



On the Design Automation of RF Circuits for Mobile Cellular Transceivers

Dimo Martev

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:

Prof. Dr. Andreas Herkersdorf

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Ulf Schlichtmann
2. Priv.-Doz. Dr.-Ing. Stephan Henzler

Die Dissertation wurde am 10.09.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 03.07.2019 angenommen.

To my parents.

Abstract

Modern cellular transceivers feature complex radio-frequency (RF) clock generation and distribution circuits, where the purity of the RF-clock signal is critical for the performance of the transceiver. The prevalent design practice is to design all performance critical building blocks using full-custom design techniques, to tightly control the design parameters. A large subset of the building blocks in the RF-clocking subsystem operate on digital signals and are therefore attractive for automated design with the aid of digital design tools. However, the commercially available tools have shortcomings when dealing with very high-speed circuits or circuits with phase noise requirements.

This thesis proposes an automated design technique for key building blocks in RF clock generation subsystems, addressing the blocks' high-speed and low power requirements. Furthermore, a methodology for automated phase noise aware-design of RF clock distribution circuits utilizing a commercial digital design toolchain is proposed. The methodology provides a rapid approach for phase noise estimation, a buffer insertion and gate sizing algorithms operating under phase noise constraints.

Contents

Contents	vii
List of Figures	xi
List of Tables	xiii
Acronyms	xv
1 Introduction	1
1.1 Contributions of This Work	3
1.2 Organization	4
2 Basics of Radio-Frequency ICs	5
2.1 Communication Systems	5
2.1.1 Wireless Communication Standards	6
2.1.2 Modulation Schemes	7
2.2 Transceiver Architecture and Requirements	9
2.2.1 General Overview	9
2.2.2 System Requirements	10
2.2.2.1 Receiver	10
2.2.2.2 Transmitter	13
2.3 Noise	14
2.3.1 Noise Sources in MOSFET Transistors	15
2.3.2 Phase Noise	15
2.3.2.1 Phase Noise in CMOS Circuits	17
2.3.3 Effects of Phase Noise on Communications Systems	19
2.4 Selected Transceiver Building Blocks	20
2.4.1 Frequency Synthesizer	20
2.4.1.1 Phase Locked Loops	20
2.4.1.2 Digitally Controlled Oscillator	23
2.4.1.3 Noise in PLLs	24
2.4.2 Mixer	25
2.4.3 RF Clock Distribution	26
3 ASIC and Full-Custom Design Flows	29
3.1 Full-Custom Design	29

3.2	ASIC Design Flow	31
3.2.1	Basic Principles of Static Time Analysis	32
3.2.2	Physical Design	35
3.3	Motivation for Applying the ASIC Flow in RF Cricuit Design	37
4	Automated Design of PLL Building Blocks	39
4.1	Multi-Modulus Divider	39
4.1.1	MMD Architecture	40
4.1.2	Constraints Development	43
4.1.2.1	Critical Paths Constraining	44
4.1.3	Clock Tree Synthesis	44
4.1.4	VHDL Design	45
4.1.5	MMD Performance and Discussion	45
4.2	Time-to-Digital Converter	48
4.2.1	TDC Architecture	50
4.2.1.1	TDC Core	50
4.2.1.2	Control Logic	51
4.2.1.3	Postprocessing	52
4.2.2	Synthesis and APR	52
4.2.2.1	TDC Core	52
4.2.2.2	Control Logic	55
4.2.2.3	Digital Logic	55
4.2.3	TDC Performance and Discussion	55
5	Automated Phase Noise-Aware Design of RF Clock Distribution Circuits	59
5.1	Static Phase Noise Analysis	59
5.1.1	Concepts of SPNA	60
5.1.2	Operation	60
5.1.3	Cell Characterization	62
5.1.4	Interconnect Parasitics	63
5.1.5	Handling the Logical Function of Gates	65
5.1.6	Limitations	66
5.1.7	SPNA Workflow	66
5.1.8	Experimental Results	68
5.2	Buffer Insertion for Phase Noise Minimization	69
5.2.1	Algorithmic Framework	70
5.2.1.1	Description	70
5.2.1.2	Power Minimization	74
5.2.1.3	Signal Transition Times	75
5.2.1.4	Miscellaneous	76
5.2.1.5	Buffer Insertion Problem	77
5.2.1.6	Complexity analysis	77

5.2.2	Buffer Insertion Workflow	77
5.2.2.1	Estimation of Phase Noise	78
5.2.3	Results	79
5.3	Gate Sizing for Power Optimization Under Phase Noise Constraints . . .	81
5.3.1	Problem Formulation	82
5.3.2	Gate Sizing	82
5.3.2.1	Constrained Optimization	84
5.3.3	Results	85
5.4	Methodology for Phase Noise-Aware Design	86
5.4.1	Physical Design	88
5.4.2	Limitations	89
5.4.3	Advantages of the Flow	89
5.4.4	Results	89
6	Summary	91
	Bibliography	93

List of Figures

1.1	RF design hexagon.	2
1.2	Design automation feasibility of RF building blocks.	3
2.1	Elements of a wireless communication system.	6
2.2	Frequency bands of the most common cellular standards.	7
2.3	Modulation of the signal characteristics.	8
2.4	Polar to rectangular conversion.	8
2.5	Square QAM constellation.	8
2.6	Direct conversion transceiver.	11
2.7	Adjacent channel interference.	12
2.8	Intermodulation between two signals.	13
2.9	Distorted 4-QAM and Error Vector Magnitude.	14
2.10	Spectral mask.	14
2.11	Sine wave in time and frequency domain.	16
2.12	Synchronous and accumulating jitter in time domain.	17
2.13	A CMOS inverter and its transient response.	18
2.14	Voltage noise and jitter.	19
2.15	Reciprocal mixing.	19
2.16	PLL block diagram.	21
2.17	PLL waveforms.	21
2.18	All-digital PLL block diagram.	22
2.19	A symmetric LC DCO.	24
2.20	Noise in PLLs.	25
2.21	Switching mixer.	26
2.22	A generic RF clock distribution network.	27
3.1	Custom design flow.	30
3.2	ASIC design flow.	31
3.3	Combinational timing arcs.	33
3.4	Sequential cell timing arcs.	33
3.5	Timing path example.	33
3.6	Timing diagram for the exemplary circuit of Fig. 3.5.	34
3.7	Input/output ports constraining.	34
3.8	Steps in physical design.	35
4.1	2/3 prescaler.	40

List of Figures

4.2	State transition diagram of the 2/3 prescaler.	41
4.3	Block diagram of the implemented MMD.	41
4.4	MMD timing.	42
4.5	Critical paths in stages I1 and I2.	43
4.6	Clock tree synthesis results.	45
4.7	Analog simulation of the MMD.	47
4.8	TDC block diagram.	50
4.9	TDC control logic.	51
4.10	TDC control logic timing diagram.	51
4.11	TDC core layout and routing.	53
4.12	Structure of (a) a std. cell register and (b) two latches.	54
4.13	DNL simulation (red) and measurement (blue).	56
4.14	INL simulation (red) and measurement (blue).	57
5.1	Unateness of arcs.	60
5.2	Exemplary netlist.	61
5.3	Graph representation of the exemplary netlist from Fig. 5.2.	61
5.4	Cell characterization testbench.	62
5.5	RC model of an interconnect.	64
5.6	Driving point interconnect modeling.	64
5.7	Reduced order model for one endpoint.	65
5.8	Exemplary binary expression tree for $Z = (A \text{ OR } B) \text{ AND } !C$	66
5.9	Workflow of SPNA.	67
5.10	Buffer insertion problem.	71
5.11	The buffer insertion algorithm.	72
5.12	Visual representation of the buffer insertion algorithm.	73
5.13	Augmented binary search tree for orthogonal range query.	74
5.14	Propagating signal transition times upstream.	75
5.15	Handling transition times during pruning.	76
5.16	Handling placement blockages.	76
5.17	Workflow for buffer insertion.	78
5.18	Phase noise and runtime for different wire segment lengths.	80
5.19	Basics of genetic algorithms.	83
5.20	The gate sizing algorithm SPNA-GS.	85
5.21	Example of the operation of the hybrid approach.	85
5.22	Design flow for automated phase noise-aware design.	87
5.23	Workflow of the proposed phase noise-aware design methodology.	88

List of Tables

4.1	MMD implementations based on different design techniques.	46
4.2	Purpose-built MMD vs. design reuse.	47
4.3	STA timing report.	48
4.4	Setup times for standard cell register vs 2 latches.	55
4.5	Summary of TDC results.	57
5.1	Noise estimation on industrial test cases in analog environment.	68
5.2	Noise estimation on test cases in digital toolchain.	69
5.3	Buffer insertion summary.	80
5.4	Buffer insertion runtime.	81
5.5	SPNA based gate sizing summary.	86
5.6	Comparison with manual design.	90

Acronyms

3GPP	3rd Generation Partnership Project
ACLR	Adjacent channel leakage ratio
ADC	Analog-to-digital converter
ADPLL	All-digital phase-locked loop
APR	Automated placement and routing
ASIC	Application-specific integrated circuit
AWGN	Additive white Gaussian noise
BPF	Band pass filter
BST	Binary search tree
BTS	Base transceiver station
CMOS	Complementary metal–oxide–semiconductor
CTS	Clock tree synthesis
DAC	Digital-to-analog converter
DCO	Digitally controlled oscillator
DDS	Direct digital synthesis
DLL	Delay-locked loop
DUT	Device under test
EVM	Error vector magnitude
FDD	Frequency division duplexing
FM	Frequency modulation
FSM	Finite state machine
HDL	Hardware description language
IC	Integrated circuit
LNA	Low noise amplifier
LO	Local oscillator

Acronyms

LTE	Long Term Evolution
MMD	Multi-modulus divider
PLL	Phase-locked loop
PSD	Power spectral density
PVT	Process, voltage and temperature
QAM	Quadrature amplitude modulation
QoR	Quality of results
RF	Radio frequency
SNR	Signal-to-noise ratio
STA	Static timing analysis
TDC	Time-to-digital converter
UE	User equipment

1 Introduction

Electronic communication systems play an important role in the modern society. The advances in integrated circuit (IC) technology have made it possible for almost every person on earth to own a smartphone – a hand-held personal computer with Internet connectivity and voice call capabilities. The enabler for this high level of integration is the shrinking device size and low price of the complementary metal–oxide–semiconductor (CMOS) technology.

CMOS was invented in 1963 [WS63] and quickly became the dominant IC technology due to its lower static power consumption and higher device density compared to bipolar integrated technology. The constantly growing digital market drives steady supply chains for CMOS, lower cost and ever-advancing technology nodes. Gordon Moore, co-founder of Intel Corp., made the observation in 1965 that the number of devices in an integrated circuit doubles every year [Moo65]. The rate of progress has decreased since then, however the down-scaling continues to modern days, due to its economic impact – lower components price, reduced power consumption and circuit delays, associated with the smaller geometries.

Historically, radio frequency (RF) circuits used gallium arsenide (GaAs) based technology due to the high transition frequency f_T and insulating substrate, or bipolar silicon for their lower cost. However, in the last few decades, CMOS transistors became comparable to bipolar transistors in transition frequency. A great deal of research was done in the 1980s and 1990s to commercialize the RF CMOS [Abi04] that enabled the usage of CMOS for RF circuits and for creating systems on chip (SoC). SoC is a single chip with digital, mixed-signal and RF signals that offers higher integration, smaller footprint and lower overall cost.

In deep submicron CMOS technologies encoding information in timing edges is preferred over voltage levels [SW⁺05]. This is justified by the increasing speed of the gates combined with the scaling down of the supply voltage from one technology node to the next, which degrades the noise immunity and amplitude accuracy. Additionally, CMOS circuits come with lower transconductance g_m , are affected by substrate noise and have higher noise. Hence, modern wireless RF transceivers tend to be fully digital in nature or at least heavily digitally assisted. Even if digital architectures and digital modulation schemes are commonly in use, the noise in some digital signals in the RF clock generation and distribution modules should be accounted for, for reasons that will be discussed in the following chapters.

The 'RF design hexagon' (Fig. 1.1), defined in [Raz98], describes the trade-offs in RF design very well. Each one of the given six design parameters can be traded with

1 Introduction

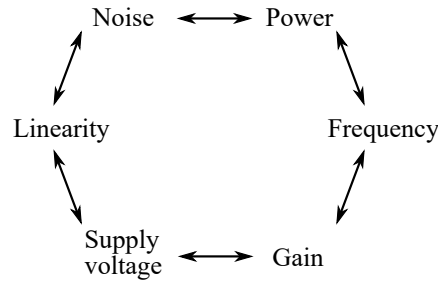


Figure 1.1: RF design hexagon.

one or more of the others in some way. For digital circuits, the focus is on the power - frequency trade-off, whereas in the digital RF circuits mentioned above there is a second important trade-off between power and noise. For digital RF, as well as standard digital logic circuitry, the linearity and gain parameters are not applicable.

There are two main design paradigms in IC design: *full-custom* (or just custom) design and *application-specific integrated circuit (ASIC) design* (semi-custom). The custom design is manual work intensive, whereas semi-custom relies heavily on design automation. Each of the two design methods has its own advantages and disadvantages.

Custom circuit design offers fine granularity of transistor and wire sizing, but requires high effort and cost in terms of headcount of designers and layouters, time, and computing power. Therefore implementing and optimizing a modern transceiver solely using full-custom methods is not feasible.

On the other hand, ASIC tools and their powerful algorithms for automated placement and routing (APR), and clock tree synthesis, mainly focus on timing performance, which serves very well for high performance circuits. These tools are capable of estimating dynamic and leakage power, based on cell characterization; voltage scaling is also possible when the libraries are characterized for supply voltages, but this is not an optimization parameter. However, ASIC tools are tuned to work for multi-million gate integrated circuits and the produced output cannot compete with the results of dedicated human effort for designs of manageable size.

Fig. 1.2 shows the building blocks that can be found in modern transceivers, loosely organized in groups by level of abstraction and gate count. In red are the purely analog building blocks such as bandgap voltage references and amplifiers that require extensive SPICE simulations and cannot be designed automatically with the current state of the design tools. At the far right are the high-performance circuits described in high level of abstraction that are in the traditional scope of ASIC tools. In blue in the middle are performance critical building blocks in RF transceivers, operating on digital signals, but are designed using full-custom methodology to meet low power, high-speed and/or noise requirements. Exemplary building blocks that belong to this class of circuits are frequency synthesizers, clock distribution network, asynchronous high-speed dividers and counters, cell arrays with their decoder logic, to name but a few. These blocks have a

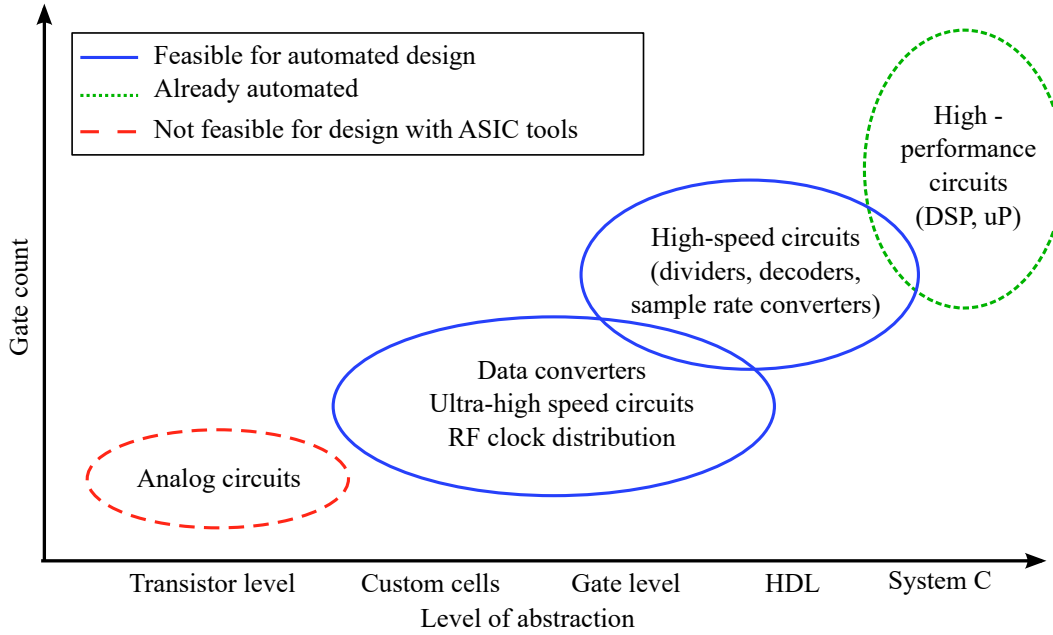


Figure 1.2: Design automation feasibility of RF building blocks.

level of abstraction higher than transistor level analog design and are feasible to be implemented with ASIC methodology if the limitations of the tools are overcome.

Hence, enabling the implementation of the above mentioned class of circuits by means of digital ASIC design tools is desirable. This will allow to completely exploit the benefits of a fully digital transceiver architecture by utilizing the advantages of the ASIC implementation methods: technology independent hardware description language (HDL) models, fast implementation speed, exhaustive timing coverage and relatively high design complexity that can be covered by a single designer (See Section 3.2).

1.1 Contributions of This Work

Taking all above mentioned aspects into consideration, the main challenge that needs to be addressed to allow the automated design of RF blocks is bridging the gap between RF requirements and design techniques, on one side, and ASIC based implementation flows on the other side, to exploit the best out of both worlds.

In this work, methods for the design of two building blocks for RF clock synthesis – multi-modulus divider (MMD) and time-to-digital converter (TDC) – within the ASIC toolchain are demonstrated. This includes selecting architectures that satisfy the low power, high frequency and low jitter requirements of the modules. Special timing and physical constraints are developed to address the specifics of the selected architecture, while guaranteeing the reproducibility and scalability of the resulting design.

1 Introduction

Furthermore, a methodology for automated design of RF clock distribution networks is developed. It includes

- a fast and accurate method for estimation of phase noise in CMOS circuits, based on pre-characterized standard cells;
- a buffer insertion algorithm for interconnect trees; and
- a gate sizing algorithm for phase noise minimization in CMOS circuits.

In this methodology, analog simulations are not necessary during the design process and a single designer can conduct the synthesis and layout of the block within a commercial digital ASIC design toolchain. This offers significant reduction of the design time.

Parts of this work were published in [MHS16b], [MHS16a], [MHS17a], [MHS17b], and [MHS18].

1.2 Organization

The remainder of this thesis is organized as follows. Chapter 2 gives a theoretical background on the basics of RF communication systems, transceiver architectures, and RF clock synthesis. The main requirements and performance measures for transceivers are also discussed. In Chapter 3 the full-custom and ASIC design flows are introduced and the advantages and limitations of the ASIC design flow are reviewed. Chapter 4 presents the automated design of two building blocks for PLLs – MMD and TDC. Chapter 5 describes a new methodology for automated design of RF clock distribution circuits under phase noise constraints. Finally, Chapter 6 concludes the thesis.

2 Basics of Radio-Frequency ICs

This chapter gives an overview of RF transceivers, RF clock synthesis and RF clock distribution. The intention of the chapter is to serve as introduction to RF transceivers, and to present the key concepts related to this work – the reader is referred to [Zha15, Raz98, OWY83, Gu05] for more profound and elaborate information on the topic.

First, a brief introduction of communications systems in general and wireless communication in particular are discussed in Section 2.1. Afterwards, transceiver architectures and requirements are presented in Section 2.2. Next, noise, its origins and its effects on the performance of transceivers are discussed in Section 2.3. Finally, Section 2.4 focuses on the transceiver building blocks relevant for this work.

2.1 Communication Systems

The modern world relies on *communication systems* to transmit information between people or machines. The most convenient and flexible type of communication is the *radio communication* that uses air as a physical transmission medium, i.e. as communication channel and does not require the deployment of connection wires to the end nodes. Air attenuates audible frequencies, but propagates higher frequencies over large distances. However, very high frequencies are attenuated by the moisture content in the air, therefore there is a limited range of frequencies in the electromagnetic spectrum that can be used for wireless transmission. These frequencies are referred to as the radio spectrum and lie in the range from 3 kHz to 300 GHz.

The radio spectrum should be shared between military, industrial, scientific and commercial services, hence it is considered a "scarce natural resource" [Rya05]. Spectrum usage is regulated by the national governments, which assign bands of the spectrum for certain usage through administrative licensing. The part of the spectrum assigned for commercial wireless communication is very limited, leading to high licensing fees. Hence, effective usage of the available bandwidth, i.e. spectral efficiency, is of highest importance. The spectral efficiency is defined as the amount of information that can be transmitted over a given bandwidth and it is usually measured in bits per second per hertz, or bits/s/Hz.

A communication system consists of a transmitter, receiver and physical channel. The physical channel is the medium connecting the receiver and transmitter and can be a copper cable, optical fiber or air, as in the case of wireless communication. Typically the channel is not perfect, and introduces noise and distortion in the system [Zha15]. The elements of a wireless communication system are shown in Fig. 2.1.

2 Basics of Radio-Frequency ICs

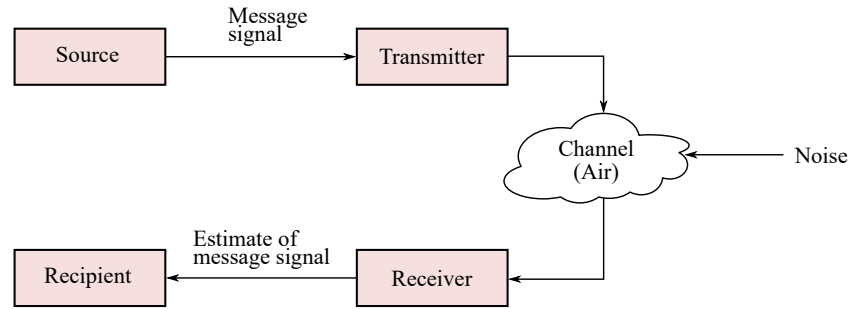


Figure 2.1: Elements of a wireless communication system.

In his influential work from 1948 [Sha48], Shannon gave the fundamental limit of the capacity C of an additive white Gaussian noise (AWGN) channel

$$C = B \cdot \log_2(1 + SNR) \quad [bits/s], \quad (2.1)$$

where B is the channel bandwidth, and the signal-to-noise ratio (SNR) represents the ratio of signal power to noise power. Besides AWGN, other factors that limit the system performance are the interference and multi-path fading. A well designed wireless communication system must offer reliable connection and satisfy performance metrics such as energy efficiency, spectral efficiency, coverage, error performance, and coexistence with other devices.

In a mobile communication system, the device held by a user is mobile and is referred to as user equipment (UE). Normally, a UE communicates with a fixed unit, a so called base transceiver station (BTS), or simply base station. Instead of only one fixed unit covering the whole area, usually the area to be covered is split in smaller regions – cells, and each cell is served by a separate BTS on a given frequency range. Neighboring cells operate on different frequencies allowing for frequency reuse. This scheme offers also a power advantage, as the distance between the UE and BTS is reduced. Such a network is called a *cellular network*.

2.1.1 Wireless Communication Standards

The rules for communication between endpoints are established with the aid of *wireless standards*. They define the modulation, timing, coding, multiple access techniques and requirements for the equipment. There exists a wide range of wireless standards with different purpose, range and data rates. The standards for cellular networks started emerging in the 1980s, with the first generation mobile technology (1G) being analog. It had low spectral efficiency, low data rates and unsecure connection, as there was no encoding of the transmitted messages. Therefore, in the early 1990s, it was replaced by the second generation (2G), which was based on digital modulation techniques, drastically increased the spectral efficiency and included encryption techniques for enhanced security. The most prominent 2G standard is GSM, which originated in Europe and is,

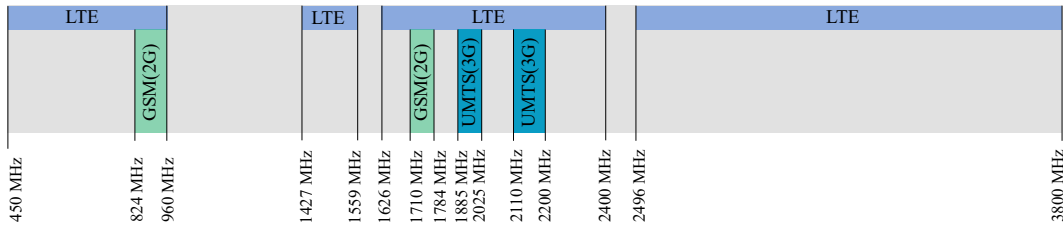


Figure 2.2: Frequency bands of the most common cellular standards.

to date, the most widely used cellular standard in the world [5G 17]. The next step was the development of the third generation (3G/UMTS), introduced in year 2000. Its goal was to increase the bandwidth and data transfer rates to allow for simultaneous voice and data transmission. It is an incremental step over 2G, reusing most of the infrastructure. The fourth generation – 4G/Long Term Evolution (LTE), specified in 2008, targets even higher data rates; it added new frequency bands, and further increased the spectral efficiency. Fig. 2.2 summarizes the frequency bands used for the different cellular standards.

2.1.2 Modulation Schemes

The message signal that has to be transmitted usually has near-zero frequency range, and is termed as *baseband signal*. The baseband signal has to be converted to RF for transmission over the communication channel, a task carried out by a *transceiver* (a portmanteau of *transmitter* and *receiver*). To achieve that, the baseband signal is being embedded into another, high frequency signal that can be transmitted. The process of embedding one signal into another signal (carrier) is known as *modulation*, and the complementary process of extracting the information from a carrier is called *demodulation* [OWY83]. There are various modulation schemes that can be used, where a data signal changes one or more of the properties of a carrier signal – amplitude, phase, and frequency.

There are two basic types of modulation possible - digital and analog, differing in the type of the signal that has to be transmitted. As the name suggests, in analog modulation the input signal is continuous, whereas in digital modulation the input is discrete in time and amplitude. Digital modulation is preferred over analog modulation for various reasons: it is more spectrally efficient, has better noise and interference immunity. Additionally, it allows for improved security by using coding techniques to prevent jamming. The digital modulation allows for complex digital signal processing, error control and correction codes can be applied to improve the performance. Figure 2.3 shows the three basic digital modulation schemes, termed *keying* to distinguish them from their analog counterparts. The source is a sequence of symbols with k finite states, hence each symbol contains n bits of information, where $n = \log_2 k$. If the source

2 Basics of Radio-Frequency ICs

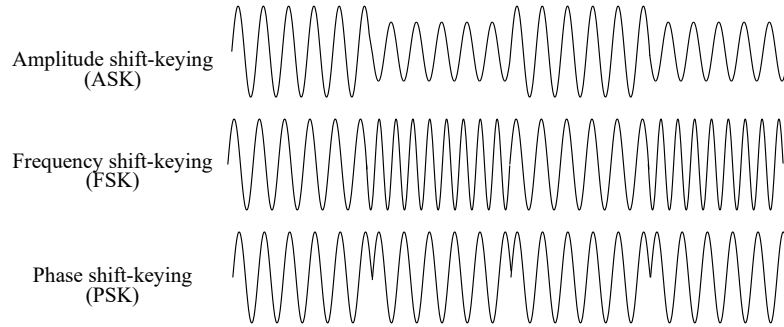


Figure 2.3: Modulation of the signal characteristics.

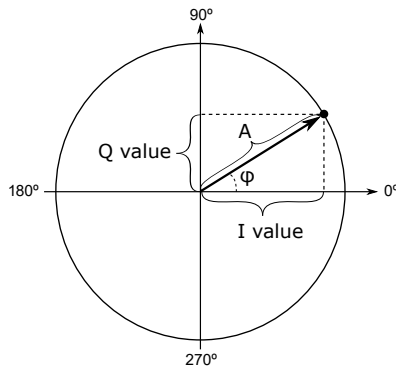


Figure 2.4: Polar to rectangular conversion.

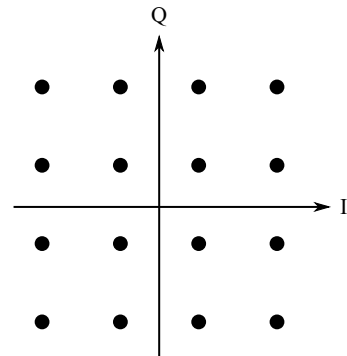


Figure 2.5: Square QAM constellation.

produces the symbols at rate R_s symbols per second, then the transmission rate is $R_b = R_s n$.

A common representation of sinusoids is the complex quantity called *phasor* [Gli11], containing the amplitude A and phase φ of the signal. A phasor can be presented as a vector in polar diagram. Alternatively, the phasor can be presented in Cartesian (rectangular) form, with the two axes called I and Q, where I is the in-phase component of the waveform and Q is the quadrature component. Converting between the two representations can be done trivially using trigonometric identities. Fig. 2.4 shows a phasor in polar form and its decomposition to I and Q components.

One of the most popular modulations, used in all modern cellular standards, is quadrature amplitude modulation (QAM), where amplitude and phase modulation are used simultaneously for increased spectral efficiency. However, instead of directly changing the phase and amplitude of the signal, the signal is split in I and Q components. As the two signals are orthogonal to each other, they do not interfere and can be sent into a single composite signal and then processed independently at the receiver part. The independent processing of the channels makes the circuit relatively simple.

Modulating the amplitude of the two out-of-phase signals and combining them can be viewed as both amplitude and phase modulation. The sent signal will then be

$$s(t) = \text{Re}\{[I(t) + jQ(t)]e^{j2\pi f_0 t}\} = I(t)\cos(2\pi f_0 t) - Q(t)\sin(2\pi f_0 t), \quad (2.2)$$

where $I(t)$ and $Q(t)$ are the modulating signals, and f_0 is the carrier frequency.

At the receiver side, the $I(t)$ and $Q(t)$ signals can be extracted by multiplying by $\cos(2\pi f_0 t)$ and $\sin(2\pi f_0 t)$, accordingly. For demodulating the $I(t)$ signal, the received signal then becomes

$$\begin{aligned} r(t) = s(t)\cos(2\pi f_0 t) &= I(t)\cos^2(2\pi f_0 t) - Q(t)\sin(2\pi f_0 t)\cos(2\pi f_0 t) = \\ &= \frac{1}{2}I(t) + \left[\frac{1}{2}I(t)\cos(4\pi f_0 t) - Q(t)\sin(4\pi f_0 t) \right]. \end{aligned} \quad (2.3)$$

Low-pass filtering $r(t)$ leaves only the $I(t)$ term. Analogously, multiplying the signal $s(t)$ by a sine wave will extract the $Q(t)$ term.

A *constellation diagram* is a useful representation of the modulated signal, commonly used for digital modulations. It displays the modulated signal as a two-dimensional scatter diagram, the possible symbol positions at the symbol sampling instants. Each constellation point represents a possible symbol. As the data is usually binary, the number of points is a power of 2. Fig. 2.5 shows a square constellation diagram of QAM with 16 states (4 bits/symbol), thus referred to as 16-QAM.

A variety of forms of QAM exist, including 8, 32, 64, 128 and 256 QAM. Higher number of constellation points enables the transmission of more bits per symbol. However, if the energy of the constellation stays the same, the constellation points should be placed closer together, which makes the transmission more susceptible to noise. Therefore, the base station selects higher order constellations when possible, but automatically switches to lower order constellations if the SNR is low.

