Title

PLL Design for FMCW Radar Systems

Author

Liu, Benyuanyi

Publication Date

2024

Peer reviewed|Thesis/dissertation

PLL Design for FMCW Radar Systems

By

Benyuanyi Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ali M. Niknejad, Chair
Professor Liwei Lin
Professor Sophia Shao

Spring 2024

PLL Design for FMCW Radar Systems

Abstract

PLL Design for FMCW Radar Systems

By

Benyuanyi Liu

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ali M. Niknejad, Chair

In this thesis, the PLL design for FMCW radar systems is illustrated. The FMCW radar imposes requirements on the PLL-based chirp generator in several aspects. Both the phase noise and the bandwidth of the PLL influence the measurement accuracy and resolution of the radar. The phase noise should be minimized, given a certain power consumption, and the bandwidth needs to be matched with the FMCW chirp slope to achieve better linearity.

For system-level design, unlike PLLs for other applications, the output of the chirp generator PLL is always changing. In many applications, such as precision measurement, the output never truly settles at each step. This necessitates careful modeling of the loop dynamics. In this thesis, conventional PLL phase noise and settling time models are presented and adapted for the chirp generator. However, these models are insufficient for optimizing the design. Therefore, a more accurate time-domain model for calculating the chirp is proposed. This model aids in designing the PLL bandwidth, calculating the acceptable chirp slope for a given PLL, and computing the dynamic phase noise. To the best of our knowledge, this is the first relatively accurate model for the entire chirp generation process.

For the design of circuit blocks, the frequency division modules, which include a dual-modulus divider and a Delta-Sigma Modulator, are presented. For the high-frequency circuit, an analytical model and a corresponding design methodology are proposed. The PLL has been taped-out, and measurements will be conducted to confirm the performance and design methodology.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

During my time at UC Berkeley as a graduate student, I have received guidance, help, and support from many people. In my PhD journey, I have been working in several fields, and it would have been impossible to learn quickly without their help. I'd like to express my thanks to them at the beginning of my thesis.

First of all, I would like to thank my advisor, Prof. Ali M. Niknejad, for his patience, guidance, and support over the years. There are ups and downs in the PhD journey, but he always encourages me to move forward. Discussions with him are always enlightening, from which I gain insights into circuits and the key problems that need solving. The same goes for his classes. Explanations, examples, and analogies help me understand the physics and circuitry intuitively. I feel very lucky to have spent my time in his group.

I wish to thank Prof. Sophia Shao, Prof. Liwei Lin, and Dr. Osama Shana'a, for being on my qualifying exam committee and dissertation committee. Their feedback is very valuable for my research.

I would also like to thank all BWRC staff, Ajith Amerasekera, Anita Flynn, Brian Richards, Jeff Anderson, James Dunn, Candy Corpus, and Mikaela Cavizo, for their help with my research and for making my life at BWRC much easier. In particular, I want to thank Candy Corpus and Mikaela Cavizo for ensuring all kinds of events run smoothly. Brian Richards and James Dunn have helped me a lot with CAD tool settings. Anita Flynn has made a tremendous contribution by keeping the lab clean and well-organized.

BWRC is a place where everyone can learn from everyone. I wish to thank BWRC students for their help, discussion, cooperation, and feedback. I thank Bo Zhao, Lorenzo Iotti, and Ananda Sankar Chakraborty for their help on my research. They were postdocs when I worked with them in different research projects and each of them taught me a lot on CAD tool usage, circuit and device modeling, design methodologies, and measurements. In particular, I would like to thank Ananda. He is both a research partner and a true friend of mine. The Zoom chats with him really relieve the stress and anxiety at the beginning of the COVID-19 pandemic. I also thank Nai-Chung Kuo, Greg Lacaille, Nima Baniasadi, Emily Navisky, Yi-an Li, Luya Zhang, Ali Ameri, Yikuan Chen, Rohit Braganza, Pengpeng Lu, Ali Moin, Zhongkai Wang, John Wright, Matthew Anderson, Jongbeom Baek, Hesham Beshary, Ethan Chou, Rami Hijab, Meng Wei, and Kunmo Kim, for their help on my research.

Last but not least, I would like to thank my parents and my wife for their love and support. They are completely behind me in my endeavors, and I couldn't have completed my PhD journey without them. I also want to thank my best friend, Yun Zhong, for the understanding, listening, help, support, funny chats, and brainstorming he provides. We have known each other for almost 20 years, and I couldn't feel luckier to have such a friend.

Chapter 1

Introduction

1.1 Introduction to FMCW Radar

In 1924, the first bistatic FMCW radar experiment was conducted by Appleton and Barnett to measure the height of the ionosphere [2], followed by the Daventry experiment in 1935 to detect aircraft [3]. During World War II, transmit-receive switch was invented and monostatic FMCW radars were built for bomb aiming and surveillance [4]. In mid 1970's, a major step forward was made as digital signal processing became available to perform the signal processing to extract information from the received signals [5]. Nowadays, FMCW radars have been widely used for civil applications (such as automotive applications [6] and gesture recognition [7]), industry measurements (such as altimeter [8], coating characterization [9], and level gaging [10]), and military applications (such as small drone detection [11]).

Today, state-of-the-art FMCW radar systems are implemented with CMOS and SiGe technologies, and both have their pros and cons. f_T and f_{max} of advanced SiGe process are higher than those of CMOS process that has comparable cost [12]. And higher f_T/f_{max} makes it more feasible to achieve both lower power and higher performance simultaneously. Because to achieve the same performance, high-f_T/f_{max} devices consume less current. And with the same current consumption, high-f_T/f_{max} devices could achieve better performance [13]. Another difference between CMOS devices and SiGe devices is maximum allowable bias voltage. The collector-base breakdown voltage of SiGe devices keeps above 4V with increasing RF performance while maximum supply voltage of CMOS devices rapidly decreases with more advanced nodes [14]. Low supply voltage of CMOS devices is excellent for reducing power consumption for digital circuits, but it also makes it more and more difficult to transmit RF power out at the cost of higher current level. However, from the perspective of radar system, CMOS process still has advantage of higher integration level. Shown in Fig. 1.1 is an example for automotive radar [15]. Today's high-end vehicles feature a multichip SiGe radar system. But as the number of sensors increases to at least ten, the big and bulky multichip SiGe

Table 1.1: State-of-the-art FMCW radar systems in F-band (90-140GHz), D-band (110-170GHz), and G-band (110-300GHz).

	MIT, JSSC 2021 [17]	Toronto, TMTT 2021 [18]	Ulm, TMTT 2019 [19]	Ulm JSSC 2021 [20]	IMEC, ISSCC 2019 [21]	Wuppertal, Trans. THz 2016 [22]	MIT, ISSCC 2022 [23]
Technology	65nm CMOS	22nm FDSOI	SiGe	SiGe	28nm CMOS	SiGe	65nm CMOS
Architecture	1TX, 1RX Frequency comb	2TX, 2RX Stepped-frequency chirp	1TX, 1RX MMIC-based	1TX, 1RX Leakage suppression	1TX, 1RX Leakage suppression	1TX, 1RX Circular polarization	1TX, 1RX Low-loss duplexing
Center frequency (GHz)	220	151	160	169	145	240	140
Chirp bandwidth (GHz)	100	9	16	20	13	60	14
Resolution (mm)	1.5	17	10	21	11	2.57	11
RX min. NF (dB)	22.2	7.5-10	25	15.5	8	22.5	12.9
Phase noise @1MHz (dBc/Hz)	N/A	-107	-87	-80	-80	N/A	N/A
Chip size (mm^2)	5	2.75	N/A	5.4	6.5	3.2	3.1
Power (mW)	840	1130 (chirp synthesizer not included)	N/A	860	500	1800	405 (chirp synthesizer not included)

radar system solution becomes too spacce-consuming [16], making CMOS technology more appealing, which can integrate into one chip radar front ends, microcontrollers (MCUs), and digital signal processing (DSP) circuits. As some wireless sensing applications are putting an increasing demand on bandwidth. The industry is addressing this challenge in many ways, one of which is by moving to higher frequencies. SiGe is applied to build the transceivers to address these applications first, due to its performance advantages. Over time, bulk CMOS would catch up and address the applications with circuits that are more integratable with other parts of the radar system. Some state-of-the-art FMCW radars systems are summrized in Table. 1.1.

The traditional architecture of FMCW radar is very simple, as shown in Fig. 1.2. The FMCW signal is sent to both transmitter antenna and receiver mixer to get beat frequencies of objects. The frequency of the FMCW signal linearly ramps, so the distance is translated into frequency. Such an architecture leads to strong noise correlation, thus noise cancellation, for short-distance objects [24]. As distance increases, the correlation becomes weaker and noise cancellation is less.

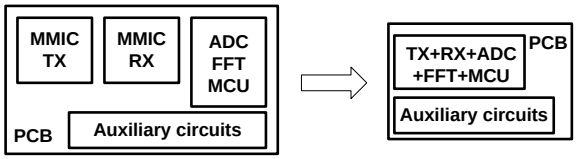


Figure 1.1: Single chip integration enabled by CMOS.

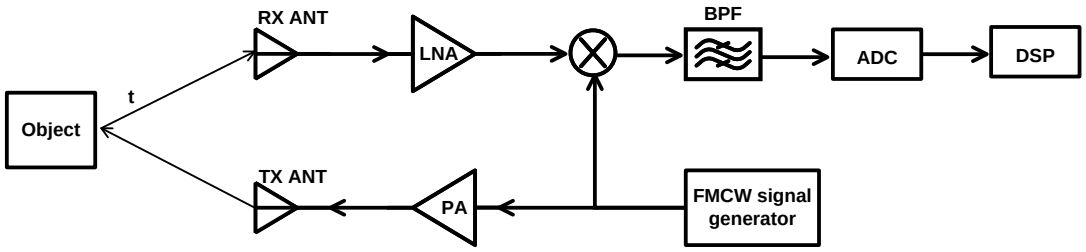


Figure 1.2: Block diagram of a general FMCW radar system.

Based on the basic architecture, researchers proposed many variants to address issues for specific applications. [25] and [26] adopt adaptive algorithm to cancel the leakage and interference. Synthetic bandwidth technique is applied in [17] to achieve ultra-wide chirp bandwidth of 100GHz. [27] and [28] implement on-chip calibration to reduce gain mismatch and phase mismatch of complex baseband.

1.2 Operating Principle of FMCW Radar

A complete FMCW radar system can measure range, velocity, and angle of multiple objects. For each measurement, the maximum measurement range and resolution are determined by system parameters. The FMCW radar measurements are based on transmitting frequency chirp, characterized by start frequency f_c, bandwidth B, and chirp duration T_c as shown in Fig. 1.3. And corresponding chirp slope is

$$S = \frac{B}{T_c} \tag{1.1}$$

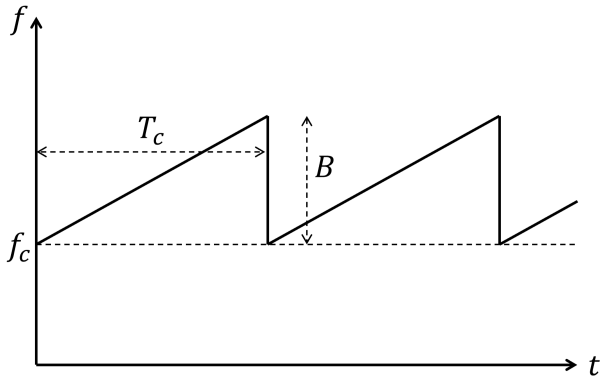


Figure 1.3: Frequency chirp for FMCW radar.

Range Measurement

When a chirp is transmitted and then the electromagnetic wave (EM wave) is reflected back to the radar, the system would mix the transmitted signal and the received one, as shown in Fig. 1.4, getting a beat frequency f_B for each object. The one-way travelling time for EM wave from radar to the object is

$$t_0 = \frac{d}{c} \tag{1.2}$$

where d is the distance between the radar and the object, c the speed of light. And the round-trip travelling is $2t_0$, so the beat frequency f_B should be

$$f_B = 2t_0 \cdot S = S \cdot \frac{2d}{c} \tag{1.3}$$

In this way, the distance d can be calculated by measuring beat frequency f_B.

The maximum range is determined by FMCW chirp bandwidth B and IF sampling frequency F_s.

$$f_{B,max} = S \cdot \frac{2d_{max}}{c} < B \tag{1.4}$$

$$f_{B,max} = S \cdot \frac{2d_{max}}{c} < \frac{F_s}{2} \tag{1.5}$$

For multiple objects, according to Discrete Fourier Transform (DFT) theory, an observation window of T_c can separate frequency components that are separated by more than

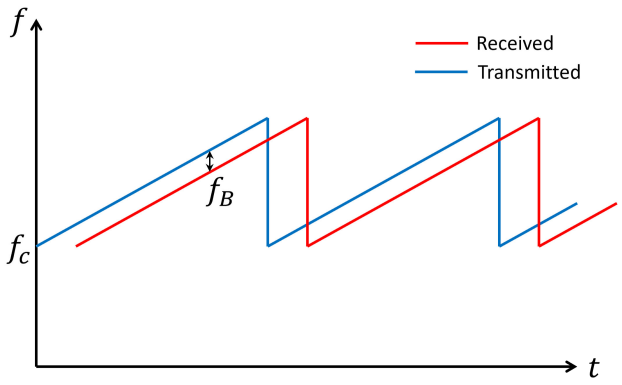


Figure 1.4: Range measurement of a single object.

$\frac{1}{T_c}$Hz, which means the frequency resolution is $\frac{1}{T_c}$. As a result, the range resolution is

$$\Delta d = \frac{c}{2B} \tag{1.6}$$

Velocity Measurement

In terms of radial velocity measurement of moving objects, the beat frequency consists additionally of Doppler frequency f_D

$$f_B = S \cdot \frac{2d}{c} + f_D = S \cdot \frac{2d}{c} + \frac{2v_r}{\lambda_c} \tag{1.7}$$

where λ_c is the corresponding wavelength of start frequency f_c. The beat frequency f_B from a single chirp contains unknowns of d and v_r in an unsolvable way. The principle of velocity measurement is that phase of the IF signal is very sensitive to small changes in object range, and that Doppler frequency's impact on beat frequency is negligible. As shown in Fig. 1.5, two chirps separated by T_c are transmitted, and received signal corresponding to each chirp will have the same beat frequency but different phases. The phase difference is $\Delta\varphi$:

$$\Delta\varphi = 2\pi f_D T_c = 2\pi \cdot \frac{\frac{2v_r T_c}{c}}{\frac{1}{f_c}} = \frac{4\pi v_r T_c}{\lambda_c} \tag{1.8}$$

$$v_r = \frac{\lambda_c \Delta\varphi}{4\pi T_c} \tag{1.9}$$

And the phase difference $\Delta\varphi$ is positive when the object moves towards radar, and negative when moving away. As a result, to assure unambiguous velocity measurement,

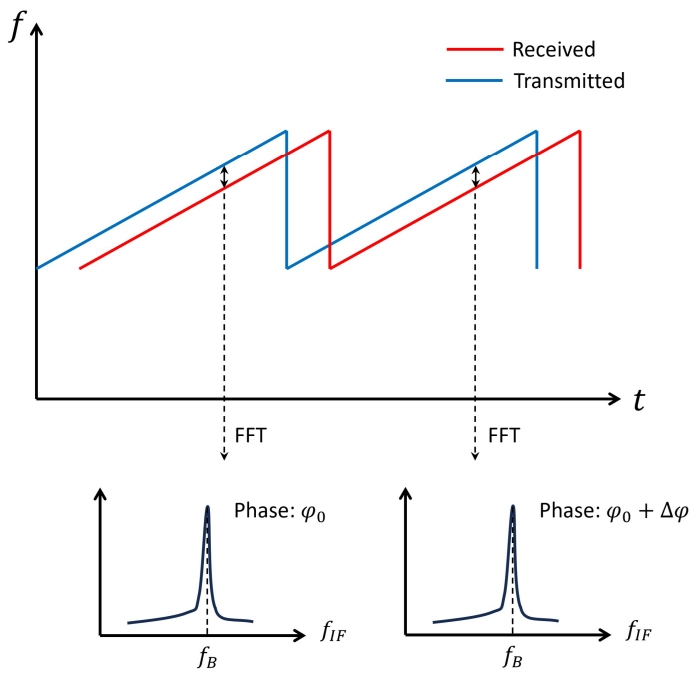


Figure 1.5: Radial velocity measurement of a single object.

$$|\Delta\varphi| < \pi \tag{1.10}$$

should be fulfilled, from which the maximum measurable velocity could be expressed as

$$|v_r|_{max} = \frac{\lambda_c}{4T_c} \tag{1.11}$$

For multiple objects, in extreme cases where two objects have equal distance from the radar but different velocities as shown in Fig. 1.6, they need to be distinguished by velocity, which can be done with Doppler FFT.

Shown in Fig. 1.7, is range-Doppler processing in FMCW radar system. N chirps are transmitted, and after mixing the transmitted signal with received one, range FFT are conducted to get the spectrum of IF signal. Object 1 and object 2 are not distinguishable on this spectrum because their beat frequencies are equal (equal distance). Phase of IF signal contains velocity information. And for each IF frequency, there are N data points with different phases. Doppler FFT is conducted to such set of phase data of each IF frequency, giving us a 2-dimensional range-Doppler map. In the case described in above, Fig. 1.7 shows the Doppler FFT result of frequency f_B, on which object 1 and object 2 could be

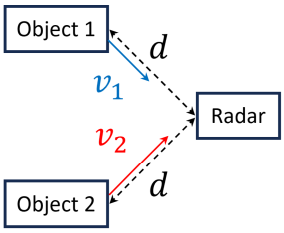


Figure 1.6: Two objects with equal distance to the radar, but different radial velocities.

distinguished for their different radial velocities. Fig. 1.7 only shows the Doppler FFT result at the IF frequency of f_B. To get a complete range-Doppler map, Doppler FFT is conducted for each IF frequency point.

As for resolution of velocity measurement, as known, a sequency of length N can separate frequencies that are separated by more than $\frac{2\pi}{N}\ rad/s$ after FFT, which is

$$2\pi\Delta f_D T_c = \frac{2\pi}{N} \tag{1.12}$$

and that gives the velocity resolution

$$\Delta v_r = \frac{\lambda_c}{2NT_c} = \frac{\lambda_c}{2T_f} \tag{1.13}$$

where T_f is the frame time and one frame consists of N chirps.

Angle Measurement

Angle measurement requires at least two antennas. Similar to velocity measurement, the angle measurement is based on the fact that the phase of received signal is sensitive to difference of EM wave travelling time caused by non-zero incident angle, as shown in Fig. 1.8.

For a single object, Fig. 1.9 shows the measurement principle with two antennas. A frame of chirp is transmitted via a TX antenna, and when the EM wave bounces back, a range-Doppler map could be drawn for each RX antenna. The peaks on these maps would be at the same location because these two antennas are observing the same object with same distance and same velocity. But the phase of the received signal would be different due to different travelling time caused by $sin\theta \cdot d_{ant}$ shown in Fig. 1.9. And corresponding phase difference is

$$\Delta\varphi = 2\pi\frac{sin\theta \cdot d_{ant}}{\lambda_c} \tag{1.14}$$

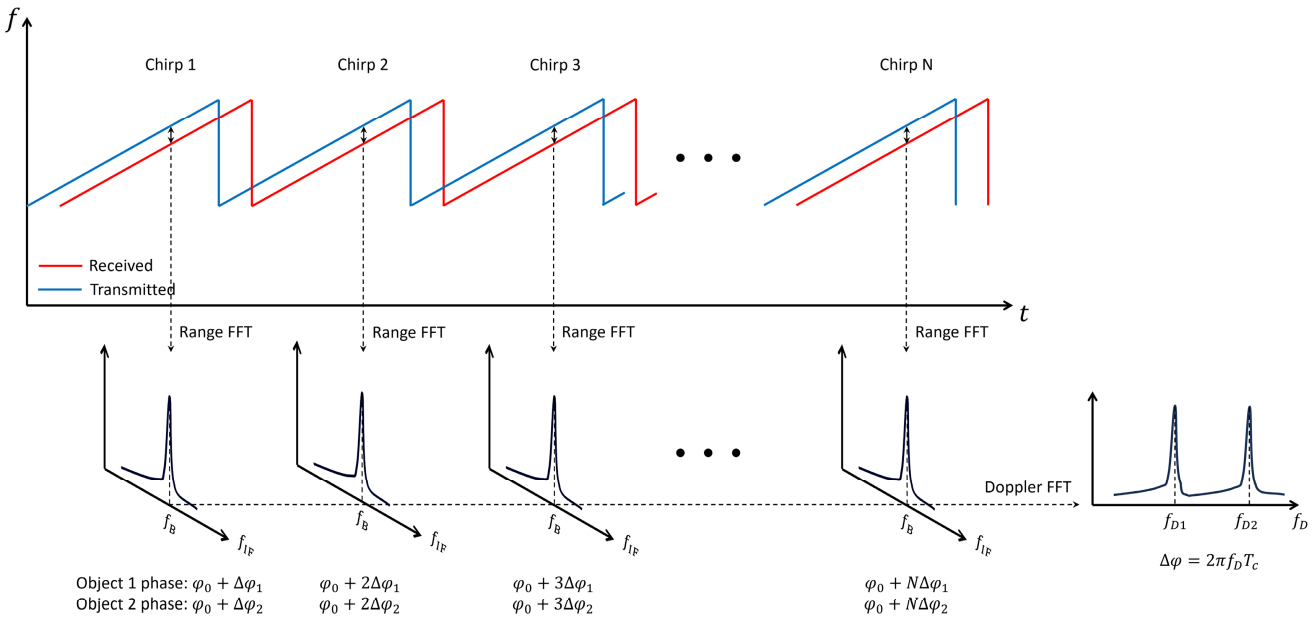


Figure 1.7: Range-Doppler signal processing in FMCW radar system.

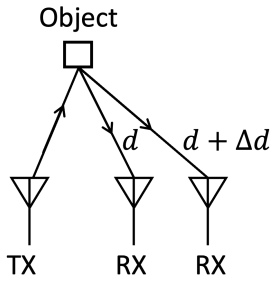


Figure 1.8: Angle measurement based on two RX antennas.

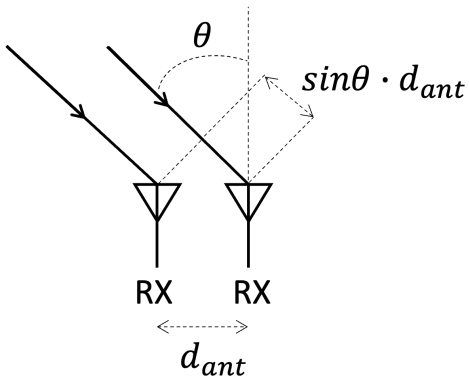


Figure 1.9: Angle measurement of a single object.

So the incident angle is

$$\theta = sin^{-1}(\frac{\Delta\varphi \cdot \lambda_c}{2\pi d_{ant}}) \tag{1.15}$$

And to assure unambiguous angle measurement,

$$|\Delta\varphi| < \pi \tag{1.16}$$

which gives

$$|\theta|_{max} = sin^{-1}(\frac{\lambda_c}{2d_{ant}}) \tag{1.17}$$

For multiple objects, K receiving antennas are needed to achieve certain angle resolution. Again, consider a extreme case shown in Fig. 1.10 where two objects have equal distance from the radar and equal radial velocity towards the radar. Thus, for each RX antenna, there will be only one peak on the range-Doppler map, as shown in Fig. 1.11. They can be distinguished from angle. In this case, angle FFT is applied to the phase sequence of the peak. And there would be two peaks in the resulting plot, one for object 1 and the other for 2.

For angle resolution, similarly to velocity resolution, the minimum distinguishable frequency separation is $\frac{2\pi}{K}$. And the "frequency" in angle FFT is the phase difference $\Delta\varphi$ of the received signals

$$\Delta\varphi = \frac{2\pi d_{ant}}{\lambda_c} \cdot [sin(\theta + \Delta\theta) - sin(\theta)] \tag{1.18}$$

$$\Delta\varphi \approx \frac{2\pi d_{ant}}{\lambda_c} cos(\theta)\Delta\theta \tag{1.19}$$

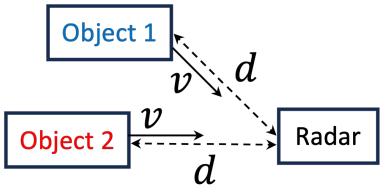
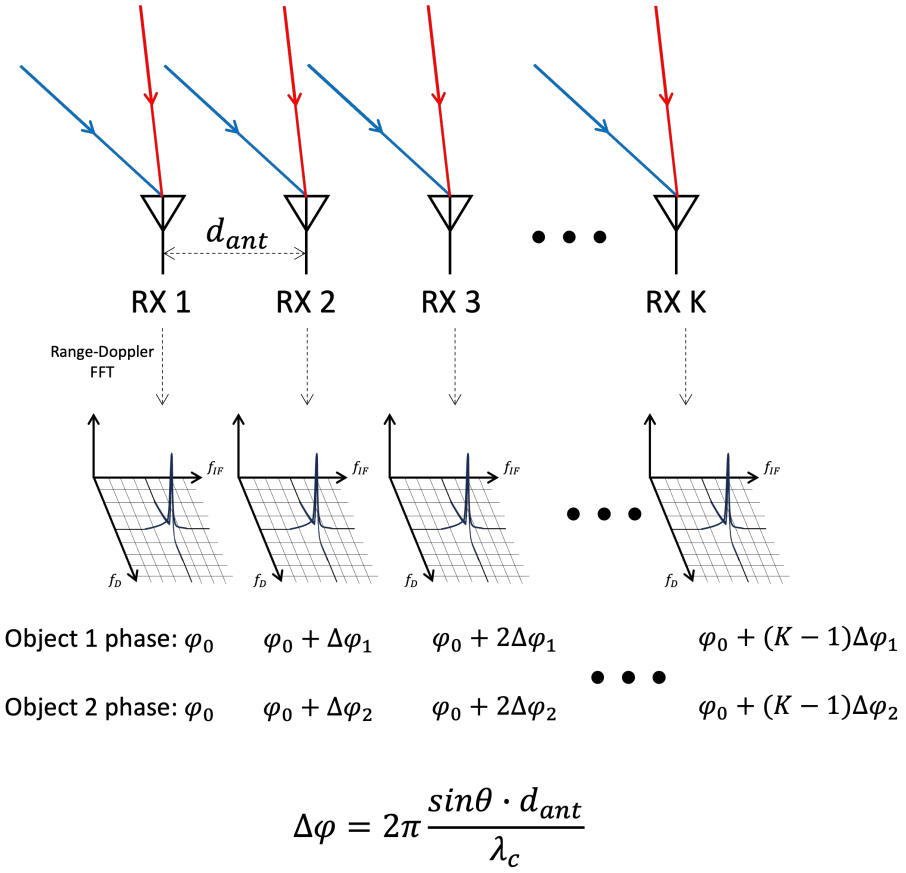


Figure 1.10: Two objects have equal distance from the data and equal velocity towards the radar.



$$\Delta\varphi = 2\pi \frac{sin\theta \cdot d_{ant}}{\lambda_c}$$

Figure 1.11: Angle FFT based on rangle-Doppler map.

Thus, the angle resolution is

$$\Delta\theta = \frac{\lambda_c}{K \cdot d_{ant} \cdot cos(\theta)} \tag{1.20}$$

Conventionally, the angle resolution is quoted assuming $\theta = 0$

$$\Delta\theta = \frac{\lambda_c}{K \cdot d_{ant}} \tag{1.21}$$

1.3 Frequency Synthesizers for FMCW Radar

Shown in the previous section are theoretical resolutions for range, velocity, and angle, assuming a perfect FMCW chirp. However, in real life, FMCW chirp has imperfections, among which nonlinearity and phase noise are the main factors that degrade resolution and contribute to measurement error.

Several types of frequency synthesizers (chirp generator) are applied to generate the chirp, and each of them has its pros and cons. Direct digital synthesizer (DDS) is the most straightforward way to generate a chirp [29]. It's a quick and easy solution for prototype testing, but it comes with high spur level and limited bandwidth.

For analog methods, the chirp generator could be implemented mainly in three ways, which are VCO-based frequency synthesizers [30], multiplier-based frequency synthesizers [31], and PLL-based frequency synthesizers. VCO-based frequency synthesizers suffer from nonlinearity issue, which requires compensation circuitry to correct the nonlinearity. Multiplier-based frequency synthesizers have low phase noise, but they suffer from nonlinearity and harmonic spurs issue, resulting in lower dynamic range. PLL-based frequency synthesizers have good linearity and low phase noise, which leads to good radar resolution with comparable hardware cost.

1.4 Design Needs of PLL-based Chirp Generator

There are several main design needs for PLLs in FMCW radar system, namely low phase noise, good linearity, fast chirp generation, and wide chirp bandwidth.

Phase noise of PLL has a great impact on FMCW radar performance [24]. In multi-target scenario, when objects are close to each other, larger in-band phase noise of PLL would lead to wider peak on IF spectrum for each object, making it harder to clearly distinguish objects, and thus lowering spatial resolution and the image quality. When objects are far from each other, larger out-of-band noise of PLL leads to higher noise tail from closer objects, which raises the noise floor for farther objects. That would make it more difficult to detect the farther objects, limiting the maximum detection range of FMCW radar.

In terms of frequency deviation and chirp linearity, phase noise contributes to the random frequency deviation, and other sources (such as digital circuit switching, PLL transient response, and nonlinear circuit blocks) contribute to the deterministic or even periodic frequency deviation. The resultant nonlinear chirp could be modeled as a sinusoidal nonlinearity term [32] or a square-law nonlinearity term [33] in instantaneous output frequency expression, both of which contribute to lower range resolution and larger range estimation error.

The basic waveform to measure range and velocity consists of a up-ramp and a down-ramp, duration of each of which is larger than 1ms [34]. That makes ranging frequency f_r and Doppler frequency f_d fall in kHz range, which is flicker noise range. As a result, the ranging resolution of FMCW radar is affected by flicker noise of the circuit. To solve the issue, designers adopt fast chirp to move f_r and f_d above flicker noise corner frequency. Duration of such chirp is around 100μs. That leads to some requirements of the PLL bandwidth [35], which would be discussed in detail in later chapters.

Another need is wide chirp bandwidth. As known, for FMCW radar, the ranging resolution is inversely proportional to chirp bandwidth [36]. To get range resolution of 10mm, chirp bandwidth needs to be as wide as 16GHz [19]. The high-resolution measurement is needed for industry quality inspection and measurement, such as accurate positioning of machines, measuring thickness of layer coating, and separating closely adjacent targets [37]. And wide bandwidth systems also offers possibility to operate in many different frequency bands [38].

As for the architecture, generally, PLL-based chirp generators can be categorized into two types. One is one-point-modulation (OPM) architecture, and the other is two-point-modulation (TPM) architecture [39].

For OPM architecture, the modulation signal is injected at only one point of the PLL. In [40], the modulation signal is imposed as part of VCO's control signal. Such PLL is simple and doesn't have additional noise source. Modulation can also be implemented by modulating reference frequency of a PLL. For instance, a chirp is generated with a direct digital frequency synthesizer (DDFS) and fed into the PLL as reference in [41]. DDFS can avoid the issue of limited bandwidth of LC-VCO, and a high-resolution DDFS can minimize the chirp nonlinearity, but power consumption of DDFS is high and it needs a large amount of memory for the lookup table and a high-resolution DAC. Similarly in [42], a fast and linear chirp as reference signal can facilitate the design of other components of PLL, but an external instrument is needed in the design. So such designs are only for prototyping or testing purposes, not for practical applications. Alternatively, the frequency modulation can be achieved by varying the feedback division ratio in fractional-N PLL [43], [44]. The generated chirp has good linearity if loop bandwidth is designed properly, and its power is significantly lower than DDFS architecture, but additional noise from Delta-Sigma Modulator should be

Table 1.2: State-of-the-art PLLs for FMCW radar systems.

