**University of Alberta**

# HIGH-SPEED ALIAS-LOCKED LOOP FREQUENCY SYNTHESIS

by

**Leendert Jan van den Berg** ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

in

**Computer, Microelectronic Devices, Circuits and Systems**

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2008

*Your file Votre référence*
*ISBN:* 978-0-494-55121-9
*Our file Notre référence*
*ISBN:* 978-0-494-55121-9

*We are at the very beginning of time for the human race. It is not unreasonable that we grapple with problems. But there are tens of thousands of years in the future. Our responsibility is to do what we can, learn what we can, improve the solutions, and pass them on.*

– Richard Feynman

To Mom and Dad,
and all ancestors who came before.

# Abstract

This work presents a divider-less frequency synthesis architecture called an alias-locked loop. The division in the feedback path of the phase-locked loop (PLL) is modified by the addition of a latch that samples the local oscillator at a much lower frequency. Thus, rather than just performing frequency division, the local oscillator is sub-sampled, generating a lower-frequency aliased signal, which is utilized as in any standard PLL. Using a sampling latch rather than a divider significantly loosens the constraints placed on the feedback circuits. With a non-linear simulation model that describes the time-domain behaviour of our architecture, stable modes of operation with bounded orbits in phase-space are demonstrated. Simulations of an implementation in 90-nm CMOS technology confirm the feasibility of the aliasing approach. In noise and offset free circuit simulations of sampling latches using a 90-nm CMOS process, we demonstrate successful sub-sampling of a 230 GHz signal, which is greater than $f_{max}$.

# Acknowledgements

The research presented here has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), CMC Microsystems and the University of Alberta.

First of all, I would like thank my supervisor, Dr. Duncan Elliott, for his advice, guidance, and continuous support during my research. His unusually astute insights into the engineering of various electronic systems and circuits have shown me how to look at design challenges from a new perspective. Additionally, I appreciate the freedom granted to me during my studies, which have enriched my experience by allowing me to be involved in various projects not related to my thesis topic.

Additionally, I would like to thank the professors that have shaped the atmosphere of the lab into a stimulating and welcoming environment with ample opportunities for collaboration: Dr. Bruce Cockburn, Dr. Vincent Gaudet, and Dr. Stephen Bates.

I also wish to thank all the individuals that have been part of the VLSI and HDCD labs for their friendship and camaraderie, including: John Koob, Tyler Brandon, Ramkrishna Swamy, Maziyar Khorasani, Dr. Amirhossein Alimohammad, Saeed Fouladi Fard, Wesam Al-Haddad, Anthony Ho, and Russell Dodd.

Finally, I wish to thank my parents and sibblings for their unconditional love and support independent of my achievements and choices in life.

# Table of Contents

# List of Tables

# List of Figures

# Nomenclature

## List of Acronyms

| | |
|---|---|
| ALL | Alias-Locked Loop, page 1 |
| BBPLL | Bang-Bang Phase-Locked Loop, page 72 |
| CMOS | Complementary Metal-Oxide Semiconductor, page 2 |
| DAC | Digital-to-Analog Converter, page 11 |
| DAS | Direct Analog Synthesis, page 11 |
| DC | Direct Current, page 39 |
| DDS | Direct Digital Synthesis, page 11 |
| DLL | Delay-Locked Loop, page 5 |
| DPTPL | Differential Pass-Transistor Pulsed Latch, page 51 |
| DRO | Dielectric Resonator Oscillator, page 26 |
| DSP | Digital Signal Processing, page 22 |
| GaAs | Gallium Arsenide, page 2 |
| GP | General Purpose, page 45 |
| HEMT | High Electron Mobility Transistor, page 38 |
| IC | Integrated Circuit, page 7 |
| InP | Indium Phosphide, page 2 |
| LCM | Least Common Multiple, page 86 |
| LPF | Low-Pass Filter, page 73 |
| MDLL | Multiplying Delay-Locked Loop, page 15 |
| MMIC | Monolithic Microwave Integrated Circuit, page 26 |

| | |
|---|---|
| MSB | Most Significant Bit, page 64 |
| PFD | Phase-Frequency Detector, page 20 |
| PLL | Phase-Locked Loop, page 1 |
| PSD | Power-Spectral Density, page 9 |
| PVT | Process Voltage Temperature, page 10 |
| RC | Resistive-Capacitive, page 73 |
| RF | Radio Frequency, page 2 |
| RMS | Root Mean Square, page 9 |
| ROM | Read-Only Memory, page 12 |
| SAFF | Sense-Amplifying Flip-Flop, page 58 |
| SNR | Signal-to-Noise Ratio, page 16 |
| VCO | Voltage Controlled Oscillator, page 2 |