2.2 Transceiver Architecture and Requirements

2.2.1 General Overview

There are various transceiver architectures available, all of them having distinct advantages and disadvantages. The main characteristics and design challenges of the existing transceiver architectures are discussed for example in [Raz96a, Raz03, Meh01].

One architecture that has gained popularity in recent years due to its simplicity is the so called *direct conversion transceiver*. For the same reason it was selected as an exemplary architecture in this work. A block diagram of one possible realization of a direct conversion transceiver is shown in Fig. 2.6. The upper part of the block diagram shows the receiver chain, and the lower part is the transmitter.

Most cellular communications standards require the use of a frequency division duplexing (FDD) scheme to enable bidirectional communication, hence a transmitter and

receiver should operate simultaneously and independently on different frequencies. To save area and components, the antenna is shared between the receiver and the transmitter with the aid of a *duplexer*. A duplexer is a three port network containing two filters – a receiver and a transmitter filter – that isolate the receiver from the transmitter. The transmitter filter removes out of band emissions¹, and the receiver filter removes interference coming from the transmitter chain.

In the receiver the signal is passed through a band pass filter (BPF) for band selection. Subsequently the signal is amplified by a low noise amplifier (LNA). The local oscillator signal, coming from the Rx synthesizer is then applied to the two mixers for I and Q with a phase shift of 90° ($\pi/2$). The mixer² is a non-linear device that does frequency conversion from RF to base-band and vice versa. It mixes the local oscillator clock and the incoming or outgoing signal for the receiver or transmitter, accordingly.

The signal is down-converted into I and Q channel signals by a quadrature demodulator. The I and Q signals are then converted to digital by an analog-to-digital converter (ADC) and further filtered to reduce the influence of nearby interferers.

In the transmitter, the baseband I and Q signals are first converted to analog and low-pass filtered to eliminate aliasing. Then they are up converted to the carrier frequency with the aid of the mixers and summed together to form a QAM modulated signal. A band pass filter suppresses unwanted emissions and noise in the composite RF signal. The signal is then amplified to the required power level by the programmable gain power amplifier (PA). The matching network provides a maximum power transfer between the PA and the antenna.

2.2.2 System Requirements

The technical specifications for both the base stations and the user equipment are given by a standardization body for each communication standard. For example the standardization body for 3G and 4G is called 3rd Generation Partnership Project (3GPP). The specifications define the test cases for each standard. Testing the system according to the defined test cases ensures reliable connection, and guarantees that there is no unintentional excessive generation and reception of interfering signals. For example, the minimum requirements for an LTE system for each band and mode are defined in [3gp11]. Each commercial transceiver must pass the tests for compliance before it can be deployed. The most important characteristics and requirements for receivers and transmitters are summarized in the sections below.

2.2.2.1 Receiver

For reliable connection, the receiver must be able to receive a weak signal in hostile environments. In full-duplex systems, the requirements are even harder as the inter-

¹See Section 2.2.2.2

²Refer to 2.4.2 for more details on mixers

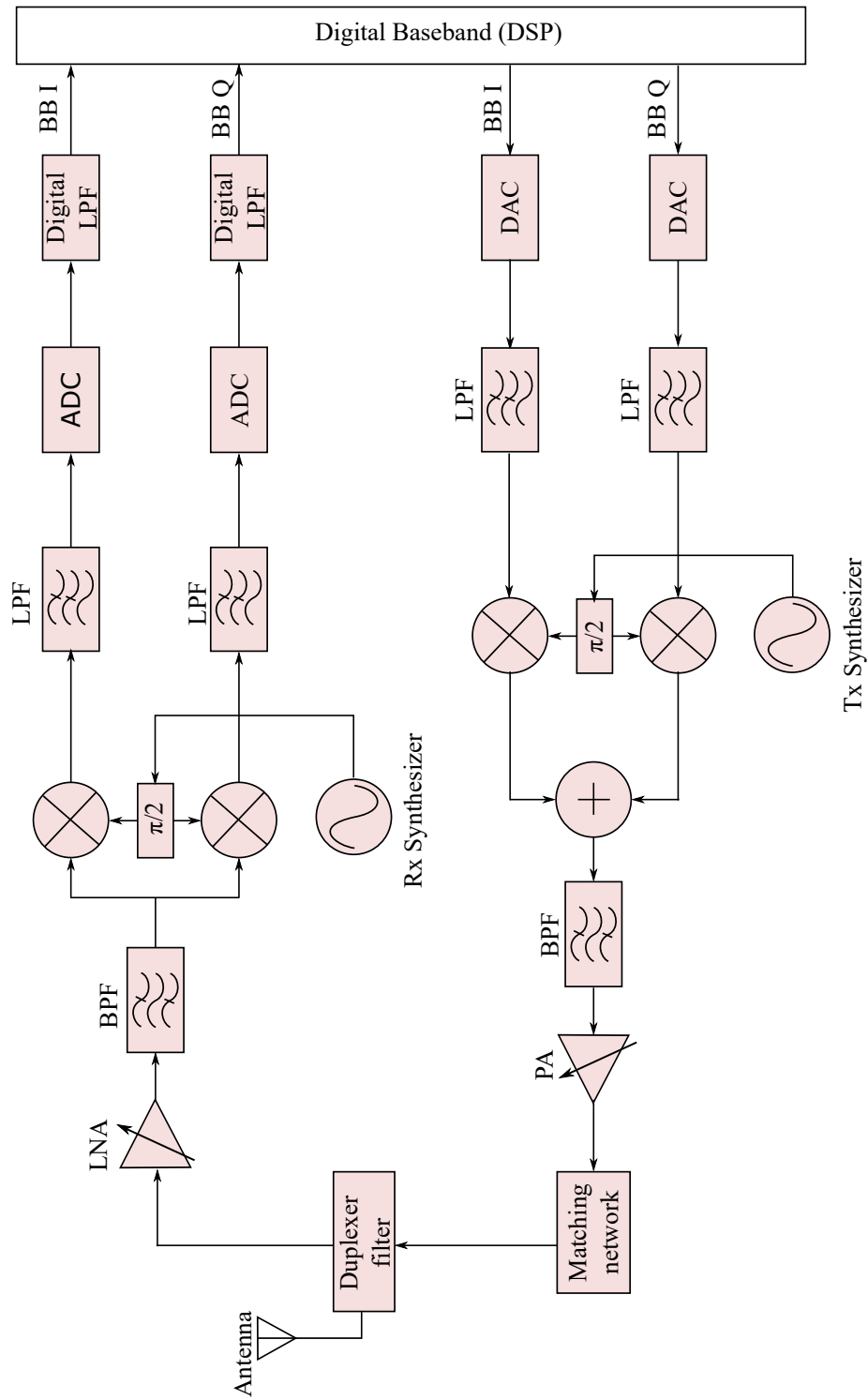


Figure 2.6: Direct conversion transceiver.

2 Basics of Radio-Frequency ICs

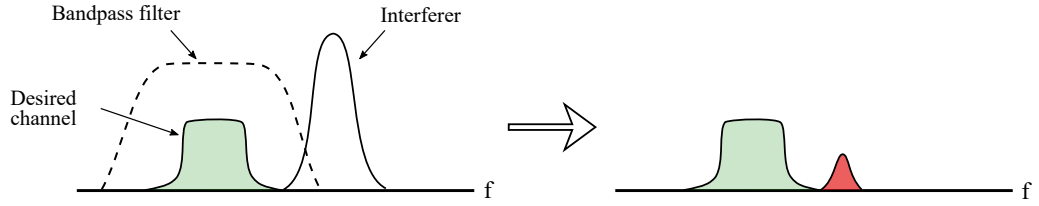


Figure 2.7: Adjacent channel interference.

ference comes from the directly neighboring transmitter operating in parallel. The key parameters defining the performance of a receiver are given below.

Reception sensitivity is the weakest signal power required to detect with SNR lower than a specified maximum value SNR_{MAX} . Excessive noise makes the signal indecipherable. The noise factor from the antenna to the input of the ADC is given as the ratio between input and output SNR, i.e. $NF = SNR_i/SNR_o = P_{S,i}/P_{S,o}/SNR_o$, where $P_{S,i}$ and $P_{S,o}$ are the input signal and noise power, accordingly. For input white noise, the noise power is given by $P_{N,i} = kT$ and it is distributed uniformly within the channel bandwidth BW . It follows that the minimum input power to achieve the required SNR_{min} is

$$P_{min} = kT \cdot BW \cdot NF \cdot SNR_{min}. \quad (2.4)$$

The sensitivity, expressed in dBm, is then $S_{min} = 10\log_{10}(P_{min})$.

Adjacent channel selectivity (ACS) is a measure of the receiver's ability to detect a signal at a given frequency in the presence of a large interferer in an adjacent channel. ACS is defined as the ratio in dB of the filter attenuation on the assigned channel to the filter attenuation on the interferer channel. Fig. 2.7 visualizes the situation where a near-by interferer cannot be filtered out as it is too close to the assigned channel and (partially) lies in the passband of the receiver's filter. As in a system test the filter attenuation cannot be measured directly, the test is instead defined to check the throughput in the presence of an interferer in the adjacent channel.

Interference blocking is similar to ACS, but measures the receiver detection when the interferer is at frequencies other than the adjacent channels.

Intermodulation spurious attenuation is a measure of the receiver that characterizes its linearity. Intermodulation is an effect that arises when two or more signals with different frequencies are applied to a nonlinear system, such as a mixer. The output of a memory-less, time-variant system is given by the Taylor series expansion

$$y_0 = \sum_{k=0}^{\infty} a_k x^k, \quad (2.5)$$

where x and y_0 are the input and output signals of the system, and a_k are the coefficients of different orders of non-linearity. When the input x is the sum of two strong sinusoidal signals $V_1 \cos(2\pi f_1 t)$ and $V_2 \cos(2\pi f_2 t)$, and this sum is raised to a power greater than one, intermodulation occurs. Using trigonometric identities, the second order products

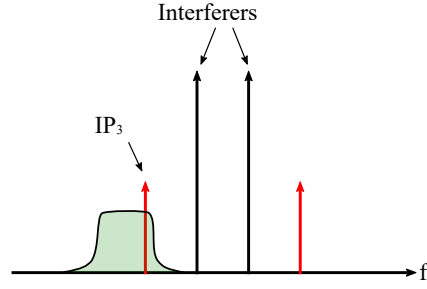


Figure 2.8: Intermodulation between two signals.

will be at frequencies $f = f_1 \pm f_2$ – far away from the assigned channel frequency and can be filtered out. The third-order products will produce frequencies $f = 2f_1 \pm f_2$ and $f = 2f_2 \pm f_1$, which is close to the original signals that created them. The third order intermodulation product can then fall into the assigned channel, corrupting the detection, as shown in Fig. 2.8. The higher order products are usually not considered, as they have insignificant signal level.

2.2.2.2 Transmitter

The transmitter by definition produces electromagnetic emissions and therefore it must comply strictly not only with the communications standards, but also with the regulatory directives of, for example, EC and FCC. The most important performance measures for transmitters are outlined below.

Maximum output power defines the total integrated power over the channel bandwidth. It effects the cell capacity and size of the cellular network and also the power consumption of the UE.

Modulation accuracy is a key parameter of every transmitter. It is represented by the error vector, which denotes the difference between the actual symbol location and the theoretical ideal symbol location. The magnitude of this difference is called error vector magnitude (EVM). Fig. 2.9 shows a 4-QAM constellation diagram in the presence of transmitter non-idealities. At each sample the non-idealities will display the constellation points randomly around the ideal position, resulting in a scattered constellation. The error vector S_{err} denotes the difference between the ideal vector S_i and the measured vector S_r . The EVM is the magnitude of the error vector normalized by the ideal symbol vector [KM04]

$$EVM = \frac{|\vec{S}_{err}|}{|\vec{S}_i|} = \frac{|\vec{S}_i - \vec{S}_r|}{|\vec{S}_i|}. \quad (2.6)$$

Out of band emissions define spectral emissions immediately outside of the assigned channel. The maximal levels are defined in terms of adjacent channel leakage ratio (ACLR) and a spectral emission mask. ACLR is the ratio between the power in the adjacent channel and the power of the assigned channel.

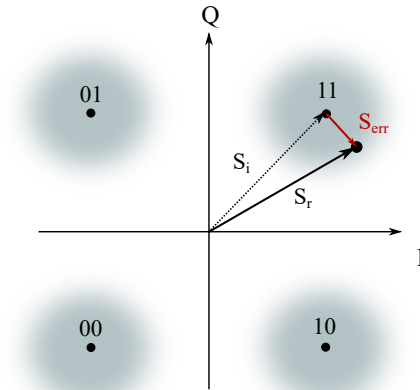


Figure 2.9: Distorted 4-QAM and Error Vector Magnitude.

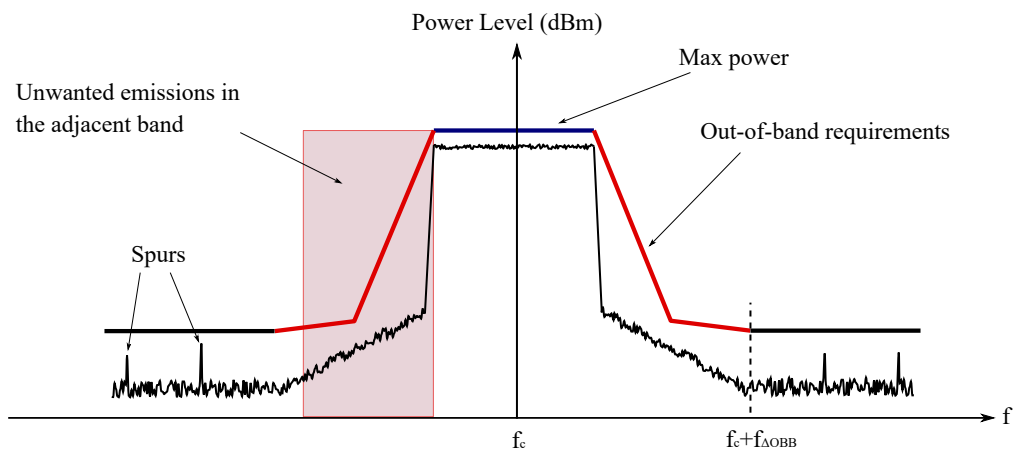


Figure 2.10: Spectral mask.

Spurious emissions refer to any emissions in the frequency range further than $f_{\Delta OOB}$ from the carrier. These emissions are caused by unwanted effects in the transmitter chain, such as intermodulation and frequency conversion products.

A *spectral mask* is a set of lines that delineates the transmitter output spectrum in the assigned channel, in the adjacent channels and the far-out (spurious) region. The transmitted signal should always be below the mask, or else a mask violation, and respectively a violation of the standard, occurs. Fig. 2.10 shows an exemplary spectral mask and the spectrum of an example signal.

2.3 Noise

Noise is defined as random, unwanted disturbance that is not related to the signal. It exists in every system, and it is one of the most important factors determining the

performance of communications systems. Random signals are usually described in terms of power spectral density (PSD), which shows the distribution of the average signal power of the variations as a function of the frequency, per unit frequency.

Usually for convenience it is assumed that the signal drives a $1\ \Omega$ load, and then the PSD has unit of volts²/Herz.

In a CMOS process, the noise produced in the MOSFET devices is the most prominent contributor.

2.3.1 Noise Sources in MOSFET Transistors

The two major noise sources in a transistor are thermal and flicker noise. In normal operation, there is a resistive channel between drain and source of the MOSFET. This channel's resistance intrinsically produces thermal (or Johnson–Nyquist) noise. The PSD of thermal noise of a resistor with resistance R is given by

$$S(f) = 2kTR, \quad (2.7)$$

where $k = 1.38 \cdot 10^{-23} \frac{J}{K}$ is the Boltzmann constant, and T is the absolute temperature. Theoretically, the spectrum of the white noise is flat, so it follows that its power should be infinite. However, in practice it starts dropping for $f > 100$ GHz, such that the power has a finite value [Cou00].

Flicker noise is low-frequency noise and has been observed in all electronic devices. The flicker noise PSD in a transistor is given by

$$S(f) = \frac{I_D^{A_f} K_f}{C_g f}, \quad (2.8)$$

where K_f and A_f are empirical coefficients, I_D is the drain current and C_g is the gate capacitance of the transistor. A more detailed analysis of noise in MOSFETs can be found for example in [CS91].

2.3.2 Phase Noise

An ideal oscillator produces an ideal sine wave

$$V(t) = A_0 \sin(\omega_0 t), \quad (2.9)$$

where ω_0 is the nominal frequency and A_0 is the nominal amplitude. The argument of the sine function is the phase of the signal and it is related to the frequency – frequency is the derivative of phase, and phase is the integral of frequency:

$$\omega = \frac{d\phi}{dt} \quad ; \quad \phi = \int_0^t \omega dt. \quad (2.10)$$

It follows that changes in frequency correspond to changes in phase and vice versa.

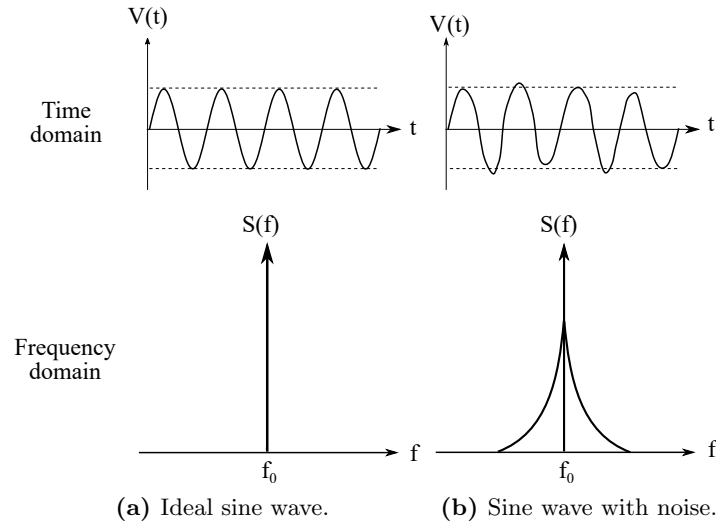


Figure 2.11: Sine wave in time and frequency domain.

An ideal sine wave has its whole energy concentrated at a single frequency, and therefore it has infinitesimally narrow bandwidth. In frequency domain the ideal sine is represented as a Dirac delta function at frequency ω_0 (Fig. 2.11a).

A real world oscillator cannot produce an ideal signal; it will always have some unwanted small, random changes in amplitude and phase. The produced signal is then expressed as

$$V(t) = (A_0 + E(t))\sin(\omega_0 t + \phi(t)), \quad (2.11)$$

where $E(t)$ represents the random fluctuations in amplitude and $\phi(t)$ the random fluctuations in phase.

The source of these random fluctuations are flicker noise and white noise at the input of the oscillator, which are upconverted to the oscillator frequency and shaped by $\frac{1}{f^2}$ under the effect of closed-loop operation of the oscillator, producing $\frac{1}{f^3}$ and $\frac{1}{f^2}$ regions in the output noise spectrum [HL98a, Raz96c]. The power at the offset frequencies are called 'skirts' because of the way they look in frequency domain. Figure 2.11 shows time and frequency domain representation of an ideal sine wave and a real world sine wave with additive noise.

For stationary noise sources, the amplitude and phase components of noise are equal. However, if we pass the signal through a limiter, the resulting waveform will have trapezoidal shape. CMOS gates can serve as a limiter: due to the saturation of the output of the gate, the amplitude fluctuations are removed, leaving only phase fluctuations in the waveform. These random fluctuations in the phase of the signal are referred to as *phase noise*. A measure for phase noise is single side band power $\mathcal{L}(f)$, pronounced "script L of F" $\mathcal{L}(f)[dBc] = Pn[dBm/Hz] - Ps[dBm]$

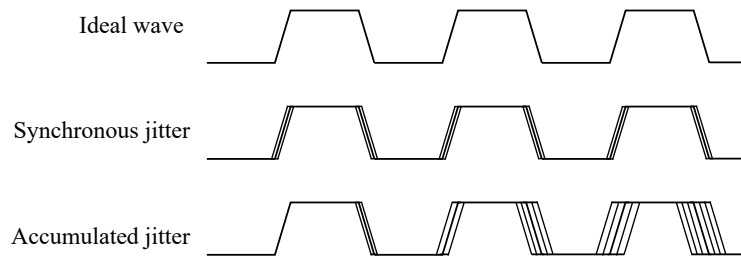


Figure 2.12: Synchronous and accumulating jitter in time domain.

The term phase noise is used when the representation is in frequency domain, whereas *jitter* is the term used when talking about the deviations in time domain. For ideal periodic signals the zero crossings are spaced in equal intervals of T_0 . In the presence of noise, the crossings are not spaced equally.

There are different ways to measure jitter, depending on the type of circuit under analysis. *Synchronous jitter* or *absolute jitter* is the deviation of the edge position from an ideal clock. This type of jitter is defined for driven circuits and is the deviation of the delay between the input clock and the produced output. *Accumulating jitter* (or *period jitter*) is the standard deviation of the *period* of the jitter. Accumulating jitter is usually used to characterize autonomous systems such as oscillators. Fig. 2.12 shows signal waveforms with synchronous and accumulating jitter. It can be observed that the phase of the accumulating jitter drifts without bound.

2.3.2.1 Phase Noise in CMOS Circuits

CMOS gates are thresholding circuits that switch their state when the voltage at the input crosses a specified threshold level. A large periodic input signal at the input of a logic gate creates a process with time-varying statistics, referred to as a *cyclostationary process*. The switching modulates the existing noise, and as a consequence the noise folds across the fundamental frequency of the modulating signal [PK00]. Due to their thresholding nature, CMOS logic gates are only sensitive to noise present at the point where the input crosses the switching threshold, ignoring any noise present at other times.

Phase noise in inverters Analyzing the noise in inverters – the most basic logic element in CMOS circuits – provides good insight into the mechanisms of noise generation in CMOS circuits in general. The noise generation in more complex logic gates follows the same pattern. The following simplifying assumptions are made:

- 1) an inverter produces an ideal voltage ramp at its output in a response to an input transition;
- 2) the phase noise manifests itself while V_{out} crosses $V_{DD}/2$; and

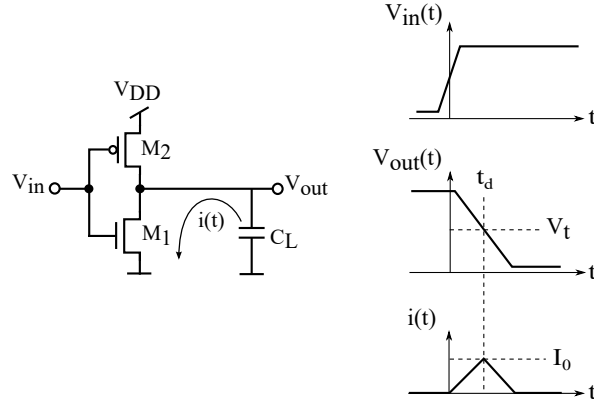


Figure 2.13: A CMOS inverter and its transient response.

3) V_{out} crosses the threshold value only once.

On a rising edge of the input signal, the pMOS transistor switches off and the nMOS transistor switches on just after the threshold crossing. The charge stored in the load capacitor C_L starts discharging through the nMOS transistor. Simultaneously, the noisy transistor's channel generates noise current that adds to the discharging current from C_L .

The noise current generated in the transistor integrates on the load capacitor C_L during the output transition and creates noise voltage v_n . The corresponding power spectral density (PSD) can be expressed as [Abi06]

$$S_{v_n}(f) = \frac{1}{C_L^2} |W_{t_d}(f)|^2 S_{i_n}(f). \quad (2.12)$$

$W_{t_d}(f)$ is the Laplace transform of the integration window with rectangular form and its magnitude is

$$|W_{t_d}(f)| = t_d \frac{\sin(\pi f t_d)}{\pi f t_d} \quad (2.13)$$

where t_d is the time needed for the output signal to reach the threshold point.

The noise voltage perturbs the position of the threshold point, causing time displacement referred to as jitter. The slew rate $SR = dv_{out}/dt$ is the rate of voltage change per unit of time and it gives the relation between the noise voltage and jitter. Fig. 2.14 shows how voltage noise converts into time displacement. For an ideal voltage ramp $SR = V_{DD}/t_t$, where t_t is transition time. The jitter spectral density is expressed as

$$S_\sigma(f) = \frac{S_{v_n}(f)}{SR^2} \quad (2.14)$$

and it is folded around dc by the modulating periodic signal f_0 . When one edge of a square wave is displaced by ΔT , the corresponding phase change $\Delta\phi = \pi f_0 \Delta T$ and the phase noise PSD is

$$S_\phi(f) = \pi^2 f_0^2 S_\sigma(f) = \frac{\pi^2 f_0^2}{SR^2} S_{v_n}(f). \quad (2.15)$$

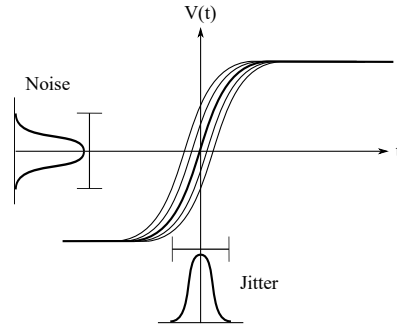


Figure 2.14: Voltage noise and jitter.

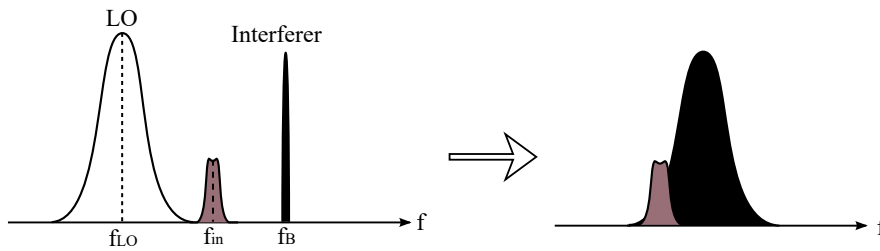


Figure 2.15: Reciprocal mixing.

Phase noise in a chain of CMOS gates Each gate in a circuit with chained logic gates generates noise. The noise produced by these gates is not correlated, hence the corresponding jitter PSDs sum as

$$S_{\sigma}(f) = \sum_{i=1}^N S_{\sigma,i}(f), \quad (2.16)$$

where $S_{\sigma,i}$ is the phase noise generated by stage i . The phase noise sums up in the same manner.

2.3.3 Effects of Phase Noise on Communications Systems

Phase noise in the local oscillator (LO) signal degrades the performance of the transceivers. Therefore, in the conceptual phases of transceiver's design, normally a maximum value of the phase noise of the LO signal, reaching the mixer – so called noise budget – is defined. The definition is usually given for a few offset frequencies, covering in-band, out-of-band and far-off regions of the spectrum.

In the receive chain, phase noise in the LO causes an effect called *reciprocal mixing* that raises the noise floor, leads to an effect called *desensitization*, i.e. reduction of the SNR. The convolution of the noisy LO signal with the interferer results in broadened interferer. The skirts of the downconverted interferer signal then corrupt the desired signal, as shown in Fig. 2.15. This reduces the reception sensitivity and the adjacent channel selectivity of the receiver.

In transmitters, phase noise increases the EVM, as EVM is a function of the integrated phase noise magnitude. Additionally, phase noise can lead to violations of the spectral mask.

2.4 Selected Transceiver Building Blocks

This section presents a more detailed view of the operation principle of the transceiver building blocks and the effect of noise on them. Some of the building blocks that operate entirely in analog fashion, such as power amplifiers, analog filters and power supplies, are out of the scope of this work and are therefore omitted.

2.4.1 Frequency Synthesizer

A frequency synthesizer is a device that takes a very stable input frequency as a reference input and generates a range of frequencies as dictated by a control input. The stability and accuracy of the produced frequency is a function of the stability and accuracy of the reference frequency. There are different techniques for frequency synthesis: direct analog synthesis, direct digital synthesis (DDS) and indirect synthesis based on phase locking.

Direct analog synthesizer, also called mix-filter-divide architecture, provides a high quality output signal. A high number of crystal oscillators are used to create various frequencies. By combination of dividing, mixing and filtering, a range of new frequencies can be derived from the already existing. This approach is not well suited for integration due to the high number of components required.

Direct digital synthesizers are widely used. They contain memory and logic to construct an output signal digitally. The signal is then passed through a digital-to-analog converter (DAC) and an analog filter to create the desired analog signal. A gigahertz capable DAC is hard to build and consumes high operating power and for that reason DDS is not used for mobile applications.

Indirect synthesis methods use an oscillator, which is controlled by another signal, such that its output is in a given relation to the input signal. The control circuit monitors the phase of the output and compares it to the input phase, giving this type of circuits the name *phase-locked loop (PLL)*. Most modern integrated transceivers use PLL for the frequency synthesis. A more detailed description of the principle of operation of PLLs is given in the following section.

2.4.1.1 Phase Locked Loops

The basic PLL consists of a phase detector (PD), a low pass filter (LPF) and a voltage controlled oscillator (VCO) (Fig. 2.16).

The VCO oscillates with frequency

$$f(t) = f_{fr} + K_{VCO} \cdot u_f(t), \quad (2.17)$$

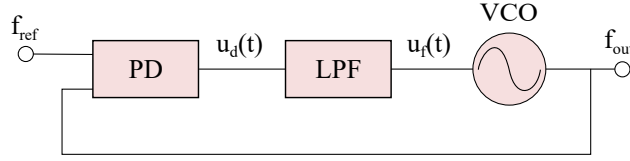


Figure 2.16: PLL block diagram.

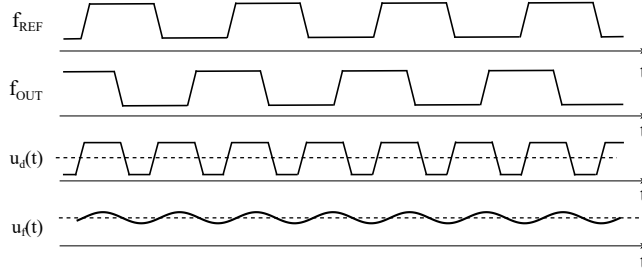


Figure 2.17: PLL waveforms.

where f_{fr} is the free running frequency of the VCO, K_{VCO} is its gain and $u_f(t)$ is the control voltage at the output of the LPF.

The phase detector compares the phases of the output f_{out} and the reference signal f_{ref} and generates the error signal $u_d(t)$ proportional to the phase difference

$$u_d(t) = K_d \cdot \theta_e. \quad (2.18)$$

Here, K_d is the gain of the PD in volts, and θ_e is the phase error in radians. The output signal u_d has both dc and ac components. The fast ac component is not desired, therefore the LPF extracts only slowly varying components out of the signal. The resulting signal $u_f(t)$, being the control signal for the VCO, contains the information about the phase difference [Bes07]. The waveforms of the discussed signals are shown in Fig. 2.17 [Raz96b].

In locked state, i.e. the synchronous state, the circuit operates as follows:

If the input and the reference frequencies are equal, the phase error θ_e is zero. The signals u_d and u_f are also 0, therefore the VCO is operating on its currently locked frequency. If a change in the reference frequency occurs, this will result in non-zero phase error θ_e , which will lead to deviation from the free running condition. The VCO will change its output accordingly up or down to reduce the error, so the output frequency always follows the reference frequency; more precisely, it follows the phase.

