الجـمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

وزارة التعليم الـعــالي و البحــث العــلمــي
Ministry of Higher Education and Scientific Research

جــامعة سعد دحلب البليدة
University SAAD DAHLAB of BLIDA

مركز تطوير التقنيات المتقدمة
Center of Development of Advances Technologies

Master Thesis

Option : Electronics
Speciality : Microelectronics

Presented by

HABOUSSI Hamza Zineddine

Design of a CMOS Frequency Divider for Phase-Locked Loop @2.4GHz

Supervisor: **Dr. SLIMANE Abdelhalim**

Co-Supervisor: **Prof. NACER Said**

Dedication

I would like to dedicate this humble work to my parents, expressing my heartfelt gratitude for everything they have done for me. Their unwavering support, love, and belief in me have been invaluable throughout my journey. Thank you, my dear parents.

Acknowledgement

In the name of Allah, the Most Gracious, the Most Merciful,

”He who does not thank the people is not thankful to Allah.” - Prophet Muhammad (peace be upon him)

I begin by expressing my profound gratitude to Allah for His boundless blessings and guidance throughout this journey. Without His divine support, this humble work would not have been possible.

I extend my sincerest appreciation to my esteemed mentor and supervisor, Dr. SLIMANE Abdelhalim, for entrusting me with this incredible opportunity. His unwavering belief in my abilities and constant guidance have been invaluable, and I am deeply grateful for his mentorship.

I would also like to express my gratitude to the entire AMICS team (Analog and Mixed-Signal Integrated Circuits and Systems) and all the members of the DMN (Nanotechnology and Microelectronics Division) at the Center of Development of Advanced Technologies (CDTA). Their support and the conducive work environment they provided played a significant role in the success of my internship.

I would like to express my special appreciation to my co-supervisor, Prof. NACER Said, for his exceptional patience and meticulous organization of the master’s program. His calm demeanor and thoughtful approach to dealing with students are truly admirable.

I am also grateful to Dr. MAHDOUM Ali for his assistance in digital circuits, and for his valuable course that greatly influenced this work.

I extend my heartfelt appreciation to all my professors in the Master's program in Microelectronics at the University of Blida. particularly Prof. AISSAT Abdelkader, Prof. ZERROUK, Prof. ALIANE Ahcene, and Prof. BOUNEMRI Amar. Their unwavering dedication, passion for teaching, and profound knowledge in the field of microelectronics have been instrumental in shaping us as students.

Lastly, thank you to all who have played a part in this endeavor, directly or indirectly. Your contributions have been invaluable in shaping this work, and I am truly grateful for your presence in my life.

ملخص: يقدم هذا العمل تصميم و محاكاة لمقسمي الترددات على عدد صحيح و على عدد كسري، مدمجين في نظام حلقة مقفلة الطور الذي يعمل على تردد ٢.٤ جيجا هرتز. تم اجراء التصميم و التحليل باستخدام برنامج كيدانس فيرتوزو و باستعمال نماذج التصميم سيموس ٩٠ نانومتر. تم استخدام نسخة لغة وصف الأجهزة الخاصة بدوائر التناظرية كاداة لنمذجة السلوكية. تظهر النتائج نجاح التكامل و الوظائف لكلا أنواع المقسمات. تسهم هذه الدراسة في فهم عمل نظام حلقة المقفلة الطور و تسليط الضوء على الاختلافات بين المقسمات ذات العدد الصحيح و ذات العدد الكسري.

كلمات المفاتيح : حلقة مقفلة الطور، مقسم الترددات ذات العدد الصحيح، تقنية سيموس، مقسم الترددات ذات العدد الكسري.

Résumé: Cette thèse présente la simulation et la conception de diviseurs de fréquence sur nombre entier et sur nombre fractionnaire, intégrés dans un système de boucle à verrouillage de phase (PLL) fonctionnant à 2,4 GHz. La conception et l'analyse ont été réalisées à l'aide de Cadence Virtuoso avec le processus CMOS 90nm. Les modèles comportementaux, implémentés en utilisant le langage de description du matériel Verilog-A, ont été utilisés pour tous les autres blocs du système PLL. Les résultats démontrent l'intégration réussie et la fonctionnalité des deux types de diviseurs. L'étude contribue à la compréhension du fonctionnement de la PLL et met en évidence les différences entre les diviseurs entiers et fractionnaires.

Mots clés: système de boucle à verrouillage, Technologie CMOS, PLL à nombre entier, PLL à nombre fractionnaire, Divideur de fréquence, Étude comportementale, Verilog-A, Cadence Virtuoso.

Abstract: This thesis presents the simulation and design of frequency dividers over integer number and over fractional number, integrated within a Phase-Locked Loop (PLL) system operating at 2.4GHz. The design and analysis were performed using Cadence Virtuoso with the CMOS 90nm process. The behavioral models, implemented using the hardware description language Verilog-A, were utilized for all other blocks of the PLL system. The results demonstrate the successful integration and functionality of both types of dividers. The study contributes to the understanding of PLL operation and highlights the differences between Integer-N and Fractional-N dividers.

Keywords: Phase-Locked Loop, CMOS Technology, Frequency Divider, Integer-N PLL, Fractional-N PLL, Behavioral modeling, Verilog-A, Cadence virtuoso.

Abbreviations and symbols list

CMOS	Complementary Metal-Oxide-Semiconductor
C_L	Load capacitance
C_{ox}	Oxide capacitance
CP	Charge Pump
H(s)	Closed-loop transfer function
LPF	Low-Pass Filter
L	Transistor length
PFD	Phase-Frequency Detector
PLL	Phase-Locked Loop
R_{eqn}	NMOS equivalent resistance
R_{eqp}	PMOS equivalent resistance
R_{on}	ON Resistance
TSPC	True Single-Phase Clock
VCO	Voltage-Controlled Oscillator
VDD	Supply voltage
V_{GS}	Gate-Source Voltage
V_{th}	Threshold voltage
W	Transistor width
XOR	Exclusive OR
ω_n	natural frequency
K_{VCO}	Voltage-Controlled Oscillator gain
ω_u	Open-loop unity gain bandwidth
ζ	Damping factor

μ_n electrons mobility	Output frequency ω_{out}
ω_{ref}	Reference frequency
ω_{fb}	feedback frequency
$\Delta\phi$	Phase difference
$\Delta\Sigma$	Delta-Sigma

Contents

1 Basics of Analog Phase-Locked Loop 17

1.1 Definition 17

1.2 Basics 17

1.3 Main Blocks 19

1.3.1 Phase Detector 19

1.3.2 XOR Gate as Phase Detector 19

1.3.3 Phase/Frequency Detector 21

1.3.4 Charge Pump 21

1.3.5 Charge Pump with Phase Frequency Detector . 22

1.3.6 Loop Filter 23

1.3.7 Voltage-Controlled Oscillator 24

1.3.8 Performance Parameters of Voltage Controlled Oscillator 26

1.3.9 Divider 27

1.4 Applications 28

1.4.1 Clock Recovery 28

1.4.2 Deskewing 28

1.4.3 Clock Generation 29

1.4.4 Clock Distribution 29

1.4.5 Frequency Synthesis 30

1.5 Conclusion 30

2 Frequency Dividers **32**
2.1 Integer-N Divider 33
2.1.1 Latch-Based Design 33
2.1.2 Dual-Modulus Prescalers 36
2.1.3 Drawbacks of Integer-N Dividers 39
2.2 Fractional-N Divider 40
2.2.1 Basics 40
2.2.2 The Idea Behind $\Delta\Sigma$ Fractional-N PLLs 41
2.2.3 Delta-Sigma Modulation Overview 45
2.2.4 Conclusion 47

3 Results and Discussion **48**
3.1 Simulation of Main Blocks 49
3.1.1 Phase-Frequency Detector 49
3.1.2 Charge Pump 53
3.1.3 Voltage-Controlled Oscillator 54
3.1.4 Integer-N Divider 61
3.1.5 Fractional-N Divider 67
3.2 Integer-N PLL 71
3.2.1 Behavioral modeling of the integer-N PLL . . . 73
3.2.2 Integration Approach for the schematics of the integer-N divider in PLL 76
3.3 Fractional-N PLL 78
3.4 Comparison of Integer-N and Fractional-N PLLs 80
3.5 Conclusion and future work 81

A Verilog-A Codes **84**
A.1 Phase-Frequency Detector code 84
A.2 Charge pump code 85
A.3 Voltage-Conctrolled Oscillator code 86
A.4 Integer-N divider code 87

A.5 Fractional-N divider code 87
A.5.1 Prescaler code 87
A.5.2 Difference amplifier code 88
A.5.3 Integrator code 89
A.5.4 Quantizer code 89
A.5.5 Digital-to-Analog code 90

List of Figures

1.1 Basic phase-locked loop. 18
1.2 Characteristics of the Phase Detector. 20
1.3 Block Diagram of exclusive OR gate (XOR). 20
1.4 Wave-forms of XOR operate as Phase Detector 20
1.5 Block Diagram of Phase/Frequency Detector. 21
1.6 Charge Pump. 22
1.7 Phase Frequency detector with a charge pump and a loop filter. 23
1.8 Output of Phase Frequency Detector. 23
1.9 First order RC filter. 24
1.10 Block diagram of VCO. 24
1.11 Linear characteristics of linear Voltage Controlled Oscillator. 25
1.12 Non-linear characteristics of the VCO. 27
1.13 Block diagram of divider. 28

2.1 Latch-based frequency divider. 33
2.2 Frequency divider based on **(a)** TSPC logic and **(b)** transmission-gate based logic [4]. 34
2.3 Schematic of the CML latch with resistvie load [4]. . . 34
2.4 **(a)** A dynamic latch with a single clocked device, **(b)** cascaded TSPC stages, and **(c)** a three-stage master-slave flip-flop operating as a frequency divider [8]. . . . 35
2.5 **(a)** divide-by-3 circuit and **(b)** its wave-forms [10]. . . 36

2.6 Dual modulus ÷2/3 circuit [10]. 37
2.7 Dual modulus ÷3/4 circuit [10]. 38
2.8 **(a)** divide-by-5 circuit and **(b)** its wave-forms [10]. . . 38
2.9 Block diagram of fractional-N PLL [12]. 41
2.10 A fractional-N PLL that generates non-integer multiples of the reference frequency, but has phase noise consisting of large spurious tones [3]. 42
2.11 A fractional-N PLL that generates non-integer multiples of the reference frequency, but has large amount of in-band phase noise [3]. 43
2.12 A $\Delta\Sigma$ fractional-N PLL example [3]. 44
2.13 (a) A simple first order $\Delta\Sigma$ modulator with a 1-bit quantizer, (b) input and output waveforms [7]. 45
2.14 A 9-level quantizer example [3]. 46
2.15 A $\Delta\Sigma$ modulator example [3]. 46

3.1 Waveforms of the behavioral simulation of the PFD in two input conditions **(a)** when the reference signal is leading **(b)** when the feedback signal is leading. 50
3.2 The output voltage versus the phase variation. 50
3.3 Modified version of the conventional PFD. 51
3.4 The E-TSPC topology. 52
3.5 Simulation results of PFD using E-TSPC topology **(a)** triggering the UP signal **(b)**triggering the DOWN signal. 53
3.6 Output voltage versus the phase variation illustrating a dead zone at π. 53
3.7 Test bench of the charge pump. 54
3.8 Waveforms of the charge pump output when **(a)** there is an UP signal resulting in charging the LPF **(b)** there is a DOWN signal resulting in discharging the LPF. . . 54
3.9 Oscillation results of the behavioral model of the VCO. 55

3.10 Frequency-Versus-Voltage relationship of the behavioral model of the VCO. 56
3.11 Diagram of 5 stages ring oscillator. 57
3.12 The implemented 5 stages ring oscillator using current-starved technique to control the frequency. 59
3.13 Oscillation results of our ring oscillator. 60
3.14 Frequency-Versus-Voltage relationship of the ring oscillator. 60
3.15 Simulation results of the behavioral model of the integer divider by 120 **(a)** Input-Output waveforms **(b)** zoomed view of the input signal. 61
3.16 Simulation results of the TSPC D-Flip Flop**(a)** Its schematics **(b)** Its Input-output waveforms. 63
3.17 Simulation results of the divide-by-2 topology **(a)** Its schematics **(b)** Its Input-output waveforms. 64
3.18 Simulation results of the divide-by-3 topology **(a)** Its schematics **(b)** Its Input-output waveforms. 66
3.19 Simulation results of the divide-by-5 topology **(a)** Its schematics **(b)** Its Input-output waveforms. 67
3.20 Simulation results of the divide-by-120 topology **(a)** Its schematics **(b)** Its Input-output waveforms **(c)** zoomed view of the input signal. 68
3.21 1-bit $\Delta\Sigma$ modulator **(a)** Its schematics **(b)** Its Input-output waveforms. 69
3.22 Simulation results of the fracitnal divider **(a)** Its test bench **(b)** Its Input-output waveforms **(c)** frequency-versus-time characteristics of the VCO output signal and the presclaer output signal. 71
3.23 Simulation results of the behavioral modeling of the integer-N PLL **(a)** test bench **(b)** Waveforms **(c)** zoomed view of the waveforms at locking state. 75

3.24 Voltage control graph of our behavioral model PLL. . . 76
3.25 Effect of the charge pump current on the loop response. 77
3.26 Simulation results of the integer-N divider schematics within the PLL **(a)** waveforms **(c)** zoomed view of the waveforms at locked state. 78
3.27 Workspace of the fractional-N PLL. 78
3.28 Simulation results of the fractional-N PLL **(a)** test bench **(b)** Frequency variations of: VCO output frequency, reference frequency, fractional divider frequency. 79

List of Tables

3.1 Selected ratios for the TSPC D-Flip Flop circuit. . . . 62
3.2 Settling time comparison between Integer-N PLL and Fractional-N PLL. 80

Introduction

Phase-locked loops (PLLs) play a crucial role in various electronic systems, particularly in communication and signal processing applications. They provide stable and synchronized output signals that are essential for reliable operation. This thesis aims to investigate and design two types of frequency dividers, namely Integer-N and Fractional-N, integrated within a PLL system operating at a frequency of 2.4 GHz. The thesis is structured as follows:

Chapter 1 provides a comprehensive overview of the basics of analog phase-locked loops. It discusses the fundamental principles of PLL operation, including phase comparison, frequency division, and feedback control. The main building blocks of a PLL, such as the phase-frequency detector (PFD), charge pump (CP), voltage-controlled oscillator (VCO), and frequency divider, are thoroughly explained. The chapter serves as a foundation for understanding the subsequent chapters.

Chapter 2 focuses specifically on the topic of frequency dividers. It begins by exploring the concept of integer dividers and their implementation using flip-flops and latches. The functionality and design considerations of integer-N dividers are discussed in detail. Subsequently, the chapter delves into fractional dividers and introduces the concept of fractional-N frequency synthesis. The use of $\Delta\Sigma$ modulation techniques for achieving fractional division is explained, highlighting their advantages and limitations.

In Chapter 3, the results of the simulation and design of the integer-N and fractional-N frequency dividers within the PLL system are presented and discussed. Various performance metrics, such as settling time, power consumption and frequency stability, are evaluated. Simulation graphs and waveforms are analyzed to assess the effectiveness of the implemented dividers. The obtained results are compared and interpreted, shedding light on the similarities and differences between integer-N and fractional-N dividers. Furthermore, the implications of the findings on the overall performance and stability of the PLL system are discussed.

Appendix A includes the Verilog-A codes for the behavioral models developed in this study, providing a valuable resource for further analysis and replication of the proposed designs.

Chapter 1

Basics of Analog Phase-Locked Loop

1.1 Definition

A Phase-Locked Loop is a crucial component in contemporary electronic systems. It functions as control system that produces an output signal with a phase that corresponds to the phase of the reference input signal. The PLL operates as closed-loop feedback system where the output phase is compared to the input one, generating a phase error. This latter impacts the oscillator to adjust its oscillation frequency.

Phase-Locked Loops have found extensive applications in various fields such as radio systems for telecommunications, frequency synthesizers, clock synchronization, computing and microprocessor clock generation, and several other electronic applications.

1.2 Basics

Figure 1.1 shows a block diagram of the basic PLL, in which a synchronized oscillator, usually a voltage-controlled oscillator (VCO), gener-

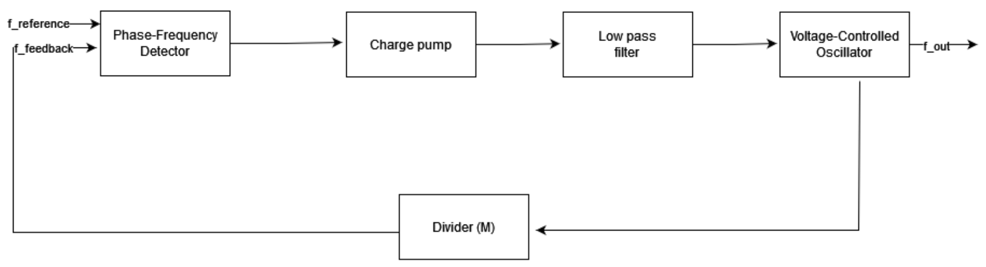


Figure 1.1: Basic phase-locked loop.

ates an output frequency that changes in response to a control voltage. The Phase Detector compares the VCO's output to the reference signal and produces an error signal, which changes proportionally to the phase between them.

To ensure that the output frequency stays locked to the reference frequency, the error signal is fed through a charge pump that converts it into a corresponding current. The current charges or discharges a capacitor, depending on the polarity of the error signal. This charge/discharge cycle generates a voltage at the output of the charge pump, which is then filtered by the loop filter.

The loop filter processes this filtered voltage to generate the oscillator control voltage. Although the loop filter can be simple low pass filter, it is typically designed to provide some advantageous response characteristic [2].

Finally, the oscillator control voltage is fed to the VCO, which adjusts the output frequency accordingly. If the output frequency is higher than the reference frequency, than the error signal would necessarily decrease with time, bringing the output frequency down toward the input frequency. Similarly, if the output frequency is lower than the reference frequency, then the error signal would increase with time, driving the output frequency up toward the input frequency. Thus the

PLL provides a negative feedback loop that keeps the output frequency locked to the reference frequency, with the help of the charge pump and the divider.
In summary the PLL consists of five blocks:
Phase Detector: To Compare the difference between the VCO frequency and the reference frequency.
Charge Pump: To control the charging and the discharging current of the loop filter.
Loop Filter: To Generate a stable control voltage for the VCO.
Voltage-Controlled Oscillator: To Generate a frequency based on the control voltage.
Divider: Divide the output frequency of the VCO down to the reference frequency.

1.3 Main Blocks

1.3.1 Phase Detector

The Phase Detector is a circuit that compare the phases of the reference signal and the feedback signal and generates an output signal that is linearly proportional to the phase error. Ideally the average output V_{out} is directly proportional to the phase difference $(\Delta\phi)$ between the two input signals V_1 and V_2. Figure 1.2 Depicts the characteristics of a phase detector, illustrating the slope of the K_{pd}, which represents the gain of the phase detector measured in volts per radian. Notably the gain of the phase detector is zero when the phase difference $(\Delta\phi)$ between the two inputs is zero.

1.3.2 XOR Gate as Phase Detector

A Phase Detector can be realized by means of an exclusive OR (XOR) gate. Figure 1.3 shows the exclusive OR gate.