	Xidian, VLSI 2022 [47]	ECNU, TCAS 2022 [48]	NTU, ISCAS 2022 [49]	ADI, ISSCC 2022 [50]	IMEC, ISSCC 2020 [51]	Toronto, TMTT 2021 [18]	IBM, JSSC 2018 [52]	DENSO, RFIC 2020 [53]
Technology	65nm CMOS	55nm CMOS	40nm CMOS	28nm CMOS	28nm CMOS	22nm FDSOI	130nm SiGe	40nm CMOS
Architecture	DS-PLL	Fractional-N	Fractional-N	ADPLL	SSPLL	Integer-N	Dual-loop	Fractional-N
Reference (MHz)	277 to 562.5	40	50	80 to 200	80	8400 to 1100	125	50
Center frequency (GHz)	8.55 to 17.11	21.24 to 24.64	4.8 to 5.5	8.8 to 12	8.3 to 11.7	160	79	76 to 77
Chirp bandwidth (GHz)	0.5	1.25	0.7	0.65	1.21	8.5	8	0.3
Chirp rate (MHz/µs)	33.33	1.818	1.22	65	94.5	5300	100	N/A
Phase noise @1MHz (dBc/Hz)	-120	-98.54	-110	-121	-109.1	-113	-97	-91
PLL power (mW)	22.1	92.1	3.3	187	11.7	300	590	N/A

considered in the design. For all the OPM architectures, PLL loop bandwidth should be designed considering some trade-offs. The loop bandwidth of PLL should be much less than modulation frequency to allow VCO frequency variations and to suppress the quantization noise. However, the cut-off frequency of the PLL transfer function should be much higher than the modulation frequency to allow enough harmonics to pass for a linear chirp. This trade-off doesn't pose a problem if fast chirp generation is not required. Otherwise, it makes the design challenging.

TPM architecture can decouple modulation frequency from PLL loop bandwidth. It injects the modulation signal to both tuning port of VCO (DCO) and feedback path [45], [46]. With two-point injection, the transfer function of the PLL could be expressed as Eq. 1.22, which is a all-pass function if g_0 is equal to f_{ref}/K_{DCO}. The TPM architecture allows fast chirp generation and provides more design freedom. But the circuitry is more complicated and there are more noise sources. Also, TPM is very sensitive to gain mismatch, as all-pass characterization disappears if g_0 is not equal to f_{ref}/K_{DCO}. As a result, calibration circuitry is needed to eliminate gain mismatch across PVT [46]. State-of-the-art PLLs for FMCW radar system are shown in Table. 1.2.

$$H_{TPM}(s) = f_{ref} \cdot \left[\frac{G_{loop}(s)}{1 + G_{loop}(s)} + \frac{g_0 K_{VCO} \cdot (1/f_{ref})}{1 + G_{loop}(s)} \right] \tag{1.22}$$

1.5 Conclusion

In this chapter, the background and design needs of the research are introduced. The proposed design is a PLL used as a chirp generator in FMCW radar systems.

The thesis is organized as follows: In Chapter 2, the architecture of the proposed PLL is explained and system-level modeling is presented. With the phase noise model and bandwidth model, some PLL specifications can be determined. In Chapter 3, a more accurate model for linearity and settling calculation is explored. This model provides comprehensive insight into PLL bandwidth design and the calculation provides more accurate data for dynamic phase noise calculation. Chapter 4 shows the dual-modulus divider and its controller, a Delta-Sigma Modulator. Chapter 5 illustrates an analytical modeling method for high-frequency circuit, which is used to design and optimize the prescaler and divider in the proposed PLL. In Chapter 6, a prototype CMOS design is illustrated. Chapter 7 draws the conclusion.

Chapter 2

PLL Architecture and System-level Modeling

2.1 Architecture of Proposed PLL

Background

In recent years, mm-wave FMCW radar at 140GHz is gaining more and more attention. Sitting between microwave frequencies and visible light frequencies, mm-wave has its own advantages.

Microwave radars, of which frequency is lower than that of mm-wave radars, suffer from large physical size [54] and limited bandwidth [55]. Compared with microwave radar, mm-wave radar has larger bandwidth, better angular resolution, better spacial resolution, and smaller physical size.

For visible light radars (LiDAR), such systems can be categorized into two types, namely flash type and scanning type [56]. Flash type LiDAR could achieve high resolution easily at the cost of the limited measurement distance (typically < 20m), for laser power needs to be diffused to all pixels. The scanning type could focus laser power on one column or one row and conduct horizontal or vertical scanning respectively, resulting in much larger measurement distance. However, its frame rate is lower due to its mechanical scanning mechanism. And to get good optical properties and thus good SNR, rotating mirror is commonly applied in the system, resulting in a bulky radar module, as shown in Fig. 2.1. Compared with LiDAR, mm-wave radar has lower resolution, but it's more compact and less sensitive to bad weather, strong light interference, and EM interference [57].

In summary, mm-wave radar systems have good angular resolution, good spacial resolution, and small physical size. Based on its compact size and precise motion sensing, 140GHz

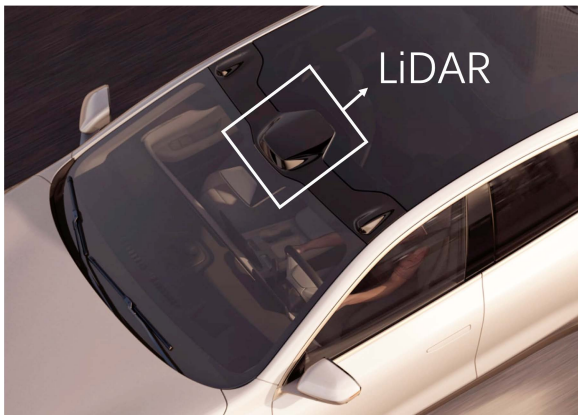


Figure 2.1: LiDAR module on a car with advanced driving assistance function.

radar could be applied to many applications, such as driving assistance, driver monitoring, patient monitoring, smart building, robotics, and gesture recognition [58].

The proposed PLL is used as the chirp generator in a 140GHz FMCW radar system. And some circuit blocks of the proposed PLL are re-used from an integer-N PLL from previous research [59], which is a transceiver operating at 140GHz.

Integer-N PLL and Fractional-N PLL

Fig. 2.2 shows the block diagram of conventional charge-pump PLL (CPPLL), consisting of phase-frequency detector (PFD), charge pump (CP), low-pass filter (LPF), voltage-controlled oscillator (VCO), and divider.

Depending on the division ratio of the divider, the PLL could be categorized into integer-N PLL and fractional-N PLL. For FMCW chirp generation, fractional-N PLL is commonly applied. Because for integer-N PLL, the bandwidth should be smaller than tenth of reference frequency, in order to avoid the stability issue due to the discrete sampling action of PFD (bandwidth should be twentieth of reference frequency considering PVT). Thus, faster settling requires higher reference frequency. However, the frequency step of the chirp is equal to the reference frequency, meaning that finer frequency step requires smaller reference frequency. As a result, there is a fundamental trade-off between fast settling and small frequency step, making it hard to generate a fast and linear chirp.

Fractional-N PLL is widely used as chirp generator because its frequency step is de-coupled from PLL bandwidth. Its frequency step could be smaller than reference frequency so in

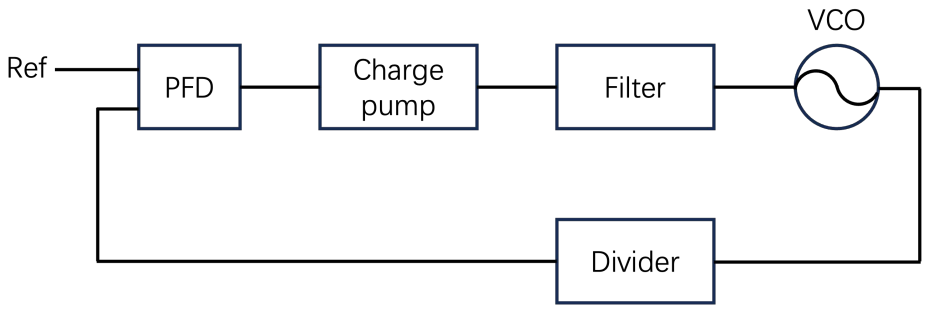


Figure 2.2: Block diagram of charge-pump PLL.

order to generate a chirp with fine step, low frequency reference is not required and much higher reference frequency is applicable.

Architecture

Different from conventional fractional-N PLL, the frequency of chirp generator's output ramps up with time linearly. And to generate the frequency ramp, modulation needs be injected at some point of the loop. Generally, there are two modulation schemes for one-point modulation, namely reference modulation and division-ratio modulation, as shown in Fig. 2.3 and Fig. 2.4 respectively.

In the case of reference-modulated architecture, the reference signal, typically a chirp, is usually supplied by an external instrument. The PLL serves as a frequency multiplier in this architecture, which is usually utilized for testing and prototyping FMCW radar systems. For division-ratio-modulated architecture, the division ratio increases linearly over time. This architecture is suitable for a fully-integrated FMCW radar system. Therefore, the division-ratio-modulated architecture is applied to the proposed PLL.

2.2 System-level Modeling

In this section, models of phase noise and PLL bandwidth are discussed. The corresponding code is in Appendix A. For phase noise, the phase noise contribution of each circuit block of the PLL will be discussed. Based on the zero-dead-zone PFD, a new model is proposed to calculate the charge pump phase noise. For PLL bandwidth, Fig. 2.5 shows the comparison of generated chirp with different PLL bandwidths. If the PLL is too slow, in other words, if the bandwidth of the PLL is too narrow, then the generated chirp cannot follow the ideal chirp. This leads to a nonlinear chirp and an unpredictable FMCW slope. On the contrary, if the PLL is too fast, or if the PLL bandwidth is very wide, every time the division ratio

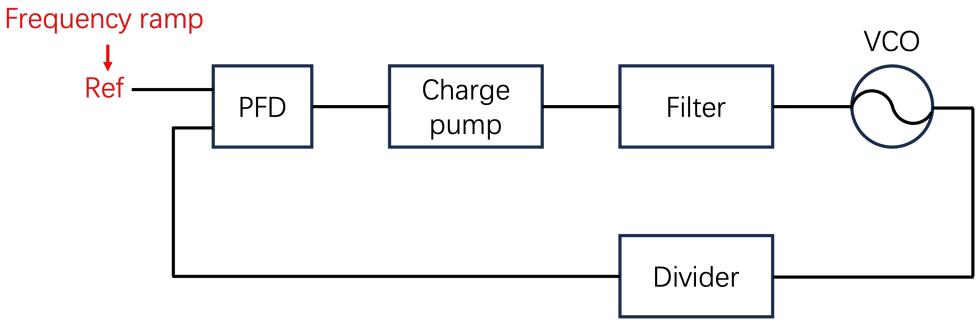


Figure 2.3: Reference-modulated PLL-based chirp generator.

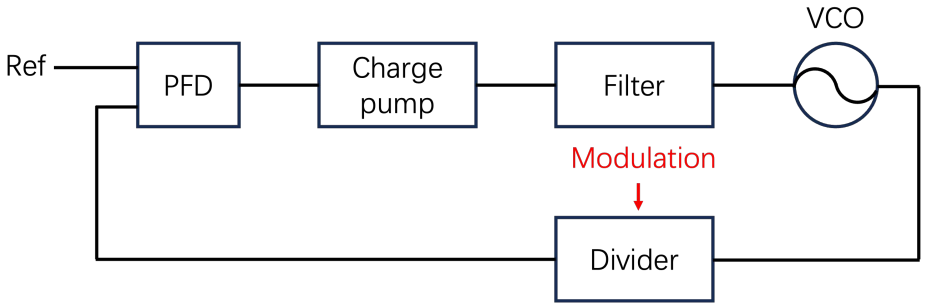


Figure 2.4: Division-ratio-modulated PLL-based chirp generator.

changes, the PLL would settle quickly. The resultant chirp would have a step shape, which results in poor linearity. Such bandwidth is denoted by BW_{settle} where the PLL can settle on each step. As a result, when designing a PLL, its bandwidth should fulfill the following relation:

$$\frac{1}{T_c} \ll BW_{PLL} \ll BW_{settle} \tag{2.1}$$

In this section, the method for calculating BW_{settle} will be shown. A more accurate method for designing PLL bandwidth will be discussed in the next chapter, which can not only provide a rough range of PLL bandwidth but also tune and optimize the chirp linearity without time-consuming circuit simulation.

Phase Noise Modeling

A general phase-domain model of a charge-pump PLL, which includes the main noise sources, is shown in Fig. 2.6. The corresponding forward gain, G(s), and feedback gain,

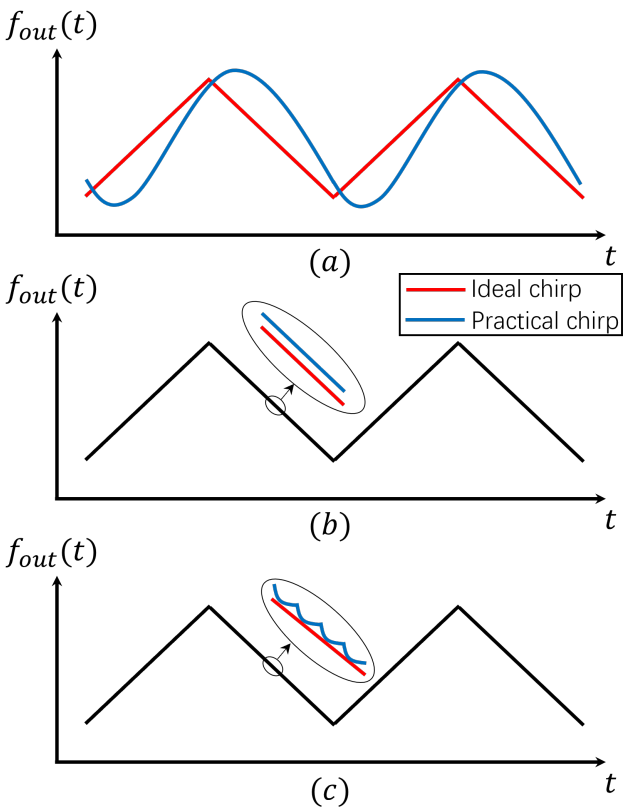


Figure 2.5: Generated chirp with different bandwidth. (a) PLL bandwidth is too narrow (b) PLL bandwidth is suitable (c) PLL bandwidth is too wide.

H(s), are given by

$$G(s) = \frac{I_{CP}K_{VCO}Z(s)}{s} \tag{2.2}$$

and

$$H(s) = \frac{1}{N} \tag{2.3}$$

respectively, where Z(s) is the transfer function of the low-pass filter. The loop gain is then given by

$$T(s) = G(s)H(s) = \frac{I_{CP}K_{VCO}Z(s)}{sN} \tag{2.4}$$

To calculate the total output noise and plot the noise contribution from each noise source, each circuit block must be simulated using a circuit simulator. The resultant phase noise spectrum can then be used to calculate the total output phase noise in MATLAB.

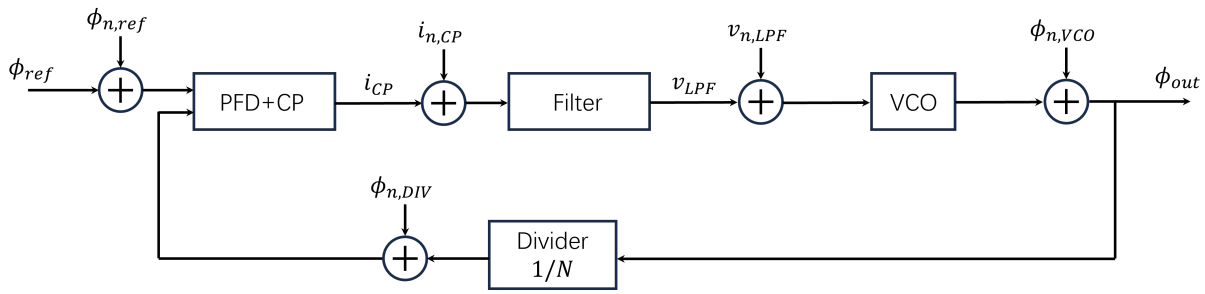


Figure 2.6: Phase-domain model of charge-pump PLL with main noise sources.

Table 2.1: Phase noise chart of Keysight E8267D in dBc/Hz.

	Offset from carrier (Hz)							
Frequency	1	10	100	1k	10k	100k	1M	10M
>2 to 3.2GHz	-58	-84	-102	-117	-134	-134	-150	-150

Reference Noise

Generally, the reference noise spectrum is not obtained from circuit simulation because the reference signal is typically generated from a crystal or testing equipment, depending on the application scenario or testing setup. The datasheet of the crystal or testing equipment would describe the phase noise spectrum of the source signal it generates. For example, the Keysight E8267D PSG vector signal generator is used as the reference source. Its datasheet [60] provides its phase noise parameters, as shown in Table. 2.1.

Based on the phase noise data, the phase noise spectrum of the reference signal,S_{ref}, can be plotted in MATLAB with interpolation and extrapolation as shown in Fig. 2.7. The noise gain from reference signal phase to output phase is

$$G_{n,ref} = \frac{\phi_{out}}{\phi_{n,ref}} = \frac{NG(s)}{N + G(s)} \tag{2.5}$$

And the corresponding power spectral density of phase noise at the output is

$$S_{ref,out} = S_{ref} \cdot |G_{n,ref}|^2 \tag{2.6}$$

Filter Noise

In the proposed design, the low-pass filter is a passive network. The noise of such a network can be modeled as a voltage noise source in series with the network's impedance $Z_{passive}(s)$,

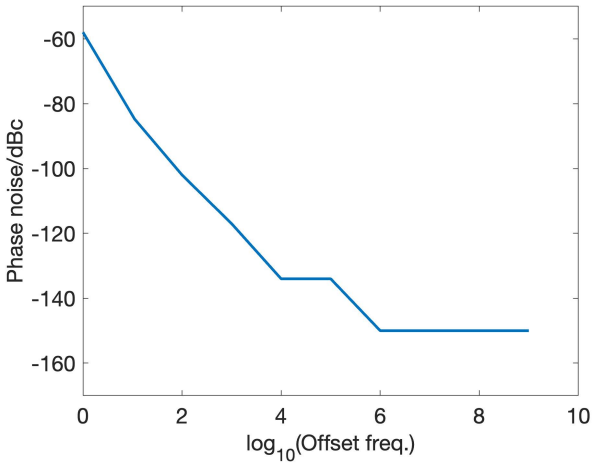


Figure 2.7: Phase noise spectrum of reference signal.

or as a current noise source in parallel with the network's admittance $Y_{passive}(s)$. For the low-pass filter, the voltage noise power is given by

$$S_{V_{n,LPF}} = 4k_B T\mathcal{R}\{Z(s)\} \tag{2.7}$$

where k_B is Boltzmann's constant, T is the temperature, and $Z(s)$ is the transfer function of the filter. The gain from the filter noise voltage to the output phase is

$$G_{n,LPF} = \frac{\phi_{out}}{v_{n,LPF}} = \frac{2\pi}{I_{CP}Z(s)} \cdot \frac{NG(s)}{N + G(s)} \tag{2.8}$$

where I_{CP} is the charging current of the charge pump. The filter noise spectrum at the output is then given by

$$S_{LPF,out} = S_{V_{n,LPF}} \cdot |G_{n,LPF}|^2 \tag{2.9}$$

VCO Noise

For the phase noise of LC oscillators, Leeson's model is most commonly used [61]. This model accounts for flicker noise and thermal noise, and the phase noise power spectral density can be expressed as

$$S_{VCO}(\Delta f) = n\left(\frac{1}{\Delta f^2} + \frac{f_c}{\Delta f^3}\right) \tag{2.10}$$

where n is the noise parameter, f_c is the corner frequency, and Δf is the offset frequency from the carrier. The values of f_c and n can be extracted or calculated from the VCO phase

noise simulation results. In the proposed design, the corner frequency $f_c = 10\text{MHz}$ and $n = 2.87\ \text{rad}^2/\text{Hz}$. The phase noise at the output is then given by

$$S_{VCO,out} = S_{VCO} \cdot |G_{n,VCO}|^2 \tag{2.11}$$

where the gain $G_{n,VCO}$ is given by

$$G_{n,VCO} = \frac{\phi_{out}}{\phi_{n,VCO}} = \frac{N}{N + G(s)} \tag{2.12}$$

Divider Noise

Divider noise in fractional-N PLL is primarily generated by the modulation signal that controls the divider. To achieve fractional division ratio, the divider must continually switch between two integer division ratios, ensuring that the average division ratio equals the desired non-integer value. In the proposed design, a Delta-Sigma Modulator is applied to control the dual-modulus divider and more details of the scheme will be covered in a later chapter. This section focuses solely on the noise contribution, which can be modeled as [62] [63]

$$S_{DSM}(\Delta f) = \frac{(2\pi)^2}{12 f_{ref}} \left[2\, sin\left(\frac{\pi \Delta f}{f_{ref}} \right) \right]^{2(m-1)} \tag{2.13}$$

where f_{ref} is the reference frequency and m is the order of Delta-Sigma Modulator. The gain from divider to the output is

$$G_{n,DSM} = -\frac{NG(s)}{N + G(s)} \tag{2.14}$$

And the noise contribution at the output is

$$S_{DSM,out} = S_{DSM} \cdot |G_{n,DSM}|^2 \tag{2.15}$$

PFD and CP Noise

The noise from the charge pump in steady-state is influenced by the conducting time of the charging and discharging currents, I_{up} and I_{dn}. A higher duty cycle increases the noise due to the injection of more noise current.

Mathematically, [64] derives the expression for the phase noise of the charge pump. Fig. 2.8 depicts a simplified charge pump, where the UP and DN signals control the charging and discharging currents, respectively, and $I_{up} = I_{dn} = I_{CP}$. When the charging current is active, its noise component $i_{n,up}$ also flows into the filter, thereby increasing the output noise. The same principle applies to the discharging current and $i_{n,dn}$. The total output noise power is the sum of the power of the charging noise and discharging noise, as they are incoherent noise sources.

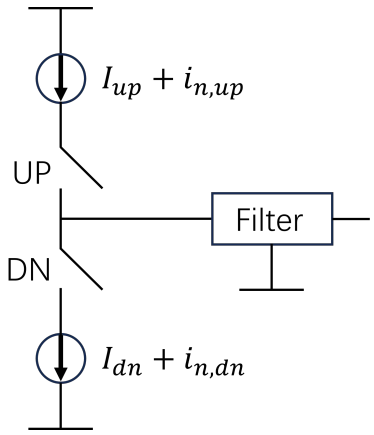


Figure 2.8: Simplified schematic of charge pump.

To compute the output noise, the waveform of the switched current can be represented as the product of the input noise current and the control signal (see Fig. 2.9).

$$i_{n,sw,up}(t) = i_{n,up} \cdot UP(t) \tag{2.16}$$

$$i_{n,sw,dn}(t) = i_{n,dn} \cdot DN(t) \tag{2.17}$$

As a result, the output noise spectrum is the convolution of input noise spectrum and squared magnitude of the Fourier transform of the control signal.

$$S_{sw,up} = S_{n,up} * \left|\mathcal{F}\{UP(t)\}\right|^2 \tag{2.18}$$

$$S_{sw,dn} = S_{n,dn} * \left|\mathcal{F}\{DN(t)\}\right|^2 \tag{2.19}$$

where the Fourier transform of the control is

$$\left|\mathcal{F}\{DN(t)\}\right|^2 (f) = \frac{\tau^2}{T_{ref}^2} \sum_{k=-\infty}^{\infty} \left[sinc^2 \left(\frac{k\tau}{T_{ref}} \right) \cdot \delta \left(f - \frac{k}{T_{ref}} \right) \right] \tag{2.20}$$

The current noise comprises two components: white noise and flicker noise. To determine the output noise spectrum, we need to calculate the output white noise power and the flicker corner frequency. The white noise power can be computed by summing the series in Eq. 2.20.

$$S_{sw,white} = S_{n,white} \cdot \frac{\tau}{T_{ref}} \tag{2.21}$$

where τ is the pulse width of $UP(t)$ or $DN(t)$ as shown in Fig. 2.9. The switched-noise power is the continuous-time noise scaled by duty cycle $\alpha_{CP} = \frac{\tau}{T_{ref}}$.

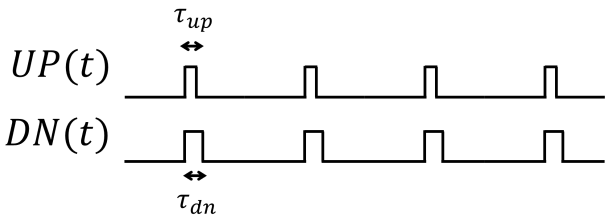


Figure 2.9: Waveform of control signals, UP and DN.

For flicker corner frequency, it's usually lower than $\frac{f_{ref}}{2}$ in fractional-N PLL, so it's not affected by noise folding due to Eq. 2.20. And the output flicker noise spectrum is

$$S_{sw,flicker} = S_{n,flicker} \cdot \left(\frac{\tau}{T_{ref}}\right)^2 \tag{2.22}$$

The switched flicker noise power is the continuous-time noise scaled by square of duty cycle. Due to different scaling factors of the white noise and the flicker noise, the corner frequency of output noise spectrum is scaled by α_{CP}.

$$f_{c,out} = f_{c,CP} \cdot \frac{\tau}{T_{ref}} \tag{2.23}$$

where $f_{c,CP}$ is the noise corner frequency of the charge pump, and $f_{c,out}$ is the corner frequency of switched noise. Thus, the output noise spectrum of charge pump is [65][66]

$$S_{i_n,CP}(f) = S_n^{white} \cdot \alpha_{CP}\left(1 + \frac{\alpha_{CP} \cdot f_{c,CP}}{f}\right) \tag{2.24}$$

The noise gain from charge pump output current to output phase is

$$G_{n,CP} = \frac{\phi_{out}}{i_{n,CP}} = \frac{2\pi}{I_{CP}} \cdot \frac{N \cdot G(s)}{N + G(s)} \tag{2.25}$$

and the noise spectrum at the output is

$$S_{CP,out} = S_{i_n,CP} \cdot |G_{n,CP}|^2 \tag{2.26}$$

In this model, the white noise S_n^{white} and corner frequency $f_{c,CP}$ are obtained from simulation of PFD and CP, the setup of which is shown in Fig. 2.10. The reference signal and divided signal are generated by two signal sources, between which a delay is applied to simulate phase noise under different steady-state phase error values. For each phase error value, the noise spectrum can be obtained from PSS simulation, from which S_n^{white} and $f_{c,CP}$ can be extracted from this spectrum.

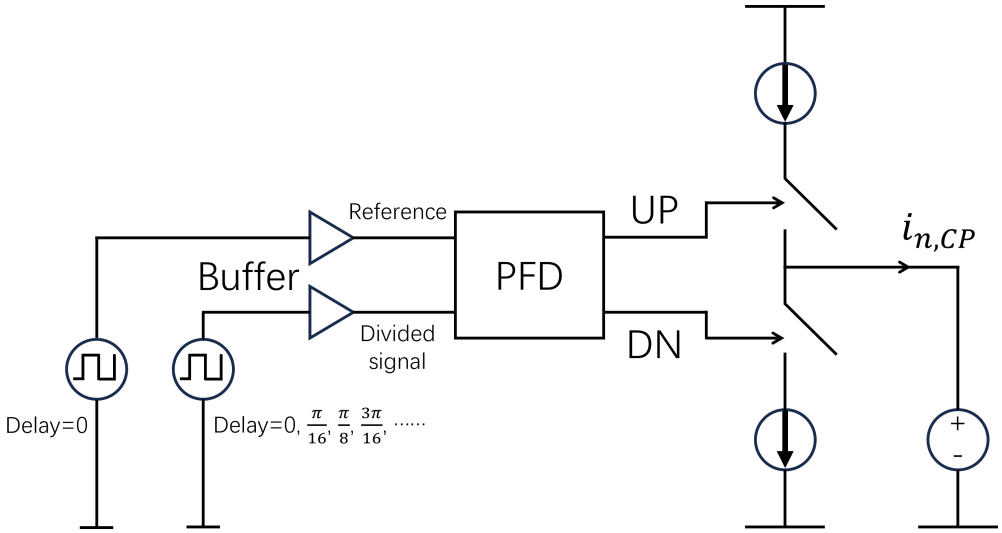


Figure 2.10: Noise simulation setup for PFD and charge pump.

However, in many PLL designs, the phase frequency detector (PFD) is implemented with dead-zone elimination structure for better linearity. For instance, in the proposed design, a conventional NAND-gate-based PFD is used as shown in Fig. 2.11 and a dead-zone elimination buffer is inserted between the NAND gate output and the reset terminal. This converts a very small phase error from a very narrow pulse into two wider pulses with a difference in width. In such architecture, the duty cycle of charging current and discharging current cannot be zero, even if the phase error is 0, which contradicts the previous model Eq. 2.24. Simulation results (Fig. 2.12) shows that after the reference signal and divided signal (overlapped with reference and not shown) rise from 0 to 1V, both UP and DN signals go high after a certain delay and then fall down to 0 together. In this simulation, reference frequency is 2.875GHz, and pulse width of UP and DN are 0.021ns and 0.025ns respectively. Thus, the average duty cycle of the charge pump is $\alpha_0 = 6.6125\%$. The overlapping behavior of UP and DN comes from the delay of the dead-zone elimination buffer, and it persists when phase error is nonzero as shown in Fig. 2.13.

As a result, Eq. 2.24 needs modification. Fig. 2.14 illustrates the phase noise of the PFD and charge pump at various phase error values. The new model incorporates two factors: the new noise floor and the new noise corner. Specifically, the previous noise floor is $S_n^{white} \cdot \alpha_{CP}$, while the new noise floor results from the power addition of incoherent noise, given by $S_n^{white} \cdot (2\alpha_0 + \alpha_{CP})$. In this case, $\alpha_0 = 6.6125\%$ as mentioned. And S_n^{white} can be determined from the data presented in Fig. 2.14 using the expression

$$S_n^{white} = \left(\frac{3.5 \times 10^{-12} A/\sqrt{Hz}}{2\alpha_0} \right)^2 \tag{2.27}$$

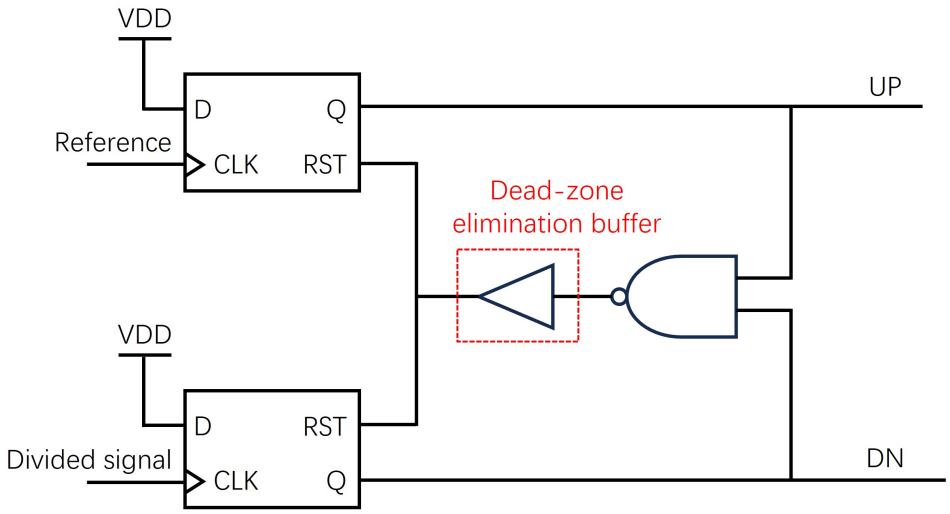


Figure 2.11: Schematic of NAND-gate PFD with dead-zone elimination buffer.

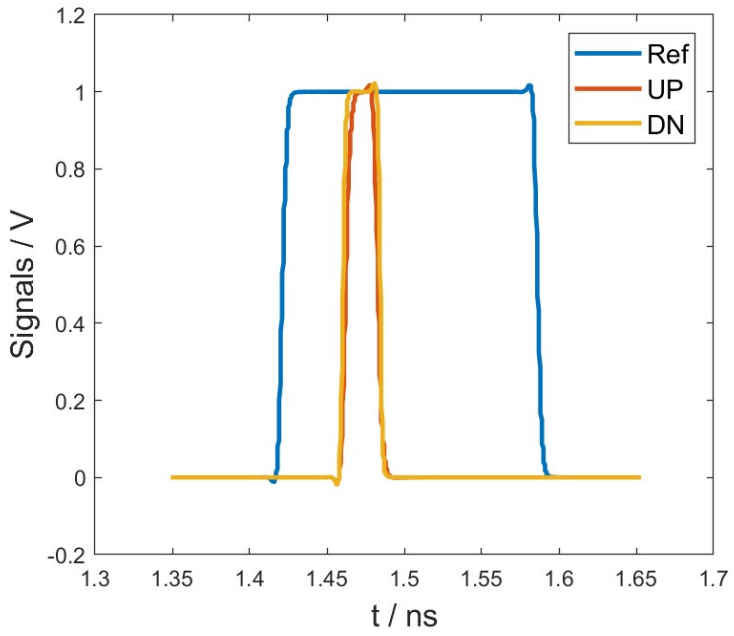


Figure 2.12: Waveform of UP and DN signals when phase error (between reference and divided signal) is 0.