If the VCO output is different from its free running frequency, then the signal $u_f(t)$ should settle to a non-zero value, ruled by the equation $u_f = \Delta f / K_{VCO}$. If the input frequency is frequency modulated, then the output signal of the LPF is the demodulated signal. Hence, the PLL may serve as a frequency modulation (FM) detector [Bes07]. However, the main use case of PLLs is frequency synthesis. In this type of application, the PLL should be able to generate a set of frequencies. One way to achieve that is to

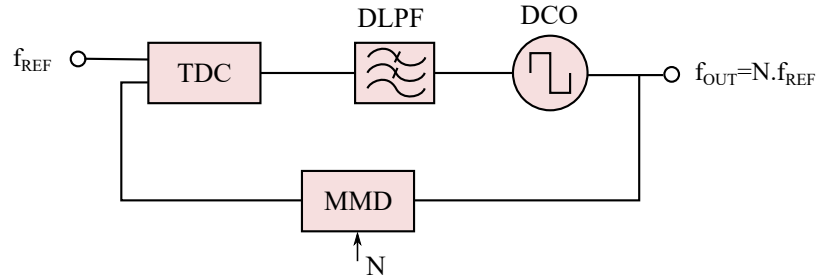


Figure 2.18: All-digital PLL block diagram.

place a divider in the feedback path of the PLL [Sep70]. The result is an increase of the output frequency by the division factor of the divider. If the divider ratio is adjustable with several divider values, i.e. if it is a MMD, then the output frequency of the PLL can also be changed based on the control word of the MMD.

When the PLL is initially turned on, it is not locked yet, and must acquire the lock. In this initial state, the VCO starts oscillating freely at f_{fr} , whereas the reference signal is on an offset of Δf . An important property of the PLL is the *acquisition range*, which is the maximum Δf within which the loop can still lock. Another performance indicator of the PLL is the acquisition time, i.e. the time for the loop to reach the locked state and thus generate the required output frequency. Details about the acquisition behavior and the factors that affect acquisition time can be found in any good book on PLLs, for example in [Bes07, Tal12].

All-Digital PLLs As deep submicron CMOS technologies scale down and the circuits have very limited voltage headroom, a new paradigm was observed: "*In a deep-submicron CMOS process, time-domain resolution of a digital signal edge transition is superior to voltage resolution of analog signals*" [Sta06]. A natural consequence of that observation was to look into ways of digitalizing the PLL, toward a so called all-digital phase-locked loop (ADPLL).

By using digital architecture, the ADPLL benefits from the ever increasing time-domain resolution and density of advanced CMOS processes. Digital design techniques also offer higher integration, portability, fast design cycles and high verification coverage (See Section 3.2). The most simple ADPLL block diagram can be constructed from the regular PLL block diagram, by replacing the building blocks with their digital counterparts – the VCO with a digitally controlled oscillator (DCO), the LPF with digital low-pass filter (DLPF) and the phase detector with a TDC. Fig. 2.18 shows the block diagram of an ADPLL, including a MMD in the feedback path for adjustable output frequency. In an ADPLL all intermediate signals are digital, except for the internal signals of the DCO. ADPLLs provide faster locking time and better portability and stability over technology processes [WSWW10, CL03]. Furthermore, analog filters require large

on-chip or external passive components, and using a digital LPF instead increases the integration. Therefore, ADPLLs are state-of-the-art in advanced integrated systems.

2.4.1.2 Digitally Controlled Oscillator

A digitally controlled oscillator creates an oscillating output signal, which frequency can be controlled by a digital word. For any circuit to be an oscillator, there is a positive feedback loop at a given frequency [LB09]. A necessary condition for an electronic circuit to oscillate is that it needs to meet the so called Barkhausen stability criteria: the loop gain must be equal to unity, and the phase shift around the loop must be equal to 0.

One commonly used oscillator topology is the ring oscillator. A *ring oscillator based DCO* has a chain of odd number of inverters in a feedback loop. Due to the odd number of inverters, the output of the last inverter is the logical NOT of the first stage. Every stage adds $180^\circ/N$ phase shift to the signal, and the inversion adds another 180° , thus fulfilling the Barkhausen criterion. The frequency of operation is $f_0 = 1/2Nt_d$, where t_d is the delay of each stage and N is the number of stages. The signal propagates around the chain and is inverted for each half period. Due to the finite delay of each inverter stage and the feedback from the output of the last stage to the first stage, the oscillation arises. The ring oscillator based DCOs can be tuned in different ways [VDTKVR03]. One of the most commonly used one is *current starving*, where the output current of the inverter is limited through a bias circuit.

Ring oscillators have wide tuning range, small area, low power and generate multi-phase signals that can be easily converted to in-phase and quadrature signals with $\pi/4$ phase difference [PKK00]. However, ring oscillators produce high phase noise which makes them unsuitable for many RF applications [Raz96c]. Therefore, another type of oscillators – *LC tank based oscillators* – are widely in use for modern RF applications.

An *LC oscillator* makes use of an inductive coil and a capacitor connected in parallel to create an LC resonant tank circuit. The resonant frequency of such a circuit is given by $f = \frac{1}{2\pi\sqrt{LC}}$. To compensate for the energy losses in the circuit, an amplifier is also required. Among the various possible LC DCO architectures, the one that is preferred for mobile transceivers is the *symmetric LC DCO*, due to the low power consumption, good symmetry and low phase noise [HL98b, HKF⁺06]. The frequency control can be achieved by changing either L or C, but due to the characteristics of the CMOS process it is easier to change the capacitance. As first proposed in [SHLB03], the capacitor can be replaced by a binary controllable varactor (variable capacitor) to tune the oscillation frequency digitally. Fig. 2.19 shows a block diagram of a symmetric LC DCO with a tunable capacitor. LC DCOs have superior phase noise performance, but the inductor and the varactor matrix take a large area on the chip. Additionally, they have a limited tuning range due to the limited size of the capacitive array, associated with significant parasitic capacitance added to the output node for each capacitor and increased phase noise [HRA05]. Due to the limited tuning range of the LC DCOs, a PLL that has to output a signal in a wide frequency range usually has multiple DCOs with various center

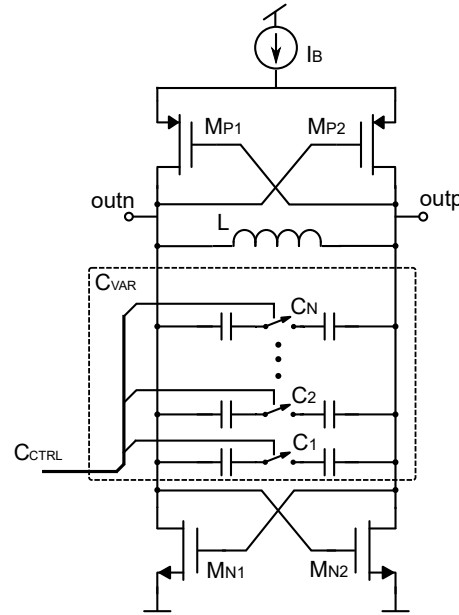


Figure 2.19: A symmetric LC DCO.

frequencies. During operation, based on the required frequency of operation, one of the DCOs is selected for the operation with the help of microwave switches or by keying DC supply voltage [Rok98].

2.4.1.3 Noise in PLLs

The TDC generates quantization noise, caused by the quantization error and it effects the system similarly to AWGN³. The DCO, as well as the reference oscillator produce noise around the carrier. These noise sources influence the output noise according to the transfer function to the output. The noise transfer function for the reference clock noise and the TDC noise corresponds to a low-pass filter with a cut-off defined by the loop bandwidth f_{LOOP} . The DCO noise is high-pass filtered under the closed-loop operation of the PLL.

In Fig. 2.20 the discussed noise sources are plotted as a function of the frequency offset, f_{offset} on a double logarithmic scale. The in-band noise close to the carrier f_0 is dominated by the reference noise. Further away from the carrier within the loop bandwidth f_{LOOP} , the noise is dominated by the TDC quantization noise. Outside the loop bandwidth there is a region dominated by the DCO noise which rolls off with $1/f^2$ slope, followed by the far-off region of white noise related sources. Further, in-depth analysis of noise in PLLs can be found in [Haj01].

³Refer to Section 4.2

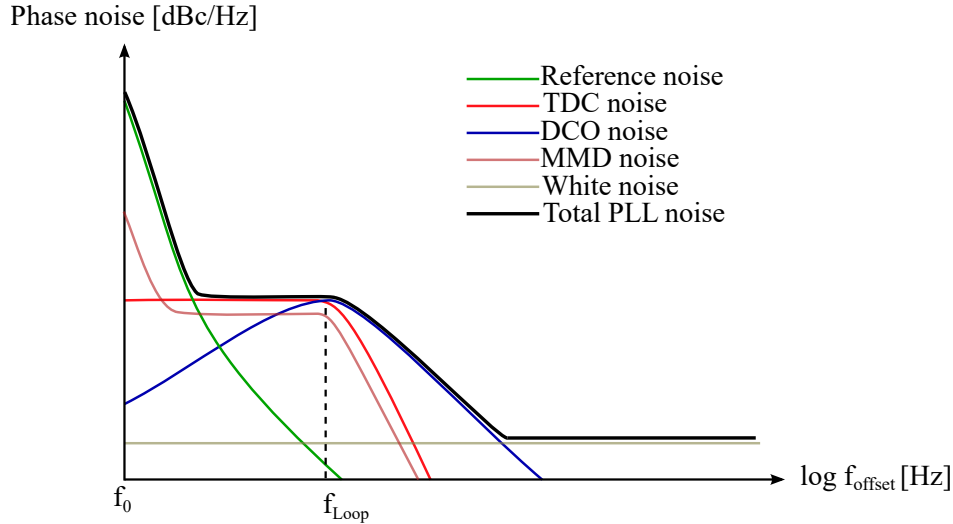


Figure 2.20: Noise in PLLs.

2.4.2 Mixer

A mixer is a device used to achieve frequency translation - up and down conversion. It has two input signals – in a transceiver these are the local oscillator clock and the incoming or outgoing signal for the receiver, or transmitter accordingly.

One possible mixer implementation is a *nonlinear mixer* that uses a device with a non-linear behavior, such as a diode or transistor. The intermodulation between the two input signals creates the desired output, but also many spurious signals.

Another mixer implementation is a *switching mixer*. It contains a switch that turns on and off with the LO frequency, which in this case is a square signal. Because of the symmetry, the even numbered harmonics drop out of the signal spectrum. This method is preferred as it has better linearity and spur performance [Lee03]. For good noise performance, the switching should be as fast as possible, therefore square waves are preferred over sines. If the LO signal has finite rise and fall, the CMOS switch will operate as amplifier in its linear region during LO transitions, acting like an amplifier to the LO noise as well as its own noise, degrading the overall mixer noise figure. The LO signal is passed through a buffer that acts as a limiter in the mixer and this waives the effect of amplitude variations in the LO signal. However, the system stays susceptible to phase variations in the LO.

A simplified schematic of a switching mixer is shown in Fig. 2.21a. The baseband signal V_{BB} is multiplied with the LO signal V_{LO} , resulting in frequency translation, as shown in Fig. 2.21b. As there are multiple copies of the original baseband signal, a filter is required to remove the unwanted frequency components.

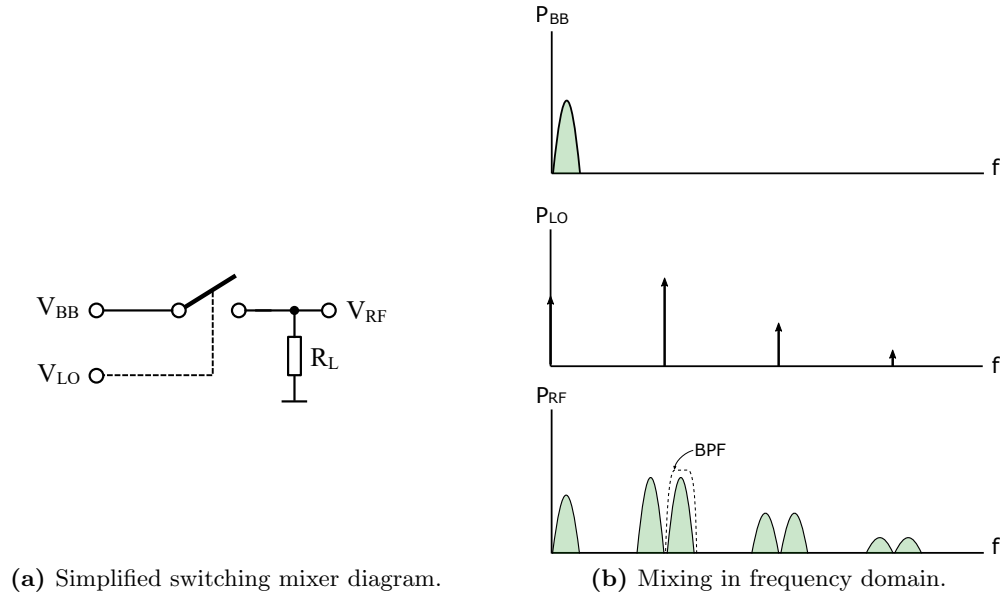


Figure 2.21: Switching mixer.

2.4.3 RF Clock Distribution

As discussed in Section 2.4.1.2, DCOs usually have a limited tuning range. As result, multiple DCOs are required in the frequency synthesizer to cover the wide range of supported frequencies. Taking power consumption and crosstalk mitigation into consideration, of course only one DCO per subsystem (Rx, Tx) is active, delivering the required local oscillator signal. The output of the active DCO is multiplexed to the downstream modules.

To meet the high-bandwidth requirements of high-end devices, mobile transceivers need to support techniques that improve the bandwidth and spectral efficiency of the system, such as carrier aggregation (CA) and Multiple Input Multiple Output (MIMO). CA increases the bandwidth by combining two or more channels into one virtual larger channel. The channels can be within one band (intra-band) or in different bands (inter-band). While in the former case a single synthesizer is sufficient [HR15] [CH15], the later requires separate synthesizer chains. MIMO is a technique of using multiple transmitters and receivers (e.g. 2x2 or 4x4) that exploits the multiple paths between transmitting and receiving antennas to improve the channel capacity [Moh12].

Both (intra-band) CA and MIMO imply multiple transmitters and receivers on the same die [ETS]. This increases the die size and requires distribution of the clock signals over large distances – up to tens of millimeters – on the chip. Additionally, depending on the mode of operation, some of the transmitter/receiver chains might not be active, so the corresponding clock signal should be gated for saving power. The above-mentioned features require additional circuitry added into the clock distribution path, including

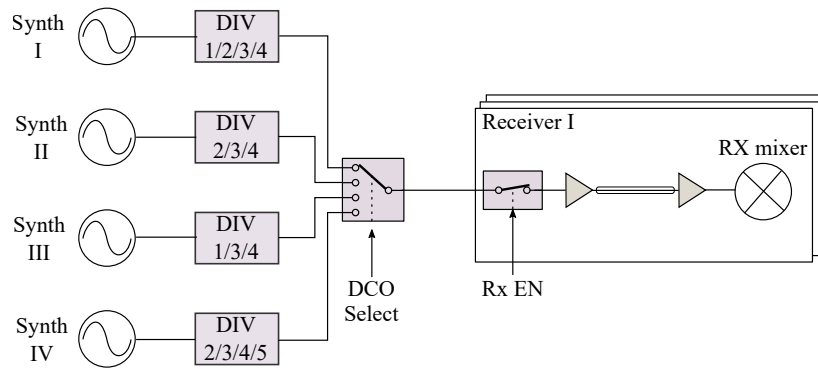


Figure 2.22: A generic RF clock distribution network.

multiplexers, programmable clock dividers, buffers, and clock gating. A generic clock distribution network, containing multiple frequency synthesizers, a multiplexer, dividers, gating logic, repeaters, and multiple endpoints (mixers) is illustrated in Fig. 2.22.

Each gate added to a signal path introduces noise into the system, as discussed in Section 2.3.2. Therefore, design of power efficient clock distribution network that meets the phase noise requirements is a challenging task.

3 ASIC and Full-Custom Design Flows

This chapter presents and compares the ASIC and custom design flows. First, Section 3.1 presents the full-custom design flow. Then, the ASIC design flow and key concepts and terminology of the static time analysis are explained in Section 3.2. Finally, the shift towards using the ASIC approach for designing radio-frequency ICs is motivated in Section 3.3.

3.1 Full-Custom Design

The full-custom design flow is in essence manual analog design, assisted by EDA tools. The steps of the flow are explained in the following paragraphs. Fig. 3.1 offers a graphical representation of the discussed flow.

Input for the flow are the design specifications that define the functionality and performance parameters for the given block, such as maximum power, area, linearity etc. Based on these specifications, the circuit designer selects a circuit implementation and manually draws a schematic within a schematic editor by placing and wiring transistors of selected size. The functionality of the designed circuits is then verified using an analog simulator. A test bench that provides input stimuli and output load (if needed) is created around the device under test (DUT). The goal of the simulation is to verify that the circuit is performing the intended function and is meeting the performance requirements. If the circuit does not meet the criteria, parameters such as the transistor sizes can be altered or another topology can be explored until the requirements are satisfied. Hence, circuit changes and simulations are executed iteratively to select the best implementation.

The completed schematic is sent to a dedicated layouter for drawing the layout. Due to the shrinking technology nodes, nowadays the number of design rules for the layout is overwhelmingly high and the layout production becomes one of the most time consuming tasks in the design flow. To achieve good circuit performance, many requirements should be considered, e.g. matching of transistors, symmetry, skew requirements etc. The more precise the constraints for the layout are, the better the results. Good results require deep understanding of the circuit functionality, as well as experience in circuit design and layout. However, the layouter is usually not the circuit designer and might not be aware of the specifics of the design, so clear communication of the circuit requirements is essential to avoid a possible performance degradation of the circuit.

After the layout is done, the parasitic effects are extracted from the layout and the results are written back to the schematic in a process referred to as back-annotation. The

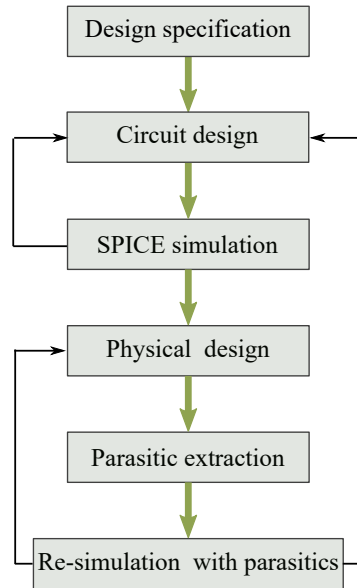


Figure 3.1: Custom design flow.

back-annotated design is then re-simulated to verify its functionality and performance, while taking into account the parasitic effects.

The complexity of most of the designs does not allow to simulate for every possible input combination. Therefore, only the critical patterns and transitions can be considered by the designer during simulation, leaving possible errors undiscovered.

The process variations in the manufacturing process of CMOS logic lead to parameter variations among the chips on a wafer. In addition, supply voltage and operating temperature affect the operation speed of the CMOS devices. Therefore, besides the verification under nominal conditions, the correct functionality should also be guaranteed under process, voltage and temperature (PVT) variations. The extremes of the parameter variations, within which the chip is still functional, are referred to as *corners*. There are multiple corners that should be considered – as example, the *slow corner* is the set of PVT parameters that leads to the slowest operation of the CMOS logic, whereas the *fast corner* has a set of PVT, leading to the fastest logic.

The analog simulation can account for corner variations by running the simulation in all possible corner PVT scenarios, but it cannot handle mismatches. For this purpose statistical Monte-Carlo simulations, requiring high number of runs for reasonable coverage, are performed. Hence, the simulation can be very time consuming and span multiple days.

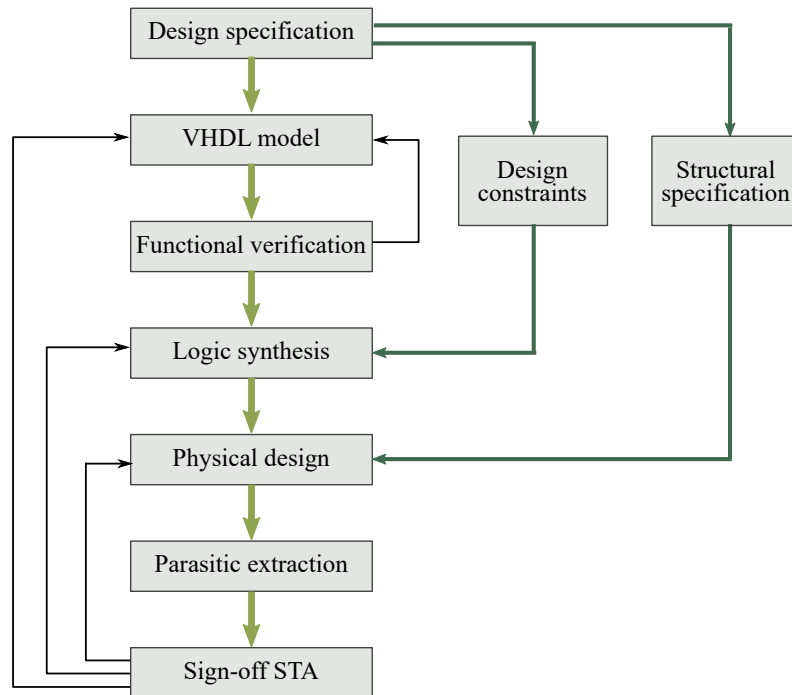


Figure 3.2: ASIC design flow.

3.2 ASIC Design Flow

Transistor-level design implementation for complex digital circuits is infeasible due to the high complexity of modern designs. Therefore, a level of abstraction is added by defining logic gates – small transistor circuits that perform a relatively simple logic function, such as NOT, NAND, have predefined layout and known properties, and are referred to as *standard cells*. A collection of standard cells is stored in a standard-cell library [BC09]. Instead of analog simulation, an alternative method of timing analysis of the circuit, based on lookup tables, is used (See Section 3.2.1). A diagram that visually illustrates the ASIC design flow is shown in Fig. 3.2.

Based on the design specification, the functionality of the design is described with the aid of a hardware description language (HDL), such as VHDL or Verilog. The functional correctness of the code is verified with a digital simulation. The quality of the verification is evaluated by its *coverage* that is the measure of what fraction of the design tested by the verification. Full test coverage checking all possible input combinations is not possible due to the prohibitive runtimes of the simulation. Therefore, a subset of the possible combinations can be tested by generating a large amount of random test vectors [AGL⁺95]. Advanced methods for increasing the quality of results exist, such as hole analysis [LMUZ02] that checks for uncovered spaces by the randomly generated test set. While these methods cannot guarantee 100% test coverage, they provide much higher test coverage than analog simulations. An alternative technique that received

increasing attention in the recent years in the so called *formal validation* that intends to remove the limitations of the functional verification. The main idea is to provide a form of mathematical proof that the design behavior holds under all conditions [Slo01]. This corresponds to 100 % coverage and is therefore especially appealing and subject of active research, however up to date it cannot completely replace simulation based verification.

After being verified, the HDL description is translated into a netlist, using standard cells in a step known as *logic synthesis* or just *synthesis*. During the translation, not only the functionality, but also the design constraints are taken into consideration. There are two types of design constraints – design rule constraints and optimization constraints. Design rule constraints are technology dependent constraints, required for the correct operation of the circuitry. They are provided by the semiconductor foundry and are hard-coded, and cannot be changed. The user is allowed to apply more strict requirements, but cannot loosen them. The design rule constraints are related to the

- Maximum and minimum capacitance of nets
- Maximum transition time of signals
- Maximum fanout of gates

Optimization constraints are set by the designer and represent the performance parameters for the design, such as area, timing and power. These constraints are based on the design specification and are specific for each design.

After synthesis, the physical realization of the design is created. The standard cells are placed on specific locations on the die, and the routing tool realizes the connections between them. More details about this step are discussed in Section 3.2.2.

Modern synthesis tools have built-in logic equivalence check, that verify the same functionality between HDL code and gate-level netlist or between two netlists. These checks are invoked after the logic synthesis and physical design steps to ensure that no erroneous change in the verified logic functionality has been introduced throughout the flow.

After the physical design is completed, the interconnect parasitics are extracted similarly to the full-custom flow. Then, the back-annotated netlist is verified for timing compliance in a *sign-off static timing analysis (STA)* run. The term "sign-off" signifies that this is the last verification check before taping out the product.

3.2.1 Basic Principles of Static Time Analysis

Static timing analysis (STA) is a gate-level analysis method for logic circuits. It verifies the timing performance of the circuit, i.e. the arrival of each signal at the signal endpoints within the predefined limits. It is, however, not verifying the functional correctness of the circuit [Xiu08]. STA abstracts the standard cells and the entire circuit as a set of timing graphs. For that purpose, each standard cell has timing models that define the delay through the cell. The delay is defined with the aid of a timing arc, which models

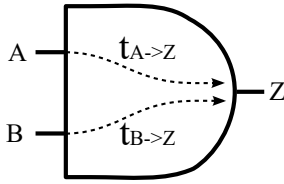


Figure 3.3: Combinational timing arcs.

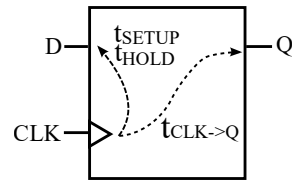


Figure 3.4: Sequential cell timing arcs.

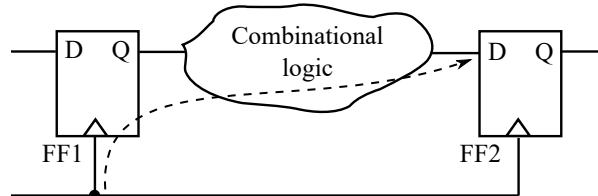


Figure 3.5: Timing path example.

the timing between two pins in a standard cell. Fig. 3.3 shows the existing timing arcs in an exemplary combinatorial AND standard cell. In this case there are two timing arcs, between the two input pins A and B, and the single output pin Z. They define the delay of the signal that propagates on that path. Sequential cells, additionally to the input to output arcs, also have setup and hold check arcs between clock and data pins (Fig. 3.4). Setup and hold checks define the time window during which the data signal should be stable before and after the clock edge, so that the data can be unambiguously captured by the sequential cell. If the setup or hold time requirement is not met, the register is a subject of *metastability*, i.e. the output settles down to a stable state very slowly or even oscillates before settling. Therefore, the occurrence of metastability should be avoided to guarantee the proper operation of the circuit.

A *timing path* is each path in the design that has a starting point and end point. A starting point is a point where data is launched by a clock edge, and at the endpoints data is captured by a clock edge. Fig. 3.5 shows an example of a timing path: here the data is launched by the register FF1 and captured by FF2. Depending on the complexity of the combinational logic, more than one timing path can exist between two flip-flops, potentially with different logic depth. A timing path is the sum of all timing arcs within standard cells, lying between the start and endpoints.

At the first clock edge (the launching edge), the data stored in FF1 is launched and the new value is available at the output FF1/Q after certain time, called the *clock-to-Q delay* ($t_{CLK \rightarrow Q}$). The signal then propagates through the combinational logic, and after a delay reaches the data pin of FF2 (FF2/D). The moment of arrival of the data at the endpoint is referred to as *arrival time*. The data reaching each register should be available before the required time, which is the capture edge time minus the setup time t_{setup} , and should be stable for certain hold time t_{hold} after the capture edge.

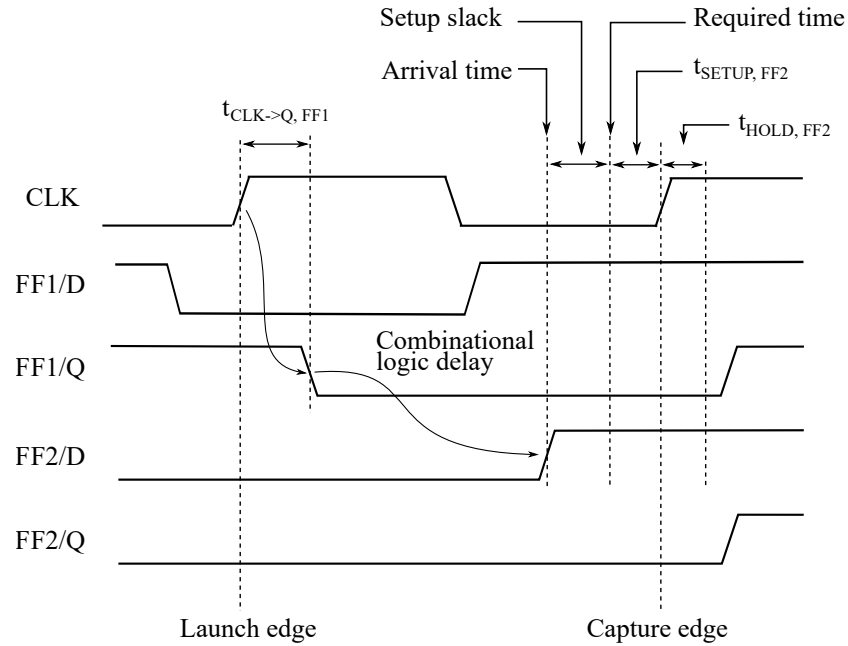


Figure 3.6: Timing diagram for the exemplary circuit of Fig. 3.5.

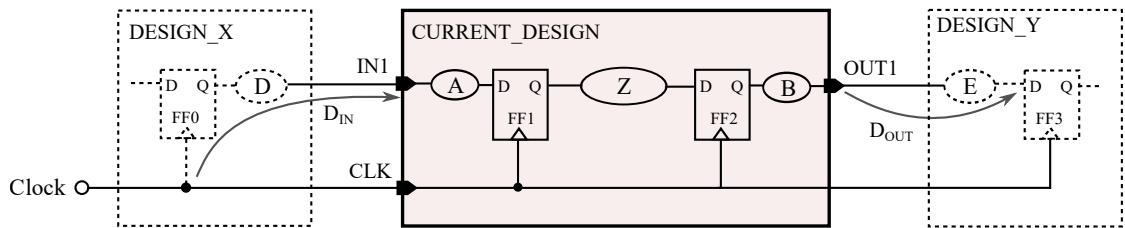


Figure 3.7: Input/output ports constraining.

A design can have externally provided clocks and internally derived clocks, so called *generated clocks*. The generated clocks are clocks that are in a fixed relation to their source clock, for example when the source clock is divided by 2. The definitions of clocks and generated clocks are central topic in STA, as the required times for each timing path are derived from the clock period.

During the design of a given block, the timing paths on the input and output ports of the design are not entirely defined. The drawing in Fig. 3.7 illustrates this.