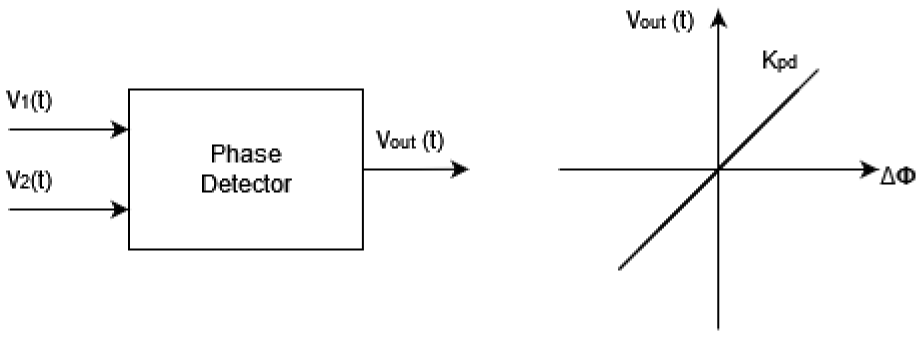


Figure 1.2: Characteristics of the Phase Detector.

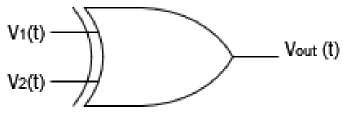


Figure 1.3: Block Diagram of exclusive OR gate (XOR).

The XOR gate produces output signal $V_{out}(t)$ on both the rising and falling edges when the two input signals have a phase difference. Figure 1.4 depicts the waveform of the XOR gate functioning as a phase detector.

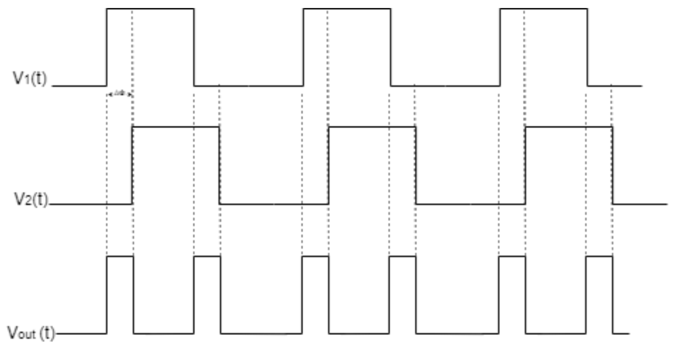


Figure 1.4: Wave-forms of XOR operate as Phase Detector

The XOR gate take the two input signals $V_1(t)$ and $V_2(t)$ and generates the phase difference $(\Delta\phi)$. The Phase Detector is related to the width of the output pulses and is proportional to the phase difference.

1.3.3 Phase/Frequency Detector

One of the critical PLL building blocks is the Phase/Frequency Detector. The conventional Phase/Frequency Detector block diagram is shown in Figure 1.5. As can be seen, the frequency information is coded at the output of a PFD that uses two D-Flip-Flops (DFF).

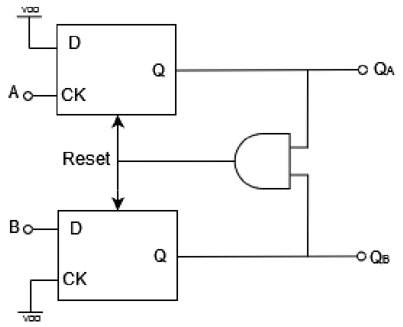


Figure 1.5: Block Diagram of Phase/Frequency Detector.

The DFF outputs are denoted Q_A (or up) and Q_B (or down). Assuming both outputs are initially low, a rising edge on the A input(reference) causes the UP signal to go high. This indicates that the VCO frequency needs to be increased in order to match the input. Conversely, when the input(B) transitions high, the Q_B output rises. signaling the need to decrease the VCO frequency to match the input. The AND gate generates a reset signal to return the PFD the zero state when both output switch high [1].

1.3.4 Charge Pump

A Charge Pump is a circuit that sources or sinks charges for controlled amount of time. Shown in Figure 1.6 is a simple realization: if S_1 is on, I_1 charges the capacitance of the loop filter, and if S_2 is on, I_2 discharges it.

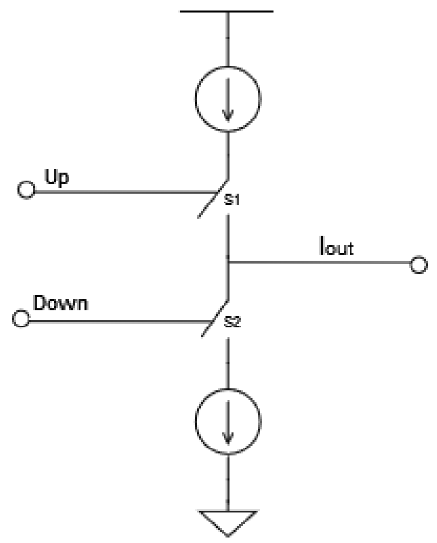


Figure 1.6: Charge Pump.

The controls are called Up and Down, respectively, as they determine whether the output voltage rises or falls [10].

1.3.5 Charge Pump with Phase Frequency Detector

Complete operation of a charge pump with a phase frequency detector is shown in figure Figure 1.7. when signal A leads signal B by a finite amount, the output signal Q_A is generated and Q_B is almost zero. Q_A will turn on the switch S_1, it will charges the capacitor C_p and Q_B will turn off the switch S_2 [9].
The charging of the capacitor, increases the control voltage V_c. Due to increase in V_c, the oscillation of the VCO also increase. The output of phase frequency detector is shown in Figure 1.8.

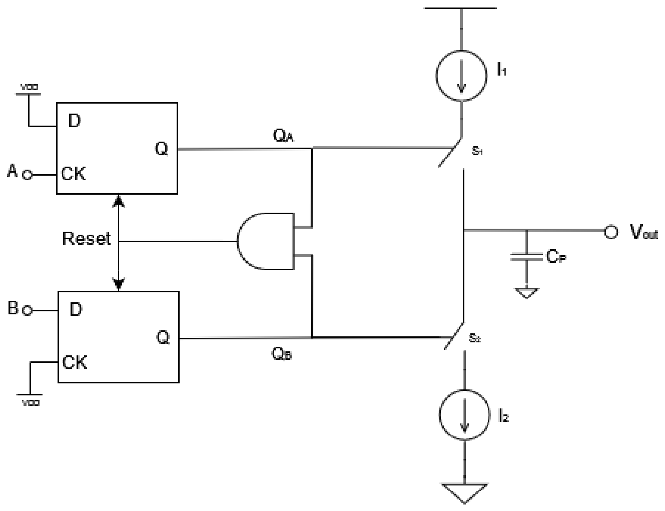


Figure 1.7: Phase Frequency detector with a charge pump and a loop filter.

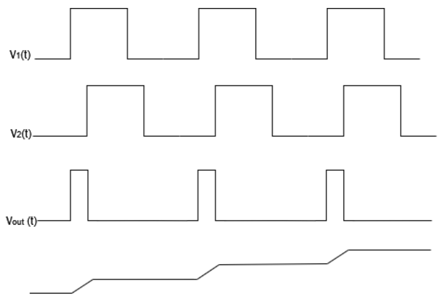


Figure 1.8: Output of Phase Frequency Detector.

1.3.6 Loop Filter

A Loop Filter is a combination of a simple capacitor with a resistor as shown in Figure 1.9. The architecture of the loop filter depends on the order of filter.
VCOs are typically regulated by voltage rather than current. Therefore, in charge pump PLL, the loop filter's role is to convert the output current of the charge pump into the VCO control voltage. Additionally, a low pass filter is necessary to prevent pulses from being fed into the VCO. The design of the loop filter is crucial in determining its key

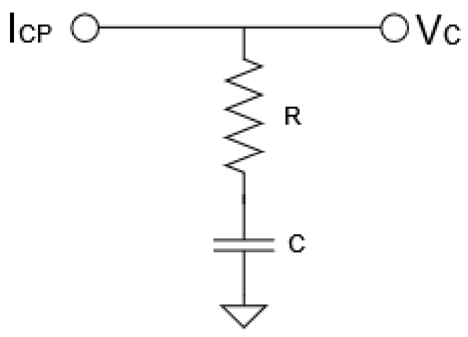


Figure 1.9: First order RC filter.

characteristics, such as loop bandwidth, settling time, and phase noise. These characteristics are highly dependent on the specific design of the loop filter.

1.3.7 Voltage-Controlled Oscillator

The output frequency of the Voltage Controlled Oscillator (VCO) ω_0 is linearly proportional to the control voltage (V_C). It is generated by the phase detector. This linear relation between control voltage and the output frequency simplifies the PLL design. The Voltage Controlled Oscillator is the heart of the PLL. The block diagram of a VCO is shown in Figure 1.10.

In order to generate the necessary control voltage, the phase detector

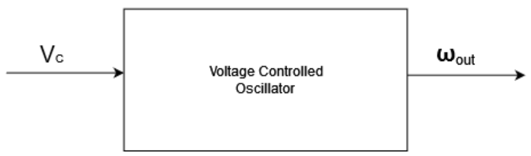


Figure 1.10: Block diagram of VCO.

must induce a phase error based of its characteristic. The figure 1.11

demonstrates the typical characteristics of a voltage controlled oscillator.

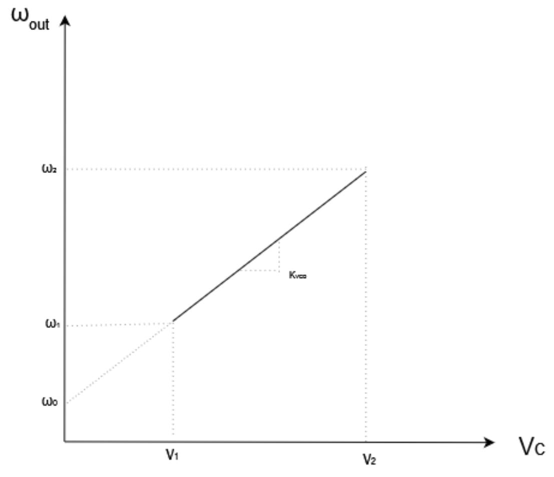


Figure 1.11: Linear characteristics of linear Voltage Controlled Oscillator.

The output of the VCO can be formulated by the following relation:

$$\omega_{out} = \omega_0 + K_{VCO}.V_C \tag{1.1}$$

here ω_0, the intercept to V_C equals to zero. K_{VCO} represents the gain or sensitivity of the VCO.It is expressed in rad/s/V. $\omega_2 - \omega_1$ is achievable tuning range of the frequency. [9]
The Voltage Controlled is fed as the input of the VCO. so we can write V_1 as:

$$V_1 = (\omega_1 - \omega_0)/K_{VCO} \tag{1.2}$$

and

$$\phi_0 = V_1/K_{PD} = (\omega_1 - \omega_0)/K_{VCO}.K_{PD} \tag{1.3}$$

In Equation (1.3), there are two key points illustrated: Firstly, the phase error changes as the input frequency varies. Secondly, in order to minimize the phase error, K_{PD} and K_{VCO} should be maximized.

1.3.8 Performance Parameters of Voltage Controlled Oscillator

Center Frequency

The Center Frequency of VCO is determined by the Phase Detector and the transfer function that links it to the VCO output. While the center frequency is typically defined as the frequency at which the phase detector is in the center of its output range, the output may not be zero at that point, depending on the phase detector implementation. To ensure proper operation of VCO, a voltage shift may be introduced between the phase detector and the VCO, although this not always necessary. The present day CMOS VCOs center frequencies can be high as 10GHz [2].

Tuning Range

The tuning range refers to the range of frequencies that the PLL can track or lock onto. This range typically determined by the design of the loop filter and the characteristics of the VCO. The tuning range is an important parameter in the PLL design, as it determines the range of frequencies that the PLL can operate over. A wider tuning range allows the PLL to operate over a broader range of frequencies, while narrower tuning range provides better frequency stability and noise performance within a smaller frequency band.

Tuning Linearity

Due to non-idealities, the tuning characteristics of the VCO are non-linear. Although the gain of the VCO is generally constant, the non-linearity can negatively impact the settling behaviour of the PLL. Therefore, it is important that the gain variation remains small throughout the tuning range. A graphical representation of the non-linear characteristics of the VCO is shown in Figure 1.12, where the VCO exhibits a high gain in the middle of the range and low gain at the two extremes [9].

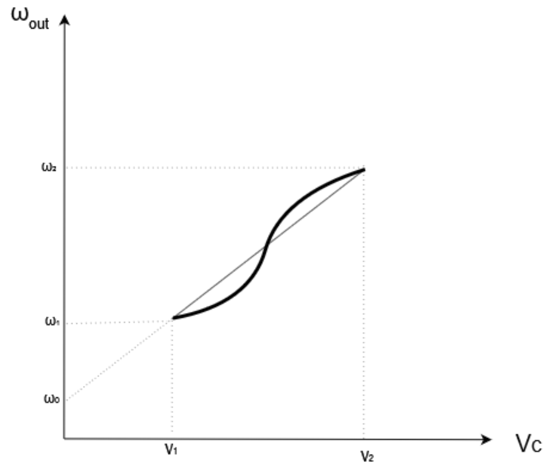


Figure 1.12: Non-linear characteristics of the VCO.

Power Dissipation

The Power Dissipation of a VCO is a crucial factor that affects its oscillations, along with speed and noise trade-offs. Typically, the power consumption of an oscillator ranges from 1 to 10 mV [9].

1.3.9 Divider

The oscillator's output frequency, F_{OUT}, is sent to the input of a divider that divides it by a factor of M, resulting in the divider output $F_{divider} = F_{OUT}/M$. $F_{divider}$ is then sent to the input of the phase detector, where it is compared to the reference frequency F_{ref}, generating an output. The PLL considered to be locked when $F_{divider}$ and $F_{ref}$are equal. Figure 1.13 shows the block diagram of a divider.

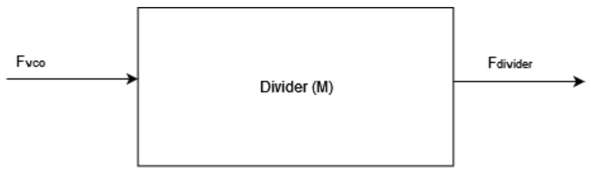


Figure 1.13: Block diagram of divider.

The relation between the output and the input of the divider can be formulated as:

$$F_{OUT} = M.F_{divider}. \tag{1.4}$$

where F_{OUT} is the oscillator's output frequency, $F_{divider}$ is the output frequency of the divider and M is the division ratio.

1.4 Applications

1.4.1 Clock Recovery

In some cases, data streams, particularly those with high speed, may not be transmitted with an associated clock signal, such as the raw data stream from a disk drive's magnetic head. In these situations, the receiver must generate a clock signal from an estimated frequency reference and synchronize with the transitions in the data stream using a PLL, a process known as clock recovery. The PLL must correct any drift in its oscillator, which necessitates a transition in the data stream occurring frequently enough. To restrict the maximum time between transitions, a line code like 8b/10b encoding is frequently employed [13].

1.4.2 Deskewing

When sending data in parallel with a clock, there may be a delay between the detected clock edge and the received data window due to the need to receive and amplify the clock signal. This delay, which is process-, temperature-, and voltage-dependent, limits the frequency at which data can be transmitted. To eliminate this delay, a deskew PLL can be used on the receive side, which matches the phase of each data flip-flop to the received clock. In this scenario, a delay-locked loop (DLL), which is a special form of a PLL, is commonly used [13].

1.4.3 Clock Generation

Electronic systems often require processors that operate at high frequencies, which are much higher than the frequencies that crystal oscillators can provide. In such cases, the processor clocks are generated by phase-locked loops (PLLs) that use a lower-frequency reference clock, typically around 50 or 100 MHz, and multiply it up to the operating frequency of the processor. For instance, if the operating frequency is multiple gigahertz and the reference crystal provides a frequency of just tens or hundreds of megahertz, the multiplication factor can be quite large [13].

1.4.4 Clock Distribution

Clock distribution is another application of PLL where a single high-frequency clock signal is generated and then distributed to multiple components of a system with precise timing. PLLs are used to synchronize the frequency and phase of the clock signal across multiple components, ensuring that they are all operating at the same frequency and with a consistent phase relationship. This helps to avoid issues such as timing skew and jitter that can cause errors and reduce the performance of the system [13].

1.4.5 Frequency Synthesis

Frequency synthesis is another important application of PLLs. In frequency synthesis, a PLL is used to generate an output signal with a frequency that is a multiple of a reference frequency. This is commonly used in communication systems, where the frequency of the signal being transmitted needs to be precisely controlled. PLLs can also be used for frequency modulation and demodulation, and in frequency synthesizers for music and sound systems [13].

1.5 Conclusion

In summary, the chapter discusses the Phase-Lock Loop (PLL) and its main building blocks, including the phase detector, charge pump, loop filter, voltage controlled oscillator and divider. These building blocks work together to generate an output signal that is locked in frequency and the phase to the input signal. The chapter also explores that application of PLLs, such as clock recovery, deskewing, clock generation, clock distribution, and frequency synthesis. The performance parameters of the VCO are critical to the overall performance of the PLL. In conclusion, PLLs are important building blocks in many electronic systems, and their applications are wide-ranging.

Chapter 2

Frequency Dividers

In the previous chapter, We learned that Phase-Locked Loop (PLL) play critical role in modern electronic systems, enabling frequency synthesis, clock generation, and synchronization. PLLs rely on frequency dividers to scale down the output of the VCO to match the reference frequency, which typically more stable and accurate. However, not all frequency dividers are created equal, and their design depends on the specific application requirements [10].

In this chapter, we will focus on the design and implementation of the two types of frequency dividers: integer and fractional. Integer dividers are simply and widely used, but suffer from limited frequency resolution and spurious signals that can degrade the system performance. We will analyze the drawbacks of the integer dividers and explain how they can mitigated through the use of fractional dividers.

Fractional dividers, on the other hand allow for finer frequency resolution and better spurious performance, and making them ideal for application that require high accuracy and stability, such as RF synthesizers. We will introduce the principles of $\Delta\Sigma$ modulation, a common technique used to implement fractional dividers.

2.1 Integer-N Divider

2.1.1 Latch-Based Design

One simple way to realize a divide-by-2 (/2) frequency divider is to use a Latch where its output is tied to the D input resulting in a feedback loop, as shown in Figure 2.1.

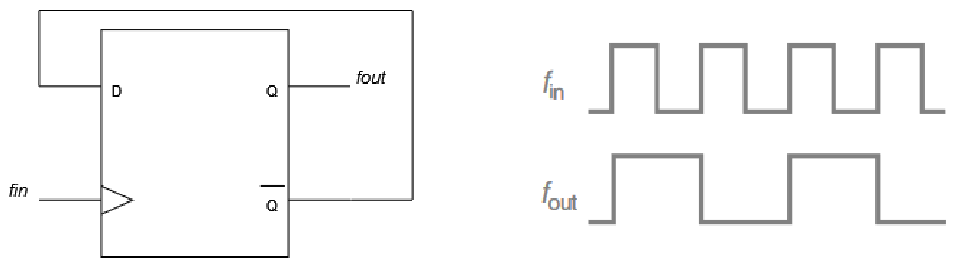
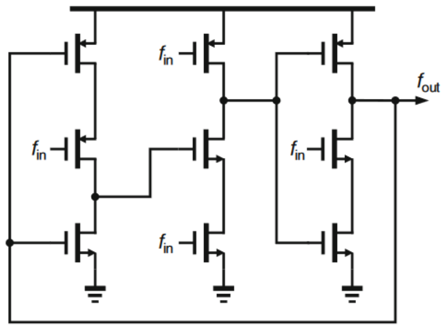


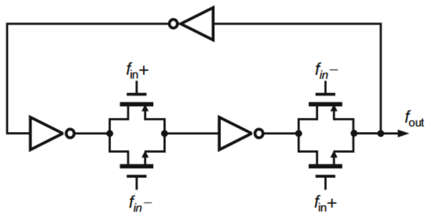
Figure 2.1: Latch-based frequency divider.