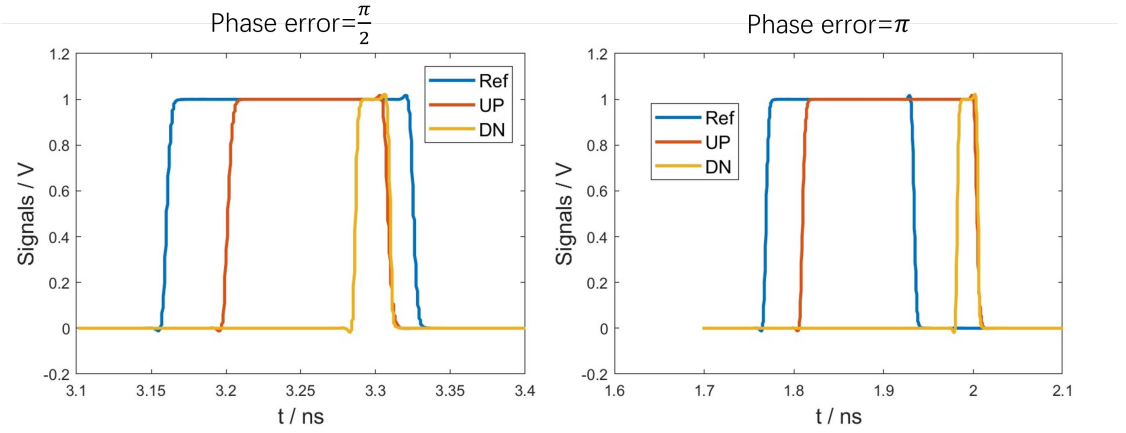


Figure 2.13: Waveform of UP and DN signals when phase error (between reference and divided signal) is $\frac{\pi}{2}$ or π.

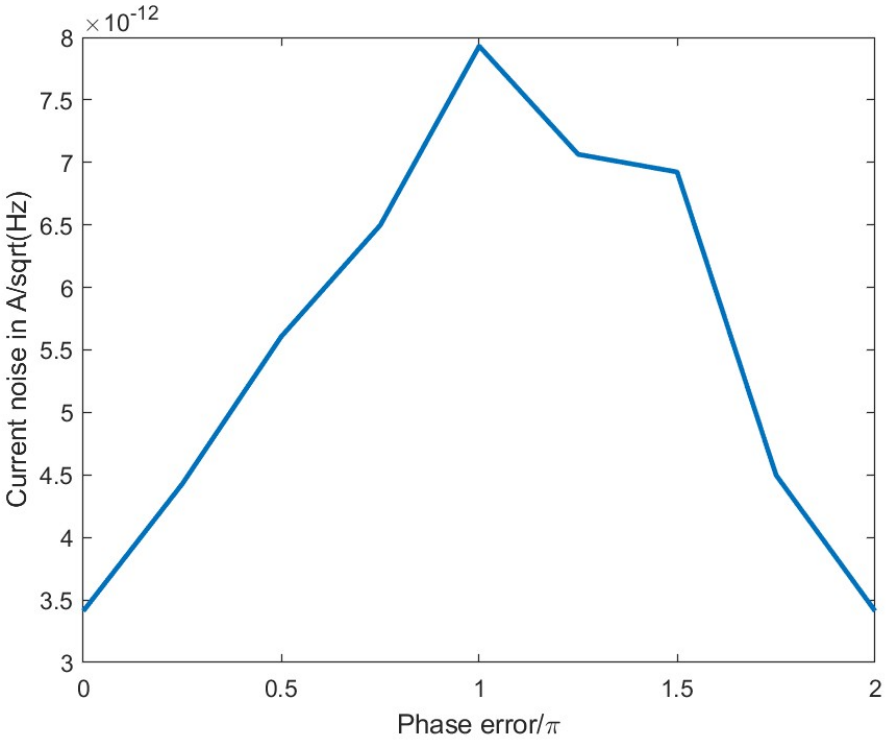


Figure 2.14: Phase noise of PFD and charge pump at different phase error values.

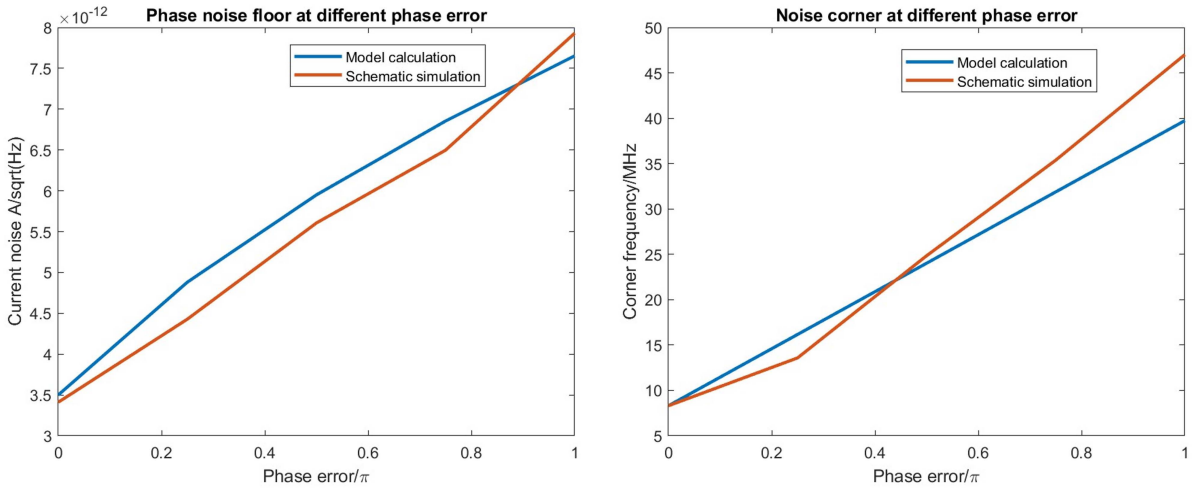


Figure 2.15: Comparison between proposed model and simulation results.

The same principle applies to the new corner frequency. The previous corner frequency, $\alpha_{CP} \cdot f_{c,CP}$, is replaced with $(2\alpha_0 + \alpha_{CP}) \cdot f_{c,CP}$. The new noise floor and new corner frequency match the simulation well as depicted in Fig. 2.15. Consequently, the proposed new phase noise model is

$$S_{i_n,CP}(f) = S_n^{white} \cdot (2\alpha_0 + \alpha_{CP}) \left(1 + \frac{(2\alpha_0 + \alpha_{CP}) \cdot f_{c,CP}}{f}\right) \tag{2.28}$$

Total Noise

By adding up the noise from all noise sources, the total phase noise at the output can be calculated as

$$S_{out,total} = S_{ref,out} + S_{LPF,out} + S_{VCO,out} + S_{DSM,out} + S_{CP,out} \tag{2.29}$$

And noise contribution can be illustrated as shown in Fig. 2.16.

PLL Bandwidth Modeling

As previously mentioned, this section encompasses the modeling and calculation of the bandwidth where PLL can settle on such frequency step, specifically BW_{settle}.

The time-domain model of CPPLL is shown as Fig. 2.17. The phase error ϕ_e is

$$\phi_e(t) = \phi_i(t) - \phi_{div}(t) \tag{2.30}$$

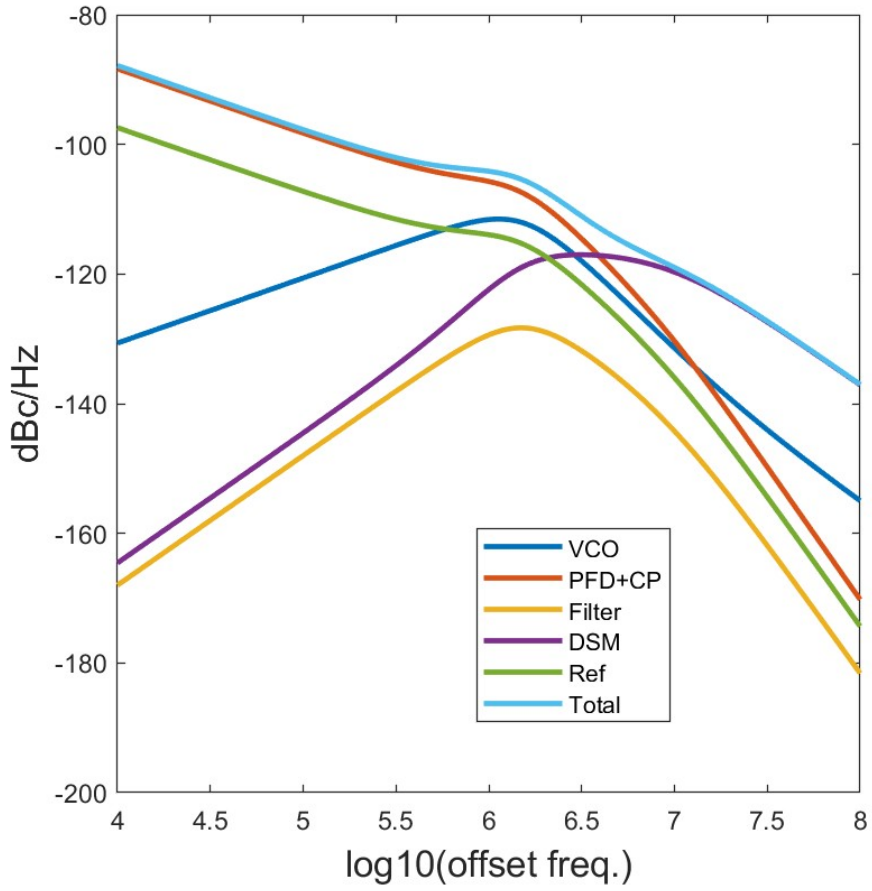


Figure 2.16: Comparison between proposed model and simulation results.

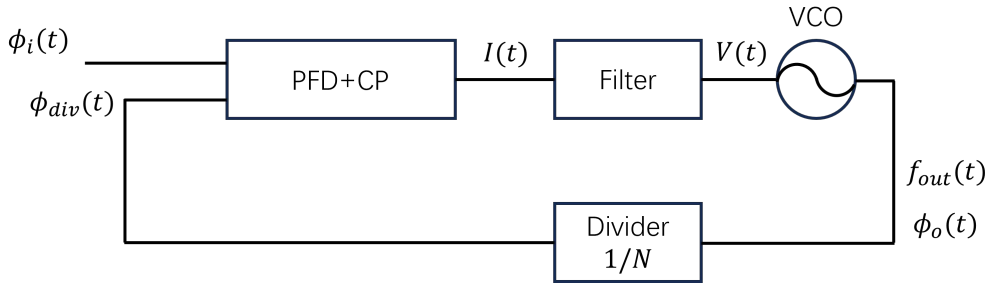


Figure 2.17: Time-domain model of charge-pump PLL.

The output current of charge pump is

$$I(t) = \phi_e(t) \cdot K_{PD} \tag{2.31}$$

where $K_{PD} = \frac{I_{CP}}{2\pi}$ is the gain of charge pump. For VCO, the frequency of its output is

$$\omega_{out}(t) = 2\pi f_{out}(t) = 2\pi \cdot K_{VCO} \cdot V(t) = \frac{d\left[N \cdot \phi_{div}(t)\right]}{dt} \tag{2.32}$$

where K_{VCO} is the VCO gain. Based on division ratio modulation for chirp generation, the output frequency

$$\omega_{out}(t) = \omega_{ref} \cdot (N_0 + \alpha \cdot t) \tag{2.33}$$

where ω_{ref} is the reference frequency, N_0 the division ratio at the beginning of the chirp, α the slope of division ratio. And the $N(t) = N_0 + \alpha t$. It's important to note that the expression for $N(t)$ assumes the division ratio of the divider changes continuously and linearly over time, which is a simplification and approximation. More accurate modeling will be discussed in Chapter 3. From Eq. 2.32 and Eq. 2.33, the output voltage of the filter can be expressed as

$$V(t) = \frac{1}{2\pi K_{VCO}} \cdot \left[\alpha\phi_{div}(t) + (N_0 + \alpha t)\frac{d\phi_{div}(t)}{dt}\right] \tag{2.34}$$

For the filter, in S-domain, its transfer function is

$$\frac{V}{I} = \frac{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s} \tag{2.35}$$

which can be written in time-domain as

$$a_n \cdot \frac{d^n V(t)}{dt^n} + \cdots + a_1 \cdot \frac{dV(t)}{dt} = b_n \cdot \frac{d^n I(t)}{dt^n} + \cdots + b_1 \cdot \frac{dI(t)}{dt} + b_0 \cdot I(t) \tag{2.36}$$

Here $V(t)$ can be substituted with $\phi_{div}(t)$ (Eq. 2.34), which is equal to $\phi_i(t) - \phi_e(t)$, and $I(t)$ can be substituted with $\phi_e(t)$ (Eq. 2.31). As a result, Eq. 2.36 can be written as a derivative equation of $\phi_e(t)$. By solving this equation, the settling process can be plotted and settling time can be calculated.

For low-pass filter, traditional topologies for 2nd-, 3rd-, and 4th-order filter is shown in Fig. 2.18 and their transfer function can be written as the general form

$$Z(s) = \frac{R_2 C_2 \cdot s + 1}{a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s} \tag{2.37}$$

where the coefficients a_1, a_2, a_3, and a_4 are summarized in Table. 2.2.

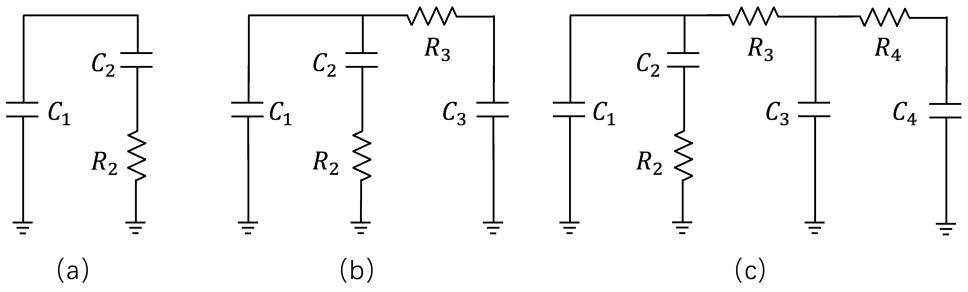


Figure 2.18: Topology for (a) 2nd-order filter (b) 3rd-order filter (c) 4th-order filter

Table 2.2: Coefficients for 2nd-, 3rd, and 4th-order low-pass filter transfer function.

2nd-order	a_1	$C_1 + C_2$
	a_2	$C_1 \cdot C_2 \cdot R_2$
	a_3	0
	a_4	0
3rd-order	a_1	$C_1 + C_2 + C_3$
	a_2	$C_2 \cdot R_2 \cdot (C_1 + C_3) + C_3 \cdot R_3 \cdot (C_1 + C_2)$
	a_3	$C_1 \cdot C_2 \cdot C_3 \cdot R_2 \cdot R_3$
	a_4	0
4th-order	a_1	$C_1 + C_2 + C_3 + C_4$
	a_2	$C_2 \cdot R_2 \cdot (C_1 + C_3 + C_4) + C_4 \cdot R_4 \cdot (C_1 + C_2 + C_3)$ $+ R_3 \cdot (C_1 + C_2) \cdot (C_3 + C_4)$
	a_3	$C_1 \cdot C_2 \cdot R_2 \cdot R_3 \cdot (C_3 + C_4) + C_4 \cdot R_4 \cdot (C_2 \cdot C_3 \cdot R_3$ $+ C_1 \cdot C_3 \cdot R_3 + C_1 \cdot C_2 \cdot R_2 + C_2 \cdot C_3 \cdot R_2)$
	a_4	$C_1 \cdot C_2 \cdot C_3 \cdot C_4 \cdot R_2 \cdot R_3 \cdot R_4$

2nd-order filter is taken as an example. When $n = 2$, Eq. 2.36 becomes

$$a_2 \cdot \frac{d^2V(t)}{dt^2} + a_1 \cdot \frac{dV(t)}{dt} = b_2 \cdot \frac{d^2I(t)}{dt^2} + b_1 \cdot \frac{dI(t)}{dt} + b_0 \cdot I(t) \tag{2.38}$$

From Eq. 2.34, $\frac{dV(t)}{dt}$ and $\frac{d^2V(t)}{dt^2}$ can be derived as

$$\frac{dV(t)}{dt} = \frac{1}{2\pi K_{VCO}} \left[2\alpha \frac{d\phi_{div}(t)}{dt} + (N_0 + \alpha t) \frac{d^2\phi_{div}(t)}{dt^2} \right] \tag{2.39}$$

$$\frac{d^2V(t)}{dt^2} = \frac{1}{2\pi K_{VCO}} \left[3\alpha \frac{d^2\phi(t)}{dt^2} + (N_0 + \alpha t) \frac{d^3\phi(t)}{dt^3} \right] \tag{2.40}$$

And based on Eq. 2.30, following relationships can be derived

$$\frac{d\phi_e(t)}{dt} = \omega_{ref} - \frac{d\phi_{div}(t)}{dt} \tag{2.41}$$

$$\frac{d^2\phi_e(t)}{dt^2} = -\frac{d^2\phi_{div}(t)}{dt^2} \tag{2.42}$$

$$\frac{d^3\phi_e(t)}{dt^3} = -\frac{d^3\phi_{div}(t)}{dt^3} \tag{2.43}$$

Substituting $I(t)$ with $\phi_e(t)$ (Eq. 2.31) in Eq. 2.38, an ordinary differential equation of $\phi_e(t)$ can be derived

$$a_2 \cdot \tfrac{-1}{2\pi K_{VCO}} \left[3\alpha \tfrac{d^2\phi_e(t)}{dt^2} + (N_0+\alpha t)\tfrac{d^3\phi_e(t)}{dt^3}\right] + a_1 \cdot \tfrac{1}{2\pi K_{VCO}} \left[2\alpha\omega_{ref} - 2\alpha\tfrac{d\phi_e(t)}{dt} - (N_0+\alpha t)\tfrac{d^2\phi_e(t)}{dt^2}\right]$$

$$= b_2 \cdot K_{PD}\frac{d^2\phi_e(t)}{dt^2} + b_1 \cdot K_{PD}\frac{d\phi_e(t)}{dt} + b_0 \cdot K_{PD} \cdot \phi_e(t) \tag{2.44}$$

The settling time can be calculated by solving Eq. 2.44 with initial conditions. For ease of solving it in tools like MATLAB, Eq. 2.44 can be re-written as following first-order system

$$y_1' = y_2 \tag{2.45}$$

$$y_2' = y_3 \tag{2.46}$$

$$y_3' = \frac{-\frac{2\pi K_{VCO}}{a_2}(b_2 K_{PD}\cdot y_3 + b_1 K_{PD}\cdot y_2 + b_0 K_{PD}\cdot y_1) + \frac{a_1}{a_2}\left[2\alpha\omega_{ref} - 2\alpha\cdot y_2 - (N_0+\alpha t)\cdot y_3\right] - 3\alpha\cdot y_3}{N_0+\alpha t} \tag{2.47}$$

Based on this model, a PLL with a bandwidth BW_{settle} can be designed if settling time is much smaller than step time (less than tenth or twentieth of step time). Furthermore, the bandwidth of the practical design should be much smaller than BW_{settle}, as indicated by Eq. 2.1. Regarding the initial conditions, the initial value of $\phi_e(t)$ is set as π to calculate the worst-case settling time. And the settling threshold is within ± 0.01 degree of the steady-state value.

The settling time for 3rd-, 4th-, and higher-order low-pass filters can be calculated similarly. However, the expressions become significantly more complex, and deriving them requires the use of symbolic computation tools such as Mathematica, MATLAB, or Python.

2.3 PLL Specifications

With the above calculations, the FMCW chirp and PLL specifications are designed as shown in the following tables, and the flowchart is shown in Fig. 2.19. The DSM parameters in Table 2.3 will be discussed in Chapter 4. Table 2.4 shows the PLL parameters, and a more detailed and accurate settling model of the PLL will be discussed in Chapter 3. The dynamic phase noise is defined as the phase noise when the PLL is generating the chirp, as opposed to the static phase noise when the PLL output frequency is fixed at a certain value. The dynamic phase noise is generally higher than the static phase noise. This is because when generating the chirp in practical applications, the PLL never settles for better linearity, resulting in non-zero phase error, ϕ_e, which leads to higher noise injected into the charge pump.

Table 2.3: FMCW chirp specifications.

FMCW bandwidth	8 GHz
Chirp duration	40 μs
Chirp slope	200 MHz/μs
Number of DSM input bits	12
DSM input control word @ start	-2047
DSM input control word @ end	-1098
Number of frequency steps	950
Frequency step/resolution	0.7 MHz

Table 2.4: PLL specifications.

Step time	0.042 μs
Settling time	1.398 μs
Mean phase error	20.798°
Dynamic phase noise @10kHz	-86 dBc/Hz
Dynamic phase noise @1MHz	-103 dBc/Hz
PLL bandwidth	2.189 MHz
Phase margin	57°

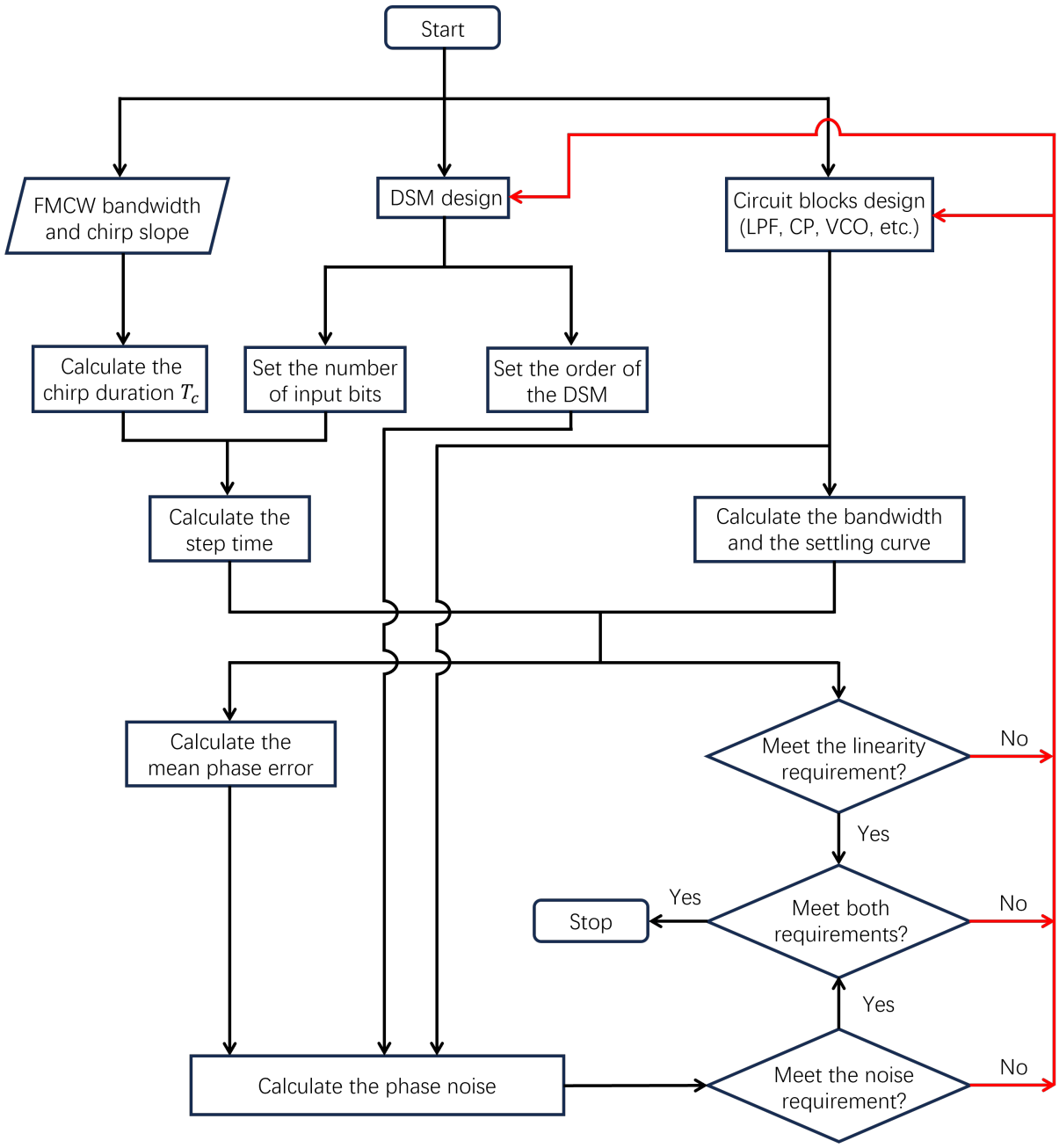


Figure 2.19: Design flowchart of the system-level specifications. The red lines represent the process of design iteration.

2.4 Conclusion

In this chapter, the choice of the PLL architecture is analyzed. Models for phase noise and PLL bandwidth are proposed, and PLL specifications are determined based on these models. More details of the PLL design and modeling will be discussed in subsequent chapters. The phase noise of the charge pump, a major contributor to phase noise, is dependent on ϕ_e. A more accurate calculation of ϕ_e is discussed in Chapter 3. Besides, the PLL bandwidth modeling in this chapter can only provide an upper limit of the PLL bandwidth. Calculations in Chapter 3 can help design the bandwidth more accurately.

Chapter 3

Linearity and Settling Model

As previously mentioned, in this chapter we discuss a more accurate model for settling. Based on this model, we propose a more precise method for designing a linear chirp. The model relies on time-domain calculations, which significantly reduce the computational time required for circuit transient simulations. In the later part of this chapter, we also delve into other modeling and calculation efforts based on s-domain analysis and the calculus of variations.

3.1 Practical Chirp

The chirp is generated by modulating the division ratio of the PLL. Ideally the division ratio increases linearly as

$$N(t) = N_0 + \alpha \cdot t \tag{3.1}$$

where N_0 is the initial division ratio and α is the slope of the $N(t)$. As a result, the ideal output frequency is

$$f_{out,ideal}(t) = f_{start} + \alpha f_{ref} t \tag{3.2}$$

where $f_{start} = N_0 \cdot f_{ref}$ is the output frequency at the beginning of the chirp.

However, the practical division ratio doesn't change continuously; instead, it changes step by step. As a result, the practical chirp is not perfectly linear. Three types of chirps are schematically illustrated in Fig. 3.1. The ideal chirp is perfectly linear, with a constant slope. In contrast, the chirp with infinite PLL bandwidth appears step-shaped because each time the division ratio changes, the output frequency quickly settles at the new frequency within a very short time. This curve can be used to mark the time points where the division ratio changes. The practical chirp, represented by the blue dashed line, deviates slightly from the ideal chirp, which is perfectly linear.

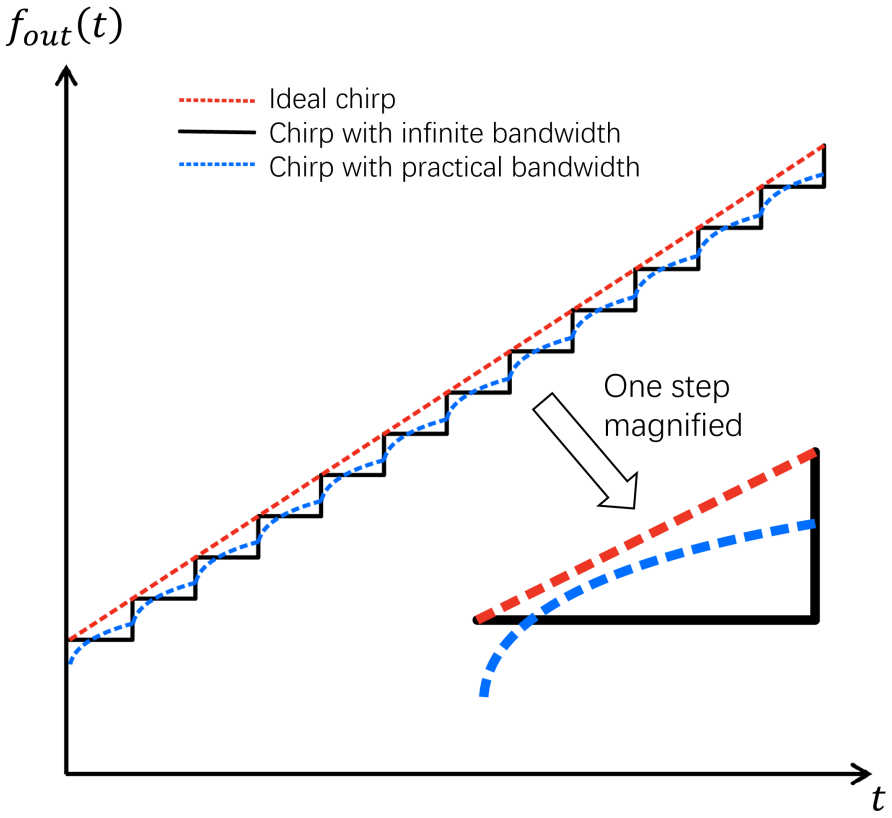


Figure 3.1: Ideal chirp and practical chirp.

To make the practical chirp as linear as possible, the proposed method calculates the output frequency, f_{out}, as a function of time, using parameters from PLL circuit blocks. Based on the instantaneous expression of $f_{out}(t)$, the root-mean-square (RMS) error between the practical chirp and ideal chirp can be calculated. This information can guide circuit optimization, especially in the design of PLL bandwidth.

3.2 Modeling

To analyze the system, the PLL cannot be considered as a time-invariant system, because the division ratio, one of the system parameters, keeps changing. However, the PLL is time-invariant in each step when the division ratio is fixed. For instance, during the step from $t_k = k\Delta t$ to $t_{k+1} = (k+1)\Delta t$ as shown in Fig. 3.2, the division ratio remains constant

$$N = N_0 + \alpha k \Delta t \tag{3.3}$$

where Δt is the step time of the FMCW chirp.

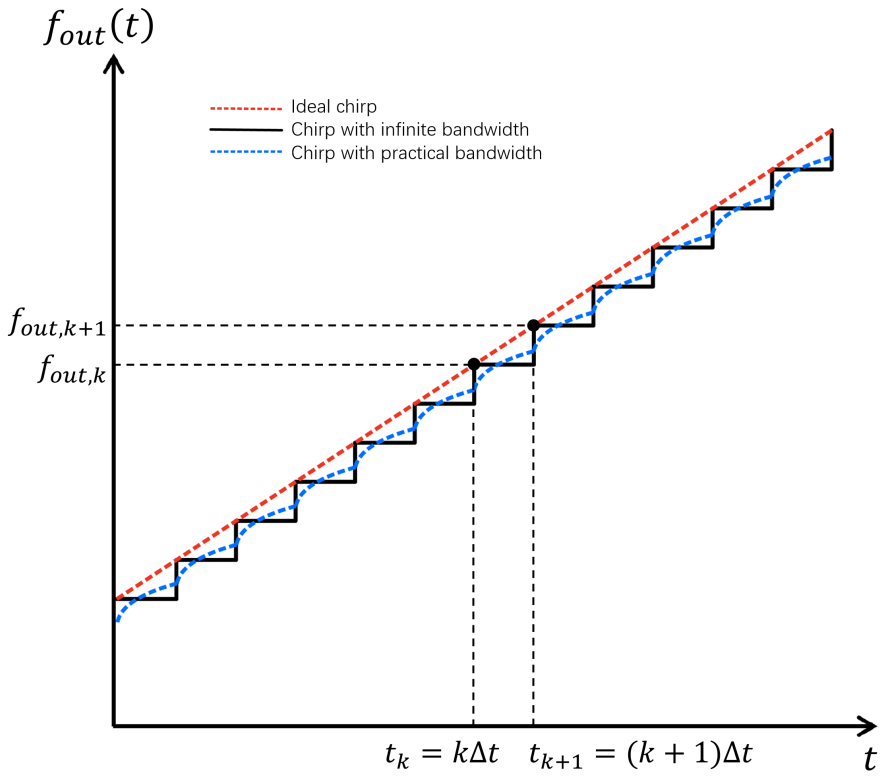


Figure 3.2: The step to be analyzed is $k\Delta t \sim (k+1)\Delta t$, where $t_k = k\Delta t$ and $t_{k+1} = (k+1)\Delta t$.