# List of Terms

| | |
|---|---|
| aliasing | The effect that causes different frequency content in a signal to become indistinguishable due to subsampling, page 1 |
| delta-sigma modulation | From the view of fractional-N synthesis, delta-sigma modulators generate an output stream of bits to control the dual-modulus counter, and where distribution the distribution of these bits is such that spurious tones in the frequency synthesizer output are minimized, page 18 |
| Dirac comb function | Infinite series of Dirac delta functions spaced at a regular interval. Also known as an impulse train, page 73 |
| frequency reduction circuit | Used to refer to the circuits in the feedback path that perform some scaling or reduction of the oscillator frequency. The output of this circuit is at lower frequency than the oscillator signal and is suitable for phase matching with a reference signal. Frequency reduction in the feedback path is traditionally done with frequency divider circuits, page 1 |
| group III-V device | Compound semiconductor device primarily composed of two or more elements from group three and group five of the periodic table of elements. Typically used for very high-performance radio-frequency integrated circuits, page 2 |
| jitter | A time-domain measure of short-term variations of a signal relative to some reference or ideal signal, page 8 |
| LC circuit | Resonant circuit built with an inductor and capacitor, which are symbolically represented by L and C, respectively, page 19 |
| lock time | The time required before a PLL becomes phase locked after a startup or when disturbed, page 18 |
| mm-wave | Millimeter-wave, referring to a radio frequency band ranging from roughly 30 GHz to 300 GHz, page 2 |
| Nyquist rate | Minimum sampling rate required to completely reconstruct a signal from it samples, described by the Shannon-Nyquist theorem, page 1 |
| phase comparator | A circuit element for producing an output signal that corresponds to the phase difference between two input waveforms, page 15 |

| | |
|---|---|
| PLL feedback path | The signal path in a PLL that connects the local oscillator to the phase comparison unit. In frequency synthesizers, this typically includes a frequency scaling unit, page 1 |
| prescaler | A frequency divider circuit specifically designed for high-speed operation, page 2 |
| Q factor | Quality factor of oscillating system - representing the ratio between energy stored and energy dissipated per cycle in the system, page 19 |
| spectral purity | A measure of the quality of the frequency output spectrum of a signal, with a more spectrally pure signal exhibiting fewer or less significant undesirable components in its frequency output spectrum, page 11 |
| SR Latch | Set-Reset latch, page 52 |
| subsampling | Sampling below the Nyquist rate of the highest frequency content of a given signal, sometimes referred to as under-sampling, page 1 |
| switching threshold | The input signal decision level at which a latch or flip-flop output has an equal probability of logic high or logic low value. Also know as logic threshold., page 28 |
| transceiver | A device that incorporates both **trans**mitter and re**ceiver** functionality into a single unit, page 2 |
| W/L | Width-over-Length ratio, physical dimensions for the drawn gate of a transistor, page 52 |
| word | In digital circuits or computing, a word refers to a grouping of bits that are processed in unison, page 11 |
| XOR | Exclusive-OR logical operation, page 20 |

# Chapter 1

# Introduction

Frequency synthesizers are circuits that take one or more input signals with known frequencies and produce one or more output signals at other frequencies. In most cases, frequency synthesizers accept one or more low frequency stable reference signals as inputs and generate a range of higher frequency signals that are mathematically related to the frequencies of the reference signals. Frequency synthesizers are found in virtually all modern communication systems, including wireless data networks, cellular phones, radio and televisions receivers, and fiber optic transceivers. Frequency synthesizers are not just found in communication systems, however, as they are also used as clock sources in other applications such as digital logic and radar systems.

Most frequency synthesizers designs are in fact phase-locked loop (PLL) circuits with some type of frequency reduction circuit in the PLL feedback path between the oscillator and the phase detector. In most frequency synthesizers, this feedback path contains a frequency divider circuit. This thesis concerns itself with a novel frequency reduction circuit in the feedback path of the PLL. In this work, I propose to perform frequency reduction by replacing the frequency divider with a sampling latch circuit operated at a frequency well below the Nyquist rate of the latch input signal. By subsampling the oscillator signal, a low frequency alias signal is produced, which can subsequently be utilized in the same manner as a frequency divided signal. We refer to a frequency synthesis PLL that uses an alias frequency as an alias-locked loop (ALL).

As predicted by Gordon E. Moore [2], the dimensions of complementary metal-oxide semiconductor (CMOS) devices have continued to shrink, and over the past forty years CMOS processes have become the technology of choice for cost-effective implementations of a wide variety of digital, analog, and to some extent, radio-frequency (RF) circuits. As device geometries have shrunk, the figures-of-merit used to measure the high-frequency performance of CMOS circuits have continued to improve. In particular, the maximum unity current-gain cut-off frequency, ($f_T$), and maximum unity power-gain frequency, ($f_{max}$) make it possible to build complete RF transceiver circuits in CMOS technology. For 45-nm bulk CMOS technologies $f_T$ and $f_{max}$ frequencies are reported to be 280 GHz and 350 GHz, respectively [3]. With such high frequency capabilities in CMOS, fully-integrated transceivers for millimetre-wave (mm-wave) frequencies are no longer the exclusive domain of group III-V devices. Traditionally, esoteric semiconductor technologies such as Gallium-Arsenide (GaAs) or Indium-Phosphide (InP) have been the dominant semiconductor devices in RF front-end circuits [4].