All the timing paths in the combinational logic block 'Z' between the flip-flops within the design are known and the STA timing engine can calculate them. However, the timing of the external designs X and Y are unknown. Therefore, the ports should be constrained in a manner that provides the required knowledge to calculate these timing paths. In this example, the designer should define the maximum and minimum delays of the paths D_{IN} and D_{OUT} , referred to as *input and output delays*, respectively. The

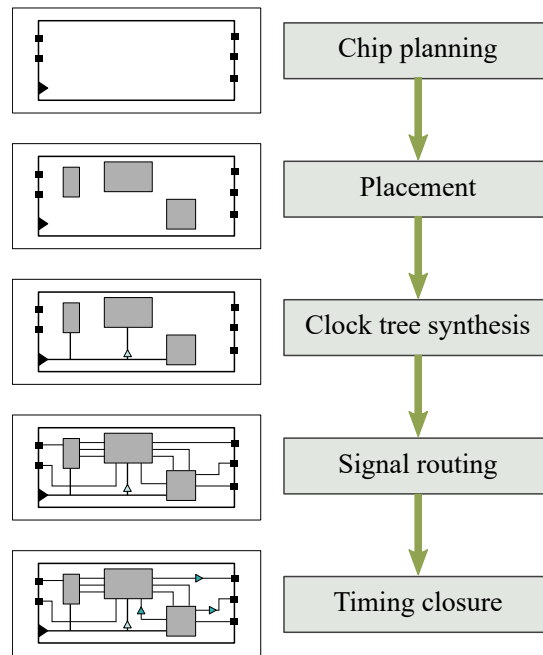


Figure 3.8: Steps in physical design.

load at port OUT1 should also be provided for proper estimation of the timing paths in the logic block 'B'.

A *multicycle path* is a timing path between a start and an end point, for which the path delay is allowed to be a multiple integer of the clock period, instead of the usual single cycle requirement. A *false path* is a path that physically exists, but will never be traversed due to the implemented logic function of the circuit. Defining a path as a false path excludes it from the timing analysis.

The setup checks are done by taking into account the longest timing path between the start and end points, and for hold checks – the shortest. Therefore, usually the slow corner is the most critical for setup checks, whereas the fast corner is more relevant for hold checks.

3.2.2 Physical Design

During physical design, the circuit of the design is converted into geometrical representation, i.e. layout. Fig 3.8 shows the major steps in physical design.

The first step is the chip planning that consists of three tasks: floorplanning, I/O placement and power planning [KLMH11]. Floorplanning means defining the shape and size of the module or the die, and determining the locations of the various subsystems. The position of the input and output ports is defined during I/O placement, and power planning specifies the power supply concept. A design can have multiple *power domains* – each power domain contains elements that share the same power supply source and

have the same voltage requirements. The power domains should be physically separated, and their allowed positions are defined with one or more *voltage areas* for each power domain. This allows for different parts of the design to have be synthesized with different voltages, in order to fulfill their requirements. High speed circuits use higher voltage, whereas the less demanding blocks are connected to lower supply voltage to save power. Moreover, power domains can be switched off to reduce the power losses related to leakage in blocks that are not active in a certain time frame.

After the chip planning, the placement step finds the location and orientation of the *hard macros* (building blocks with fixed functionality and layout). Afterwards, the standard cells are placed, attempting to minimize the overall length of connections between elements. The main objective of the placement is to produce a layout that is routable, i.e. every connection in the design can be implemented. The user can add physical constraints during placement to ensure certain goals. For example, placement blockages can be defined on certain positions, so that no cells can be placed in that area at all or can be placed only until given utilization is reached. Utilization is the ratio between the area occupied by standard cells and the whole area. Furthermore, the relative placement of a group of cells can be defined, that is the positions of the cells relative to each other is specified. Similarly, a group of cells can be constrained to be placed within a given area, however the relative position in this case is not specified.

Clock trees have a central role in synchronous designs. The task of a clock tree is to distribute the system clock to all the registers in the design and ensure low *clock skew*, i.e. the arrival of the clock signal at the same time at all endpoints. As the signals in the clock tree have a high toggle rate, the design should ensure optimal sizing for minimizing power consumption. The task of the clock tree synthesis (CTS) step is to create the clock tree, while meeting the above-mentioned requirements. Buffers and/or inverters are inserted to re-power the clock signal when necessary and in such manner that the skew between the leaves of the tree is minimal. If clock gating is implemented, then the CTS also includes the clock gating cells where specified. Clock gating is a popular technique for reducing dynamic power dissipation by turning off the clock towards parts of the design that are inactive currently.

Next, during the routing step, the physical connections are realized using wires in the metal layers, while the total interconnect length should be minimized. The metal layer and the width of each net is defined automatically by the tool. However, it is possible for the user to set advanced routing techniques for certain nets. For example, the width of a net may be increased to reduce the RC related delay, or the spacing to neighboring nets can be increased to minimize crosstalk. Alternatively, shielding can be applied at each side of a net, which on the other hand increases the sidewall capacitance of the net and degrades the performance. Modern ICs have very high transistor density and therefore require the use of many metal layers to allow for routability, which in turn increases the total number of vias in the design. The number of vias has impact on the product

yield [Gol08] and therefore a via minimization process is usually executed post-route to mitigate this effect.

Afterwards, timing closure is performed to ensure that the timing constraints are met after the parasitics and corresponding RC delays in the interconnects are considered. Typical modifications that can be done to mitigate these problems are resizing of gates, inserting buffers on long paths to decrease the load that the preceding gate drives, and modifications of the clock tree.

Finally, filler structures are added at poly-silicon and each metal layer to create a more uniform density. This is a requirement from semiconductor producers, to reduce the *dishing* effect during chemical mechanical planarization. Dishing is the decreasing of the metal height at given locations due to over-polishing and is an undesirable effect.

3.3 Motivation for Applying the ASIC Flow in RF Circuit Design

Traditionally, high speed and RF building blocks are designed using the custom design flow. This allows the designer to closely control all the performance parameters of the building block in hand. However, there is an incentive to move the design of the building blocks into the ASIC flow, and the reasons for that are discussed below.

The custom design flow provides continuous transistor sizing, as discussed in Section 3.1. While this offers high flexibility, this approach has the drawback of being very time consuming and can be done effectively only for small blocks. Additionally, continuous transistor sizing in custom design becomes obsolete with newest FinFET transistor technologies in deep submicron processes. Due to the nature of the FinFET, the gate sizing is discrete [BCJP03, LAG⁺03] and is defined by the number of fins between drain and source. In that regard, custom design is losing its advantage of low granularity compared to ASIC implementations.

Furthermore, in custom design, migrating to a different technology or making minor changes is not possible without devoting manual design and layout efforts. Due to the increasing complexity of the circuits, the time needed for completing a traditional analog design is continuously growing. The layout creation is also impeded by the complexity and increasing number of physical design rules in deep sub-micron CMOS processes. The standard-cell library based approach, on the other hand, relies heavily on automated tools that provide means for faster redesign cycles, reuse of existing designs and therefore reduce the time-to-market.

In custom design, blocks are usually optimized once for a certain use-case. If similar functionality is needed elsewhere, the blocks are reused as-is, even if the requirements are different to avoid time intensive redesign cycles. This results in sub-optimal designs with considerable power or area overheads. On the other hand, using the heavily automated ASIC design flow allows the designers to tune each design to the requirements

for a particular use case, reducing the overhead and hence providing better optimized, purpose-built designs.

Additionally, the functional verification in the ASIC design flow is much faster and has higher coverage as compared to the few handcrafted test cases that can be verified with analog simulation in the custom design flow. The use of STA for analyzing the timing performance of the circuit offers full coverage for the timing verification.

Transferring the design within the ASIC toolchain also has the big advantage of using the effective automated place and route capabilities of commercial design tools. This way the designers benefit from high design speed, good reproducibility of the results and reduced workload. Additionally, as HDL descriptions are generally technology independent, transferring the design to other technology nodes is considerably faster, as the manual work is reduced to minimum.

Despite all the above listed advantages of the ASIC design flow, there are certain limitations that have to be overcome in order to apply it in the design of transceiver building blocks.

One shortcoming is that the STA checks do not verify the design's functionality. Hence the architecture must be well-understood and multiple design aspects should be considered by the designer when defining the timing and structural constraints, such as clock propagation, clock-domain crossing, multi-cycle and false paths.

Another limitation of the ASIC flow is the limited performance parameters that it can verify. The flow cannot cope with analog performance parameters, such as linearity, gain and noise. Whereas linearity and gain are not relevant for digital circuits, (phase) noise is a central performance parameter for RF clock signals, as discussed in Chapter 2. Hence, using commercial ASIC design tools is only possible if phase noise analysis capabilities are built into the toolchain.

4 Automated Design of PLL Building Blocks

A digital phase-locked loop (PLL) operates entirely on digital signals (See Section 2.4.1.1), thus a large part of the PLL design is already automated – most digital blocks, such as the digital low pass filter, the control logic, and sample rate converters, are designed within an ASIC toolchain.

However, the time-to-digital converter (TDC) and the multi-modulus divider (MMD) in the RF clock synthesizer PLL are designed using the full-custom design flow, so that their performance requirements, discussed below, can be tightly controlled. The noise generated in the TDC and MMD affects the overall performance of the transceiver, as discussed in Section 2.3.3. Additionally, mobile transceivers have stringent low power requirements. The MMD operates on the frequency of the digitally controlled oscillator (DCO), hence it also has tight high-frequency requirements. The quantization noise, related to the resolution of the TDC, and its measurement precision affect the overall phase noise performance of the PLL.

Enabling the implementation of these blocks by ASIC design tools will utilize advantages, such as fast implementation speed, exhaustive timing coverage and technology independent design. Once a robust design method is established, the designs can be easily transferred into more advanced technology node and benefit from the process improvement, further closing the performance gap to custom designs.

In this chapter we target the design automation of these two building blocks, to assist the design of PLLs for cellular transceivers' RF clock synthesis. The architecture of the blocks has to be selected such that it is favorable for HDL description, logic synthesis and automated physical design, and it meets the above listed performance requirements.

The architecture, constraining and challenges of both building blocks are discussed in the following sections. Section 4.1 discusses the design and constraining of the MMD, and Section 4.2 presents the TDC design.

4.1 Multi-Modulus Divider

A frequency synthesizer able to produce variable output frequencies requires a multi-modulus divider in its feedback path, as discussed in Section 2.4.1.1. The most important parameters of a MMD are the power consumption, operating frequency and operating range. Additionally, for frequency synthesis the jitter is also important, as it contributes to the total phase noise of the clock synthesizer circuit (See Section 2.4.1.3).

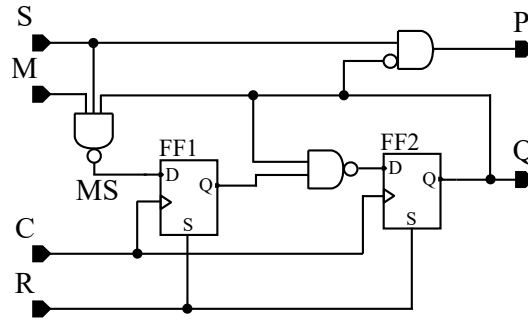


Figure 4.1: 2/3 prescaler.

Synchronous implementations, such as [AHGR11], are possible and offer well-defined timing paths for timing constraints. However, due to the high switching activity of all the registers, synchronous designs suffer from high power consumption. Furthermore, synchronous designs cannot achieve as high operating frequency as asynchronous implementations. Therefore, most of the modern MMD architectures rely on dual-modulus prescalers connected as ripple counter, such as the architecture reported in [VFL⁺00].

4.1.1 MMD Architecture

Taking the requirements for RF frequency synthesizers into account, an architecture based on cascaded prescaler blocks was selected. This provides high operating frequency, and low power consumption. The modular structure of this architecture enables a wide range of division ratios to be achieved.

However, the long chain of cascaded cells causes large accumulated jitter. To mitigate this, the output is re-synchronized with the input clock signal. This reduces the output pulse width and the duty cycle of the signal, but the phase detector in the PLL is only sensitive to the rising edges of the reference and the divided output signals, therefore the duty cycle of the output signal of the MMD is not relevant.

The architecture relies on prescalers with variable modulus (division factor). Fig 4.1 shows one possible realization of a 2/3 prescaler. It is a realization of a finite state machine (FSM) with 4 states, consisting of two registers and combinational logic. The four states are encoded with two bits, stored in the two registers FF1 for MSB, and FF2 for LSB. The state transition diagram of the block is shown in Fig. 4.2.

For division by 2, only the register FF2 is changing state, inverting its output on every rising clock edge, while the register FF1 stays at static '1'. This is the cycle $S2 \rightarrow S3 \rightarrow S2$. The state $S1$ is reached only when the MS signal is low. Reaching the state $S1$ adds one additional step to the cycle, which is now $S2 \rightarrow S1 \rightarrow S3 \rightarrow S2$, and causes the division by 3. The state $S0$ is not part of the normal operation cycle and is unreachable under normal conditions. The ports P and S, and the associated logic gates serve for signaling within the MMD and will be discussed below.

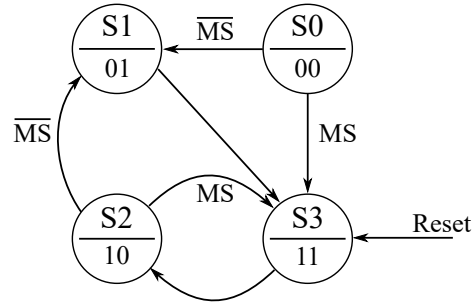


Figure 4.2: State transition diagram of the 2/3 prescaler.

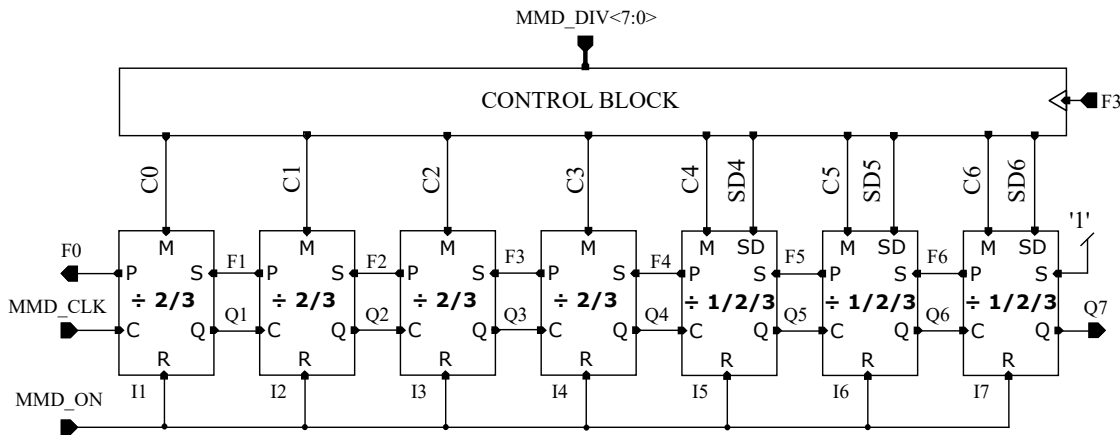


Figure 4.3: Block diagram of the implemented MMD.

A block diagram of the MMD is shown in Fig. 4.3. Each prescaler can divide by 2 or 3, and the last three blocks can be also bypassed, i.e. divide by 1. In this configuration, the divider has a maximum divider ratio of 255, and a minimum of 16. This structure is similar to an asynchronous counter, counting down from all-one to all-zero state. The divided clock is observed on the Q7 output of the last divide-by-1/2/3 stage. Through the feedback path, it propagates back to stage I1, where it is re-synchronized with the input signal. The additional control block contains registers, storing the control word for division, and logic for creating the control signals.

The feedback path (signals F6-F0) transmits the information about the state of subsequent stages back to the first stage – when all succeeding stages have a '0' at the output, the feedback signal F[0-6] is '1'. This is a key feature for the operation of the MMD, and makes the constraining for logic synthesis and clock tree synthesis a very delicate and crucial task. The circuit operates as follows:

In the most trivial case, when the first 4 stages divide by 2 and the last three stages are bypassed, the result is a division by 16. Divisions of 32, 64 and 128 are also straight forward to achieve – by bypassing only the last two, the last or none of the stages, respectively. To achieve an arbitrary odd divider ratio, the first stage must swallow one

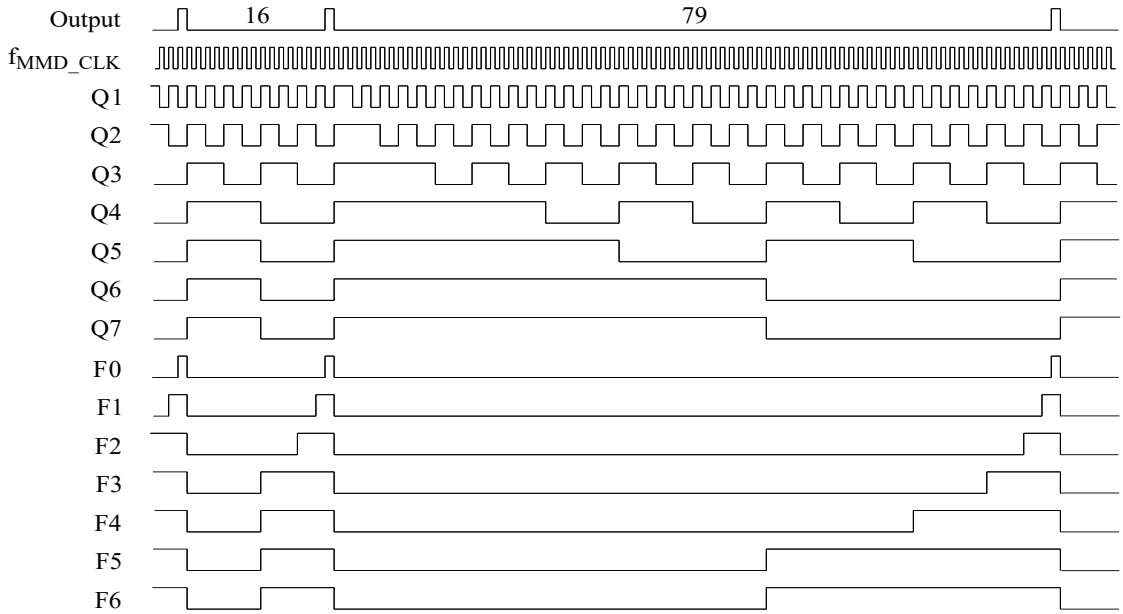


Figure 4.4: MMD timing.

of the input pulses. Swallowing a pulse means that the stage "overlooks" one pulse from the input train and keeps its output high for one clock period longer, thus the stage divides by 3 only once. If the stage divides by three all the time, the resulting division ratio of the MMD will always be multiple of 3. This necessitates using end-of-cycle logic, consisting of the feedback (Fx) signals.

An illustration of the operation of the MMD is shown in the timing diagram in Fig. 4.4. The first divider cycle shows the operation for dividing by 16 (last three stages bypassed), and the second cycle shows the division of 79. In this case, the pulse swallowing can be observed in the beginning of the divider cycle for signals Q1-Q4.

The feedback signal controls the stages such that they swallow a pulse only once at the beginning of each divider cycle, detected by the end-of-cycle logic, when the F signal was high. The swallowing itself occurs when all F signals are low again. By applying the same line of thought to the following stages, one can observe that swallowing a single pulse in the second stage increases the overall divider ratio by two. Swallowing in the third stage increases it by 4. The base ratio is 2^N , where N is the number of stages. Swallowing a pulse in each of the stages adds 2^{n-1} to the final ratio, where n is the current stage number. The highest divider ratio can be achieved when none of the states are bypassed and each of them swallows a pulse. It is calculated using the equation $DIV_{MAX} = \sum_{i=0}^N 2^i$. In the presented implementation $DIV_{MAX} = 255$.

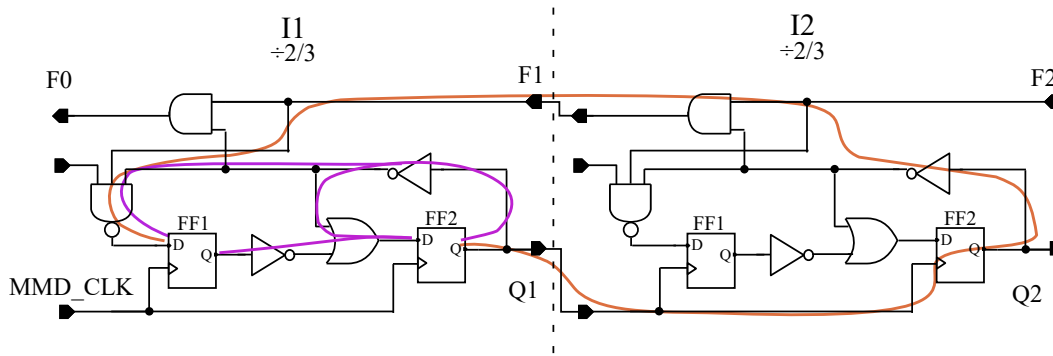


Figure 4.5: Critical paths in stages I1 and I2.

4.1.2 Constraints Development

The presented divider architecture contains multiple clock domains, interacting with each other, implying complex clock constraining, and various multicycle and false paths. The proper constraining is therefore critical – not only for the operation of the circuit, but also for the overall quality of the results. The constraining should reflect the reality as close as possible to avoid over- or underconstraining. Therefore the knowledge about functional details is an essential part of the constraining step. The focus is set on handling high input frequency and the asynchronous nature of the design by constraining, as well as reproducibility and scalability of the resulting design.

Clock Constraining

Defining the clocks properly is a central topic in static timing analysis. It is especially important for this divider structure; each block generates a clock and these clocks have to be defined accordingly. Additionally, each clock is treated as data in the feedback path, which further increases the constraining complexity.

The most straight forward approach for defining the clocks is to define only one master clock, which clocks all the registers of the design. That approach would clearly be overconstraining, because the frequency at the output of every stage, being the clock for the next stage, is at least divided by two. This implies defining generated clocks for every stage, so that the logic is optimized for the frequency it really operates at, and not for f_{MMD_CLOCK} .

Definition points for sources of the clocks and generated clocks are crucial for correct synthesis in the synthesis tool. Since all the clocks in the design are also considered as data, this becomes a critical point for writing constraints and the way they are interpreted and correctly implemented by the tool. The general recommendation is not to define clocks on hierarchical pins, but only on output pins of registers. On the other hand, defining a clock on a data path in the design is *segmenting* this data path, meaning that the timing arc through that point is broken and calculation of the

timing path cannot be done. For example, defining the clock on the output pin of the register segments the timing path through it. This has dramatic consequences in the synthesis step. The tool is being misled and cannot optimize the design correctly, simply replacing the tech-independent cells from the elaboration step with standard cells without optimization. The result is a poor design, with excessive area usage and bad timing. This behavior implies creating constraints for the synthesizer, which do not exactly reflect the "physical reality", but allow the tool to synthesize a reasonable circuit. The generated clocks are defined on the hierarchical output pins of every stage. After synthesis, on the contrary, all the clocks can be defined on the output pins of the registers. This is recommended for correct calculation of the inserted latency of the clocks.

This fact leads to the solution of creating two different timing constraint files - one prior to synthesis and a post-synthesis one. The former for guiding the synthesizer to properly create the netlist, and the latter for correctly reflecting the physical requirements of the design, so that the timing reports are trustworthy.

4.1.2.1 Critical Paths Constraining

During design compilation, the tool does not take into account the latency of the clocks, caused by the delay of the clock distribution network. This is an important observation for the constraining of the paths in the design. Let us consider the path from I2 to I1, shown in red in Fig. 4.5. It should be covered in 2 MMD_CLK cycles, at 4 GHz this equals 500 ps. On the other hand, the clock-to-q delay of a register can roughly be 130 ps. This means that the tool is calculating the delay of the path with an error of 25 % of the clock period, when omitting the clock latency! A workaround for this issue is to define some latency to the generated clocks, based on an assumed clock-to-q delay of a register. This manually defined latency is removed after synthesis by applying the new constraint during APR as mentioned above.

4.1.3 Clock Tree Synthesis

Clock tree synthesis (CTS) is the step, in which the clock tree is created and buffers are inserted if required. Its goal is to create the clock tree in a way that does not violate the design rules and to keep the skew of the clock below a user defined value. However, in the MMD design the structure of the clock tree and the logic are merged together. Every combinational gate in the clock tree is considered by the tool to have pin delay that in fact is the delay of the gate. Because of this pin delay, the CTS encounters transition violation and inserts buffers in the input net to compensate for the slow transition, and/or buffers the output to reduce the load of the violating gate. This is an irresolvable situation and it occurs at all the generated clocks that are not connected exclusively to register clock pins, but also fan-out to combinational gates. Considering the structure of the MMD, we can claim that it inherently contains the clock tree, so

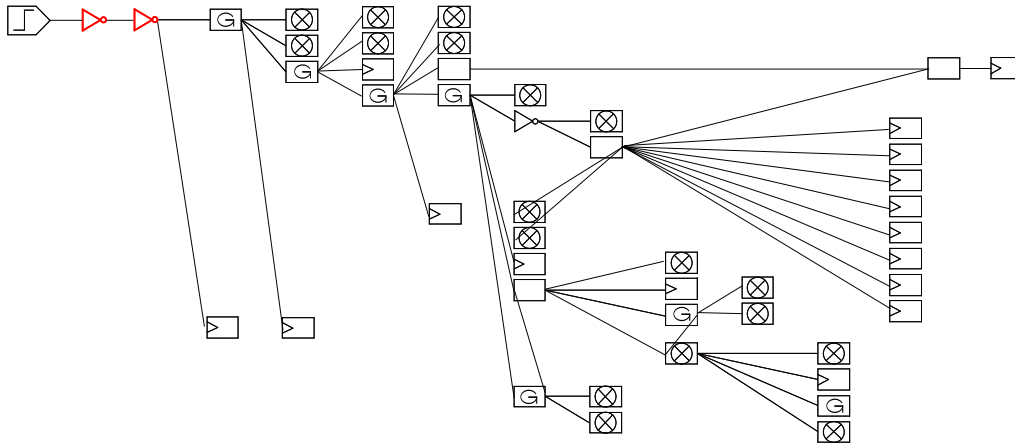


Figure 4.6: Clock tree synthesis results.

synthesizing a clock tree for it is not necessary. Therefore, the clock tree synthesis in the MMD is suppressed. The CTS tool is only allowed to buffer the root of the clock tree to take care of input transition constraints. Figure 4.6 shows the CTS results. Only the gates that are part of the clock tree are shown in this graph. Blocks with 'G' letter inside are sources of generated clocks, i.e. registers, blocks with an 'X' symbol define stop points for the clock. The empty rectangles define combinational logic. There are only two buffers inserted by CTS (marked in red) at the root of the clock tree, fixing input transition violation.

4.1.4 VHDL Design

The divider blocks, as well as the control block, are modeled behaviorally. The top level is a structural description, connecting the behavioral blocks together, and inserting some additional glue logic. This type of coding is readable and easy to maintain. The prescaler instantiations are written with parametrized generic statements, allowing for uncomplicated change of the achievable division ratios of the MMD.

For modeling the divide-by-2/3 behavior, FSMs are used. Medvedev FSM was selected to avoid additional logic on the critical paths. A Medvedev FSM does not have output logic, but the states themselves are the output. Note the lack of logic gates between the pins FF2/Q and Q that lie on the critical paths (Fig. 4.5).

4.1.5 MMD Performance and Discussion

The pre-silicon verification showed correct operation of the implemented divider to maximum frequency of 4.3 GHz, based on the STA reports under worst conditions (slow process, low temperature, low voltage). Analog simulations, on the other hand, showed operability even up to 5.6 GHz. The difference is based on the fact that on-chip variations are not taken into account in the analog simulations.

Table 4.1: MMD implementations based on different design techniques.

CMOS Process		28 nm	
Nominal Supply Voltage		1.1 V	
	Synchronous counter	This work	Hand-crafted MMD
Design effort	LOW	MED	HIGH
Max frequency (nominal, analog sim.)		5.6 GHz	10 GHz
Max frequency (worst case, STA)	2.5 GHz	4.3 GHz	
Chip area	130 μm^2	140 μm^2	210 μm^2
Power@2GHz	213 μW	153 μW	109 μW

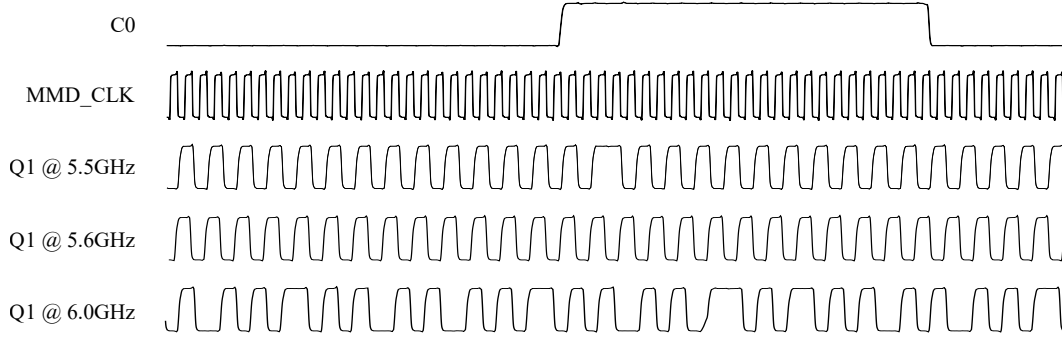
Table 4.1 displays the results from different implementations of the MMD. For comparison reasons, a standard digital synchronous counter with programmable reset was created to serve as the most straight-forward multi-modulus divider reference. The hand-crafted MMD is the reference design from the full-custom point of view: manually designed and laid out. This circuit contains true single-phase clocked (TSPC) registers in instance I1 optimized to handle the high input frequency from the DCO. The reported maximum frequency for it is solely based on analog simulations, since it contains custom made, not characterized cells and cannot be analyzed within the STA flow. This performance gap between the full-custom approach and this work could be closed by using a more advanced cell library, containing high-speed TSPC cells, in the semi-custom flow [CK04]. The quality of results increases from left (synchronous divider) to right (hand-crafted): the maximum achievable frequency increases and at the same time the power consumption decreases. Omitting the base effort for constraint and VHDL coding, the implementation effort can be shown in the table. The high effort of the full-custom layout is justified by the fact that for example increasing or decreasing the divider ratio or moving to a new technology requires a full redesign of the module. The low effort of the synchronous counter is based on the fact that abstract behavioral modeling and standard constraining can be used, as opposed to the presented work.

Other technologies such as SiGe BiCMOS [RSR⁺09], or different architectures of circuit design techniques such as TSPC-based logic [JFF⁺12], might achieve even higher operating frequencies than the hand-crafted MMD used as comparison. However, the focus of this section is to highlight how designs which are designed in high-effort full-custom approaches today can alternatively be designed using "fully automated" semi-custom techniques, resulting in significantly higher designer productivity and improved design portability.

Table 4.2 summarizes briefly results of two MMD designs, synthesized for a maximum frequency of 4.3 GHz and 2 GHz, respectively, both operating at 2 GHz. One can

Table 4.2: Purpose-built MMD vs. design reuse.

	Designed for 2 GHz	Designed for 4.3 GHz	Overhead
Cell area	94 μm^2	119 μm^2	26.6 %
Power@2GHz	127 μW	153 μW	20.5 %

**Figure 4.7:** Analog simulation of the MMD.

clearly see that purpose-built design has less area and power consumption, highlighting the advantage of the ASIC design flow – instead of reusing an existing design, synthesized and laid out for higher frequency, one can easily redesign it for the required maximum frequency or even different divider ratios, saving power and area related to over-constraining.

