor is smaller than 1.4V. That's far enough to guarantee the transistor works properly.

4) ESD issue. As the drain of switch transistors will be connected to the IO cell and the bondpad, ESD is also an important concern. In our design, the buffer stage (including NAND gates and inverters) will have a 1.2V supply. That on-chip supply will be connected to VDD of the IO cell for PA output. Since we know that peak voltage between local ground and PA output is smaller than 1.4V. The diode in IO cell will not be turned on and will not influence the performance of DPA.

5) Current handling capability. For on-chip power amplifier, since significant amount of current will flow through the switches, we do need to investigate current capability of transistors. The design kit of the process specifies DC current maximum value (to prevent electro-migration) and RMS current maximum value (to prevent the excessive heat kill the chip). Basically it's the metal of drain and source that may have problem. In the layout the metal lines are made wide enough to satisfy the requirement of current handling capability.

6-5 DPA layout overview

The layout of the critical modules of DPA, catip_adpll_dpaswitcharray and catip_adpll_dapcaparray, are shown in Fig F-1 and Fig F-2. Fig F-1 shows a 7*5 array of switch units. The insider 5*3 array are the effective switch units and the most outside circle of switch units are dummy cells for better matching performance. The switch units for the adjacent columns are flipped so that they can share the same VDD (or VSS). The metal lines for the ground and supply signal are made wide to handle the current. In Fig F-2 the similar topology is applied for the switchable capacitor array. Nevertheless, there is no dummy cell since the matching is not a critical concern. The connection to the MOM capacitor in the capacitor unit is manually made wider to handle the RMS current in the capacitor.

As the final floorplan and the package are not fixed, also due to the limited time, the layout of the whole DPA (for HB and LB) is not done. The output signals, the ground and the supply shall go to the pads via the thickest metal to minimize the parasitic resistance. The ground signal and the supply signal shall go close to each other to minimize the parasitic inductance and gain some decoupling capacitance for free. Small capacitors can be inserted into catip_adpll_dpaswitcharray to localize the high frequency current. Some large capacitor can be put close to the whole array to localize the low frequency current. We may need to put additional buffer(s) before the inverter in 6-5 if the path from the DCO block output to the DPA input is too long.

6-6 DPA Simulation Results

Some simulations with extracted parasitics from the layout of modules have been done to evaluate the DPA performance. The power control capability of DPA for HB and LB are presented in Fig 6-11 and Fig 6-12. The output power doesn't change much as long as ACW is large, i.e. the switch resistance is negligible. It changes quite a lot when ACW is small, as we expected.

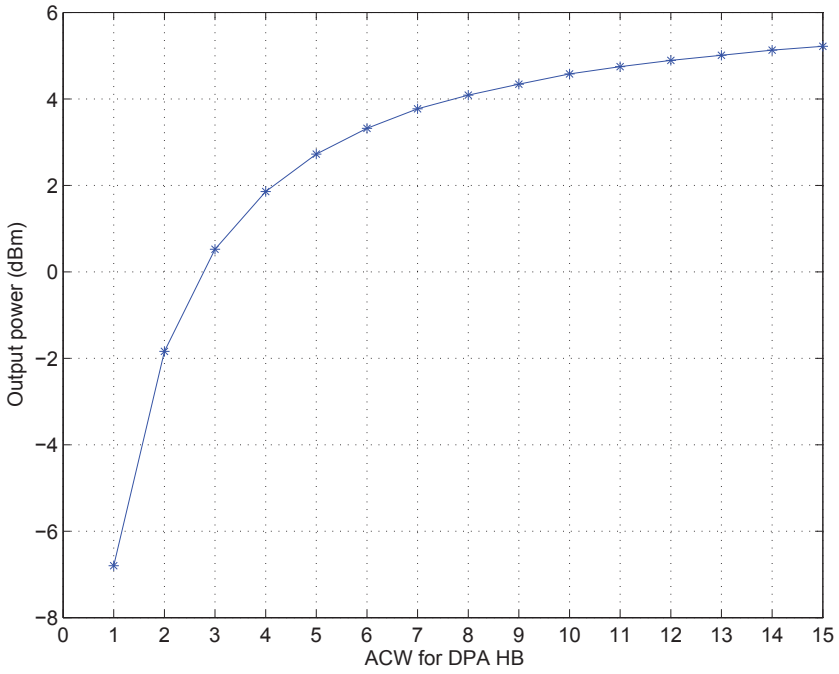

Figure 6-11: Power control capability of DPA HB.

The drain efficiencies (DE) and the power added efficiencies (PAE) of DPA for HB and LB are plotted in Fig 6-13 and Fig 6-14. For large ACW values, DE is around 60% and PAE is around 40%. The PAE value first increases and then becomes a little lower as we sweep the ACW value. That is because the output power doesn't increase too much yet the power consumption for the buffer stages increase faster when more switch units are turned on.

The phase noise performance for HB and LB are plotted in Fig 6-15 and Fig 6-16.

The DPA output power with respect to the frequency is shown in Table 6-1 and Table 6-2.

For the corner simulation, the typical, slow and fast corners at 27°C are simulated as well as the typical corner with different temperature (27°C, 85°C and -40°C). The corner simulation result for DPA HB and LB are listed in Table 6-3 and Table tab:DPALBOutputPowerCorner.

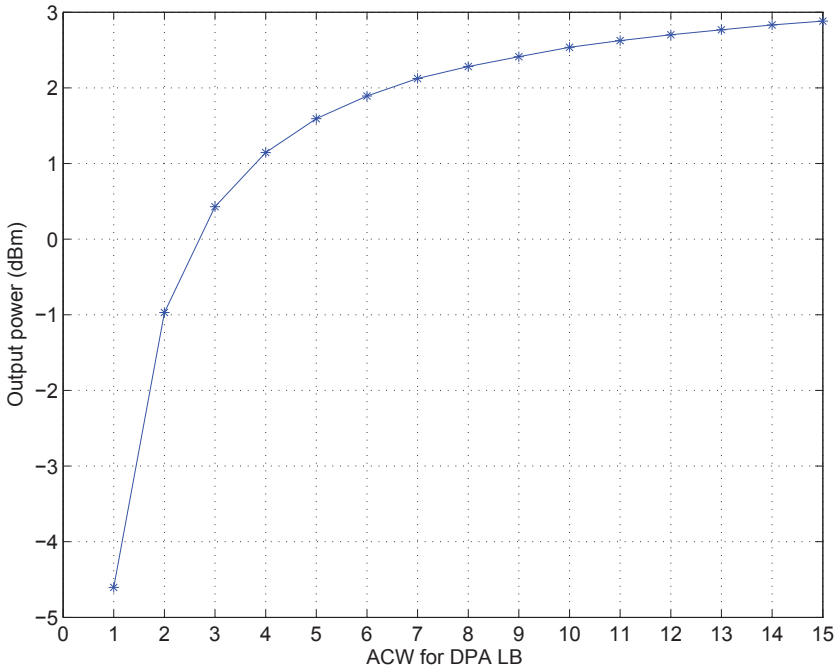

Figure 6-12: Power control capability of DPA LB.

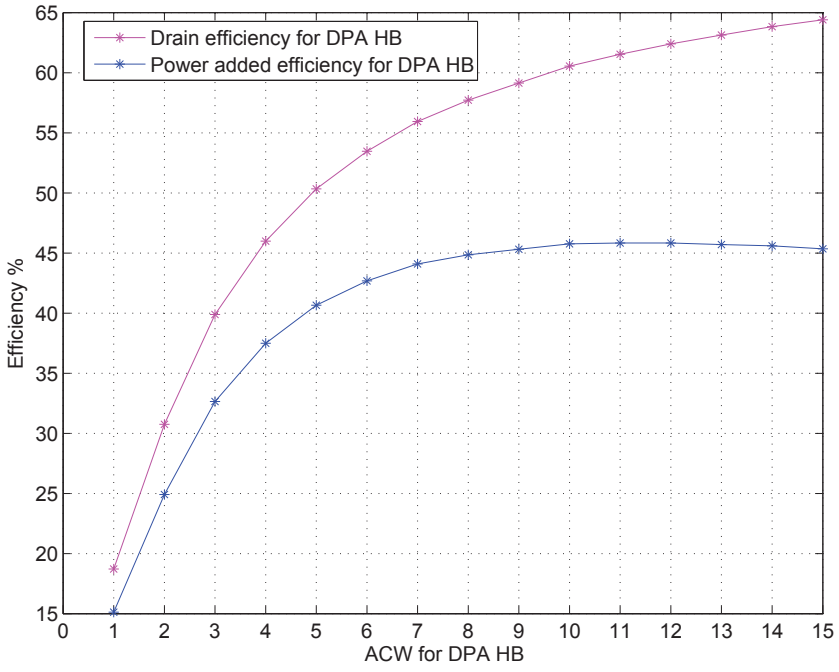

Figure 6-13: DPA efficiency curve with respect to ACW (HB).

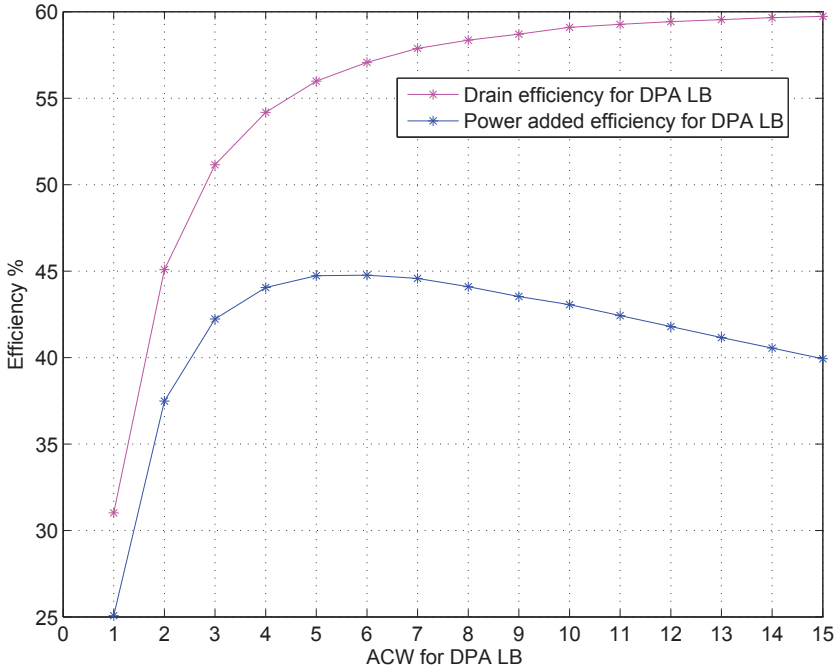

Figure 6-14: DPA efficiency curve with respect to ACW (LB).

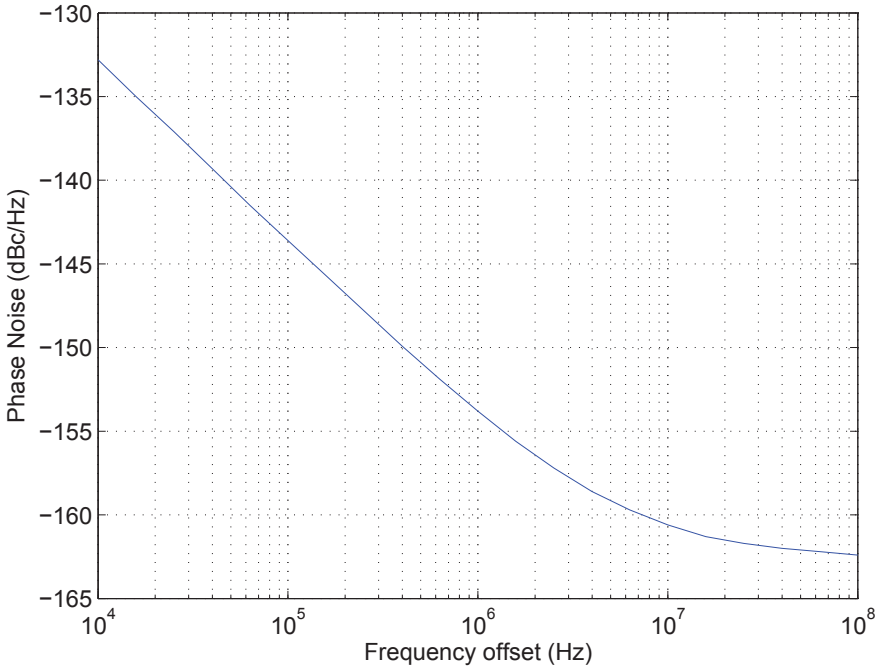

Figure 6-15: Phase noise performance of DPA HB.

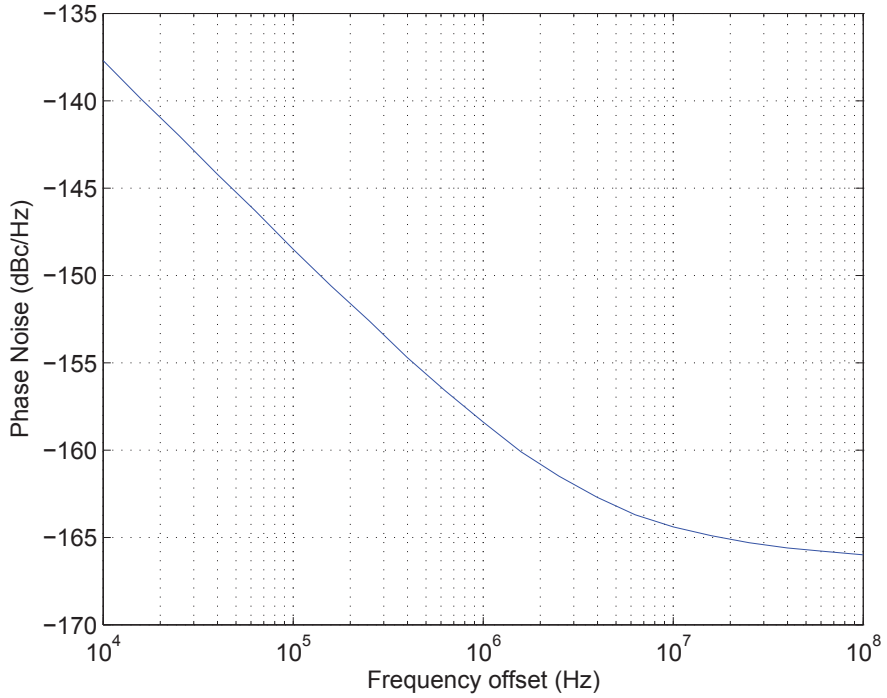

Figure 6-16: Phase noise performance of DPA LB.

Table 6-1: DPA HB output power with respect to frequency (ACW=15).

Frequency (GHz)	3.3	3.4	3.5	3.6	3.7	3.8
Output power (dBm)	5.527	5.427	5.293	5.14	4.94	4.786

Table 6-2: DPA LB output power with respect to frequency (ACW=15).

Frequency (GHz)	2.3	2.4	2.5	2.6	2.7
Output power (dBm)	3.725	3.343	2.882	2.377	1.853

Table 6-3: DPA HB output power for corner simulation.

Corner	TT 27°C	FF 27°C	SS 27°C	TT 85°C	TT -40°C
Output power (dBm)	5.219	5.631	4.666	5.8	5.237

Table 6-4: DPA LB output power for corner simulation.

Corner	TT 27°C	FF 27°C	SS 27°C	TT 85°C	TT -40°C
Output power (dBm)	2.882	3.13	2.59	2.872	2.902

Chapter 7

Conclusion

7-1 Contribution of This Thesis

This work consists of the system level design of ADPLL for the WiMAX standard and a simple transistor level design of DPA. The main contributions of this work are:

1. The architecture and building blocks of the ADPLL system are presented, with details on the function of every module. Modeled and described in Verilog-AMS/Verilog, the whole system is simulated in time domain and the performance is presented. The phase noise performance fits into the specification except for only 0.55 dB violation at the frequency offset of 100 kHz. The settling time of the frequency synthesizer is around 15 us.

2. The spur mechanism for the near-integer N cases of ADPLL is analyzed. The spur suppression techniques, the phase rotation algorithm and the FREF dithering algorithm, are applied to the system and the spurs are effectively suppressed.

3. The top level issues of the ADPLL system are taken care of. The test plan for the whole system and the critical modules, DCO and TDC, are proposed. The operation modes of ADPLL are specified, with the requirement for the SPI block, the sequencer block and the memory modules.

4. A simple DPA as the output buffer of ADPLL is designed in the 40 nm process. The layout of catip_adpll_dpaswitcharray and catip_adpll_dpacaparray, the essential parts of the DPA block, are presented. The primitive simulation results of the DPA circuitry are given.

7-2 Future Work

The expected future work can be categorized into three aspects:

1. The system design aspect. It is worth to be noticed that in the scope of this thesis, the TDC normalization and the DCO normalization are both done in an off-chip way. As we don't have a powerful digital signal processor, the division in the TDC normalization as in Fig 2-23 is done off-chip. The DCO normalization is done via the DCO open loop test as in Section 5-3. A commercial chip shall realize the normalizations on-chip. For the TDC normalization algorithm, a correlation method is proposed in [11] and shall be a good research topic. For the DCO normalization algorithm, the LMS calibration algorithm has been proposed [36], which may be possible to be applied in our ADPLL system also.

   What's more, the mismatch of DCO is not modeled in the work of this thesis. A loop-up table of the DCO capacitor value can be built from the data of the DCO transistor-level simulation result and used in the system simulation. A polar transmitter system is expected to be built based on the system. When the modulation for the transmitter operation is added to ADPLL, the DCO mismatch may lead to the performance degradation.

2. The DPA design aspect. Due to the limited time, the DPA implemented here is very basic. It needs further verification and optimization. The monte-carlo simulation should be done to investigate the influence of the mismatch on the DPA performance. When the package is determined, the value of the off-chip components (inductors and capacitors) shall be chosen to maximize the output power within the corresponding band (HB or LB). The influence of the variation of the parasitics and the component values on the DPA performance should also be explored.

   To be eligible for integration into a polar transmitter of non-constant amplitude modulation, we may need to add extra bits to ACW to increase the tuning granularity. Impedance transformation may be needed in the matching network so as to change the terminal load to be less than 50 Ohm to increase the output power. Some other topologies like inverse Class D power amplifier can also be a option for the future improved design[37].

3. The chip creation aspect. A primitive pinout intended for this ADPLL chip is presented in Fig 7-1. Besides the essential modules that are under design also finished (TDC, DCO, DPA), there are some red blocks here, which means they are not taken care of. Among them, the design of FREF slicer (with the FREF dithering) and the retimer+incrementor will take a significant effort. SRAM is red due to the temporary lack of memory compiler for 40 nm process.

   After the sizes and the interfaces of these blocks can be delivered, we will go to floorplan and determine the package we need. Then the chip will be assembled and ready for the top level chip creation steps.