A key building block for fully integrated transceivers is the frequency synthesis PLL used to generate the mm-wave frequency signals required in the transceiver. Building integrated PLLs at any frequency is challenging, since the design of these circuits requires knowledge of control systems theory, digital, analog and RF circuit design. PLLs for mm-wave frequencies present additional challenges to the designer because of poorly characterized device technology at such high frequencies and difficulties encountered in testing systems at these frequencies. Furthering the effort toward integrating mm-wave PLLs, several CMOS voltage-controlled oscillator (VCO) circuits have been published operating at mm-wave frequencies [5, 6]. Some of these VCOs [7, 8, 9] operate at or exceed the $f_{max}$ of the process they are implemented in.

The critical component that limits the maximum operating frequency for PLLs [10], and the focus of this work, is the development of an alternative to high speed frequency prescalers or dividers that operate at frequencies near the $f_{max}$ of the process. This thesis suggests that using a sampling latch, rather than a frequency divider, may aid in this effort, since the sampling latch output is at a much lower frequency than the VCO frequency. Using a sampling latch that is operated in this fashion significantly relaxes the constraints placed on the feedback circuit. A sampling latch can have reset and clock-to-output delays that span multiple VCO clock cycles. The sampling window for such a latch need not be shorter than the VCO cycle time either, as long as the latch output value consistently represents the phase of the VCO signal at the sampling instant. Problems with DC bias and mismatch in such a latch can cause the duty cycle of its output signal to deviate from 50 %, which in the worst case is evident as a stuck-at-one or stuck-at-zero output. This duty cycle deviation can be corrected with appropriate control and tuning circuitry in the latch. Finally, power consumption for a subsampling latch is reduced compared to a frequency divider, mostly due to the much lower operating frequency of the sampling latch.

## 1.1 Thesis Organization

The remainder of this dissertation discusses ALL-based frequency synthesizers in further detail. Chapter 2 provides background information on frequency synthesizers and discusses important properties used to quantitatively evaluate frequency synthesizers designs. A number of common frequency synthesis circuits are discussed along with the suitability of each such circuit for high-speed operation. To set the context for the alias-locked loop, a brief overview of the basic building blocks of a frequency synthesis PLL is provided. The chapter concludes with the latest research on frequency synthesis at mm-wave frequencies, with specific attention paid to CMOS-based mm-wave PLLs. Chapter 3 describes the ALL architecture in detail, discusses the impact of subsampling on the PLL design, and describes the constraints that guide the design of the sampling latch. Chapter 4 presents circuit designs and simulation results for several sampling latch topologies, and discusses their suitability as aliasing units in the feedback path. Chapter 5 describes an idealized non-linear time-domain model used to simulate the ALL architecture, presents the results of the non-linear simulations, and shows successful simulated operation of an ALL system in 90-nm CMOS technology. Finally, chapter 6 concludes this thesis by summarizing the results and suggesting future research to be carried out.

# Chapter 2

# Background

## 2.1 Overview

This chapter starts by presenting the history of frequency synthesis. Next, the key measures used to evaluate frequency synthesizers are introduced, including concerns such as settling time and output signal statistics. Then a large section is devoted to describing various frequency synthesis methods. A distinction is made between direct, indirect, and hybrid synthesis methods. A brief description and explanation of the operating principle is provided for direct analog and direct digital synthesizers. For indirect synthesis methods both the delay-locked loop (DLL) and the PLL are described, together with an example of a hybrid synthesis circuit.

Since the ALL is essentially a PLL circuit, the chapter then outlines the key building blocks that make up most ordinary PLLs. To explain the operating principle behind the ALL, a section is devoted to an explanation of aliasing as it applies to frequency reduction in the PLL feedback path. This section also includes a review of previous work done on the use of aliasing in PLLs. Finally, the chapter concludes with a review of the state-of-the-art in high-speed frequency synthesis. We note that for modern CMOS processes, the fastest frequency dividers in the literature are nowhere near fast enough to keep up with the fastest reported VCOs.

## 2.2 Frequency Synthesizer History

For efficient usage of the frequency spectrum a method of distinguishing between the various users of the medium needed to be developed. In modern communication systems with advanced signal processing algorithms the same span of wireless spectrum can have multiple users. Early wireless communications systems instead avoided interference between users of the medium by dividing the frequency spectrum into channels. This can be achieved by assigning the transmitter and transceiver a fixed frequency for communication, but the ability to select a particular channel as needed allows for much greater flexibility.