Finally, Figure 4.7 and Table 4.3 demonstrate the correlation between the analog simulations and the STA reports. The timing diagram shows the pulse swallowing control signal (C0), the input pulse train (f_{MMD_CLK}) and the output of the first stage (Q1) at three different input frequencies. The table sums up the worst slacks from the APR tool. The path from I1/FF2 to I1/FF1 is the one responsible for the pulse swallowing, and the path from I1/FF2 to I1/FF2 is the very first divide-by-2 logic. At 5.5 GHz the design operates normally and the pulse swallowing is taking place. If the frequency is increased, the pulse swallowing logic fails and a further increase causes a fail of the divide-by-two logic. Despite the difference in absolute numbers, qualitatively the results from STA analysis and analog simulations showed good correlation, which demonstrates the reliability of the standard-cell based flow. Hence, no extensive and time consuming analog simulations for verification will be required for a redesign.

The synthesizable design was ported into a more advanced technology with higher layout complexity within one man-week, whereas a full-custom design is estimated to require at least 5 man-weeks, which underlines the benefits of the presented approach.

Table 4.3: STA timing report.

Startpoint	Endpoint	Slack
I1/FF2	I1/FF1	1.25 ps
I1/FF2	I1/FF2	13.5 ps

4.2 Time-to-Digital Converter

A time-to-digital converter (TDC) measures the time difference between two events. It is an integral part of digital phase-locked loops, serving as a phase detector, measuring the time difference between the rising edge of the reference signal and the (divided) output signal (See Sec. 2.4.1.1).

There are different approaches for measuring time. One traditional approach is to convert the measured time interval into voltage and consequently digitize the voltage with the aid of an analog-to-digital converter (ADC). As the ADC has a limited resolution, increasing the measurement range of the TDC would imply improving the resolution and vice versa [Hen10]. A way to achieve that is to first measure the time interval coarsely and then use fine quantization for the reminder. This technique is used for example in [MK98], to increase the dynamic range of the TDC. These techniques, even though functional, are analog in nature and suffer from the problems that analog circuitry faces in deep sub-micron processes.

Therefore, to fully benefit from the process scaling, fully digital implementations are preferred. The most basic digital method of measuring a time interval is to use a high-speed counter. Counting the number of periods of the reference clock can be used to measure the unknown interval. This architecture has the advantage of being very simple and the dynamic range is limited only by the number bits of the counter. However, its resolution is limited by the maximum operating frequency of the clock. Even in deeply scaled CMOS technologies, the maximum operating frequency of a counter cannot exceed 10 GHz, yielding a resolution of 100 ps. Additionally, as the start and stop of the measurement are completely asynchronous to the reference clock, the quantization error is twice the period of the clock.

In order to measure even smaller time intervals, a resolution finer than one clock period should be achieved. This can be done by producing multiple delayed copies of the signal with a delay line. A start signal propagates through the delay line and on the rising edge of the stop signal a snapshot of the delay line is taken. The transition point of the start signal in the latched data gives information on the time difference between the start and the stop signals. The resolution of the measurement in this case is limited to one unit element delay. The basic performance figures of a TDC are the resolution and the maximum time interval for the measurement. Additionally, as any other data converter, TDCs exhibit linear and non-linear errors limiting their performance.

Different approaches for TDCs and in particular for the implementation of the delay line exist to improve resolution and linearity.

A delay-locked loop (DLL) based TDC can be used when a periodic reference signal is available. It is based on a delay loop built by current-starved inverters. By adjusting the available current of the delay elements, the delay of the loop is tuned to be equal to the period of the reference signal. The reference signal is then sampled with registers on the rising edge of the stop signal as in the basic delay line TDC. The DLL architecture has the advantage that it does not require calibration due to the effect of the control loop [HRW12].

A gated ring oscillator TDC [SP09] contains a loop of current-starved inverters. Using the current-starving control, the oscillator runs only during measurement and is frozen afterwards. The next measurement releases the ring and the residue from the previous measurement interval is transferred. This allows for noise-shaping of the quantization error. For speeding up the ring, multi-path functionality can be added -- instead of a delay element i taking its input only from the preceding element $i-1$, multiple-input elements allow interpolation between the signals, e.g. by shorting $i-1$, $i-3$, $i-5$ etc. in case of inverters.

In contrast to the delay based architectures above, a delay-cell-less TDC [SJPK13] uses registers with adjustable setup times to sample the same reference signal. Instead of using delay cells to create phase difference between the different sampling elements, the same functionality is achieved by adjusting the setup time of the registers. Fine tuning of the setup time allows for sub gate delay resolution.

All of the advanced architecture listed above have the disadvantage of being very analog intensive. A Vernier based TDC has two delay lines for the start and the stop signals. The delay elements of the line for the start signal are slightly slower than these in the stop line. During measurement the start signal propagates through the start delay line and is being slowly overtaken by the stop signal. The stop signal catches up on each stage with the difference of the delays between the two dedicated elements. The position where the stop caught up with the start signal is detected by a register. This architecture is very promising for being synthesizable, as shown in [YW11], but it has the disadvantage of big latency that is also dependent on the measured time interval.

Other synthesizable TDCs exist, exploiting mismatches in the layout. These topologies use parallel sampling elements [KHO⁺09] or parallel operation of complete TDCs [VBSC12]. Instead of minimizing the effects of non-symmetries and mismatch, these circuits take advantage of the differences in the circuits by spatial averaging. Even though effective, this approach is power hungry and area intensive due to the parallel circuitry.

We target low-power TDC, therefore redundancy has to be avoided. In this work we minimize the non-linearities by design measures and layout effort. The decision for a looped inverter ring based TDC is driven by the fact that this architecture is purely digital in nature and well suited to fulfil the requirements of cellular transceivers [VLB⁺06].

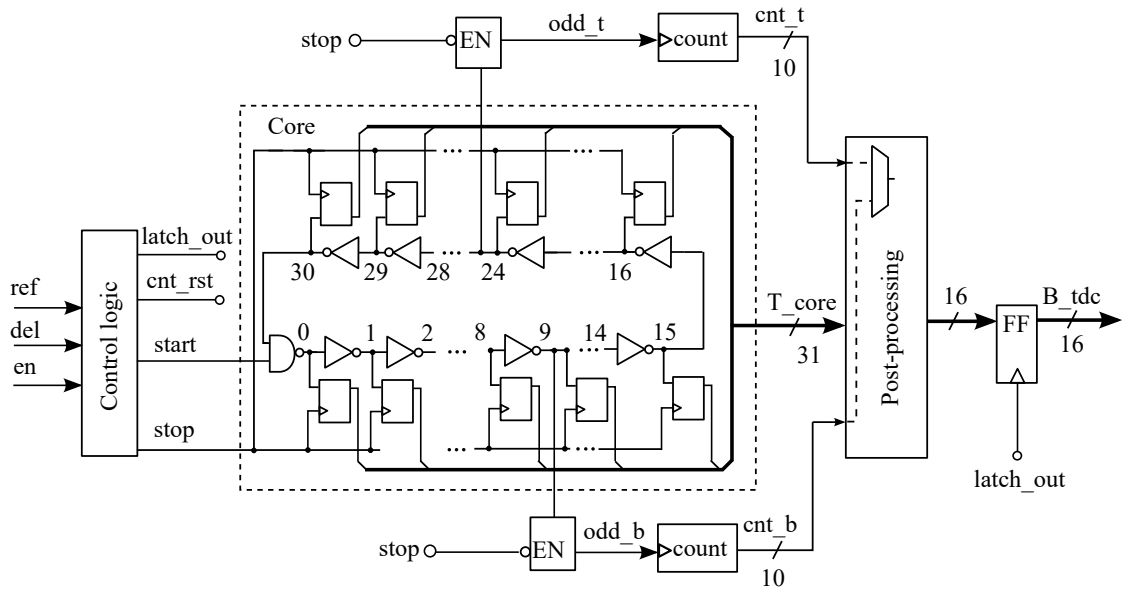


Figure 4.8: TDC block diagram.

4.2.1 TDC Architecture

The selected architecture is a time-to-digital converter in looped configuration. Compared to a basic delay line, this increases the effective length and hence offers extended measurement intervals. The architecture consists of control logic for start and stop, the TDC core, loop counters and digital post-processing (Fig. 4.8).

Due to the asynchronous nature of the design, there is always a possibility of metastability occurring in the loop counter's logic. To account for that, this architecture contains two counters tapping the ring on opposite sides. When the measurement is done, the post-processing logic selects the result from the counter that was further away from the switching event and is therefore guaranteed to have a valid state [Hen10].

4.2.1.1 TDC Core

The TDC core consists of a looped delay line and sampling elements. The delay line contains 30 inverters plus one NAND gate that doubles as an inverter and control element for the chain. Additionally, there are two latches tapping the chain for the two loop counters. They are enabled only while the measurement is running.

The inverter is the fastest gate in the CMOS technology, therefore it is selected to serve as a delay element for the ring. The inversion, however, adds challenges to the design. In this case both the rising and falling edges are used for the measurement. The rising and falling delays of the inverters are only partially correlated and the process variations affect them independently. This leads to systematic asymmetries in the design. Furthermore, the sampling elements see alternating transitions on their inputs. This

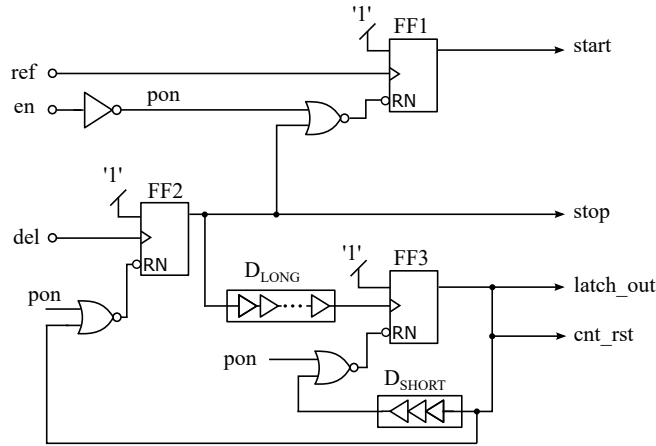


Figure 4.9: TDC control logic.

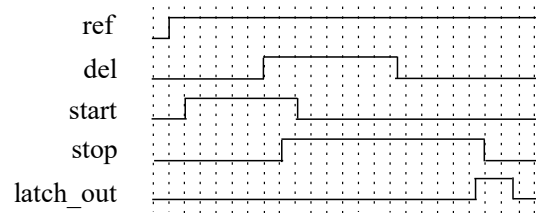


Figure 4.10: TDC control logic timing diagram.

results in additional non-linearity, because registers usually have different setup time for rising and for falling edges on the data input.

4.2.1.2 Control Logic

The input signals ‘ref’ and ‘del’ cannot be used directly to steer the TDC core, as there are no guarantees about the incoming pulse width. Therefore, the control logic block (Fig. 4.9) generates the internal signals necessary for the operation of the TDC. These are the start and stop signals for the TDC core, and the clock for latching out the measured data (latch_out), which also doubles as reset signal for the loop counters (Fig. 4.10). These signals are generated by using delay cells and self-resetting registers.

The delay of D_{LONG} defines the time given to the post-processing logic to finish the processing of the data. The delay cell D_{SHORT} in the feedback path from the output to the reset pin of the register FF3 is responsible for creating the falling edge of the latch_out pulse. The delay should be long enough to satisfy the minimum pulse width requirement of the registers in the core.

Any asymmetries in the load and wiring of the registers of this block will introduce an additive offset error in the input-output characteristic of the TDC. However, offset

error is not critical for most of the applications and can also be calibrated by adding a correction term to the output.

4.2.1.3 Postprocessing

The task of the post-processing block is to convert the output of the TDC core and the counters into a binary word. The output of the core is in pseudo-thermometer code, whereas the counters create binary coded outputs. The counters operate only on the rising edge from the delay line, so they count up every second (even) cycle. The value of the counter should therefore be taken with a weight of 62, i.e. twice the loop length. To account also for an odd pass of the ring, the falling edge of the counter clock signal need to be taken into consideration, with a weight of 31. To select the output of the counter with a guaranteed valid state and the clock signal with a stable value, an additional signal *hit_t* is used, which indicates a detection of the switching event in the 15 MSBs (top) of the core. Therefore, the output of the TDC is evaluated as (shown in decimal):

$$B_{tdc_{10}} = B_{core_{10}} + (hit_t \cdot \overline{odd_b} + \overline{hit_t} \cdot \overline{odd_t}) \cdot 31_{10} \\ + (hit_t \cdot cnt_b + \overline{hit_t} \cdot cnt_t) \cdot 62_{10}$$

where *B_core* is the position of the switching event in the core, *odd_b* and *odd_t* are the clock signals for the counters, and *cnt_b* and *cnt_t* are the outputs of the bottom and top counter, respectively.

Based on the presented architecture, block functionality and technology, the targeted resolution of the TDC is 8 ps.

4.2.2 Synthesis and APR

The design and layout of the TDC should be done in a way that minimizes the linear and non-linear errors. The sources of non-linear errors are concentrated in the core block. The control logic block contributes a linear offset error only. The usage of the ASIC design flow, presented in Section 3.2, leads to the following limitations: 1) No transistor level optimization is possible, because the tools rely on a limited set of standard cells. 2) there are no high-level constraints to describe analog behavior and constrain a design for analog requirements like symmetry, matching. 3) Due to the concept of STA, based on timing windows, the timing cannot be as precise as a Spice simulation.

4.2.2.1 TDC Core

Due to the principle of operation, timing violations will inevitably occur in the sampling elements. Therefore the STA checks for the delay line are turned off and the requirements are fulfilled by employing other design constraints.

The delay line and the sampling elements are placed in a fixed relation to each other using a structured placement approach (Fig. 4.11). The placement is organized in



Figure 4.11: TDC core layout and routing.

columns; the inverters are in the middle in two columns, on both sides flanked by the corresponding sampling elements. This keeps the design compact, the wiring short, and the device matching high. The orientation of the delay chain is such that the positions of the latches for the two loop counters (shaded in the figure) are on the top and bottom side of the core. This allows to place them close to the ring and results in very short routing.

To eliminate the effects of the local variations in the clock tree for the core register, no buffers are used. Instead, an H-shaped wire that serves as clock distribution network is created using the approach outlined below. To reduce the RC delay in the network, the routing is done extra wide, as marked with purple in Fig. 4.11.

To drive the large load of this clock net, a very strong buffer is needed. However, a buffer of that size is not available in the standard cell library. Therefore, a *super buffer* was designed, created by four inverters in parallel and placed next to each other (marked red in the figure). By doing so, we achieved a good transition time and low RC delay on

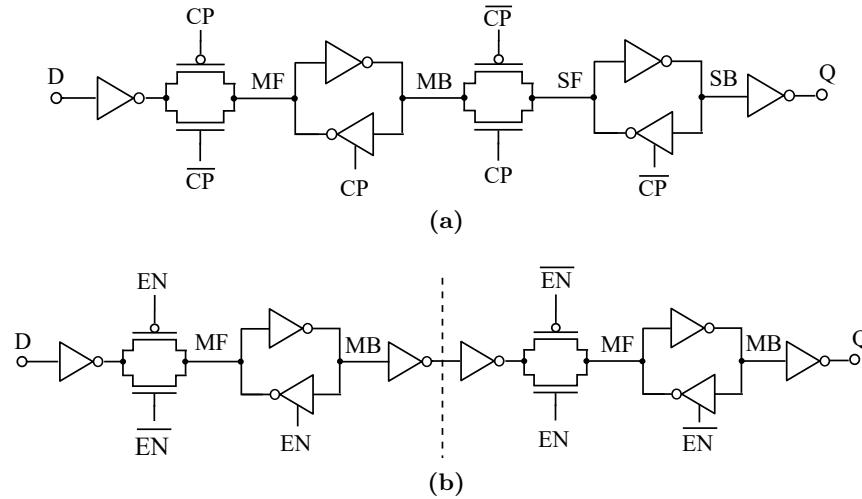


Figure 4.12: Structure of (a) a std. cell register and (b) two latches.

the stop signal without the need for new standard cells. The achieved skew in the whole line is less than 0.8 ps.

The routing of the block was realized in a multi-step approach that starts with a guided routing of critical nets and ends with auto-routing of nets belonging to "standard" timing paths, as in the post-processing unit. The routing of the very critical nets of the delay line (in blue) is done in prioritized manner, based on constraints deduced from system, circuit and technology knowledge. This is done as first routing step, to avoid any limitation from obstacles in the metal layers. Then, the clock net of the sampling elements is routed. The clock net for the stop signal is H-shaped to keep it as balanced as possible and to decrease skew. To achieve the H-shaped tree, dedicated routing constraints, such as routing corridors and blockages, have been defined. By properly selecting the allowed regions one can impose the final routing shape. The remaining standard logic signals were routed as the last step, without priority or further "manual" constraints.

The biggest available standard-cell inverters were selected for the delay chain to reduce the impact of local variation. The bigger size also reduces the effective fanout of the cells and therefore reduces the influence of the wiring variations.

Memory effect on setup time of register

A regular standard-cell register exhibits dependency of the setup time on the previous state of the register. Spice simulations show differences of up to 5 ps in the setup time for a rising transition on the D pin for $Q_{(T-1)} = 0$ vs $Q_{(T-1)} = 1$, where $Q_{(T-1)}$ indicates the state stored in the register during the previous cycle. This effect is caused by the internal structure of the available master-slave standard cell register, shown in Fig. 4.12a.

To achieve an area optimized cell, a regular standard cell register does not have a buffer between the master and slave latch. Therefore the charge stored in the node SF

Table 4.4: Setup times for standard cell register vs 2 latches.

T_{setup}	Register		Two latches	
	rise _D	fall _D	rise _D	fall _D
$Q_{T-1} = 0$	13.35 ps	2.23 ps	16.26 ps	17.21 ps
$Q_{T-1} = 1$	10.95 ps	6.55 ps	15.93 ps	18.54 ps
$ \Delta $	2.4 ps	4.23 ps	0.32 ps	0.74 ps
$ \Delta $	0.3 LSB	0.54 LSB	0.04 LSB	0.09 LSB

is directly visible through the transmission gate and serves as load of the master latch, influencing its switching time and hence the setup time of the whole sampling element. This effect will manifest itself as a dependency of the TDC measurement on the previous result and should be minimized.

To get rid of this unwanted effect, the master latch should be decoupled from the slave latch by an inverter or buffer. The simplest way to do so is to build the register from two latches (see Fig. 4.12b). Since the outputs of the standard cells are usually buffered, the desired modification is achieved without the need of using custom cells. Table 4.4 summarizes the effect for a register and two latches. The last row shows the substantial reduction of the effect when decoupling is introduced. The absolute value of the setup time is also increased, but it has no effect on the TDC performance.

4.2.2.2 Control Logic

The structure of the control logic is hard to describe in terms of behavioral modeling, therefore structural VHDL was used. This has the additional advantage that all registers will be guaranteed to have the same size, reducing the offset error. However, the description is still technology independent. Because of the complex structure and feedback loops, STA is meaningless in this block; there are no launch and capture paths in the regular sense. Therefore no clocks are defined and the STA checks are turned off. To account for possible wiring asymmetries, the cells of the block are restricted to be placed in a specific region close to each other.

4.2.2.3 Digital Logic

The backend logic is placed and routed automatically using only timing constraints. The loop counters are clocked with the loop frequency, i.e. $2 \cdot 31 \cdot T_{\text{delay}}$. The setup time for the digital logic is defined by the delay D_{LONG} .

4.2.3 TDC Performance and Discussion

The TDC is fabricated using a 28 nm CMOS process. The area of the TDC is $1200 \mu\text{m}^2$, including all discussed modules. The simulated DNL performance of the implemented TDC is shown in Fig. 4.13 in red. The simulation was done with $Q_{T-1} = 1$ (to cover

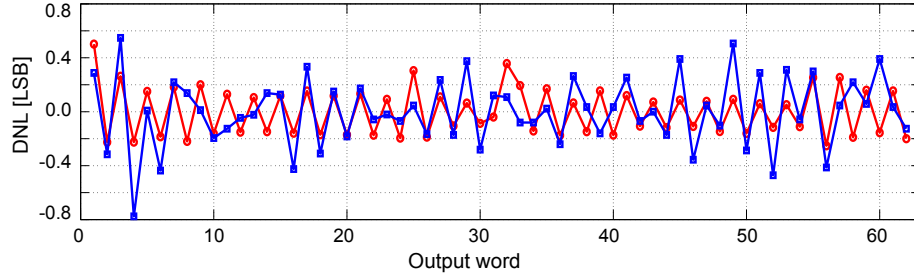


Figure 4.13: DNL simulation (red) and measurement (blue).

worst case) for all registers, which implies setup time rise-fall asymmetry of 2.56 ps. This combines with the asymmetry of the rise and fall delay of the inverters (~ 0.5 ps), leading to speeding up of the rising edges and slowing down of falling edges by 2 ps and can be observed as alternation of the DNL characteristic (falling edges are on odd positions in the characteristic). Furthermore, the disturbances in the regularity of the delay line lead to outliers in the DNL curve: The slower NAND gate is clearly visible as high DNL in position 1 and 32, the bending points of the ring at positions 8 and 24, and the tapping latches (at 10, 25) are also causing disturbance. Additionally, due to the inverting nature of the ring, it requires an odd number of stages, and hence there is a gap in the loop at position 24 (Fig. 4.11). This systematic mismatch could be further reduced by linearization in a post-processing step.

The measured DNL characteristic is shown in Fig. 4.13 in blue. Due to random local mismatch, the setup times for rise and fall of the registers, and the rise and fall delay of inverters can fluctuate independently. This amplifies the DNL effect shown in the simulation, when in phase, and damps each other otherwise. The amplified effect is visible e.g. at positions 3, 4, 49, 52 and canceling out leads to zero DNL e.g. at positions 12, 13, 43.

Fig. 4.14 shows the simulated and measured INL curves. The simulated INL curve (in red) shows an expected behavior - the humps are caused by the slow NAND gates in the beginning of each cycle. At the end of every cycle of 31 elements, the average step size reaches the ideal value, leading to a zero in the INL characteristic. The measured INL curve (in blue) follows a U-shaped pattern; the drop in the beginning of the curve is caused by the accumulated effect of the outliers with low indices in the characteristic, as seen in the DNL curve.

The effect of random mismatch could be reduced by introducing even bigger buffers to the flow or by applying higher supply voltage, trading-off design performance for power.

Table 4.5 summarizes the results from simulation and measurement. The focus of this work is the move of full custom designs into the semi-custom world, therefore the performance is not compared against other state-of-the-art designs. The achieved resolution of 8.7 ps satisfies cellular requirements [VLB⁺06]. The reasons for the difference between

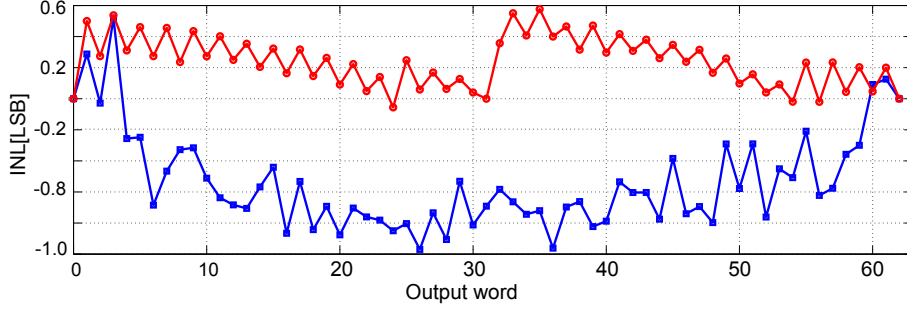


Figure 4.14: INL simulation (red) and measurement (blue).

Table 4.5: Summary of TDC results.

CMOS Process		28 nm		
Supply Voltage		1.1 V		
Area		0.0012 mm ²		
Measurement range		16 bits		
	Resolution	DNL _{MAX}	DNL _{RMS}	Power
Simulation	8.0 ps	0.5 LSB	0.17 LSB	1.7 mW
Measurement	8.7 ps	0.8 LSB	0.27 LSB	—

measured and simulated DNL were outlined above. One reason for the deviation in resolution is the model accuracy.

The listed power dissipation is the average power during the measurement of one time interval and is therefore independent from the sampling frequency. Deeper investigation on transfer characteristics and power values from measurements are not available, because the TDC was built in an ADPLL system with shared power supply within a complete transceiver.

A full-custom counterpart based on the same architecture, but using a differential delay line targeting low non-linear errors, shows a resolution of 10.4 ps, DNL_{MAX}=0.31 LSB and DNL_{RMS}=0.26 LSB, occupying 2800 μm^2 .

The synthesizable design was transferred into a more advanced technology with higher layout complexity within 40 man-hours, underlining the benefits of the presented approach.

5 Automated Phase Noise-Aware Design of RF Clock Distribution Circuits

Modern radio-frequency (RF) transceivers feature complex clock generation and distribution systems in order to provide all the frequencies required for the various supported bands and modes of operation (Section 2.4.3). Given the large distances that have to be covered, re-powering of the signal with the aid of CMOS gates is required. On the other hand, CMOS gates added to a signal path introduce noise into the system.

Noise in the clock signal induces random fluctuations in the phase of the clock, referred to as phase noise (Section 2.3.2). Phase noise in the receiver path causes reciprocal mixing, where a downconverted interferer signal is superimposed onto the desired signal, degrading the signal to noise ratio of the receiver. In transmitters, phase noise leads to an increase of the error vector magnitude, which is a measure of the modulation accuracy. Hence, oscillator phase noise and the contributions arising from the RF clock signal distribution circuitry has to be kept as low as possible. This is a very challenging requirement, considering the large die size and the tight power targets that mobile platforms need to meet.

Current design practice is to design the RF clock distribution subsystem within a custom design flow, using the methods described in Chapter 3. This requires significant manual efforts of analog experts and layouters.

This chapter presents methods and tools for estimation and minimization of phase noise in RF clock distribution networks. First, Section 5.1 presents a method for rapid phase noise analysis, based on lookup tables. Next, Section 5.2 introduces an algorithm for phase noise-aware buffer insertion in interconnect trees. An algorithm for power minimization under phase noise constraints in CMOS circuits is proposed in Section 5.3. Finally, Section 5.4 presents a complete methodology for phase noise-aware design of RF clock distribution circuits, incorporated into an ASIC toolchain.

5.1 Static Phase Noise Analysis

This section presents a method for phase noise estimation that is referred to as static phase noise analysis (SPNA). The name was selected to allude to static timing analysis (STA), on which principles SPNA is based. The basic concepts of SPNA are derived from STA and are described below.

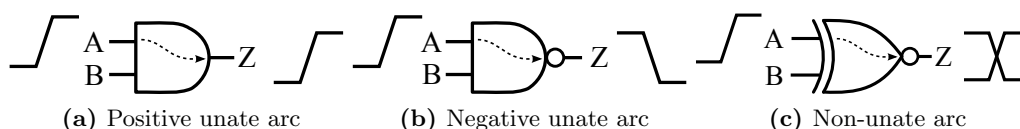


Figure 5.1: Unateness of arcs.

5.1.1 Concepts of SPNA

Standard cell: A standard cell is a pre-designed logic gate with fixed functionality and layout. Each standard cell has known noise performance and output transition times under a given range of operating conditions (output load and input signal transition time). The set of standard cells available to the designer is called standard cell library.

Arc: An arc is a path from input to output pin of a standard cell. Each arc contains information about the phase noise and transition times of the output signal under given conditions. It also defines the condition under which an arc can be traversed, depending on the function of the cell. If the arc does not exist, then there is no path between the pins or the path cannot be traversed.

Unateness of arcs: An arc can be positive unate, negative unate or non-unate. An arc is *positive unate* if a transition on the input pin causes a transition with the same polarity, or no change, on the output pin. If the output transition is in reversed polarity with regard to the input, the arc is *negative unate*. If the polarity of the output transition depends not only on the input pin, but also the state of other inputs, then the arc is *non-unate* [BC09]. Deciding if a non-unate arc is positive or negative unate based on the states of gate inputs is called *resolving* the unateness. The unateness is important for the analysis as it indicates the polarity of the signal that propagates through a cell. Figure 5.1 shows examples for the different unateness of arcs.

Static signals: A signal is static, if it does not change during the operation of the circuit. In SPNA, all data (control) signals are considered static. At points where data and clock signal merge (clock inversion, clock enabling etc.) the data signal should be in a defined static value for the tool to be able to predict the behavior, the noise performance of the gate and to resolve the unateness. This is not a limitation, as in general data signals are used to configure the clock distribution path during power-up and are rarely switched during operation.

5.1.2 Operation

SPNA operates on a netlist, the parasitics description, and the standard cell library. The design constraints, i.e. the environment of the analyzed circuit should also be defined; this includes the clock source position, transition times, driver strength and static values for the data primary inputs. Fig. 5.2 presents an exemplary netlist, including most of the structures that can be found in clock distribution networks, such as clock division, clock gating, clock inversion and multiplexing.

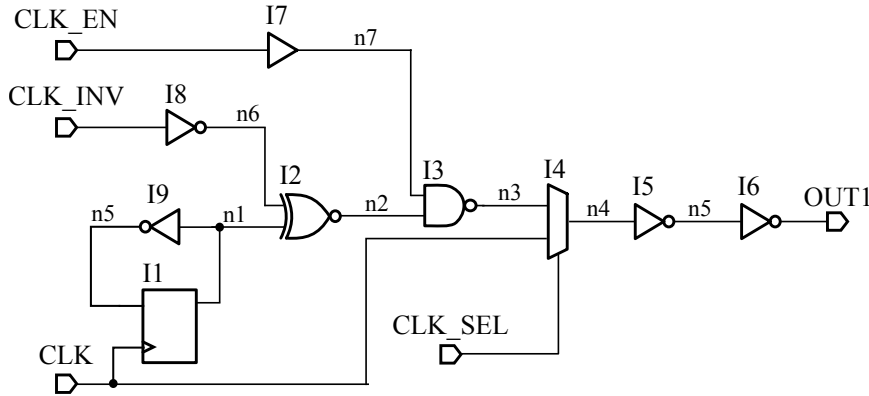


Figure 5.2: Exemplary netlist.

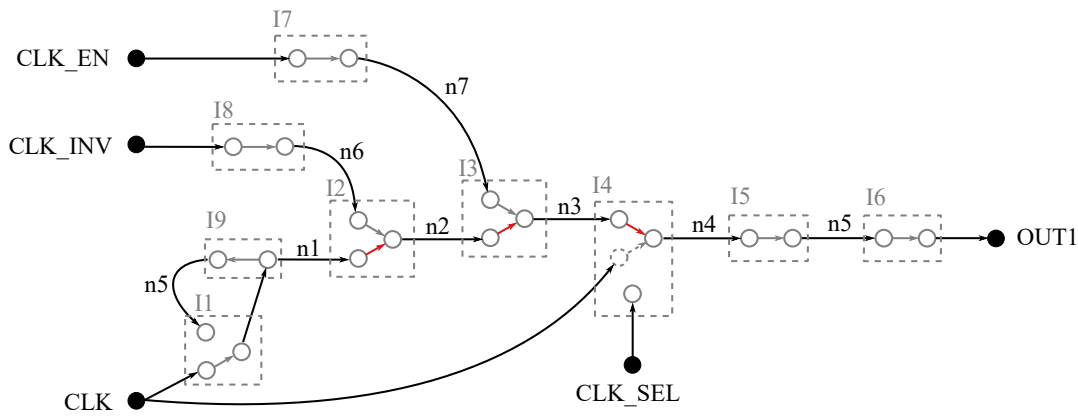


Figure 5.3: Graph representation of the exemplary netlist from Fig. 5.2.