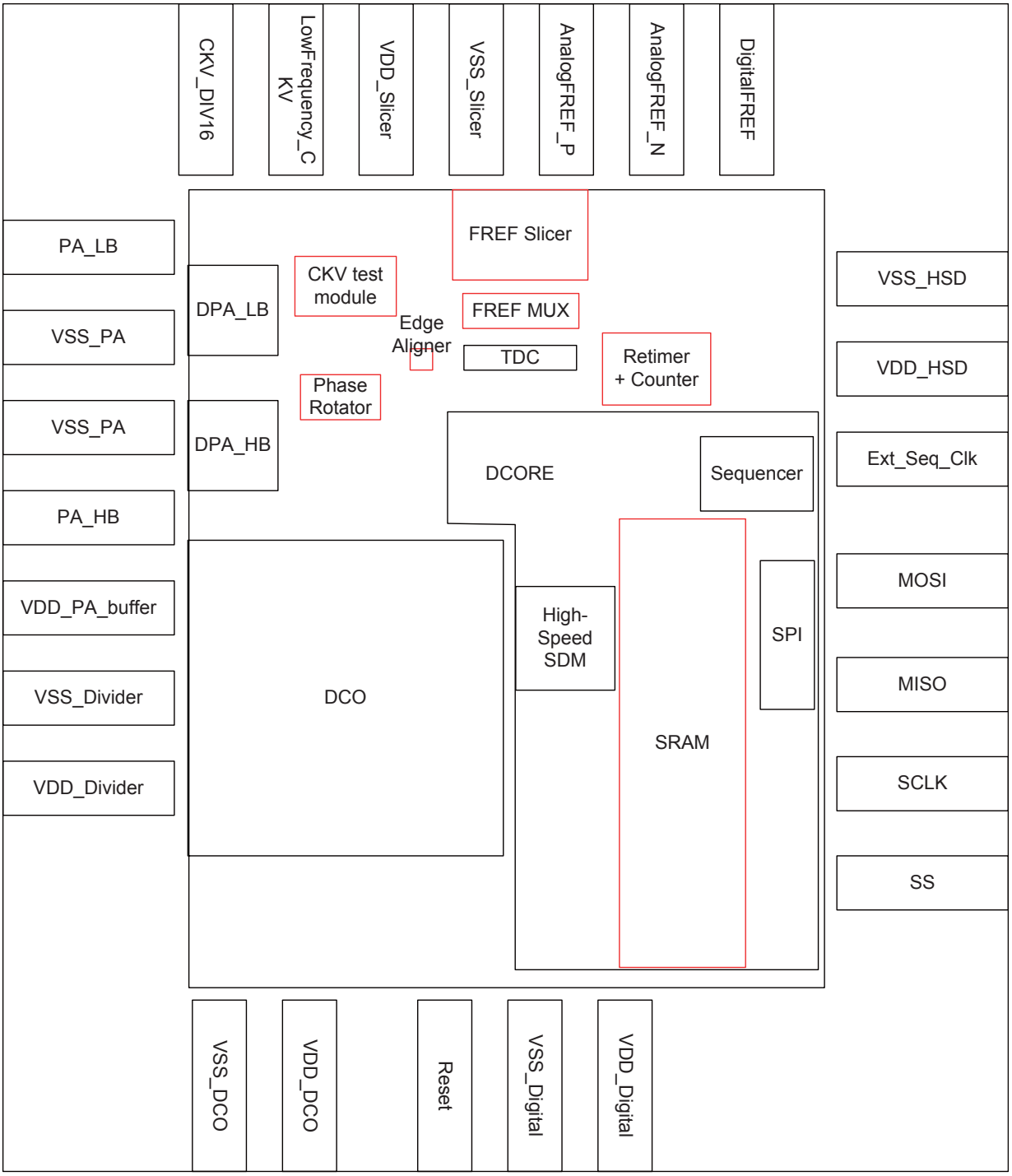

Figure 7-1: Primitive Pinout for this ADPLL chip

To summarize, there is enough space for innovation, improvement and optimization based on current progress of the ADPLL project.

Appendix B

Top-level Interface of ADPLL

Table B-1: ACORE Interface

Signal	Word length	Input/Output/Supply	Comments
VDD_Slicer	1	Supply	VDD for FREF slicer
VSS_Slicer	1	Supply	VSS for FREF Slicer
VDD_HSD	1	Supply	VDD for TDC and Incrementor
VSS_HSD	1	Supply	VSS for TDC and Incrementor
VDD_Divider	1	Supply	VDD for buffers and dividers in DCO block
VSS_Divider	1	Supply	VSS for buffers and dividers in DCO block
VDD_DCO	1	Supply	VDD for DCO core
VSS_DCO	1	Supply	VSS for DCO core
VDD_buffer	1	Supply	VDD for buffer stage in DPA block
VSS_PA	1	Supply	VSS for PA stage and buffer stage in DPA block
PA_CapBias	1	Supply	Voltage bias for DPA capacitor bank
PA_HB	1	Output to pad	DPA HB output signal
PA_LB	1	Output to pad	DPA LB output signal
CKVD16	1	Output to pad	CKV divided by 16, for test
CKR	1	Output to DCORE	Retimed FREF
CKVD8	1	Output to DCORE	CKV divided by 8. Generated in incrementor. Used in $\Sigma\Delta$ modulator for fractional bits in tracking bank.
TDCQ	40	Output to DCORE	TDC raw output
PHV_SMP	10	Output to DCORE	Incrementor output
FREF	1	Output to DCORE	Reference clock

Signal	Word length	Input/Output/Supply	Comments
Lowfrequency_CKV	1	Input from pad	Off-chip signal may used as CKV for TDC test
DigitalFREF	1	Input from pad	Reference clock signal generated from device
AnalogFREF_P	1	Input from pad	Differential reference clock signal generated from crystal oscillator
AnalogFREF_N	1	Input from pad	Differential reference clock signal generated from crystal oscillator
SEL_EDGE	1	Input from DCORE	SEL_EDGE signal for retimer and incrementor
FREFdither	1	Input from DCORE	Signal to enable/disable FREF dithering algorithm
FREF_SEL	1	Input from DCORE	Select FREF signal between DigitalFREF and AnalogFREF
CKV_SEL	1	Input from DCORE	Choose CKV from phase rotator output or external signal
DPA_ACW_HB	4	Input from DCORE	Amplitude control word for DPA HB
DPA_ACW_LB	4	Input from DCORE	Amplitude control word for DPA LB
DPA_CCW	4	Input from DCORE	Control word of capacitor bank for DPA
PR_en	1	Input from DCORE	Enable signal for Phase Rotator
PR_rst_a	1	Input from DCORE	Asynchronous reset for Phase Rotator
DCO_IN_P_DS	7	Input from DCORE	Tuning words for drain and source terminal of switches in PVT bank
DCO_IN_P_G	7	Input from DCORE	Tuning words for gate terminal of switches in PVT bank
DCO_IN_A	6	Input from DCORE	Tuning words for Acquisition bank
DCO_IN_T	64	Input from DCORE	Tuning words for integer part of tracking bank
DCO_IN_TF	1	Input from DCORE	Tuning word for fractional part of tracking bank
DCO_tail_resistor	5	Input from DCORE	Control word for tail resistor in DCO core
DIV_1_5_EN	1	Input from DCORE	Enable/Disable divide-by-3 divider after DCO core
HB_PA_DIV2_EN	1	Input from DCORE	Enable/Disable divide-by-2 divider which generates input signal of DPA HB
LB_PA_DIV2_EN	1	Input from DCORE	Enable/Disable divide-by-2 divider which generates input signal of DPA LB
HB_BUF_EN	1	Input from DCORE	Enable/Disable buffer of HB signal

Signal	Word length	Input/Output/Supply	Comments
LB_BUF_EN	1	Input from DCORE	Enable/Disable buffer of LB signal
Test_Divider_rst_a	1	Input from DCORE	Asynchronous reset signal for divide-by-16 divider used for test
Clk_Gating_Enable	1	Input from DCORE	Enable/Disable clock gating signal in TDC block (expected feature)

Table B-2: DCORE Interface

Signal	Word length	Input/Output/Supply	Comments
VDD_Digital	1	Supply	VDD for the whole DCORE
VSS_Digital	1	Supply	VSS for the whole DCORE
CKVD	1	Input from ACORE	CKV divided by 8. Used in $\Sigma\Delta$ modulator for fractional bits in tracking bank.
CKR	1	Input from ACORE	Retimed reference clock
TDC_Q	40	Input from ACORE	TDC raw output
PHV_SMP	10	Input from ACORE	Incrementor output
SEL_EDGE	1	Output to ACORE	SEL_EDGE signal for retimer and incrementor
FREFdither	1	Output to ACORE	Signal to enable/disable FREF dithering algorithm
FREF_SEL	1	Output to ACORE	Select FREF signal between Digital-FREF and AnalogFREF
FREF	1	Input from ACORE	Reference clock used for TDC test
CKV_SEL	1	Output to ACORE	Choose CKV from phase rotator output or external signal
DPA_ACW_HB	4	Output to ACORE	Amplitude control word for DPA HB
DPA_ACW_LB	4	Output to ACORE	Amplitude control word for DPA LB
DPA_CCW	4	Output to ACORE	Control word of capacitor bank for DPA
PR_en	1	Output to ACORE	Enable signal for Phase Rotator
PR_rst_a	1	Output to ACORE	Asynchronous reset for Phase Rotator
DCO_IN_P_DS	7	Output to ACORE	Tuning words for drain and source terminal of switches in PVT bank
DCO_IN_P_G	7	Output to ACORE	Tuning words for gate terminal of switches in PVT bank
DCO_IN_A	6	Output to ACORE	Tuning words for Acquisition bank
DCO_IN_T	64	Output to ACORE	Tuning words for integer part of tracking bank
DCO_IN_TF	1	Output to ACORE	Tuning word for fractional part of tracking bank

Signal	Word length	Input/Output/Supply	Comments
DCO_tail_resistor	5	Output to ACORE	Control word for tail resistor in DCO core
DIV_1_5_EN	1	Output to ACORE	Enable/Disable divide-by-3 divider after DCO core
HB_PA_DIV2_EN	1	Output to ACORE	Enable/Disable divide-by-2 divider which generates input signal of DPA HB
LB_PA_DIV2_EN	1	Output to ACORE	Enable/Disable divide-by-2 divider which generates input signal of DPA LB
HB_BUF_EN	1	Output to ACORE	Enable/Disable buffer of HB signal
LB_BUF_EN	1	Output to ACORE	Enable/Disable buffer of LB signal
Test_Divider_rst_a	1	Output to ACORE	Asynchronous reset signal for divide-by-16 divider used for test
Clk_Gating_Enable	1	Output to ACORE	Enable/Disable clock gating signal in TDC block (expected feature)
MISO	1	Output to pad	SPI standard interface (Master In Slave Out)
MOSI	1	Input from pad	SPI standard interface (Master Out Slave In)
SCLK	1	Input from pad	SPI standard interface (SPI clock)
SS_n	1	Input from pad	SPI standard interface (Select signal)
RESET	1	Input from pad	Reset for SPI and associated registers

Table B-3: Interface of LSD block

Signal	Word length	Input/Output/Supply	Comments
VDD	1	Supply	VDD for LSD block
VSS	1	Supply	VSS for LSD block
SEL_EDGE	1	DCORE output	SEL_EDGE signal for retimer and incrementor
FREF	1	DCORE input	Reference clock used for TDC test
DCO_TUNE_TF	5	Output to $\Sigma\Delta$ modulator	Fractional part of TB tuning word fed to $\Sigma\Delta$ modulator
DCO_IN_P_DS	7	DCORE output	Tuning words for drain and source terminal of switches in PVT bank
DCO_IN_P_G	7	DCORE output	Tuning words for gate terminal of switches in PVT bank
DCO_IN_A	6	DCORE output	Tuning words for Acquisition bank
DCO_IN_T	64	DCORE output	Tuning words for integer part of tracking bank
Mem_Clk	1	Output to SysMem	Clock for SysMem (CKR)

Signal	Word length	Input/Output/Supply	Comments
CTL_ARSTZ	1	Input from Sequencer	Asynchronous Reset for OP, OA and OT block
CTL_PLL_P	1	Input from Sequencer	Enable/Disable PVT path in LSD
CTL_PLL_A	1	Input from Sequencer	Enable/Disable Acquisition path in LSD
CTL_PLL_T	1	Input from Sequencer	Enable/Disable Tracking path in LSD
CTL_SRST_P	1	Input from Sequencer	Synchronous reset for OP block
CTL_SRST_A	1	Input from Sequencer	Synchronous reset for OA block
CTL_SRST_T	1	Input from Sequencer	Synchronous reset for OT block
FCW	32	Input from SPI	Frequency command word
GS_trigger	1	Input from Sequencere	Trigger gear-shifting event for tracking mode
Ktdc	13	Input from SPI	Ktdc value for TDC normalization
Ktdc_norm_start	1	Input from Sequencer	Start command for TDC normalization
Ktdc_norm_stop	1	Input from Sequencer	Stop command for TDC normalization
LSD_CKR	1	DCORE input	Clock for LSD (CKR)
LSD_Enable_IIR	4	Input from Sequencer	Enable signal for IIR filter bank in tracking path
LSD_G_word	2	Input from Sequencer	Choose proportional coefficient after gear-shifting
LSD_RST	1	Input from Sequencer	Reset signal for LSD block
Lambda0	2	Input from SPI	Coefficient for 1st IIR filter
Lambda1	2	Input from SPI	Coefficient for 2nd IIR filter
Lambda2	2	Input from SPI	Coefficient for 3rd IIR filter
Lambda3	2	Input from SPI	Coefficient for 4th IIR filter
MEM_DCO_P	7	Input from SPI	Preset tuning word value for PVT bank
MEM_DCO_A	6	Input from SPI	Preset tuning word value for Acquisition bank
MEM_DCO_T	6+5	Input from SPI	Preset tuning word value for Tracking bank
MEM_GAIN_P	3	Input from SPI	Kdco estimation value for PVT bank
MEM_GAIN_A	8	Input from SPI	Kdco estimation value for Acquisition bank
MEM_GAIN_T	11	Input from SPI	Kdco estimation value for Tracking bank
PHV_SMP	10	DCORE input	Incrementor output
PR	1	Input from SPI	Enable Phase rotation algorithm on digital side
Rho	3	Input from SPI	Integral gain for type-II PLL mode
SEQ_T2_trigger	1	Input from sequencer	Trigger the transition from type-I PLL to type-II PLL
SRST	1	Input from Sequencer	Reset for loop filter of tracking bank
TDC_Q	32	DCORE input	TDC raw output (output of dummy cells discarded)
CTL_REG_DCO	1	Input from Sequencer	Save DCO tuning words to registers
PHE	32	Output to SysMem	PHE signal for system snapshot
dPHE	32	Output to SysMem	dPHE signal for system snapshot

Signal	Word length	Input/Output/Supply	Comments
Proportional_out	32	Output to SysMem	Proportional_out signal in loop filter for system snapshot
Filter_out	32	Output to SysMem	Filter_out signal in loop filter for system snapshot
TDC_RISE	5	Output to TDCMem	TDC decoder output in closed-loop operation for test
TDC_RISE_test	5	Output to TDCMem	TDC decoder output in open-loop operation for test
REG_DCO_P	7	Output to SPI	Registered PVT bank tuning word
REG_DCO_A	6	Output to SPI	Registered Acquisition bank tuning word
REG_DCO_T	11	Output to SPI	Registered Tracking bank tuning word
Tv_avg	12	Output to SPI	Accumulated |TDC_FALL-TDC_RISE| value for Ktdc normalization
TypeII_OVp	1	Output to SPI	Overflow status in integral path of loop filter
TypeII_OVn	1	Output to SPI	Overflow status in integral path of loop filter
PHE_OVp	1	Output to SPI	Overflow status of PHE
PHE_OVn	1	Output to SPI	Overflow status of PHE

Table B-4: Interface of Sequencer

Signal	Word length	Input/Output/Supply	Comments
VDD	1	Supply	VDD for SPI block
VSS	1	Supply	VSS for SPI block
Enable_IIR	4	Input from SPI	Specify enable signal for IIR filter bank in tracking path
G_word	2	Input from SPI	Specify proportional coefficient after gear-shifting
MEM_TIME_P	7	Input from SPI	Specify the time to start PVT mode
MEM_TIME_P2A	7	Input from SPI	Specify the time of mode switchover from PVT mode to Acquisition mode
MEM_TIME_Ktdc_Norm	7	Input from SPI	Specify the time to start Ktdc normalization
MEM_TIME_A2T	7	Input from SPI	Specify the time of mode switchover from Acquisitioin mode to Tracking mode
MEM_TIME_GS	7	Input from SPI	Specify the time of gear-shifting event
MEM_TIME_Lambda_Shift	7	Input from SPI	Specify the time to activate IIR filter bank

Signal	Word length	Input/Output/Supply	Comments
MEM_TIME_TypeCon	7	Input from SPI	Specify the time of transition from Type-I PLL to Type-II PLL
MEM_TIME_REGDCO	7	Input from SPI	Specify the time to save DCO tuning words
reset	1	Input from SPI	reset for other blocks in DCORE
ARST	1	Output to ΣΔ modulator	Asynchronous reset for ΣΔ modulator
CTL_ARSTZ	1	Output to LSD	Asynchronous Reset for OP, OA and OT block
CTL_PLL_P	1	Output to LSD	Enable/Disable PVT path in LSD
CTL_PLL_A	1	Output to LSD	Enable/Disable Acquisition path in LSD
CTL_PLL_T	1	Output to LSD	Enable/Disable Tracking path in LSD
CTL_SRST_P	1	Output to LSD	Synchronous reset for OP block
CTL_SRST_A	1	Output to LSD	Synchronous reset for OA block
CTL_SRST_T	1	Output to LSD	Synchronous reset for OT block
CTL_REG_DCO	1	Output to LSD	Save DCO tuning words to registers
GS_trigger	1	Output to LSD	Trigger gear-shifting event for tracking mode
Ktdc_norm_start	1	Output to LSD	Start command for TDC normalization
Ktdc_norm_stop	1	Output to LSD	Stop command for TDC normalization
SEQ_T2_trigger	1	Output to LSD	Trigger the transition from type-I PLL to type-II PLL
SRST	1	Output to LSD	Reset for loop filter of tracking bank
LSD_Enable_IIR	4	Output to LSD	Enable signal for IIR filter bank in tracking path
LSD_G_word	2	Output to LSD	Choose proportional coefficient after gear-shifting
LSD_RST	1	Output to LSD	Reset signal for LSD block