As the popularity of wireless transmission grew, the bandwidth and spacing for radio channels became more well defined in order to better utilize the available spectrum. This development necessitated the need for improved frequency control, and early radio systems incorporated manually tuned oscillators, which, although tunable, suffered from drift in the output frequency. Then, the advent of crystal oscillators made frequency sources that have very good phase noise and frequency stability properties commonplace. Unfortunately crystal oscillators have very little tuning capability, and in early systems each channel used by the system would require a separate crystal oscillator specifically tuned for that channel. The limitations and expense of systems containing multiple oscillators set the stage for the development of frequency synthesis circuits.

Circuits that achieve good tuning range and phase noise are mixing synthesizers. These synthesizers combine the output of two or more crystal oscillators at different frequencies, producing an output signal that contains the various beat and harmonic frequencies given by the sum and difference of the crystal oscillator frequencies. The complexity of selecting the desired frequency, which would usually require filtering operations, makes these mixing synthesizers systems unattractive in most situations. Instead, PLL-based techniques have become the method of choice for most frequency synthesis applications [11].

Although some of the concepts underlying PLLs were published by Appleton

[12] in 1922 and De Bellescize [13] in 1932, it was not until the advent of television in the 1950's that the PLL became widely used, although in this role, the PLL did not act as a frequency synthesizer. An early design with frequency division in the feedback path of the PLL was patented in 1970 [14]. Around the same time, integrated circuits (ICs) started to take off, and as with other circuits, the PLL has benefited from the continued reduction of feature sizes in semiconductor technologies. The improvements in IC technology permitted the eventual full integration of all of the building blocks of PLL into a single device, while simultaneously reducing power consumption and increasing the frequency capability of PLLs.

## 2.3 Frequency Synthesizer Properties

Several measures can be used in the specification of the output of a frequency synthesizer. The specification for a frequency synthesizer will depend on the application because the constraints for a system clock generator for a high-speed digital IC will be different from the specification of a frequency synthesizer for a wireless transceiver. The properties of the reference clock signal will also guide the design. When only a lower frequency reference signal is available, the frequency synthesizer must perform frequency multiplication to reach the desired output frequency. The remainder of this section provides some of the key specification used to describe a frequency synthesizer.

### 2.3.1 Tuning Range

The tuning range is the range of frequencies that can be synthesized by the system. Besides specifying the maximum and minimum frequencies synthesizable, the granularity of increments in synthesized frequency is also important, especially in wireless transceivers where predefined channel frequencies specify the synthesized frequencies that must be generated. With a fixed reference clock, this will determine the frequency multiplication factors that the system must support.

### 2.3.2 Settling Time

Another measure of frequency synthesizer performance is how fast the system can switch between generated frequencies, and whether such switching has any side effects on the output signal. Rapid switching between synthesized frequencies is required by certain wireless standards that change the radio channel that the transceiver occupies while in use. A somewhat related measure is the start-up time required before the system becomes locked to the reference signal and produces the desired frequency at its output port. In certain frequency synthesizer architectures, the desired synthesized frequency is available in a single reference clock cycle, while in others architectures, such as PLLs, the system can take a many reference clock cycles before it reaches steady state.

### 2.3.3 Output Signal Statistics

Of great importance in the specification for a frequency synthesizer are the statistics of the output signal. The synthesized signal can be looked at from a time and frequency domain perspective. In most frequency synthesizers applications, short and long-term stability of the signal statistics are desirable. In reality, periodic and random variations in frequency synthesizers affect the quality of the output signal. A brief overview of the different measures of the signal quality is presented in this section.

#### 2.3.3.1 Time Domain

Clock generators for digital ICs are often evaluated on the basis of the time domain behaviour. The short-term measure of random and periodic time domain variations of the output signal relative to a reference or ideal signal is usually termed "jitter", and is usually measured for each cycle. Several types of jitter are commonly used [15] in specifications and the literature. A widely used measure is period jitter, which represents the clock period deviation of the measured signal from an ideal signal. Cycle-to-cycle jitter is another common term, and is usually defined as a measure in the variation in the clock period between adjacent cycles of the signal.

Whatever measure of jitter is applied, it is the statistical calculations on the jitter measurements that are usually of interest. In practice, jitter statistics are commonly expressed in terms of the peak-to-peak or root mean square (RMS) values of the jitter distribution.

#### 2.3.3.2 Frequency Domain

The output of frequency synthesizers for communication systems are typically evaluated on the basis of the frequency domain behaviour of the system. For most communication systems, the power spectral density (PSD) plot of the generated signal would ideally look like an impulse at the desired frequency. In reality, random noise sources in circuits will cause phase noise, which is in essence a frequency domain measure of jitter. More formally, phase noise is the short-term random variation in the frequency (and phase) of a signal [16]. The random noise will cause the output spectrum of the synthesizer output to be distributed in frequency. When plotted, the PSD graph for a free-running oscillator tends to have a sharp peak at the desired frequency, with smoothly decreasing sides. For illustration, Figure 2.1 shows the power spectrum of an oscillator, centred at a frequency $f_c$.