The Choice of circuit implementation of the latch depends on the input frequency and the CMOS technology. For example, in a 65-nm CMOS process, that latch based on static logic functions well for input frequency below $\sim 100MHz$. As the input frequency increases to a range from several hundreds of MHz to several GHz, the latch based on dynamic logic such as the true single-phase clocking (TSPC) logic and the transmission-gate-based logic can be employed to achieve high-speed operation with low power consumption as shown in Figure 2.2(a) and (b), respectively [4].

To expand the input frequency range, the latch based on current-mode logic (CML) can be employed. Figure 2.3 shows the CML latch with resistive load. Operating in the current mode, CML dividers can



(a)



(b)

Figure 2.2: Frequency divider based on **(a)** TSPC logic and **(b)** transmission-gate based logic [4].

achieve much faster speed than the both TSPC and transmission-gate-based dividers. Moreover, CML dividers do not require input signals with full swings, which renders them more suitable for high-frequency applications. The maximum operating frequency that CML dividers can achieve depends on the loading capacitance and the required output swing. Typically in a 65-nm CMOS process, CML dividers can function well at input frequencies up to 10 GHz [4].

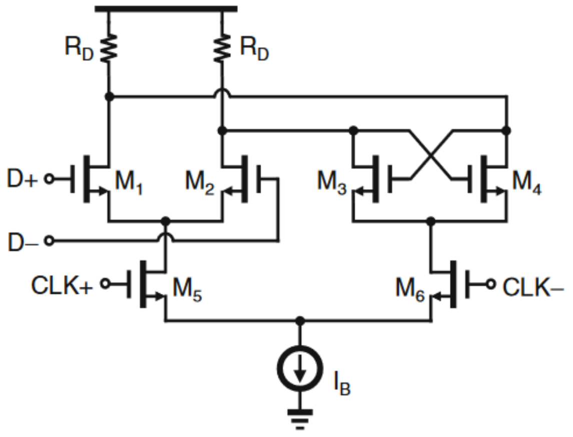


Figure 2.3: Schematic of the CML latch with resistvie load [4].

TSPC Principles

In Figure 2.4(a) when CK is high, the latch reduces to an inverter and operates properly. When CK is low, the circuit is in the store mode and retains the output state if A does not change or has only a low-to-high transition. if we precede this structure with a Domino stage that incorporates N-type logic [Figure 2.4(b)], we cascade is not. Specifically, suppose the second stage must store a ONE(so that X=ZERO). For this state to be overwritten when CK is high, A_2 must rise, which is not possible because M_6 is off. Similarly if the second stage is storing a ZERO, only a fall in A_1 can overwrite it, which cannot occur because M_6 is off.

In addition to less hardware and power, TSPC logic also affords designs having lower phase noise. With fewer transistors and faster transitions in the signal path, TSPC techniques lead to less phase noise in circuits such as frequency dividers and phase/frequency detectors (PFDs) [8].

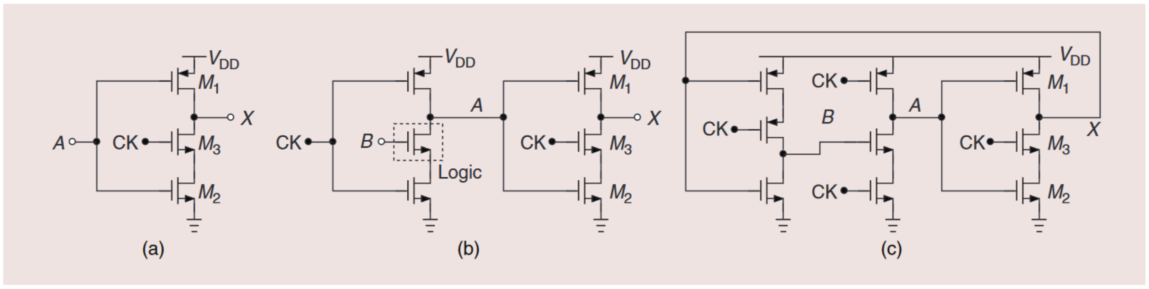


Figure 2.4: **(a)** A dynamic latch with a single clocked device, **(b)** cascaded TSPC stages, and **(c)** a three-stage master-slave flip-flop operating as a frequency divider [8].

2.1.2 Dual-Modulus Prescalers

In some applications, a feedback divider is required to have a modulus that changes in increment of unity, such as N to N+1. To achieve this, high-speed dividers called dual modulus "prescalers" are used, which can switch between two different moduli such as 2 and 3, or 3 and 4 and so on, These circuits are commonly referred as "÷2/3" and "÷3/4" arrangements, respectively [10].

Divide-by-3 circuit

An example of ÷3 circuit is shown in Figure 2.5(a), where two D flip-flops cycle through the three necessary states.

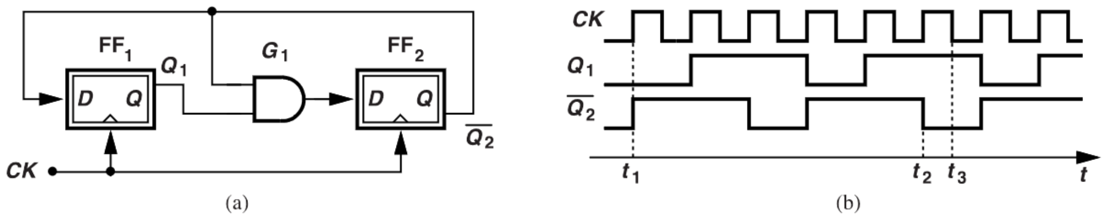


Figure 2.5: **(a)** divide-by-3 circuit and **(b)** its wave-forms [10].

It is worth noting that the next value of Q_2 is equal to $\overline{Q_1.Q_2}$. By utilizing the wave-forms illustrated in Figure 2.5(b) and assuming that the flip-flops switch their output on the rising edge of the clock, we observe that if $Q_1\overline{Q_2} = 00$ and CK goes high (at t_1), Q_1 remains at ZERO while $\overline{Q_2}$ rises to ONE. During the next clock cycle, Q_1 takes on the value of $\overline{Q_2}$ and goes high, while $\overline{Q_2}$ remains unchanged. As a result, $Q_1\overline{Q_2} = 11$, while produces a ONE at the AND output, necessitating that the next value of $\overline{Q_2}$ be ZERO. In third clock cycle, Q_1 remains high while $\overline{Q_2}$ falls. It's worth nothing that the state $Q_1\overline{Q_2} = 00$ does not occur again. The duty cycle of the two outputs is 1/3, which is a useful characteristic in some applications but undesired in others [10].

Dual modulus ÷2/3 circuit

The ÷3 configuration that is presented in Figure 2.5(a) can be modified to include ÷2 mode, as demonstrated in Figure 2.6.

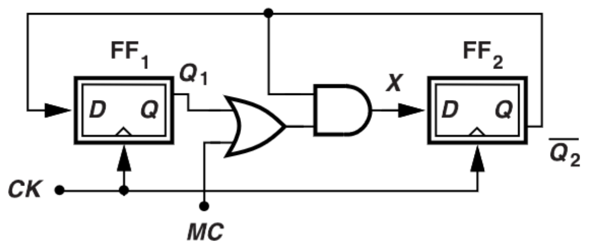


Figure 2.6: Dual modulus ÷2/3 circuit [10].

Essentially, when a division by 2 is required, the circuit reduces to a single flip-flop. if the input of called ”modulus control” (MC) is set to low, Q_1 passes through the OR gate, and the circuit operates as a ÷3 divider. Conversely, if MC is high, Q_1 is rendered ineffective, and the AND gate directs Q_2 to the output X, transforming FF_2 into a ÷2 circuit [10].

Dual modulus ÷3/4 circuit

In a similar fashion, one may consider the implementations of ÷3/4 circuits, which can be derived from the ÷3 circuits, and still require only two flip-flops. To achieve this, a modification is required such that when MC is high, the structure reduces to a ÷4 circuit, i.e., two flip-flops in a loop. Figure 2.7 illustrates such an arrangement, where OR gate is inserted in the loop around FF_2. When $MC = 0$, the circuit divides by 3, but when MC=1, the OR gate is included, and the circuit acts as two flip-flops in a loop, resulting in division by 4. However, the additional gates and the longer delay that results from them reduce the maximum operating speed [10].

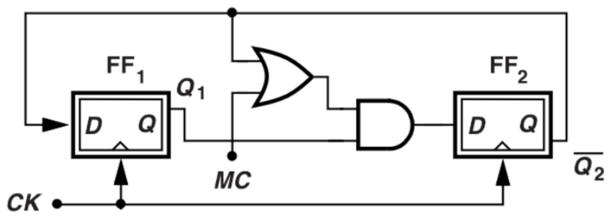


Figure 2.7: Dual modulus ÷3/4 circuit [10].

Divide-by-5 circuit

The Dual modulus dividers discussed earlier operate synchronously, meaning their flip-flops are clocked simultaneously. However, for larger divide ratios, we can combine one of these prescalers with additional asynchronous dividers. For instance, Fig 2.8(a) shows how a ÷2/3 circuit(Div23) is followed by an asynchronous ÷2 stage to create a ÷5 divider.

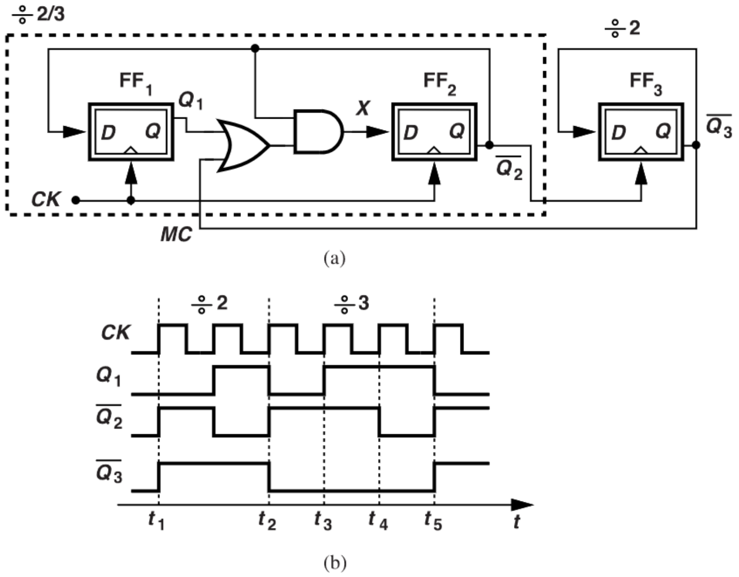


Figure 2.8: **(a)** divide-by-5 circuit and **(b)** its wave-forms [10].

To understand how it works, let's assume that the flip-flops change their outputs on the rising edge of their respective clocks. Note that Q_1 always follows $\overline{Q_2}$ after one clock delay. We start with $Q_1\overline{Q_2Q_3} = 000$ [Figure 2.8(b)] and notice that $MC = \overline{Q_3} = 0$, $X = 0$, and the $\div 2/3$ stage divides by 3. When CK goes high at $t = t_1$, $\overline{Q_2}$ rises, but Q_1 remains low. The second (asynchronous) stage responds to the rising edge on Q_2 by changing Q_3 to 1, causing Div23 to divide by 2 in the next cycle. Hence, $\overline{Q_2}$ and Q_1 simply toggle between 0 and 1 on the subsequent rising edges of CK. This continues until t_2, at which point $\overline{Q_2}$ rises and $\overline{Q_3}$ falls, putting Div23 in $\div 3$ mode. At $t = t_3$, Q_1 goes high, but $\overline{Q_2}$ does not change. The present state of $Q_1\overline{Q_2} = 11$ leads $\overline{Q_2} = 0$ at $t = t_4$. Q_1 falls at t_5, while $\overline{Q_2}$ rises, and so does $\overline{Q_3}$. The circuit is now in the same state as at $t = t + 1$, revealing its periodic behaviour. We notice that (a) Div23 divides by 2 for two input cycles (from t_1 to t_2) and by 3 for three input cycles (from t_2 to t_5), and (b) only $\overline{Q_3}$ generates a periodic output with the proper period and can be detected by the next stage. In analyzing dividers, we can also consider the number of input pulses that result in one output pulse instead of unput edges. In Figure 2.8, for example, Div23 receives two clock pulses between t_1 and t_2 and three between t_2 and t_5 [10].

2.1.3 Drawbacks of Integer-N Dividers

The integer-N frequency divider has been widely used in many electronic systems for frequency synthesis. However, it suffers from several drawbacks that limit its performance. One of the main limitations is the coarse of frequency resolution that can only be achieved in integer multiples of the reference frequency. This means that the output frequency of the PLL can only be tuned to a limited set of values, which may not be sufficient for some applications that require fine frequency resolution. In order to achieve finer frequency tuning, the reference frequency can be reduce. However, this trade-off between frequency resolution and loop bandwidth can have adverse effects on the PLL performance, such as increased settling time, VCO noise, and in-band noise contributed by the reference source, PFD, charge pump, and divider. These limitations can be addressed by using a fractional-N frequency divider, which allows for finer frequency resolution without sacrificing the loop bandwidth. The fractional-N divider uses a combination of integer and fractional division, which enables the PLL to generate frequencies that are not limited to integer multiples of the reference frequency. The most common technique used to implement fractional-N dividers is $\Delta\Sigma$ modulation, which allows for high resolution and low spurious levels. So, the fractional-N frequency divider offers a solution to the fundamental trade-off between frequency resolution and loop bandwidth in integer-N PLLs, making it a valuable tool for frequency synthesis in various applications [3].

2.2 Fractional-N Divider

2.2.1 Basics

Figure 2.9 illustrates a fractional-N PLL, which is a modified version of the well-known integer-N PLL. In an integer-N PLL, the frequency divider produces one rising edge for every N rising edges of $v_{PLL}(t)$ during steady state. In steady state, the rising edge of $v_{REF}(t)$ and $v_{DIV}(t)$ are aligned, with a possible constant offset, leading to $f_{PLL} = N \cdot f_{REF}$.

The selection of N determines the desired channel, but the frequency resolution is restricted to f_{REF}. In contrast, a fractional-N PLL adjusts the frequency division modulus once per reference period, based on a series of small integers $y[n]$.

In the original fractional-N PLL, a fractional value α is digitally accumulated, and 1-bit carry out is used as $y[n]$. Consequently, $v_{REF}(t)$

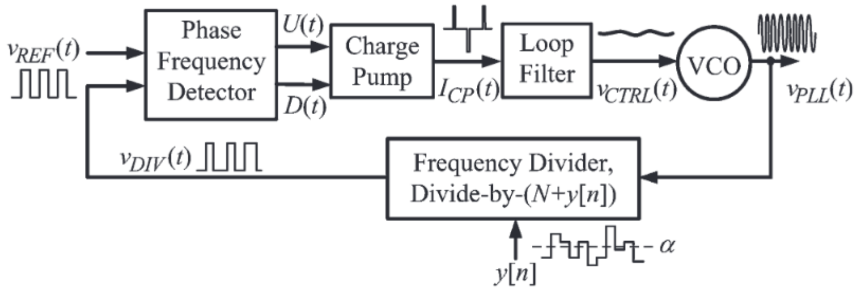


Figure 2.9: Block diagram of fractional-N PLL [12].

and $v_{DIV}(t)$ are in phase synchronization only in an average sense. The PLL still locks, however, now $f_{PLL} = (N + \text{avg}(y[n])) \cdot f_{REF} = (N + \alpha) \cdot f_{REF}$, which means that f_{PLL} is a fractional multiple of f_{REF}, hence the name fractional-N PLL. However, $v_{PLL}(t)$ and $v_{DIV}(t)$ exhibit instantaneous frequency errors of $(y[n] - \alpha) \cdot f_{REF}$ that cause undesirable errors in the phase of the output $v_{PLL}(t)$. Since $y[n]$ is periodic with a period that depends on α, these phase errors manifest as strong undesirable tones called fractional spurs in the power spectral density (PSD) of $v_{PLL}(t)$. The fractional spurs occur at frequency offsets that are integer multiples of $\alpha \cdot f_{REF}$ away from the center frequency. One of the most popular techniques to generate $y[n]$ is using $\Delta\Sigma$ modulator [12].

2.2.2 The Idea Behind $\Delta\Sigma$ Fractional-N PLLs

The example of generating the 2.403GHz second Bluetooth channel frequency with a reference frequency of 19.68 MHz is used to illustrate the idea behind $\Delta\Sigma$ fractional-N PLLs. Initially two "bad" fractional-N PLLs that obtain the desired frequency but perform poorly against phase noise are described. Then the $\Delta\Sigma$ fractional-N PLL technique is presented as a method for enhancing the phase noise performance.

The output frequency of an integer-N PLL with a reference frequency of 19.68 MHz is 2.40096 GHz when the divider modulus, N, is set to 122 and 2.42064 GHz when N is set to 123. The issue is that N would to be set to the non-integer number of 122+51/492 in order to obtain the target frequency of 2.403 GHz. This cannot be implemented directly because the divider modulus must be an integer value. Nevertheless, since the divider modulus can be changed for each reference period, one choice is to alternate between N=122 and N=123 so that the average modulus over numerous reference periods converges to 122+51/492. In this instance, the requisite 2.403 GHz average PLL output frequency is obtained. The majority of fractional-N PLLs are based on this core concept [3].

While producing non-integer multiples of the references frequency can be accomplished by dynamically adjusting the divider modulus, there is a cost in the form of increased phase noise. During each reference period the difference between the actual divider modulus and the average, i.e., ideal, divider modulus represents error that gets injected into the PLL and results in increased phase noise. The properties of the series of divider moduli determine how much the phase noise is enhanced, as will be discussed below [3].

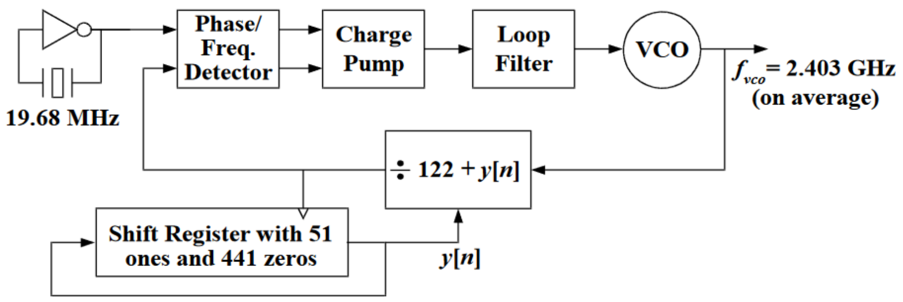


Figure 2.10: A fractional-N PLL that generates non-integer multiples of the reference frequency, but has phase noise consisting of large spurious tones [3].