Table B-5: Interface of SPI block

Signal	Word length	Input/Output/Supply	Comments
VDD	1	Supply	VDD for SPI block

Signal	Word length	Input/Output/Supply	Comments
VSS	1	Supply	VSS for SPI block
MISO	1	DCORE output	SPI standard interface (Master In Slave Out)
MOSI	1	DCORE input	SPI standard interface (Master Out Slave In)
SCLK	1	DCORE input	SPI standard interface (SPI clock)
SS_n	1	DCORE input	SPI standard interface (Select signal)
RESET	1	DCORE input	Reset for SPI and associated registers
FCW	32	Output to LSD	Frequency command word
Ktdc	13	Output to LSD	Ktdc value for TDC normalization
Lambda0	2	Output to LSD	Coefficient for 1st IIR filter
Lambda1	2	Output to LSD	Coefficient for 2nd IIR filter
Lambda2	2	Output to LSD	Coefficient for 3rd IIR filter
Lambda3	2	Output to LSD	Coefficient for 4th IIR filter
MEM_DCO_P	7	Output to LSD	Preset tuning word value for PVT bank
MEM_DCO_A	6	Output to LSD	Preset tuning word value for Acquisition bank
MEM_DCO_T	6+5	Output to LSD	Preset tuning word value for Tracking bank
MEM_GAIN_P	3	Output to LSD	Kdco estimation value for PVT bank
MEM_GAIN_A	8	Output to LSD	Kdco estimation value for Acquisition bank
MEM_GAIN_T	11	Output to LSD	Kdco estimation value for Tracking bank
Rho	3	Output to LSD	Integral gain for type-II PLL mode
CKV_SEL	1	DCORE output	Choose CKV from phase rotator output or external signal
FREF_SEL	1	DCORE output	Select FREF signal between DigitalFREF and AnalogFREF
FREFdither	1	DCORE output	Signal to enable/disable FREF dithering algorithm
PR	1	Output to LSD	Enable Phase rotation algorithm on digital side
PR_en	1	DCORE output	Enable signal for Phase Rotator
DPA_ACW_HB	4	DCORE output	Amplitude control word for DPA HB
DPA_ACW_LB	4	DCORE output	Amplitude control word for DPA LB
DPA_CCW	4	DCORE output	Control word of capacitor bank for DPA

Signal	Word length	Input/Output/Supply	Comments
DIV_1_5_EN	1	DCORE output	Enable/Disable divide-by-3 divider after DCO core
HB_PA_DIV2_EN	1	DCORE output	Enable/Disable divide-by-2 divider which generates input signal of DPA HB
LB_PA_DIV2_EN	1	DCORE output	Enable/Disable divide-by-2 divider which generates input signal of DPA LB
HB_BUF_EN	1	DCORE output	Enable/Disable buffer of HB signal
LB_BUF_EN	1	DCORE output	Enalbe/Disable buffer of LB signal
DCO_tail_resistor	5	DCORE output	Control word for tail resistor in DCO core
Test_Divider_rst_a	1	DCORE output	Asynchronous reset signal for divide-by-16 divider used for test
Clk_Gating_Enable	1	DCORE output	Enable/Disable clock gating signal in TDC block (expected feature)
Enable_IIR	4	Output to sequencer	Specify enable signal for IIR filter bank in tracking path
G_word	2	Output to Sequencer	Specify proportional coefficient after gear-shifting
MEM_TIME_P	7	Output to Sequencer	Specify the time to start PVT mode
MEM_TIME_P2A	7	Output to Sequencer	Specify the time of mode switchover from PVT mode to Acquisition mode
MEM_TIME_Ktdc_Norm	7	Output to Sequencer	Specify the time to start Ktdc normalization
MEM_TIME_A2T	7	Output to Sequencer	Specify the time of mode switchover from Acquisitioin mode to Tracking mode
MEM_TIME_GS	7	Output to Sequencer	Specify the time of gear-shifting event
MEM_TIME_Lambda_Shift	7	Output to Sequencer	Specify the time to activate IIR filter bank
MEM_TIME_TypeCon	7	Output to Sequencer	Specify the time of transition from Type-I PLL to Type-II PLL
MEM_TIME_REGDCO	7	Output to Sequencer	Specify the time to save DCO tuning words
Seq_Clk_SEL	1	Output to Sequencer	Select the clock for sequencer
REG_DCO_P	7	Input from LSD	Registered PVT bank tuning word
REG_DCO_A	6	Input from LSD	Registered Acquisition bank tuning word
REG_DCO_T	11	Input from LSD	Registered Tracking bank tuning word

Signal	Word length	Input/Output/Supply	Comments
Tv_avg	12	Input from LSD	Accumulated |TDC_FALL-TDC_RISE| value for Ktdc normalization
SysMem_SEL	2	Output to SysMem	Choose SysMem input between PHE, dPHE, Filter_output and Filter_output
SysMemOut	32	Input from SysMem	Word of SysMem with address of Addr_SysMem
TDCMem_SEL	1	Output to TDCMem	Choose TDCMem input between TDC_RISE and TDC_RISE_test
TDCMemOut	5	Input from TDCMem	Word of TDCMem with address of Addr_TDCMem
TDCMemClk_SEL	1	Output to TDCMem	Choose TDCMem clock between CKR and FREF
Addr_SysMem	12	Output to SysMem	Address of SysMem word to be read out
Addr_TDCMem	12	Output to TDCMem	Address of TDCMem word to be read out
TypeII_OVp	1	Input from LSD	Overflow status in integral path of loop filter
TypeII_OVn	1	Input from LSD	Overflow status in integral path of loop filter
PHE_OVp	1	Input from LSD	Overflow status of PHE
PHE_OVn	1	Input from LsD	Overflow status of PHE
Core_RESET	1	Output to sequencer	reset for other blocks in DCORE

Appendix C

Register Map of the SPI Block

Table C-1: The register map of the SPI block.

Physical Register	Address	Corresponding Internal Register	Meaning(Internal control signal)	Default value
CONTROL	6'h00	control	Control word of SPI	8'h00
STATUS	6'h01	status	Status word of SPI (2'b10: Idle; 2'b00: command input; 2'b01: data access)	8'h00
FCW0_0	6'h04	fcw_reg	Frequency command word (32 bits)	219815571
FCW0_1	6'h05			
FCW0_2	6'h06			
FCW0_3	6'h07			
SysMem_0	6'h09	addr_sysmem_reg	Read address for SysMem (12 bits)	0
SysMem_1	6'h0A			
TDCMem_0	6'h0B	addr_tdcmem_reg	Read address for TDCMem(16 bits)	0
TDCMem_1	6'h0C			
DPAACW	6'h10	dpa_acw_reg	{DPA_ACW_HB[3:0], DPA_ACW_LB[3:0]}	8'b0000_0000
DPACCW	6'h11	dpa_ccw_reg	DPA_CCW[3:0]	4'b0010
DCOdivctrl	6'h12	dco_divctrl_reg	{HB_BUF_EN, LB_BUF_EN, HB_PA_DIV2_EN, LB_PA_DIV2_EN, DIV_1_5_EN}	5'b00000
DCOtailres	6'h13	dco_tailres_reg	DCO_tail_resistor[4:0]	5'b00000
ClkGating	6'h14	clkgating_reg	{Test_Divider_rst_a, Seq_Clk_SEL, Clk_Gating_Enable}	3'b100
ckvfrefsel	6'h16	ckvfref_sel_reg	{SysMem_SEL[1:0], FREF_SEL, CKV_SEL, FREFdither, PR, PR_en}	6'b11_1000
MEMDCOP	6'h19	memdcop_reg	MEM_DCO_P[6:0]	0
MEMDCOA	6'h1A	memdcoa_reg	MEM_DCO_A[5:0]	0

Physical Register	Address	Corresponding Internal Register	Meaning(Internal control signal)	Default value
MEMDCOT_0	6'h1B	memdcot_reg	MEM_DCO_T[10:0]	0
MEMDCOT_1	6'h1C			
MEMGAINP	6'h1D	memgainp_reg	MEM_GAIN_P[2:0]	4
MEMGAINA	6'h1E	memgaina_reg	MEM_GAIN_A[7:0]	51
MEMGAINT_0	6'h1F	memgaint_reg	MEM_GAIN_T[10:0]	655
MEMGAINT_1	6'h20			
RhoGword	6'h21	rhogword_reg	{Rho[1:0],G_word[2:0]}	5'b11111
Lambda	6'h22	lambda_reg	{Lambda3[1:0], Lambda2[1:0], Lambda1[1:0], Lambda0[1:0]}	8'b00_10_10_01
EnableIIR	6'h23	enableiir_reg	Enable_IIR[3:0]	4'b0011
Ktdc_0	6'h24	Ktdc_reg	Ktdc[12:0]	2846
Ktdc_1	6'h25			
TIMEP	6'h28	timep_reg	MEM_TIME_P[6:0]	0
TIMEP2A	6'h29	timep2a_reg	MEM_TIME_P2A[6:0]	0
TIMEKtdc	6'h2A	timektdc_reg	MEM_TIME_Ktdc_Norm[6:0]	0
TIMEA2T	6'h2B	timea2t_reg	MEM_TIME_A2T[6:0]	0
TIMEGS	6'h2C	timegs_reg	MEM_TIME_GS[6:0]	0
TIMELambda	6'h2D	timelambda_reg	MEM_TIME_Lambda_Shift[6:0]	0
TIMEType	6'h2E	timetype_reg	MEM_TIME_TypeCon[6:0]	0
TIMEREGDCO	6'h2F	timeregdco_reg	MEM_TIME_REGDCO[6:0]	0

Appendix D

Figures for TDC Open-Loop Test

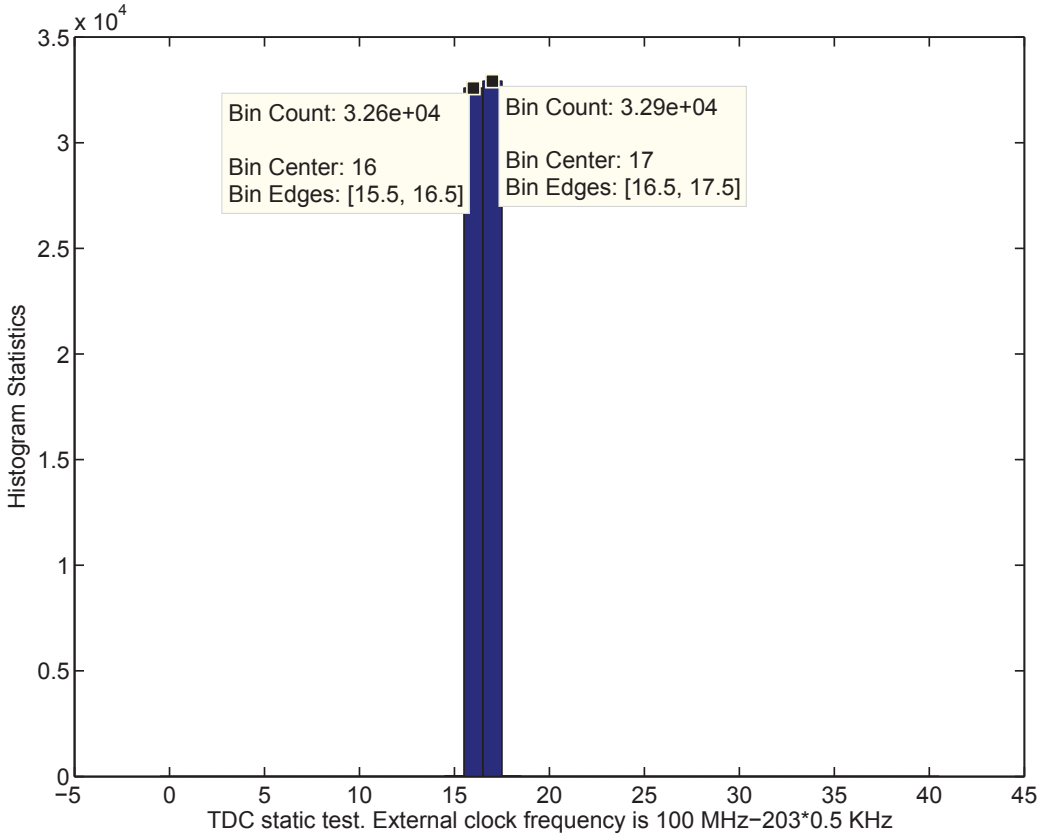

Figure D-1: TDC static test histogram (f_{ext} is 100 MHz-203*0.5 kHz).

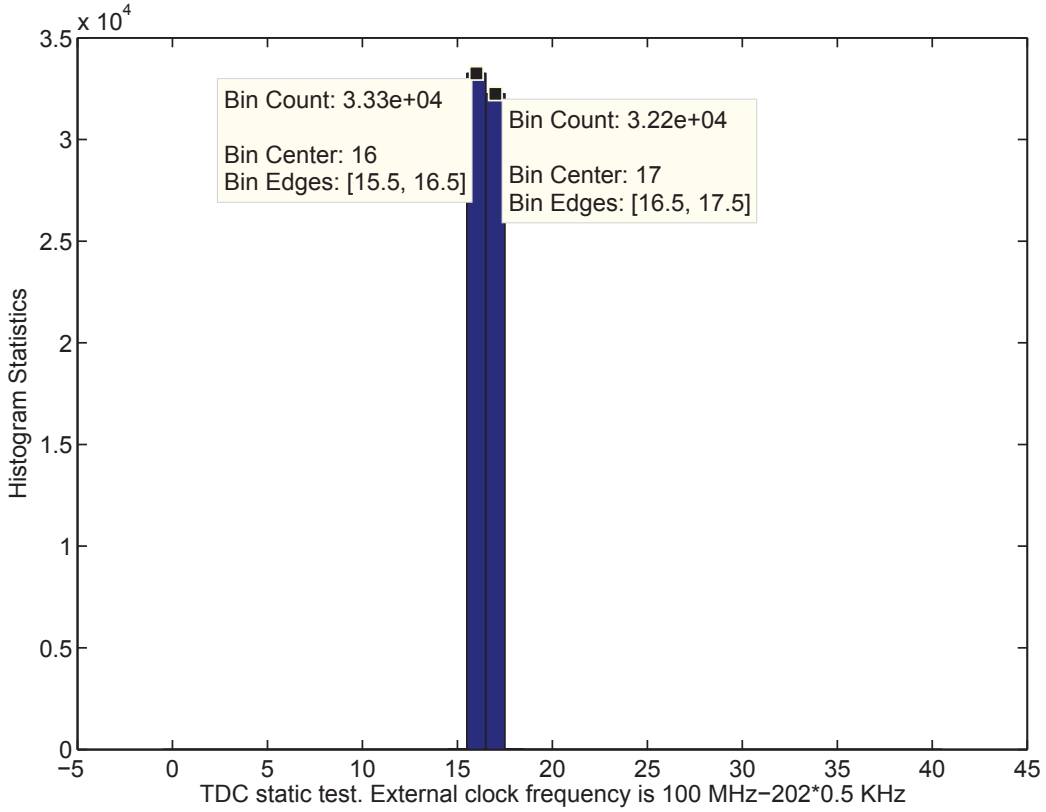

Figure D-2: TDC static test histogram (f_{ext} is 100 MHz-202*0.5 kHz).

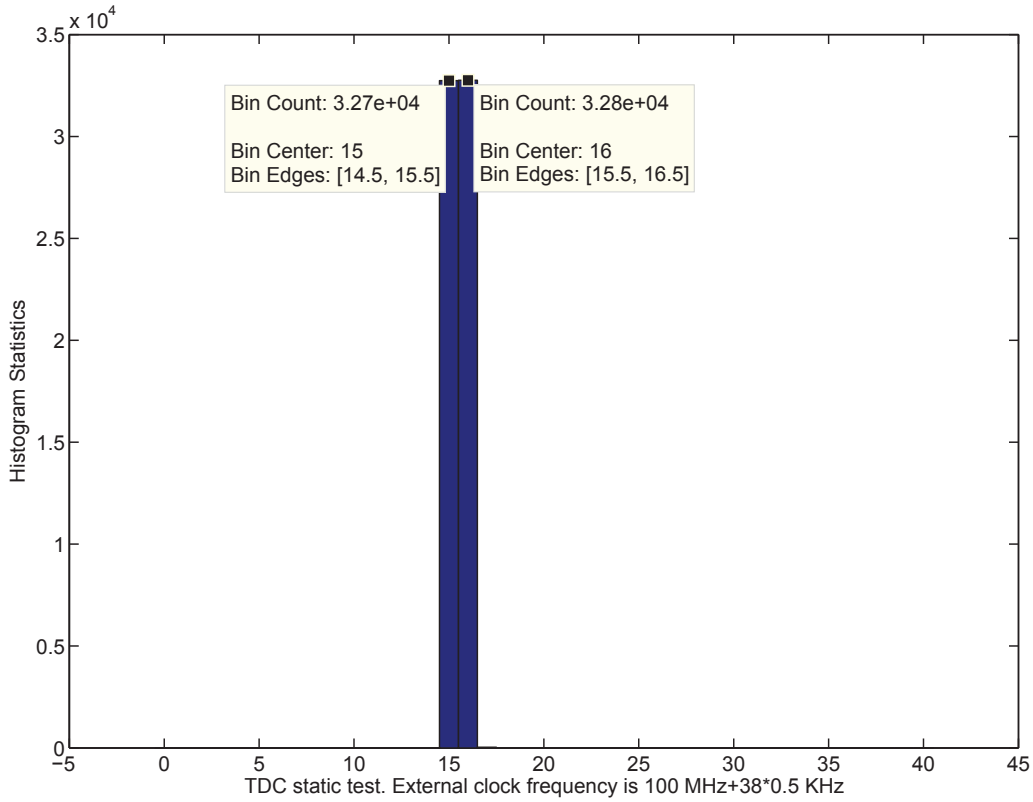

Figure D-3: TDC static test histogram (f_{ext} is 100 MHz+38*0.5 kHz).