The time-domain derivation is performed for one step, which is schematically represented in Fig. 3.3. While part of the derivation is the same as the settling analysis in Chapter 2, the PLL is not expected to settle at each step in this chapter. Consequently, the initial conditions for each step, or the boundary conditions between steps, differ from those in the previous analysis.

Differential Equation of $\phi_e(t)$ in Each Step

The time-domain model of the PLL is shown in Fig. 3.4. The basic relations still hold

$$\phi_e(t) = \phi_i(t) - \phi_{div}(t) = \phi_i(t) - \frac{1}{N}\phi_o(t) \tag{3.4}$$

$$I(t) = \frac{I_{CP}}{2\pi}\phi_e(t) \tag{3.5}$$

$$V(t) = \frac{1}{2\pi K_{VCO}}\frac{d\phi_o(t)}{dt} = \frac{N}{2\pi K_{VCO}}\left(\frac{d\phi_i(t)}{dt} - \frac{d\phi_e(t)}{dt}\right) \tag{3.6}$$

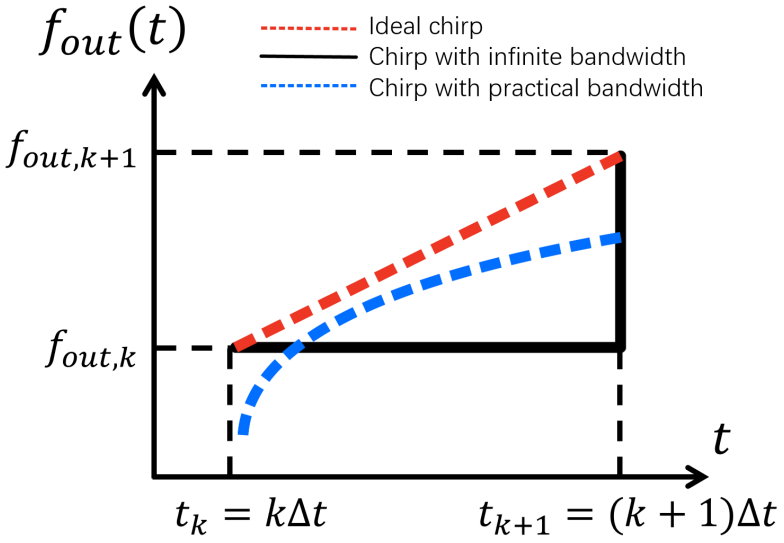


Figure 3.3: The step $k\Delta t \sim (k+1)\Delta t$.

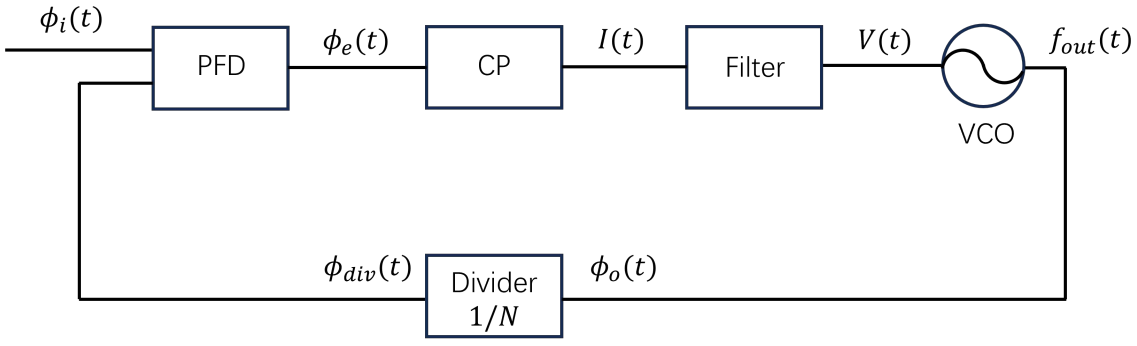


Figure 3.4: The time-domain model of the PLL

And the core equation is the time-domain $V(t) - I(t)$ relationship of the 2nd-order low-pass filter as mentioned in Chapter 2

$$a_2 \cdot \frac{d^2V(t)}{dt^2} + a_1 \cdot \frac{dV(t)}{dt} = b_1 \cdot \frac{dI(t)}{dt} + b_0 I(t) \tag{3.7}$$

Substituting Eq. 3.5 and Eq. 3.6 into Eq. 3.7 leads to a equation of $\phi_e(t)$

$$a_2 \frac{d^3\phi_e(t)}{dt^3} + a_1 \frac{d^2\phi_e(t)}{dt^2} + \frac{b_1 \cdot I_{CP} \cdot K_{VCO}}{N} \cdot \frac{d\phi_e(t)}{dt} + \frac{b_0 \cdot I_{CP} \cdot K_{VCO}}{N} \cdot \phi_e(t) = 0 \tag{3.8}$$

In MATLAB, to use the differential equation solver 'ode45', the equation should be rewritten as a first-order ODE system as following

$$x_1 = \phi_e(t) \tag{3.9}$$

$$x_2 = \frac{dx_1}{dt} \tag{3.10}$$

$$x_3 = \frac{dx_2}{dt} \tag{3.11}$$

$$\frac{dx_3}{dt} = -\frac{1}{a_2}\left(a_1 \cdot x_3 + \frac{b_1 \cdot I_{CP} \cdot K_{VCO}}{N_k} \cdot x_2 + \frac{b_0 \cdot I_{CP} \cdot K_{VCO}}{N_k} \cdot x_1\right) \tag{3.12}$$

Step Edge Analysis

The crucial part of analyzing the chirp is understanding what happens at the moment when the division ratio changes and how these changes affect the signals. Because such information is necessary to determine the initial values or boundary conditions in each step. At $t = t_k$ on Fig. 3.3, the division ratio switches from $N(t = t_k^-) = N_0 + \alpha(k-1)\Delta t$ to $N(t = t_k^+) = N_0 + \alpha k \Delta t$. At this moment, all instantaneous signals labelled in Fig. 3.4 remain unchanged. The only sudden change occurs in the division ratio and the output frequency $f_{out}(t)$ doesn't change at the moment. Consequently, there is a sudden change in the frequency of the divided signal, $f_{div}(t)$, indicating that the accumulation rate of $\phi_{div}(t)$ slightly decreases at $t = t_k$. The previous rate of accumulation or derivative is

$$\left.\frac{d\phi_{div}(t)}{dt}\right|_{t=t_k^-} = 2\pi f_{div}(t_k^-) = 2\pi \frac{f_{out,k}}{N_0 + \alpha(k-1)\Delta t} \tag{3.13}$$

where $f_{out,k}$ is the output frequency at $t = k\Delta t$. And the new derivative is

$$\left.\frac{d\phi_{div}(t)}{dt}\right|_{t=t_k^+} = 2\pi f_{div}(t_k^+) = 2\pi \frac{f_{out,k}}{N_0 + \alpha k \Delta t} \tag{3.14}$$

And their difference is defined as A_k

$$A_k = \left.\frac{d\phi_{div}(t)}{dt}\right|_{t=t_k^+} - \left.\frac{d\phi_{div}(t)}{dt}\right|_{t=t_k^-} = 2\pi\left(\frac{f_{out,k}}{N_0 + \alpha k \Delta t} - \frac{f_{out,k}}{N_0 + \alpha(k-1)\Delta t}\right) \tag{3.15}$$

As a result, the switching of division ratio introduces a step function $A_k \cdot u(t)$ to the quantity $\frac{d\phi_{div}(t)}{dt}$. And other quantities labelled in Fig. 3.4 don't experience any sudden change at the switching moment, due to the low-pass characteristic of the loop.

As the differential equation is in terms of $\phi_e(t)$, the sudden change should also be connected to the derivative of $\phi_e(t)$. From Eq. 3.4, we get

$$\frac{d\phi_e(t)}{dt} = \frac{d\phi_i(t)}{dt} - \frac{d\phi_{div}(t)}{dt} \tag{3.16}$$

So the quantity $\frac{d\phi_e(t)}{dt}$ also experiences a sudden change, $-A_k \cdot u(t)$, at $t = k\Delta t$.

From $\phi_e(t)$ to $f_{out}(t)$

The primary objective of modeling is to determine the instantaneous output frequency $f_{out}(t)$ and to optimize the chirp linearity from it. This involves calculating $\phi_e(t)$ using the provided equations and then converting it into $f_{out}(t)$. Two methods exist for this conversion, both of which establish a relationship between $\phi_e(t)$ and $f_{out}(t)$.

The first method employs the filter equation Eq. 3.7. In this method, $\phi_e(t)$ substitutes $I(t)$ with Eq. 3.5, and $f_{out}(t)$ substitutes $V(t)$ with

$$V(t) = \frac{1}{K_{VCO}} \cdot f_{out}(t) \tag{3.17}$$

This results in an equation in terms of $f_{out}(t)$ and $\phi_e(t)$

$$\frac{a_2}{K_{VCO}} \cdot \frac{d^2 f_{out(t)}}{dt^2} + \frac{a_1}{K_{VCO}} \cdot \frac{df_{out(t)}}{dt} = \frac{I_{CP} \cdot b_1}{2\pi} \cdot \frac{d\phi_e(t)}{dt} + \frac{I_{CP} \cdot b_0}{2\pi} \cdot \phi_e(t) \tag{3.18}$$

The standard form of the first-order ODE system, which is compatible with the MATLAB 'ode45' solver, is then used. This system includes three equations

$$y_1 = f_{out}(t) \tag{3.19}$$

$$y_2 = \frac{dy_1}{dt} \tag{3.20}$$

$$\frac{dy_2}{dt} = \frac{K_{VCO}}{a_2}\left(\frac{I_{CP} \cdot b_1}{2\pi} \cdot \frac{d\phi_e(t)}{dt} + \frac{I_{CP} \cdot b_0}{2\pi} \cdot \phi_e(t)\right) - \frac{a_1}{a_2} y_2 \tag{3.21}$$

Given $\phi_e(t)$ in each step, $f_{out}(t)$ can be computed using these equations.

The second method is simpler. It utilizes the relation of $\phi_e(t)$ and $f_{out}(t)$ in the feedback loop. As mentioned, $\phi_e(t)$ is the difference between reference signal phase and divided signal phase

$$\phi_e(t) = \phi_i(t) - \phi_{div}(t) \tag{3.22}$$

and in feedback loop, the divider gives $\phi_{div}(t) = \frac{1}{N}\phi_o(t)$, allowing $\phi_e(t)$ to be expressed in terms of $\phi_o(t)$

$$\phi_e(t) = \phi_i(t) - \frac{1}{N}\phi_o(t) \tag{3.23}$$

Differentiating on both sides results in

$$\frac{d\phi_e(t)}{dt} = 2\pi f_{ref} - \frac{2\pi}{N} \cdot f_{out}(t) \tag{3.24}$$

Consequently $f_{out}(t)$ can be calculated directly from $\frac{d\phi_e(t)}{dt}$ with

$$f_{out}(t) = N \cdot \left(f_{ref} - \frac{1}{2\pi} \cdot \frac{d\phi_e(t)}{dt} \right) \tag{3.25}$$

Note that in this equation, at the step edges, $\frac{d\phi_e(t)}{dt}$ undergoes a sudden change. Despite this, $f_{out}(t)$ remains unaffected due to simultaneous changes in N. Calculations confirm that both methods produce the same results.

Chirp Calculation

Given sufficient understanding of the dynamics in each step and the sudden changes at step edges, we can compute $\phi_e(t)$ and thus the instantaneous output frequency $f_{out}(t)$ numerically, step by step. By connecting these steps, we can determine $f_{out}(t)$ for the entire duration of the chirp. The initial conditions are set during calculation of the first step $t = 0 \sim \Delta t$ and the final values of all quantities are saved and used as the initial values of the next step $t = \Delta t \sim 2\Delta t$, with the addition of the sudden-change term (the step function). The entire chirp can be calculated numerically by repeating this process for each step. The procedure is depicted in Fig. 3.5. Using this method, RMS FM error can be computated for each set of PLL parameters $(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1)$. Based on these calculations, optimization can be performed. The code is presented in Appendix B.

3.3 Calculation Results and Chirp Linearity

In the proposed design, the chirp is generated by the PLL-based chirp generator followed by a $\times 6$ frequency multiplier. The FMCW chirp bandwidth is 8GHz. Thus, at output of the PLL, the chirp bandwidth is 1.33GHz. And PLL's output frequency starts from 23GHz and ends at 24.33GHz.

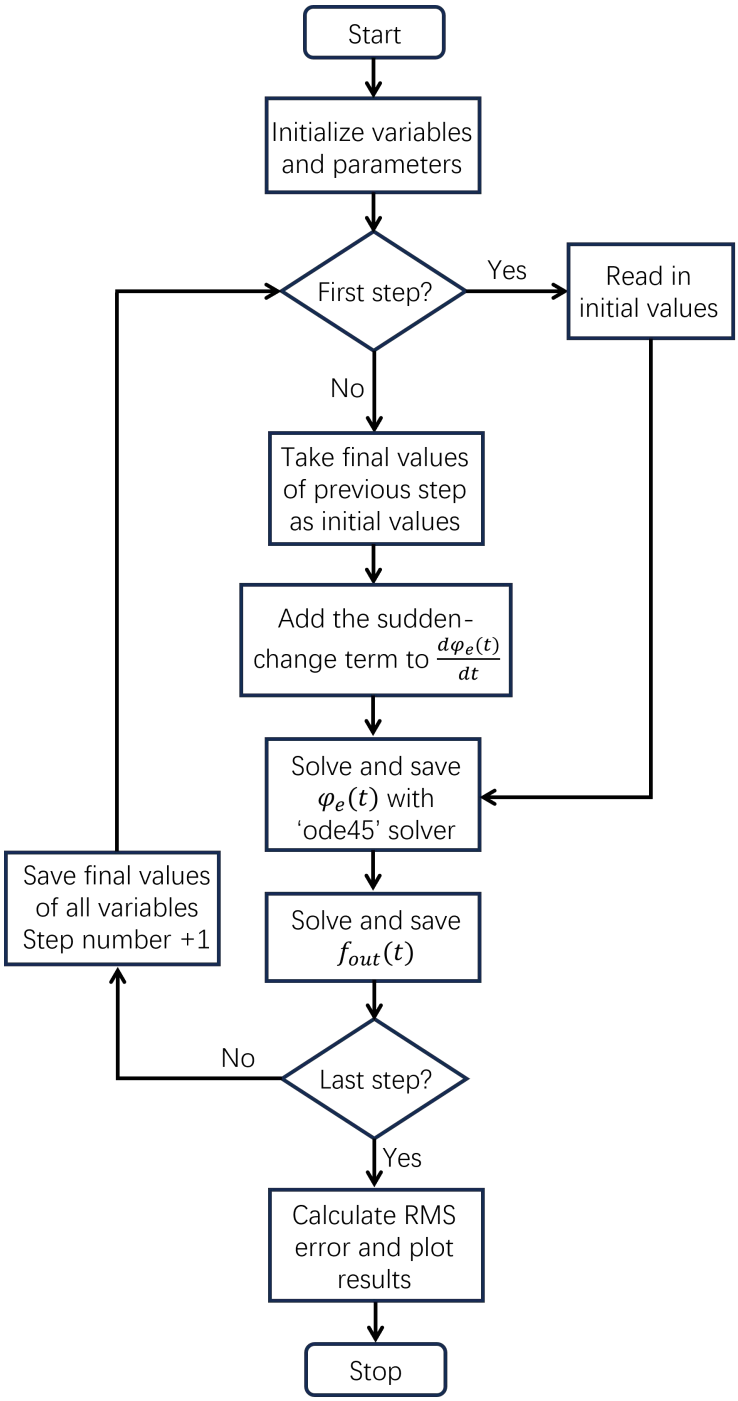


Figure 3.5: The flow chart of calculating $f_{out}(t)$

The chirp calculation can assist in designing the PLL in two ways. One way is to optimize the linearity of the chirp generated by the PLL, given a chirp slope. In this scenario, the PLL parameters are swept, and the corresponding chirps are calculated. The parameter combination that generates the most linear chirp is selected. The other way is to determine the range of chirp slope with acceptable linearity, given the PLL parameters. In this case, the PLL parameters are fixed, and the chirp slope values are swept. For each chirp slope value, the non-linearity metric is calculated. This allows designers to understand the range of chirp slope that the PLL can support with sufficient linearity.

The nonlinearity is measured by the FM RMS error between the generated chirp and a perfectly linear chirp. This metric is commonly used for characterizing PLL-based chirp generators. After calculating the chirp $f_{out}(t)$, the first 5% of it is cut off, because in practical application, the beginning part of a chirp is not used for mixing with received signal, but for starting chirp, TX, and ADC [67]. Then the average difference between the ideal chirp and generated chirp is calculated and denoted as

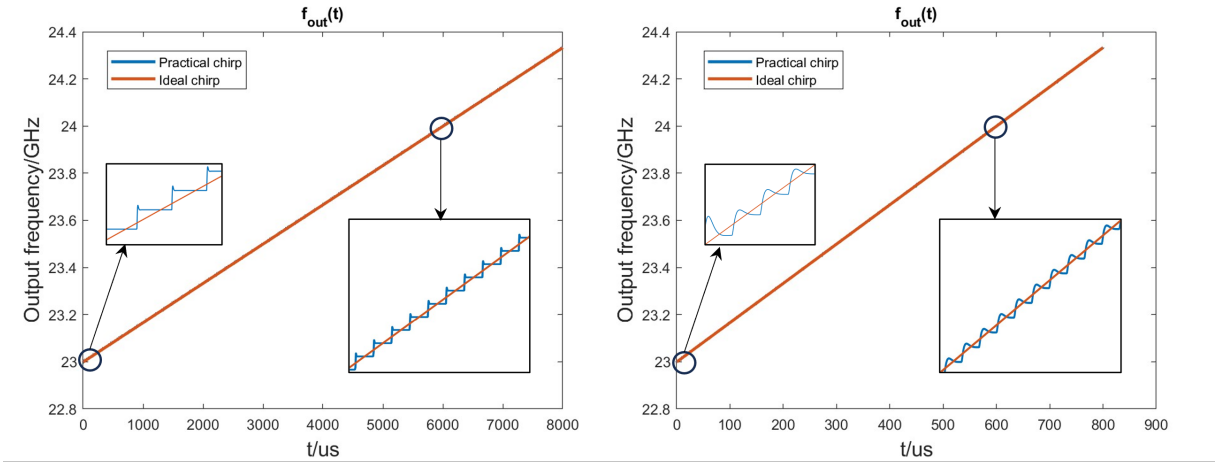
$$\Delta f = average\left[f_{ideal}(t) - f_{out}(t)\right] \tag{3.26}$$

In many applications, a slight shift from the designed frequency value does not affect the performance of the FMCW radar system, as long as the chirp bandwidth is as designed and the linearity is good. So, in the optimization of linearity, this Δf acts as a DC frequency error that has nothing to do with linearity. It should be subtracted from the ideal chirp before calculating the RMS error.

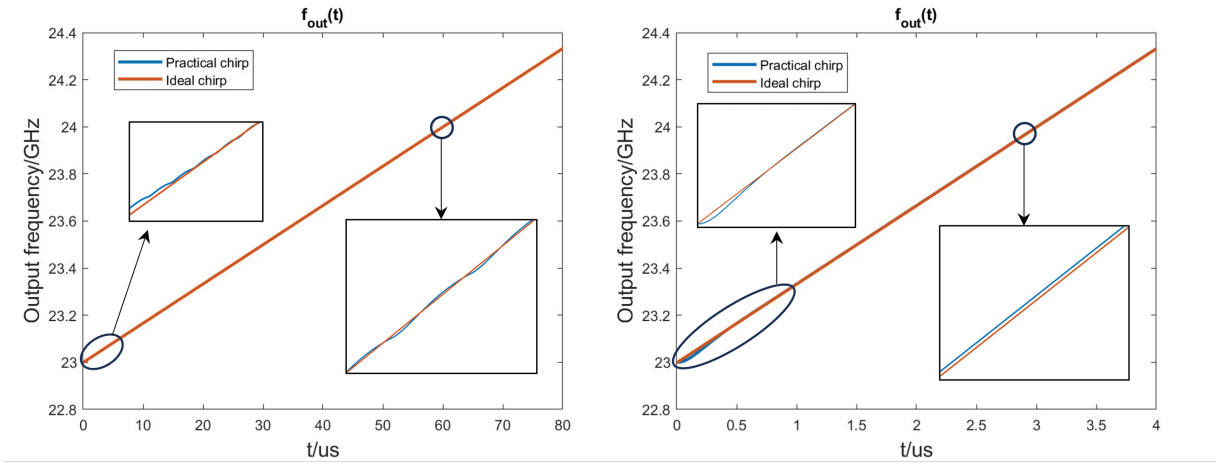
Fig. 3.6 shows the calculation results of a PLL with a 2.19MHz bandwidth. The chirp slope labeled in the picture is the slope at the output of the PLL, before the $\times 6$ frequency multiplier. When the chirp slope is small, the generated chirp has a step shape and settles quickly at the beginning, both due to the fast PLL. As the chirp slope gets larger, the generated chirp becomes more linear and smooth. However, it takes longer to settle at the beginning. In summary, the nonlinearity mainly comes from step-shaped chirp and slow settling at the beginning. The PLL bandwidth should be chosen to balance this trade-off.

Removing the initial 5% of the generated chirp can eliminate the settling phase if the settling occurs rapidly. However, if the settling is not quick, the chirp may still be in the process of settling after the first 5%, contributing to nonlinearity. Therefore, the optimal PLL bandwidth should be the one that allows settling within the first 5% of the chirp duration. It's important to note that the initial phase does not necessarily constitute 5% of T_{chirp}. The settling time is determined by the specific requirements of the application.

The error introduced by the stair-shaped chirp is also related to the total number of steps, or the frequency step size. The smaller the frequency step size, the smaller the RMS error.



(a) Chirp slope=$\frac{1MHZ/\mu s}{6}$ (b) Chirp slope=$\frac{10MHZ/\mu s}{6}$



(c) Chirp slope=$\frac{100MHZ/\mu s}{6}$ (d) Chirp slope=$\frac{2000MHZ/\mu s}{6}$

Figure 3.6: A PLL with 2.19MHz bandwidth at different chirp slopes.

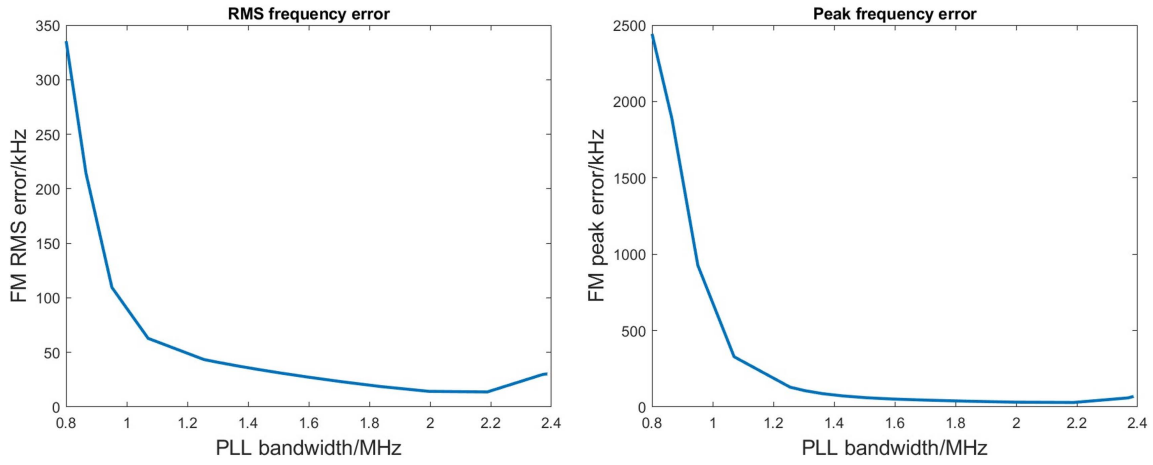


Figure 3.7: Calculation results for 200MHz/μs FMCW chirp slope. Both RMS error and peak error are plotted.

A finer frequency step requires more hardware resources and a higher clock frequency for the Delta-Sigma Modulator (DSM). This will be discussed in detail in the next chapter.

When designing the PLL, its parameters can be swept to find the optimal combination for linearity. For instance, in the proposed architecture, the C_1 in the 2nd-order filter can be swept to tune the PLL bandwidth. The FM RMS error and peak error can be calculated for each bandwidth. In Fig. 3.7, the FMCW chirp slope is set as 200MHz/μs (the corresponding slope at the output of the PLL is $\frac{200\text{MHz}/\mu\text{s}}{6}$), and the optimal range of the PLL bandwidth is 1.8MHz $\sim$ 2.2MHz. In this calculation, the bit depth of the DSM input is 12.

During the measurement phase, the chirp calculation can be used to determine the range of the chirp slope with acceptable linearity. Fig. 3.8 shows the calculation results for a PLL bandwidth of 2.19MHz. The chirp slope (after the $\times$6 frequency multiplier) is swept, and the optimal chirp slope this PLL can support is between 200MHz/μs and 500MHz/μs. When the chirp slope is small, the main source of error is the stair-shaped chirp. When the chirp slope is large, the main source of error is the settling part at the beginning of the chirp.

As mentioned in Chapter 2, the $\phi_e(t)$ data from the calculation results can be utilized in the phase noise calculation to obtain the dynamic phase noise. By comparing the calculated dynamic phase noise with the calculated static phase noise, the dynamic phase noise of a practical chirp generator can be estimated from the static phase noise measurement, which is simpler to conduct.

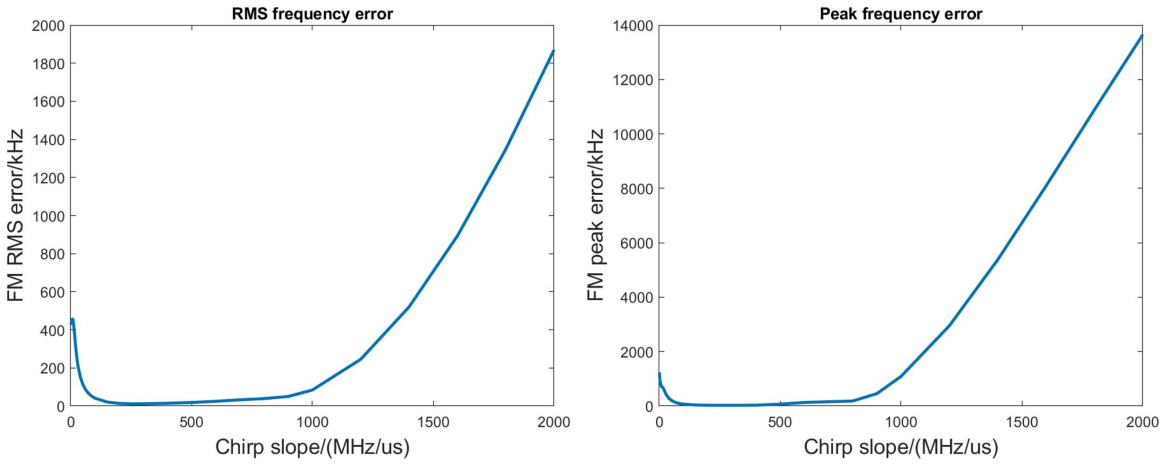


Figure 3.8: Calculation results for PLL with 2.19MHz bandwidth. Both RMS error and peak error are plotted.

3.4 Discussion

In this section, two other modeling efforts will be discussed: one is on s-domain analysis and the other is on the calculus of variations. This section discusses their pros and cons, and suggests directions for future research efforts.

S-domain Analysis

Generally speaking, to analyze loop dynamics, one tool is time-domain differential equations, which was utilized in the previous part of this chapter. Another tool is s-domain analysis, which can greatly simplify the mathematical expression of differential equations. It's also convenient to convert s-domain expressions back to the time domain. Thus, s-domain analysis is widely used in circuit modeling and signal response calculation.

However, in this case, s-domain analysis doesn't have much advantage over time-domain analysis. This is because in FMCW chirp modeling, when focusing on a particular step, for instance from t_k to t_{k+1}, the initial values are not 0. So the s-domain expression isn't as clean and simple compared with differential equations. Some basic formulas of Laplace transform are listed below

$$\mathscr{L}\{f(t)\} = F(s) \tag{3.27}$$
$$\mathscr{L}\{f'(t)\} = sF(s) - f(0) \tag{3.28}$$
$$\mathscr{L}\{f''(t)\} = s^2F(s) - sf(0) - f'(0) \tag{3.29}$$
$$\mathscr{L}\{f'''(t)\} = s^3F(s) - s^2f(0) - sf'(0) - f''(0) \tag{3.30}$$

As the system is not time-invariant across the whole chirp duration, s-domain analysis, like time-domain analysis, can only be applied to one step at a time when the division ratio is fixed. As a result, the initial values at the current step have to be found by calculating all previous steps one by one. In this aspect, s-domain analysis doesn't have an advantage because it can't circumvent the step-by-step calculation in time-domain analysis.

Besides, the sudden changes occurring at step edges inject a step function $-A_k \cdot u(t)$ into $\frac{d\phi_e(t)}{dt}$. The step function is a singularity function that needs to be specially handled in s-domain analysis, which increases the complexity of derivation. But in the time domain, we only need to add $-A_k$ to the final value of $\frac{d\phi_e(t)}{dt}$ from the last step to get its initial value after the sudden change.

However, in FMCW chirp modeling, s-domain analysis has its value. The derivation shown previously in this chapter is based on a 2nd-order low-pass filter. The analytical expressions are not too complicated. But for a higher-order filter, the expressions become much more complicated in the form of differential equations. This requires the help of symbolic computation tools or s-domain analysis, or both.

Calculus of Variations

Calculus of variations is a field of mathematics that studies the minimization or maximization of functionals, which are real-valued functions whose inputs are functions or curves [68][69]. A classical problem in this field is the Brachistochrone problem, the goal of which is to find the curve of fastest descent connecting points A and B in gravity field, given that A and B are in the plane with A lying above B (but not directly above B). The curve is the independent variable or input of the function, and the travel time is the dependent variable or output of it. The calculus of variations is a powerful tool to solve such problems.

For FMCW chirp design and optimization, the calculus of variations can also be helpful. In the step shown in Fig. 3.3, a functional can be defined as

$$D_k(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1) = \int_{t_k}^{t_{k+1}} \left[f_{ideal}(t) - f_{out}(t)\right]^2 \, dt \tag{3.31}$$

This quantifies the deviation of $f_{out}(t)$ from $f_{ideal}(t)$. The optimization goal is to minimize the total deviation

$$D_{total}(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1) = \frac{1}{T_{chirp}} \sum_{k=0}^{SN-1} D_k \tag{3.32}$$

where SN is the total number of steps. If linearity is the only optimization goal, then a constant offset frequency Δf is allowed between practical and ideal chirps. In this case, the

functional of step $t_k \sim t_{k+1}$ is defined as

$$d_k(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1) = \int_{t_k}^{t_{k+1}} [f_{ideal}(t) - \Delta f - f_{out}(t)]^2 \, dt \tag{3.33}$$

and the corresponding summation is

$$d_{total}(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1) = \frac{1}{T_{chirp}} \sum_{k=0}^{SN-1} d_k \tag{3.34}$$

Applying the calculus of variations can lead to optimal design and it can also provide some insight into the PLL. However, to make the problem solvable, the analytical expression of $f_{out} = f_{out}(t, K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1)$ is required. This is probably possible for a PLL with a first-order filter, but in practical design, 2nd- or higher-order filters are used. Thus, analytical expression of $D_k(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1)$ and $d_k(K_{VCO}, I_{CP}, a_1, a_2, b_0, b_1)$ will be very complicated. Besides, the nonzero initial values of each step will make it even more complicated, leading to exponentially higher computation cost during optimization. In other words, much of the issue of applying this method originates from the chirp's nature that it is not one uniform chirp, but many segments that are dependent on each other.

Based on the discussion of s-domain analysis and calculus of variations, the time-domain numerical calculation might be the most suitable method of optimizing the chirp linearity. It does require some iterations tuning PLL parameters, to balance phase noise, chirp linearity, and some other metrics. But it's the most flexible and fastest method, considering generality and computational cost. However,research interest could be directed towards s-domain analysis or calculus of variations, if researchers figure out a way to model the chirp as a whole. The switching division ratio might be able to be approximated as or equivalent to an input signal, so that the system is considered as time-invariant. Then, those two methods would be powerful. And calculus of variations can even directly provide the optimal values for K_{VCO}, I_{CP}, etc.