Accordingly, over each set of 492 consecutive references periods, the divider modulus is set to 122 or 123 a total of 441 times and 51 times, respectively, in the fractional-N PLL seen in Figure 3. The average modulus is therefore 122+51/492, as needed. The moduli sequence is periodic with a period of 492, so it repeats at a rate of 40 kHz. Consequently, the difference between the actual divider moduli and their average is a periodic sequence with a repeat rate of 40 kHz, so the resulting phase noise is periodic and is comprised of spurious tones at integer multiples of 40 kHz. Many of the spurious tones occur at low frequencies, and they can be very large. Unfortunately, having a narrow PLL bandwidth is the only method to suppress the tones, which renders the potential benefits of the fractional PLL useless [3].

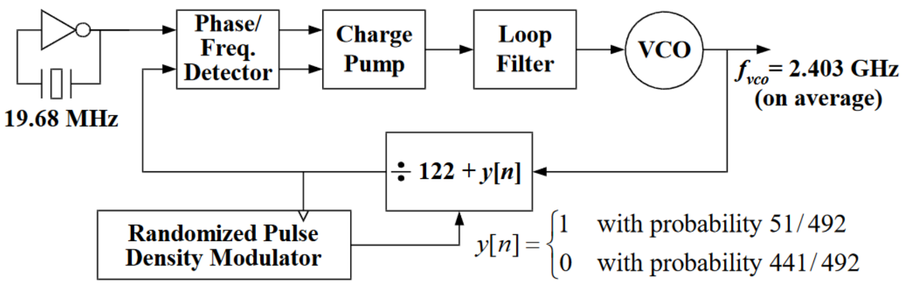

Phase/ Freq. Detector
Charge Pump
Loop Filter
VCO
f_{vco} = 2.403 GHz (on average)
19.68 MHz
÷ 122 + $y[n]$
Randomized Pulse Density Modulator
$y[n] = \begin{cases} 1 & \text{with probability } 51/492 \\ 0 & \text{with probability } 441/492 \end{cases}$

Figure 2.11: A fractional-N PLL that generates non-integer multiples of the reference frequency, but has large amount of in-band phase noise [3].

To eliminate spurious tones, introducing randomness can be useful to break up the periodicity in the moduli sequence while still achieving the desired average modulus. Taking Figure 2.11 as example, a digital block is used to generate a sequence $y[n]$ that approximates a sampled sequence of independent random variables that take either 0 or 1 with probabilities 441/492 and 51/492, respectively. During the n^{th} reference period the divider modulus is set to 122+$y[n]$, so the sequence of moduli has the desired average, but also to introduce white noise instead of spurious tones. However, if the PLL bandwidth is not small, the white noise contribution may still be significant due to integration by the PLL transfer function [3].

The sequence y[n] in fractional-N PLL can be expressed as $y[n] = x + e_m[n]$, where x is the desired fractional part of the modulus and $e_m[n]$ is undesired quantization noise resulting from using integer moduli instead of ideal fractional values. In the first example, $e_m[n]$ is periodic and causes spurious tones at multiples of 40 kHz, while in the second example, $e_m[n]$, is white noise. The PLL attenuates the portion of $e_m[n]$ outside its bandwidth, but the portion within its bandwidth contributes significant phase noise unless the PLL bandwidth is kept low [3].

The Figure 2.12 presents PSD plots of the output phase noise due to $\Delta\Sigma$ modulator quantization noise, $e_m[n]$, in two computer simulated versions of the example $\Delta\Sigma$ fractional-N PLL with different loop bandwidth (50 kHz and 500 kHz). The PSD of $e_m[n]$ increases with frequency [3], resulting in significantly smaller phase noise of PSD for the

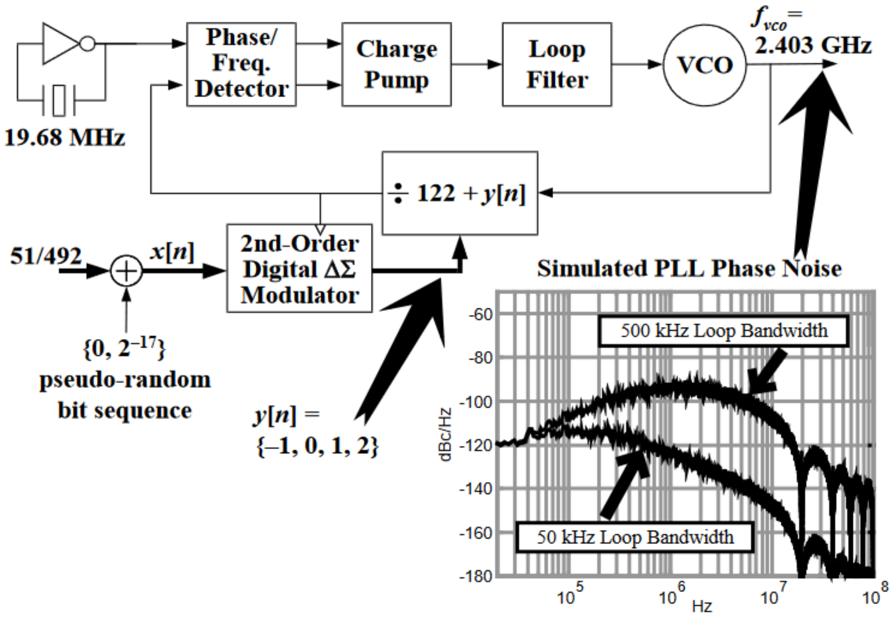


Figure 2.12: A $\Delta\Sigma$ fractional-N PLL example [3].

50 kHz bandwidth PLL compared to the 500 kHz bandwidth PLL. The

former meets requirements for a local oscillator in a direct conversion Bluetooth transceiver, while the latter falls short of the requirements by at least 23dB [3].

2.2.3 Delta-Sigma Modulation Overview

Basic Operation

An illustration of a first-order $\Delta\Sigma$ modulator can be seen in Figure 2.13(a). It employs a 1-bit quantizer, which is essentially a signal comparator acting as an analog-to-digital converter (ADC), along with a digital-to-analog converter (DAC) equipped with two switches that generate $\pm V_{REF}$. The high gain of the comparator enforces a virtual ground at a specific node denoted as X. Due to the discrete-time nature of the loop, only the average value of V_X remains close to zero. This property allows for the cancellation of the average difference be-

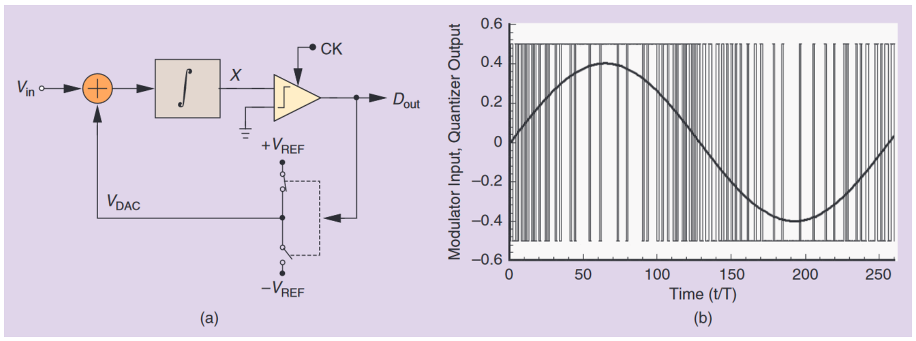


Figure 2.13: (a) A simple first order $\Delta\Sigma$ modulator with a 1-bit quantizer, (b) input and output waveforms [7].

tween the analog input signal, V_{in}, and the DAC output voltage, represented by the variable t, D_{out}. Consequently, the modulator achieves a running average of the digital output that closely tracks the input analog signal. It's important to note that the 1-bit quantizer utilized in this modulator does introduce a significant amount of quantization noise [7].

An Example Uniform Quantizer

Figure 2.14 shows the input-output characteristics of a 9-level uniform quantizer with interger output levels. Inputs with magnitude less than 4.5 are rounded to the nearest integer output, while inputs greater than 4.5 or less than -4.5 result in output of 4 or -4, respectively; such values are said to overload the quantizer. The quantization noise can be represented as an additive noise source using the equation $e_q = y[n] - r[n]$ [3].

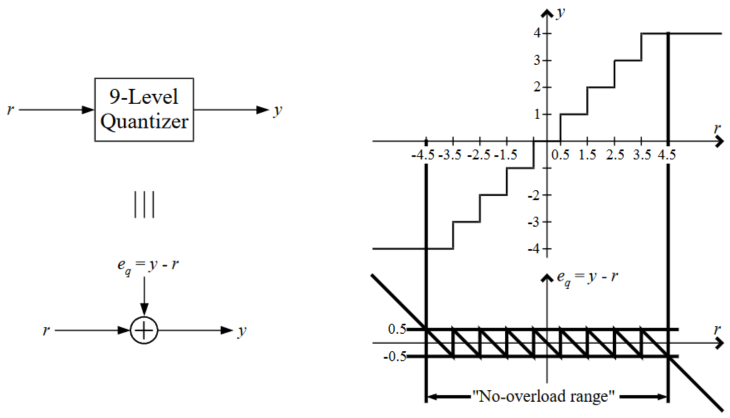


Figure 2.14: A 9-level quantizer example [3].

Second-order ΔΣ Modulator

An example of second-order ΔΣ modulator architecture is shown in Figure 2.15.

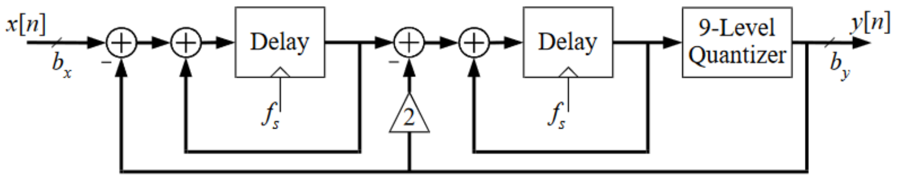


Figure 2.15: A ΔΣ modulator example [3].

The structure employs the same 9-level quantizer presented above, in this scenario there are two feedback loops surrounding the quantizer that is preceded by two discrete-time integrators (also called accumulators) that introduce delay. Each integrator has a transfer function of $z^-1/(1 - z^{-1})$ which means its n^{th} output sample is the sum of all input samples for times k¡n. The modulator can be viewed as a two-input, single-output, linear-time-invariant, discrete-time system, with the quantizer represented as an additive noise source. The output $y[n]$ is the sum of input $x[n-2]$ and overall quantization noise $e_m[n]$, where $e_m[n]$ is given by [3]:

$$e_m[n] = e_q[n] - 2e_q[n-1] + e_q[n-2] \tag{2.1}$$

2.2.4 Conclusion

In conclusion, this chapter provided an overview of frequency dividers, focusing on both integer-N and fractional-N divider architectures. The integer-N divider section explored different designs, starting with latch-based implementations. These designs utilize latches to divide the input frequency by an integer number. Additionally, the chapter discussed dual-modulus prescalers, which allow for frequency division by two different values, enabling higher overall division ratios. However, the integer-N dividers were shown have certain drawbacks, including limited resolution and spurious output frequencies.

The fractional-N divider section introduced the concept of fractional frequency division. It outlined the basics of fractional-N dividers and delved into the idea behind $\Delta\Sigma$ fractional-N phase-locked loops (PLLs). These PLLs employ $\Delta\Sigma$ modulation techniques to achieve fractional frequency to achieve fractional frequency division by modulating the division ratio. The chapter provided an overview of $\Delta\Sigma$ modulation and its role in fractional-N dividers.

Chapter 3

Results and Discussion

This chapter presents the findings and analysis of simulations conducted on various blocks of a Phase-Locked Loop (PLL) system. The chapter begins with individual simulations of the main blocks, such as the Phase Frequency Detector(PFD), Charge Pump (CP), Voltage-Controlled Oscillator (VCO), and dividers in its two types integer and fractional. Using two types of modeling: Behavioral using Verilog-A and Transistor-level using Cadence Virtuoso with the GPDK 90nm process. These simulations lay the foundations for the understanding the behavior and characteristics of each block in isolation. All Verilog-A codes used in the behavioral modeling can be found in Appendix A.

Subsequently, the individual blocks are integrated into a complete system, starting with the simulation and analysis of the Integer-N PLL. The results and observations from this simulation are discussed in detail. Following that, the same process is repeated for the Fractional-N PLL, allowing for a comprehensive comparison between the two types of dividers.

3.1 Simulation of Main Blocks

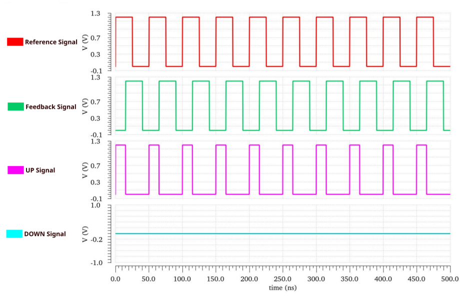
3.1.1 Phase-Frequency Detector

In this section, we present the results obtained from the behavioral simulation of the Phase-Frequency Detector (PFD) block using Verilog-A, followed by transistor-level simulation using an extended True Single-Phase Clock (E-TSPC) D Flip-Flop toplogy. The objective of these simulation was to evaluate the performance and functionality of the PFD block in a practical circuit implementation.

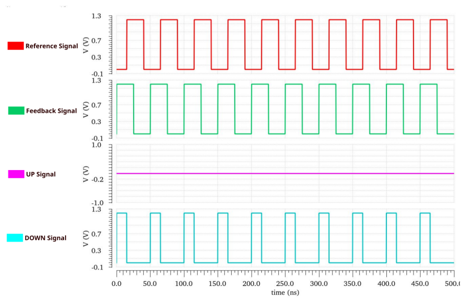
Behavioral Modeling

By modeling the PFD block at a higher level of abstraction, we aimed to verify its functionality and analyze its response under various input conditions. Therefore we conducted simulations under two specific conditions. Firstly we examined the PFD block's response when the reference signal was leading delaying the feedback signal with 15ns, triggering the UP signal as it is shown in Figure 3.1(a). This scenario represents a common operating condition where the feedback signal is delayed than the reference signal. Secondly, we investigated the PFD block behaviour when the feedback signal was leading, resulting in the down signal being triggered as it is depicted in Figure 3.1(b).

Figure 3.2 shows the characteristics of the output voltage versus phase variation, provides valuable insights into the behaviour of the PFD in relation to phase difference between the inputs signals. This particular plot demonstrate the PFD's ability to accurately respond to small phase variations.



(a)



(b)

Figure 3.1: Waveforms of the behavioral simulation of the PFD in two input conditions **(a)** when the reference signal is leading **(b)** when the feedback signal is leading.

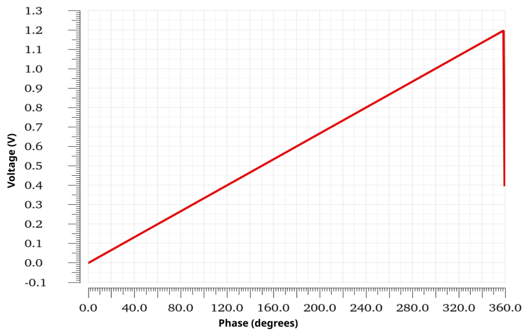


Figure 3.2: The output voltage versus the phase variation.

Transistor-Level Modeling

When designing a PFD, several factors should be taken into account. Firstly, since the PFD operates at low frequencies compared to the other blocks in the system, its design should be adequate to this purpose. Additionally, it is crucial to ensure that the design exhibits high sensitivity in detecting signals at high speeds to avoid the dead zone.

The conventional design of the PFD, as illustrated in Figure 1.5, utilizes an AND gate to trigger the reset signal. However, this design exhibits a delay in the reset signal due to the inherent delay introduced by the AND gate. To address this issue, we have introduced modifications to the design by directly applying the reset signal to the clock

(CLK) signal. This modification ensures that the reset signal is fast and operated without any delay, triggering as soon as the CLK signal goes high, as shown in Figure 3.3.

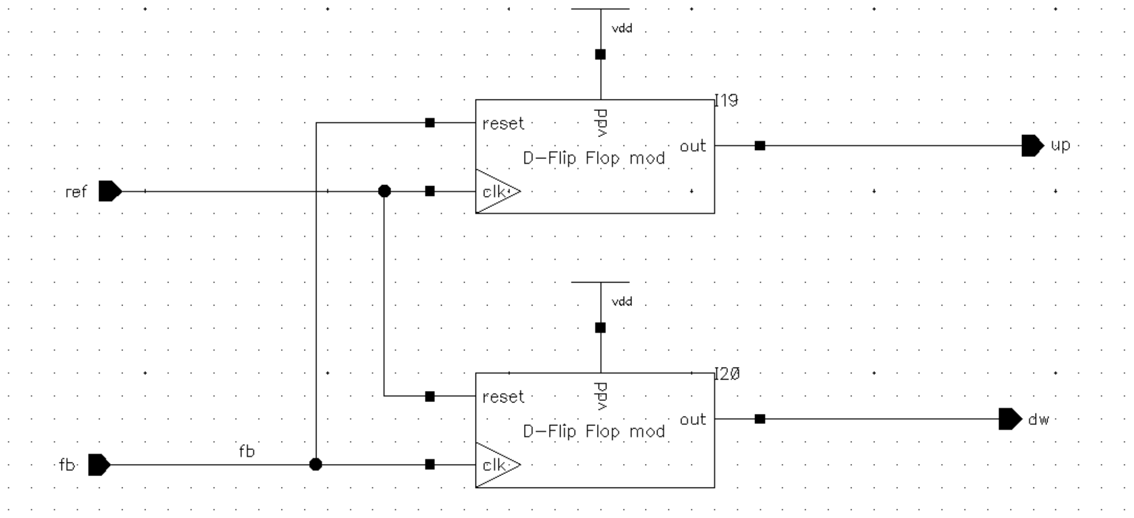


Figure 3.3: Modified version of the conventional PFD.

In our design, we opted for the topology of E-TSPC over the TSPC, due to its advantages such as a reduced number of transistors, which leads to a decreased propagation delay and lower power consumption, its schematics is shown in Figure 3.4.

The implemented PFD design demonstrates a power consumption of only $27.09\mu W$ at a frequency of 20MHz, which is notably low. Figure 3.5 illustrates the triggering of the UP and DOWN signals in two scenarios: (a) when the feedback signal is delayed relative to the reference signal, and (b) when the reference signal is delayed relative to the feedback signal

These results showcase the effectiveness of our modified PFD design in detecting the difference of phase between the reference signal and the feedback signal.