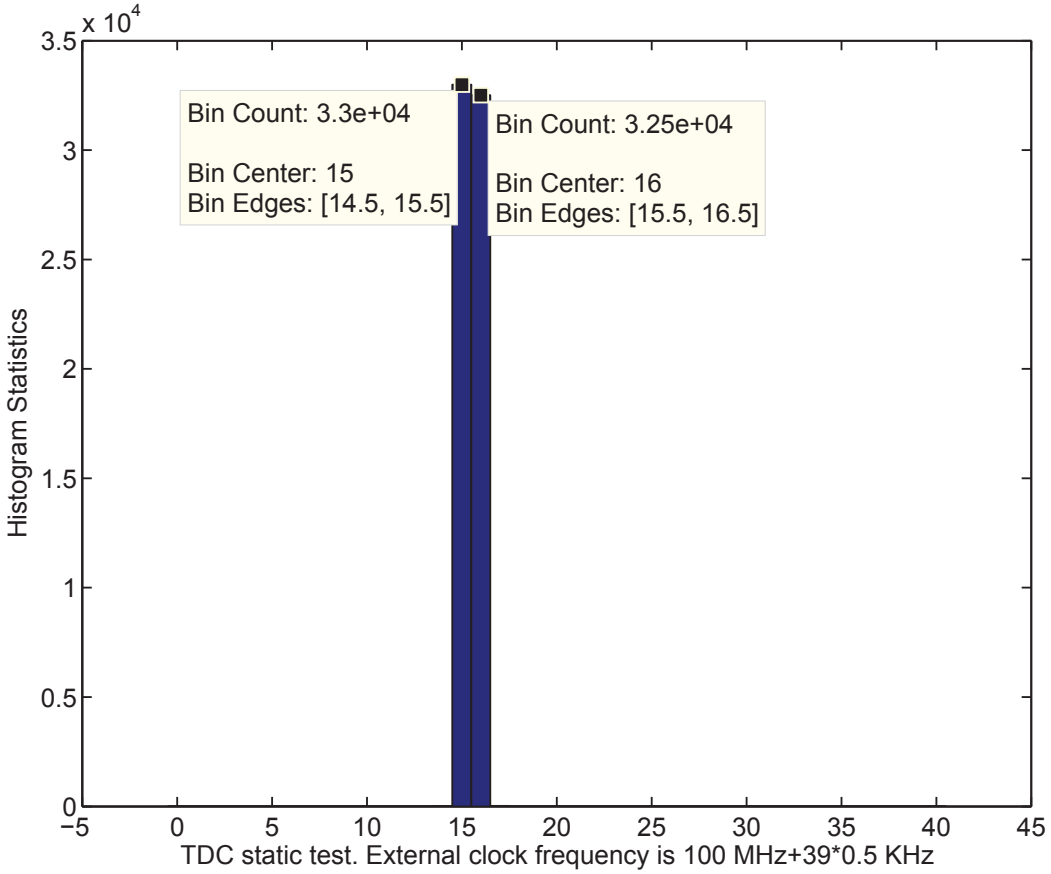

Figure D-4: TDC static test histogram (f_{ext} is 100 MHz+39*0.5 kHz).

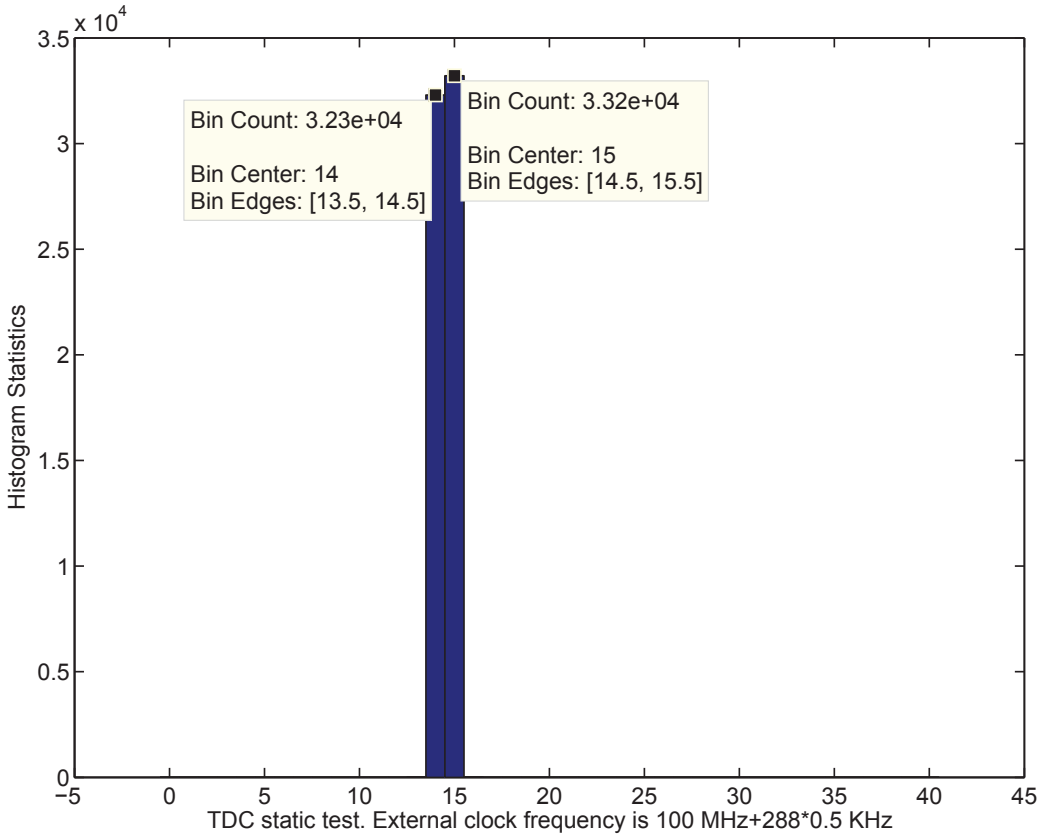

Figure D-5: TDC static test histogram (f_{ext} is 100 MHz+288*0.5 kHz).

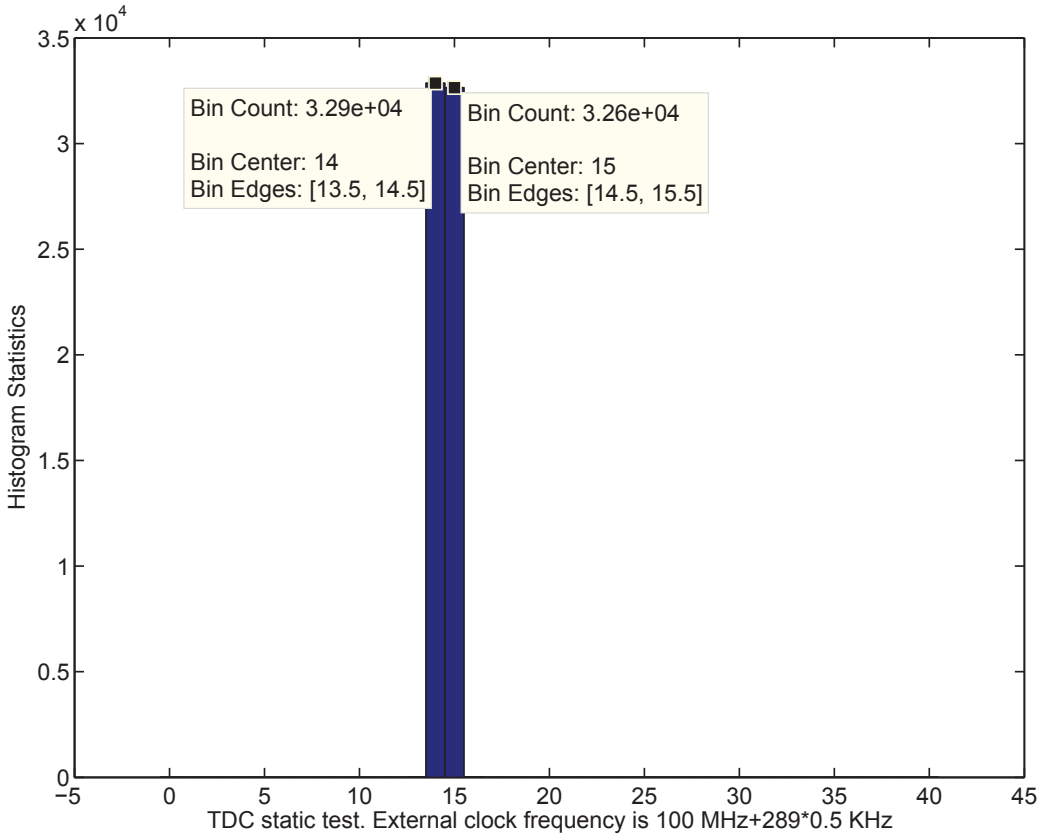

Figure D-6: TDC static test histogram (f_{ext} is 100 MHz+289*0.5 kHz).

Appendix E

Figures for ADPLL Top Level Simulation

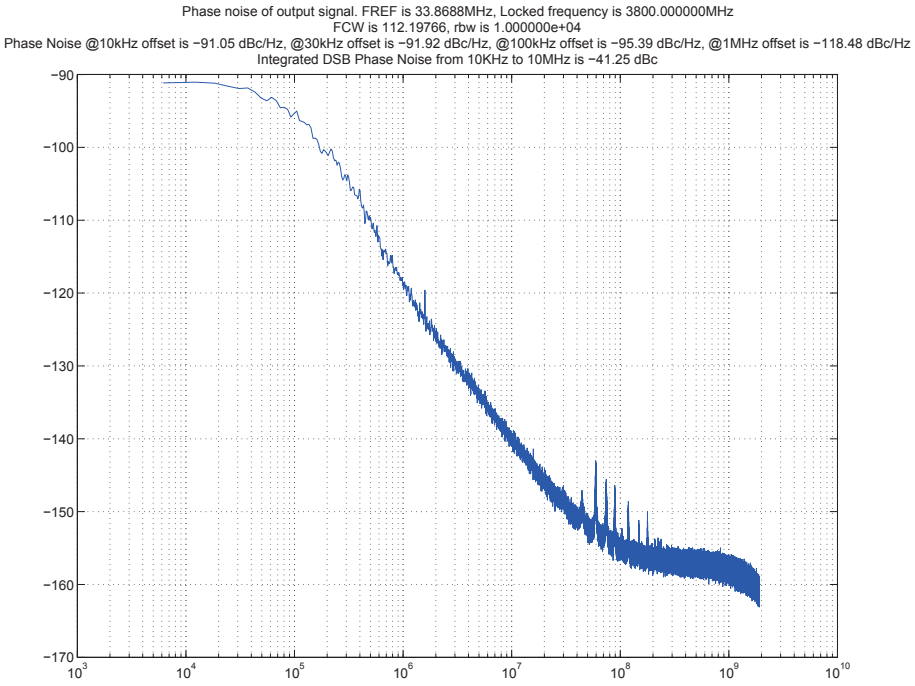

Figure E-1: Phase noise result. f_v=3800 MHz,T_{inv}=12.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

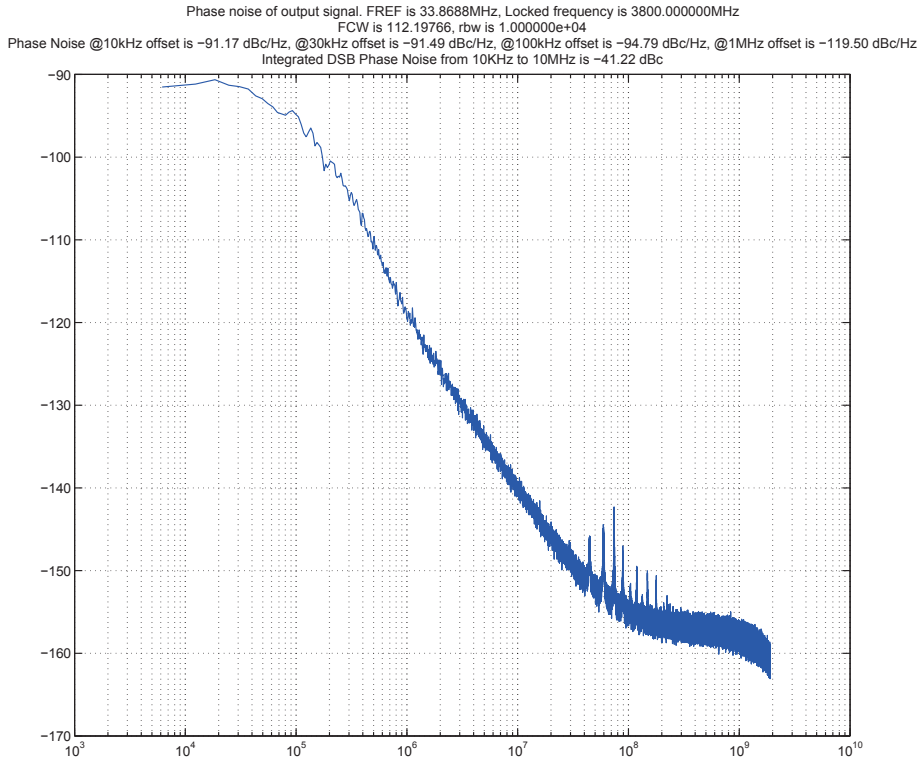

Figure E-2: Phase noise result. f_v=3800 MHz,T_{inv}=11.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

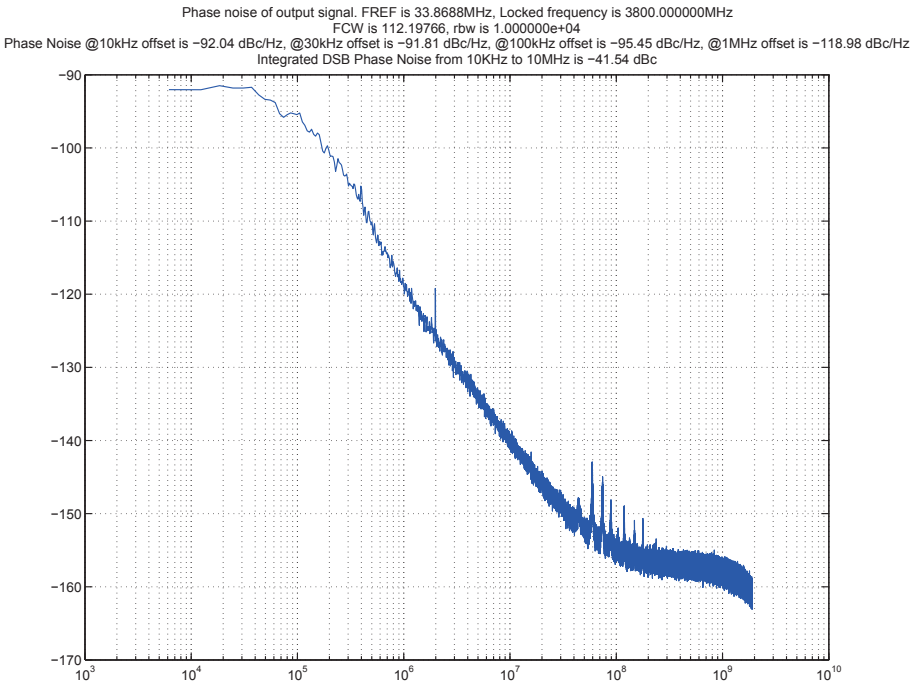

Figure E-3: Phase noise result. f_v=3800 MHz,T_{inv}=10.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

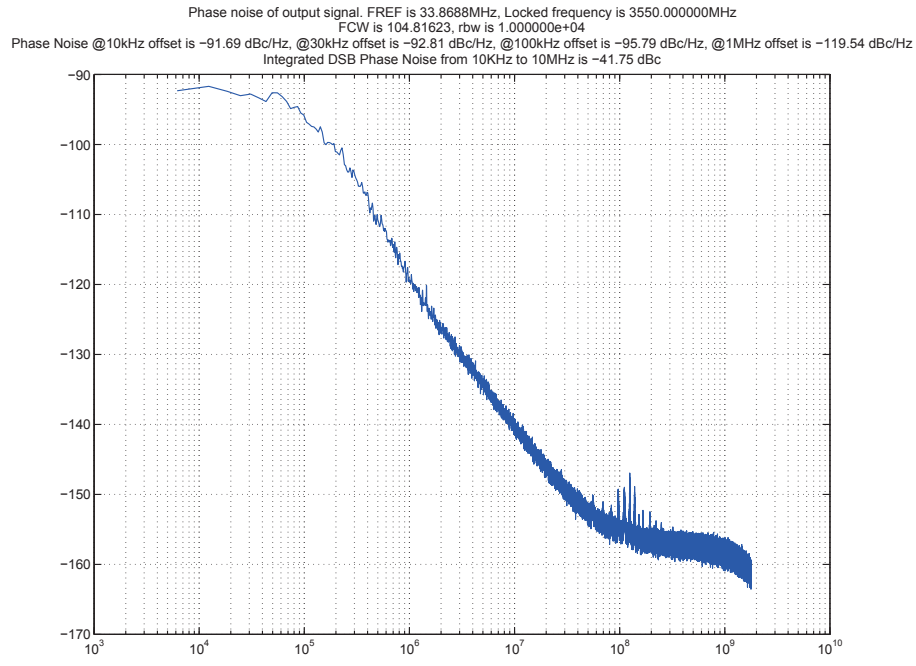

Figure E-4: Phase noise result. f_v=3550 MHz,T_{inv}=12.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

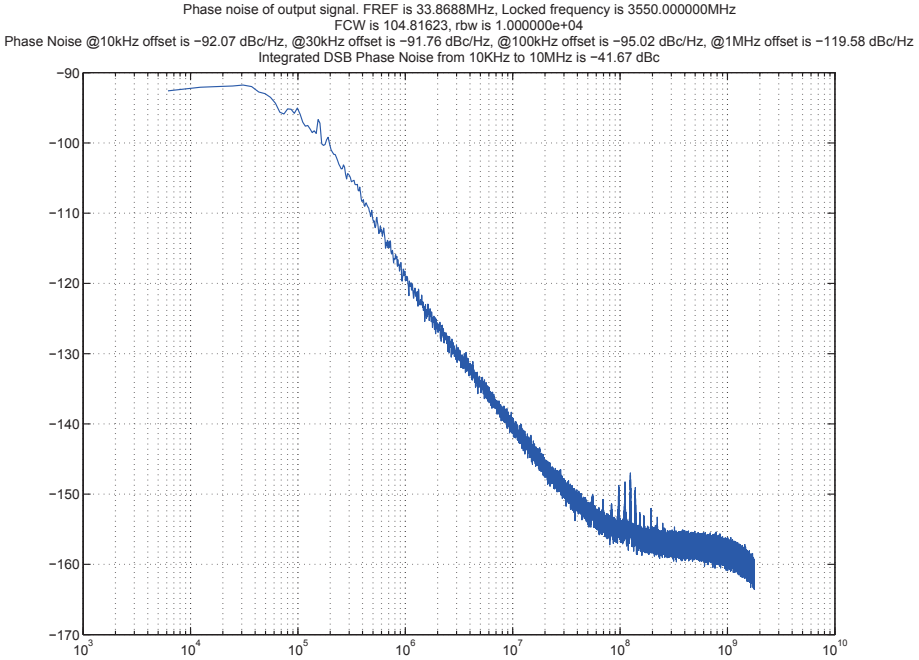

Figure E-5: Phase noise result. f_v=3550 MHz,T_{inv}=11.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