3.5 Conclusion

In this chapter, we proposed a more accurate time-domain model for PLL bandwidth design. This model is based on the step-wise time-invariant nature of the chirp generator. The time-domain method might offer the best balance between accuracy and computational cost, compared to circuit transient simulation and other modeling approaches. To the best of the author's knowledge, such a design and optimization method has never been proposed or published before. The conventional method can only provide a lower limit of $\frac{1}{T_{chirp}}$ and an upper limit of BW_{settle} for PLL bandwidth design. In contrast, the proposed method provides more comprehensive guidance on PLL bandwidth. And further research suggestions are provided.

Chapter 4

Frequency Division

The frequency division of the chirp generator involves the design of a dual-modulus divider and the generation of its control. The fractional-N division ratio is achieved by alternating the division ratio between two integers, resulting in an average division ratio that is a fractional number. This necessitates a rapid switch in the division ratio by the divider. The operating principle increases the phase noise, so an appropriate switching scheme must be chosen. This chapter will present the design of the dual-modulus divider and the Delta-Sigma Modulator that controls it.

4.1 Dual-Modulus Divider

In proposed design shown in Fig. 4.1, the FMCW bandwidth is 8GHz and the frequency of FMCW chirp ranges from 138GHz to 146GHz, which is implemented with the generated chirp from 23GHz to 24.33GHz, followed by a $\times 6$ frequency multiplier. The reference frequency is 2.875GHz, so the division ratio of the divider is from 8 to 8.4638.

As shown in Fig. 4.2, this division ratio range can be implemented by a divide-by-8/9 divider, which consists of a divide-by-2/3 prescaler followed by two divide-by-2 dividers in cascade. In Fig. 4.2(a), the MC (modulus control) signal controls the division ratio. When $MC = 1$, output of the OR gate is always 1 and the second-stage DFF operates as a divide-by-2 divider. When $MC = 0$, output of the first stage turns to 0 every two clock cycles, swallowing one clock cycle from the second stage, resulting in a divide-by-3 divider. In Fig. 4.2(b), when $MC = 0$, the divider works in divide-by-8 mode because the divide-by-2/3 divider always works in divide-by-2 mode. When $MC = 1$, the MC terminal of the divide-by-2/3 divider turns to 0 every 8 clock cycles, making the divide-by-2/3 divider works as a divide-by-3 divider for one clock cycle, resulting in divide-by-9 operation.

However, such 8/9 divider is not fast enough. The proposed design, which uses TSMC 28nm technology, has a maximum input frequency for the 2/3 prescaler of 48.5GHz in divide-

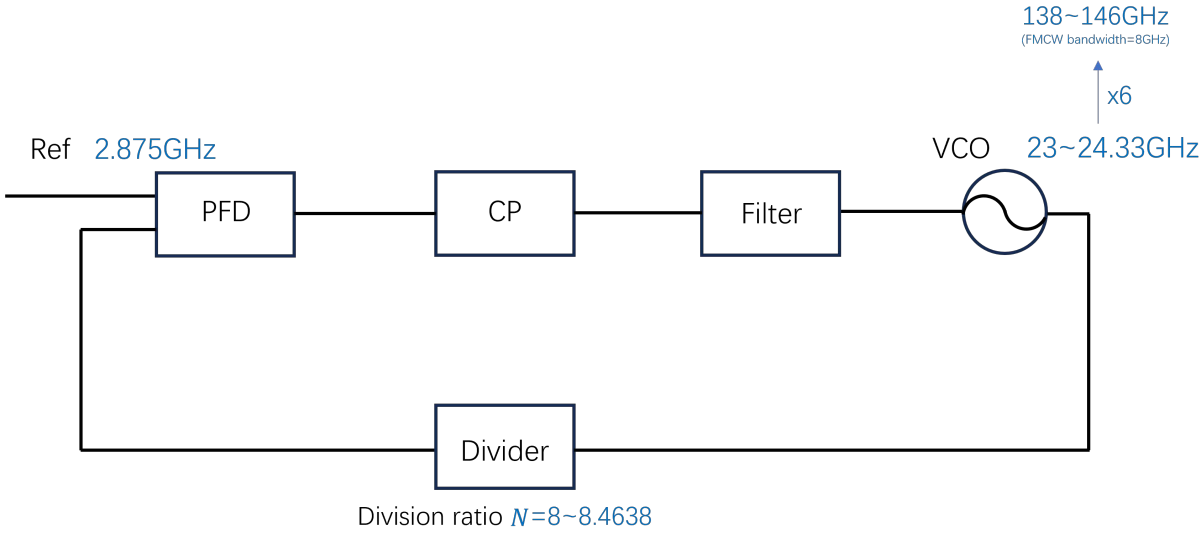


Figure 4.1: Chirp frequency and division ratio.

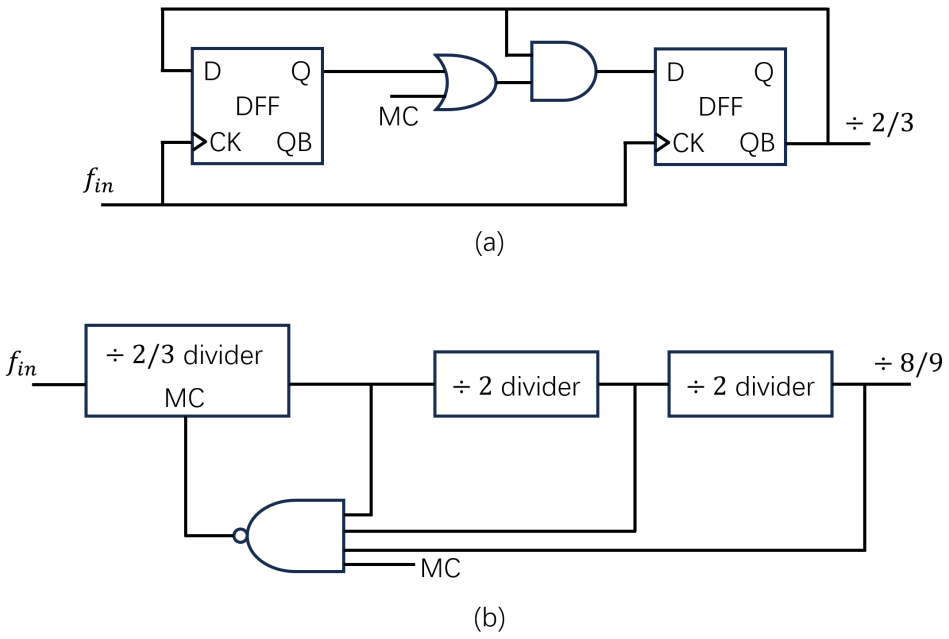


Figure 4.2: (a) A divide-by-2/3 prescaler. (b) A 8/9 dual-modulus frequency divider.

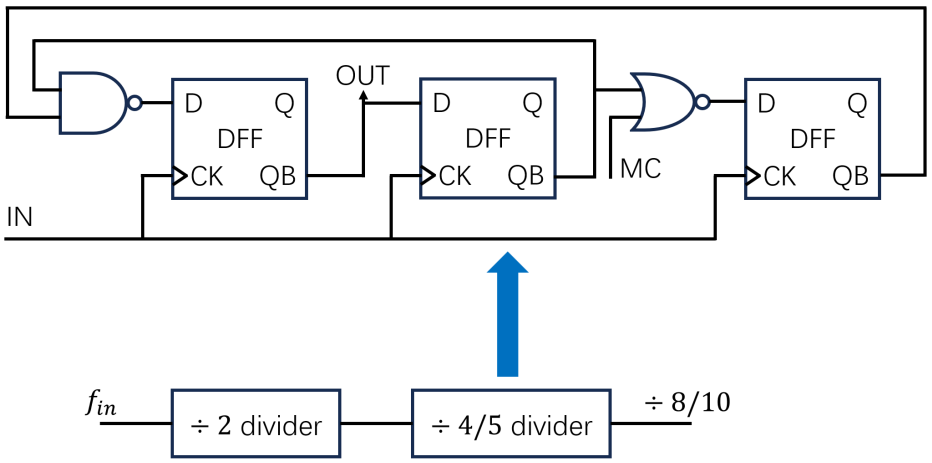


Figure 4.3: The block diagram of divide-by-8/10 divider and schematic of the 4/5 divider.

by-2 mode and 46GHz in divide-by-3 mode. The 8/9 divider, however, is an asynchronous circuit. For it to operate correctly, the output of the NAND gate needs to remain at 1 for 8 clock cycles and then switch to 0 before the next clock edge arrives. The total delay, which includes 3 clock-to-Q delays and the NAND delay, makes it difficult to operate at high frequencies. According to simulations, the divide-by-9 mode fails to work when the input exceeds 15GHz, a frequency much lower than the VCO output frequency.

As an alternative, a divide-by-8/10 divider is used in the proposed design, consisting of a divide-by-2 prescaler followed by a divide-by-4/5 divider. The division ratio range, 8/10, is doubled compared to that of the 8/9 divider, resulting in a loss of 1-bit resolution. Consequently, in the control signal design, the input of the control circuit should be 1-bit longer, which will be presented in detail in the following section. Fig. 4.3 shows the schematic of the 8/10 divider. The divide-by-2 prescaler significantly reduces the design requirements of the 4/5 divider, as the input frequency is around 12GHz. Besides, the 4/5 divider is a synchronous circuit with 2-input logic gates, allowing it to operate at high frequency.

Schematic simulation shows the maximum operating frequency of the 8/10 divider is 60GHz. And its layout is shown in Fig. 4.4.

4.2 Delta-Sigma Modulator

The 8/10 divider needs to be controlled so that its division ratio alternates between 8 and 10 to achieve a non-integer average division ratio. Fig. 4.5 presents two methods for implementing the fractional-N division ratio. Fig. 4.5(a) is the most basic architecture. It

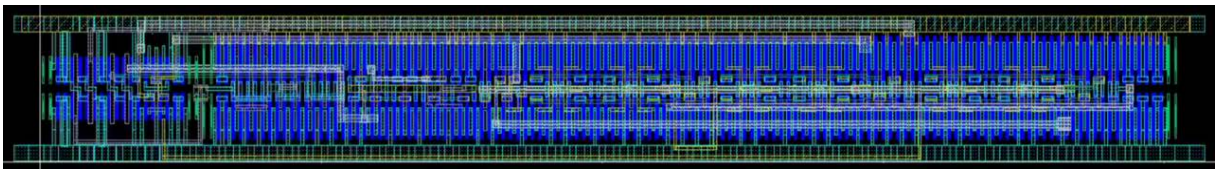


Figure 4.4: Layout of the divide-by-8/10 divider.

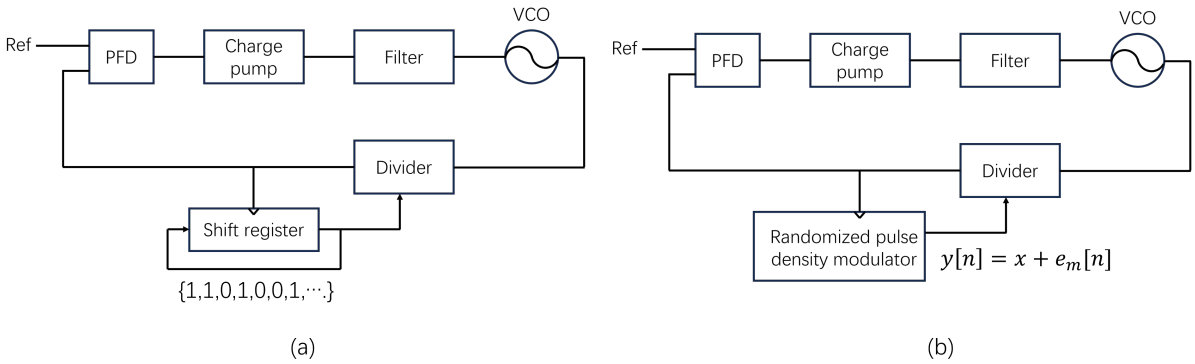


Figure 4.5: Implementations of fractional-N division ratio.

stores the controls bits in a shift register. This architecture is simple, but the difference between the actual divider modulus and their average is a periodic sequence, resulting in spur tone issues. Fig. 4.5(b) randomizes the sequence, where x is the fractional part of the modulus and $e_m[n]$ is the zero-mean quantization noise. This scheme results in $e_m[n]$ being a white noise. However, the noise at low frequency is still high for some applications. To lower the phase noise, the Delta-Sigma Modulator (DSM) is applied in the proposed design (Fig. 4.6). The DSM can generate the sequence of modulus such that most of the noise power is well above the desired bandwidth of PLL. The high-frequency noise is suppressed by the loop filter, improving the noise performance [70].

Division Ratio Quantization

To control the divide-by-8/10 divider presented in the previous section, a DSM with a 1-bit output is applied, as illustrated in Fig. 4.7. From a circuit perspective, the fractional part (a fractional number between 0 and 0.4638) is fed into the DSM, which generates the control bit sequence to control the 8/10 divider. This causes the division ratio to alternate between 8 and 10. Logically, this process is equivalent to feeding a sequence of $x[n]$ ($x[n]$ can be 0 or 2) into a divider, where the division ratio is $(8 + x[n])$. For the convenience of DSM coding, all numbers are converted to signed numbers. A divider with a division ratio

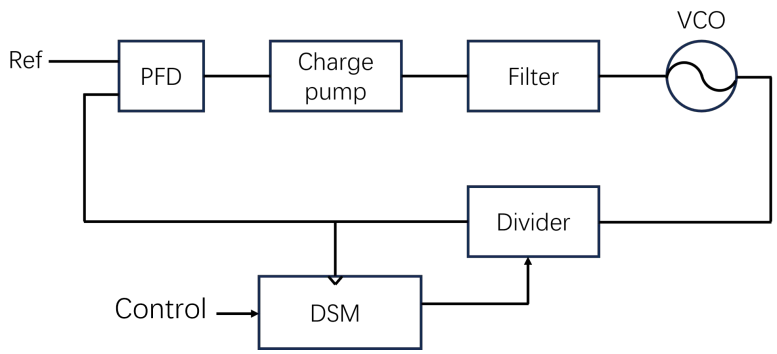


Figure 4.6: Chirp generator with a Delta-Sigma Modulator.

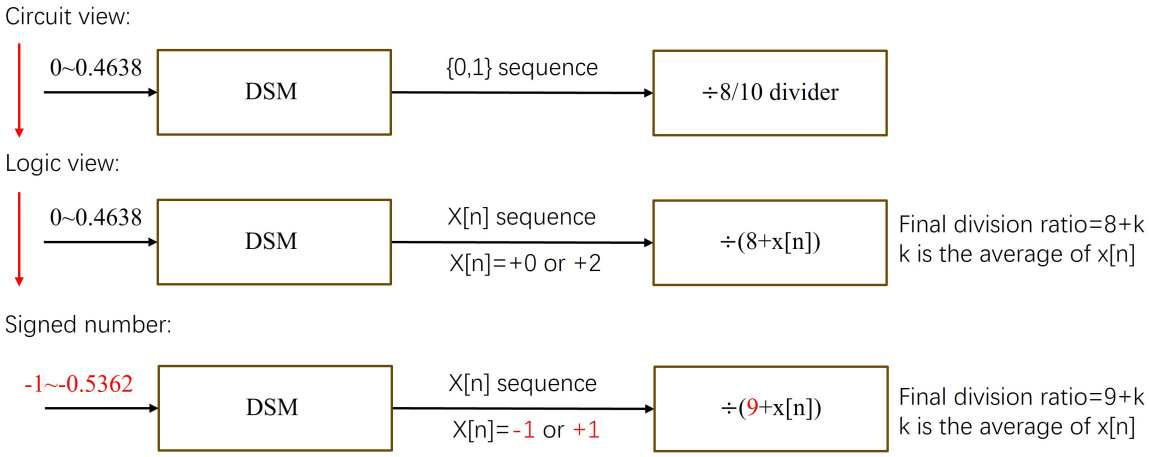


Figure 4.7: 1-bit DSM controls the division ratio of the divide-by-8/10 divider.

of $(9 + x[n])$ is controlled by a sequence of -1 and 1. Correspondingly, the fractional part of the division ratio, or the input of the DSM, ranges from -1 to -0.5362.

For quantization, the fractional part k is quantified as $q_k = floor\left(k \cdot 2^{bit\_in-1}\right)$, where bit_in is the number of DSM input bits, and the function $floor$ rounds the variable to the nearest integer less than or equal to it. The range is as follows

$$-1 \leq k < 1 \tag{4.1}$$

$$-2^{bit\_in-1} \leq q_k < 2^{bit\_in-1} \tag{4.2}$$

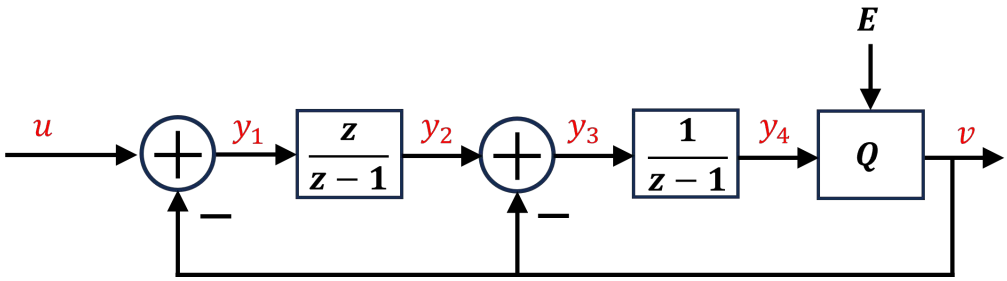


Figure 4.8: Block diagram of the 2nd-order 1-bit output DSM.

FMCW radar parameters can be calculated. The Frequency resolution f_{res}, total number of steps N_{step}, and step time T_{step} can be expressed in terms of bit_in

$$f_{res} = \frac{f_{ref}}{2^{bit\_in-1}} \tag{4.3}$$

$$N_{step} = floor\left[(N_1 - 9) \cdot 2^{bit\_in-1}\right] - floor\left[(N_0 - 9) \cdot 2^{bit\_in-1}\right] + 1 \tag{4.4}$$

$$T_{step} = \frac{T_{chirp}}{N_{step}} \tag{4.5}$$

where N_0 is the division ratio at the beginning of the chirp ($N_0 = 8$) and N_1 is the one at the end of the chirp ($N_1 = 8.4638$).

Implementation of DSM

Fig. 4.8 shows the block diagram of the 1-bit output DSM. The signal transfer function and noise transfer function are

$$TF_{signal} = \frac{1}{z} \tag{4.6}$$

$$TF_{noise} = \frac{1}{\left(1 - \frac{1}{z}\right)^2} \tag{4.7}$$

The DSM is implemented in Verilog, as shown below. The number of input bits is determined by the linearity calculation. The number of extension bits, denoted as bit_ext, is selected to ensure that the internal signals do not overflow.

```
1 `timescale 1ns / 1ps
2 module  DSM #(
3     parameter bit_in=12,
4     parameter bit_ext=11
5 )(
```

```
6       input   [bit_in-1:0]  u ,
7       input                 clk ,
8       input                 rstn,
9       output                v
10  );
11      reg     [bit_in+bit_ext-1:0]  int1;
12      reg     [bit_in+bit_ext-1:0]  int2;
13      reg                           v_reg;
14      wire    [bit_in+bit_ext-1:0]  y1;
15      wire    [bit_in+bit_ext-1:0]  y2;
16      wire    [bit_in+bit_ext-1:0]  y3;
17      wire    [bit_in+bit_ext-1:0]  y4;
18      wire    [bit_in+bit_ext-1:0]  feedback;
19      assign  y1       =   $signed(u)-$signed(feedback);
20      assign  y2       =   $signed(int1)+$signed(y1);
21      assign  y3       =   $signed(y2)-$signed(feedback);
22      assign  y4       =   $signed(y3)+$signed(int2);
23      assign  feedback=    ~v_reg?($signed(-1)<<<(bit_in-1)):
24                           ($signed(1)<<<(bit_in-1));
25      always  @(posedge clk )  begin
26          if(!rstn)
27              int1     <=  'd0;
28          else
29              int1     <=  y2;
30      end
31      always  @(posedge clk )  begin
32          if(!rstn)
33              int2     <=  'd0;
34          else
35              int2     <=  y4;
36      end
37      always  @(posedge clk )  begin
38          if(!rstn)
39              v_reg    <=  1'b1;
40          else
41              v_reg    <=  ~y4[bit_in+bit_ext-1];
42      end
43      assign  v=v_reg;
44  endmodule
```

The Verilog code can be inserted into Cadence Virtuoso to run transient simulations with other circuits blocks. However, during the design phase, it's faster to debug and tune parameters of the DSM in MATLAB. The corresponding code is provided below. The MATLAB

code is equivalent to the Verilog code.

```
%% 2nd-order Delta-Sigma Modulator operation
y1(1)=u(1)-v(1)*2^(bit_in-bit_out);
y3(1)=y2(1)-v(1)*2^(bit_in-bit_out);
for k=2:length(t)
    y4(k)=y4(k-1)+y3(k-1);
    % quantizer / comparator
    if y4(k)<0
        v(k)=-1;
        v_real(k)=0;
    else
        v(k)=1;
        v_real(k)=2;
    end
    if k>=aver_bit_N
        aver_v(k-aver_bit_N+1)=sum(v(k-aver_bit_N+1:k))/
           aver_bit_N*2^(bit_in-bit_out);   % Note: to compare
           the average output with input, it's multipled by 2^(
           bit_in-bit_out)
        aver_v_real(k-aver_bit_N+1)=sum(v_real(k-aver_bit_N+1:k
           ))/aver_bit_N;                  % the logic average
           output next stage will see
    end
    y1(k)=u(k)-v(k)*2^(bit_in-bit_out);
    y2(k)=y1(k)+y2(k-1);
    y3(k)=y2(k)-v(k)*2^(bit_in-bit_out);
end
```

The code for simulating the DSM is presented in Appendix C.

4.3 Simulation Results

The DSM is simulated and verified using three types of test: the DC test, the sine-wave test, and the ramp test, both with MATLAB and in Cadence Virtuoso.

DC Test

A DC test involves fixing the input of the DSM, calculating the average value of the output, and checking if it equals the input. Fig. 4.9 presents the simulation results. The input is set as -0.2345. The corresponding quantified input is -480, and the average of the output

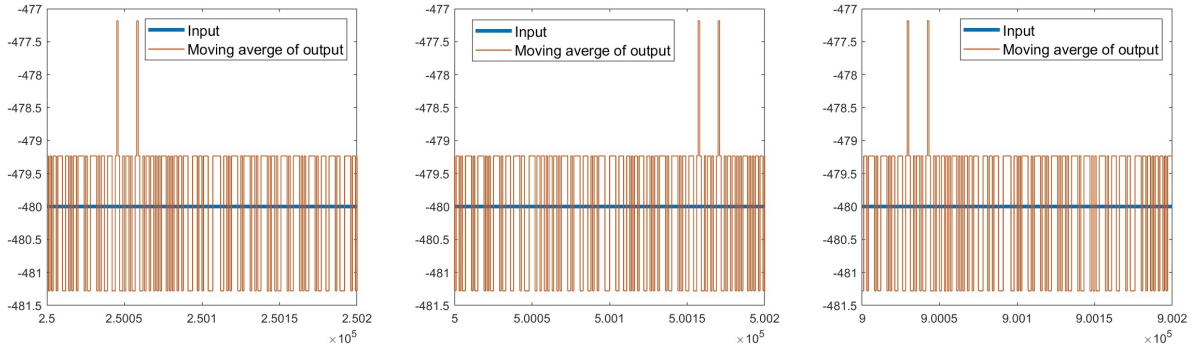


Figure 4.9: Simulation results for DC input.

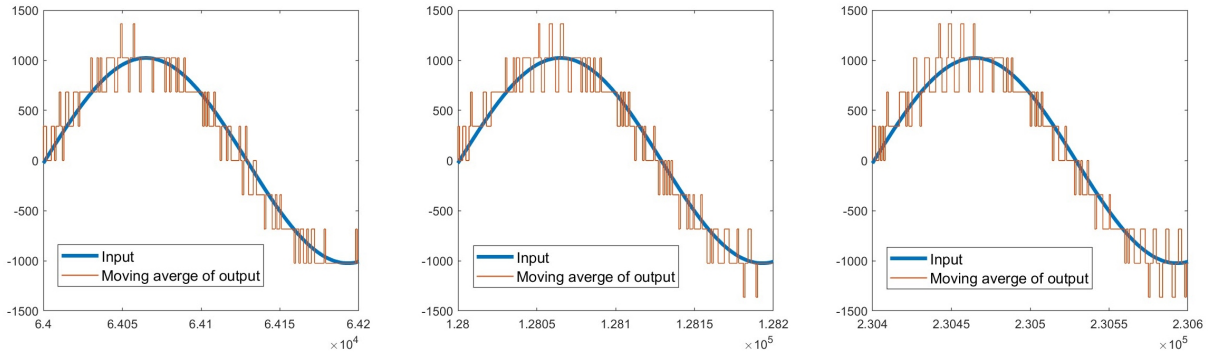


Figure 4.10: Simulation results for sine-wave input.

signal is −481.28. In Fig. 4.9, the input is compared to the moving average of the output, which is the average value of the latest 2000 outputs. The number of recent outputs used to calculate the moving average can be set to other values. This test verifies the functionality that the average value of the DSM output equals the input, even though the output is 1-bit long. Other input values can also be tested. For instance, when the input is −0.5362, the quantified input is −1098, and the average output is −1097.728.

Sine-wave Test

A sine-wave test is used to verify the noise shaping function by applying a sine-wave input. Fig. 4.10 displays the simulation results. In this instance, a 300kHz sine-wave is applied. The average output follows the input signal. Similar to the DC test, the moving average only calculates the average value for the most recent 2000 outputs. The spectrum of the output signal is calculated and depicted in Fig. 4.11. The noise shaping effect is clearly visible.

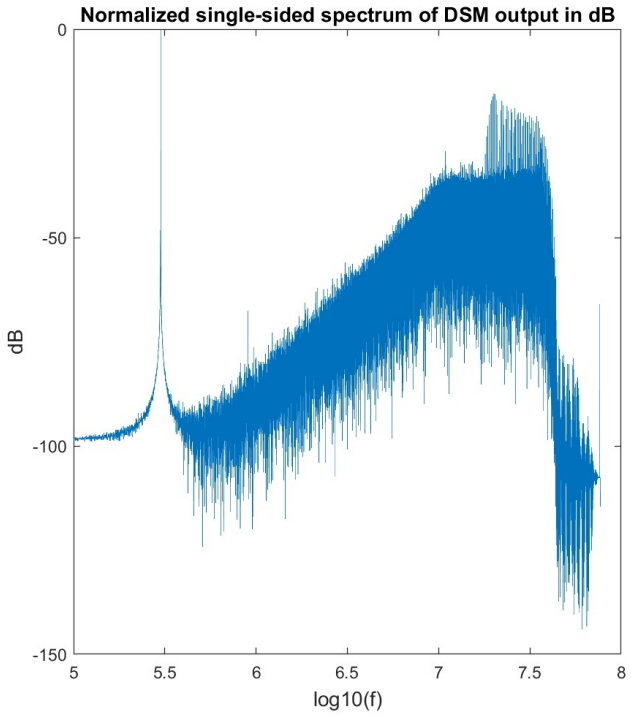


Figure 4.11: Output spectrum with a sine-wave input.

Ramp Test

In the ramp test, a ramp input is applied to simulate the chirp generation operation. The moving average of the output should follow the input ramp. Fig. 4.12 shows the simulation results. In this case, the chirp slope is set to 10MHz/μs. At each step, the input switches to a new value, and the output follows it. The division ratio is expected to start from 8 and end at 8.4638. The simulation results align with the design. For this test, the moving average is calculated from the most recent 500 outputs instead of 2000, as there are only approximately 2400 sampling points at each step.

Verification in Cadence Virtuoso

The sine-wave test and the ramp test are conducted in Cadence Virtuoso. Fig. 4.13 and Fig. 4.14 show their simulation setup respectively. The simulation results match the ones in MATLAB.

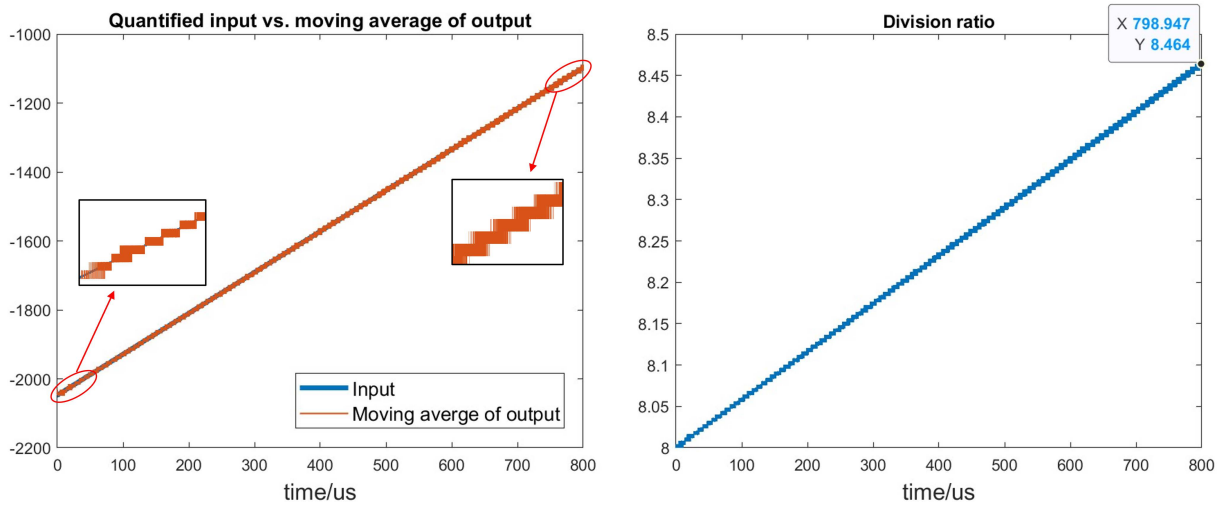


Figure 4.12: Simulation results for the ramp test.

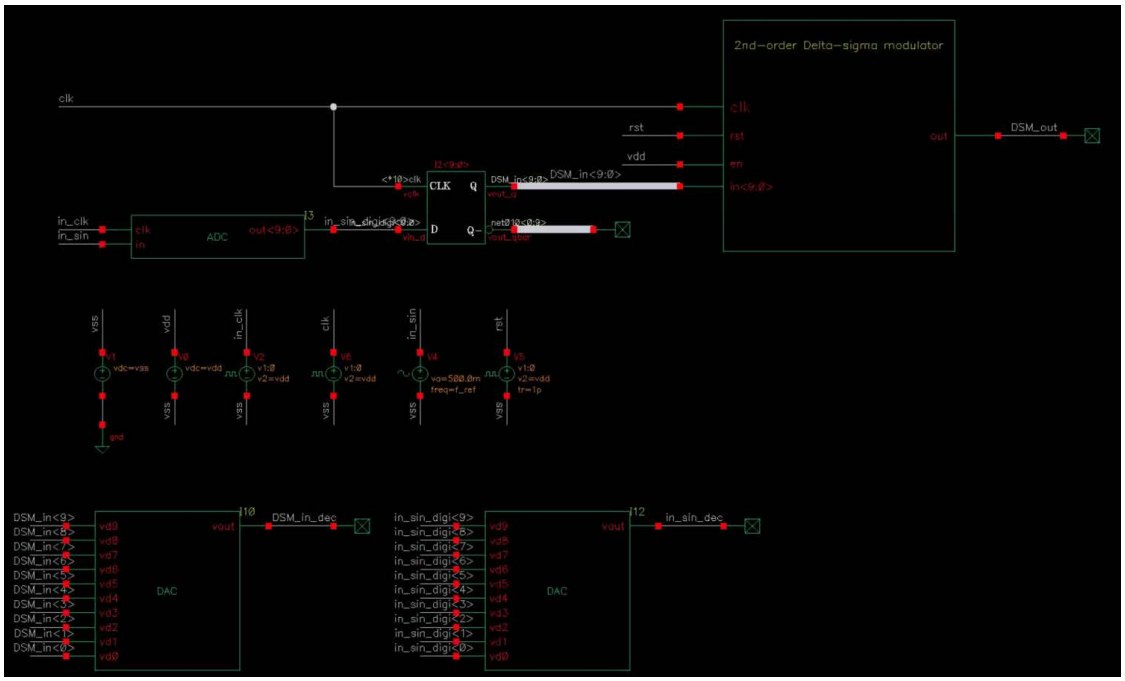


Figure 4.13: Simulation setup for the sine-wave test.