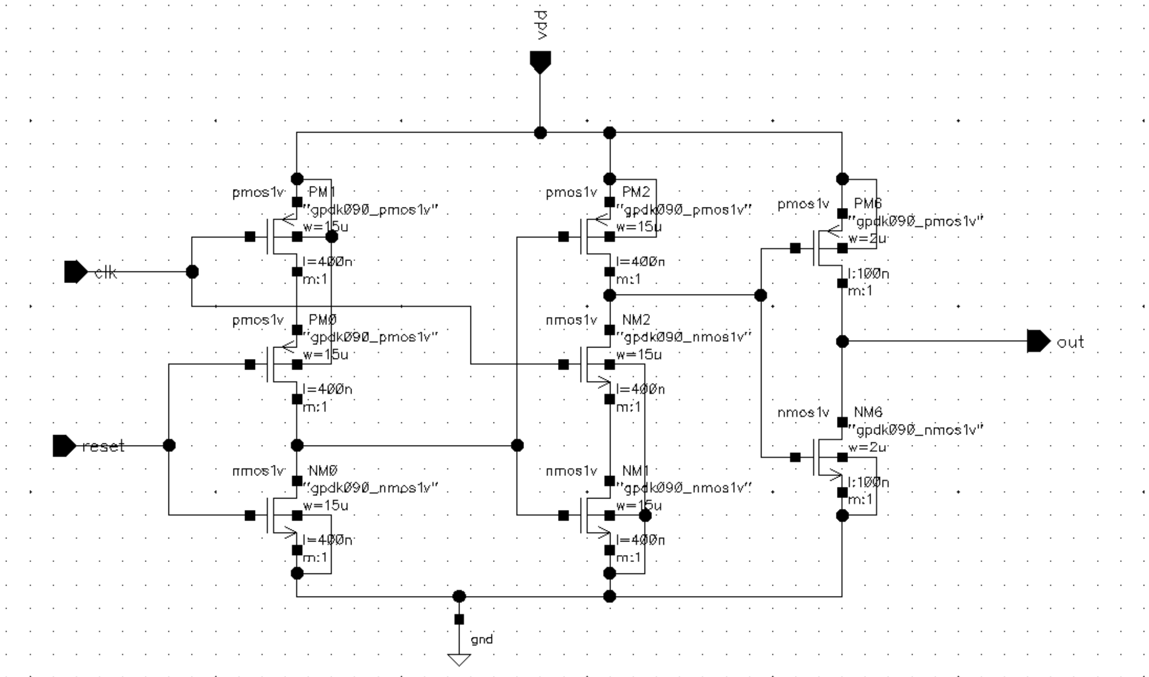
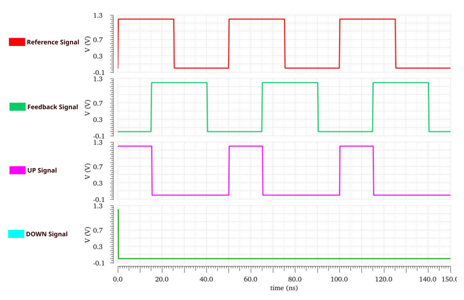
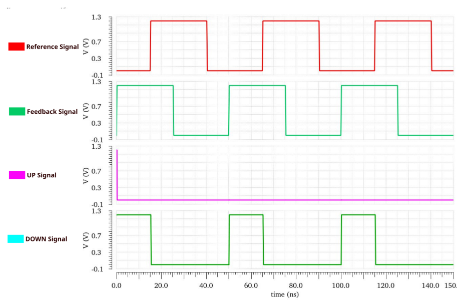

vdd
pmos1v PM1 "gpdk090_pmos1v" w=15u l=400n m:1
pmos1v PM2 "gpdk090_pmos1v" w=15u l=400n m:1
pmos1v PM6 "gpdk090_pmos1v" w=2u l:100n m:1
clk
pmos1v PM0 "gpdk090_pmos1v" w=15u l=400n m:1
nmos1v NM2 "gpdk090_nmos1v" w=15u l=400n m:1
out
reset
nmos1v NM6 "gpdk090_nmos1v" w=2u l:100n m:1
nmos1v NM0 "gpdk090_nmos1v" w=15u l=400n m:1
nmos1v NM1 "gpdk090_nmos1v" w=15u l=400n m:1
gnd

Figure 3.4: The E-TSPC topology.

Our design despite its merits, does have a few drawbacks that need to be addressed. One notable disadvantage is that when either of the signals experience a delay exceeding π radians, the leading signal will be at low state in this case and fails to trigger the reset signal. Consequently, this creates a very short dead zone where the PFD fails to detect the phase difference. Figure 3.6 illustrates this scenario by depicting the output voltage versus the phase variation. It is evident from the graph that the PFD's response is limited in situations where the phase delay exceeds π radians.



(a)



(b)

Figure 3.5: Simulation results of PFD using E-TSPC topology **(a)** triggering the UP signal **(b)**triggering the DOWN signal.

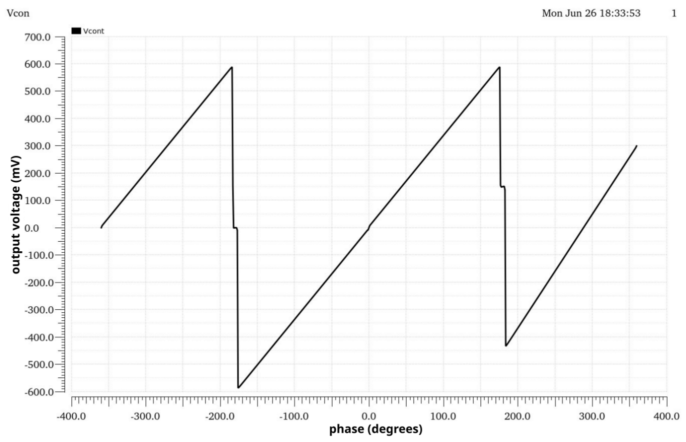


Figure 3.6: Output voltage versus the phase variation illustrating a dead zone at π.

3.1.2 Charge Pump

In this section, we have implemented a behavioral model of a charge pump. A test bench, depicted in Figure 3.7, was constructed to include both the PFD and the charge pump blocks, followed by the loop low-pass filter. The simulation results are presented in Figure 3.8.
Figure 3.8(a) illustrates the behavior of the charge pump when there is a rising edge of the UP signal. In this scenario, the charge pump generates a positive current. Consequently, the loop pass filter charges as demonstrated in the plot.

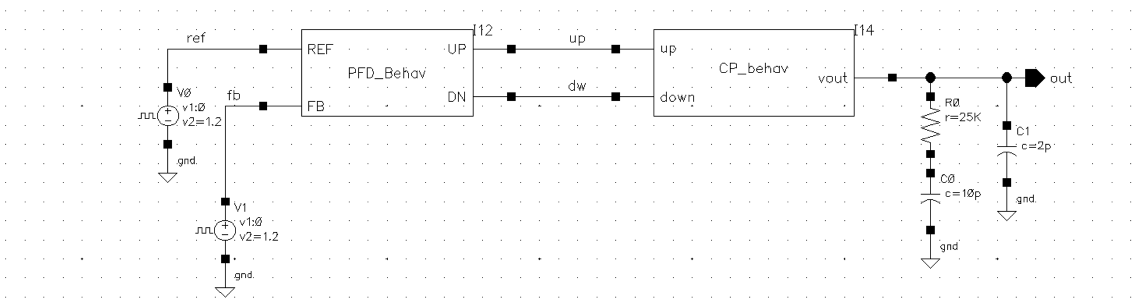
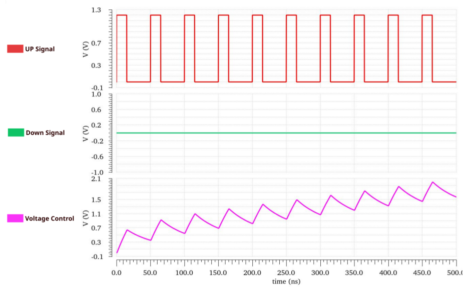
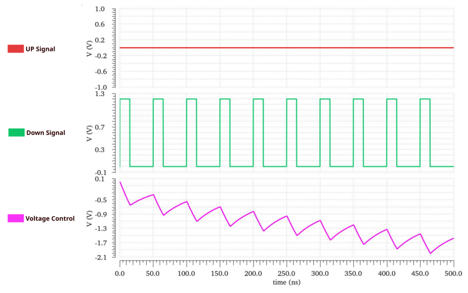


Figure 3.7: Test bench of the charge pump.

On the other hand, Figure 3.8(b) shows the response of the charge pump when there is a rising edge of the DOWN signal. In this case, the charge pump produces a negative current, leading the loop pass filter to discharge.



(a)



(b)

Figure 3.8: Waveforms of the charge pump output when **(a)** there is an UP signal resulting in charging the LPF **(b)** there is a DOWN signal resulting in discharging the LPF.

3.1.3 Voltage-Controlled Oscillator

Behavioral Modeling

The behavioral model of the VCO was designed to operate within a frequency range of 2.36 GHz to 2.44GHz, providing a bandwidth of

80MHz. The output is in the form of a square wave, which facilitates its connection to the divider block. The expected voltage range of the output signal is from 0V to VDD, where VDD is 1.2V in this particular case.

Figure 3.9 illustrates the oscillation results obtained from the VCO block, showcasing the square wave output waveform. It demonstrates the successful generation of the desired oscillation.

Furthermore, in Figure 3.10, the characteristics of the VCO are presented, specifically the frequency-versus-voltage relationship. Ideally, in an ideal VCO, this relationship is linear, indicating that K_{VCO} remains constant across the voltage range. This linear characteristic is desirable for stable and predictable frequency generation.

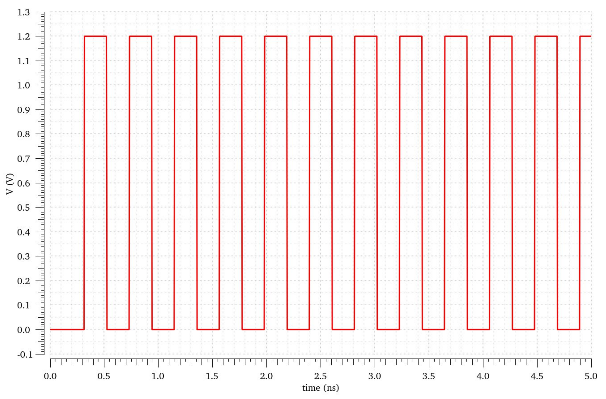


Figure 3.9: Oscillation results of the behavioral model of the VCO.

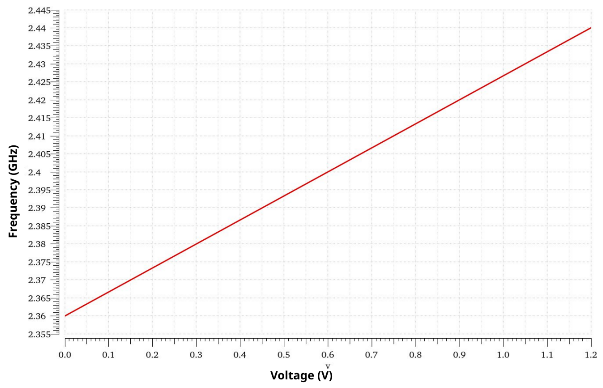


Figure 3.10: Frequency-Versus-Voltage relationship of the behavioral model of the VCO.

Transistor-Level Modeling

Our choice for the ring oscillator architecture as VCO, is because the ring oscillator is the most competitive and promising circuit for high frequency and wide bandwidth, It has many advantages such as easily implemented using the CMOS technology, require low input voltage, exhibit low power dissipation, and has large tuning range [5].

The ring oscillator should contain odd number of inverters with the output of the last inverter should be tied to the input of the first inverter. To achieve a sustained oscillation, a phase shift of 2π and unity voltage gain are required. The phase shift equally divided by the each inverter. The oscillation frequency of the ring oscillator is determined by the propagation delay of the stage and the number of inverter.

Figure 3.11 illustrates a ring oscillator comprising five stages of ring inverters. Each inverter exhibit a propagation delay τ_{pd} characterized by the average delay between low-to-high and high-to-low transitions of its individual stages as depicted in the following equation [5]:

$$\tau_{pd} = (\tau_{phl} + \tau_{plh})/2 \tag{3.1}$$

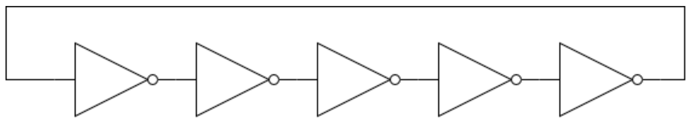


Figure 3.11: Diagram of 5 stages ring oscillator.

For N stages of ring oscillator the oscilaltion frequency is given as:

$$f_{osc} = 1/2N\tau_p \tag{3.2}$$

where $\tau_p = \tau_{phl} = \tau_{plh}$, the ring oscillator that we have implemented has 5 stages, so the frequency of oscillation in our case is:

$$f_{osc} = 1/10\tau_p \tag{3.3}$$

In our design, we opted for a 5-stage as number of stages to obtain a greater number of available multi-phase outputs. However, this decision posed a challenge in terms of reducing the oscillation frequency due to the increased propagation time delay caused by each cell. To address this issue, we employed a strategy of adjusting the capacitance load of each stage by manipulating the W/L ratio of the NMOS and PMOS transistors. This approach allowed us to strike a balance between the desired multi-phase outputs and the desired oscillation frequency.

The oscillation frequency of a ring oscillator is influenced by the delay of its charging pull-up network (PMOS) and discharging pull-down network (NMOS). The calculation of the charging time, is determined by the RC delay and can be expressed as $t_{plh} = 0.7R_pC_{out}$ where R_p is the PMOS resistance. Similarly, the discharging time, t_{phl} is calculated as $t_{phl} = 0.7R_nC_{out}$ where R_n is the NMOS resistance. The delay of each cell in the oscillator can be controlled by adjusting the resistance and load capacitance.

To achieve a high oscillation frequency it is crucial to minimize the delay. This can be accomplished by ensuring that each stage has a low resistive path for charging and discharging, as well as smaller load capacitance. One approach to vary the resistance is by employing multiple PMOS transistors in parallel during the charging phase and multiple NMOS transistors during the discharging phase. Parallel connection of PMOS reduces the equivalent resistance R_{eqp} during the charging, while parallel NMOS reduce the equivalent resistance R_{eqn} during the discharging [6].

The overall propagation delay of the inverter is defined as follows:

$$t_p = 0.69 C_L (\frac{R_{eqn} + R_{eqp}}{2}) \tag{3.4}$$

A MOS is operating as voltage controlled resistor, its equivalent resistance is given as:

$$R_{on} = \frac{1}{\mu_n C_{ox} \frac{W}{L} (V_{GS} - V_{th})} \tag{3.5}$$

Where V_{GS}: The gate-source voltage, V_{th}: The threshold voltage, μ_n: the mobility of electrons, C_{ox}: the oxide capacitance.

The propagation delay of inverter is directly proportional to the W/L ratio of the transistor. By adjusting the W/L ratio, it is possible to control and modify the delay of the inverter.

An alternative method widely used to regulate the oscillation frequency is through utilization of current-starved inverters, as illustrated in Figure 3.12. This approach allows for precise frequency control. The basic concept involves a PMOS transistor (PM8) controlled by the voltage control and it mirrors its current to the PM4, so PM4 replace the VDD and will act as current source to supply a limited current to the inverters. By employing this technique, the input-to-output delay of each inverter can be modified. Consequently, the maximum current drawn by each inverter is fixed for a given voltage control setting. This limitation affects the charging and discharging rate of the capacitors, thereby altering the delay of each inverter and ultimately restricting the output frequency. As a result, the maximum frequency does not reach the VDD level due to the imposed current limitation.

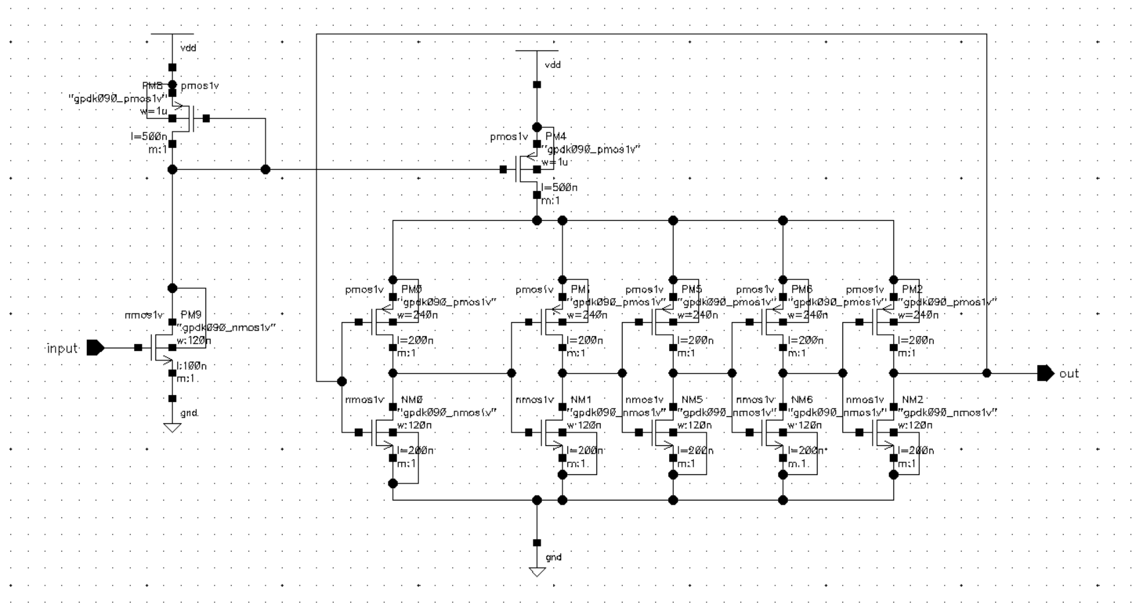


Figure 3.12: The implemented 5 stages ring oscillator using current-starved technique to control the frequency.

Figure 3.13 illustrates the oscillation results obtained from the implemented ring oscillator. It can be observed that the peak amplitude of the oscillation does not reach the VDD level of 1.2, indicating a limited voltage swing. Additionally, the power consumption of this architecture is measured to be $101.44\mu W$, which can be attributed to the increased channel length employed in the design.

In Figure 3.14, the frequency-versus-voltage characteristic of our oscillator is depicted, revealing a non-linear relationship ranging. As shown in the graph, the K_{VCO} remains zero from 0V to 0.3V due to the transistor PM9 not reaching its threshold voltage. To achieve a more

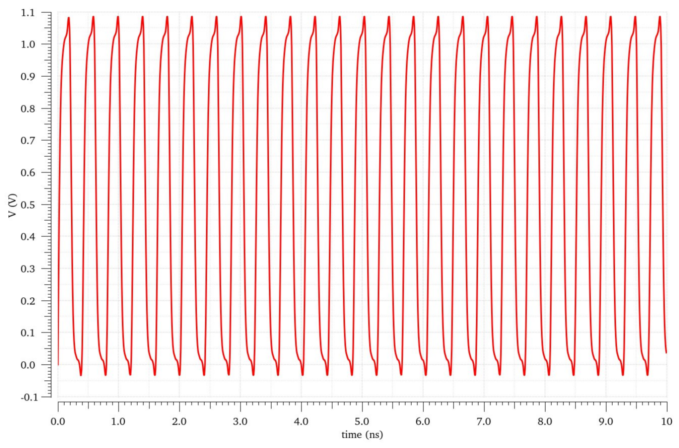


Figure 3.13: Oscillation results of our ring oscillator.

extended linear region, precise adjustments can be made to the propagation delays by carefully tuning the W/L ratios of the components.