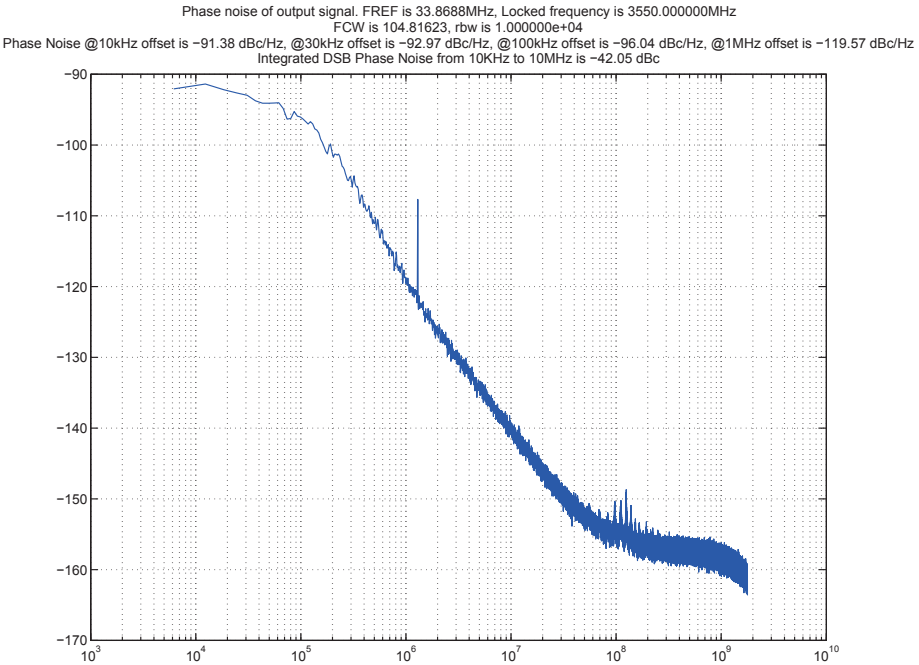

Figure E-6: Phase noise result. f_v=3550 MHz,T_{inv}=10.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

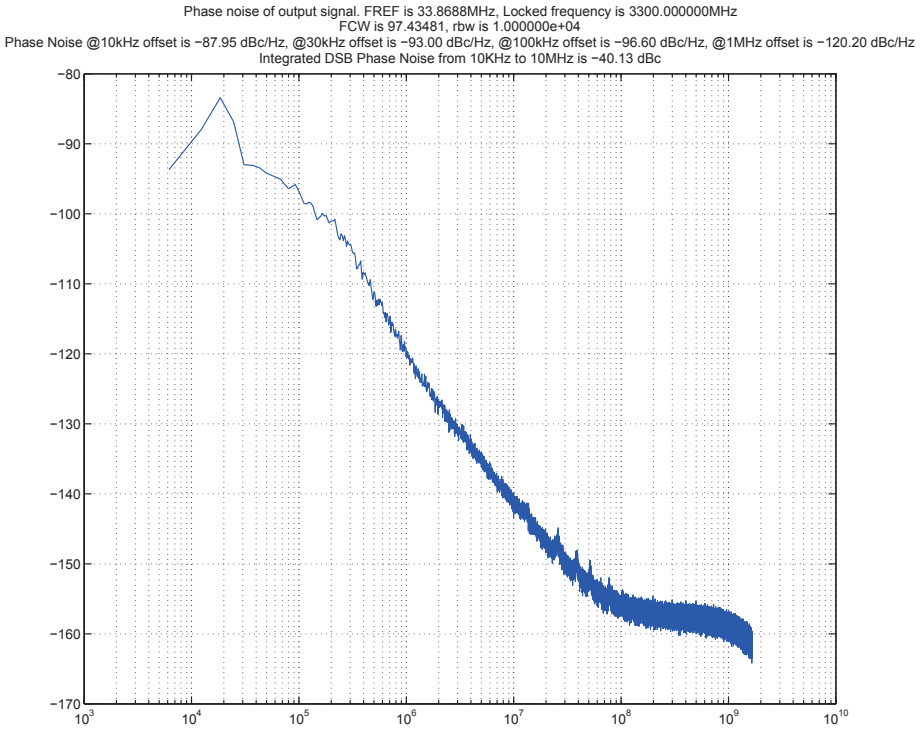

Figure E-7: Phase noise result. f_v=3300 MHz,T_{inv}=12.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

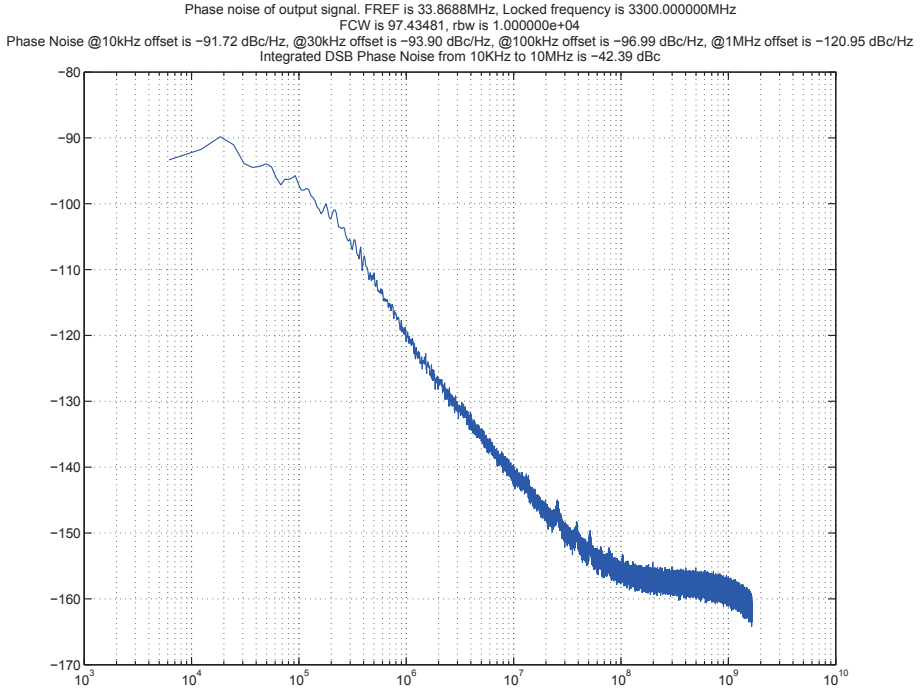

Figure E-8: Phase noise result. f_v=3300 MHz,T_{inv}=11.5 ps, f_R=33.8688 MHz.RBW=10 kHz.

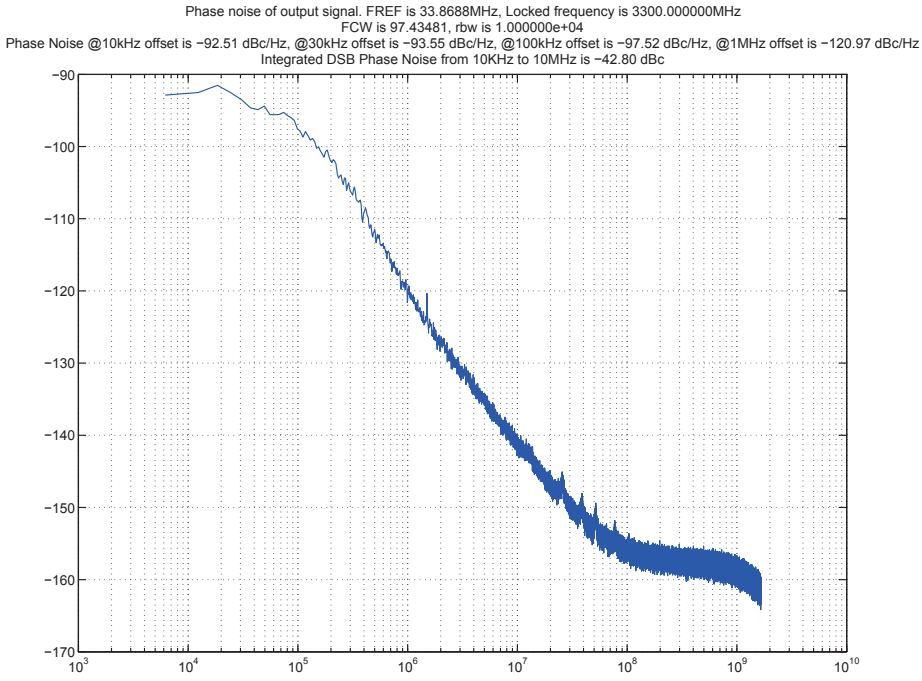

Figure E-9: Phase noise result. f_v=3300 MHz, T_{inv}=10.5 ps, f_R=33.8688 MHz. RBW=10 kHz.

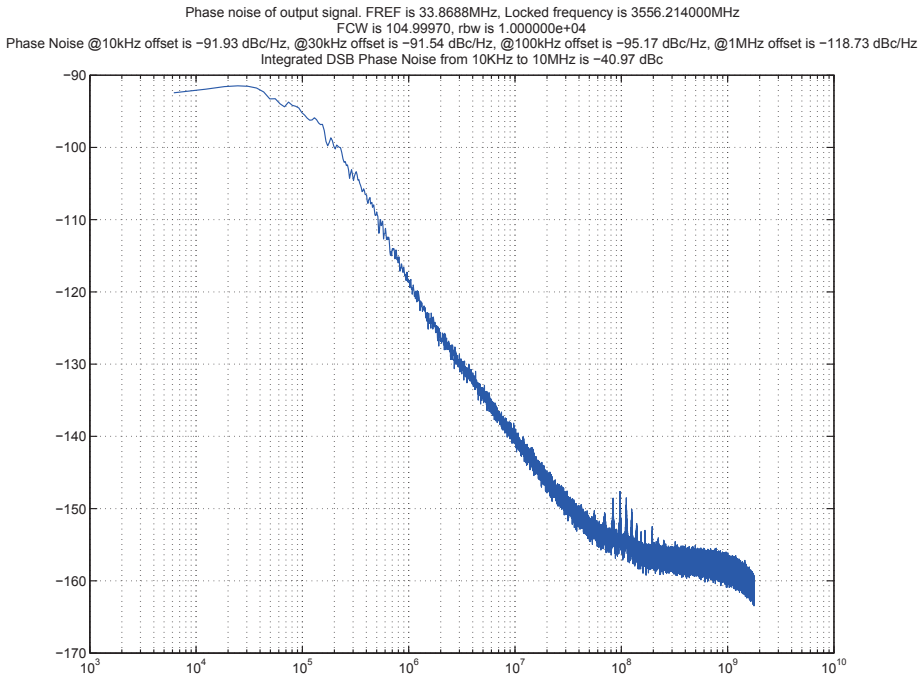

Figure E-10: Phase noise result. f_v=3556.214 MHz, T_{inv}=12.5 ps, f_R=33.8688 MHz. RBW=10 kHz.

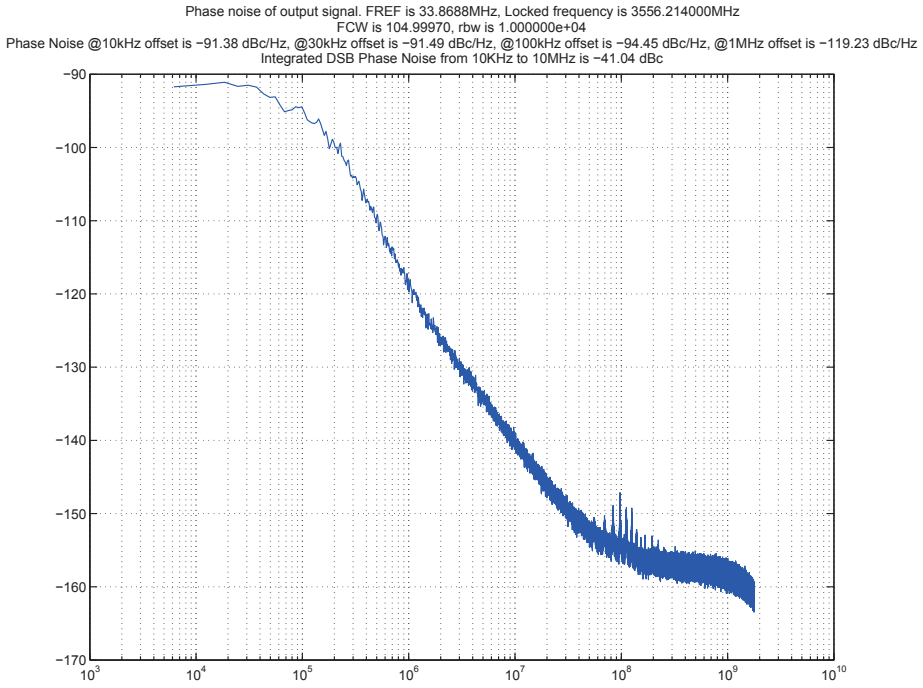

Figure E-11: Phase noise result. f_v=3556.214 MHz, T_{inv}=11.5 ps, f_R=33.8688 MHz. RBW=10 kHz.

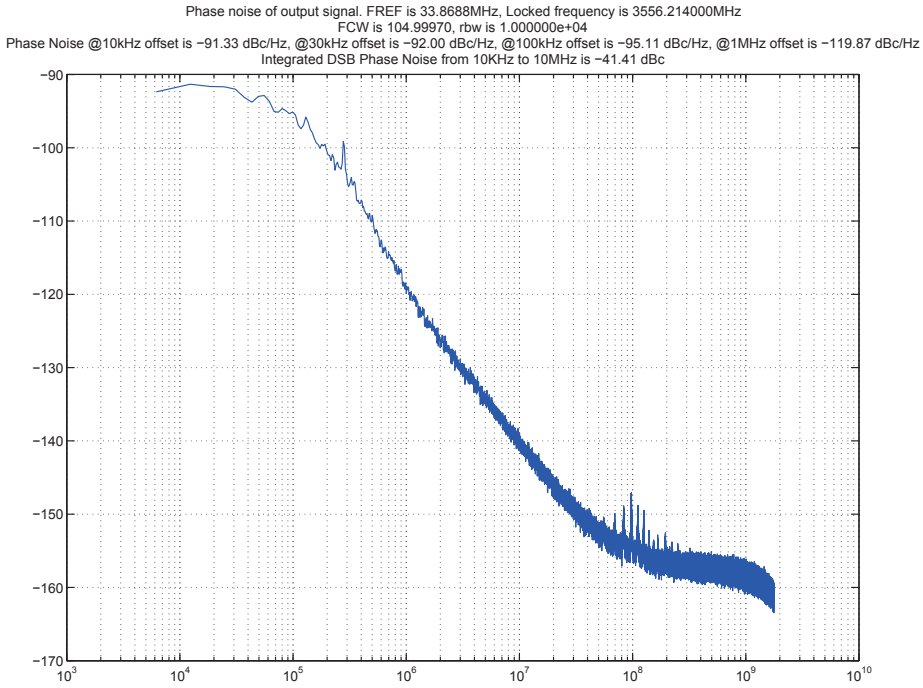

Figure E-12: Phase noise result. f_v=3556.214 MHz, T_{inv}=10.5 ps, f_R=33.8688 MHz. RBW=10 kHz.

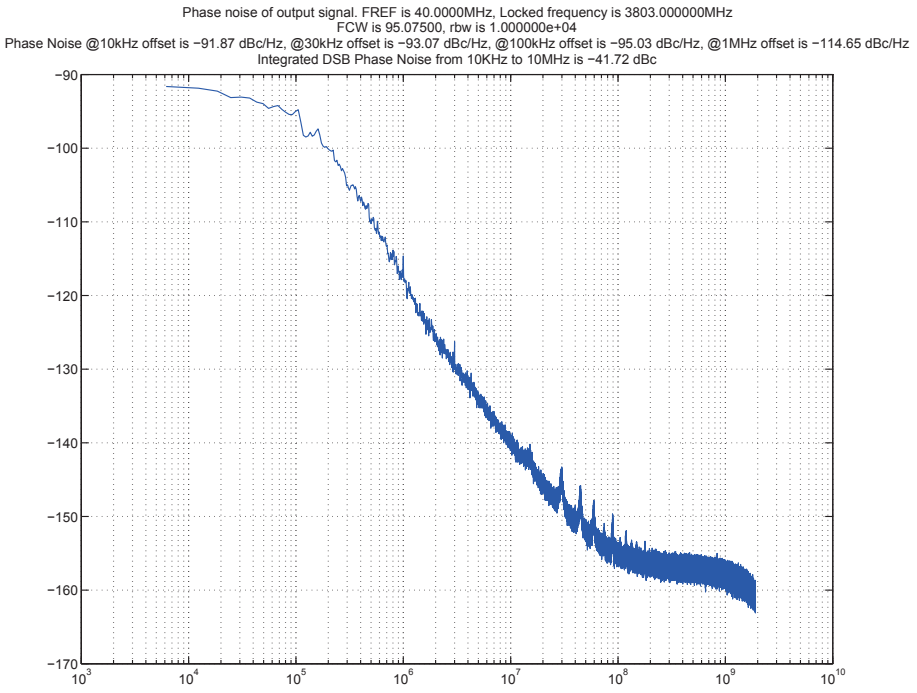

Figure E-13: Phase noise result.f_v=3803 MHz,T_{inv}=12.45 ps,f_R=40 MHz.RBW=10 kHz.

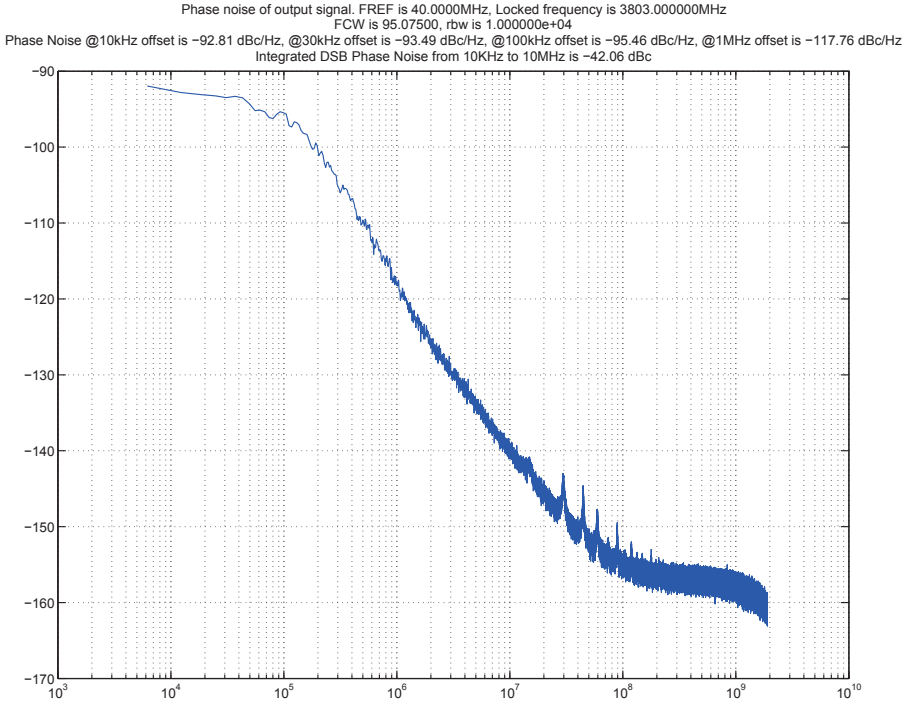

Figure E-14: Phase noise result.f_v=3803 MHz,T_{inv}=11.5 ps,f_R=40 MHz.RBW=10 kHz.