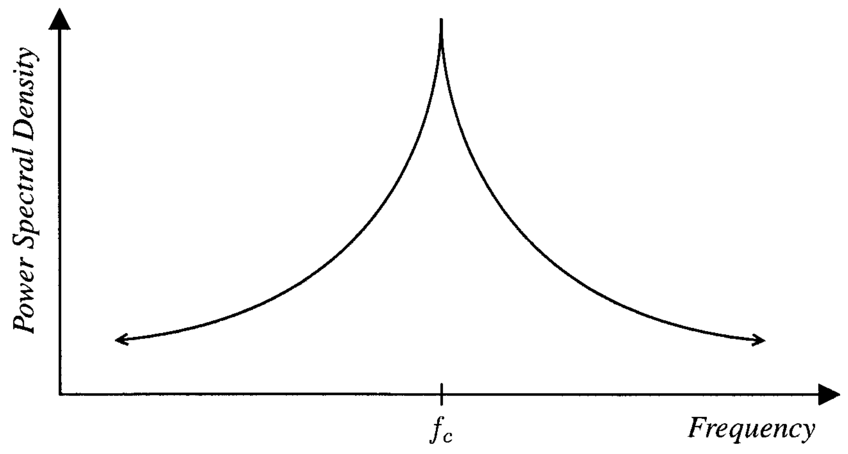

Figure 2.1: Illustrative power spectral density plot of an oscillator

Rather than using absolute power levels, phase noise figures are most often given relative to the carrier power at $f_c$. Such phase noise figures are quoted per unit bandwidth (one Hertz) at some frequency offset relative to the carrier frequency [16]. A typical performance figure for phase noise may thus be expressed like: "the output phase noise is -90 dBc/Hz at a 100 kHz offset".

Although the output spectrum of a free-running oscillator may look like that in Figure 2.1, most frequency synthesizers are synchronized with some reference frequency, which usually introduce additional peaks at specific frequencies into the output spectrum. Peaks caused by the reference clock are usually referred to as reference spurs. Spurious tones may also arise from other non-idealities in the circuits used to implement the frequency synthesizer, and vary depending on the type of architecture used for frequency synthesis. Like phase noise measures, the power of spurious tones in the spectrum is expressed relative to the power of the carrier frequency.

### 2.3.4 Other Concerns

When manufacturing a frequency synthesizer for commercial purposes, cost sensitivity and competitive reasons also give rise to several other concerns that play a role in the design process. For mobile applications, power consumption is of key importance, and high speed circuits like frequency synthesizers can be power hungry. When a frequency synthesizer is implemented on an IC, the silicon area occupied by the chip plays a significant factor in the total manufacturing cost of the design. The cost of IC based implementations must also consider packaging issues and the testability of the design. Further concerns that should be evaluated for commercial designs include the suitability of the design for rapid and reliable reproduction of the device on a mass scale and across variations in process, voltage and temperature (PVT).

## 2.4 Frequency Synthesis Methods

As alluded to in the history section of this chapter, PLL-based systems are not the only way to perform frequency synthesis. Frequency synthesis methods can be classified into the following three categories [17]:

- Direct synthesis
- Indirect synthesis
- Hybrid synthesis

### 2.4.1 Direct Synthesis

Direct synthesis generates the desired frequency using open loop methods only. Direct synthesis methods can be further classified into direct analog synthesis (DAS) and direct digital synthesis (DDS).

#### 2.4.1.1 Direct Analog Synthesis

DAS uses analog processing elements such as mixers, filters, multipliers, dividers and switches and operates on one or more stable reference frequency sources. One of main benefits of the DAS approach over other methods is that very fast frequency switching is possible. If carefully designed, a DAS system can also achieve a very high spectral purity [1] with low phase noise and spurious tones. However, the analog approach becomes complicated and expensive when a large number of distinct output frequencies are required, since the number of filters, multipliers and mixers is directly dependent on the number of output frequencies required. Because all the components operate in the analog domain, the frequency capability of DAS circuits are primarily limited by the process technology itself, and can thus generate high output frequencies through upconversion or multiplication methods. An block diagram of a sample DAS system is shown in Figure 2.2

#### 2.4.1.2 Direct Digital Synthesis

A more common direct synthesis approach includes a large number of digital circuits to generate signals of an arbitrary frequency. An early DDS system was described by Tierney, et al. in 1971 [18]. The basic idea behind DDS is to use a digital circuit to generate digital control words for a digital-to-analog converter (DAC). By generating an appropriate sequence of control words for the DAC, it is possible to