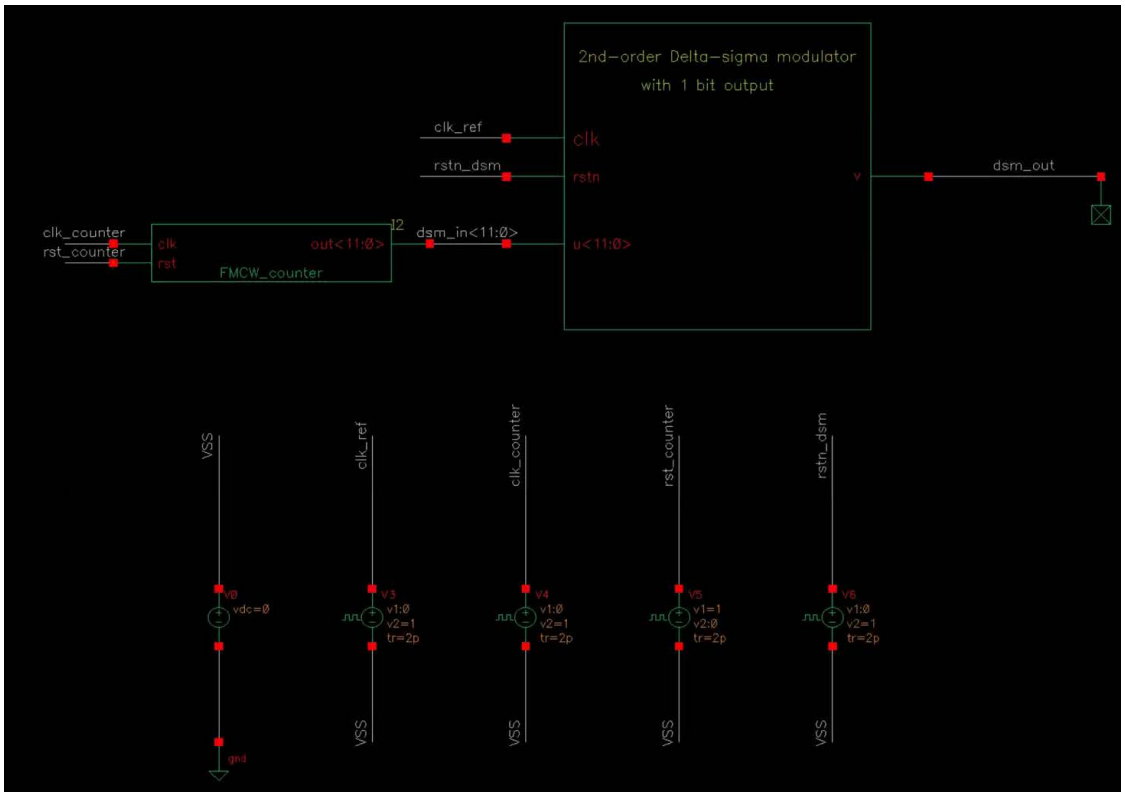


Figure 4.14: Simulation setup for the ramp test.

4.4 Conclusion

As a crucial part of the fractional-N PLL, the frequency division modules are discussed in this chapter. A divide-by-8/10 divider is chosen, and a DSM is used to control it, thereby implementing the fractional division ratio. The DSM is simulated and verified using both MATLAB and Cadence Virtuoso.

Chapter 5

High-Frequency Circuit Design and Modeling

5.1 Introduction

Modern integrated circuitry utilize high-speed Complementary Metal-oxide Semiconductor (CMOS) devices as digital logic elements extensively, including frequency dividers [71], as latches and adders/multipliers in high-speed links, and in high-speed mixed-signal circuits. Increasingly, analog and even RF circuits utilize digital logic gates as high speed building blocks in digital friendly approaches to realize traditionally all analog transceivers [72], power amplifier [73], and frequency synthesizers [74].

In the proposed PLL, there are circuit blocks like prescaler and divider that operate at high frequencies. In order to design and optimize high-speed digital circuitry, one requires an accurate analytical and predictable delay model. While propagation delay models for both sub-threshold and super threshold region of MOSFET operation are widely known [1][75] – a void still exists for a universal physics based delay model valid for both sub-threshold as well as super threshold regions. In many delay models the approach is heavily empirical fitting based, raising concern for accuracy and consistency under diversified operating conditions [1][76][77]. A plethora of logic issues has been addressed in literature to model the sub-threshold or super-threshold propagation delay, yet only a few put forth a physics based model [78]. Most importantly, to the best of our knowledge, there is no reported modeling work on the impact of bias dependent mobility over the propagation delay – although the direct influence of effective mobility on the drain current and its critical dependency on applied voltage and temperature [79] are well studied. So from the designers perspective, a universal physics based delay model coupled with a simple yet accurate PVT dependent mobility equation is of cardinal importance.

Within a vast PVT range, an accurate propagation delay model ensures correct estimation of the circuit's maximum operating frequency, during the circuit design phase itself. It is well known that propagation delay is highly sensitive to factors like input transition time (rise time and fall time), device dimensions and PVT variations [75][77]. The present work furnishes a complete physics based universal delay model which handles all these factors along with a novel mobility and stacking model, while being easily translatable to MOSFET based advanced devices like NCFET [80]. Grounded on the charge based EKV MOSFET model[79], a custom charge based delay modeling approach is adopted all-throughout. Another salient reason behind choosing a charge based approach is that, unlike threshold voltage based model (e.g. BSIM 1 to 4 [81]), it has a single drain current equation valid for all regimes. Extensive simulations have been performed using commercial foundry 22nm, 28nm CMOS processes along with UC Berkeley's in-house 90nm NCFET process (BSIM4.5 extraction[81]) in transient simulation platform to ensure the validity of the proposed model.

This chapter is divided into six sections, including the introduction. Section II describes the analytical proposed universal charge based propagation delay model, derived on a single MOSFET with a capacitive load. The next section formulates the stacking effect of several MOSFETs, in charge based approach. The fourth section describes the bias dependent analytical mobility model, which is an unique contribution of this work in the domain of digital / mixed signal circuits. In the fifth section the combined model is validated for an array of scenarios ranging from single transistor including transistor stacks and a ring oscillator. The sixth section concludes the chapter.

5.2 Analytical Delay Model Derivation

As discussed in the introduction, we strive for establishing a charge based process independent generalised analytical delay model for CMOS based circuits. The modeling methodology is based on primarily working out the delay model for a single N-type MOSFET with a single lumped capacitive load at drain (involving intrinsic Miller capacitance), as depicted in Fig. 5.1.

The drain to source current (I_{DS}) for an NMOSFET is given by EKV model as [79],

$$I_{DS} = 2m\mu_{eff}C_{ox}(W/L)\phi_t^2(i_f - i_r) \tag{5.1}$$

Here m = sub-threshold slope factor, μ_{eff} = effective mobility, C_{ox} = oxide capacitance per unit area, $W(L)$ = width(length) of the MOSFET, $\phi_t = kT/q$ = thermal voltage, $i_{f(r)}$ = MOSFET normalized source(drain) current (see Appendix D) [79]. In Eq. 5.1 the quantities m and $i_{f(r)}$ are only dependant on input gate-to-source voltage (V_g), whereas μ_{eff} is dependant both on V_g and drain-to-source or the output voltage (V_{out}) – and their expressions are listed in Appendix D.

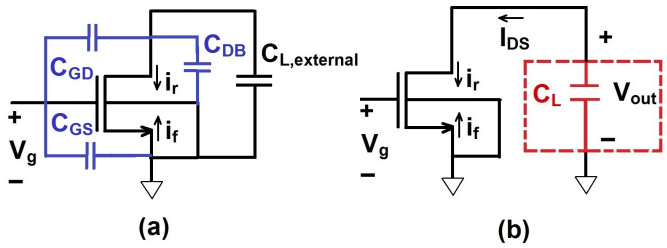


Figure 5.1: Normalized forward and reverse terminal currents in NMOS transistor with (a) intrinsic capacitance network and external capacitance load, (b) lumped equivalent output load capacitance (C_L).

In Fig. 5.1, if the load C_L is to discharge through the NMOSFET, when V_g makes a low-to-high transition (0 to V_{DD}) – then V_{out} swings from V_{DD} to 0. So by definition, the 'high-to-low' propagation delay (t_{PHL}) is the gap between the time instants when $V_g = V_{DD}/2$ and $V_{out} = V_{DD}/2$ [1]. So in order to work out a process-independent, analytical and fully portable model of t_{PHL}, we need to effectively focus only on the input and output ranges of ($V_{DD} \geq V_g \geq V_{DD}/2$) and ($V_{DD} \geq V_{out} \geq V_{DD}/2$) respectively. This simple observation enables us to make several important approximations in the t_{PHL} modeling approach:

1. As within the range $V_{DD} \geq V_g, V_{out} \geq V_{DD}/2$ the charge contribution of the source terminal is always much more than that of drain terminal – we neglect i_r in Eq. 5.1.

2. The bias-dependent sub-threshold slope factor m is replaced by the near Fermi-level approximated sub-threshold slope factor m_0.

3. Instead of using the effective mobility μ_{eff} as function of bias voltages V_g and V_{out}, we use $\mu_{eff} = \mu_{eff}^{(t_r)}$ as function of input voltage slope (rise time) t_r. This way μ_{eff} becomes independent of time, and easier to handle.

Hence after applying the above approximations in Eq. 5.1 we arrive at,

$$I_{DS} = 2m_0\mu_{eff}^{(t_r)}C_{ox}(W/L)\phi_t^2 i_f \tag{5.2}$$

We compare Eq. 5.2 with the drain current equation Eq. 5.1 for $V_{DD}/2 \leq V_g, V_{out} \leq V_{DD}$ through Fig. 5.2 – which ratifies our approximation.

At the drain terminal of the NMOSFET in Fig. 5.1,

$$C_L\frac{dV_{out}}{dt} = -K_n^{(t_r)}i_f \tag{5.3}$$

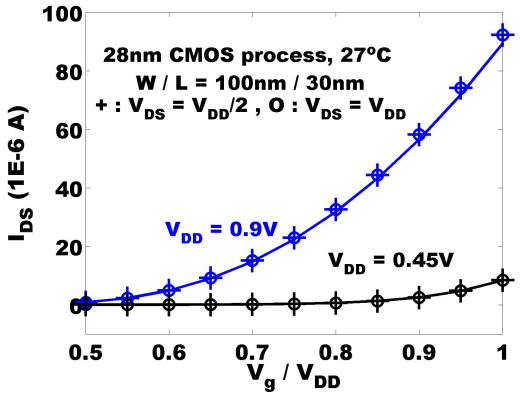


Figure 5.2: Comparison of approximated drain current model Eq. 5.2 (symbols) with exact drain current model Eq. 5.1 (lines).

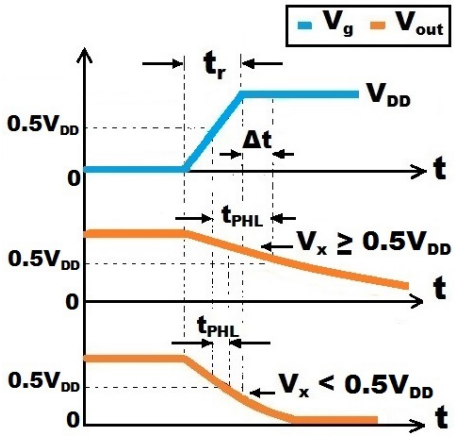


Figure 5.3: Input and output waveform schematic.

where $K_n^{(t_r)} = 2m_0\mu_{eff}^{(t_r)}C_{ox}(W/L)\phi_t^2$ is the trans-conductance parameter. As i_f is a function of V_g as given in Eq. D-1(in Appendix D) – which is in-turn a function of time (t). From Fig. 5.3, V_g can be defined as,

$$V_g(0 \leq t \leq t_r) = V_{DD}(t/t_r) \tag{5.4}$$

$$V_g(t > t_r) = V_{DD} \tag{5.5}$$

To calculate the time difference in between the instances when $V_g = V_{DD}/2$ and $V_{out} = V_{DD}/2$ (or the propagation delay t_{PHL}), we first need to find $V_x = V_{out}$ at $t = t_r$ Fig. 5.1.

Again from Eq. D-1 it can be noted that i_f is a function of v_p (which again is a function of V_g). Hence changing variable from t to v_p (expression given in Eq. D-3) and solving Eq. 5.3 for V_x we get,

$$V_x = V_{DD} - A_{V_{DD}} K_n^{(t_r)} t_r / C_L \Delta v_p \tag{5.6}$$

$$A_{V_{DD}} = \frac{1}{12}\boldsymbol{W_l}^3(2e^{v_{p(V_{DD})}}) + \frac{3}{8}\boldsymbol{W_l}^2(2e^{v_{p(V_{DD})}}) + \frac{1}{2}\boldsymbol{W_l}(2e^{v_{p(V_{DD})}}) \tag{5.7}$$

where $\Delta v_p = v_{p(V_{DD})} - v_{p(0)}$, $v_{p(V_{DD}(0))} = v_p$ at $V_g = V_{DD}(0)$

It logically follows that there might arise two cases (shown in Fig. 5.3) when,

1. $V_x \geq V_{DD}/2$ or when V_{out} is yet to arrive at $V_{DD}/2$ within $0 \leq t \leq t_r$. So V_{out} discharges from V_x up to $V_{DD}/2$ within $t_r < t \leq t_r + \Delta t$ (Fig. 5.3) – when V_g has stabilized Eq. 5.5. Thus using Eq. 5.3 and Eq. 5.5,

$$C_L \int_{V_x}^{V_{DD}/2} dV_{out} = -K_n^{(t_r)} i_{f(V_{DD})} \int_{t_r}^{t_r+\Delta t} dt \tag{5.8}$$

   Here $i_{f(V_{DD})} = i_f$ at $V_g = V_{DD}$ (D-1). Hence solving Eq. 5.8 for Δt,

$$\Delta t = \frac{C_L(V_x - V_{DD}/2)}{K_n^{(t_r)} i_{f(V_{DD})}} \tag{5.9}$$

   So with reference to Fig. 5.3 the delay model for $V_x \geq V_{DD}/2$ condition is given by,

$$t_{PHL} = \frac{t_r}{2} + \Delta t = \frac{t_r}{2} + \frac{C_L(V_x - V_{DD}/2)}{K_n^{(t_r)} i_{f(V_{DD})}} \tag{5.10}$$

2. $V_x < V_{DD}/2$ or when V_{out} has already discharged beyond $V_{DD}/2$ within $0 \leq t \leq t_r$. So in this case we need to solve the governing differential equation Eq. 5.3 directly to obtain t_{PHL} as,

$$C_L \int_{V_{DD}}^{V_{DD}/2} dV_{out} = -K_n^{(t_r)} \int_0^{\tau} i_f dt \tag{5.11}$$

   Here $\tau = t_r/2 + t_{PHL}$. Changing variable to v_p from t in Eq. 5.11 and solving

$$C_L V_{DD} \Delta v_p / t_r K_n^{(t_r)} = \frac{1}{6}\boldsymbol{W_l}^3(2e^{v_{p(\tau)}}) + \frac{3}{4}\boldsymbol{W_l}^2(2e^{v_{p(\tau)}}) + \boldsymbol{W_l}(2e^{v_{p(\tau)}}) \tag{5.12}$$

   Here $v_{p(\tau)} = v_{p(0)} + \tau \Delta v_p / t_r$. Solving Eq. 5.12 gives

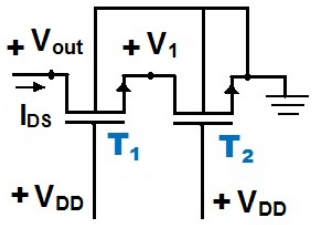


Figure 5.4: NMOS transistor stacking.

$$t_{PHL} = \frac{t_r}{\Delta v_p}[log_e\beta_r + 2\beta_r - v_{p(0)}] - \frac{t_r}{2} \tag{5.13}$$

where

$$\beta_r = \frac{1}{4}(\theta_r + 1/\theta_r - 3) \tag{5.14}$$

$$\theta_r = (4\sqrt{36a_r^2 + 27a_r + 5} + 24a_r + 9)^{1/3} \tag{5.15}$$

$$a_r = C_L V_{DD}\Delta v_p / t_r K_n^{(t_r)} \tag{5.16}$$

Thus the equations Eq. 5.10 and Eq. 5.13 depict the analytical model for t_{PHL}.

5.3 Transistor Stacking

In the previous section, we discussed the analytical delay model for a single NMOS transistor with a capacitive load. In real world digital circuits, we frequently encounter transistor stacking in various digital building blocks e.g. gates, latches etc. In this work we demonstrate the model using a 2-transistor stack as in Fig. 5.4, which can be easily extended to the generalised N-transistor case.

Referring to Fig. 5.4 and using Eq. 5.1 we can write the expression of drain current I_{DS} of the NMOS stack arrangement as follows

$$I_{DS} = K_{n,T_2}^{(t_r)}[(q_{i,1}^2 + q_{i,1}) - (q_{i,out}^2 + q_{i,out})] = K_{n,T_1}^{(t_r)}[(q_{i,Gnd}^2 + q_{i,Gnd}) - (q_{i,1}^2 + q_{i,1})] \tag{5.17}$$

Here $K_{n,T_{1(2)}}^{(t_r)} = K_n^{(t_r)}$ for transistor $T_{1(2)}$ and $q_{i,1(Gnd)(out)}$ = Normalized inversion charge Eq. D-2 at V_1(Ground)(V_{out}). Performing $I_{DS}/K_{n,T_1}^{(t_r)} + I_{DS}/K_{n,T_2}^{(t_r)}$ on Eq. 5.17 yields

$$I_{DS} = K_{n,equiv}^{(t_r)}\left[(q_{i,Gnd}^2 + q_{i,Gnd}) - (q_{i,out}^2 + q_{i,out})\right] \tag{5.18}$$

Here $K^{(t_r)}_{n,equiv} = (1/K^{(t_r)}_{n,T_1} + 1/K^{(t_r)}_{n,T_2})^{-1}$ which is the equivalent trans-conductance parameter for the stacked 2-NMOSFET arrangement (Fig. 5.4). Evidently this can be easily extended generalised for N-stack case as

$$K^{(t_r)}_{n,equiv} = [\sum_{i=1}^{N} 1/K^{(t_r)}_{n,T_i}]^{-1} \tag{5.19}$$

So by substituting Eq. 5.19 in Eq. 5.6 - Eq. 5.13, we may get the delay model for transistor stacking arrangement.

5.4 Modeling the Effective Mobility

The existing models of effective mobility (μ_{eff}) tackles the PVT variation of μ_{eff} through semi-empirical functions of electric field [79][81][82] . This approach poses enormous challenge to model t_{PHL} – because the non-linear electric field terms become tough to manipulate under time variance. As μ_{eff} affects t_{PHL} through the pivotal I_{DS} equation Eq. 5.2 and in digital circuits we primarily use square pulses with finite rise time t_r, we intended to find an efficacious model for μ_{eff} as a function of t_r only.

It is well studied that inside the channel the effective mobility degrades mostly due to the gate electric field [82], because it is comparatively stronger than the drain-source electric field as oxide thickness $t_{ox} \ll L$ (L = channel length). Thus we adapt a twofold modeling strategy, where we first extract μ_{eff} at only the average output swing voltage $V_{DD}/2$ for three V_g points in-between $V_{DD}/2$ and V_{DD} and then work out t_r values corresponding to them – finally constructing a function $\mu^{(t_r)}_{eff}$ varying with t_r, using these extracted μ_{eff} and obtained t_r. We start the derivation by rearranging the equation Eq. 5.12

$$t_r = C_L V_{DD} \Delta v_p / 2 A_{v_{p(\tau)}} K^{(t_r)}_n \tag{5.20}$$

$$A_{v_{p(\tau)}} = \frac{1}{12}\boldsymbol{W_l}^3(2e^{v_{p(\tau)}}) + \frac{3}{8}\boldsymbol{W_l}^2(2e^{v_{p(\tau)}}) + \frac{1}{2}\boldsymbol{W_l}(2e^{v_{p(\tau)}}) \tag{5.21}$$

Recalling $v_{p(\tau)} = v_{p(0)} + \tau \Delta v_p / t_r$ and $\tau = t_r/2 + t_{PHL}$ we trivially find $v_{p(\tau)}$ for individual $t_{PHL} = 0, t_r/4$ and $t_r/2$ or $\tau = t_r/2, 3t_r/4, t_r$. By the definition of t_{PHL}, within $0 \leq t \leq t_{PHL}$ the drain current I_{DS} swings from high to low until $V_{out} = V_{DD}/2$ – and within this time v_p swings up from $v_p = v_{p(0)} + \Delta v_p/2$ to $v_p = v_{p(0)} + \tau \Delta v_p / t_r$ Eq. 5.12. So we extract $\mu^{(t_r)}_{eff}$ for the range $(v_{p(0)} + \Delta v_p/2) \leq v_p \leq (v_{p(0)} + \tau \Delta v_p / t_r)$ at the mean v_p, i.e. $v_{p(\mu_{eff}ext)} = v_{p(0)} + \Delta v_p (1/4 + \tau/2t_r)$ (or at equivalent $V_g \approx V_{g(\mu_{eff}ext)} = V_{DD}(1/4 + \tau/2t_r)$).

Table 5.1: Variables involved in $\mu_{eff}^{(t_r)}$ modeling

τ	$v_{p(\tau)}$	$A_{v_{p(\tau)}}$	$V_{g(\mu_{eff}ext)}$	$\mu_{eff}^{(t_r)}(K_n^{(t_r)})$ extracted	$t_{r(V_{g(\mu_{eff}ext)})}$ (Eq. 5.20)
$t_r/2$	$v_{p(V_{DD}/2)} = v_{p(0)} + \frac{\Delta v_p}{2}$	$A_{v_{p(V_{DD}/2)}}$	$V_{DD}/2$	$\mu_{(V_{DD}/2)}(K_{n(V_{DD}/2)})$	$t_{r(V_{DD}/2)} = \frac{C_L V_{DD} \Delta v_p}{2K_{n(V_{DD}/2)} A_{v_{p(V_{DD}/2)}}}$
$3t_r/4$	$v_{p(3V_{DD}/4)} = v_{p(0)} + \frac{3\Delta v_p}{4}$	$A_{v_{p(3V_{DD}/4)}}$	$5V_{DD}/8$	$\mu_{(5V_{DD}/8)}(K_{n(5V_{DD}/8)})$	$t_{r(5V_{DD}/8)} = \frac{C_L V_{DD} \Delta v_p}{2K_{n(5V_{DD}/8)} A_{v_{p(3V_{DD}/4)}}}$
t_r	$v_{p(V_{DD})} = v_{p(0)} + \Delta v_p$	$A_{v_{p(V_{DD})}}$	$3V_{DD}/4$	$\mu_{(3V_{DD}/4)}(K_{n(3V_{DD}/4)})$	$t_{r(3V_{DD}/4)} = \frac{C_L V_{DD} \Delta v_p}{2K_{n(3V_{DD}/4)} A_{v_{p(V_{DD})}}}$

For extraction, we substitute the I_{DS} from SPICE simulator to Eq. 5.1, at above bias conditions. So the extracted $\mu_{eff}^{(t_r)}$ in this way, includes the secondary effects like channel length modulation. The scheme is elaborated in Table. 5.1.

From Eq. 5.20 - Eq. 5.21 it can be observed that $A_{v_{p(\tau)}}$ dominates the numerator of Eq. 5.20 because of its inherent bias dependent exponential terms, and moreover is a monotonically increasing function of $v_{p(\tau)}$. So intuitively it can be said that, $A_{v_{p(V_{DD}/2)}} \ll A_{v_{p(V_{DD})}}$ or $A_{v_{p(3V_{DD}/4)}}$ – which means $t_{r(V_{DD}/2)} \gg t_{r(5V_{DD}/8)}$ or $t_{r(3V_{DD}/4)}$. As for $t_r \to t_{r(V_{DD}/2)}$ we have $\mu_{eff}^{(t_r)} \to \mu_{(V_{DD}/2)}$, $\mu_{eff}^{(t_r)}$ saturates to $\mu_{(V_{DD}/2)}$ as $t_r \to t_{r(V_{DD}/2)}$ when $t_{r(V_{DD}/2)}$ is very large. We exploit this saturating behaviour to model $\mu_{eff}^{(t_r)}$ as a function of t_r. We interpolate Table. 5.1 data to 'Logistic function' [83] having the following generic form for real x,

$$f_{Logistic}(x) = \frac{K}{1 + \exp[-\alpha(x - \delta)]} \tag{5.22}$$

where K, α, δ are real constants and can be found out using appropriate boundary conditions. It trivially follows that $K = \lim_{x\to\infty} f_{Logistic}(x) =$ the maximum value that $f_{Logistic}(x)$ can attain. Therefore using Eq. 5.22 we may write the functional form of $\mu_{eff}^{(t_r)}$ with respect to t_r as,

$$\mu_{eff}^{(t_r)} = \frac{\mu_{(V_{DD}/2)}}{1 + \exp[-\alpha_r(t_r - \delta_r)]} \tag{5.23}$$

From previous discussions and involving Table. 5.1, we may infer that $\lim_{t_r\to\infty} \mu_{eff}^{(t_r)} = \mu_{(V_{DD}/2)}$. Moreover using Table. 5.1 data we may calculate α_r and δ_r having respective forms

$$\alpha_r = \frac{log_e[\frac{\mu_{(5V_{DD}/8)}}{\mu_{(3V_{DD}/4)}}(\frac{\mu_{(V_{DD}/2)} - \mu_{(3V_{DD}/4)}}{\mu_{(V_{DD}/2)} - \mu_{(5V_{DD}/8)}})]}{t_{r(5V_{DD}/8)} - t_{r(3V_{DD}/4)}} \tag{5.24}$$

$$\delta_r = t_{r(3V_{DD}/4)} - \frac{1}{\alpha_r} log_e[\frac{\mu_{(3V_{DD}/4)}}{\mu_{(V_{DD}/2)} - \mu_{(3V_{DD}/4)}}] \tag{5.25}$$

Eq. 5.23 is the mobility model as a function of t_r.

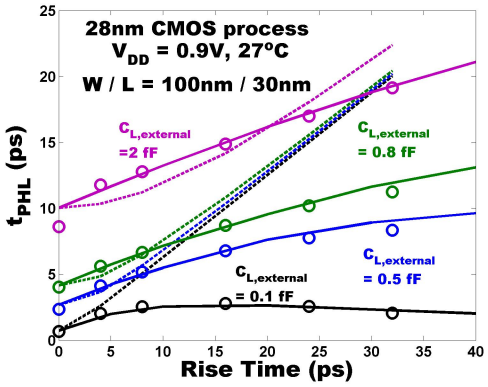


Figure 5.5: Variation of delay with rise-time (t_r) (Line: proposed model, symbol: simulation, dotted line: delay model in [1]).

5.5 Delay Model Validation

We validated our proposed model using commercial foundry processes – 22nm and 28nm CMOS as well as our in-house Berkeley-NCFET 90nm process, across a variable range of device dimensions, V_{DD} as well as rise time t_r. Moreover, we have demonstrated the validity of this model under large temperature and capacitive load variation – which indeed should demonstrate the robustness, efficacy and practicality of the model.

In order to quantify the effective output capacitive load (C_L) accurately, we have used $C_L = C_{L,external} + C_{L,intrinsic}$. Here $C_{L,external}$ is the external load attached to the output, whereas $C_{L,intrinsic}$ is the intrinsic capacitance of the MOSFET. $C_{L,intrinsic}$ is obtained through SPICE simulation.

Fig. 5.5 depicts the variation of the proposed model with rise time (t_r) and external capacitive load ($C_{L,external}$) against SPICE simulation and the popular empirical delay model given by the equation $t_{PHL} = \sqrt{t^2_{PHL(t_r=0)} + (t_r/2)^2}$ [1]. This study clearly shows the accuracy of our proposed model. Fig. 5.6 depicts the variation of the proposed model with rise time (t_r) under power supply variation.

Dependence of the proposed delay model over device channel width (W) is shown in Fig. 5.7 under variable external loading ($C_{L,external}$). We use $C_{L,intrinsic} = C_{L,intrinsic,nominal}(W/W_{nominal})$ = cumulative intrinsic capacitance of the transistors, including parasitics. $W_{nominal}$ = nominal channel width for which the $C_{L,intrinsic,nominal}$ is extracted.

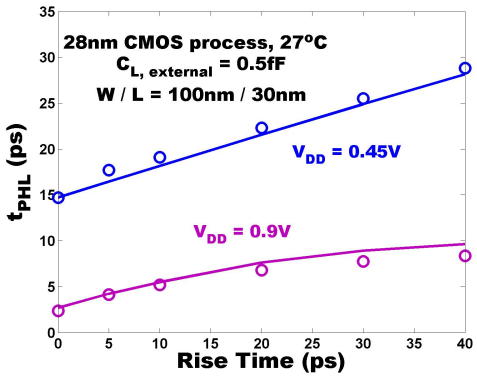


Figure 5.6: Variation of delay with rise-time (t_r) and bias voltage (V_{DD}) (Line: proposed model, symbol: simulation).

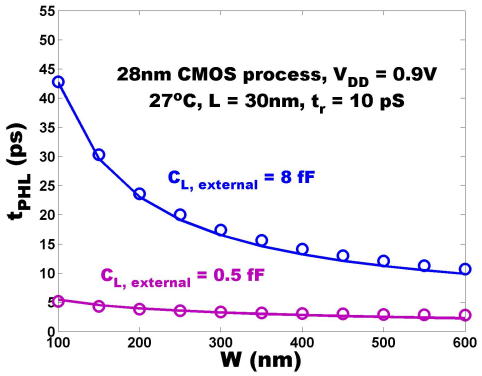


Figure 5.7: Variation of delay with NMOS channel width (W) and capacitive load ($C_{L,external}$) (Line: proposed model, symbol: simulation).

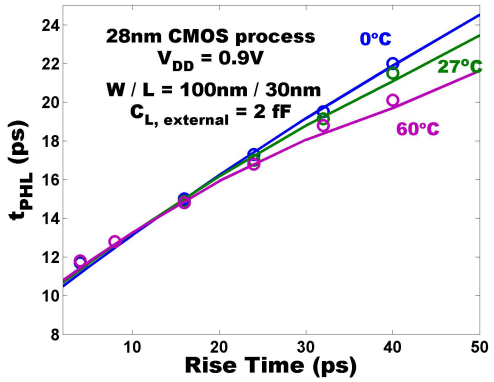


Figure 5.8: Variation of delay with rise time (t_r) and temperature (Line: proposed model, symbol: simulation).

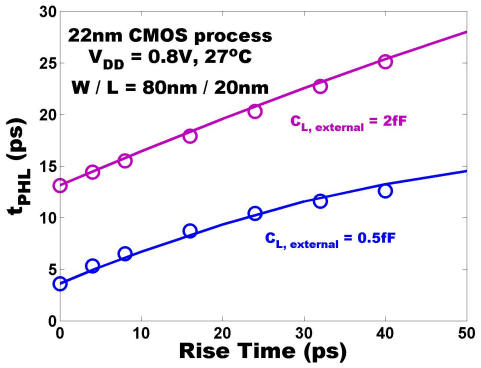


Figure 5.9: Variation of delay with rise time (t_r) using 22nm CMOS process (Line: proposed model, symbol: simulation).

When the operating temperature changes, along with degradation in effective mobility $\mu_{eff}^{(t_r)}$ [75] the intrinsic carrier concentration n_i as well as the bandgap E_g vary [82]. Hence the drive current also responds to that change, ultimately affecting the propagation delay. Therefore the validation of the a delay model with respect to temperature change is critical and we have furnished that study through Fig. 5.8.

Apart from 28nm CMOS process, we have validated the proposed model in 22nm CMOS process in Fig. 5.9 and Fig. 5.10.