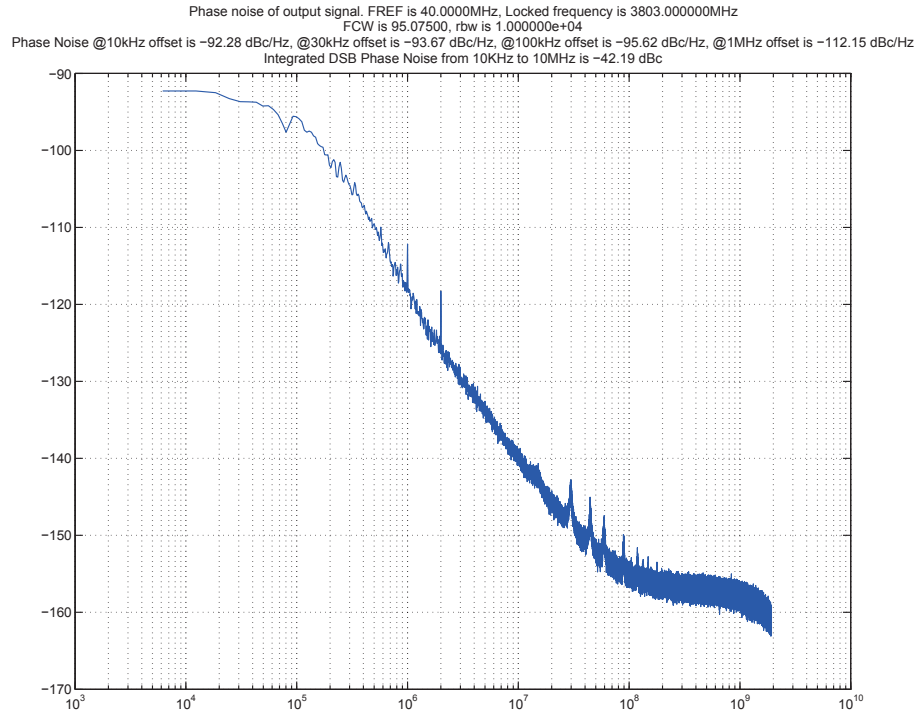

Figure E-15: Phase noise result.f_v=3803 MHz,T_{inv}=10.5 ps,f_R=40 MHz.RBW=10 kHz.

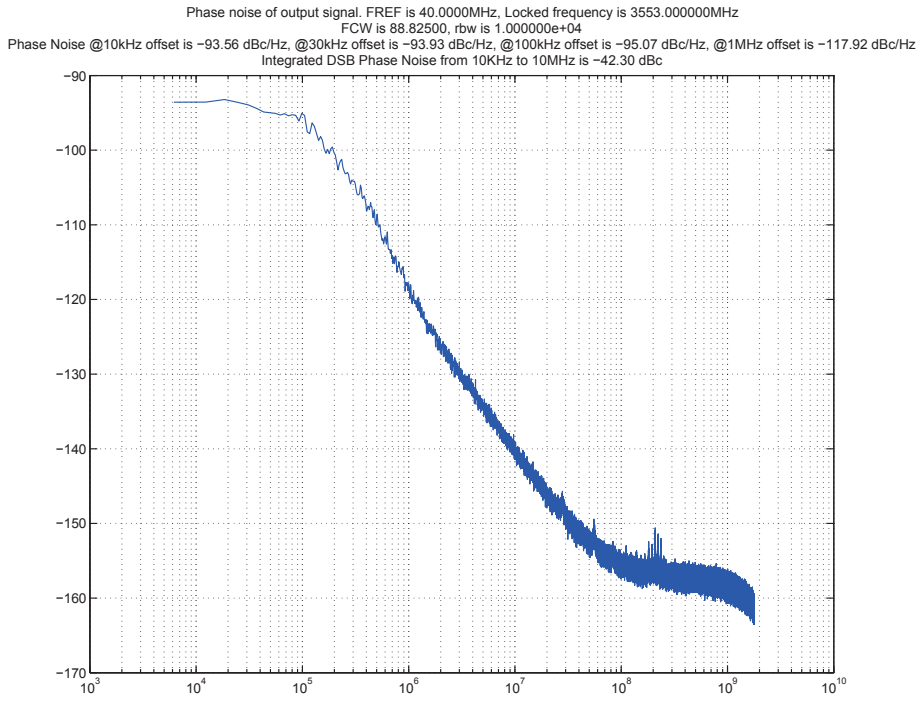

Figure E-16: Phase noise result.f_v=3553 MHz,T_{inv}=12.45 ps,f_R=40 MHz.RBW=10 kHz.

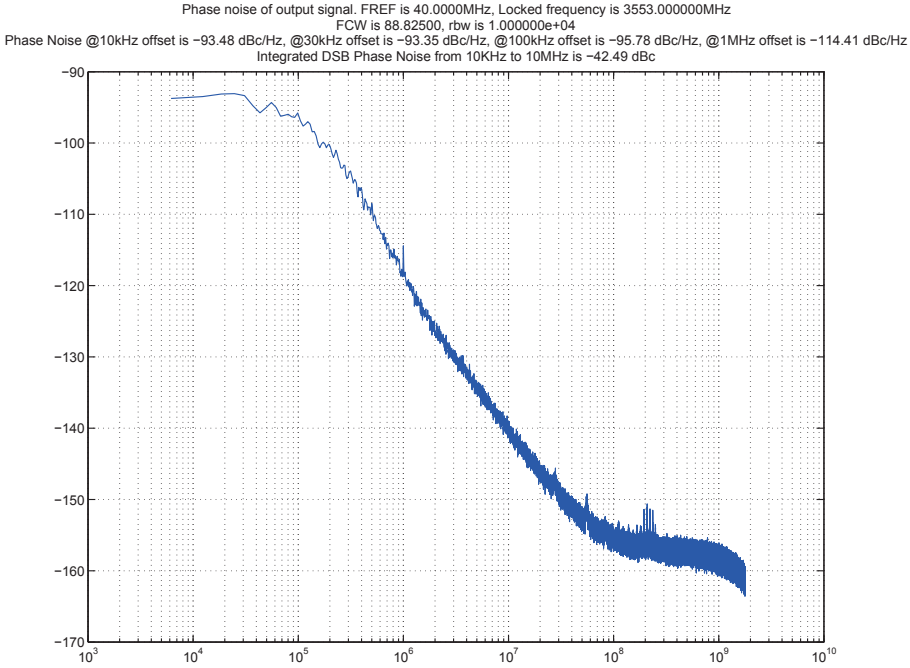

Figure E-17: Phase noise result.f_v=3553 MHz,T_{inv}=11.5 ps,f_R=40 MHz.RBW=10 kHz.

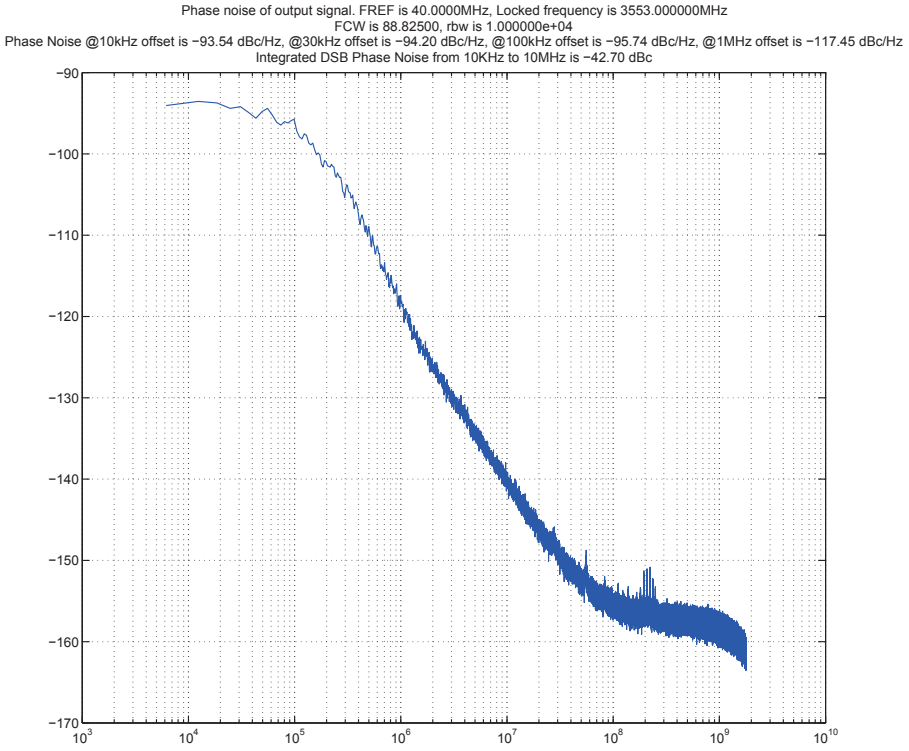

Figure E-18: Phase noise result.f_v=3553 MHz,T_{inv}=10.5 ps,f_R=40 MHz.RBW=10 kHz.

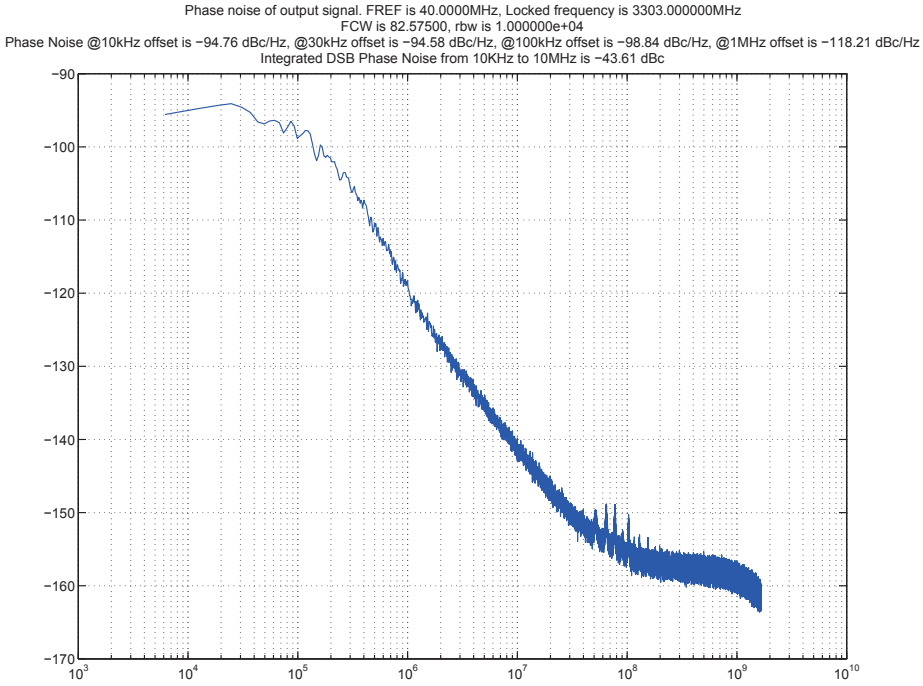

Figure E-19: Phase noise result.f_v=3303 MHz,T_{inv}=12.45 ps,f_R=40 MHz.RBW=10 kHz.

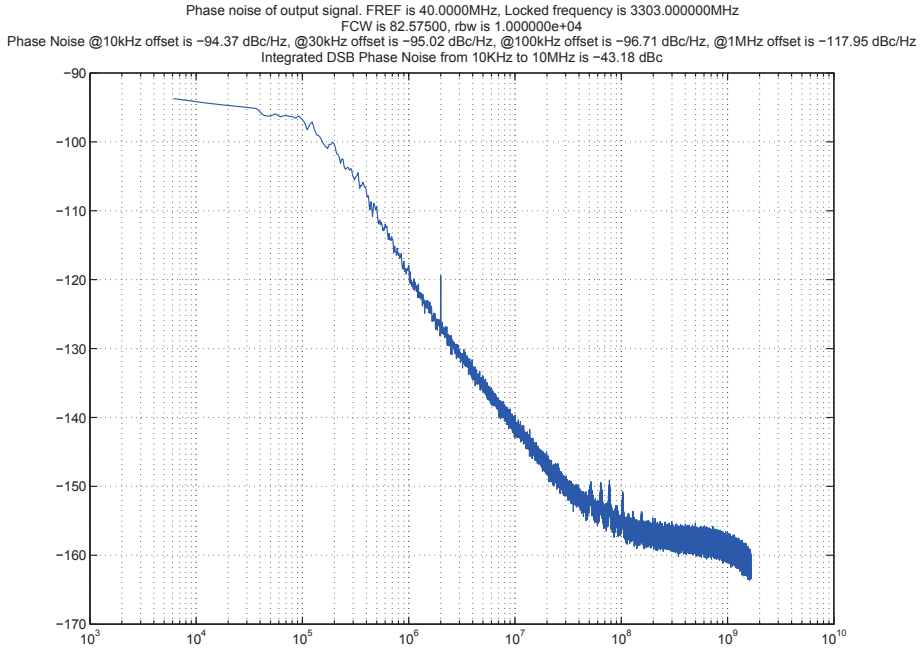

Figure E-20: Phase noise result.f_v=3303 MHz,T_{inv}=11.5 ps,f_R=40 MHz.RBW=10 kHz.

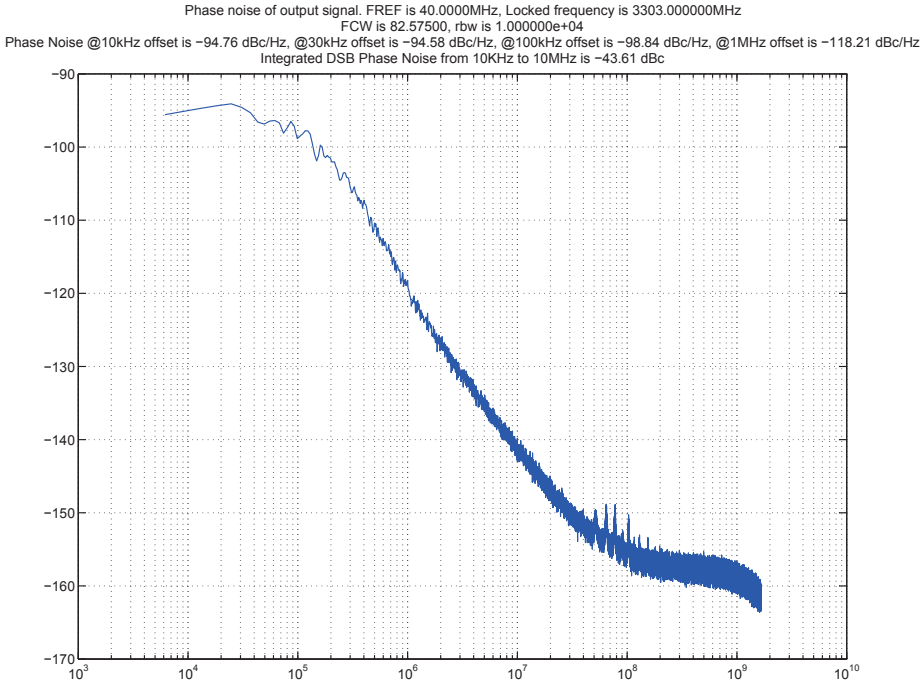

Figure E-21: Phase noise result.f_v=3303 MHz,T_{inv}=10.5 ps,f_R=40 MHz.RBW=10 kHz.

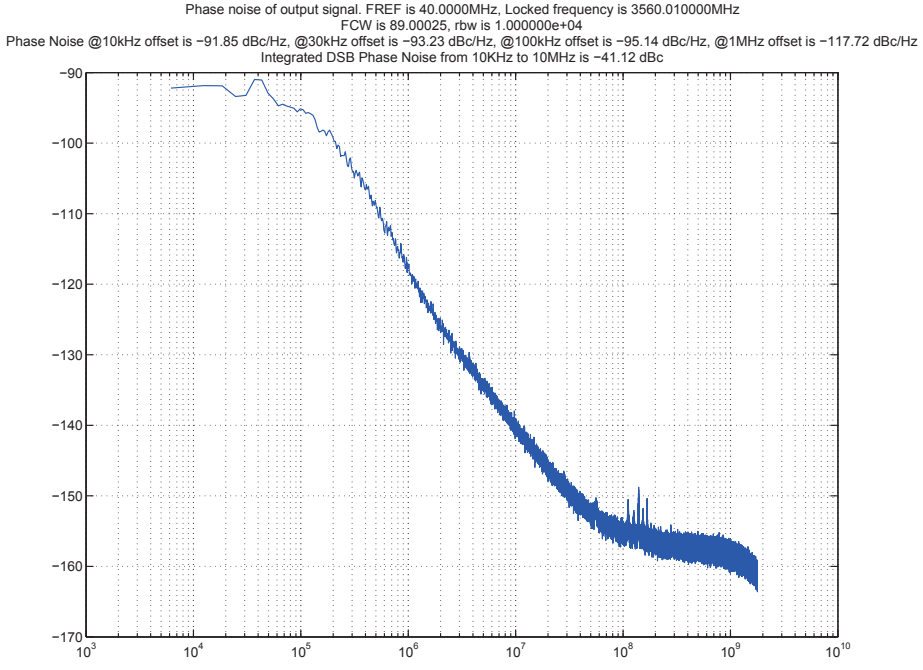

Figure E-22: Phase noise result.f_v=3560.01 MHz,T_{inv}=12.45 ps,f_R=40 MHz.RBW=10 kHz.

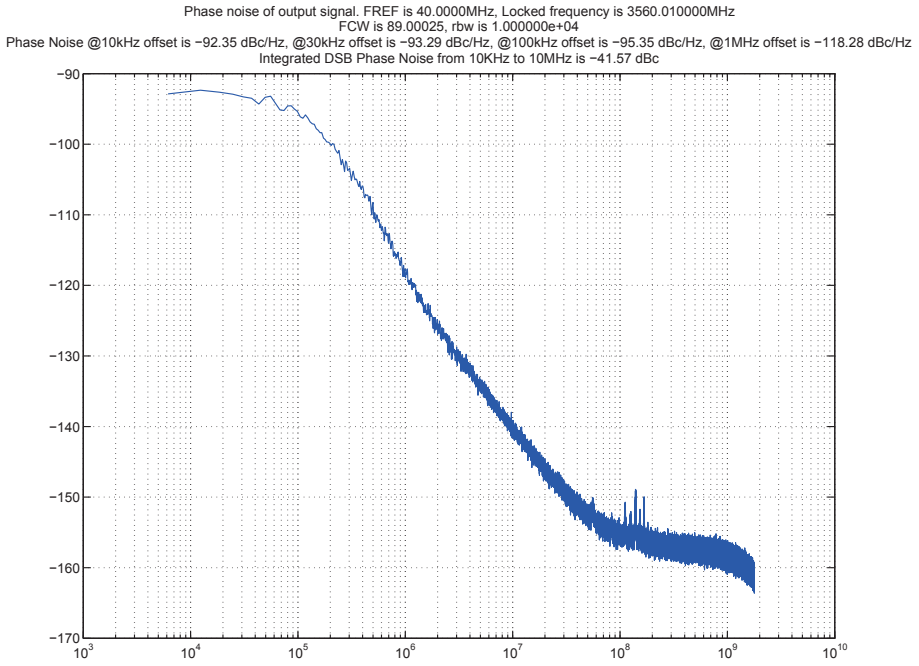

Figure E-23: Phase noise result. f_v=3560.01 MHz,T_{inv}=11.5 ps,f_R=40 MHz.RBW=10 kHz.