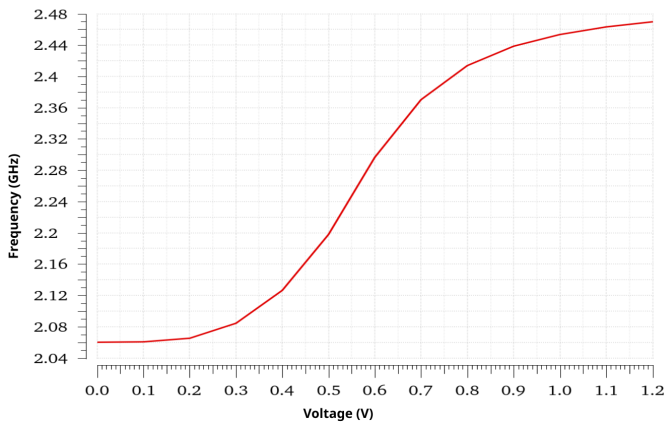
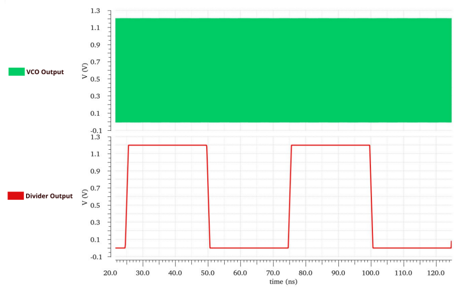


Figure 3.14: Frequency-Versus-Voltage relationship of the ring oscillator.

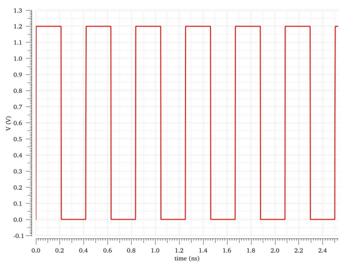
3.1.4 Integer-N Divider

Behavioral Modeling

The purpose of behavioral modeling of the integer divider is to achieve division by integer ratio. The results of our behavioral model are presented in Figure 3.15, displaying the waveforms of the output and input signals. In our specific case, the divider successfully divides a 2.4GHz input signal by 120 resulting in an output frequency of 20MHz.



(a)



(b)

Figure 3.15: Simulation results of the behavioral model of the integer divider by 120 **(a)** Input-Output waveforms **(b)** zoomed view of the input signal.

Transistor-Level Modeling

Our design of the integer-N divider with a ratio of 120 is based on the d-flip flop TSPC topology. We choose this topology due to its remarkable features, including the low power consumption and high speed at high frequencies. These advantages make it an ideal choice for our intended operation at 2.4GHz. It is worth nothing that at lower frequencies, critical nodes have less ability to store charges, increasing the risk of glitches and incorrect results.

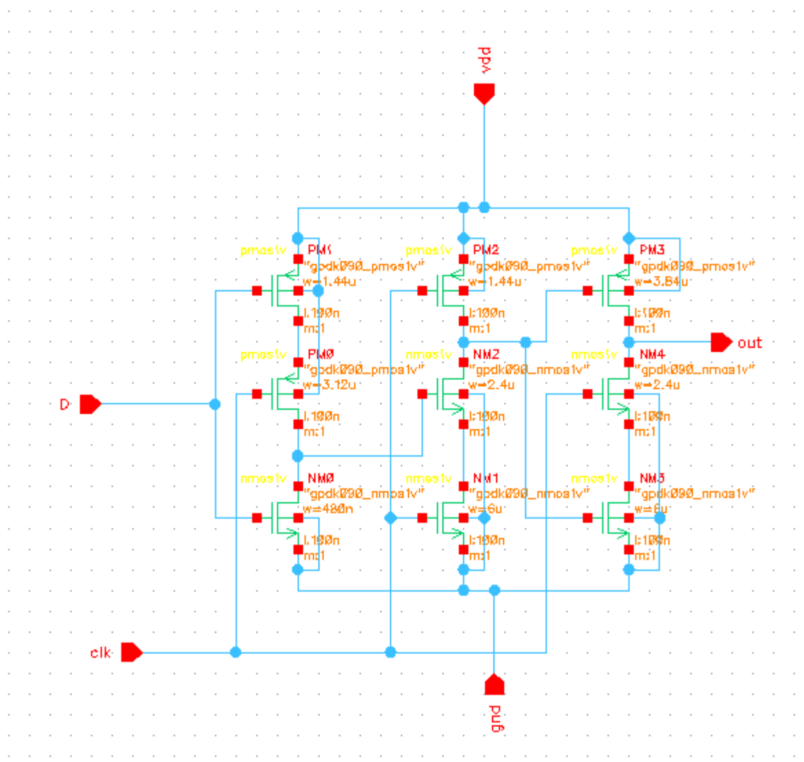
In our case, our objective is to design a divide-by-120 circuit. To accomplish this, we utilized three main blocks: a divide-by-5 circuit (div5), a divide-by-2 circuit (div2), and a divide-by-3 circuit (div3). These specific blocks were chosen because their output frequencies are multiples of 120. It should be notes that 120 can be expressed as the product of 5,3 and multiple factors of 2 ($120 = 5 * 3 * 2 * 2 * 2$).

Regarding the sizing methodology, we selected ratios to ensure sufficient current-driving capability. We defined two ratio, namely n and p, where $n = W_n/L$ and $p = 2n$, we set n to 1.2 and p will be 2.4. We carefully selected individual W/L ratios to guarantee that the PDN (Pull-Down Network) can provide a discharge current at least equal to that of a NMOS transistor with W/L=n, while the PUN (Pull-Up Network) can provide a charging current at least equal to that of a PMOS transistor with W/L=p. This approach ensures a worst-case gate delay equal to a basic inverter [11].
Figure 3.16(a) illustrates the D-Flip Flop topology with the selected sizes, as detailed in Table 3.1. It can observed that in the hold stage (second stage), we chose higher NMOS ratios compared to the pre-charge stage (first stage) NMOS transistors. This decision was made to ensure proper discharging as the hold stage plays a very critical role in providing a reliable signal to the evaluation stage (third stage). Figure 3.16(b) illustrates the input and output waveforms, showing the proper functionality of the D-flip flop.

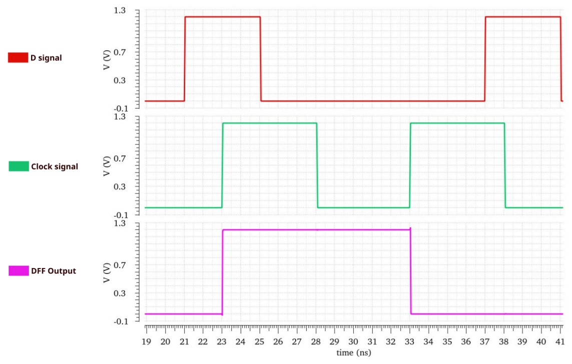
Pre-charge stage	Hold stage	Evaluation stage
6p	6p	16p
13p	20n	20n
4n	50n	50n

Table 3.1: Selected ratios for the TSPC D-Flip Flop circuit.

The divide-by-2 topology mentioned in Figure 2.2(a), is implemented and depicted in Figure 3.17(a) where the D-flip flop with its output connected to its input. This configuration causes the D-flip flop to alternate between two states during each clock cycle, resulting in halving



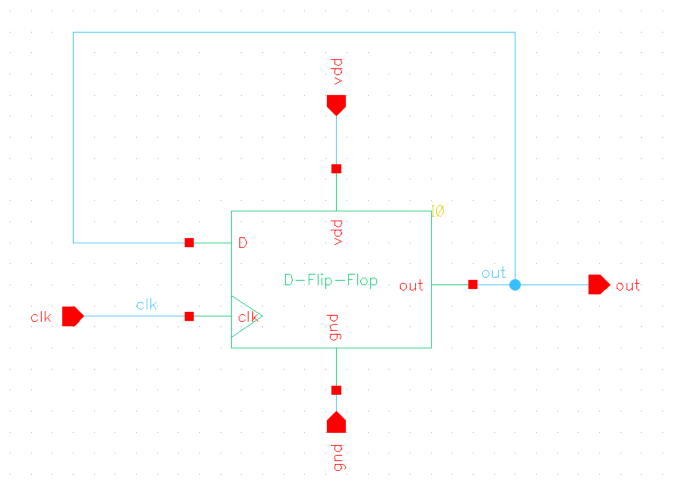
(a)



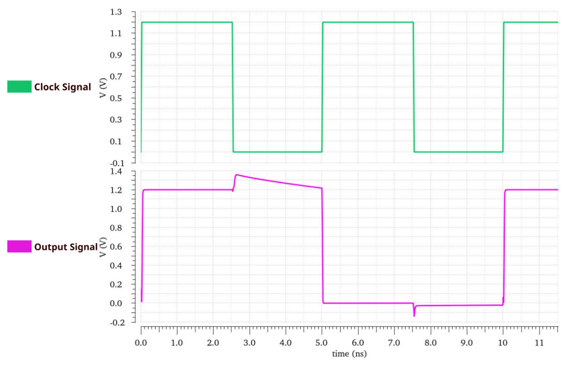
(b)

Figure 3.16: Simulation results of the TSPC D-Flip Flop**(a)** Its schematics **(b)** Its Input-output waveforms.

the output frequency. The corresponding waveforms are illustrated in Figure 3.17(b), clearly showing that hte output signal is exactly half of the input signal. Furthermore, The power consumption of this topology is $10.28\mu W$ at 200MHz.



(a)



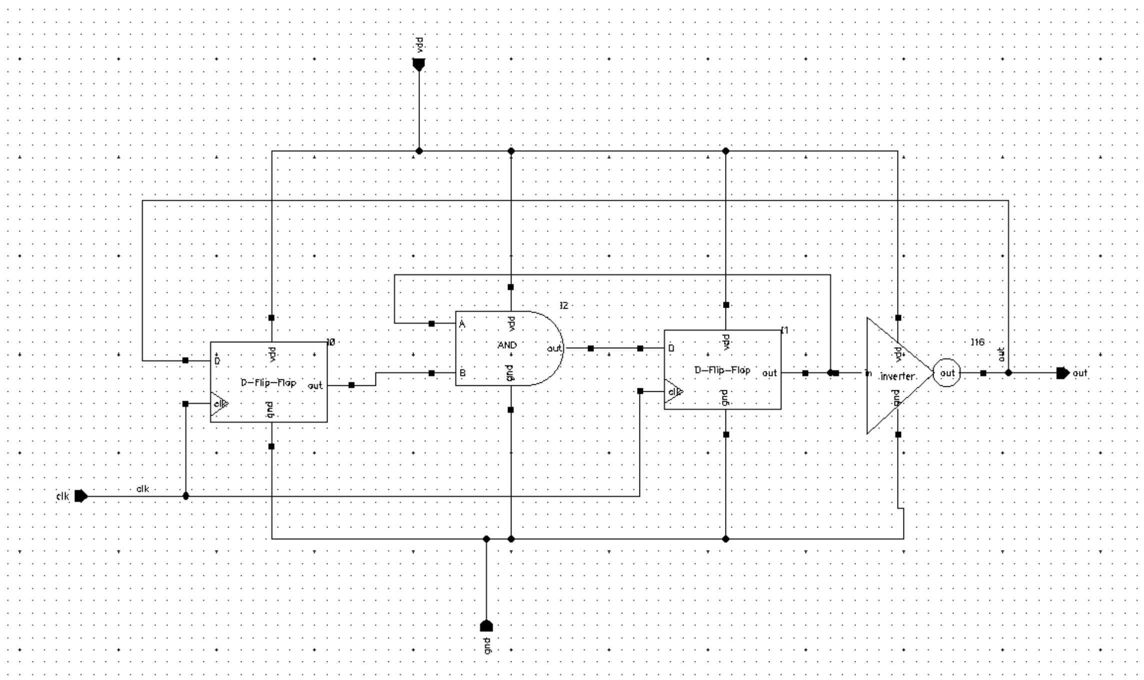
(b)

Figure 3.17: Simulation results of the divide-by-2 topology **(a)** Its schematics **(b)** Its Input-output waveforms.

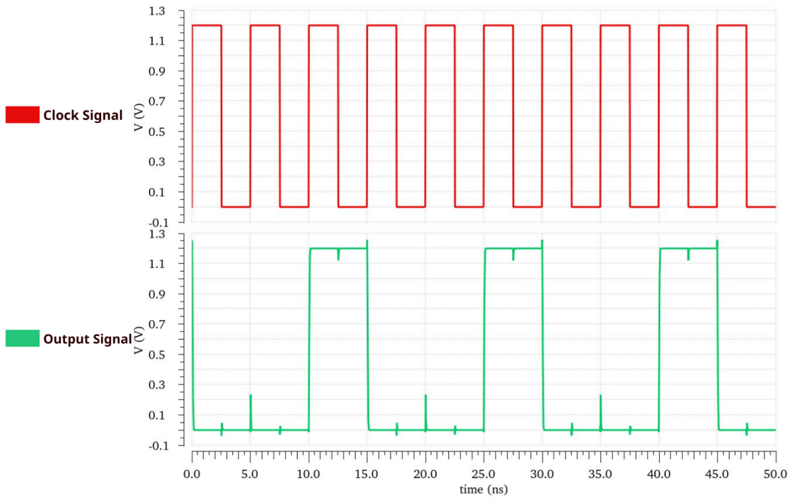
The topology of our divide-by-3 circuit is presented in Figure 3.18(a). As explained in section 2.1.2, in order to achieve division by 3, we need to generate a pattern of 110 or 011. To accomplish this, we have utilized the same topology as topology of the Figure 2.5 but, with an additional feature: sensing the Q output of the second flip-flop and propagating the required inversion of the inputs to the AND gate. This modification ensures the accurate generation of the divide-by-3 output, as depicted in Figure 3.18(b). The power consumption of this topology at a frequency of 200MHz amounts to $23.89\mu W$.

Figure 3.19(a) displays the implementation of the divide-by-5 topology, which its theory was discussed in section 2.1.2. The design consists of a divide-by-23 circuit connected to an asynchronous divide-by-2 circuit. The outpu of the divide-by-2 circuit is then connected to the MC or B input of the OR gate. This input effectively alternated between the values 2 and 3, resulting in the creation of a divide-by-5 counter. Furthemore at frequency of 1GHz, the power consumption of this topology is measured to be $97.66\mu W$. The input-output waveforms illustration the behavior of this circuit can be observed in Figure 3.19(b).

By combining the 3 blocks presented above in a suitable configuration, we can achieve the desired divide-by-120 functionality. The topology of our implementation is depicted in Figure 3.20(a), and the corresponding input-output waveforms are illustrated in Figure 3.20(b). This topology demonstrates a power consumption of $458.09\mu W$ when functioning at 2.4GHz.

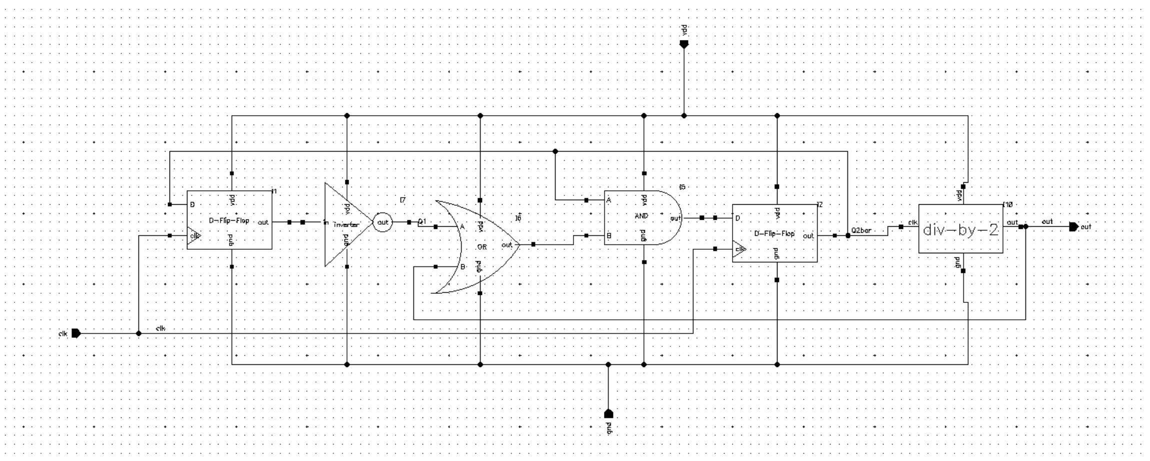


(a)

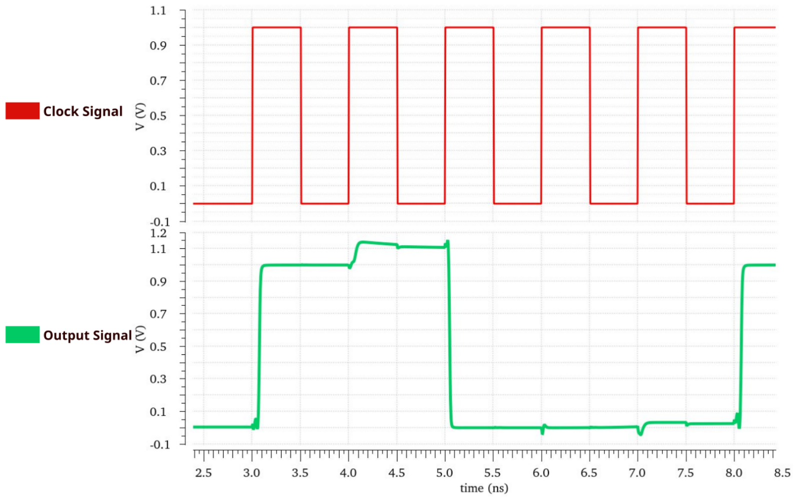


(b)

Figure 3.18: Simulation results of the divide-by-3 topology **(a)** Its schematics **(b)** Its Input-output waveforms.



(a)

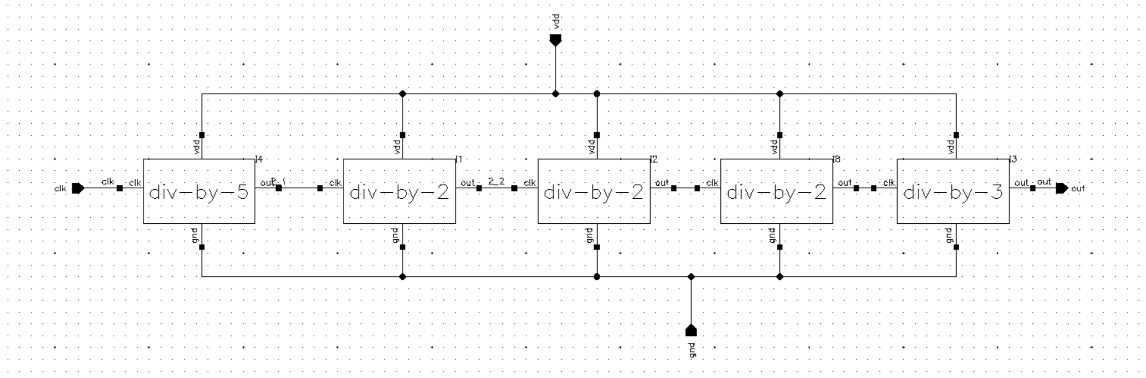


(b)

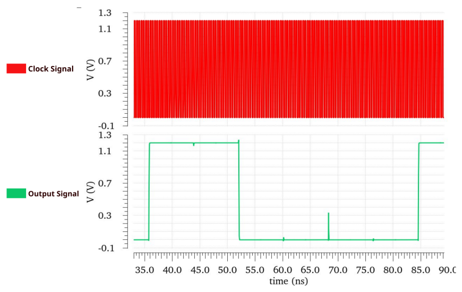
Figure 3.19: Simulation results of the divide-by-5 topology **(a)** Its schematics **(b)** Its Input-output waveforms.