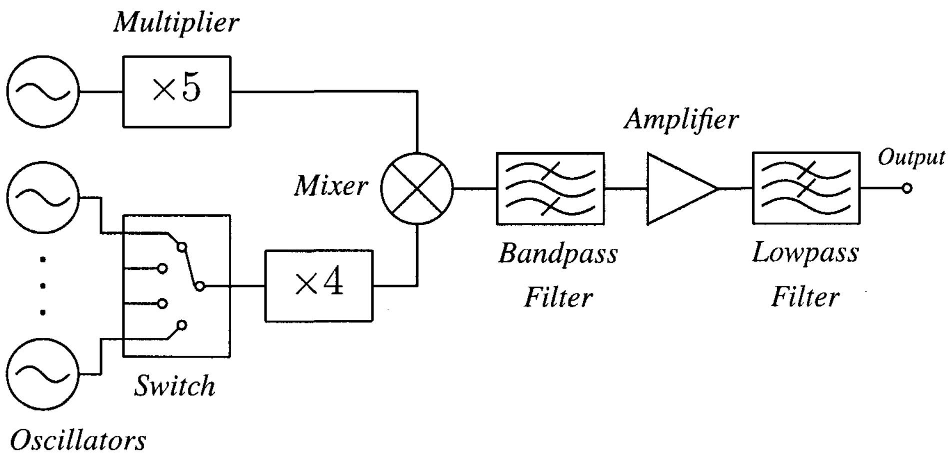

Figure 2.2: A direct analog synthesis system block diagram, based on [1]

produce a sinusoidal output. The high-level architecture of a direct digital frequency synthesizer is depicted in Figure 2.3. The figure shows the waveforms before and after the low-pass filter, which is connected to most DDS systems in order to smooth out the DAC output.

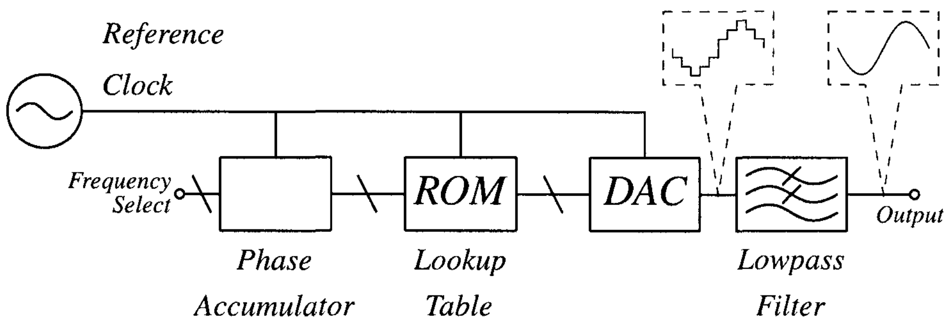

Figure 2.3: High-level direct digital synthesis architecture block diagram

To generate a sinusoidal output of a programmable frequency, the blocks in a typical DDS circuit may behave as follows. Referencing Figure 2.3, the external clock source, *Clock*, drives all the digital components in the phase accumulator, lookup table and DAC. A simple phase accumulator may consist of a counter that counts up to a programmable value, which when reached, causes the counter to wrap and start over. The counter output is then scaled to represent a phase between 0 and $2\pi$. This phase value is then directed to the lookup table that maps phase values to the corresponding sine wave amplitude values. At its simplest, the lookup table could be as simple as a read-only memory (ROM). The amplitude values are then used to control the DAC that converts the binary representation of the amplitude to an analog voltage. Finally, a low-pass filter eliminates the staircase-like output of the DAC to smooth the waveform. Several improvements and modifications for the described architecture have been published [19],[20], but the principle of operation remains the same.

The flexibility of the digital logic in the DDS architecture means this system is not just limited to generating sinusoidal outputs. Given the appropriate lookup table programming, a DDS can generate almost arbitrary periodic waveforms. Furthermore, the programmability allows for a wide range of frequencies to produced. Since a DDS system does not contain a feedback loop, switching between different frequencies can take as little as one cycle of the input clock and can be done without introducing phase discontinuities.

Because DDS circuits reconstruct a waveform from sampled data, the maximum output frequency of such a circuit is limited by the Nyquist-Shannon sampling theorem [21]. This theorem holds that to completely reconstruct a given signal, the sampling rate must be at least twice the maximum frequency contained by the given signal. Although the external reference clock can be operated at a very high frequency, the digital circuitry and DAC will limit the maximum frequency achievable. For Nyquist limited DDS circuits the maximum output frequency that can be produced is about 40 % of the clock signal [22]. DDS systems can change their output frequency in very fine steps, but suffer from spurious signals in their output spectrum. These spurious signals are in part attributable to quantization noise arising from the digital representation of the amplitude and truncation errors when low-order bits of the phase are discarded [23]. These spurious signals can thus be reduced by increasing the resolution of the phase and amplitude representation, but this requires larger look-up tables, DACs and phase accumulators. High-speed high-resolution DACs are difficult to design and DDS systems with low spurious noise are therefore more expensive in area utilization, complexity and power consumption. Because CMOS based DDS is severely limited in frequency by the maximum operating speed of the digital components and the DAC, the highest speed DDS circuits are implemented in esoteric semiconductors families [22], and DDS is generally not suitable for frequency synthesis anywhere near the $f_{max}$ of a process.