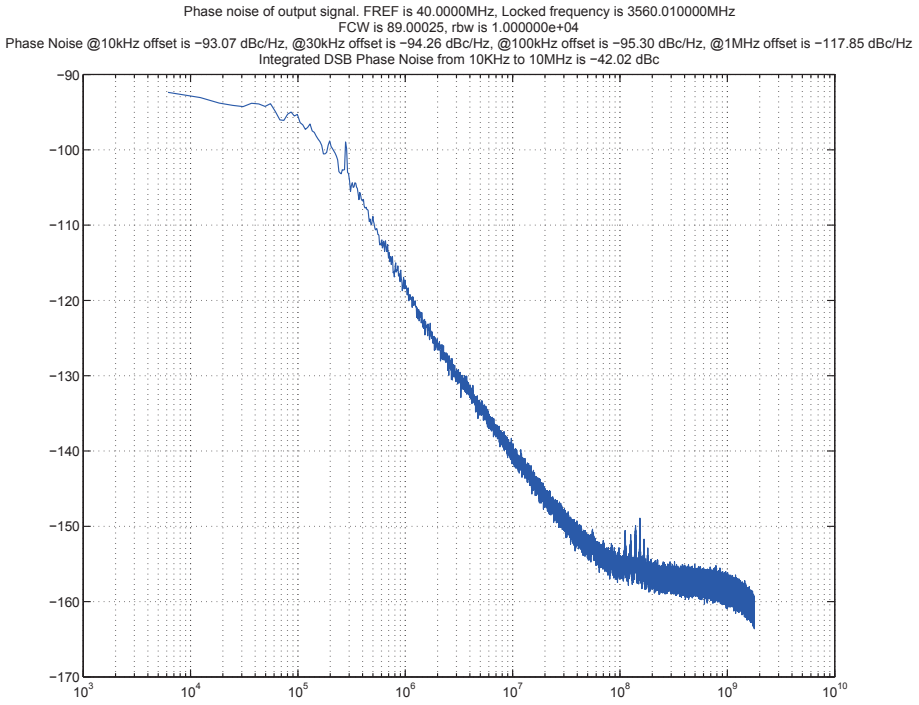

Figure E-24: Phase noise result. f_v=3560.01 MHz,T_{inv}=10.5 ps,f_R=40 MHz.RBW=10 kHz.

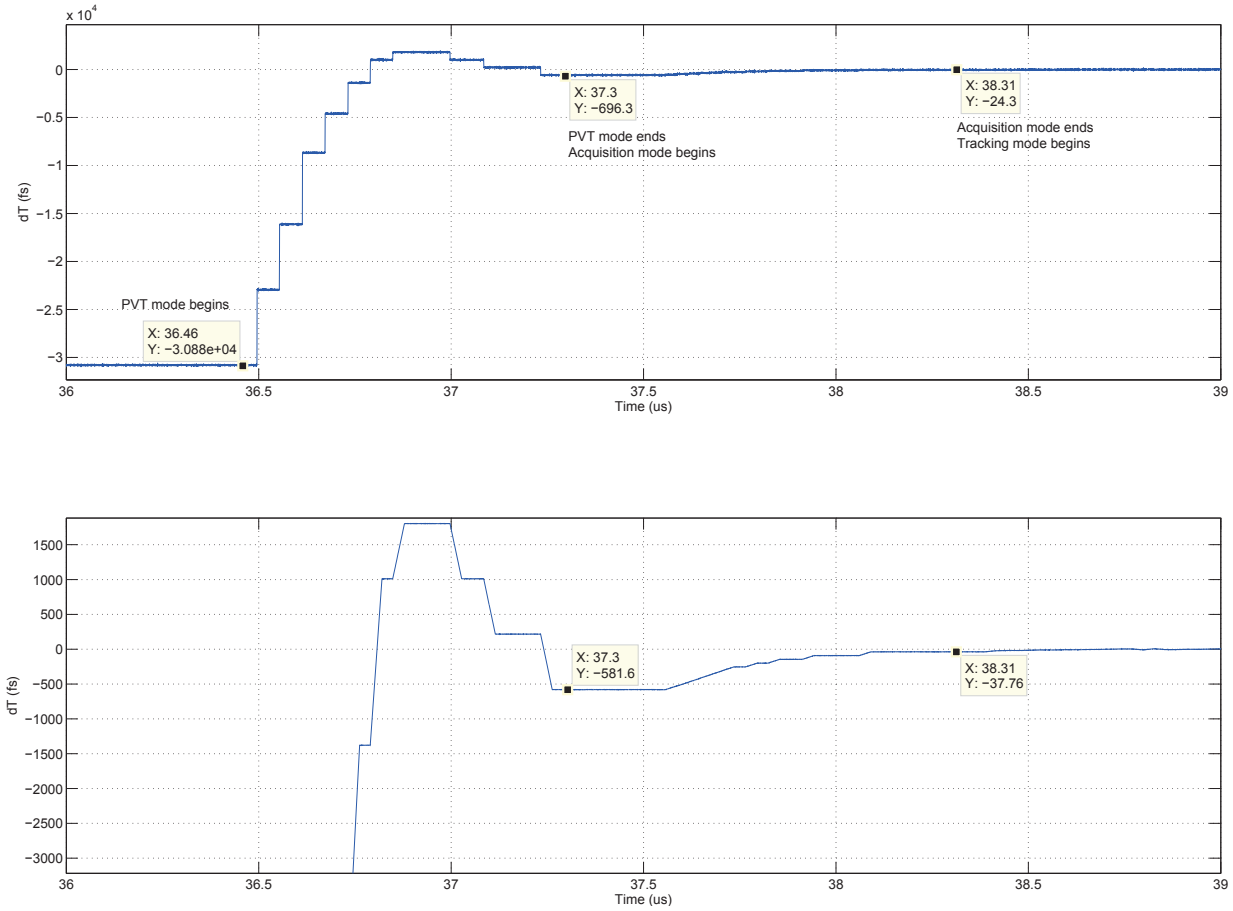

Figure E-25: ADPLL settling transient in PVT mode and acquisiton mode($f_v = 3300$ MHz)

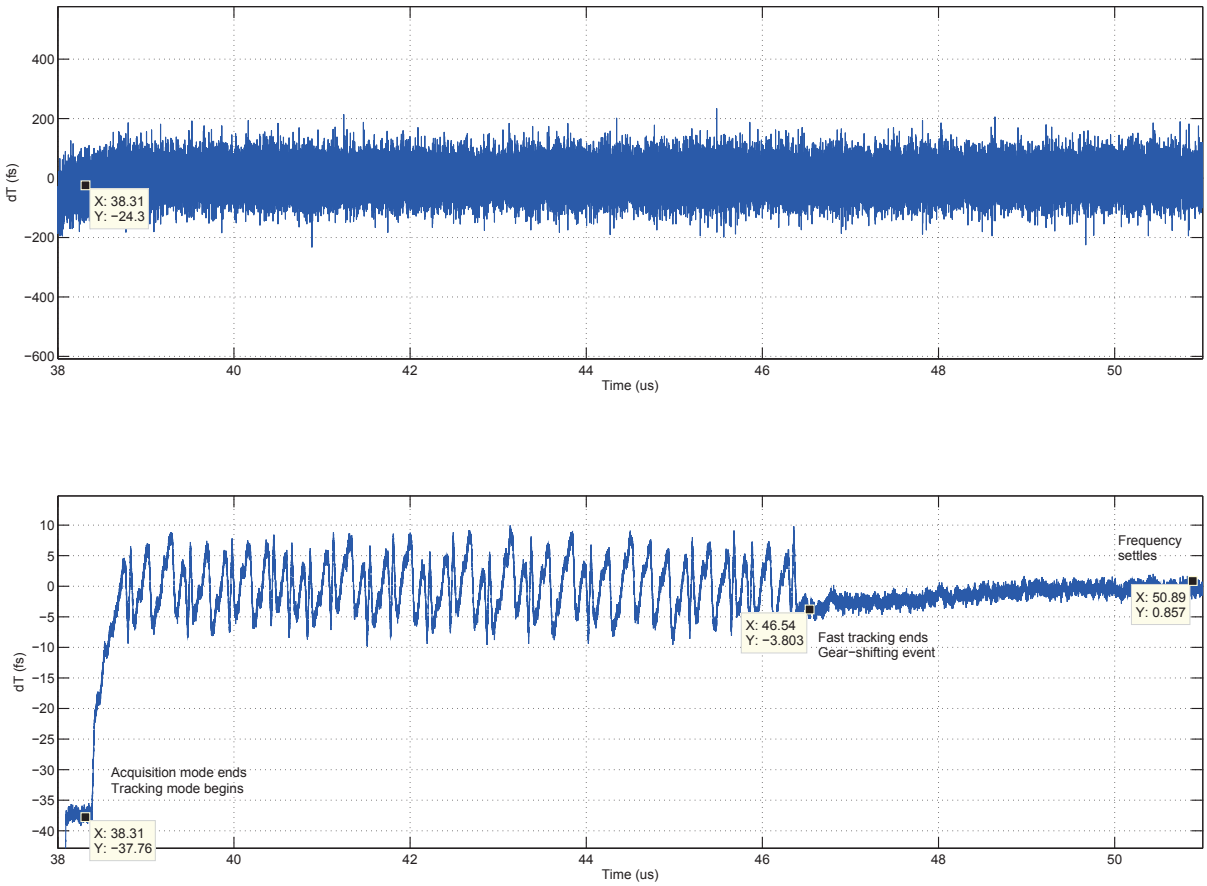

Figure E-26: ADPLL settling transient in tracking mode($f_v = 3300$ MHz)

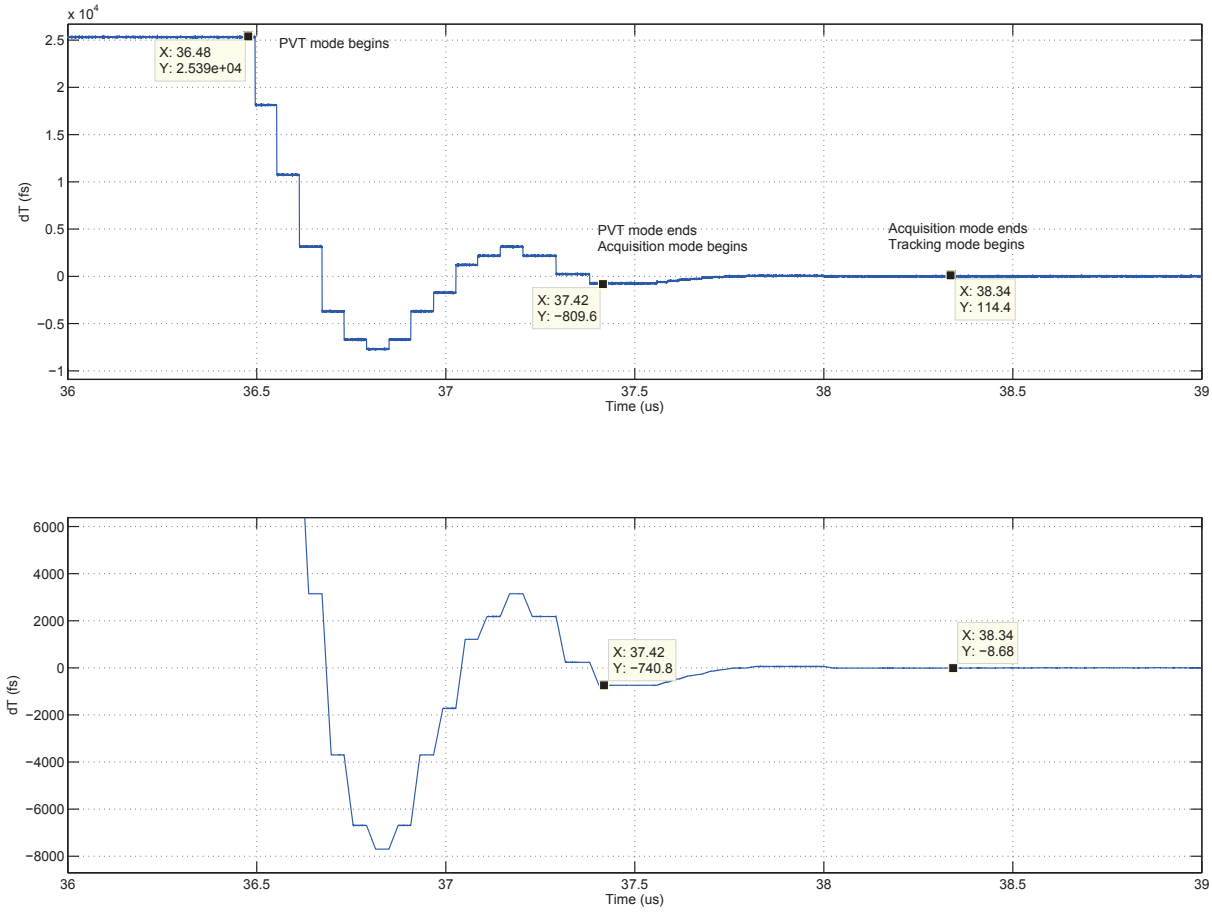

Figure E-27: ADPLL settling transient in PVT mode and acquisiton mode($f_v = 4050$ MHz)

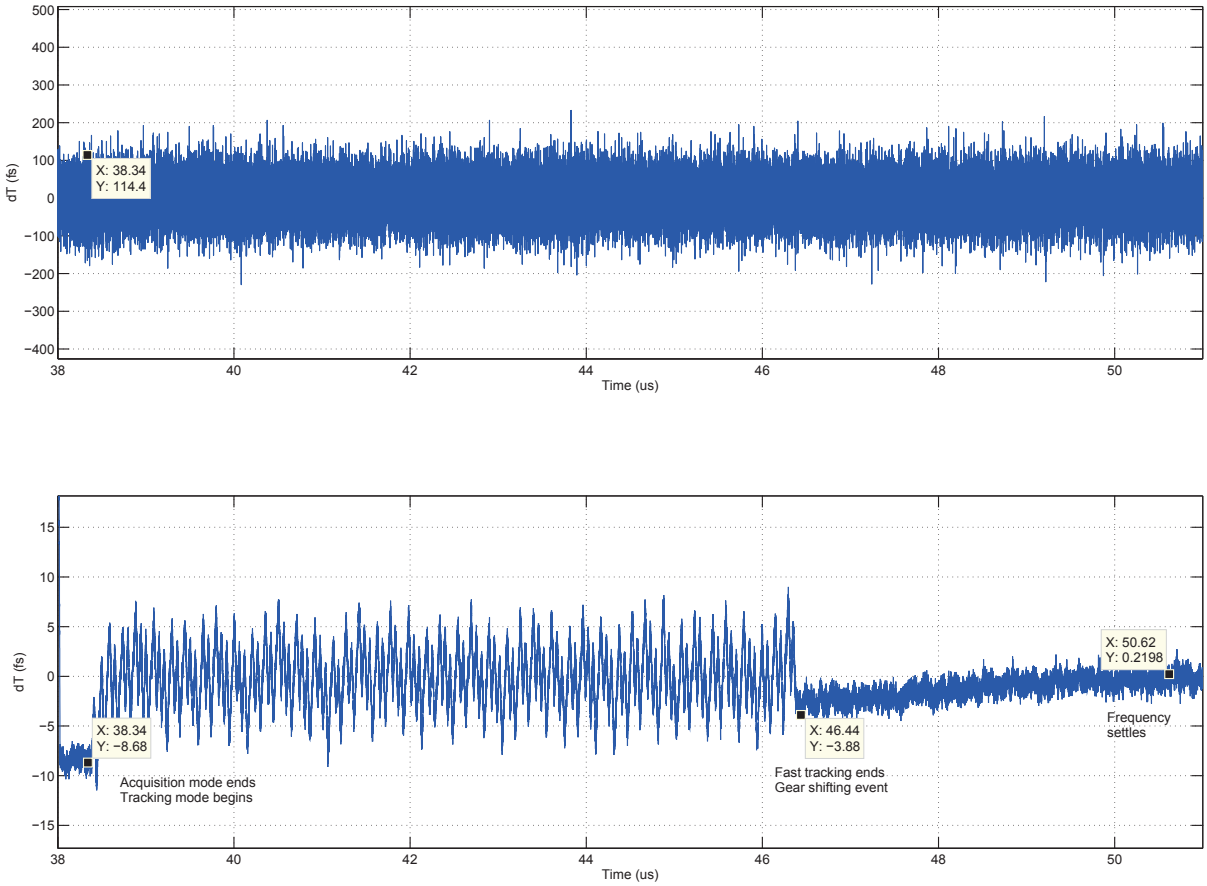

Figure E-28: ADPLL settling transient in tracking mode($f_v = 4050$ MHz)

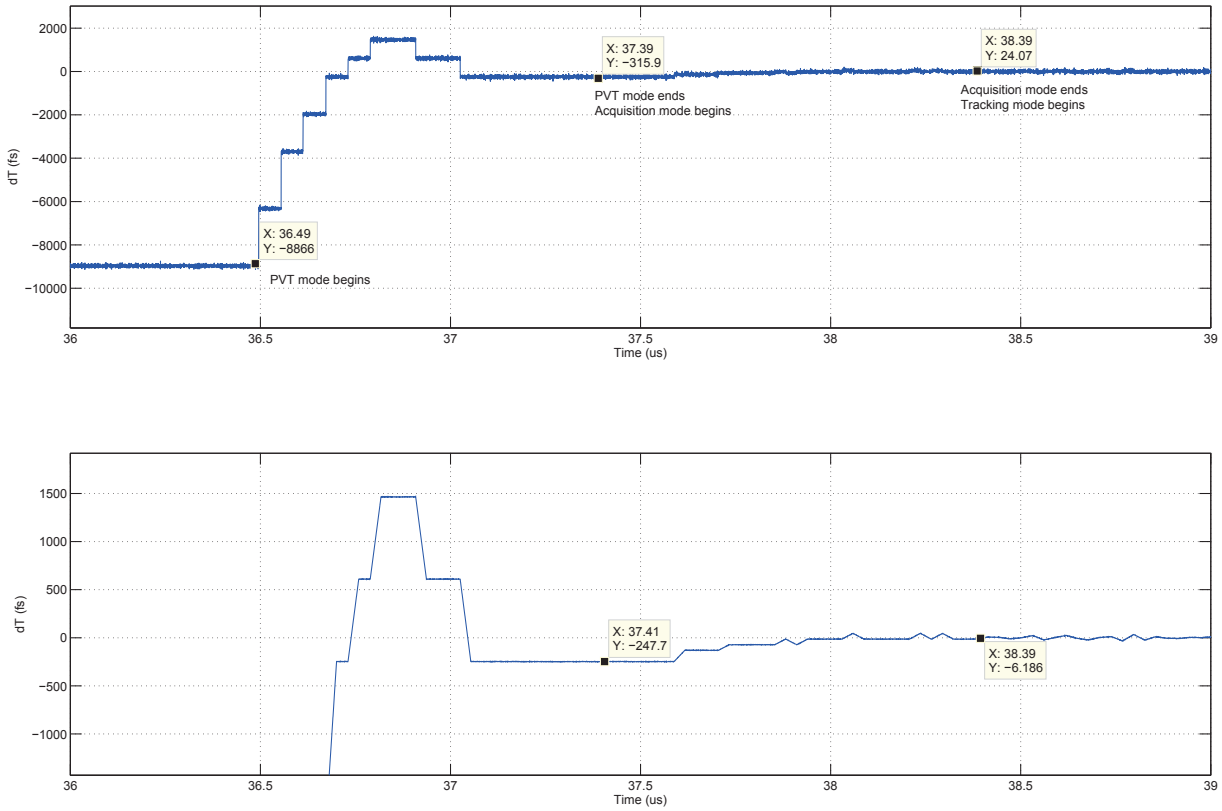

Figure E-29: ADPLL settling transient in PVT mode and acquisiton mode($f_v = 3556.214$ MHz)