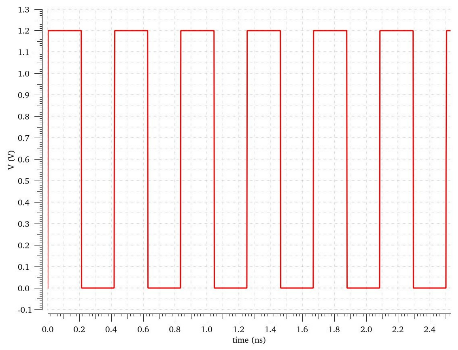
3.1.5 Fractional-N Divider

In order to simplify the design of our fractional divider using $\Delta\Sigma$ technique, we opted for a 1-bit $\Delta\Sigma$ modulator. The schematics of the 1-bit modulator is depicted in Figure 3.21(a), and its input-output wavefroms are illustrated in Figure 3.21(b). The average output of



(a)





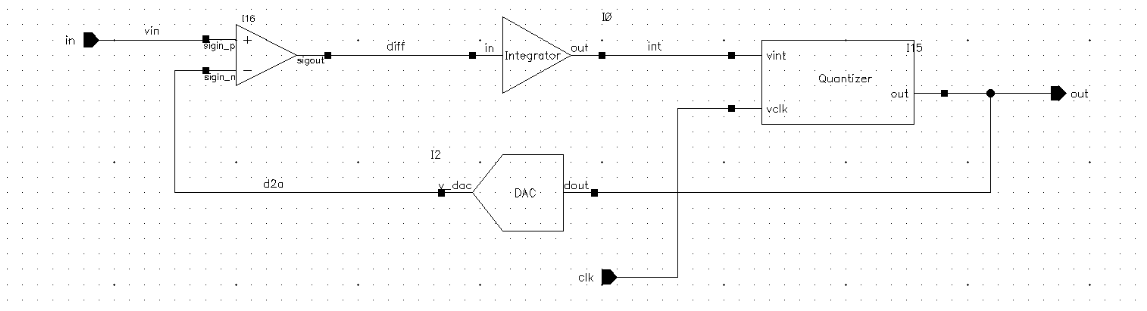
(b) (c)

Figure 3.20: Simulation results of the divide-by-120 topology **(a)** Its schematics **(b)** Its Input-output waveforms **(c)** zoomed view of the input signal.

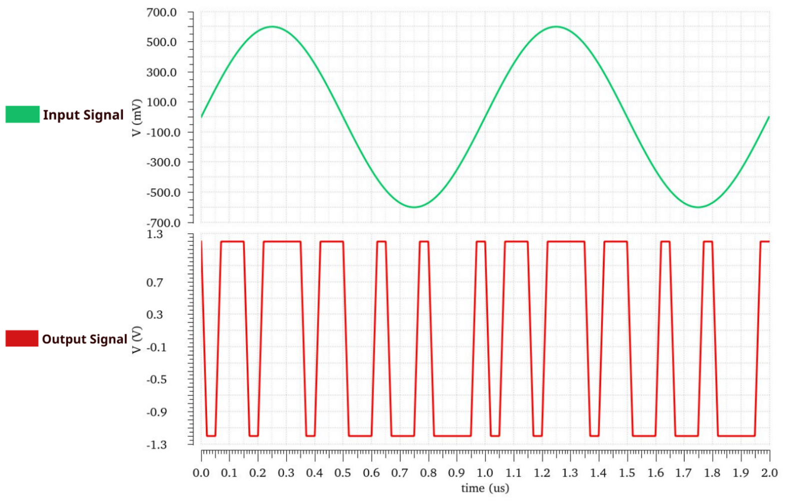
the digital signal corresponds to the actual analog sinusoidal input, demonstrating the correct operation of our 1st order $\Delta\Sigma$ modulator.

Moving on the design of the fracitonal divider, a comprehensive study was condcuted to determine the appropriate switching sequence for the prescaler ratios, which is crucial for achieving the desired fractional division. Our objective was to obtain a division of 120.5 using a presclaer that alternates between the ratios 120 and 121. To accomplish this, we employed a systematic approach.

The fractional part of the desired division, 0.5, was represneted as a



(a)



(b)

Figure 3.21: 1-bit $\Delta\Sigma$ modulator **(a)** Its schematics **(b)** Its Input-output waveforms.

fraction, resulting in 1/2. To find a common denominator for the ratios (120 and 121), we determined that the least common multiple was 14520. By converting the fractional part to the common denominator, we obtained a numerator of 7260, yielding our division ratio would have to be represented as follows $120 + 7260/14520$. Interstingly this numerator 7260 can be explained using this equation:

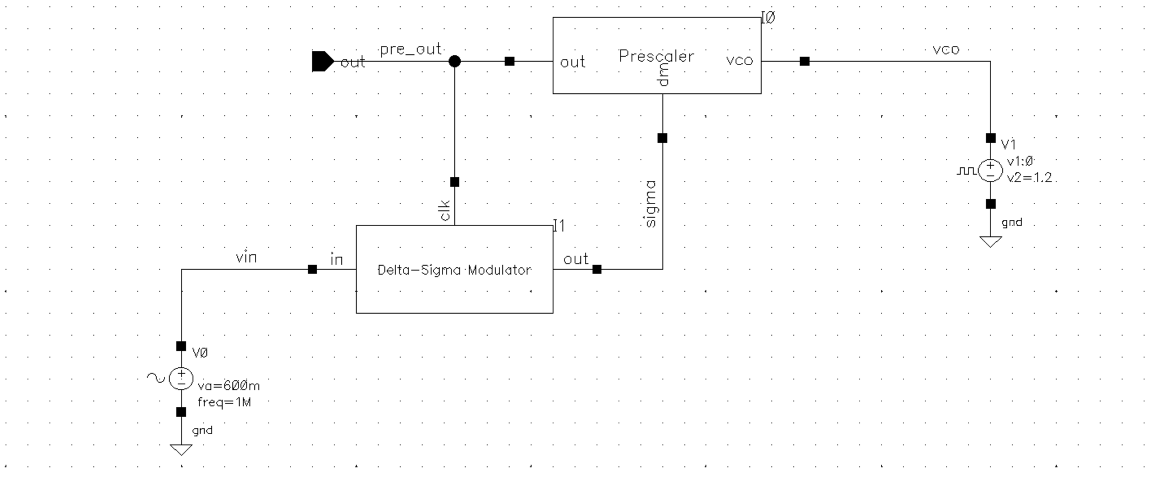
$$\begin{aligned} Numerator &= [A(N+1)] + [(B-A)N] \\ &= AN + A + BN - AN \end{aligned}$$

$$= \quad BN + A \tag{3.6}$$

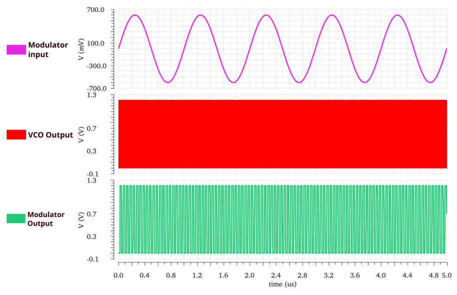
where from the equation 3.6 the factors A and B equal to 60 both. Consequently, it was determined that toggling between the ratio of 120 and 121 every 60 periods of the VCO output frequency would achieve the desired fractional division of 120.5. This relationship can be represented by the following equation:

$$\frac{A+B}{\frac{A}{N}+\frac{B}{N+1}} = \frac{60+60}{\frac{60}{120}+\frac{60}{121}} = 120.5 \tag{3.7}$$

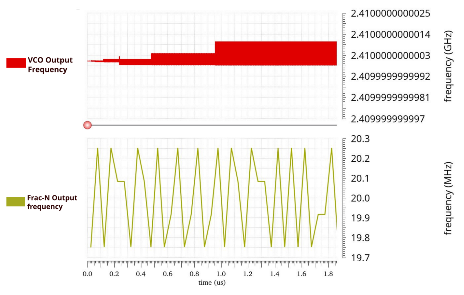
The implementation of the studied fractional divider is depicted in Figure 3.22(a), accompanied by the corresponding input-output waveforms presented in Figure 3.22(b). These waveforms demonstrate that the rising-edge of the $\Delta\Sigma$ modulator output triggers a transition in the prescaler ratio to 121, Notably, the frequency output of the prescaler toggle between two frequency outputs 20.08MHz and 19.91MHz resulting in an average frequency of 20MHz, indicating the successful attainment of the desired fraction of 120.5. this accomplishment is further supported by the VCO output frequency of 2.4GHz, as illustrated in Figure 3.22(c).



(a)



(b)



(c)

Figure 3.22: Simulation results of the fracitnal divider **(a)** Its test bench **(b)** Its Input-output waveforms **(c)** frequency-versus-time characteristics of the VCO output signal and the presclaer output signal.

3.2 Integer-N PLL

Following the individual simulation of each block, the next step involves integrating them into a system to build a PLL. This section focuses on studying a second-order PLL and determination the appropriate components for its loop filter based on the transfer function. The design specifications of our system are as follows:
Reference frequency $= 20MHz$

Output frequency = $2.4GHz$
Supply voltage = $1.2V$
VCO gain = 2π*400MHz Mrad/s/V
Based on the given reference frequency and output frequency, it becomes apparent that the division ratio is 120.
The closed-loop transfer function of a second-order PLL is given by [10]:

$$H(s) = \frac{\frac{I_p K_{VCO}}{2\pi C_1}(R_1 C_1 s + 1)}{s^2 + \frac{I_p}{2\pi} K_{VCO} R_1 s + \frac{I_p K_{VCO}}{2\pi C_1}} \tag{3.8}$$

with:

$$\omega_n = \sqrt{\frac{I_p K_{VCO}}{2\pi C_1}} \tag{3.9}$$

$$\zeta = \frac{R_1}{2}\sqrt{\frac{I_p K_{VCO} C_1}{2\pi}} \tag{3.10}$$

where ω_n is the natural frequency of the loop filter, it measures the response time of the loop.

By examining equation 3.8, we can observe that the closed-loop transfer function exhibits a low-pass response characterized by one zero and two poles. This response implies that the output retains slow input phase fluctuations while effectively filtering out rapid phase changes. Furthermore, equation 3.9 reveals a positive correlation between ζ and C_1, which demonstrates a desirable trend [10].

In order to accommodate the discrete-time nature of the loop, it is necessary to decrease the open-loop unity gain bandwidth (ω_u) to one-tenth the reference frequency, the ω_u can be expressed as:

$$\omega_u^2 = (2\zeta^2 + \sqrt{4\zeta^4 + 1})\omega_n^2 \tag{3.11}$$

To achieve a faster and more optimal response with better overshoot and ringing, we select $\zeta = 1$. This choice promotes improved transient behavior and stability within the system. substituting the ζ value in 3.11 gives us:

$$\omega_u = 2.1\omega_n \tag{3.12}$$

Thus:

$$\omega_u = 2.1\omega_n = \frac{\omega_{ref}}{10} \tag{3.13}$$

At this stage we are faced with two equations and three unknowns, which allows for some flexibility in slicing the loop parameters. One parameter that offers freedom in choice is the charge pump current, which can be adjusted to match the size of the loop filter capacitor. In this particular case, a value of $100\mu A$ was chosen for I_p. The rationale behind this specific current value will be elaborated upon in the subsequent simulation results section.

From 3.9 and 3.13 we have:

$$2.1\sqrt{\frac{I_p K_{VCO}}{2\pi M C_1}} = \frac{2\pi * (20MHz)}{10} \tag{3.14}$$

From 3.14 we obtain:

$$C_1 = \frac{I_p K_{VCO}}{2\pi M * 35.8 * 10^{12}} = \frac{100 * 10^{-6} * 400 * 10^{6} * 2\pi}{2\pi * 120 * 35.8 * 10^{12}} = 9.3pF \tag{3.15}$$

Equation 3.10 gives us:

$$R_1 = 35.9K\Omega \tag{3.16}$$

For stability purposes we select $C_2 = 0.2C_1 = 1.86pF$.

3.2.1 Behavioral modeling of the integer-N PLL

After calculating the loop filter parameters and obtaining all the required values, we proceeded to simulate the behavioral model of our PLL. The test bench configuration of the system, shown in Figure 3.23(a), included all blocks in their behavioral form. The resulting waveforms, illustrated in Figure 3.23(b), demonstrated the proper functionality of the up and down signals, as well as the expected divider output frequency of 20MHz, indicating the accurate VCO output frequency.

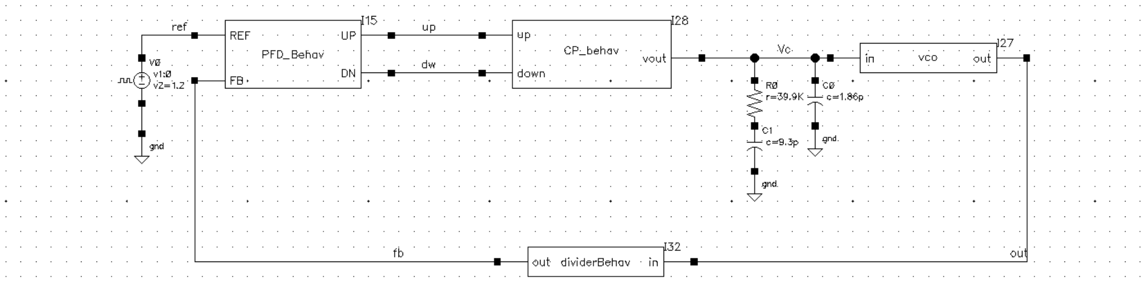
Additionally, Figure 3.24 presented the graph of the voltage control, providing insight into the behavior of our loop. Notably, the critical response of the system was evident, which aligns with our deliberate choice of a ζ value of 1. Furthermore, it was observed that the settling of the loop occurred at approximately $8\mu s$ at 0.6V, signifying the convergence of the system at a stable state.

The simulation results of the loop response graph with three different values of charge pump current $10\mu A, 100\mu A, 1mA$ are depicted in Figure 3.25, this graph offers valuable insights into the behavior of the system.

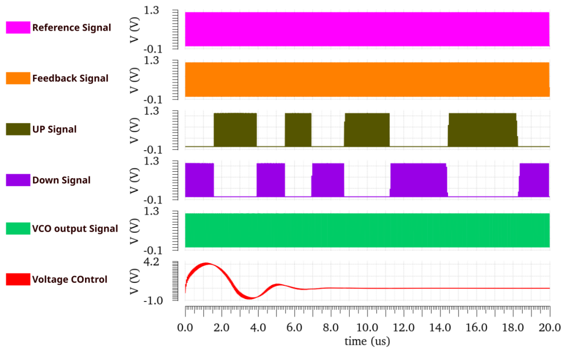
When the charge pump current is set to $10\mu A$, the peak of the response is relatively low and does not even reach the desired VDD voltage level. This suggests that the charge pump current is insufficient to generate an adequate control voltage to achieve the desired system performance. The output response lacks the necessary voltage swing to drive the system effectively-

On the other hand, when the charge pump current is increased to $1mA$, the response exhibits an overshoot beyond the desired VDD voltage level. This excessive response indicates that the charge pump current is too high, resulting in an uncontrolled increase in voltage. The system loses stability and is unable to maintain the desired damping ratio of 1. This behavior can be detrimental and lead to potential issues such as reduced system reliability, increased noise or even component damage in real PLL.

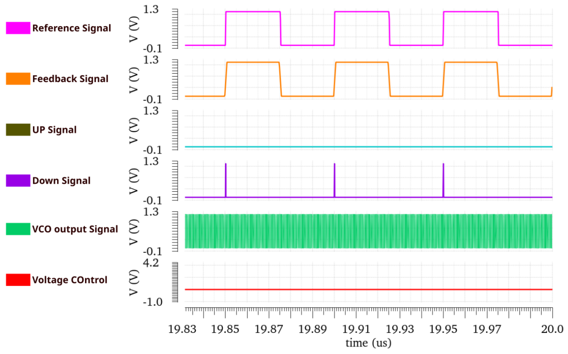
The simulation with a charge pump current of $100\mu A$ demonstrates a more favorable outcome. The peak of the response lies slightly above



(a)



(b)



(c)

Figure 3.23: Simulation results of the behavioral modeling of the integer-N PLL **(a)** test bench **(b)** Waveforms **(c)** zoomed view of the waveforms at locking state.

the VDD voltage level, which aligns well with the desired damping ratio of 1. This indicates that the charge pump of $100\mu A$ strikes a balance, providing sufficient control voltage without causing excessive overshoot

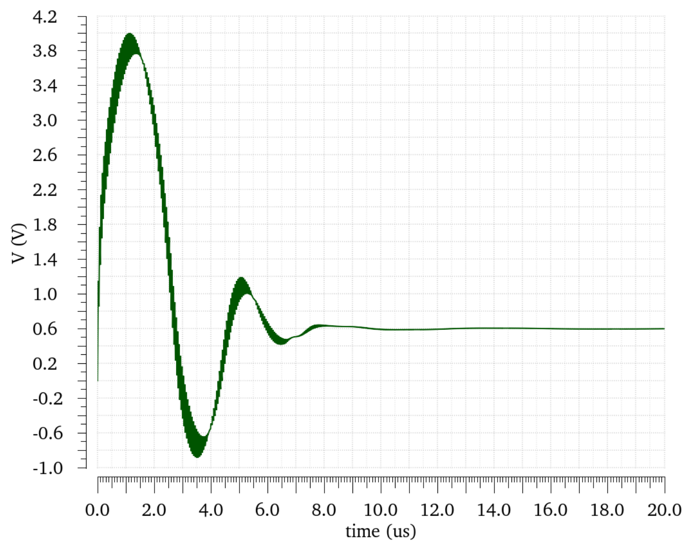


Figure 3.24: Voltage control graph of our behavioral model PLL.

or instability. The system exhibits a stable response with appropriate voltage levels, ensuring reliable and controlled operation.

Considering these circumstances, the selection of $100\mu A$ as the charge pump current is justified due to uts ability to achieve the desired system performance.