The circuit is modeled as a directed graph, with the vertices representing gate pins, primary inputs and primary outputs, and the edges representing nets and timing arcs within gates. Each pin is associated with a gate and each gate has properties such as a name, pointer to a corresponding cell type, and pins. Each net has a name, fanin and fanout lists, corresponding parasitics, transition time and accumulated jitter. Fig. 5.3 shows the directed graph representation of the netlist from Fig. 5.2.

The SPNA starts by reading the netlist, the cell library, and the environmental information that defines the state of the primary inputs and outputs: the port where the clock is applied, the load on the ports, and the state of the control signals. The signals CLK_INV , CLK_EN and CLK_SEL are control signals responsible for the clock inversion, clock enabling and clock multiplexing, respectively. Such signals are set at start-up to define the mode of operation and do not switch dynamically during runtime. The constant values in the circuit are propagated through the netlist, enabling the arcs and resolving their unateness where applicable, defining how and if the signals propagate between gate pins. In the given example, the clock gating cell I3 requires a logic

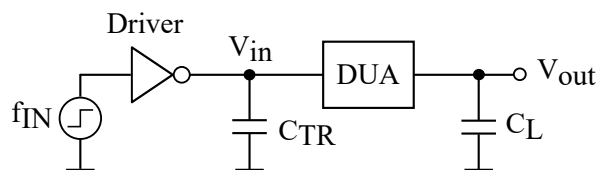


Figure 5.4: Cell characterization testbench.

'1' at the upper input pin, so that the arc, connecting the nets $n2$ and $n3$, is enabled. The gate then serves as an inverter for the CLK signal. Similarly, the logic value of CLK_SEL defines which arc in the multiplexer I4 exists. The logic value of CLK_INV input propagates to the net $n6$ and is used to resolve the unateness of the arcs of gate I2.

Thereafter the clock signal is propagated through the network in a breadth-first manner, starting from the defined clock input port. For each gate in the transitive fanout, the noise and output transition time are estimated by using its input transition, output load, and the cell description, as outlined in Section 5.1.3. The output load of a gate is a function of the input capacitance of the gates in the fanout and the length and size of the wires (See Section 5.1.4). The clock stops propagating at primary outputs or when a net with a static value is reached. In this example, for $CLK_EN = 1$ and $CLK_SEL = 1$, the clock will propagate through gates I1 to I6 and will reach the primary output OUT1.

The jitter is independent of the frequency of the signal (Section 2.3.2), and can be accumulated regardless of frequency changes within the circuit. This allows SPNA to operate without knowledge of the functionality of the circuits or necessity for the user to define clock domains within the design. Under the premise of independent noise sources, the jitter generated by each cell on the signal path can be summed, using equation (2.16). The resulting output phase noise is then calculated, using (2.15).

5.1.3 Cell Characterization

The backbone of the above mentioned method is the phase-noise/jitter information of each cell under various operating conditions. Each cell from the library is characterized for output jitter and transition times using large signal noise analysis within an analog circuit simulator. The testbench consists of a driver cell, a back annotated view of the device under analysis (DUA), and power supply. A non-ideal driver cell is required to model a real world waveform and the driver impedance. The simulations are performed in the most critical process corner for noise, which varies with the technology node. However, the proposed method is not limited to a single PVT set.

The cell is simulated with different combinations of loads C_L , and input transitions t_{in} (rising or falling). The distinct input transitions are produced by varying the capacitance C_{TR} (Fig. 5.4). The simulations run on a fixed input frequency f_{in} and thanks to the low complexity of the simulated schematic, they execute very fast. Due to the noise folding, all the noise power is folded into the frequency range $[-f_{in}/2; f_{in}/2]$. The noise

power due to flicker and white sources on an arbitrary offset frequency f_a within this range are then evaluated. The white noise is uniformly distributed and its mean-square value is

$$\langle v_n^2 \rangle = \int_{-f_{in}/2}^{f_{in}/2} S_v(f) df = S_v^{wh}(f_a) \cdot f_{in}. \quad (5.1)$$

RF circuits operate on frequencies f_{in} higher than the corner frequency of the flicker noise, hence the flicker noise does not undergo folding and its power at 1 Hz is estimated as

$$S_v^{fl}(1) = S_v^{fl}(f_a) \cdot f_a. \quad (5.2)$$

The dependencies of $\langle v_n^2 \rangle$ and $S_v^{fl}(1)$ on the load and input transition time are then organized into two-dimensional tables. In between the existing data points, a linear interpolation algorithm computes the missing values. Note that both values are independent of f_{in} .

Static signals and logic gates

The arcs within a multi-input logic gate usually depend on the state of the other inputs, thus during the characterization of a gate the state of each input should be considered. For example, in a two input XNOR gate (Fig. 5.1c), the unateness, the noise performance and transition times of the arc $A \rightarrow Z$ are dependent on the state of B. Therefore, this arc should be characterized twice, once for each valid logic level ('1' or '0') of the input B.

Registers

In STA, register require special handling, because of their synchronization character. However, in SPNA the only relevant information about a register is the CLK-to-Q arc and its unateness. A register with a non-static D-pin value is assumed to toggle and therefore propagates the clock signal, so a rising edge on the CLK pin (for positively clocked device) causes both rising and falling transitions on the output. The rate of change, i.e. the output frequency is not relevant, as discussed in Section 5.1.2.

5.1.4 Interconnect Parasitics

With shrinking technology nodes, the interconnect resistance per unit length is increasing, impacting the transition times, and hence the noise performance of the circuit. There are two effects that have to be considered:

- 1) due to the shielding effect of the resistance, the waveform at the near end (driving point) of the interconnect is changing; and
- 2) the RC delay changes the slope of the transition at the far end of the net.

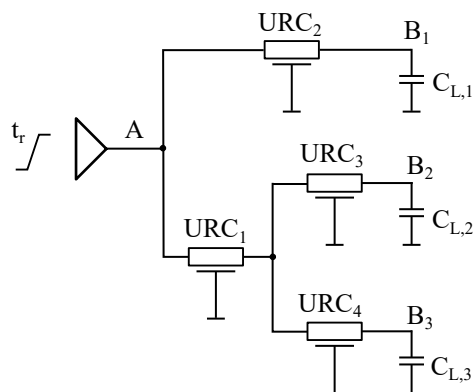


Figure 5.5: RC model of an interconnect.

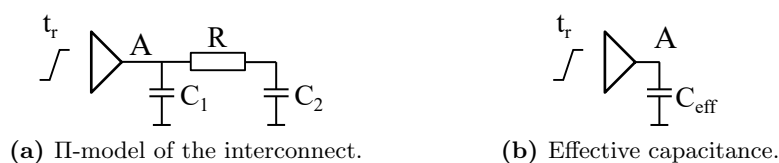


Figure 5.6: Driving point interconnect modeling.

An example of a net with with three endpoints is shown in Fig. 5.5. Each uniform RC (URC) element represents metal segments and $C_{L,1-3}$ are the load capacitances at endpoints B_{1-3} , respectively. Each of the URC elements has a specific resistance and capacitance value, which is a function of the total length of the segment and the per unit resistance and capacitance of the wire.

To capture the effect that the resistance has on the gate behavior and at waveform at the near end of the RC net, a Π -model (Fig. 5.6a) for the interconnect can be used. It consists of only two capacitors and a resistance, but is capturing the driving point impedance very well [OS89]. However, a problem of how to estimate the behavior of the driving gate with this load persists. Creating a lookup table consisting of various C_1 , C_2 , R and input transition values is impracticable, due to the multiple dimensions and hence very large size. One way to mitigate the problem is to use the "effective capacitance" method, proposed in [QPP94]. The main idea is to find a single capacitor C_{eff} (Fig. 5.6b) such that it approximates the behavior of the gate, as if it was loaded with a Π -load. The effective capacitance is calculated using an iterative approach, the details of which are explained in [QPP94]. Based on that load, the transition time can be estimated, using the existing lookup tables of the driver cell.

When the driving point transition time is found using the effective capacitance model, the transitions at the endpoints of the interconnect have to be estimated. As an alternative to transient simulation, this can be done by finding the transfer function of the interconnect, multiplying it with the driving point input in s-domain and converting back to time domain. However, this becomes increasingly complex for larger nets,

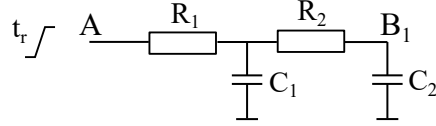


Figure 5.7: Reduced order model for one endpoint.

therefore model-order reduction techniques, such as [PR90], can be applied to reduce the complexity. In [XC11], a method that takes advantage of the topological structure of the RC interconnects is proposed to further reduce the model order topologically to a second order system. This method, called fast waveform evaluation (FWE), offers good accuracy and fast run-times and is therefore also used in this work.

Fig. 5.7 shows a reduced order model of the interconnect from Fig. 5.5, between points A and B₁. The transfer function of this simplified model may be written as

$$H(s) = \frac{1}{s^2 R_1 R_2 C_1 C_2 + s(R_1 R_2 + R_2 C_2 + R_1 C_1) + 1}. \quad (5.3)$$

The input waveform is assumed to be a step with finite rise time, and may be expressed in s-domain as the superposition of a rising ramp and a delayed falling ramp with the same slope:

$$V_i(s) = \frac{V_{DD}}{t_r} \left(\frac{1}{s^2} - \frac{e^{-t_r s}}{s^2} \right), \quad (5.4)$$

where t_r denotes the 0-100% transition time, and V_{DD} is the supply voltage.

Multiplying Equation 5.3 with 5.4 and converting back to time domain yields the following expression for the output waveform:

$$V_{B_1}(t) = \frac{V_i}{t_r} [at + b + c_a e^{p_1 t} + c_b e^{p_2 t}] - \frac{V_i}{t_r} u(t - t_r) [a(t - t_r) + b + c_a e^{p_1(t-t_r)} + c_b e^{p_2(t-t_r)}]. \quad (5.5)$$

By solving the equation for a few values of t , the slope of the output signal can be quickly estimated.

5.1.5 Handling the Logical Function of Gates

The boolean function of each gate is stored in ASCII text in the cell library. During the read-in of the data, this function should be parsed into a form suitable for machine interpretation. A convenient data structure for this task is the *binary expression tree*. A binary expression tree stores the boolean expression in a tree form, where each leaf stores an operand and each non-leaf node contains a single operator. Each subtree represents a subexpression and therefore each subtree should be evaluated before performing the operation at the common root node. In that sense, the tree represents the associativity of the operands [HR96]. To perform the evaluation of the equation stored in the binary

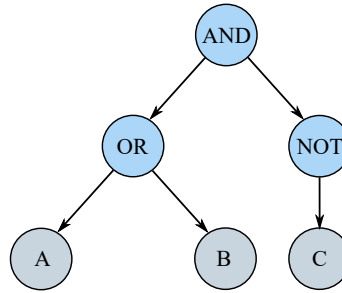


Figure 5.8: Exemplary binary expression tree for $Z = (A \text{ OR } B) \text{ AND } !C$.

expression tree, the tree is evaluated recursively, starting from the root and evaluating the left and the right subtrees. Fig. 5.8 shows an exemplary binary expression tree for the boolean function $Z = (A \text{ OR } B) \text{ AND } !C$.

During runtime, the function stored in the binary expression tree is used to resolve the unateness of each arc in the design. Starting from every net or primary input with defined static value, the values are propagated to the fanout gate. At each gate, a check is done if the output is resolved by the given input, using the binary expression tree. In the case of resolved output, the resolvent ('0' or '1') is propagated to the fanout gates.

5.1.6 Limitations

There is an intrinsic limitation of the SPNA method, related to the lookup table based approach. Similarly to STA, this method can only work with the set of already characterized cells. However, if a new cell is introduced to the flow, it can be easily and quickly characterized following the approach outlined in section 5.1.3 and added to the cell library. It is a one-time effort and the cell can then be used for all circuit analyses in the future.

5.1.7 SPNA Workflow

This section presents the workflow for phase noise analysis using the SPNA method. The description follows the chart in Fig. 5.9.

A netlist description of the circuit and information about its parasitics are necessary for the operation of the SPNA. The *design exchange format* (DEF) was selected as the format for netlist representation, because it is an open EDA file format that can be produced by most commercial tools. The *standard parasitic exchange format* (SPEF) was chosen for transferring the parasitics information, due to its simplicity and wide acceptance in industry. The SPEF file contains the extracted parasitics for every net in the layout in ASCII format. Together the DEF and SPEF files contain all the necessary data to fully describe the design.

The starting point for the analysis is a phase noise critical placed and routed design derived from rtl2gds activities in the digital toolchain.

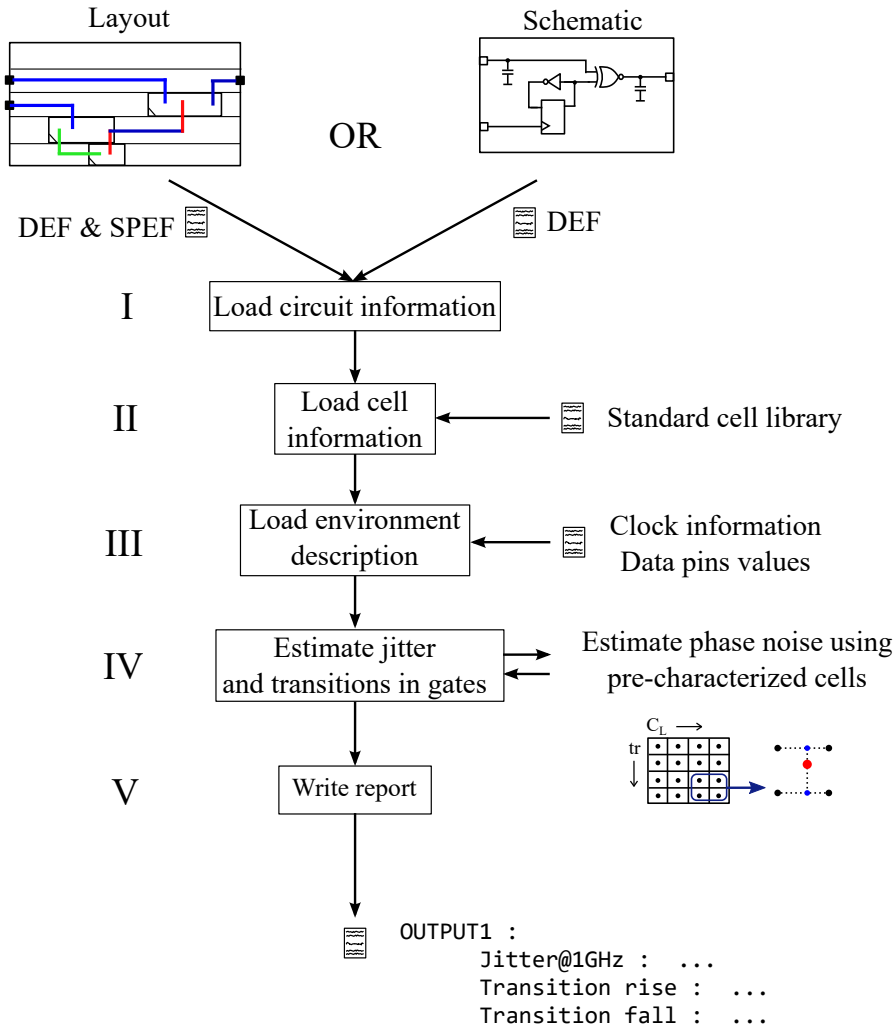


Figure 5.9: Workflow of SPNA.

I. First, the netlist of the schematic and its parasitics are written in the DEF and SPEF files, accordingly. Alternatively, the SPNA can also operate on a schematic with explicit parasitic components from an analog toolchain. In this case explicit capacitances need to be placed in the schematic to model the layout parasitics.

II. Additionally, the description of each cell is loaded, containing its function, transition times and noise performance, as discussed in Section 5.1.3.

III. Thereafter, the design constraints describing the clock and static signals are imported.

IV. Then, the noise estimation takes place, following the procedure described in Section 5.1.2.

V. After the SPNA completes the noise estimation, the output is stored in a report file. All the primary output ports, to which the clock signal propagates under the given

conditions, are listed with their corresponding jitter. The phase noise produced by the circuits is then calculated using the provided output frequency and equation (2.15).

SPNA interfaces with commercial EDA tools using automated routines, written in the scripting language used by the respective tool.

5.1.8 Experimental Results

The SPNA code was written in C++ and runs on an Intel(R) Xeon(R) CPU E5-2670, operating at 2.60 GHz. A standard cell library in a 28 nm CMOS process containing 30 cells is used. The cells have various logic functions and driver strengths from 1X to 8X. The smallest driver we used (1X) has $W_p = 0.35 \mu\text{m}$, $W_n = 0.27 \mu\text{m}$ and the sizes of the others scale accordingly. The simulations were performed using slow process corner, 0.99 V supply voltage, and temperature of 125° C.

To validate the SPNA method, four industrial reference designs implemented in an analog toolchain (using schematic editor) were considered. The results are summarized in Table 5.1. The circuits that were selected represent typical building blocks for clock distribution systems. Test case 1 contains clock gating and multiplexing logic, with 6 gates in the critical path. Test 2 features a clock divider with enabling function (5 gates). Test 3 is a clock buffer circuit with 10 buffers in a chain, and Test 4 is a derivative of Test 3, without the parasitic capacitances of the interconnects. The circuits operate at input frequency of 3.5 GHz, under a square wave stimulus.

The SPNA analysis and the reference analog simulation (SPICE) used explicit parasitic capacitances for the simulation. The error in all four cases is lower than 7.2% and the SPNA runtime is significantly shorter than analog simulation. The two-dimensional lookup tables describing the phase noise of the standard cells are predominantly concave, therefore the interpolation tends to overestimate the noise value, causing the observed error. The load-noise characteristic is very steep close to the origin (load below 2 fF), making the interpolation error larger for smaller loads. The result of this effect is visible in Test 4 in the table.

To verify the operability of the SPNA method in conjunction with an ASIC toolchain, a second set of test cases was created (Table 5.2). These generic test cases were selected to feature typical functional blocks in a clock distribution network and include physical designs (layouts). Test 1-3 contain clock dividers, enabling and gating logic with 4 to 6

Table 5.1: Noise estimation on industrial test cases in analog environment.

No	Phase noise (rad ² /Hz)		Runtime (sec)		Error (%)	Speed-up
	SPNA	SPICE	SPNA	SPICE		
1	1.68e-15	1.60e-15	0.03	1220	5.13	38437
2	8.13e-16	7.99e-16	0.035	1010	1.73	28857
3	9.69e-16	9.59e-16	0.04	950	1.03	22500
4	1.19e-16	1.11e-16	0.042	900	7.13	21428

Table 5.2: Noise estimation on test cases in digital toolchain.

No	Phase noise (rad ² /Hz)		Runtime (sec)		Error (%)	Speed-up
	SPNA	SPICE	SPNA	SPICE		
1	8.65e-16	8.33e-16	0.029	255	3.87	8793
2	1.73e-15	1.71e-15	0.035	310	1.17	9393
3	4.44e-16	4.26e-16	0.027	130	4.13	4814
4	1.43e-15	1.48e-15	0.042	460	-3.46	11219

gates on the critical path. Test 4 is a clock buffering network, spanning 1.2 mm. Here, the SPNA estimation is based on the capacitive parasitic extraction from the digital toolchain, whereas the analog simulator uses extracted RC view. Additionally to the error from linear interpolation, the error caused by neglecting the interconnect resistance is affecting the accuracy, especially noticeable in test 4. Another source of error is the misalignment between the extraction engines of the digital and analog toolchains. The accuracy of prediction is again very good, and the runtime is thousand-fold faster.

The proposed method provides phase noise analysis of logic circuits in digital toolchains, a functionality that was previously not available. This, combined with the automated place and route capabilities of the digital flow eliminates the need of setting up and performing analog simulations and handing over the task of layout generation to a layout designer. This allows for much faster design iterations in the hands of a single designer, profoundly reducing the total design time. Furthermore, usually gate sizing optimizers are applied iteratively until convergence, thus a circuit optimizer based on the proposed method will highly benefit from the offered analysis speed-up.

5.2 Buffer Insertion for Phase Noise Minimization

The RF clock signal needs to be distributed over large distances in the chip due to the required spatial separation of transmitter, receiver and oscillators. To ensure the signal quality of this high frequency signal, the transition time of the signal is improved by re-powering by logic gates. On the other hand, each logic gate contributes noise to the signal, which adds to the phase noise from the oscillator. Besides, the high switching activity in the distribution network creates power dissipation concerns for mobile devices.

The prevailing design practice is to construct clock distribution paths manually, using a custom design toolchain. The designer creates a buffer tree and selects positions for the buffers, based on best practices and experience. The parasitic capacitances are included in the schematic, estimating the wire length and the average capacitance per unit length. Then gate-size optimization is executed using commercial tools; these tools are usually simulation based and run an analog simulation to estimate the circuit's noise performance. This method gives a very accurate performance estimation, however it is time consuming. Additionally, the gate sizing optimizers cannot modify the count and

positions of the buffers in the tree, thus the quality of results (QoR) stands or falls by the selected initial netlist and buffer positions.

In this section we present a buffer insertion algorithm for phase noise minimization that operates within an ASIC toolchain. This way we benefit from the advantages of the digital implementation methods, such as exhaustive timing coverage and fast implementation speeds. The algorithm uses the lookup tables based approach, presented in Section 5.1, for estimating the phase noise produced by each buffer and analog simulations are not required. Furthermore, there is no need to provide an initial netlist for the buffer tree, making the QoR independent of the designer's experience.

5.2.1 Algorithmic Framework

From a conceptual point of view the problem of phase noise minimization in interconnect trees can be seen as a discrete buffer insertion problem. The discrete nature of this problem is related to the fact that the commercial automatic place and route tools operate on a predefined placement grid with a limited set of pre-characterized cells.

A related and well-studied problem is the problem of buffer insertion for delay optimization. Van Ginneken [vG90] suggests a bottom-up dynamic programming approach with quadratic time complexity. Lillis [LCL95] presented an extension to the algorithm for power minimization and accounts for signal transitions. We use a similar approach, where the delay information is replaced with phase noise, as explained below.

5.2.1.1 Description

The starting point is a net with a single source and various sinks/ports at different locations within a given floorplan (Fig. 5.10a). The goal is to buffer the net in a way that the distributed signal arrives at each sink with a phase noise not exceeding a predefined maximum value.

The interconnect is modeled as a tree, where the root node is the driver and the leaves are the endpoints of the net. The edges represent the wires, and the nodes are the possible buffer positions. The leaf vertices have capacitance $C(m)$ and maximum allowed phase noise $N(m)$, whereas the edges have capacitance $C_e(e)$.

The proposed algorithm uses dynamic programming to break down the problem hierarchically. As an entry point, a floorplan with placed macros in the place and route tool is required (Fig. 5.10a). The rectilinear Steiner tree is found by routing the net using the built-in algorithms of the place-and-route tool. Additional vertices are added if necessary to split the tree in similarly sized wire segments, in order to provide sufficient insertion points for buffers.

Fig. 5.10b shows an abstract tree representation of the floorplan from Fig. 5.10a. The algorithm processes the tree recursively, using post-order traversal (Fig. 5.11). Post-order traversal is effectively bottom-up traversal, where the node processing is done on returning from the call stack.

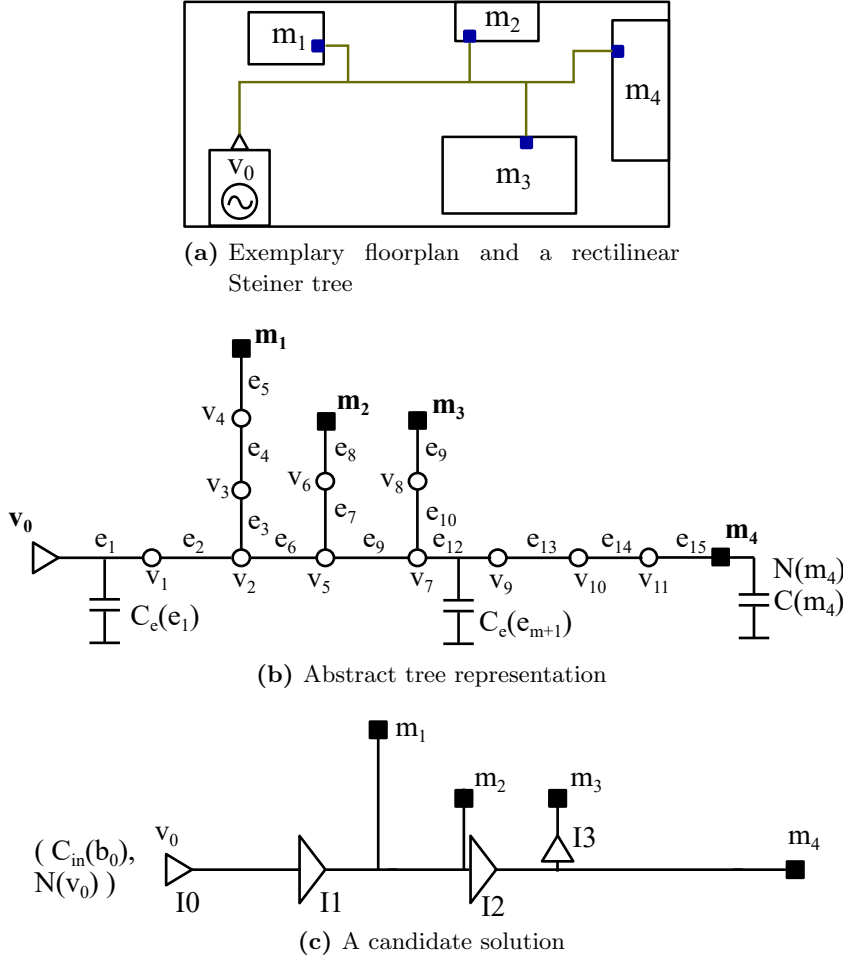


Figure 5.10: Buffer insertion problem.

For each vertex v_i a solution set S_i , holding all the possible solutions, is generated. A solution is a specific realization of the sub-tree originating at the given vertex, and has a pair of properties (c, n) , where c is the input capacitance at the root of the sub-tree, and n is the required maximum noise at this point.

Base case: For each sink vertex m_j , a solution set S_{m_j} consisting of a single solution with $c = C(m_j)$, and noise $n = N(m_j)$ is created (Fig. 5.11, line 3-4).

Add wire capacitance : At each vertex, the input capacitance of each solution in the successor's solution set is updated with the capacitance added by the wire, $c = c + l.c_u$, where l is the wire length and c_u is the wire capacitance per unit length (line 8-9).

Repeater insertion:

For every vertex v_i of the tree, a set S_i of candidate solutions is generated (line 8, 10-12). The solutions consist of all possible combinations of the solution set of the successor vertex with every buffer b in the buffer library B and the trivial case of no buffer. When

```

1 procedure BUFFER_INSERTION(node)
2   S ← {};
   Base case:
3   if type(node) == leaf then
4     S ← {(C(leaf), N(leaf))};
5     return S;
   Recursive call:
6   SL ← buffer_insertion(node->left);
7   SR ← buffer_insertion(node->right);
   Add wire capacitance:
8   for all s ∈ SL ∪ SR do
9     C(s) = C(s) + Cedge;
   Add buffers:
10  for all b ∈ B do
11    N = N(s) - N(b);
12    S ← S ∪ {(Cin(b), N)};
   Merge subtrees:
13  for all sL ∈ SL do
14    for all sR ∈ SR do
15      C = C(sL) + C(sR);
16      N = min(N(sL), N(sR));
17      S ← S ∪ {(C, N)};
18  remove_redundancy(S);
19  return S;

```

Figure 5.11: The buffer insertion algorithm.

a buffer is added, the new solutions have an input capacitance equal to the buffer's input capacitance $C_{in}(b)$, and the required noise is the noise of the successor node minus the noise produced by this buffer $n = n_{i+1} - n(b)$. In the case of no buffer added, the noise of the new solution is equal to the noise of the successor and the input capacitance is the (updated) successor's capacitance.

Branch merging: At tree vertices representing a branch point, repeaters are inserted for both the left and right child. Thereafter, the solution sets of the left and right child are merged by combining each solution from the left solution set with each solution from the right set (line 13-17). The input capacitance of the new solution is the sum of the capacitances of the two branches $c = c_L + c_R$, and the required noise is the more stringent of the two $n = \min(n_L, n_R)$.

Solution set pruning: The size of the solution set grows geometrically for every buffer position in the routing tree. To reduce the solution set size, Ginneken [vG90] proposed a pruning technique to recognize inferior solutions early. For any two solutions $s = (c, n)$

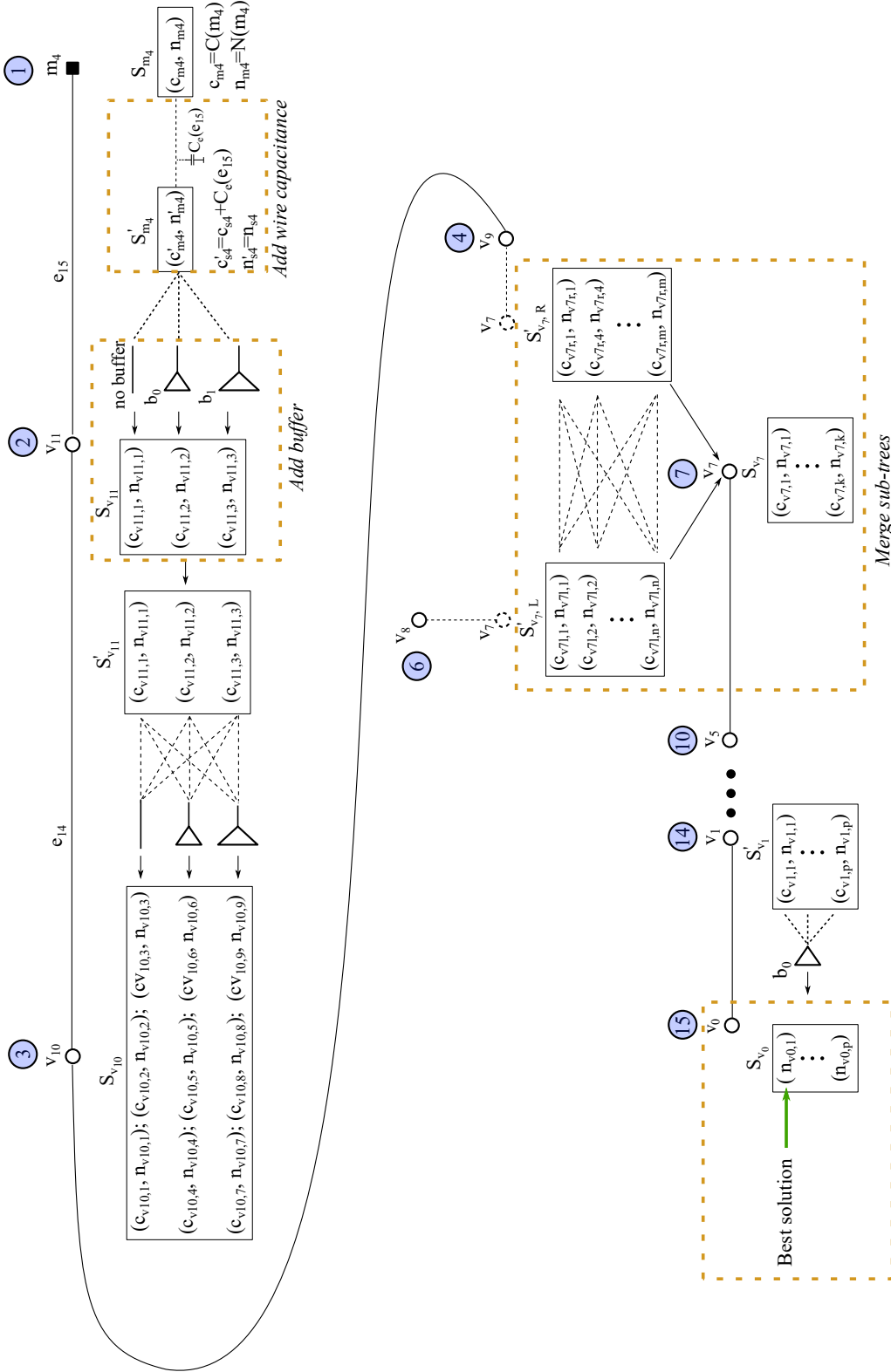


Figure 5.12: Visual representation of the buffer insertion algorithm.