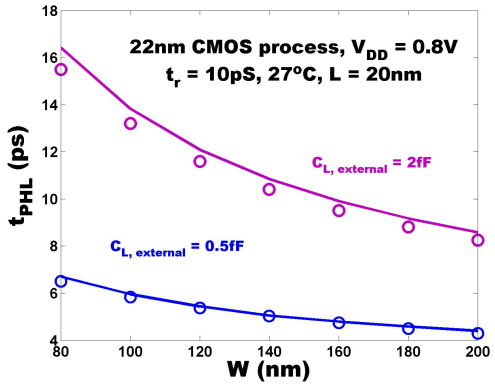


Figure 5.10: Variation of delay with channel width (W) using 22nm CMOS process (Line: proposed model, symbol: simulation).

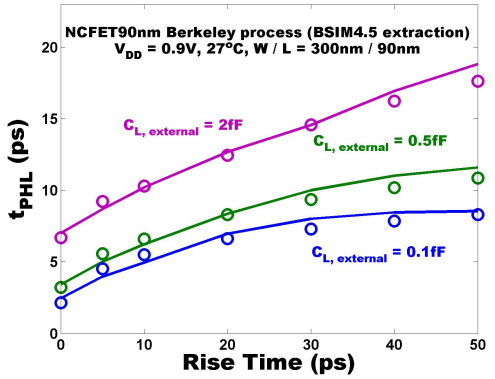


Figure 5.11: Variation of delay with rise time (t_r) using in-house Berkeley 90nm NCFET process (BSIM4.5 extraction) (Line: proposed model, symbol: simulation).

We further validate our claim that the proposed delay model is equally effective for more advanced transistors based on planar MOSFET architecture. Planar Negative Capacitance FET (NCFET) is such a device – invented at UC Berkeley [80], which already has gained momentum due to its promise for sub-60mV/ decade subthreshold slope. Using the measurement data from the in-house fabricated 90 nm planar NCFET device, we first extract a BSIM4.5 model card [81] – where $t_{ox} = 2.8nm$ in the BSIM4.5 extraction. Based on the model card we validate the delay model for a single transistor (Fig. 5.11), as well as a 2-transistor stack (Fig. 5.12, Fig. 5.13).

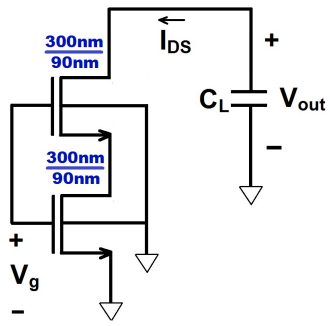


Figure 5.12: Schematic of the two transistor stack connected to an effective capacitive load (C_L).

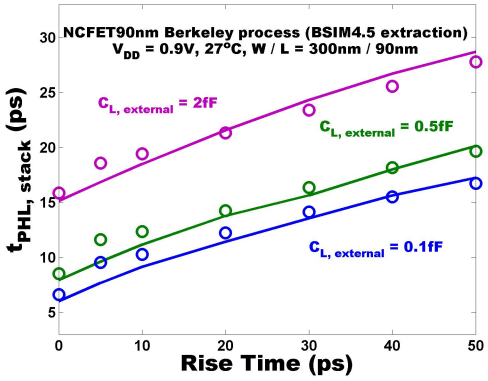


Figure 5.13: Variation of delay with rise time (t_r) for a 2 transistor stack (Fig. 5.12) using in-house Berkeley 90nm NCFET process (BSIM4.5 extraction) (Line: proposed model, symbol: simulation).

The proposed model is further applied in standard circuit implementation. We implemented a 31-stage as well as a 61-stage CMOS Ring Oscillator [1] with NMOS driver (W/L = 100nm / 30nm) and PMOS load (W/L = 130nm / 30nm) in the 28 nm CMOS process. The variation of oscillation frequency with the supply voltage V_{DD} is captured accurately by the proposed model, as shown in Fig. 5.14.

5.6 Conclusion

In this chapter we have proposed a charge based analytical and portable propagation delay model for CMOS high-speed circuits. This computationally efficient and accurate

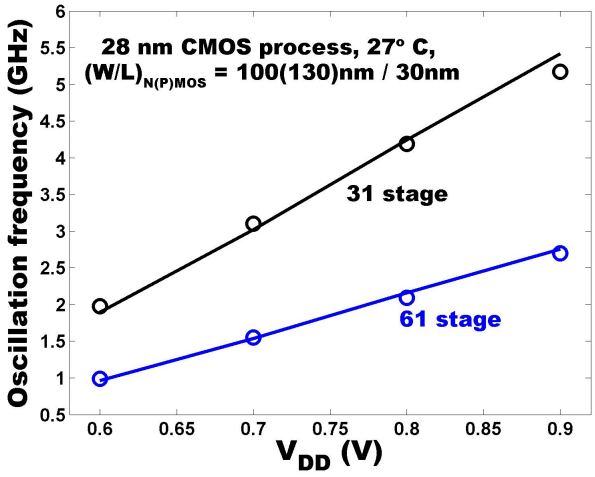


Figure 5.14: Variation of Ring Oscillator oscillation frequency with supply voltage V_{DD} (Line: proposed model, symbol: simulation).

model covers all the regions of MOSFET operation and is shown to be process independent. A major contribution of this work is the simple mobility model it proposes, which addresses the inevitable electric field dependent mobility degradation phenomenon as a function of input slope. We also demonstrated that the proposed charge based delay model is equally applicable for more advanced transistors (e.g. planar NCFET) derived from planar MOSFET architecture.

Chapter 6

Prototype and Measurement Plan

6.1 Prototype

The proposed PLL was taped out in July 2023. Fig. 6.1 shows the layout of the PLL core, and Fig. 6.2 presents the pad definition. In this version, the reference signal is fed through a GSSG probe from an external instrument, and the divided output is sent to a GSG pad. The DSM is implemented off-chip, and the control signal is fed into the chip through the pad MC.

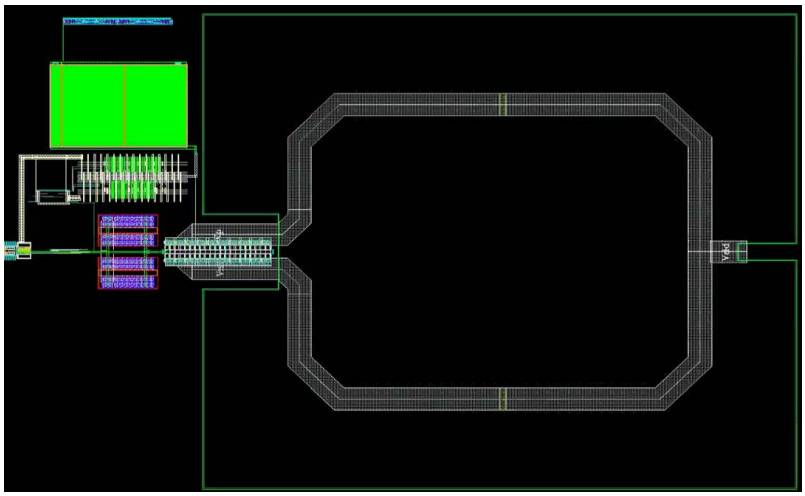


Figure 6.1: Layout of the proposed PLL core.

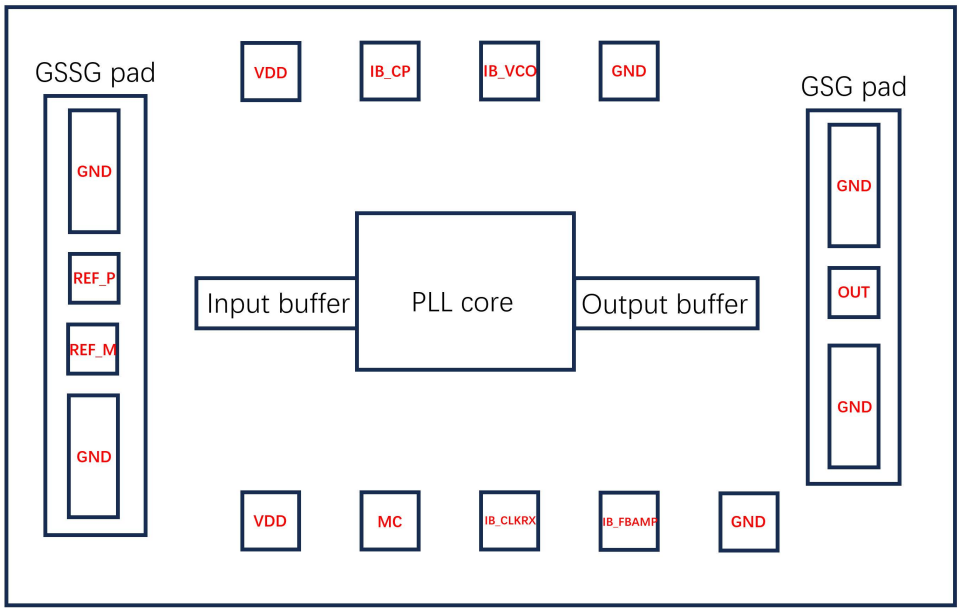


Figure 6.2: Pads of the taped-out chip.

6.2 Measurement Plan

For measurement of the stand-alone PLL as a chirp generator, the key metrics are phase noise and chirp linearity. As mentioned in Chapter 2, in FMCW radar applications, the dynamic phase noise is important for the radar performance. But in practical measurement, only static phase noise can be measured, as an indicator of the dynamic phase noise of the PLL. According to calculations, the dynamic phase noise is $2 \sim 3$ dBc/Hz higher than the static one.

1. Fix the division ratio and measure the output frequency to verify the functionality of the PLL.
2. Vary the division ratio to measure the frequency range the PLL can operate.
3. Fix the division ratio and measure the phase noise.
4. Vary the bias current of the charge pump and measure the phase noise.
5. Modulate the division ratio and measure the output frequency to verify the chirp generating functionality. Plot the $f_{out}(t) - t$ curve and calculate linearity metrics, such as FM RMS error and peak error.
6. Vary the chirp slope. Measure and plot $f_{out}(t) - t$ curve and calculate linearity metrics.

Chapter 7

Conclusions

In this thesis, we have illustrated the PLL design for FMCW radar systems. The FMCW radar imposes requirements on the PLL-based chirp generator in several aspects. Both the phase noise and the bandwidth of the PLL influence the measurement accuracy and resolution of the radar. The phase noise should be minimized, given a certain power consumption, and the bandwidth needs to be matched with the FMCW chirp slope to achieve better linearity.

For system-level design, unlike PLLs for other applications, the output of the chirp generator PLL is always changing. In many applications, such as precision measurement, the output never truly settles at each step. This necessitates careful modeling of the loop dynamics. In this thesis, conventional PLL phase noise and settling time models are presented and adapted for the chirp generator. However, these models are insufficient for optimizing the design. Therefore, a more accurate time-domain model for calculating the chirp is proposed. This model aids in designing the PLL bandwidth, calculating the acceptable chirp slope for a given PLL, and computing the dynamic phase noise. To the best of our knowledge, this is the first relatively accurate model for the entire chirp generation process.

For the design of circuit blocks, the frequency division modules, which include a dual-modulus divider and a Delta-Sigma Modulator, are presented. For the high-frequency circuit, an analytical model and a corresponding design methodology are proposed. The PLL has been taped-out, and measurements will be conducted in the near future.

In conclusion, this thesis presents the design of the PLL used for the FMCW radar chirp generator. Considerable effort has been expended on the fine and detailed modeling of the PLL. For the first time, a time-domain calculation method is proposed to model the entire chirp.

Bibliography

[1] S.-M. Kang and Y. Leblebici, *CMOS digital integrated circuits*, 3rd ed. Boston, Mass. [u.a.]: McGraw-Hill, 2003, includes bibliographical references and index.

[2] H. Griffiths, "Early history of bistatic radar," 2008.

[3] R. Watson-Watt, *Three Steps to Victory*. Odhams Press, 1957.

[4] A. Stove, "Modern fmcw radar - techniques and applications," in *First European Radar Conference, 2004. EURAD.*, 2004, pp. 149–152.

[5] G. Bergland, "Fast fourier transform hardware implementations–an overview," *IEEE Transactions on Audio and Electroacoustics*, vol. 17, no. 2, pp. 104–108, 1969.

[6] J.-J. Lin, Y.-P. Li, W.-C. Hsu, and T.-S. Lee, "Design of an FMCW radar baseband signal processing system for automotive application," *SpringerPlus*, vol. 5, no. 1, Jan. 2016.

[7] B. Dekker, S. Jacobs, A. Kossen, M. Kruithof, A. Huizing, and M. Geurts, "Gesture recognition with a low power fmcw radar and a deep convolutional neural network," in *2017 European Radar Conference (EURAD)*, 2017, pp. 163–166.

[8] C. Rajkumar and A. Bazil Raj, "Design and development of dsp interfaces and algorithm for fmcw radar altimeter," in *2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT)*, 2019, pp. 720–725.

[9] C. Sklarczyk, "Contactless characterization of coatings with a microwave radar sensor," in *Nondestructive Characterization of Materials XI*, R. E. Green, B. B. Djordjevic, and M. P. Hentschel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 743–748.

[10] D. Brumbi, "Low power fmcw radar system for level gaging," in *2000 IEEE MTT-S International Microwave Symposium Digest (Cat. No.00CH37017)*, vol. 3, 2000, pp. 1559–1562 vol.3.

[11] J. Park, D.-H. Jung, K.-B. Bae, and S.-O. Park, "Range-doppler map improvement in fmcw radar for small moving drone detection using the stationary point concentration

technique," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 5, pp. 1858–1871, 2020.

[12] A. Joseph, V. Jain, S. N. Ong, R. Wolf, S. F. Lim, and J. Singh, "Technology positioning for mm wave applications: 130/90nm sige bicmos vs. 28nm rfcmos," in *2018 IEEE BiCMOS and Compound Semiconductor Integrated Circuits and Technology Symposium (BCICTS)*, 2018, pp. 18–21.

[13] F. Dielacher, "Silicon technology trends," Infineon, Tech. Rep., Jan. 2020. [Online]. Available: https://vimeo.com/386238030

[14] P. Magnée, D. Leenaerts, M. Van der Heijden, T. V. Dinh, I. To, and I. Brunets, "The future of sige bicmos: bipolar amplifiers for high-performance millimeter-wave applications," in *2021 IEEE BiCMOS and Compound Semiconductor Integrated Circuits and Technology Symposium (BCICTS)*, 2021, pp. 1–7.

[15] "Miniaturization of mmWave sensors enabled by CMOS technology - Embedded processing - Technical articles - TI E2E support forums." [Online]. Available: https://e2e.ti.com/blogs_/b/process/posts/miniaturization-of-mmwave-sensors-enabled-by-cmos-technology

[16] B. Sene, D. Reiter, H. Knapp, H. Li, T. Braun, and N. Pohl, "An automotive d-band fmcw radar sensor based on a sige-transceiver mmic," *IEEE Microwave and Wireless Components Letters*, pp. 1–4, 2021.

[17] X. Yi, C. Wang, X. Chen, J. Wang, J. Grajal, and R. Han, "A 220-to-320-ghz fmcw radar in 65-nm cmos using a frequency-comb architecture," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 2, pp. 327–339, 2021.

[18] A. Zandieh, S. Bonen, M. S. Dadash, M. J. Gong, J. Hasch, and S. P. Voinigescu, "155 ghz fmcw and stepped-frequency carrier ofdm radar sensor transceiver ic featuring a pll with lt;30 ns settling time and 40 fs rms jitter," *IEEE Transactions on Microwave Theory and Techniques*, vol. 69, no. 11, pp. 4908–4924, 2021.

[19] A. Dürr, D. Schwarz, S. Häfner, M. Geiger, F. Roos, M. Hitzler, P. Hügler, R. Thomä, and C. Waldschmidt, "High-resolution 160-ghz imaging mimo radar using mmics with on-chip frequency synthesizers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 9, pp. 3897–3907, 2019.

[20] M. Kucharski, W. A. Ahmad, H. J. Ng, and D. Kissinger, "Monostatic and bistatic ¡italic¿g¡/italic¿-band bicmos radar transceivers with on-chip antennas and tunable tx-to-rx leakage cancellation," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 3, pp. 899–913, 2021.

[21] A. Visweswaran, K. Vaesen, S. Sinha, I. Ocket, M. Glassee, C. Desset, A. Bourdoux, and P. Wambacq, "9.4 a 145ghz fmcw-radar transceiver in 28nm cmos," in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019, pp. 168–170.

[22] J. Grzyb, K. Statnikov, N. Sarmah, B. Heinemann, and U. R. Pfeiffer, "A 210–270-ghz circularly polarized fmcw radar with a single-lens-coupled sige hbt chip," *IEEE Transactions on Terahertz Science and Technology*, vol. 6, no. 6, pp. 771–783, 2016.

[23] X. Chen, M. I. W. Khan, X. Yi, X. Li, W. Chen, J. Zhu, Y. Yang, K. E. Kolodziej, N. M. Monroe, and R. Han, "A 140ghz transceiver with integrated antenna, inherent-low-loss duplexing and adaptive self-interference cancellationfor fmcw monostatic radar," in *2022 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2022, pp. 80–81.

[24] D. Dhar, P. van Zeijl, D. Milosevic, H. Gao, and A. van Roermund, "Modeling and analysis of the effects of pll phase noise on fmcw radar performance," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[25] M. Rameez, M. Dahl, and M. I. Pettersson, "Adaptive digital beamforming for interference suppression in automotive fmcw radars," in *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 0252–0256.

[26] X. Dao, M. Gao, C. Li, and Y. Wang, "Adaptive leakage signal cancellation algorithm in heterodyne fmcw system," *Digital Signal Processing*, vol. 108, p. 102882, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105120042030227X

[27] A. Figueroa, N. Joram, and F. Ellinger, "Automatic delay and phase mismatch calibration in fmcw mimo radar," in *2020 17th European Radar Conference (EuRAD)*, 2021, pp. 402–405.

[28] K. Subburaj, A. Mani, K. Dandu, K. Bhatia, K. Ramasubramanian, S. Murali, R. Sachdev, P. Gupta, S. Samala, D. Shetty, Z. Parkar, S. Ram, V. Dudhia, D. Breen, S. Bharadwaj, S. Bhatara, and B. Ginsburg, "Digitally assisted mm-wave fmcw radar for high performance," in *2020 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, 2020, pp. 1–4.

[29] J. Vankka, *Direct Digital Synthesizers Theory, Design and Applications*. Springer US, 2001.

[30] A. Mostajeran, A. Cathelin, and E. Afshari, "A 170-GHz fully integrated single-chip FMCW imaging radar with 3-d imaging capability," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 10, pp. 2721–2734, oct 2017.

[31] J. Zhang, S. S. Ahmed, and A. Arbabian, "Effects of reference frequency harmonic spurs in FMCW radar systems," in *2021 IEEE Radar Conference (RadarConf21)*. IEEE, may 2021.

[32] S. Ayhan, S. Scherr, A. Bhutani, B. Fischbach, M. Pauli, and T. Zwick, "Impact of frequency ramp nonlinearity, phase noise, and SNR on FMCW radar accuracy," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 10, pp. 3290–3301, oct 2016.

[33] P. V. Brennan, Y. Huang, M. Ash, and K. Chetty, "Determination of sweep linearity requirements in FMCW radar systems based on simple voltage-controlled oscillator sources," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 1594–1604, jul 2011.

[34] Y.-S. Son, H.-K. Sung, and S. W. Heo, "Automotive frequency modulated continuous wave radar interference reduction using per-vehicle chirp sequences," *Sensors*, vol. 18, no. 9, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/2831

[35] R. W. Austin Harney, "Fractional-n pll-based frequency sweepgenerator for fmcw radar," https://www.armms.org/media/uploads/1304696084.pdf, Apr. 2010.

[36] T. Instruments, "Intro to mmwave sensing: Fmcw radars," https://training.ti.com/intro-mmwave-sensing-fmcw-radars-module-1-range-estimation, Apr. 2017.

[37] B. Welp, G. Briese, and N. Pohl, "Ultra-wideband fmcw radar with over 40 ghz bandwidth below 60 ghz for high spatial resolution in sige bicmos," in *2020 IEEE/MTT-S International Microwave Symposium (IMS)*, 2020, pp. 1255–1258.

[38] T. Jaeschke, C. Bredendiek, S. Küppers, and N. Pohl, "High-precision d-band fmcw-radar sensor based on a wideband sige-transceiver mmic," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 12, pp. 3582–3597, 2014.

[39] W. Deng, H. Jia, and B. Chi, "Silicon-based FMCW signal generators: A review," *Journal of Semiconductors*, vol. 41, no. 11, p. 111401, Nov. 2020. [Online]. Available: https://doi.org/10.1088/1674-4926/41/11/111401

[40] Y.-B. Hsieh and Y.-H. Kao, "A fully integrated spread-spectrum clock generator by using direct vco modulation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 7, pp. 1845–1853, 2008.

[41] Y. Kim, T. J. Reck, M. Alonso-delPino, T. H. Painter, H.-P. Marshall, E. H. Bair, J. Dozier, G. Chattopadhyay, K.-N. Liou, M.-C. F. Chang, and A. Tang, "A ku-band cmos fmcw radar transceiver for snowpack remote sensing," *IEEE Transactions on Microwave Theory and Techniques*, vol. 66, no. 5, pp. 2480–2494, 2018.

[42] F. Rodriguez-Morales, C. Leuschen, C. L. Carabajal, J. Paden, J. A. Wolf, S. Garrison, and J. W. McDaniel, "An improved uwb microwave radar for very long-range measurements of snow cover," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7761–7772, 2020.

[43] Y.-P. Su, C.-Y. Huang, and S.-J. Chen, "A 24-ghz fully integrated cmos transceiver for fmcw radar applications," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 11, pp. 3307–3317, 2021.

[44] K.-W. Ha, J.-Y. Lee, G.-H. Ko, Y.-J. Kim, J.-G. Kim, and D. Baek, "Fully integrated dual-mode x-band radar transceiver using configurable receiver and local oscillator," *IEEE Access*, vol. 8, pp. 151 403–151 414, 2020.

[45] W. Wu, R. B. Staszewski, and J. R. Long, "A 56.4-to-63.4 ghz multi-rate all-digital fractional-n pll for fmcw radar applications in 65 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 5, pp. 1081–1096, 2014.

[46] Z. Shen, H. Jiang, F. Yang, Y. Wang, Z. Zhang, J. Liu, and H. Liao, "A 24 ghz self-calibrated all-digital fmcw synthesizer with 0.01

[47] J. Xiao, N. Liang, B. Chen, and M. Liu, "An 8.55–17.11-GHz DDS FMCW chirp synthesizer PLL based on double-edge zero-crossing sampling PD with 51.7-fssubrms/sub jitter and fast frequency hopping," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 3, pp. 267–276, mar 2022.

[48] J. Li, C. Shi, Z. Chen, H. Deng, J. Chen, and R. Zhang, "A 24 GHz FMCW/doppler dual-mode frequency synthesizer with 68.8 kHz RMS FM error and 1.25 GHz chirp bandwidth," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2518–2522, may 2022.

[49] C.-T. Chen, Y.-H. Yang, and T.-C. Lee, "A type-3 FMCW radar synthesizer with wide frequency modulation bandwidth," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, may 2022.

[50] H. Shanan, D. Dalton, V. Chillara, and P. Dato, "A 9-to-12ghz coupled-rtwo fmcw adpll with 97fs rmsjitter, -120dbc/hz pn at 1mhz offset, and with retrace timeof 12.5ns and 2µs chirp settling time," in *2022 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2022, pp. 146–147.

[51] P. T. Renukaswamy, N. Markulic, S. Park, A. Kankuppe, Q. Shi, P. Wambacq, and J. Craninckx, "17.7 a 12mw 10ghz fmcw pll based on an integrating dac with 90khz rms frequency error for 23mhz/µs slope and 1.2ghz chirp bandwidth," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 278–280.

[52] J. Vovnoboy, R. Levinger, N. Mazor, and D. Elad, "A dual-loop synthesizer with fast frequency modulation ability for 77/79 ghz fmcw automotive radar applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 5, pp. 1328–1337, 2018.

[53] T. Usugi, T. Murakami, Y. Utagawa, S. Kishimoto, M. Kohtani, I. Ando, K. Matsunaga, C. Arai, T. Arai, and S. Yamaura, "A 77 ghz 8rx3tx transceiver for 250 m long range

automotive radar in 40 nm cmos technology," in *2020 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, 2020, pp. 23–26.

[54] C. Li, Z. Peng, T.-Y. Huang, T. Fan, F.-K. Wang, T.-S. Horng, J.-M. Munoz-Ferreras, R. Gomez-Garcia, L. Ran, and J. Lin, "A review on recent progress of portable short-range noncontact microwave radar systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 5, pp. 1692–1706, may 2017.

[55] X. Chen, M. I. W. Khan, X. Yi, X. Li, W. Chen, J. Zhu, Y. Yang, K. E. Kolodziej, N. M. Monroe, and R. Han, "A 140ghz transceiver with integrated antenna, inherent-low-loss duplexing and adaptive self-interference cancellation for FMCW monostatic radar," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*. IEEE, feb 2022.

[56] K. YOSHIOKA, "A tutorial and review of automobile direct ToF LiDAR SoCs: Evolution of next-generation LiDARs," *IEICE Transactions on Electronics*, vol. E105.C, no. 10, pp. 534–543, oct 2022.

[57] S. H. V. Bhupathiraju, J. Sheldon, L. A. Bauer, V. Bindschaedler, T. Sugawara, and S. Rampazzi, "EMI-LiDAR: Uncovering vulnerabilities of LiDAR sensors in autonomous driving setting using electromagnetic interference," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, may 2023.

[58] IMEC, "140 ghz radar for gesture recognition technology and driver monitoring," https://www.imec-int.com/en/expertise/radar-sensing-systems/140ghz-radar-modules, 2019.

[59] E. Chou, N. Baniasadi, H. Beshary, M. Wei, E. Naviasky, L. Iotti, and A. Niknejad, "A low-power and energy-efficient d-band CMOS four-channel receiver with integrated LO generation for digital beamforming arrays," in *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*. IEEE, sep 2022.

[60] Keysight. (2023) E8267d psg vector signal generator. [Online]. Available: https://www.keysight.com/us/en/assets/7018-01210/data-sheets/5989-0697.pdf

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

[62] B. Miller and B. Conley, "A multiple modulator fractional divider," in *44th Annual Symposium on Frequency Control*. IEEE, 1900.

[63] H. Arora, N. Klemmer, J. Morizio, and P. Wolf, "Enhanced phase noise modeling of fractional-n frequency synthesizers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 2, pp. 379–395, feb 2005.

[64] A. L. Lacaita, S. Levantino, and C. Samori, *Integrated frequency synthesizers for wireless systems*. Cambridge University Press, 2007.

[65] F. Herzel, A. Ergintav, and Y. Sun, "Phase noise modeling for integrated PLLs in FMCW radar," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 3, pp. 137–141, mar 2013.

[66] Y. Pan and J. Xu, "Phase noise analysis of fractional-n PLL based frequency ramp generator for FMCW radar," in *2015 Asia-Pacific Microwave Conference (APMC)*. IEEE, dec 2015.

[67] V. Dham, "Programming chirp parameters in ti radar devices application report programming chirp parameters in ti radar devices," 2017. [Online]. Available: https://www.ti.com/lit/an/swra553a/swra553a.pdf?ts=1711103429506

[68] J. Calder, "The calculus of variations," 2017. [Online]. Available: https://www-users.cse.umn.edu/~jwcalder/5588S17/calculus_of_variations.pdf

[69] I. M. Gelfand and S. V. Fomin, *Calculus of Variations*. Dover Publications, Incorporated, 2012.

[70] B. Baker, "How delta-sigma adcs work, part 1," Texas Instruments Incorporated, 2011. [Online]. Available: https://www.ti.com

[71] Z. Deng and A. M. Niknejad, "The speed–power trade-off in the design of cmos true-single-phase-clock dividers," *IEEE Journal of Solid-State Circuits*, Nov. 2010.

[72] S. Krishnamurthy and A. M. Niknejad, "10-35ghz passive mixer-first receiver achieving +14dbm in-band iip3 for digital beam-forming arrays," in *2020 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*. IEEE, Aug. 2020.

[73] D. Chowdhury, L. Ye, E. Alon, and A. M. Niknejad, "An efficient mixed-signal 2.4-ghz polar power amplifier in 65-nm cmos technology," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 8, pp. 1796–1809, Aug. 2011.

[74] F. Bohn, H. Wang, A. S. Natarajan, S. Jeon, and A. Hajimiri, "Fully integrated frequency and phase generation for a 6amp;x2013;18ghz tunable multi-band phased-array receiver in cmos," in *2008 IEEE Radio Frequency Integrated Circuits Symposium*. IEEE, Jun. 2008.

[75] M. Chanda, A. S. Chakraborty, and C. K. Sarkar, "Complete delay modeling of sub-threshold cmos logic gates for low-power application," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 29, no. 2, pp. 132–145, Feb. 2015.

[76] A. Wang and A. Chandrakasan, "A 180-mv subthreshold fft processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[77] F. Frustaci, P. Corsonello, and S. Perri, "Analytical delay model considering variability effects in subthreshold domain," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 3, pp. 168–172, Mar. 2012.

[78] A. Valentian, O. Thomas, A. Vladimirescu, and A. Amara, "Modeling subthreshold soi logic for static timing analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 6, pp. 662–669, Jun. 2004.

[79] C. Enz, *Charge-based MOS transistor modeling*, E. A. Vittoz, Ed. Chichester, England ;: John Wiley, 2006, includes bibliographical references (p. [291]-298) and index.

[80] J. C. Wong and S. Salahuddin, "Negative capacitance transistors," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 49–62, Jan. 2019.

[81] "Bsim group." [Online]. Available: http://bsim.berkeley.edu

[82] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement," *Correspondence mathematique et physique*, vol. 10, pp. 113–129, 1838.

[83] S. M. Sze, *Physics of semiconductor devices*, fourth edition ed., Y. Li and K. K. Ng, Eds. Hoboken, NJ: Wiley, 2021.