3.2.2 Integration Approach for the schematics of the integer-N divider in PLL

After simulating the behavioral model of our integer-N divider, we proceed to integrate the divide-by-120 schematics in place of the behavioral divider in order to assess its performance in comparison to the behavioral divider. Figure 3.26(a) presents the simulation results, demonstrating that the PLL functions correctly and maintains stability, while the integrated divider replicates the behavior observed in the behavioral model. Notably, we observed a noteworthy increase in the settling time, which now measures at $10\mu s$. This extended settling

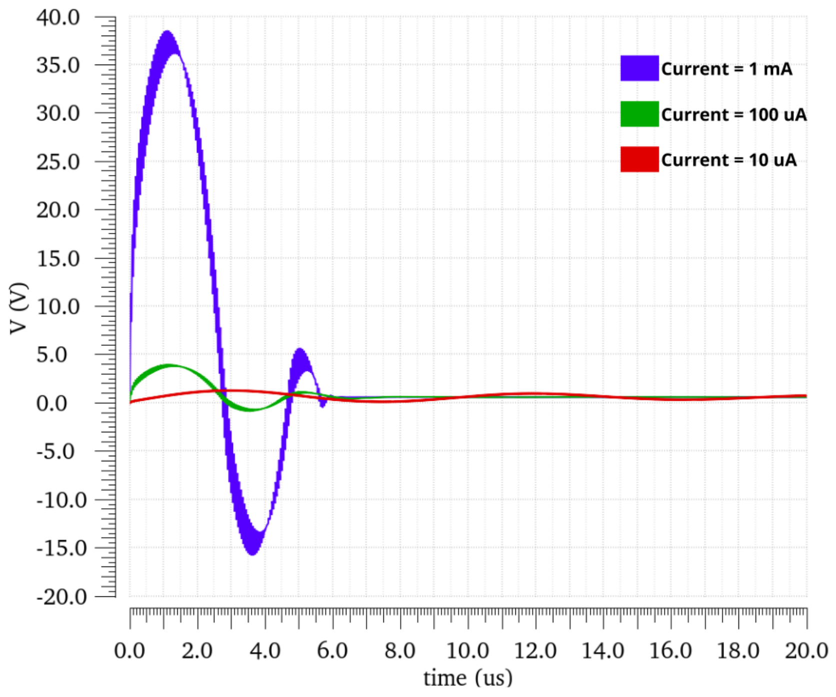
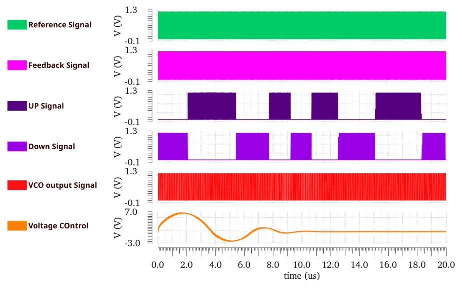
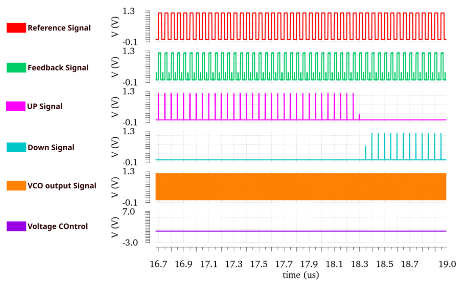


Figure 3.25: Effect of the charge pump current on the loop response.

time can be attributed to various factors, with one significant factor being the propagation delay inherent in the circuit. As the number of stages and transistors increases in the div-by-120 circuit, the total propagation delay accumulates, resulting in an extended settling time.



(a)



(b)

Figure 3.26: Simulation results of the integer-N divider schematics within the PLL **(a)** waveforms **(c)** zoomed view of the waveforms at locked state.

3.3 Fractional-N PLL

Figure 3.27 illustrates the workspace of a fractional-N PLL, where the integer-N divider has been substituted with a fractional-M divider incorporating a delta-sigma modulator and a prescaler. Figure 3.28(a)

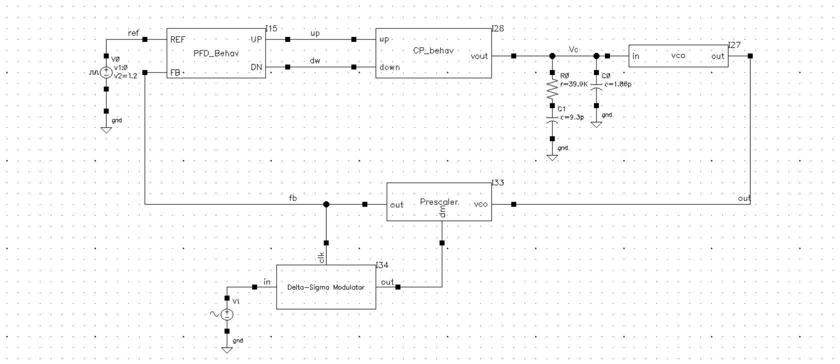
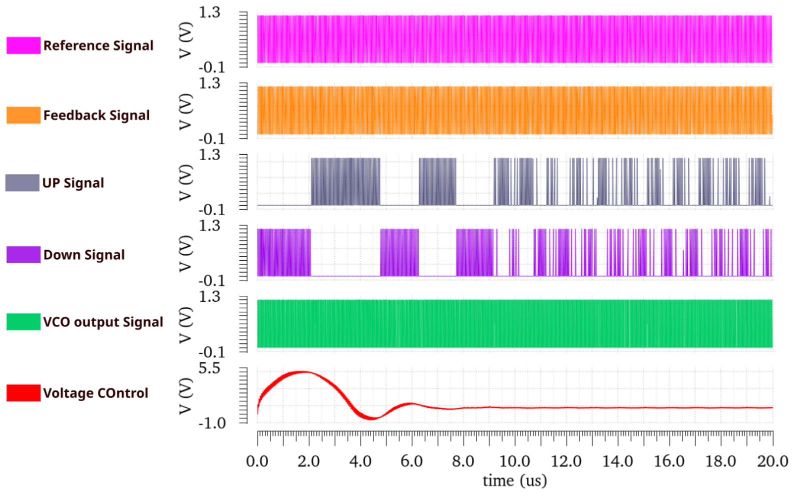


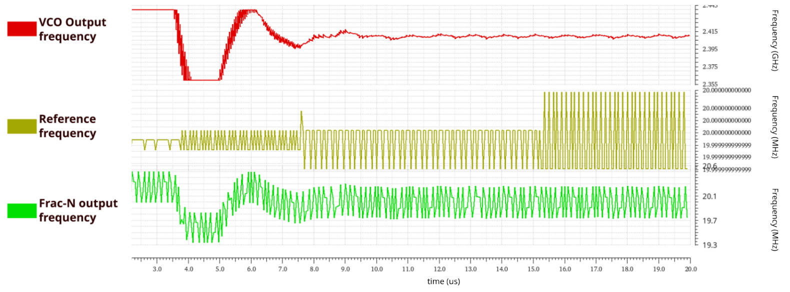
Figure 3.27: Workspace of the fractional-N PLL.

illustrates the waveforms of the fractional-N PLL operation, where an interesting observation is the settling time of approximately $6.5\mu s$. This faster settling time is commonly observed in fractional-N PLLs. The cause for this can be attributed to the wider loop filter bandwidth achievable in fractional-N PLLs.

Figure 3.28(b) represents the frequency variation of these signals: reference signal, VCO output, and the fractional divider output or the feedback signal. The analysis demonstrates that the VCO output remains stable at 2.41GHz, while the fractional divider toggles between two ratios to achieve the fraction of 120.5.



(a)



(b)

Figure 3.28: Simulation results of the fractional-N PLL **(a)** test bench **(b)** Frequency variations of: VCO output frequency, reference frequency, fractional divider frequency.

3.4 Comparison of Integer-N and Fractional-N PLLs

After conducting simulations of the two types of PLLs. It is interestingly to compare them with the available results that we have. We have observed that the overall response between the two types is generally similar. However, there are notable distinctions that differentiate them. One significant advantage of the fractional-N PLL is its higher resolution to the integer-N PLL, even it is not remarkable in our results, because of the lower order $\Delta\Sigma$ modulator. The fractional-N PLL allows for finer frequency adjustments and can achieve more precise frequency synthesis due to its ability to generate fractional division ratios. This higher resolution is particularity beneficial in applications that require accurate frequency control.

Another noteworthy difference lies in the settling time of the PLLs as depicted in table 3.2. Our simulations revealed that the fractional PLL exhibits a faster settling time compared to the integer-N PLL. This characteristics makes fractional-N PLLs well suited for applications that demand rapid frequency locking or dynamic frequency adjustments.

While both types of PLLs have their advantages and applications,

Type of PLL	Integer-N	Fractional-N
VCO frequency	$2.4GHz$	$2.41GHz$
Settling time	$8\mu s$	$6.5\mu s$

Table 3.2: Settling time comparison between Integer-N PLL and Fractional-N PLL.

the higher resolution and faster settling time of the fractional-N PLL provide distinct benefits in certain scenarios. Careful consideration of the specific requirements and constraints of the application is necessary to determine the most suitable PLL type for the optimal performance and functionality.

3.5 Conclusion and future work

In conclusion, the simulations and discussions carried out in this project have yielded valuable insights into the operation and characteristics of the PLL system. Through individual simulations of key blocks and subsequent integration into complete PLL systems, the behaviors of both Integer-N and Fractional-N dividers have been thoroughly examined.

The results indicate that the system-level performance of the Integer-N and Fractional-N PLLs shows notable similarities, with both types demonstrating effective frequency synthesis and control. However, the Fractional-N PLL exhibits higher resolution and faster settling times, making it suitable for applications that require precise frequency adjustments and rapid locking.

For future work, there are several avenues to explore. One important direction is to design all the blocks at the transistor-level and integrate them into the PLL system, moving beyond behavioral modeling. This transistor-level design will provide a more accurate representation of the circuit behavior and performance. Additionally, incorporating parameters such as phase noise and jitter, which were not considered in this study, would enhance the understanding of the PLL system's overall performance. Lastly, a fully detailed layout of the PLL would be a valuable undertaking, ensuring the practical implementation of the system and enabling further optimization and refinement.

Bibliography

[1] Mehdi Ayat, Behnam Babaei, Reza Ebrahimi Atani, Sattar Mirzakuchaki, and Babak Zamanlooy. Design of a 100mhz x2013; 1.66ghz, 0.13x00b5;m cmos phase locked loop. In *2010 International Conference on Electronic Devices, Systems and Applications*, page 154–158, Kuala Lumpur, Malaysia, Apr 2010. IEEE.

[2] William F. Egan. *Phase-Lock Basics*. Wiley, 1 edition, Oct 2007.

[3] Ian Galton. Delta-sigma fractional-n phase-locked loops.

[4] Howard Cam Luong and Jun Yin. *Transformer-Based Design Techniques for Oscillators and Frequency Dividers*. Springer International Publishing, Cham, 2016.

[5] M K Mandal and B C Sarkar. Ring oscillators: Characteristics and applications.

[6] Jan M. Rabaey. *Digital Integrated Circuit*. January 2002.

[7] Behzad Razavi. The delta-sigma modulator [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 8(2):10–15, 2016.

[8] Behzad Razavi. Tspc logic [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 8(4):10–13, 2016.

[9] Behzad Razavi. *Design of analog CMOS integrated circuits*. McGraw-Hill Education, New York, NY, second edition edition, 2017.

[10] Behzad Razavi. *Design of CMOS Phase-Locked Loops: From Circuit Level to Architecture Level.* Cambridge University Press, 1 edition, Jan 2020.

[11] Adel S. Sedra and Kenneth C. Smith. *Microelectronic circuits.* The Oxford series in electrical and computer engineering. Oxford University Press, New York; Oxford, seventh edition edition, 2015.

[12] Pin-En Su and Sudhakar Pamarti. Fractional-n phase-locked-loop-based frequency synthesis: A tutorial. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 56(12):881–885, 2009.

[13] Wikipedia contributors. Phase-locked loop — Wikipedia, the free encyclopedia, 2022. [Online; accessed 30-April-2023].

Appendix A

Verilog-A Codes

A.1 Phase-Frequency Detector code

```
module pfd(REF,FB,UP,DN);
  parameter real VDD = 1.2;
  parameter real thresh = VDD/2; //declare threshold voltage
  parameter real trise = 10p, tfall = 10p, td = 0; //rise, fall, and delay times
  input REF,FB; //declare reference and feedback signals
  output UP,DN; //declare up and down signals
  electrical REF,FB,UP,DN;
  real upval,dnval; // flags parameters to track the up and down signals state
  analog begin

    // Check if the REF signal is in rising edge and dnval is not surpassing the thresh value if yes give to the upval the VDD value otherwise assign 0 value to both parameters.
    @(cross(V(REF)-thresh,1))
      if(dnval < thresh)
        upval = VDD;
      else begin
        upval = 0;
        dnval = 0;
      end
    // Check if the FB signal is in rising edge and upval is not surpassing the thresh value if yes give to the dnval the VDD value otherwise assign 0 value to both parameters.
    @(cross(V(FB)-thresh,1))
      if(upval < thresh)
```

```
        dnval = VDD;
      else begin
        upval = 0;
        dnval = 0;
      end
        // transition of output signals UP and DN based on the flags
    parameters upval and dnval
    V(UP) <+ transition(upval,td,trise,tfall);
    V(DN) <+ transition(dnval,td,trise,tfall);
         end
endmodule
```

A.2 Charge pump code

```
module cp(UP, DN, out);
input UP, DN; //declaer input and output signals
output out;
electrical UP, DN, out;
parameter real Ip = 100u; //charge pump current
parameter real vh=+1.2;     // high state of input voltage
parameter real vl=0;      // low state of input voltage
parameter real vth=(vh+vl)/2;   // threshold voltage value
parameter real tt=1n ;  // transition time of output signal
parameter real td=0 ; // delay time from input to output
real i_flag = 0; // flag parameter that store the current state

analog begin
// cross event that occur in both state rising and falling edge of
   UP and DN signals
@(cross(V(UP)-vth,0));
@(cross(V(DN)-vth,0));
// check if UP signal is in rising edge and DN signal in 0V state if
    yes assign positive current to i_flag
if((V(UP)>vth) && (V(DN)<vth)) i_flag = Ip;
// check if DN signal is in rising edge and UP signal in 0V state if
    yes assign negative current to i_flag
else if((V(UP)<vth) && (V(DN)>vth)) i_flag = -Ip;
// otherwise assign 0A to i_flag
else i_flag = 0;
//transition of output signal based on the state of the i_flag
   parameter
I(out) <+ transition(i_flag, td, tt);
end

endmodule
```

A.3 Voltage-Conctrolled Oscillator code

```
module vco(in,out);
 // declare input outputs signals
input in; electrical in;
output out; electrical out;
parameter real vmin=0;      // input voltage that corresponds to
   minimum output frequency
parameter real vmax=1.2 from (0:inf);     // input voltage that
   corresponds to maximum output frequency
parameter real fmin=2.36G from (0:inf);         // minimum output
   frequency
parameter real fmax=2.44G from (fmin:inf);     // maximum output
   frequency
parameter real vl=0;                           // low output voltage
parameter real vh=1.2;                        // high output voltage
parameter real tt=0.01/fmax from (0:inf);       // output transition
    time
parameter real ttol=1u/fmax from (0:0.1/fmax);  // time tolerance
real freq, phase;
integer n; //flag parameter

analog begin
    // compute the freq from the input voltage
    freq = (V(in) - vmin)*(fmax - fmin) / (vmax - vmin) + fmin;

    // bound the frequency
    if (freq > fmax) freq = fmax;
    if (freq < fmin) freq = fmin;

    // bound the time step to assure no cycles are skipped (in other
     words specify the simulation steps to ensure the simulation can
   capture enough samples of output)
    $bound_step(0.6/freq);

    // phase is the integral of the freq modulo 2 pi
    phase = 2*`M_PI*idtmod(freq, 0.0, 1.0, -0.5);

    // identify the point where switching occurs, and assign to n 0
   or 1 value
    @(cross(phase + `M_PI/2, +1, ttol) or cross(phase - `M_PI/2, +1,
     ttol))
        n = (phase >= -`M_PI/2) && (phase < `M_PI/2);

    // transistion  the output based on the n parameter
    V(out) <+ transition(n ? vh : vl, 0, tt);
end
endmodule
```

A.4 Integer-N divider code

```
module div(in,out);
output out; electrical out; // declare input output signals
input in; electrical in;
parameter real vh=+1.2;   // high value voltage of output singal
parameter real vl=0;    // low value voltage of output singal
parameter real vth=(vh+vl)/2; // threshold voltage at input
parameter integer ratio=120;  // divide ratio
parameter integer dir=1 from [-1:1] exclude 0;
        // dir=1 for positive edge trigger
        // dir=-1 for negative edge trigger
parameter real tt=1n from (0:inf);  // transition time of output
   signal
parameter real td=0 from [0:inf); // average delay from input to
   output
integer count, n; // count parameter to count the cycles of the
   input signal, where is flag parameter that reutrn if the state of
    division

analog begin
    @(cross(V(in) - vth, dir)) begin
  count = count + 1; // count input transitions
  if (count >= ratio)
      count = 0;
    // if 2*count (1 periode) is greater than 120, than assign 0
   otherwise 1
  n = (2*count >= ratio);
    end
    V(out) <+ transition(n ? vh : vl, td, tt);
end
endmodule
```

A.5 Fractional-N divider code

A.5.1 Prescaler code

```
module prescaler(dm,vco,out);
input dm,vco; //dm is the delta-sigma output singal, vco is the vcoi
    output signal
output out;
electrical dm,vco,out;
parameter real vh=+1.2;
parameter real vl = 0;
parameter real vth=(vh+vl)/2;
parameter integer N =120; // ratio of N
parameter integer N1=121; // ratio of N+1
parameter real tt=1n from (0:inf);
parameter real td=15f;
integer count,n,enable;
analog begin
//check if delta-sigma output is in rising edge, if yes assign 1 to
    enable parameter otherwise 0
if (V(dm)>=vth)
  enable=1;
else begin
  enable=0;
end
// check if vco output signal is in rising edge
@(cross(V(vco)-vth,1,enable)) begin
// check the value of enable to know which division ratio to divide
if (enable==1) begin
  count = count + 1; // count input transitions
  if (count >= N)
      count = 0;
  n = (2*count >= N);
end
else begin
  count = count + 1; // count input transitions
  if (count >= N1)
      count = 0;
  n = (2*count >= N1);
end
end
V(out) <+ transition(n ? vh : vl, td, tt);
end
endmodule
```

A.5.2 Difference amplifier code

```
module diff(vin,v_dac,vdiff);
input vin,v_dac; // vin is the analog input signal, v_dac is the
    output signal of the digital-to-analog converter
output vdiff;
electrical vdiff,vin,v_dac;

```

```
analog begin
V(vdiff)<+ V(vin)-V(v_dac);
end
endmodule
```

A.5.3 Integrator code

```
module inegrator(in, out);
input in;
output out;
electrical in, out;
parameter real out0 = 0;
parameter real gain = 1;

   analog
    V(out) <+ gain*idt(V(in), 0) + out0;
endmodule
```

A.5.4 Quantizer code

```
module quantizer(vclk,vint,out);
input vclk,vint;
output out;
electrical vclk,vint,out;
parameter real vth=0.0 ;
parameter real vtrans=0.6 ;
parameter real trise=20n from (0:inf);
parameter real tfall=20n from (0:inf);
parameter real tdel=0.0 from [0:inf);
real vout_val;
real hi, lo;

analog begin
      @ ( initial_step ) begin
         vout_val = 1.2;
         hi=1.2;
         lo=-1.2;
      end
//check if clk voltage signal is in rising edge based on the
   transition voltage value, if yes assign high state otherwise
   assign low state
@ (cross(V(vclk)-vtrans, 1.0))begin
 if (V(vint) > vth)
            vout_val = hi ;
   else
            vout_val = lo ;
```

```
end
V(out) <+ transition(vout_val, tdel, trise, tfall);
end
endmodule
```

A.5.5 Digital-to-Analog code

```
module d2a(dout,v_dac);
input dout; electrical dout;
output v_dac; electrical v_dac;
parameter real vtrans=0.5 ;
parameter real vref = 1 from [0:inf);
parameter real v_high=1;

analog begin
V(v_dac)<+ v_high*V(dout);
end
endmodule
```