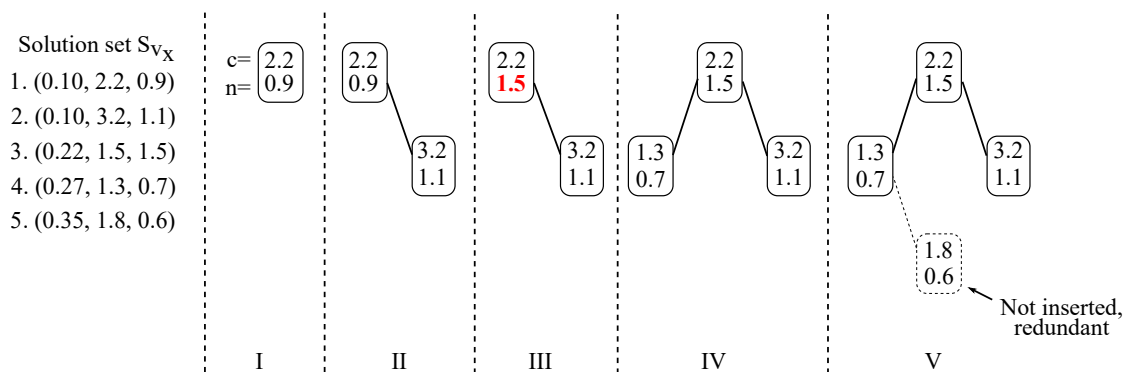


Figure 5.13: Augmented binary search tree for orthogonal range query.

and $s' = (c', n')$, if $c < c'$ and $n > n'$, then the solution s' is inferior and can be removed. This check can easily be implemented by sorting the solutions by increasing c value, and linear scan of the sorted solution set. For each solution s_k , the noise value n_k is compared to the largest value n_{max} found in all the previously visited solutions. As the solutions are visited in order of increasing c , if $n_k < n_{max}$, then the solution s_k is inferior to at least one previously visited solution and therefore can be pruned.

Fig. 5.12 offers a graphical representation of the above outlined operation of the algorithm for the exemplary tree from Fig. 5.10b. The numbers in blue circles show the order of visiting the vertices and comments on the operation are given in *italic*.

5.2.1.2 Power Minimization

Although phase noise minimization is the main task that needs to be achieved, this should not happen at the cost of huge power dissipation. To account for the dynamic power dissipation, a third property p for each solution is included. The dynamic power consumption can be approximated by $P_{dyn} = fCV_{DD}^2$, and since f and V_{DD} are constant for a given optimization problem, the sum of the capacitances C introduced by each stage of the logic gates can be used as a measure for the power.

There is now a 3-tuple of properties for each solution: (p, c, n) . The solution $s = (p, c, n)$ is inferior to another solution $s' = (p', c', n')$ if all three inequalities $p > p'$, $c > c'$, and $n < n'$ hold. Detecting such a situation requires higher computational effort as compared to the case of only two properties. Lillis [LCL95] suggested an orthogonal range query approach to tackle this task.

The main idea is the following: First, the solutions are ordered by increasing power p . Next, starting from the solution with smallest p value, an augmented binary search tree (BST) is created by ordering the solutions by c value. Each node of the BST is augmented with the smallest value of n found in the current node and in the left subtree. When a new node is inserted, the tree is recursively examined starting from the root and the capacitance and noise values of the new node, c_k and n_k correspondingly,

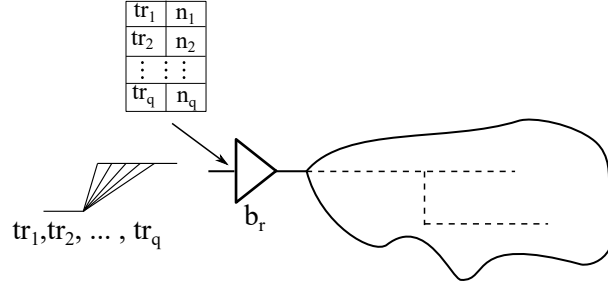


Figure 5.14: Propagating signal transition times upstream.

are compared with the capacitance and noise of the current node – c_t and n_t . During that process, there are four possible cases:

- 1) $c_k < c_t$ and $n_k < n_t \Rightarrow$ Explore left subtree ;
- 2) $c_k < c_t$ and $n_k > n_t \Rightarrow$ Non-redundant solution ;
- 3) $c_k > c_t$ and $n_k < n_t \Rightarrow$ Redundant solution ;
- 4) $c_k > c_t$ and $n_k > n_t \Rightarrow$ Explore right subtree.

In the case of subtree exploration, if the subtree does not exist, a node is created with the values of c_k and n_k .

Fig. 5.13 shows an example of an augmented BST for a given routing tree node v_x . The ordered solution set is on the left and the steps of creating the tree are enumerated with roman numerals. In step I, the root node is created using the entries of solution 1. Step II applies case 2, updating the best value of n . In step III, case 1 is applied and the value of the predecessor is also updated to contain the largest n value of the left subtree. Step IV applies case 1 twice, and in step 5, case 3 is applied, hence marking solution 5 redundant.

5.2.1.3 Signal Transition Times

The analysis in Section 2.3.2 demonstrated that the generated phase noise is related to the signal transitions. Therefore, the transition times of the signal should be considered. Instead of using a single phase noise value n for the solutions, a table containing pairs of (input transition, required noise) for q possible transition times is generated (Fig. 5.14). The upstream node uses this table to determine the required noise for a given transition using linear interpolation between the entries. As a side effect, the sinks can be easily constrained for maximum transition time by simply removing the entries in the table that are larger than the required transition time [MC99]. This can be used to estimate and bound the phase noise generated by the subsequent stages.

One disadvantage of this approach is that it further increases the complexity of the problem of recognizing redundant nodes. Consider two solutions s and s' , with $p > p'$

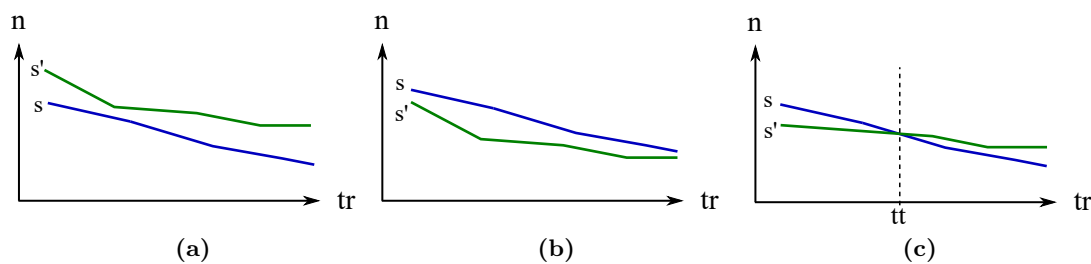


Figure 5.15: Handling transition times during pruning.

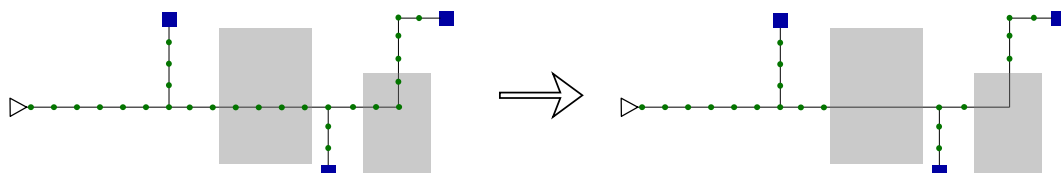


Figure 5.16: Handling placement blockages.

and $c > c'$. Now, while checking for redundancy using the augmented BST, the function cannot only check for one value of n vs n' , but it should compare all q entries in both tables. Fig. 5.15 shows the three possible outcomes of such a comparison. In sub-figure a, $n[i] < n'[i]$ for each $i \in 0..q$ and clearly the solution s is redundant. In sub-figure b, $n[i] > n'[i]$ for each $i \in 0..q$ and therefore the solution is not redundant. However, in the last example, there are entries, where $n > n'$ and also some where $n < n'$. During the construction of the solution set for the given vertex, the input transition time is not known and the solution cannot be marked as redundant. Hence, when considering the signal transition times, smaller amount of solutions can be identified as redundant, increasing the average-case complexity of the algorithm.

5.2.1.4 Miscellaneous

To enable the usage of cells with signal inversion, such as inverters, *the signal polarity* should be tracked to prevent unintentional signal inversion. Therefore, two disjoint solutions sets are used, each containing only the solutions requiring either an inverted or non-inverted input. During merging, only solutions with the same polarity can be combined. At the root, the negative solution set is ignored.

Placement blockages in the design are accounted for by deleting the insertion points that run over existing blockages. Thus, buffer insertion will not be considered at these locations and the resulting netlist complies with the placement blockages. Fig. 5.16 shows the handling of the placement blockages in a generic design – the endpoints are marked in blue, the possible buffer insertion points are shown on their physical locations in green, and the placement blockages are represented by gray rectangles.

5.2.1.5 Buffer Insertion Problem

With the above-outlined setup, the *buffer insertion problem* can be formulated as :

Given a routing tree $T = (V, E)$, root vertex v_0 , sink vertices (m_0 to m_n , sink capacitances $C(m_i)$, maximum noise at sinks $N(m_i)$, edge capacitances $C_e(e)$, and a buffer library B , find a solution that maximizes $N(v_0)$, subject to $P_{dyn} \leq P_{dyn,max}$.

5.2.1.6 Complexity analysis

For the case when the power consumption is not taken into account, there are at most $|B| \cdot (n + 1)$ non-redundant candidates for the tree [LCL95], where n is the number of vertices in the tree and $|B|$ is the size of the buffer library. The reason is that if a buffer is inserted at a node, the value of the input capacitance c is fixed, and there can be only one non-redundant solution – the one having the best noise value. For merging, all the combinations from the left and the right tree have to be combined, leading to an overall time complexity of $O(n^2 \cdot |B|^2)$. When the power consumption is considered, each solution has a 3-tuple of properties. In this case adding a buffer does not leave a single best solution for the subtree. In fact, in worst case all of the solutions might be non-redundant. Hence, the maximum number of candidates for a tree is $|B|^n$, leading to worst case non-polynomial time complexity of $O(|B|^n)$. In practice, the experiments demonstrate that the average runtime is considerably lower because of the pruning of the solution sets.

5.2.2 Buffer Insertion Workflow

To design a circuit with phase noise requirements, the designer needs to know at which offset frequencies it should be evaluated. In the case of the LO distribution path, the requirements for the phase noise are usually given for two offset frequencies from the carrier - one in-band and one far-off. The local oscillator itself produces $1/f^3$ and $1/f^2$ noise around the carrier, caused by up-converted white and flicker noise. The oscillator has a much higher influence on the noise in the in-band region than the distribution network.

Equations (2.7), (2.8) and (2.15) demonstrate that phase noise due to white and flicker noise both depend on the signal transition time and g_m , thus there are no contradicting requirements for reducing the noise contributions from these two noise types simultaneously. Consequently, for the synthesis of LO distribution paths, usually only the far-off noise – dominated by white noise – is a concern and only one far-off frequency f_{os} is passed to the algorithm, at which the noise should be minimized.

The algorithmic approach outlined in Section 5.2.1 is used within a methodology that has been developed based on a commercial ASIC design tool. A prerequisite is a

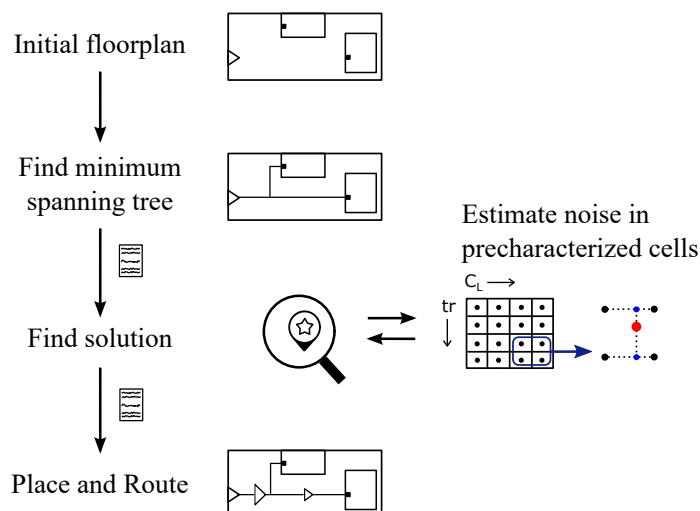


Figure 5.17: Workflow for buffer insertion.

floorplan with placed hard macros, unplaced standard cells and no signal routing. The nets with phase noise requirements are buffered and routed with high priority, before the automatic routing for the rest of the nets. This ensures maximum freedom for placing and routing of the cells on the noise critical paths.

As entry point the rectilinear Steiner tree should be computed, so that the paths are short and the parasitic capacitance is low. It is identified using the built-in routing algorithms of the ASIC tool. The tree structure is then extracted by traversing the tree and writing the topology into a text file. The rooted tree data structure needed for the algorithm is created based on the data stored in the file. Then the algorithm presented in Section 5.2.1 is executed to construct the routing tree. After the algorithm terminates, the solution with the proposed buffer structure is written into another text file. An automated procedure imports the buffer structure into the ASIC tool and modifies the netlist accordingly. As a last step, the nets are routed using the native commands and routing algorithms of the tool. The process is repeated for every net with a phase noise requirement in the design. Fig. 5.17 visualizes the outlined workflow.

To simplify the usage of the methodology, TCL procedures automating the workflow are written. They are used within the ASIC tool, similarly to the native tool commands. The user has to define the nets to be buffered, the maximum allowed phase noise and transition time for the sinks.

5.2.2.1 Estimation of Phase Noise

In the framework, presented in Sec 5.2.1, every time a buffer is inserted, the noise it introduces in the system has to be estimated. The buffer insertion algorithm operates with a limited set of cells, which allows the use of characterization tables, the approach used for the SPNA method in Section 5.1. This approach offers good accuracy, capturing

the higher-order effects affecting the cell performance without complex analysis and with low computational complexity. However, the interconnect resistance is not considered during runtime for the reasons outlined below.

The buffer insertion algorithm only runs on nets that have large parasitic capacitance, i.e. nets that span large distances. These nets are critical for the noise performance of the circuit and therefore extra wide wires in the low-ohmic global interconnect layers are used for the routing. This reduces the effect of the resistance on the signal transition times. Furthermore, even after pruning the sub-optimal solutions, the size of the solution sets created during runtime is very large. The "effective capacitance" method (Section 5.1.4) is iterative and needs multiple estimations before it finds a solution. Additionally, the transfer function of the downstream RC network has to be reevaluated for each solution at each vertex. Hence, regarding the interconnect resistance would slow down the creation of the table by the factor of the number of estimations. For large amount of solutions, this will have a heavy toll on runtime. Therefore, the effect of the parasitic resistance during the buffer insertion is neglected. Simulations show that ignoring the resistance introduces only a small error (<2%) in the predictions. The effects of the misestimation can be mitigated during the subsequent design-wide global sizing (See Section 5.3), where the interconnect resistance is considered by the phase noise estimation engine.

5.2.3 Results

The buffer insertion algorithm was implemented in C, and the scripts for interfacing with the commercial VLSI tool were written in TCL. The algorithm runs on an Intel(R) Xeon(R) CPU X5675, operating at 3.0 GHz. The test circuits are implemented in a bulk CMOS 28 nm process.

Four different buffer types with the same W_p/W_n ratio are used; the smallest (1X) having $W_p = 0.35\mu\text{m}$, $W_n = 0.27\mu\text{m}$ and $L = 0.03\mu\text{m}$, 2X, 4X, and the biggest buffer is 8X. Comparison between the predictions of the phase noise and analog simulations with extracted parasitic RC elements for 11 different floorplans are summarized in Table 5.3.

One source of error is the dependence of the phase noise of the cells on the driver impedance – simulations have shown that an increase in the driving impedance increases the noise generated by the characterized cell for the same input transition and load. This dependence was not investigated in Chapter 5.1.3 – better understanding and reducing this error is subject of future work. The effect of this error is particularly noticeable in test 2 in Table 5.3. In this case, the buffer tree has a long buffer chain (16 buffers) with multiple positions at which the driver impedance of a buffer is very different from the impedance of the driver used for the characterization. Tests 1 and 6 both also have very long buffer chains, however all inserted buffers are driven by drivers that have been used during characterization, hence this type of error is not present.

Table 5.3: Buffer insertion summary.

No	Phase noise				No of buffers	Error [dB]	Run-time [sec]
	Predicted		Simulated				
	dBc/Hz	rad ² /Hz	dBc/Hz	rad ² /Hz			
1	-133.05	1.21e-14	-132.19	1.48e-14	18	-0.86	41
2	-134.04	9.67e-15	-131.85	1.64e-14	16	-2.19	54
3	-139.01	3.08e-15	-138.58	3.40e-15	8	-0.43	49
4	-141.54	1.72e-15	-142.88	1.26e-15	8	1.33	11
5	-141.72	1.65e-15	-142.08	1.52e-15	10	0.36	73
6	-143.61	1.07e-15	-144.33	9.05e-16	36	0.72	24
7	-145.37	7.12e-16	-145.70	6.59e-16	4	0.33	33
8	-145.76	6.50e-16	-145.81	6.44e-16	4	0.04	85
9	-147.14	4.74e-16	-147.29	4.58e-16	6	0.15	40
10	-148.02	3.86e-16	-147.93	3.95e-16	4	-0.10	114
11	-149.71	2.62e-16	-149.17	2.96e-16	2	-0.54	12

Furthermore, the algorithm works with fixed unit capacitance c_u for the wires, whereas in a real-world design the capacitance of a wire fluctuates depending on the surroundings. Another contributor is the error from the linear interpolation. The effects of varying the wire segment length in the routing tree are demonstrated using a setup with total wire length of 6 mm and 8 sinks. Reducing the segment length increases the number of possible buffer positions, which allows for better QoR. However, wire segment lengths below $50\mu\text{m}$ do not bring any significant improvement of QoR, whereas the runtime quickly becomes prohibitive (Fig 5.18).

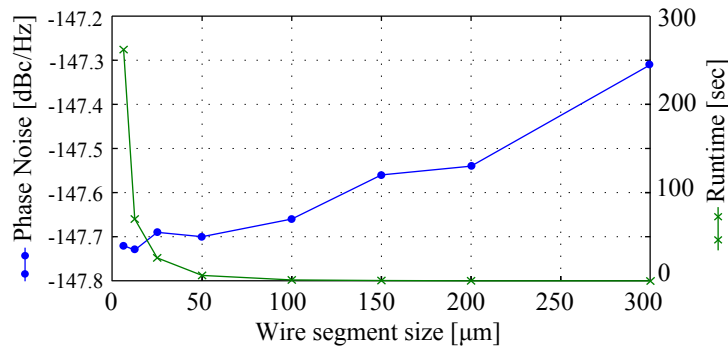
**Figure 5.18:** Phase noise and runtime for different wire segment lengths.

Table 5.4: Buffer insertion runtime.

Buffer library size			4
Wire segment size			50 μm
No	# of sinks	# of edges	Runtime [sec]
1	1	100	1.22
2	1	500	40.19
3	2	100	1.79
4	2	500	65.19
5	5	100	1.61
6	5	500	90.19
7	10	100	4.9
8	10	500	254.4

Table 5.4 shows a summary of multiple runs of the optimization flow with different numbers of sinks and edges, and a fixed wire segment length of 50 μm . The quoted runtime is only the algorithm runtime and does not include the time for placing and routing (usually well below 60 sec). This is a significant speed-up of the design time as compared to a typical manual design, where the estimated design time is in the range of man-days. The LO path in a typical transceiver system has fewer than 10 sinks, and 500 edges at 50 μm each guarantee that the presented methodology is applicable for even the biggest industrial designs.

5.3 Gate Sizing for Power Optimization Under Phase Noise Constraints

Gate sizing for delay minimization is a well-studied problem in VLSI design. In digital circuits the problem is usually defined as area or power minimization under delay constraints. Many works assumed a simple RC model and used Elmore delay calculation to approximate the delay. Early work, such as [Cir87], was based on transistor sizing and used Lagrangian relaxation to find the continuous solution to the non-linear programming problem. Others, assuming convexity applied different methods of solving the problem - geometric programming [FD03], convex programming [SRVK93].

Gate sizing, as opposed to transistor sizing, is a discrete problem and requires a different approach. Hu [HKH09] proposed an approach that uses the solution to the continuous transistor sizing problem to quickly narrow down the search space and consecutively finds the local optimum using dynamic programming. Coudert [Cou97] uses a lookup table delay model for high accuracy and proposes a relaxation method for solving the gate sizing problem.

In this work, we tackle the problem of power minimization under *phase noise constraints*. It is similar to the minimization under delay constraints in its discrete nature.

Empirical data shows that the problem is non-convex and non-linear and a closed form formula for the objective function does not exist. Usually the circuit size to be optimized is relatively small – a few dozen gates.

5.3.1 Problem Formulation

The problem of gate sizing in RF clock distribution networks can be defined as power minimization under phase noise and transition constraints. The noise budget is defined in the specification of the block, and the transition constraint guarantees that the applied clock signal propagates properly through the design.

For the estimation of phase noise, we use the lookup table based method, presented in Section 5.1, that offers fast runtime and good accuracy.

The circuit topology is modeled by the directed graph $T = (V, E)$, where the vertices represent the gates and the edges represent the nets. The gate sizes in the design are stored in the vector GT , where GT_i is the size of the indexed gate. We again use the sum of gate capacitances as a measure for the dynamic power, as discussed in Section 5.2.

Based on the above mentioned framework, the gate sizing problem can be formalized as:

$$\begin{aligned} \text{minimize} \quad & P_{dyn} = \sum_{i=1}^n C_{Gate}(GT_i), \\ \text{subject to} \quad & SPNA(T, GT) \leq S_{\phi, MAX}; \\ & TM(T, GT) \leq Tr_{max}; \\ & GT_i \leq GT_i^u, \forall i = \{1, \dots, n\}; \\ & GT_i \in \mathbb{N}_0, \forall i = \{1, \dots, n\}. \end{aligned}$$

C_{Gate} is the sum of all transistor gate capacitances of the indexed logic gate; n is the number of gates in the design; $SPNA()$ returns the maximal phase noise; $TM()$ returns the slowest transition found in the circuit; $S_{\phi, MAX}$ is the maximum allowed phase noise, introduced by the circuit, and Tr_{max} is the maximum signal transition time. GT_i^u is the upper bound for the size of each gate, defined by the availability of standard cells.

5.3.2 Gate Sizing

The problem formulated above is a discrete gate sizing problem, which was recognized as NP-hard in [Nin94]. NP-hard problems cannot be solved by exact methods in polynomial time, justifying the usage of approximate algorithms.

A global search strategy is required to find the global optimum, i.e. the best value within the entire search space. One of the most popular approaches for attacking a global optimization problem is the genetic algorithm (GA) metaheuristic. First presented by Holland [Hol75], GAs are population based strategies, built upon the principles of natural selection. They simulate evolution and rely on the processes of mutation, selection

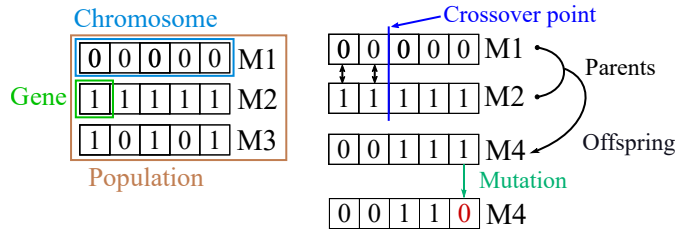


Figure 5.19: Basics of genetic algorithms.

and crossover. Fig. 5.19 summarizes the basic concepts of genetic algorithms. The initial population contains individuals described by variables (genes), organized in a set (chromosome). Matching this to the problem formulation in Section 5.3.1, a gene corresponds to the size of an individual gate, and the chromosome is the vector GS , storing the gate sizes of the whole design. Creating new individuals follows stochastic rules – a crossover point is selected at random and the genes of the two individuals are exchanged. To maintain diversity, random mutations are introduced by flipping bits in the chromosome of individuals with a given low probability. Only the best individuals in a population are allowed to generate an offspring (elitism).

GAs are simple, derivative-less, quickly locate the region of the global optimum, and work well with highly non-linear problems. They are good for global search and locating optimal regions, however they perform poorly in locating the local minimum. That is why they are often combined with a local improvement heuristic to form a hybrid genetic algorithm (HGA) [EmHNB06].

We selected a two-step approach, as suggested in [Hol75]. The genetic algorithm performs initial global search to find a good starting point for the local search, which in turn locates the local minimum.

Local search is a heuristic optimization approach that checks the neighborhood of the current solution to find an improved solution. The most basic local search heuristic is the hill-climbing method, a greedy method that always selects a move towards improvement and for this reason it is easily trapped into local extrema. The problem search space is highly non-linear and contains a lot of shallow local extrema, hence a strategy to circumvent this drawback is required. One of the most popular is *tabu search* [Glo86]. It introduces short term memory that forbids the search to go back to already visited points and avoids cycling.

The neighborhood $N(s)$ for a solution ($s \in S$) is the set of individual solutions that can be reached in the next step. The selected neighborhood for the local search is the set of all vectors from S at an Euclidean distance of one from the current solution. This means the solutions where only one gate in the netlist was resized one step up or down from its current size. This keeps the neighborhood small ($2 \cdot |N(s)|$) while the exploration of sizing of every gate is still guaranteed. Exploring multi-moves is not practical, as this increases the neighborhood size without contributing to the exploration; every composite move can be executed as a sequence of single step resizings. As the defined neighborhood is

reasonably small, it can be explored entirely and we use the *best improvement* as pivoting rule, i.e. we evaluate every neighbor and select the one with the best fitness.

5.3.2.1 Constrained Optimization

The most general and common approach for addressing constrained optimization problems is using a *penalty method*. In its essence, a penalty method applies a penalty function to the objective function to approximate the constrained problem to an unconstrained optimization problem. The use of penalty function "strongly encourages" the optimizer to select a point that is in the feasible region. A commonly used penalty function, that was selected for its simplicity and computational efficiency, is the sum of squared violations [JB02]:

$$P(x) = \sum_{i=1}^m \max\{0, g_i(x)\}^2, \quad (5.6)$$

where g_i is the amount by which the i -th constraint was violated. The augmented objective function then becomes

$$\theta(c, x) = f(x) + cP(x), \quad (5.7)$$

where $f(x)$ is the original objective function and c is the penalty parameter. For sufficiently large c , the penalty term yields a heavy cost for violations and the minimization of the augmented function produces a feasible solution. In the gate sizing problem the penalty function is the sum of the squared violations of the noise and transition constraints, $S_{\phi, \text{MAX}}$ and Tr_{max} , respectively. We use a static penalty method where the penalty parameter does not change over iterations to avoid complex tuning and the possible overfitting of the coefficients.

Fig. 5.20 describes in pseudocode the proposed gate sizing algorithm SPNA-GS, which minimizes the power consumption in two-step approach. The algorithm first executes global search using GA. The initial population set is a union of the original gate sizes, extracted from the netlist, and randomly generated solutions. GA drives the design into the feasible region and in a few consecutive steps finds an optimal region. After descending in a good optimal region, the genetic algorithm becomes ineffective, due to its stochastic nature. Hence a stop condition is defined as maximum number of iterations without an improvement above a given threshold. After the stop condition is reached, the GA terminates and the result is used as a starting point for the local tabu search that refines the search and finds a (near-)optimal solution.

Fig. 5.21 is a graphical demonstration of the operation of the SPNA-GS algorithm. On the power-noise graph, the dashed line is the theoretical optimal power-noise curve and the grayed zone is an infeasible region, as the noise exceeds the maximum value $S_{\phi, \text{max}}$. The green arrows show the steps done by the GA and the blue arrows the tabu search. First, GA finds the feasible region and in a few consecutive steps searches

5.3 Gate Sizing for Power Optimization Under Phase Noise Constraints

```

1 procedure SPNA-GS( T, GT,  $S_{\phi,max}$ ,  $Tr_{max}$ )
2   best  $\leftarrow$   $\infty$ ;
3   GA:
4   Pop  $\leftarrow$  RandomPop(size(Pop-1))  $\cup$  GT
5   while !StopCondition(MaxItGA, tholdGA) do
6     Crossover(Pop); Mutate(Pop);
7     Pop  $\leftarrow$  Elitism_selection(Pop);
8     if  $\theta$ (best(Pop)) <  $\theta$ (best) then
9       best  $\leftarrow$  best(Pop);
10  Tabu search:
11  while !StopCondition(MaxItTS, tholdTS) do
12    for all n  $\in$  N(s) do
13      if  $\theta$ (n) <  $\theta$ (best)  $\cup$  n  $\notin$  TabuList then
14        best  $\leftarrow$  n;
15      s  $\leftarrow$  best;
16      TabuList = TabuList  $\cup$  best ;
17  return best

```

Figure 5.20: The gate sizing algorithm SPNA-GS.

an optimal region. Then, the tabu search (blue arrows) refines the search and finds a (near-)optimal solution by making small local moves.

5.3.3 Results

The optimization method, presented in Section 5.3, cannot guarantee optimality of the result. Therefore the verification of the proposed method is done by comparing the produced results (GS) with the results from a commercial optimizer (COPT) operating within an analog simulator, using the same gate library as input. The power consumption and phase noise values for eight optimized circuits are shown in Table 5.5. All the designs

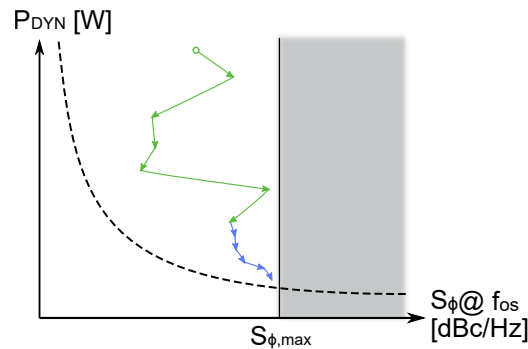


Figure 5.21: Example of the operation of the hybrid approach.