### 2.4.2 Indirect Synthesis

Indirect synthesis refers to those systems that utilize a feedback loop to control and stabilize the output frequency. Indirect synthesis systems include DLL and PLL circuits. The main difference between DLL and PLL circuits is that PLL circuits use feedback to control the frequency of a local oscillator, while DLL circuits use feedback to adjust the delay of one or more delay elements. While DLL circuits are rarely used for frequency synthesis, PLL circuits are widely used and very popular as frequency synthesizers, with a large number of books and papers that describe their operation and design.

#### 2.4.2.1 Delay-Locked Loop Frequency Synthesis

Although typically used for clock alignment or clock deskewing in settings such as interface circuits for computer memories [24], a DLL can also be used as a frequency synthesis system [25]. One method [26] to perform frequency multiplication with a DLL uses an input reference clock that is passed through a tapped delay line constructed from a number of tunable delay elements connected in series. A block diagram for the tapped delay-line based DLL is graphically depicted in Figure 2.4.

In the delay-line based DLL, feedback is used to adjust the total delay through all stages of the delay line so that the phase of the input and output signals of the delay line match. If the input and output phases are matched, the waveforms at the taps on the delay line are then at regularly spaced phase offset and equally distributed over one period of the reference clock. The edges of these waveforms can then be combined in an edge combiner to generate a frequency that is a multiple

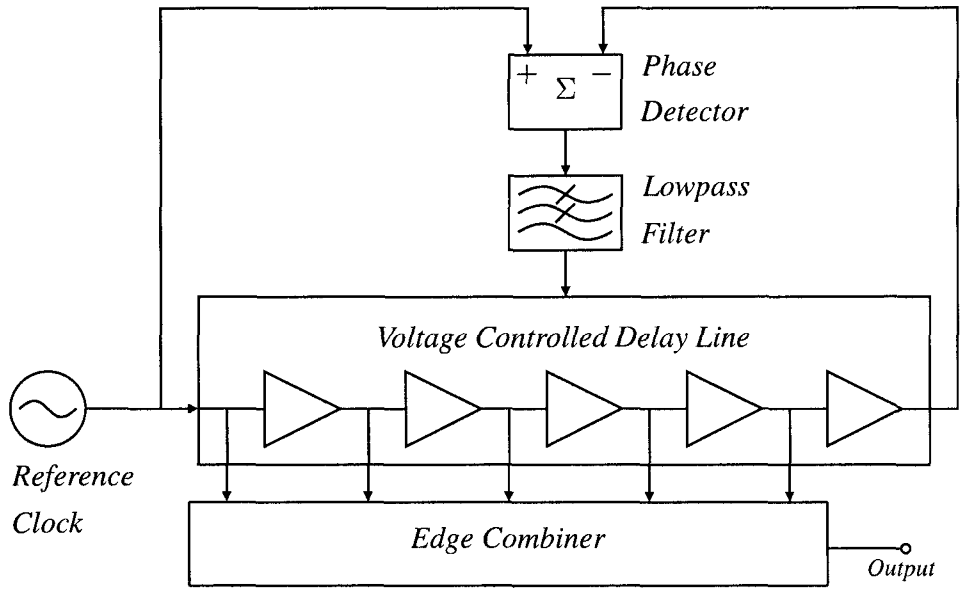

Figure 2.4: Block diagram of a tapped delay-line delay-locked loop frequency multiplier

of the reference clock.

An improvement on the line-delay based DLL is the multiplying delay-locked loop (MDLL) [25]. The MDLL uses a single delay element, avoiding some of spurious tones that appear when the delay elements in a delay line are not closely matched. In this method, the single inverting delay element is connected to the reference clock through a multiplexer, as shown in Figure 2.5. The logic block is used to control the multiplexer and tune the delay element. This logic typically includes at least a counter, a phase comparator, and a loop filter.

The general operating principle for an MDLL architecture is as follows. At the rising edge of every reference clock period, the reference clock is connected to the delay element through a multiplexer. After the arrival of a rising edge on the reference clock the multiplexer is switched immediately, connecting the delay element to itself in feedback mode and essentially creating a ring oscillator. After having counted $N$ rising edges at the output of the delay element, the multiplexer is switched back to the reference clock input, resetting the counter and activating a phase detector. With the multiplexer switched back to the reference clock signal