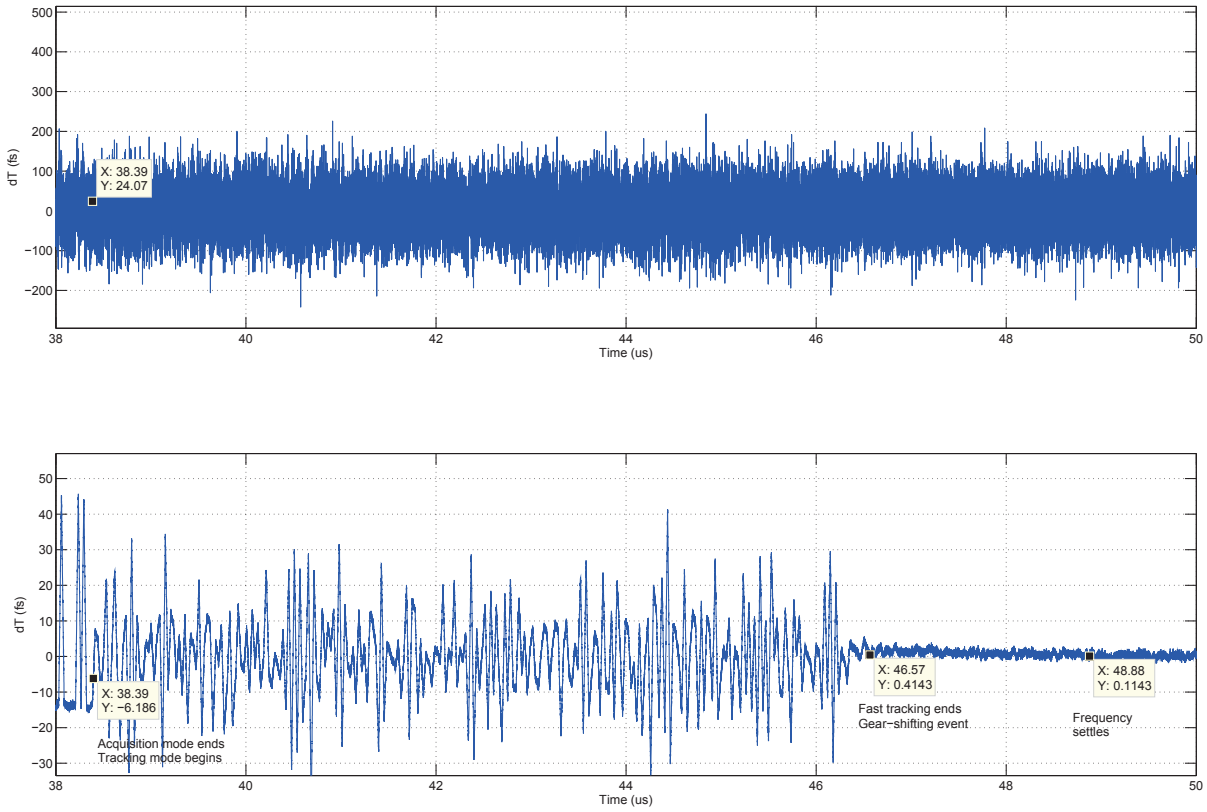

Figure E-30: ADPLL settling transient in tracking mode($f_v = 3556.214$ MHz)

Appendix F

DPA Layout

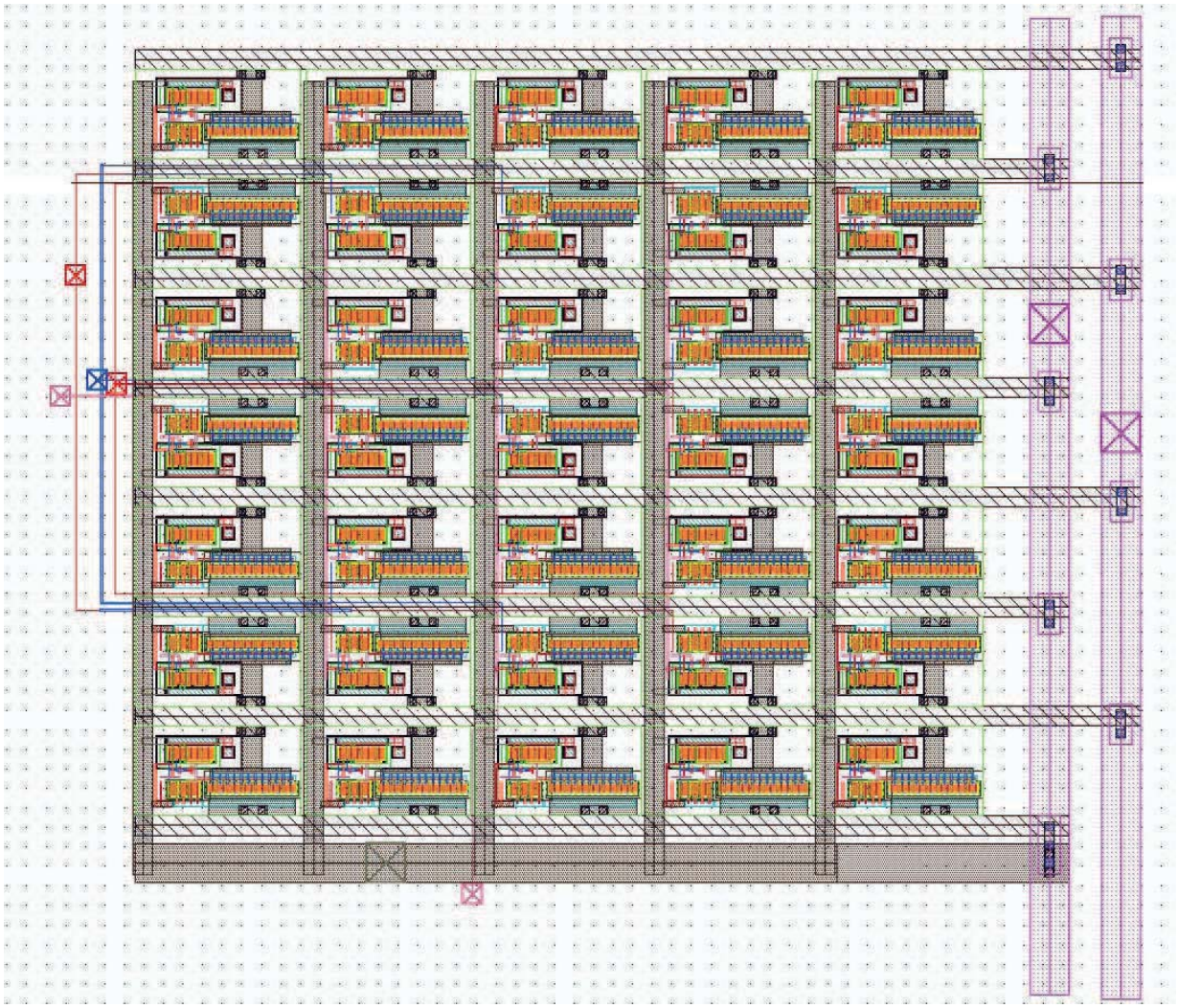

Figure F-1: Layout of the module catip_adpll_dpaswitcharray

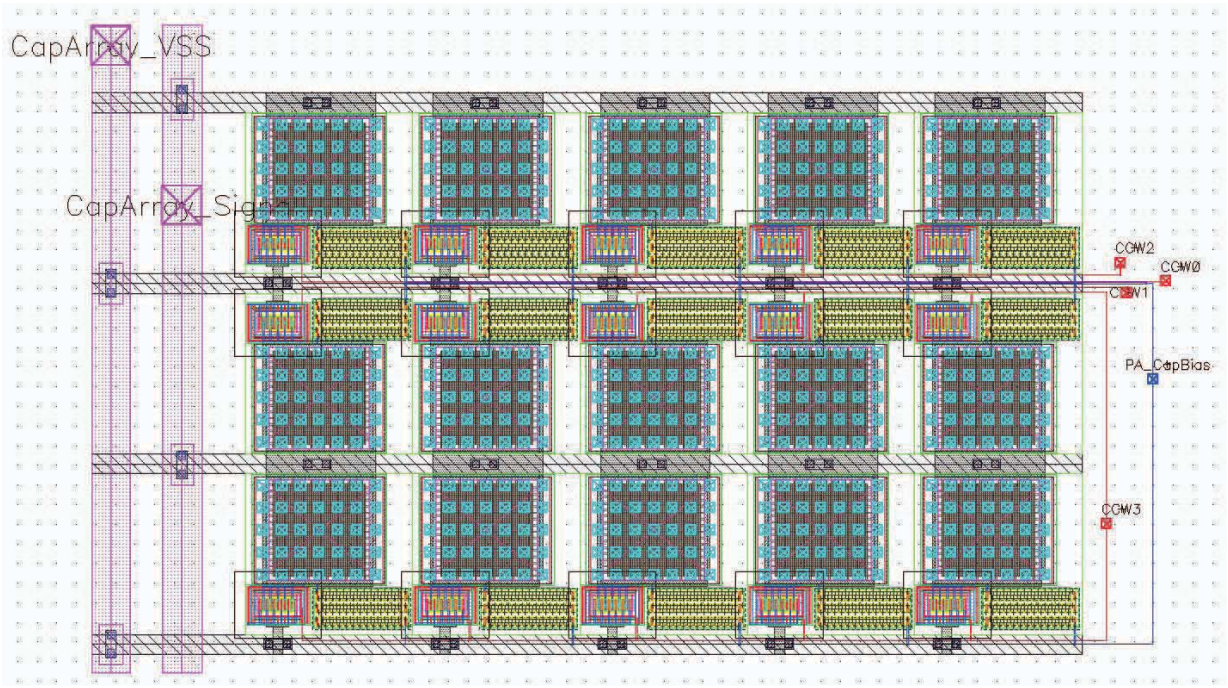

Figure F-2: Layout of the module catip_adpll_dpacaparray

Bibliography

[1] R. Staszewski, C. Hung, K. Maggio, J. Wallberg, D. Leipold, and P. Balsara, "All-digital phase-domain TX frequency synthesizer for Bluetooth radios in 0.13μm CMOS," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*. IEEE, 2004, pp. 272–527.

[2] P. Colantonio, F. Giannini, R. Giofrè, M. Medina, D. Schreurs, and B. Nauwelaers, "High frequency Class E design methodologies," in *Gallium Arsenide and Other Semiconductor Application Symposium, 2005. EGAAS 2005. European*. IEEE, 2005, pp. 329–332.

[3] R. Staszewski, J. Wallberg, S. Rezeq, C. Hung, O. Eliezer, S. Vemulapalli, C. Fernando, K. Maggio, R. Staszewski, N. Barton *et al.*, "All-digital PLL and GSM/EDGE transmitter in 90nm CMOS," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*. IEEE, 2005, pp. 316–600.

[4] T. Lee, *The design of CMOS radio-frequency integrated circuits*. Cambridge Univ Pr, 2004.

[5] F. Gardner, *Phaselock techniques*. Wiley-Blackwell, 2005.

[6] H. Darabi, H. Jensen, and A. Zolfaghari, "Analysis and Design of Small-Signal Polar Transmitters for Cellular Applications," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 6, pp. 1237–1249, 2011.

[7] S. Drago, D. Leenaerts, B. Nauta, F. Sebastiano, K. Makinwa, and L. Breems, "A 200 μA Duty-Cycled PLL for Wireless Sensor Nodes in 65 nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 7, pp. 1305–1315, 2010.

[8] R. Staszewski, I. Bashir, and O. Eliezer, "RF built-in self test of a wireless transmitter," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 2, pp. 186–190, 2007.

[9] Y. Park and D. Wentzloff, "An All-Digital 12 pJ/Pulse IR-UWB Transmitter Synthesized From a Standard Cell Library," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 5, pp. 1147–1157, 2011.

[10] J. Borremans, K. Vengattaramane, V. Giannini, B. Debaillie, W. Van Thillo, and J. Craninckx, "A 86 MHz–12 GHz Digital-Intensive PLL for Software-Defined Radios, Using a 6 fJ/Step TDC in 40 nm Digital CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 10, pp. 2116–2129, 2010.

[11] M. Zanuso, S. Levantino, C. Samori, and A. Lacaita, "A Wideband 3.6 GHz Digital $\Delta\Sigma$ Fractional-N PLL With Phase Interpolation Divider and Digital Spur Cancellation," *IEEE journal of solid-state circuits*, vol. 46, no. 3, pp. 627–638, 2011.

[12] T. Tokairin, M. Okada, M. Kitsunezuka, T. Maeda, and M. Fukaishi, "A 2.1-to-2.8-GHz Low-Phase-Noise All-Digital Frequency Synthesizer With a Time-Windowed Time-to-Digital Converter," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 12, pp. 2582–2590, 2010.

[13] E. Temporiti, C. Weltin-Wu, D. Baldi, M. Cusmai, and F. Svelto, "A 3.5 GHz Wideband ADPLL With Fractional Spur Suppression Through TDC Dithering and Feedforward Compensation," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 12, pp. 2723–2736, 2010.

[14] R. Staszewski, P. Balsara, and I. ebrary, *All-digital frequency synthesizer in deep-submicron CMOS*. Wiley Online Library, 2006.

[15] P. Effendrik, "Time-to-Digital Converter (TDC) for WiMAX ADPLL in State-of-The-Art 40-nm CMOS," Master's thesis, Delft University of Technology, 2011.

[16] M. Lee, M. Heidari, and A. Abidi, "A low-noise wideband digital phase-locked loop based on a coarse–fine time-to-digital converter with subpicosecond resolution," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 10, pp. 2808–2816, 2009.

[17] L. Xu, S. Lindfors, K. Stadius, and J. Ryynanen, "A 2.4-GHz low-power all-digital phase-locked loop," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 8, pp. 1513–1521, 2010.

[18] K. Kundert, "Predicting the phase noise and jitter of PLL-based frequency synthesizers," pp. 46–69, 2003.

[19] M. Perrott, "CppSim System Simulator."

[20] V.-A. committee, "SystemVerilogAMS."

[21] D. Leeson, "A simple model of feedback oscillator noise spectrum," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 329–330, 1966.

[22] E. Temporiti, C. Weltin-Wu, D. Baldi, R. Tonietto, and F. Svelto, "A 3 GHz fractional all-digital PLL with a 1.8 MHz bandwidth implementing spur reduction techniques," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 3, pp. 824–834, 2009.

[23] R. Staszewski, K. Waheed, S. Vemulapalli, P. Vallur, M. Entezari, and O. Eliezer, "Elimination of spurious noise due to time-to-digital converter," in *Circuits and Systems Workshop,(DCAS), 2009 IEEE Dallas*. IEEE, 2009, pp. 1–4.

[24] R. Staszewski, S. Vemulapalli, and K. Waheed, "An all-digital offset PLL architecture," in *Radio Frequency Integrated Circuits Symposium (RFIC), 2010 IEEE*. IEEE, 2010, pp. 17–20.

[25] K. Waheed, M. Sheba, R. Staszewski, F. Dulger, and S. Vamvakos, "Spurious free time-to-digital conversion in an ADPLL using short dithering sequences," in *Custom Integrated Circuits Conference (CICC), 2010 IEEE*. IEEE, 2010, pp. 1–4.

[26] R. Staszewski, K. Waheed, S. Vemulapalli, F. Dulger, J. Wallberg, C. Hung, and O. Eliezer, "Spur-free all-digital PLL in 65nm for mobile phones," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*. IEEE, 2011, pp. 52–54.

[27] *WiMAX Forum Mobile Radio Specification*, WiMAX FORUM, 2011.

[28] *SPI Block Guide V03.06*, Motorola, Inc., 2003.

[29] Wikipedia, "Serial Peripheral Interface Bus."

[30] A. Ba, "Ultra-low-power Digitally-controlled Oscillator for Event-driven Transmitter," Master's thesis, Delft University of Technology, 2011.

[31] P. Effendrik, W. Jiang, M. van de Gevel, F. Verwaal, and R. Staszewski, "Time-to-digital converter (TDC) for WiMAX ADPLL in 40-nm CMOS," in *Proc. of 20th European Conference on Circuit Theory and Design (ECCTD'11)*. IEEE, 2011, pp. 370–373.

[32] N. Sokal and A. Sokal, "Class E - A new class of high-efficiency tuned single-ended switching power amplifiers," *Solid-State Circuits, IEEE Journal of*, vol. 10, no. 3, pp. 168–176, 1975.

[33] F. Raab, "Idealized operation of the class E tuned power amplifier," *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 725–735, 1977.

[34] R. Staszewski, K. Muhammad, D. Leipold, C. Hung, Y. Ho, J. Wallberg, C. Fernando, K. Maggio, R. Staszewski, T. Jung *et al.*, "All-digital TX frequency synthesizer and discrete-time receiver for Bluetooth radio in 130-nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 12, pp. 2278–2291, 2004.

[35] P. Cruise, C. Hung, R. Staszewski, O. Eliezer, S. Rezeq, K. Maggio, and D. Leipold, "A digital-to-RF-amplitude converter for GSM/GPRS/EDGE in 90-nm digital CMOS," in *Radio Frequency integrated Circuits (RFIC) Symposium, 2005. Digest of Papers. 2005 IEEE*. IEEE, 2005, pp. 21–24.

[36] R. Staszewski, J. Wallberg, C. Hung, G. Feygin, M. Entezari, and D. Leipold, "LMS-based calibration of an RF digitally controlled oscillator for mobile phones," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 3, pp. 225–229, 2006.

[37] D. Chowdhury, L. Ye, E. Alon, and A. Niknejad, "An Efficient Mixed-Signal 2.4-GHz Polar Power Amplifier in 65-nm CMOS Technology," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 8, pp. 1796–1809, 2011.