Table 5.5: SPNA based gate sizing summary.

Run	$S_{\phi, \max}$ [rad ² /Hz]	Power [μ W]		Noise [rad ² /Hz]		Runtime [sec]	
		GS	COPT	GS	COPT	GS	COPT
1	1.25e-14	353.5	370.1	1.29e-14	1.248e-14	5.1	15500
2	9.7e-15	259.9	275.5	1.03e-14	9.68e-15	7.3	14100
3	3.1e-15	205.3	209.5	3.12e-15	3.09e-15	4.2	10700
4	2.1e-16	100.9	96.8	2.08e-16	2.09e-16	4.4	13200
5	6.0e-16	105.9	104.5	5.92e-16	5.96e-16	1.3	11100
6	5.0e-16	78.6	79.1	5.16e-16	4.97e-16	4.0	13500
7	8.0e-16	95.5	99.3	7.73e-16	7.97e-16	2.4	15300
8	6.0e-16	272.8	276.0	5.93e-16	5.95e-16	1.4	12000

are with topologies containing typical building blocks for RF clock distribution systems, with gate count between 10 and 15 gates.

In all 8 cases, the GS provides good results, very close to the solution from the commercial optimizer. The first three test cases correspond to the first three test cases from Table 5.3. The gate sizing algorithm works with a denser library and more accurate noise prediction, hence the prediction error is lower and the results are closer to the target value, as compared to the buffer sizing algorithm alone. In cases 4 and 5, the power consumption is slightly higher for GS, due to a small positive error in the noise estimation, averting the algorithm from selecting a solution closer to the maximum allowed value. In cases 1, 2, 3 and 6 there is a slight overshoot of the noise value, also caused by prediction inaccuracy. Generally, the quality of results is limited by the prediction accuracy of the SPNA tool.

The runtime is in the range of a few seconds for the proposed gate sizing method, whereas for the commercial optimizer it is in the range of 3-4 hours. As both optimizers use iterative approach, evaluating multiple solutions to select the next best solution, the huge advantage in runtime, shown in Section 5.1.8 pays manifold here. Additionally, using the simulation based optimizer necessitates setting up the simulation by creating the optimization variables and defining ranges for them for each gate, which also adds a few hours of manual work.

5.4 Methodology for Phase Noise-Aware Design

This section presents a complete methodology for the automated design of RF clock distribution circuits using an ASIC toolchain. It is based on the methods for cell characterization for phase noise proposed in previous sections, the rapid phase noise analysis method SPNA, the buffer insertion algorithm and the gate sizing optimizer. It operates in conjunction with a commercial RTL-to-GDS design flow.

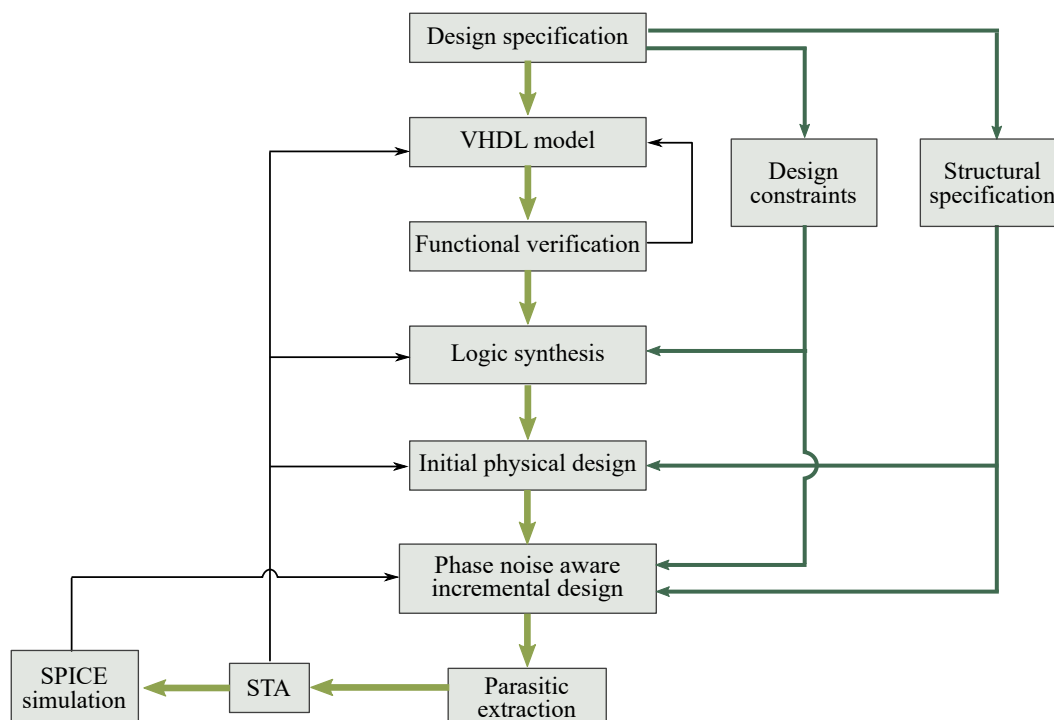


Figure 5.22: Design flow for automated phase noise-aware design.

Fig. 5.22 presents the design flow for automated phase noise-aware design. It enhances the ASIC design flow with phase noise-aware capabilities and with a sign-off SPICE simulation for final verification of the design.

Taking into consideration the design specification, a HDL description of the RF clock distribution module is created; it can be behavioral, structural or a mixed description. Based on that description, logic synthesis is performed using a commercial logic synthesis tool to produce a gate-level netlist. During that step, the phase noise sensitive parts of the circuit are left constraint-free. Thus, the synthesis tool only generates a RTL netlist, but does not insert buffers and does not size gates to meet timing constraints in this circuitry – its only task is to produce a netlist, logically equivalent to the HDL description. The netlist is then imported into the physical design tool, an initial placement and routing of the design are executed and the parasitics of the resulting layout are extracted. After the initial placement, the block undergoes incremental phase noise aware buffering and gate sizing, as will be discussed later. This incremental step modifies the netlist to meet the phase noise requirements of the circuit. After parasitic extraction, STA and SPICE simulations are run to verify that the produced circuit is performance-compliant. If the requirements are not met, refinements in the HDL model, constraints and settings of the ASIC flow are done and another design iteration is performed.

The workflow for phase noise aware-design is shown in Fig. 5.23. The task is to produce a phase noise compliant design, based on an initial layout. First, the nets that

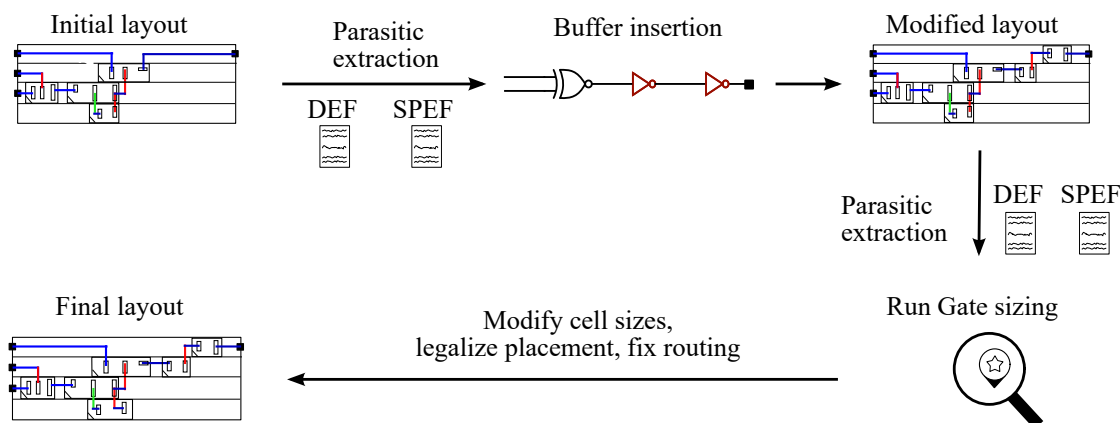


Figure 5.23: Workflow of the proposed phase noise-aware design methodology.

exceed a given transition time threshold are identified and the buffer insertion algorithm, described in Section 5.2 is run on them. The netlist is modified according to these results, and the new buffers are placed on the corresponding positions. The buffered nets are thereafter rerouted to reflect the changes and the parasitics are extracted anew. The gate sizing algorithm from Section 5.3 is carried out, using the provided phase noise budget and the clock constraints. It minimizes the dynamic power in the design, subject to given noise constraints, using the optimization procedure described in Section 5.3. The cells in the netlist are then changed according to the results from the gate sizer, their placement is legalized and the design is routed anew to connect the adjusted cells, producing the final layout.

5.4.1 Physical Design

In the proposed methodology, the physical design of the phase noise critical building blocks is executed using the automated ASIC design tools. For good quality of results, physical constraints should be applied to guide the placer and the routing tools in certain aspects, as discussed below.

The cells, which functionality is closely related to each other, should be constrained to be placed physically close. This decreases the parasitics and improves the circuit performance. The position of certain building blocks, such as multiplexing and clock gating usually should also be fixed by the designer, during the floorplanning to ensure best performance.

The interconnect resistance deteriorates the noise performance and should therefore be reduced. This is done by using the global interconnect layers for routing. Due to the increased layer thickness, their resistance is an order of magnitude lower than the local interconnects in the low metal layers.

In traditional timing-driven design, wire sizing is done to minimize the usage of routing resources, and to tune the trade-off of added capacitance and decreased resistance.

However, for phase noise aware design, there is a limited amount of critical nets, and they are given priority without jeopardizing the overall routability of the design.

Additionally, in modern technology nodes the wires are usually much taller than wide. Current aspect ratio of height to width lies around 2 and grows with shrinking nodes. Therefore, the peripheral (sidewall) capacitance dominates the fringing field and inter-layer capacitance [Int13]. Hence, the width of the wires can be increased to reduce the resistance, with only a minor increase in the capacitance. Increased spacing to neighboring routes can be used to reduce the peripheral capacitance of the wires.

5.4.2 Limitations

The proposed flow relies on set of pre-characterized cells from a library, hence transistor level sizing for fine tuning is not possible. The quality of the results depends on the available cells in the library: a denser library yields better results. However, a very dense library will increase the search space tremendously. There is also additional effort related for the design, layout and characterization of the library cells. Therefore, a balance between reasonable runtime and good quality of results should be found.

5.4.3 Advantages of the Flow

The proposed design flow benefits from using SPNA as phase noise estimation engine by significantly reducing the time for analysis. As gate sizing optimization is an iterative process, the benefit from this speedup manifolds. Additionally, the manual setup of the traditional analog simulation and the gate sizing optimization is now eliminated, reducing the required designer's manual work by hours.

An even bigger advantage is the integration of the flow within the digital toolchain. This way we benefit from the automated place and route capabilities of the flow. The absence of manual layout eliminates also the need of handover of the task to a layouter and the corresponding waiting times. Now a whole design cycle from RTL change to back-annotated simulations can be performed within a few hours and a couple of refinement iterations are possible within one working day of a single designer.

5.4.4 Results

To demonstrate the benefits of the proposed methodology, a test was conducted to compare the results of the buffer insertion algorithm and consecutive gate sizing with the results of manual design. The test case is an LO path from a real-world design, spanning approximately 3 mm with two endpoints. The simulation results summarized in Table 5.6 show that the outcome of the proposed methodology is comparable to the dedicated effort of an experienced designer.

Table 5.6: Comparison with manual design.

Target noise ≤ -158 dBc ($3.89\text{e-}17$)	Noise [rad^2/Hz]	Power [mW]
Proposed methodology	$3.75\text{e-}17$	1.29
Manual design	$3.41\text{e-}17$	1.42
Delta	$0.34\text{e-}17$	-0.13

6 Summary

Design automation has been a focus of extensive research since the dawn of the electronic industry. Nevertheless, the design of performance critical building blocks in the RF clocking system has traditionally been done using custom design techniques, to address the performance requirements for mobile transceivers, such as low power and phase noise. However, as reviewed in Chapter 3, custom design has lower test coverage and is time intensive, which corresponds to slower time to market and higher non-recurring costs of the end product.

The goal of this work was to enable the design of performance critical digital building blocks for RF circuits in the digital toolchain. For this purpose, a set of building blocks feasible for design automation were identified and methodologies were proposed to overcome the limitations of the existing tools.

Chapter 4 presented methods for automated design of a high-speed multi-modulus divider (MMD) and a time-to-digital converter (TDC), used within RF clock synthesizers. Adequate architectures were selected to satisfy the low power and high speed requirements of the blocks, and timing and physical constraints of these blocks were developed. The designs were implemented in a 28 nm bulk CMOS technology within a commercial ASIC toolchain. Pre-silicon verification of the MMD showed a maximum operating frequency of 4.3 GHz under worst conditions in 28 nm bulk CMOS technology, limited only by the available standard cells. The implemented TDC achieved resolution of 8.7 ps, $DNL_{MAX}=0.8$ LSB and $DNL_{RMS}=0.27$ LSB. The design architecture and methodology overcome the limitations of the tools and the standard cell library and the achieved performance figures satisfy the requirements for modern cellular transceivers.

A methodology for CMOS circuit design under phase noise constraints was presented in Chapter 5. It operates in conjunction with a commercial digital design flow and consists of a fast method for phase noise analysis, an algorithm for buffering of interconnects under noise constraints and a gate sizing algorithm for phase noise minimization under power constraints. The methodology is based on standard cells and is therefore fully compatible with the commercial physical design toolchain. The interfacing with the design tools was automated with scripts in this work. The method for phase noise analysis is based on lookup tables and it demonstrated high accuracy and fast runtimes. The deterministic buffer insertion algorithm is invoked on long nets with phase noise requirements in the design. It provides optimal results for the given cell library and offers substantial speed-up as compared to custom design and the observed run times are low even for very large topologies. The gate sizing algorithm uses the method for

6 Summary

phase noise estimation and searches for gate sizes that minimize the power consumption under phase noise requirements.

The proposed methodologies allow the design of performance critical RF building blocks directly in the digital toolchain. This way the designers benefit from high design speed, good reproducibility of the results and reduced manual efforts. Furthermore, as the descriptions of the blocks are HDL based, they are technology independent. Enabling the creation of purpose-designed building blocks, optimized for a specific use case fosters the reuse of HDL code and constraints, and not the reuse of GDS.

In the future, new cellular standards with higher level of integration, and increased usage of carrier aggregation necessitating multiple antennas with frequency division multiplexing, are expected. Thus, an always increasing number of synthesizers will be necessary on chip to support all required frequency bands. As each synthesizer has different frequency range and noise requirements, purpose-designed building blocks are necessary to minimize the power consumption and/or area. Furthermore, the growing number of oscillators require more complex RF clock distribution networks and each design needs to be tailored to the given floorplan. The proposed methodologies can therefore improve the design of performance-compliant systems on chip in design efforts, time to market and power consumption.

Bibliography

- [3gp11] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) Radio Transmission and Reception. Technical Report 3GPP TS 36.101 Version 11.8.0 Release 11, 2011.
- [5G 17] 5G Americas and Ovum. Number of Global Mobile/Wireless Subscriptions by Technology From 2010 to 2017, June 2017.
- [Abi04] A. Abidi. RF CMOS Comes of Age. *IEEE Journal of Solid-State Circuits*, 39(4):549–561, Apr 2004.
- [Abi06] A. Abidi. Phase Noise and Jitter in CMOS Ring Oscillators. *IEEE Journal of Solid-State Circuits*, 41(8):1803–1816, Aug 2006.
- [AGL⁺95] A. Aharon, D. Goodman, M. Levinger, Y. Lichtenstein, Y. Malka, C. Metzger, M. Molcho, and G. Shurek. Test Program Generation for Functional Verification of PowerPC Processors in IBM. In *32nd Design Automation Conference*, pages 279–285, June 1995.
- [AHGR11] S. Abdel-Hafeez and A. Gordon-Ross. A Gigahertz Digital CMOS Divide-by-N Frequency Divider Based on a State Look-Ahead Structure. *Circuits, Systems, and Signal Processing*, 30(6):1549–1572, Dec 2011.
- [BC09] J. Bhasker and R. Chadha. *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer, New York, NY, USA, 2009.
- [BCJP03] K. Bernstein, C. Chuang, R. Joshi, and R. Puri. Design and CAD Challenges in Sub-90 nm CMOS Technologies. In *ICCAD-2003. International Conference on Computer Aided Design*, pages 129–136, Nov 2003.
- [Bes07] R. Best. *Phase Locked Loops: Design, Simulation, and Applications*. McGraw-Hill Professional, 2007.
- [CH15] R. Chen and H. Hashemi. Reconfigurable SDR Receiver with Enhanced Front-End Frequency Selectivity Suitable for Intra-Band and Inter-Band Carrier Aggregation. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pages 1–3, Feb 2015.
- [Cir87] M. A. Cirit. Transistor Sizing in CMOS Circuits. In *Proceedings of the 24th ACM/IEEE Design Automation Conference, DAC '87*, pages 121–124, New York, NY, USA, July 1987. ACM.

Bibliography

- [CK04] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC and Custom - Tools and Techniques for High-Performance ASIC Design*. Kluwer Academic Publishers, 2004.
- [CL03] C. Chung and C. Lee. an All-Digital Phase-Locked Loop for High-Speed Clock Generation. *IEEE Journal of Solid-State Circuits*, 38(2):347–351, Feb 2003.
- [Cou97] O. Coudert. Gate Sizing for Constrained Delay/Power/Area Optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):465–472, Dec 1997.
- [Cou00] L. Couch. *Digital and Analog Communication Systems*. Prentice Hall, 2000.
- [CS91] Z. Chang and W. Sansen. *Low-Noise Wide-Band Amplifiers in Bipolar and CMOS Technologies*. Kluwer Academic Publishers, 1991.
- [EmHNB06] T. El-mihoub, A. Hopgood, L. Nolle, and A. Battersby. Hybrid Genetic Algorithms: A Review. *Engineering Letters*, 13, August 2006.
- [ETS] ETSI. LTE; Feasibility Study for Further Advancements for E-UTRA (LTE-Advanced). Technical Report 3GPP TR 36.912, Version 11.0.0, Release 11.
- [FD03] J. P. Fishburn and A. E. Dunlop. *TILoS: A Posynomial Programming Approach to Transistor Sizing*, pages 295–302. Springer US, Boston, MA, 2003.
- [Gli11] T. Glisson. *Sinusoids, Phasors, and Impedance*, pages 383–477. Springer Netherlands, Dordrecht, 2011.
- [Glo86] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Comput. Oper. Res.*, 13(5):533–549, May 1986.
- [Gol08] K. Golshan. *Physical Design Essentials: An ASIC Design Implementation Perspective*. Springer Science & Business Media, 2008.
- [Gu05] Q. Gu. *RF System Design of Transceivers for Wireless Communications*. Springer, 2005.
- [Haj01] A. Hajimiri. Noise in Phase-Locked Loops. In *2001 Southwest Symposium on Mixed-Signal Design*, pages 1–6, Feb 2001.
- [Hen10] S. Henzler. *Time-To-Digital Converters*. Springer, 2010.

- [HKF⁺06] X. He, W. Kong, T. Firestone, R. Newcomb, and M. Peckerar. Phase Noise Optimization of A Symmetric CMOS LC VCO. In *2006 IEEE International Symposium on Industrial Electronics*, volume 4, pages 2820–2823, July 2006.
- [HKH09] S. Hu, M. Ketkar, and J. Hu. Gate Sizing for Cell-Library-Based Designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(6):818–825, June 2009.
- [HL98a] A. Hajimiri and T. H. Lee. A General Theory of Phase Noise in Electrical Oscillators. *IEEE Journal of Solid-State Circuits*, 33(2):179–194, Feb 1998.
- [HL98b] A. Hajimiri and T. H. Lee. Phase Noise in CMOS Differential LC Oscillators. In *1998 Symposium on VLSI Circuits. Digest of Technical Papers*, pages 48–51, June 1998.
- [Hol75] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan, 1975.
- [HR96] M.R. Headington and D.D. Riley. *Data Abstraction and Structures Using C++*. Jones and Bartlett Publishers, 1996.
- [HR15] S. C. Hwu and B. Razavi. An RF Receiver for Intra-Band Carrier Aggregation. *IEEE Journal of Solid-State Circuits*, 50(4):946–961, April 2015.
- [HRA05] E. Hegazi, J. Rael, and A. Abidi. *Basics of LC Oscillators*, pages 1–9. Springer US, Boston, MA, 2005.
- [HRW12] Y. Han, W. Rhee, and Z. Wang. Design and Analysis of a Robust All-Digital Clock Generation System with a DLL-Based TDC. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 3152–3156, April 2012.
- [Int13] International Technology Roadmap for Semiconductors. Interconnect Chapter, 2013.
- [JB02] P.. Jensen and J. Bard. *Operations Research Models and Methods*. Wiley, 2002.
- [JFF⁺12] M. Jung, J. Fuhrmann, A. Ferizi, et al. A 10 GHz Low-Power Multi-Modulus Frequency Divider Using Extended True Single-Phase Clock Logic. In *Microwave Integrated Circuits Conf., 2012 7th European*, pages 508 – 511, Oct 2012.

Bibliography

- [KHO⁺09] V. Kratyuk, P. K. Hanumolu, K. Ok, U. K. Moon, and K. Mayaram. A Digital PLL With a Stochastic Time-to-Digital Converter. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(8):1612–1621, Aug 2009.
- [KLMH11] A. B. Kahng, J. Lienig, I. Markov, and J. Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2011.
- [KM04] R. Krüger and H. Mellein. *UMTS: Introduction and Measurement*. Rohde & Schwarz, 2004.
- [LAG⁺03] T. Ludwig, I. Aller, V. Gernhoefer, J. Keinert, E. Nowak, R. V. Joshi, A. Mueller, and S. Tomaschko. FinFET Technology For Future Microprocessors. In *2003 IEEE International Conference on SOI*, pages 33–34, Sept 2003.
- [LB09] R. Ludwig and G. Bogdanov. *RF Circuit Design: Theory and Applications*. Pearson International Edition. Prentice-Hall, 2009.
- [LCL95] J. Lillis, C.-K. Cheng, and T. Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. In *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, pages 437–447, Nov 1995.
- [Lee03] T. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits, Second Edition*. Cambridge University Press, 2003.
- [LMUZ02] O. Lachish, E. Marcus, S. Ur, and A. Ziv. Hole Analysis for Functional Coverage Data. In *Proceedings 2002 Design Automation Conference*, pages 807–812, June 2002.
- [MC99] N. Menezes and C. Chen. Spec-Based Repeater Insertion and Wire Sizing for On-Chip Interconnect. In *VLSI Design, 1999. Proceedings. Twelfth International Conference On*, pages 476–482, Jan 1999.
- [Meh01] J. L. Mehta. Transceiver Architectures for Wireless ICs. *Signal Magazine*, page 76–96, Feb 2001.
- [MHS16a] D. Martev, S. Hampel, and U. Schlichtmann. Fully Synthesized Time-To-Digital Converter for Cellular Transceivers. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–5, June 2016.
- [MHS16b] D. Martev, S. Hampel, and U. Schlichtmann. Synthesis-Based Methodology for High-Speed Multi-Modulus Divider. In *2016 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1–4, June 2016.

- [MHS17a] D. Martev, S. Hampel, and U. Schlichtmann. A Method for Phase Noise Analysis of RF Circuits. In *Proceedings of the on Great Lakes Symposium on VLSI 2017, GLSVLSI '17*, pages 227–231, New York, NY, USA, 2017. ACM.
- [MHS17b] D. Martev, S. Hampel, and U. Schlichtmann. Methodology for Automated Phase Noise Minimization in RF Circuit Interconnect Trees. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.
- [MHS18] D. Martev, S. Hampel, and U. Schlichtmann. Automated Phase Noise-Aware Design of RF Clock Distribution Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018.
- [MK98] K. Maatta and J. Kostamovaara. A High-Precision Time-To-Digital Converter for Pulsed Time-Of-Flight Laser Radar Applications. *IEEE Transactions on Instrumentation and Measurement*, 47(2):521–536, Apr 1998.
- [Moh12] F. M. Mohammadi, A. Ghannouchi. *RF Transceiver Design for MIMO Wireless Communications*. Springer, 2012.
- [Moo65] G. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8), Apr 1965.
- [Nin94] W. Ning. Strongly NP-hard Discrete Gate-Sizing Problems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(8):1045–1051, Aug 1994.
- [OS89] P. R. O’Brien and T. L. Savarino. Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation. In *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 512–515, Nov 1989.
- [OWY83] A.V. Oppenheim, A.S. Willsky, and I.T. Young. *Signals and Systems*. Prentice-Hall signal processing series. Prentice-Hall, 1983.
- [PK00] J. Phillips and K. Kundert. Noise in Mixers, Oscillators, Samplers, and Logic: An Introduction to Cyclostationary Noise. In *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference*, pages 431–438, May 2000.
- [PKK00] C. Park, O. Kim, and B. Kim. A 1.8-GHz Self-Calibrated Phase-Locked Loop with Precise I/Q Matching. In *2000 Symposium on VLSI Circuits. Digest of Technical Papers*, pages 242–243, June 2000.

Bibliography

- [PR90] L. T. Pillage and R. A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(4):352–366, Apr 1990.
- [QPP94] J. Qian, S. Pullela, and L. Pillage. Modeling the "Effective capacitance" for the RC interconnect of CMOS gates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1526–1535, Dec 1994.
- [Raz96a] B. Razavi. Challenges in Portable RF Transceiver Design. *IEEE Circuits and Devices Magazine*, 12(5):12–25, Sep 1996.
- [Raz96b] B. Razavi. *Design of Monolithic Phase-Locked Loops and Clock Recovery Circuits: A Tutorial*, pages 1–39. Wiley-IEEE Press, 1996.
- [Raz96c] B. Razavi. A Study of Phase Noise in CMOS Oscillators. *IEEE Journal of Solid-State Circuits*, 31(3):331–343, Mar 1996.
- [Raz98] B. Razavi. *RF Microelectronics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.
- [Raz03] B. Razavi. RF CMOS Transceivers for Cellular Telephony. *IEEE Communications Magazine*, 41(8):144–149, Aug 2003.
- [Rok98] A. Rokita. Multiple Oscillator Frequency Synthesizer with Common Phase Lock Loop. In *12th International Conference on Microwaves and Radar. MIKON-98. Conference Proceedings*, volume 3, pages 833–837 vol.3, May 1998.
- [RSR⁺09] M. Ray, W. Souder, M. Ratcliff, et al. A 13 GHz Low Power Multi-Modulus Divider Implemented in 0.13 um SiGe Technology. In *Silicon Monolithic ICs in RF Systems, 2009. IEEE Topical Meeting on*, pages 1–4, Jan 2009.
- [Rya05] P. Ryan. Treating the Wireless Spectrum as a Natural Resource. *Environmental Law Reporter*, page 10620, Sept 2005.
- [Sep70] R.B. Sepe. Frequency multiplier and frequency waveform generator, December 29 1970. US Patent 3,551,826.
- [Sha48] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.
- [SHLB03] R. Staszewski, C. Hung, D. Leipold, and P. T. Balsara. A First Multigigahertz Digitally Controlled Oscillator for Wireless Applications. *IEEE Transactions on Microwave Theory and Techniques*, 51(11):2154–2164, Nov 2003.

- [SJK13] M. Song, I. Jung, S. Pamarti, and C. Kim. A 2.4 GHz 0.1-Freq-Bandwidth All-Digital Phase-Locked Loop With Delay-Cell-Less TDC. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(12):3145–3151, Dec 2013.
- [Slo01] A. Slobodová. Formal Verification Methods for Industrial Hardware Design. In *Proceedings of the 28th Conference on Current Trends in Theory and Practice of Informatics Piestany: Theory and Practice of Informatics*, SOFSEM '01, pages 116–135, London, UK, UK, 2001. Springer-Verlag.
- [SP09] M. Z. Straayer and M. H. Perrott. A Multi-Path Gated Ring Oscillator TDC With First-Order Noise Shaping. *IEEE Journal of Solid-State Circuits*, 44(4):1089–1098, April 2009.
- [SRVK93] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. Kang. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(11):1621–1634, Nov 1993.
- [Sta06] R. Staszewski. *All-Digital Frequency Synthesizer in Deep-Submicron CMOS*. John Wiley & Sons, 2006.
- [SW⁺05] R. Staszewski, J.L. Wallberg, et al. All-Digital PLL and Transmitter for Mobile Phones. *IEEE J. Solid-State Circuits*, 40(12):2469–2482, Dec 2005.
- [Tal12] D. Talbot. *Frequency Acquisition Techniques for Phase Locked Loops*. Wiley-IEEE Press, 2012.
- [VBSC12] K. Vengattaramane, J. Borremans, M. Steyaert, and J. Craninckx. Spatially Oversampled TDC with Digital Resolution Enhancement. *Analog Integrated Circuits and Signal Processing*, 70(3):311–322, Mar 2012.
- [VDTKVR03] J. Van Der Tang, D. Kasperkovitz, and A. Van Roermund. *High-Frequency Oscillator Design for Integrated Transceivers*, volume 748. Springer Science & Business Media, 2003.
- [VFL⁺00] C.S. Vaucher, I. Ferencic, Locher, et al. A Family of Low-Power Truly Modular Programmable Dividers in Standard 0.35- μ m CMOS Technology. *Solid-State Circuits, IEEE Journal of*, 35(7):1039–1045, July 2000.
- [vG90] L.P.P.P. van Ginneken. Buffer Placement in Distributed RC-Tree Networks for Minimal Elmore Delay. In *Circuits and Systems, 1990., IEEE International Symposium on*, pages 865–868 vol.2, May 1990.

Bibliography

- [VLB⁺06] U. Vollenbruch, Yue Liu, T. Bauernfeind, et al. Requirements for Time-To-Digital Converters in the Context of DPLL Based Frequency Synthesis and GSM Modulation. In *Microwave Symp. Dig., IEEE MTT-S Int.*, pages 1817–1820, June 2006.
- [WS63] F. Wanlass and C. Sah. Nanowatt Logic Using Field-Effect Metal-Oxide Semiconductor Triodes. In *1963 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, volume VI, pages 32–33, Feb 1963.
- [WSWW10] C. T. Wu, W. C. Shen, W. Wang, and A. Y. Wu. A Two-Cycle Lock-In Time ADPLL Design Based on a Frequency Estimation Algorithm. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(6):430–434, June 2010.
- [XC11] J. Xu and M. H. Chowdhury. Fast Waveform Estimation (FWE) for Timing Analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(5):846–856, May 2011.
- [Xiu08] L. Xiu. *VLSI Circuit Design Methodology Demystified: A Conceptual Taxonomy*. Wiley-IEEE Presss, 2008.
- [YW11] P. Youngmin and D.D. Wentzloff. A Cyclic Vernier TDC for ADPLLs Synthesized From a Standard Cell Library. *IEEE Trans. Circuits Syst. I, Reg. Papers*, July 2011.
- [Zha15] K.Q.T. Zhang. *Wireless Communications: Principles, Theory and Methodology*. Wiley, 2015.