Appendices

Appendix A

Matlab Code for Noise and Settling Time Calculation

```
clear all
close all
clc

%% Parameters

% PLL circuit block parameters
Icp=150e-6;
Kpd=Icp/2/pi;
Kvco=50e6;
N=8;
order_lpf=2;     % order of the LPF
dsm_order=2;     % order of the DSM
ref_noise_on=1; % noise of reference 1=on 0=off
ref_source=1;    % options of reference source in BWRC lab:
                 % source option 1: HP 83732A, worse phase noise
                    performance
                 % source option 2: Agilent E4438C, moderate
                    phase noise performance
                 % source option 3: Agilent E8267D, best phase
                    noise performance

% FMCW chirp parameters
FMCW_bandwidth=8e+9;
chirp_slope=200    *1e+6/1e-6;
T_chirp=FMCW_bandwidth/chirp_slope;
```

```
ref_freq=2.875e+9; % reference frequency
fref=ref_freq;
wref=2*pi*fref;
PFD_cycle=1/ref_freq;

% Delta-Sigma modulator (DSM) and divider parameters
bit_in=12;            % input bit number of DSM.
                      % This is a 1-bit-output DSM
N_freq_multi=6;       % frequency multiplication ratio after VCO
f1_vco=23e+9;         % VCO starting frequency
f2_vco=f1_vco+FMCW_bandwidth/N_freq_multi;%VCO ending frequency
N0=f1_vco/fref;       % starting division ratio=8
N1=f2_vco/fref;       % ending division ratio=8.4638

% Choose to use divide-by-8/10 prescaler.
% Following designs are based on such prescaler.

alpha=(N1-N0)/T_chirp;    % slope of division ratio
q_N0=ceil((N0-9+0.0001)*2^(bit_in-1));
q_N1=ceil((N1-9)*2^(bit_in-1));
N_step=q_N1-q_N0+1;       % number of frequency steps
T_step=T_chirp/N_step;
N_PFD_cycle=T_chirp/N_step/PFD_cycle;% number of PFD cycles in
   one chirp step
freq_resolution=fref/2^bit_in;% frequency step/resolution

% simulation parameters for phase noise
fstart = 10e3;
fstep = 1e3;
fend = 100e6;
foff = fstart:fstep:fend;% offset frequency from the carrier
s = 1i*2*pi*foff;
Kboltzmann = 1.38e-23;
T_room = 300;

%% Noise sources
% reference signal: from testing instrument
% source option 1: HP 83732A
% source option 2: Agilent E4438C
% source option 3: Agilent E8267D
if ref_source==1
    n_ref_floor=10^(-140/10);
```

```
    fc_ref=2.8184e+6;
    phi_ref=sqrt(n_ref_floor*(fc_ref./foff+1));
elseif ref_source==2
    n_ref_floor=10^(-140/10);
    fc_ref=1e+6;
    phi_ref=sqrt(n_ref_floor*(fc_ref./foff+1));
else
    n_ref_floor=10^(-140/10);
    fc_ref=200e+3;
    phi_ref=sqrt(n_ref_floor*(fc_ref./foff+1));
end
% PFD+CP
phi_e_mean=0.363;% from linearity calculation
phi_e_mean_degree=phi_e_mean/pi*180;
alpha0=6.6125/100;
ncurr0=3.5e-12/sqrt(alpha0*2);
fc0_ncurr=8.3111e+6/(alpha0*2);
if abs(phi_e_mean)<pi
    duty_cycle=abs(phi_e_mean)/2/pi;
elseif abs(phi_e_mean)>pi && abs(phi_e_mean)<2*pi
    duty_cycle=(2*pi-abs(phi_e_mean))/2/pi;
end
n_floor=ncurr0*sqrt(alpha0*2+duty_cycle);
fc_pfdcp=fc0_ncurr*(alpha0*2+duty_cycle);
ncurr_pfdcp=n_floor*sqrt(fc_pfdcp./foff+1);% noise current
                                           %spectrum, in A/sqrt(Hz)
% LPF
if order_lpf==2% 2nd-order LPF
    C1=1e-12;
    C2=12e-12;
    R2=15e+3;
    A0=C1+C2;
    A1=C1*C2*R2;
    Zlpf=(1+R2*C2*s)./s./(A0+A1*s);
elseif order_lpf==3% 3rd-order LPF
    C1=5.4e-12;
    C2=54.5e-12;
    C3=54.5e-12;
    R2=34.38e+3;
    R3=34.38e+3;
    A0=C1+C2+C3;
    A1=C2*R2*(C1+C3)+C3*R3*(C1+C2);
```

```
    A2=C1*C2*C3*R2*R3;
    A3=0;
    Zlpf=(1+s*R2*C2)./(s.*(A0+s*A1+s.^2*A2+s.^3*A3));
else% 4th-order LPF
    C1=5.4e-12;
    C2=54.5e-12;
    C3=54.5e-12;
    C4=54.5e-12;
    R2=34.38e+3;
    R3=34.38e+3;
    R4=34.38e+3;
    A0=C1+C2+C3+C4;
    A1=C2*R2*(C1+C3+C4)+C4*R4*(C1+C2+C3)+R3*(C1+C2)*(C3+C4);
    A2=C1*C2*R2*R3*(C3+C4)+C4*R4*(C2*C3*R3+C1*C3*R3+C1*C2*R2+C2
       *C3*R2);
    A3=C1*C2*C3*C4*R2*R3*R4;
    Zlpf=(1+s*R2*C2)./(s.*(A0+s*A1+s.^2*A2+s.^3*A3));
end
ncurr_filter=sqrt(4*Kboltzmann*T_room.*real(1./Zlpf));
% above: noise current of the filter

% VCO
PN_1M = -105;  % phase noise @ 1 MHz [dBc]
fc_vco = 10e6; % 1/f^3 cut frequency
fout = 23e9;   % output frequency
f1M=1e6;
n_floorVCO = 10^(PN_1M/10)*f1M^2/(1+fc_vco/f1M);
phi_vco = sqrt(n_floorVCO*(1./foff.^2+fc_vco./foff.^3));

% DSM
fs=fout/N;
nv_dsm=sqrt((2*pi)^2/12/fs*(2*sin(pi*foff./fs)).^(2*(dsm_order
   -1)));

%% Transfer functions

Gfwd= Icp*Kvco*Zlpf./s;
Grev=1/N;
T=Gfwd*Grev;
Gnref=N*Gfwd./(N+Gfwd);
% PFD and CP
Gicp=2*pi/Icp*Gnref;
```

```
% LPF
Glpf = (2*pi/Icp).*Gnref;
% VCO
Gvco=1./(1+T);
% divider
Gdiv=-Gnref;

%% Settling time calculation
% this calculation assumes the step time is infinite

% 2nd-order LPF parameters
b0=1;
b1=R2*C2;
b2=0;
a1=A0;
a2=A1;

t_end=30e-6;
t_step=0.1e-9;
t=[0:t_step:t_end];    % time axis
y0=[pi wref/1000 0];   % initial conditions
[t,y]=ode45(@(t,y) odefcn(t,y,b0,b1,b2,a1,a2,Kvco,N0,wref,Kpd,
   alpha),t,y0);
% above: function definition is at the end of the program
% calculate settling time
dphie=(y(:,1)-y(end,1))/pi*180;
th_settling=0.01;           % settling threshold in degree
[M_settle,I_settle]=min(abs(dphie-th_settling));
T_settling=I_settle*t_step; % settling time in us

%% Output phase noise calculation
% PFD+CP noise at the output
ncurr_pfdcp_o = ncurr_pfdcp .* Gicp;
% LPF noise at the output
ncurr_filter_o = abs(ncurr_filter) .* Glpf;
% VCO phase noise at the output
phi_vco_o = phi_vco .* Gvco;
% DSM phase noise at the output
nv_dsm_o=nv_dsm.*Gdiv;
% total noise
phase_noise_total=abs(ncurr_pfdcp_o).^2+abs(ncurr_filter_o).^2+
   abs(phi_vco_o).^2+abs(nv_dsm_o).^2;
```

```
phi_vco_o_dB = 10*log10(abs(phi_vco_o).^2);
ncurr_pfdcp_o_dB = 10*log10(abs(ncurr_pfdcp_o).^2);
ncurr_filter_o_dB = 10*log10(abs(ncurr_filter_o).^2);
nv_dsm_o_dB=10*log10(abs(nv_dsm_o).^2);
phase_noise_total_dB = 10*log10(abs(phase_noise_total));

%% Plots

% output phase noise contribution from circuit blocks
figure(1)
plot(log10(foff),phi_vco_o_dB,'LineWidth',2);
hold on
plot(log10(foff),ncurr_pfdcp_o_dB,'LineWidth',2);
plot(log10(foff),ncurr_filter_o_dB,'LineWidth',2);
plot(log10(foff),nv_dsm_o_dB,'LineWidth',2);
plot(log10(foff),phase_noise_total_dB,'LineWidth',2);
% plot(log10(foff),total_o_dB,'LineWidth',2);
hold off
legend('VCO','PFD+CP','Filter','DSM','Total');
xlabel('log10(offset freq.)','fontsize',14)
ylabel('dBc/Hz','fontsize',14);
title('Phase noise contribution without ref noise');
% output phase noise including reference noise
if ref_noise_on==1
    phi_ref_o = phi_ref .* Gnref;
    phase_noise_total_w_ref=abs(ncurr_pfdcp_o).^2+abs(
       ncurr_filter_o).^2+abs(phi_vco_o).^2+abs(nv_dsm_o).^2+
       abs(phi_ref_o).^2;
    phi_ref_o_dB=10*log10(abs(phi_ref_o).^2);
    phase_noise_total_w_ref_dB = 10*log10(abs(
       phase_noise_total_w_ref));

    figure(2)
    plot(log10(foff),phi_vco_o_dB,'LineWidth',2);
    hold on
    plot(log10(foff),ncurr_pfdcp_o_dB,'LineWidth',2);
    plot(log10(foff),ncurr_filter_o_dB,'LineWidth',2);
    plot(log10(foff),nv_dsm_o_dB,'LineWidth',2);
    plot(log10(foff),phi_ref_o_dB,'LineWidth',2);
    plot(log10(foff),phase_noise_total_w_ref_dB,'LineWidth',2);
    %plot(log10(foff),total_o_dB,'LineWidth',2);
```

```
    hold off
    legend('VCO','PFD+CP','Filter','DSM','Ref','Total');
    xlabel('log10(offset freq.)','fontsize',14)
    ylabel('dBc/Hz','fontsize',14);
    %title('Phase noise contribution with ref noise');
end
% % frequency response of PLL T
% figure(3)
% plot(log10(foff),log10(abs(T)));
% title('T of PLL');
% xlabel('log(offset frequency)','fontsize',14)
% ylabel('log10(T)','fontsize',14);
PLL_bandwidth= interp1(abs(T),foff,1);
PLL_phase_margin=180+interp1(foff,angle(Gfwd),PLL_bandwidth)/pi
   *180;
% transfer function of the low-pass filter
% figure(4)
% plot(log10(foff),mag2db(abs(Zlpf)));
% title('LPF transfer function');
% xlabel('log(offset frequency)','fontsize',14)
% ylabel('dB(Z_{LPF})','fontsize',14);

% % noise spectrum of DSM
% figure(5)
% plot(log10(foff),mag2db(abs(nv_dsm)));
% title('DSM phase noise');
% xlabel('log(offset frequency)','fontsize',14)
% ylabel('DSM phase noise','fontsize',14);

% put important plots in one window, which are figure 1, 4, and
    7
figure(8)
subplot(1,3,1)% figure 1
    plot(log10(foff),phi_vco_o_dB,'LineWidth',2);
    hold on
    plot(log10(foff),ncurr_pfdcp_o_dB,'LineWidth',2);
    plot(log10(foff),ncurr_filter_o_dB,'LineWidth',2);
    plot(log10(foff),nv_dsm_o_dB,'LineWidth',2);
    plot(log10(foff),phase_noise_total_dB,'LineWidth',2);
    hold off
    legend('VCO','PFD+CP','Filter','DSM','Total');
    xlabel('log10(offset freq.)','fontsize',16)
```

```
    ylabel('dBc/Hz','fontsize',16);
    title('Phase noise contribution without ref noise','
       FontSize',16);
subplot(1,3,2)% figure 4
    plot(log10(foff),mag2db(abs(Zlpf)));
    title('LPF transfer function','FontSize',16);
    xlabel('log10(offset freq.)','fontsize',16)
    ylabel('dB(Z_{LPF})','fontsize',16);
subplot(1,3,3)% figure 7
    plot(t*1e+6,dphie,'LineWidth',3);
    title("Settling",'FontSize',16);
    xlabel('t/us','FontSize',16);
    ylabel('\phi_e^d in degree','FontSize',16);
x0=500;
y0=200;
width=1600;
height=600;
set(gcf,'position',[x0,y0,width,height]);

%% Print out some information and results
disp(['-------------------------------------------------']);
disp([9,'       Report']);
disp(['-------------------------------------------------']);
% FMCW parameters
disp([9,'FMCW paramters']);
disp([9,'FMCW bandwidth',9,9,num2str(FMCW_bandwidth/1e+9),9,'
   GHz']);
disp([9,'Chirp duration',9,9,9,num2str(T_chirp/1e-6),9,' us']);
disp([9,'Chirp slope',9,9,9,num2str(chirp_slope/1e+12),9,' MHz/
   us']);
disp([9,'DSM input bit number',9,9,num2str(bit_in)]);
disp([9,'DSM input control word@start',9,num2str(q_N0)]);
disp([9,'DSM input control word@end',9,num2str(q_N1)]);
disp([9,'Frequency step number',9,num2str(N_step)]);
disp([9,'Frequency step/resolution',9,num2str(round(
   freq_resolution/10^6*100)/100),9,'MHz']);
disp(['-------------------------------------------------']);
% PLL parameters
T_step_us=T_step*1e+6;          % T_step in us
T_settling_us=T_settling*1e+6;% T_settle in us
pn_10k=phase_noise_total_dB(1);
pn_1M=phase_noise_total_dB(find(foff==1e+6));
```

```
disp([9,'PLL paramters']);
disp([9,'Step time ',9,9,9,num2str(T_step_us),9,' us']);
disp([9,'Settling time ',9, 9,9,num2str(T_settling_us),9,' us'
   ]);
disp([9,'Mean phase error ',9,9,num2str(phi_e_mean_degree),9,'
   degree']);
disp([9,'Phse noise @10kHz',9,9,num2str(round(pn_10k)),9,' dBc/
   Hz']);
disp([9,'Phse noise @1MHz ',9,9,num2str(round(pn_1M)),9,' dBc/
   Hz']);
disp([9,'PLL bandwidth ',9,9,num2str(PLL_bandwidth/1e+6),9,'
   MHz']);
disp([9,'PLL phase margin ',9,9,num2str(round(PLL_phase_margin)
   ),9,'degree']);
disp(['-------------------------------------------------']);

%% Function definition
% rewrite 3rd-order ODE as 1st-order system
function dydt = odefcn(t,y,b0,b1,b2,a1,a2,Kvco,N0,wref,Kpd,
   alpha)
    dydt=zeros(3,1);
    dydt(1)=y(2);
    dydt(2)=y(3);
    dydt(3)=(   -2*pi*Kvco/a2.*(b2*Kpd*y(3)+b1*Kpd*y(2)+b0*Kpd*
       y(1))+a1/a2.*(2*alpha*wref-2*alpha*y(2)-(N0+alpha*t).*y
       (3))-3*alpha*y(3)   )./(N0+alpha*t);
end
```

Appendix B

Matlab Code for Chirp Calculation

```
clear all
close all
clc

% FMCW chirp parameters
FMCW_bandwidth=8e+9;
chirp_slope=200 *(1e+6/1e-6);    % chirp slope in MHz/us
T_chirp=FMCW_bandwidth/chirp_slope;
fref=2.875e+9;                   % reference frequency
wref=2*pi*fref;
PFD_cycle=1/fref;

N_freq_multi=6;% frequency multiplication ratio after VCO
f1_vco=23e+9;
f2_vco=f1_vco+FMCW_bandwidth/N_freq_multi;
N0=f1_vco/fref;                  % start division ratio=8
N1=f2_vco/fref;                  % end division ratio=8.4638
alpha=(N1-N0)/T_chirp;

bit_in=12;                       % input bit number of DSM
q_N0=ceil((N0-9+0.0001)*2^(bit_in-1));
q_N1=ceil((N1-9)*2^(bit_in-1));
N_step=q_N1-q_N0+1;
T_step=T_chirp/N_step;
N_PFD_cycle=T_chirp/N_step/PFD_cycle;

% PLL circuit block parameters
Icp=150e-6;
```

```
Kpd=Icp/2/pi;
Kvco=50e6;

% LPF
C1=1e-12;
C2=12e-12;
R2=15e+3;

b1=R2*C2;
b0=1;
a2=C1*C2*R2;
a1=C1+C2;


% Some simulation parameters
Np=1000;        % number of time points in one step
%N_step=200;    % for debugging

%
store_tphie=zeros(1,Np*N_step);
store_tfout=zeros(1,Np*N_step);
store_phie=zeros(1,Np*N_step);
store_dphie=zeros(1,Np*N_step);
store_d2phie=zeros(1,Np*N_step);
store_fout=zeros(1,Np*N_step);
store_dfout=zeros(1,Np*N_step);

for k=0:N_step-1          % calculate the time-domian response
   step by step
    Nk=N0+alpha*k*T_step;
    % calculate the initial values for step k
    % varibale x0 is for phie calculation
    % variable y0 is for fout calculation
    if k==0                      % the first step
        x0=[2*pi*0.04   0   0];
        y0=[N0*fref   0];
    else                         % other steps
        Nk_1=N0+alpha*(k-1)*T_step;
        step_term=-2*pi*store_fout(k*Np)*(1/Nk-1/Nk_1);
        % no sudden change
        phie_initial=store_phie(k*Np);
        % sudden change
```

```
        dphie_initial=store_dphie(k*Np)+step_term;
        % no sudden change
        d2phie_initial=store_d2phie(k*Np);
        x0=[phie_initial dphie_initial d2phie_initial];
        fout_initial=store_fout(k*Np);
        dfout_initial=store_dfout(k*Np);
        y0=[fout_initial dfout_initial];
    end
% calculate the time-domian response of phie with 'ode45'
    t=linspace(0,T_step,Np);
    [t,x]=ode45(@(t,x) ode_phie(t,x,a1,a2,b0,b1,Icp,Kvco,Nk),t,
       x0);
% above: function expression definition is at the end of the
   program
    tphie=t;
    % calculate corresponding fout based on phie
    t=linspace(0,T_step,Np);
    [t,y]=ode45(@(t,y) ode_fout(t,y,a1,a2,b0,b1,Icp,Kvco,x(:,1)
       ,x(:,2),tphie),t,y0);
    tfout=t;
    % store the calculated values
    if k==0
        store_tphie(k*Np+1:(k+1)*Np)=tphie;
        store_tfout(k*Np+1:(k+1)*Np)=tfout;
    else
        store_tphie(k*Np+1:(k+1)*Np)=tphie+store_tphie(k*Np);
        store_tfout(k*Np+1:(k+1)*Np)=tfout+store_tfout(k*Np);
    end
    store_phie(k*Np+1:(k+1)*Np)=x(:,1);
    store_dphie(k*Np+1:(k+1)*Np)=x(:,2);
    store_d2phie(k*Np+1:(k+1)*Np)=x(:,3);
    store_fout(k*Np+1:(k+1)*Np)=y(:,1);
    store_dfout(k*Np+1:(k+1)*Np)=y(:,2);
end

% linearity calculation compared with ideal chirp
% ideal fout(t)
tt=linspace(0,T_chirp,length(store_tfout));
fstart=N0*fref;
fout_ideal=fstart+alpha*fref.*tt;
fout_ideal= interp1(tt,fout_ideal,store_tfout);
fout_ideal(end)=fout_ideal(end-1);
```

```
% move the ideal chirp for plotting
store_deltaf=store_fout-fout_ideal;
deltaf=mean(store_deltaf);
fout_ideal_plot=fout_ideal+deltaf;
% calculate FM rms error
% cut_time=0.2*1e-6;  % cut the 'settling' part
cut_time=0.05*T_chirp;% cut the first 5% duration of the chirp
[~,nn]=min(abs(store_tfout-cut_time));
t_cut=store_tphie(nn:end);
fout_cut=store_fout(nn:end);
fout_ideal_cut=fout_ideal(nn:end);
deltaf_cut_sequence=fout_cut-fout_ideal_cut;
deltaf_cut_mean=mean(deltaf_cut_sequence);
FM_rms_error=rms(deltaf_cut_sequence-deltaf_cut_mean);
disp([9,'FM rms error ',9,9,9,num2str(FM_rms_error/1000),'  kHz
   ']);
disp([9,'FM rms error percentage ',9,num2str(FM_rms_error/
   f1_vco*100),'  %']);
% calculate FM peak error
[FM_peak_error,~]=max(abs(deltaf_cut_sequence-deltaf_cut_mean))
   ;
disp([9,'FM peak error  ',9,9,num2str(FM_peak_error/1000),'
   kHz']);

% linearity calculation with

% Plots

figure % phie vs. time
plot(store_tphie/1e-6,store_phie,'LineWidth',2);
xlabel('t/us','fontsize',14)
ylabel('\phi_e','fontsize',14);
title('\phi_e vs. time');

figure % fout vs. time and ideal fout vs. time
plot(store_tfout/1e-6,store_fout/1e9,'LineWidth',2);
hold on
plot(store_tfout/1e-6,fout_ideal_plot/1e9,'LineWidth',2);
legend('Practical chirp','Ideal chirp');
xlabel('t/us','fontsize',14)
ylabel('Output frequency/GHz','fontsize',14);
title('f_{out}(t)');
```

```
figure % deltaf vs. time
plot(store_tfout/1e-6,store_deltaf/1e3,'LineWidth',2);
xlabel('t/us','fontsize',14)
ylabel('f_{out}-f_{out,ideal}/kHz','fontsize',14);
title('Delta f(t)');

figure % fout_cut vs. time and ideal fout_cut vs. time
plot(t_cut/1e-6,fout_cut/1e9,'LineWidth',2);
hold on
plot(t_cut/1e-6,(fout_ideal_cut+deltaf_cut_mean)/1e9,'LineWidth
   ',2);
legend('Practical chirp','Ideal chirp');
xlabel('t/us','fontsize',14)
ylabel('Output frequency/GHz','fontsize',14);
title('f_{out}(t) without beginning settling part');

figure % deltaf_cut vs. time
plot(t_cut/1e-6,(deltaf_cut_sequence-deltaf_cut_mean)/1e3,'
   LineWidth',2);
xlabel('t/us','fontsize',14)
ylabel('f_{out}-f_{out,ideal}/kHz','fontsize',14);
title('Delta f(t)');

% Function definition
% rewrite 3rd-order ODE as 1st-order equation system
function dxdt = ode_phie(t,x,a1,a2,b0,b1,Icp,Kvco,Nk)
    dxdt=zeros(size(x));
    dxdt(1)=x(2);
    dxdt(2)=x(3);
    dxdt(3)=(-1/a2) * (a1*x(3)+b1*Icp*Kvco/Nk*x(2)+b0*Icp*Kvco/
       Nk*x(1));
end
function dydt = ode_fout(t,y,a1,a2,b0,b1,Icp,Kvco,phie,dphie,
   tphie)
    phie= interp1(tphie,phie,t);
    dphie= interp1(tphie,dphie,t);
    dydt=zeros(size(y));
    dydt(1)=y(2);
    dydt(2)=Kvco/a2*Icp/2/pi*(b1*dphie+b0*phie)-a1/a2*y(2);
end
```

Appendix C

Matlab Code for DSM Simulation

```
clear all
close all
clc

%% DSM parameters
osr=128;                        % over-sampling rate
bit_in=12;                      % Bit number of input signal u
bit_ext=11;
bit_internal=bit_in+bit_ext; % Bit number of internal signals
bit_out=1;                      % Bit number of output siganl v

%% Signal initialization
input_type=2;                   % 0: 300kHz sin-wave input
                                % 1: DC input
                                % 2: ramp

if input_type==0                % sin-wave input
    input_freq=300e+3;          % Input frequency
    input_w=2*pi*input_freq; % Input frequency in rad/s
    f_Ny=input_freq*2;          % Nyquist frequency
    fs=f_Ny*osr;                % Sampling frequency
    N_cycle=1000;
    t=0:1/fs:N_cycle/input_freq;% Sampling time points
    in=sin(input_w*t);
    % quantization
    u=zeros(1,length(t));
    in=(in+1)/4;
    u=floor(in/(1/2^(bit_in)));
    u=u-2^(bit_in-2);
```

```
elseif input_type==1          % DC input
    fs=1e+9;                  % Sampling frequency
    t=0:1/fs:1e+6/fs;
    in=-0.2345;               % DC input (between -1<=DC < 1)
    u=q_in*ones(1,length(t));
else                          % ramp input
    FMCW_bandwidth=8e+9;
    chirp_slope=10*1e+6/1e-6;
    T_chirp=FMCW_bandwidth/chirp_slope;
    N_cycle=1; % number of chirp in this simulation
    fs=2.875e+9;
    N_freq_multi=6;% frequency multiplication ratio after VCO
    f1_vco=23e+9;
    f2_vco=f1_vco+FMCW_bandwidth/N_freq_multi;
    fref=2.875e+9;
    N0=f1_vco/fref;
    N1=f2_vco/fref;
    q_N0=ceil((N0-9+0.0001)*2^(bit_in-1));
    q_N1=ceil((N1-9)*2^(bit_in-1));
    q_dN=q_N1-q_N0+1;
    t=0:1/fs:N_cycle*T_chirp;
    t_step=T_chirp/q_dN;
    in_one_chirp=zeros(1,(length(t)-1)/N_cycle);
    for ii=1:(length(t)-1)/N_cycle
        in_one_chirp(ii)=floor(t(ii)/t_step)+q_N0;
    end
    in=repmat(in_one_chirp,1,N_cycle);
    u=zeros(1,length(t));
    u=[in(1) in];             % make it same length as t
end

fs_FFT=fs*2;                  % Sampling frequency for FFT
y1=zeros(1,length(t));
y2=zeros(1,length(t));
y3=zeros(1,length(t));
y4=zeros(1,length(t));
v=ones(1,length(t));          % logic output, -1 or 1
v_real=zeros(1,length(t));
if input_type==0              % moving average filter
    aver_bit_N=floor(osr/10);
end
if input_type==1
```

```
    aver_bit_N=2000;
end
if input_type==2
    aver_bit_N=500;
end
aver_v=zeros(1,length(t)-aver_bit_N+1);
% average output next stage sees
aver_v_real=zeros(1,length(t)-aver_bit_N+1);

%% 2nd-order Delta-sigma modulator operation
y1(1)=u(1)-v(1)*2^(bit_in-bit_out);
y3(1)=y2(1)-v(1)*2^(bit_in-bit_out);
for k=2:length(t)
    y4(k)=y4(k-1)+y3(k-1);
    % quantizer / comparator
    if y4(k)<0
        v(k)=-1;
        v_real(k)=0;
    else
        v(k)=1;
        v_real(k)=2;
    end
    if k>=aver_bit_N
        aver_v(k-aver_bit_N+1)=sum(v(k-aver_bit_N+1:k))/
           aver_bit_N*2^(bit_in-bit_out);   % Note: to compare
           the average output with input, it's multipled by 2^(
           bit_in-bit_out)
        aver_v_real(k-aver_bit_N+1)=sum(v_real(k-aver_bit_N+1:k
           ))/aver_bit_N;                       % the logic
           average output next stage will see
    end
    y1(k)=u(k)-v(k)*2^(bit_in-bit_out);
    y2(k)=y1(k)+y2(k-1);
    y3(k)=y2(k)-v(k)*2^(bit_in-bit_out);
end

% Check overflow of internal signals
overflow_y1=0;
overflow_y2=0;
overflow_y3=0;
overflow_y4=0;
if(max(y1)>2^(bit_internal-1)-1 | min(y1)<-(2^(bit_internal-1)
```

```
   -1))
    overflow_y1=1;
end
if(max(y2)>2^(bit_internal-1)-1 | min(y2)<-(2^(bit_internal-1)
   -1))
    overflow_y2=1;
end
if(max(y3)>2^(bit_internal-1)-1 | min(y3)<-(2^(bit_internal-1)
   -1))
    overflow_y3=1;
end
if(max(y4)>2^(bit_internal-1)-1 | min(y4)<-(2^(bit_internal-1)
   -1))
    overflow_y4=1;
end

%% Plots

if input_type==0
% Spectrum of input and output signals, only for sin-wave input
% Spectrum of output signal v
window_enable=0;% hanning windowing for FFT
v_resample=resample(v,fs_FFT,fs);
if window_enable==1
    v_os=fft(hann(length(v_resample)).*v_resample');% windowing
else
    v_os=fft(v_resample);% no windowing
end
L=length(t)*(fs_FFT/fs);
v_P2=abs(v_os/L);
v_P1=v_P2(1:floor(L/2)+1);
v_P1(2:end-1)=2*v_P1(2:end-1);
f=fs_FFT*(0:L/2)/L;
% Spectrum of quantized input signal u
u_resample=resample(u,fs_FFT,fs);
if window_enable==1
    u_os=fft(hann(length(u_resample)).*u_resample');% windowing
else
    u_os=fft(u_resample);% no windowing
end
u_P2=abs(u_os/L);
u_P1=u_P2(1:floor(L/2)+1);
```

```
u_P1(2:end-1)=2*u_P1(2:end-1);
% Plot both spectrum
figure(1)
subplot(1,3,1)
u_P1_re_db=mag2db(u_P1)-max(mag2db(u_P1));
plot(log10(f),u_P1_re_db);
xlabel('log10(f)','FontSize',12)
ylabel('dB','FontSize',12)
xlim([5 8])
title('Normalized single-sided spectrum of DSM input in dB','
   FontSize',12);
subplot(1,3,2)
v_P1_re_db=mag2db(v_P1)-max(mag2db(v_P1));
plot(log10(f),v_P1_re_db);
xlabel('log10(f)','FontSize',12)
ylabel('dB','FontSize',12)
xlim([5 8])
title('Normalized single-sided spectrum of DSM output in dB','
   FontSize',12);
subplot(1,3,3)
plot(log10(f),u_P1_re_db,'LineWidth',0.8);
hold on
plot(log10(f),v_P1_re_db,'LineWidth',0.8);
legend('Input','Output','FontSize',12);
title('Input and Output spectrum','FontSize',12);
xlim([4 7])
x0=100;
y0=100;
width=1600;
height=600;
set(gcf,'position',[x0,y0,width,height]);
end

% Signals
figure(2)
subplot(6,1,1)
plot(u,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
title('u','FontSize',12);
subplot(6,1,2)
stairs(v,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
```

```
title('v','FontSize',12);
subplot(6,1,3)
plot(y1,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
title('y_1','FontSize',12);
subplot(6,1,4)
plot(y2,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
title('y_2','FontSize',12);
subplot(6,1,5)
plot(y3,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
title('y_3','FontSize',12);
subplot(6,1,6)
plot(y4,'LineWidth',1)
xlim(1e+5/4*[1.5 1.53]);
title('y_4','FontSize',12);
x0=500;
y0=100;
width=600;
height=1000;
set(gcf,'position',[x0,y0,width,height]);

% locolized plot of input vs. moving average of output
figure(3)
subplot(1,3,1)
plot(u,'LineWidth',3);
hold on
stairs(aver_v,'LineWidth',1);
xlim(floor(length(t)/4)+[0 200]);
legend('Input','Moving averge of output','FontSize',12);
subplot(1,3,2)
plot(u,'LineWidth',3);
hold on
stairs(aver_v,'LineWidth',1);
xlim(floor(length(t)/2)+[0 200]);
legend('Input','Moving averge of output','FontSize',12);
subplot(1,3,3)
plot(u,'LineWidth',3);
hold on
stairs(aver_v,'LineWidth',1);
xlim(floor(length(t)/10*9)+[0 200]);
```

```
legend('Input','Moving averge of output','FontSize',12);
x0=500;
y0=100;
width=1600;
height=400;
set(gcf,'position',[x0,y0,width,height]);

figure(4)% plot of input vs. moving average of output
plot(t*10^6,u,'LineWidth',3);
hold on
stairs(t(aver_bit_N:end)*10^6,aver_v,'LineWidth',1);
xlabel('time/us','fontsize',14)
legend('Input','Moving averge of output','FontSize',12);
title('Quantified input vs. moving average of output','FontSize
   ',12);

% plot the average division ratio for ramp input
if input_type==2
    figure(5)
    % aver_v_real is the fractional part of the division ratio
    stairs(t(aver_bit_N:end)*10^6,8+aver_v_real,'LineWidth',1)
    xlabel('time/us','fontsize',14)
    title('Division ratio');
end

%% Print results
if overflow_y1==0 && overflow_y2==0 && overflow_y3==0 &&
   overflow_y4==0
    disp(['Internal signals:  no overflow']);
else
    disp(['Internal signals:  overflow!']);
end
if input_type==1
    disp('Input is DC signal');
    disp(['u',9,9,'=',num2str(u(end))]);
    disp(['v averge',9,'=',num2str(aver_v(end))]);
end
```

Appendix D

Analytical Relationships

The normalized source(drain) currents are given by [79],

$$i_{f(r)} = q_{i,S(D)}^2 + q_{i,S(D)} \tag{D-1}$$

Here $q_{i,S(D)}$ is the normalised terminal charge for source (drain) and is given by [79],

$$q_{i,S(D)} = 0.5\boldsymbol{W_l}(2e^{v_p - V_{S(D)}/\phi_t}) \tag{D-2}$$

Here $\boldsymbol{W_l}(x)$ is the 'Lambert W function' for $x \Rightarrow \boldsymbol{W_l}(x)e^{\boldsymbol{W_l}(x)} = x$, $V_{S(D)}$ = terminal voltage at source (drain) w.r.t bulk terminal (always grounded for NMOS)[79], v_p is a dimensionless quantity and function of gate-to-bulk voltage V_g [79] as described subsequently,

$$v_p = (\Gamma_p - 2\phi_F)/\phi_t - log_e(4m\sqrt{\Gamma_p}/\gamma) \tag{D-3}$$

Here $\gamma = \sqrt{2q\epsilon_{si}N_A}/C_{ox}, \epsilon_{si}$ = Silicon permittivity, q = electronic charge, N_A = channel doping concentration, $\phi_F = \phi_t \ln(N_A/n_i)$, n_i = intrinsic carrier concentration, m and Γ_p are the sub-threshold slope factor and the pinch-off surface potential respectively [79] having expressions,

$$\Gamma_p = V_g - V_{fb} - [\sqrt{\gamma^4/4 + (V_g - V_{fb})^2} - \gamma^2/2] \tag{D-4}$$

$$m = 1 + \gamma/2\sqrt{\Gamma_p + 4\phi_t} \tag{D-5}$$

The sub-threshold slope factor (m) takes the following form (m_0) near Fermi level,

$$m_0 = 1 + \gamma/2\sqrt{2\phi_F} \tag{D-6}$$