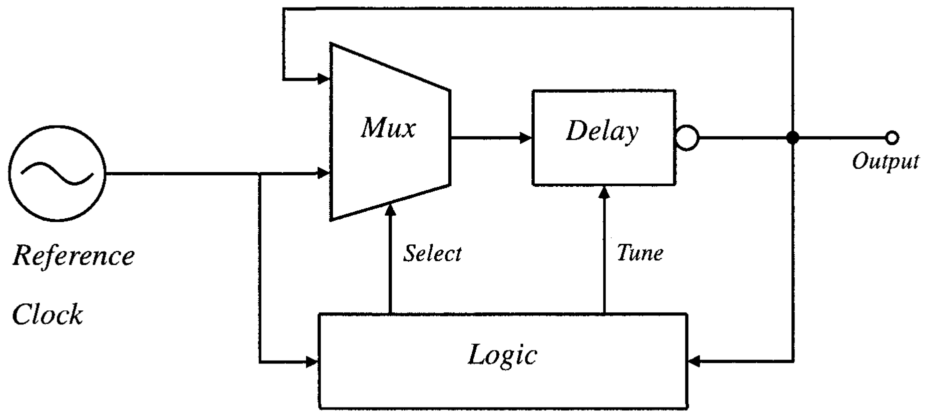

Figure 2.5: Block diagram for a multiplying delay-locked loop

(which is still low), a new rising edge at the output of the delay element output is generated and compared to the next rising edge on the reference clock by the phase detector. The measured phase error is then used to tune the delay element so that in locked operation exactly $N$ clock cycles of identical duration are generated for every reference clock period.

Since DLL architectures do not accumulate jitter over multiple reference clock cycles like PLLs do, DLL based frequency synthesizers can achieve very good phase-noise measures. The periodic correction of phase alignment does, however, cause this architecture to suffer from some deterministic jitter [27]. Overall, the DLL is a suitable architecture for many frequency synthesis applications. Unfortunately, the current-starved delay line construct limits the maximum operating speed to frequencies well below the $f_{max}$ of the process.

#### 2.4.2.2 Phase-Locked Loop Frequency Synthesis

PLL circuits find application in a wide range of settings, and one way of viewing the principles that guide their design is by examining the signal-to-noise ratio (SNR) of the signal used for phase-locking. Frequency synthesis PLLs usually operate in a high SNR environment, since the reference clock is produced locally by a strong and stable oscillator such as a quartz crystal. In contrast, PLLs used for carrier recovery from wireless signals operate in low SNR environments where the reference signal is received with noise and the carrier attenuated and distorted by several radio propagation effects that characterize wireless channels. Since SNR impacts the design of a PLL significantly, this thesis concerns itself solely with high SNR applications.

As previously mentioned, a PLL uses feedback methods to control a local oscillator. A standard frequency synthesis PLL is depicted in Figure 2.6. The principle of operation for a basic frequency synthesis PLL like the one shown in Figure 2.6 is as follows. The output signal of the PLL is generated by a tunable local oscillator, usually implemented as a VCO. The VCO output is also connected to a frequency divider in the feedback path. The frequency divided VCO signal is compared to a reference clock signal in a phase comparator. The phase comparator generates an error signal that corresponds to the phase error between the divided signal and reference clock. This error signal is then processed by a low-pass loop filter, and the output signal from the low-pass filter in turn tunes the VCO. This feedback system attempts to adjust the phase error to zero and results output signal that is phase-locked to the reference clock. This frequency of this output signal is a multiple of the reference clock frequency, with the frequency multiplication factor determined the frequency divider.

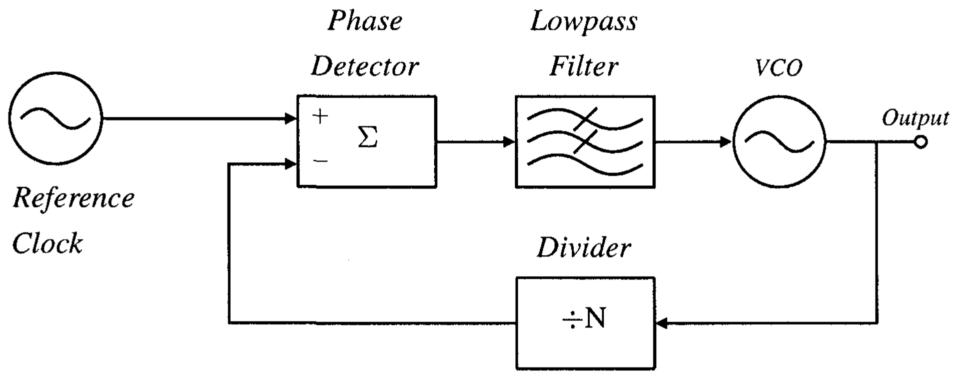

Figure 2.6: Block diagram of a frequency synthesis phase-locked loop

In the most basic PLL, the frequency divider in the feedback path divides by an integer factor N, and hence this type of PLL is referred to as an integer-N frequency synthesizer. If the divide factor N is programmable, an integer-N frequency synthesizer can be coarsely tuned, with the frequency resolution increments equal to the reference clock frequency. Thus, for closely spaced channels in wireless standards, the reference clock frequency should be small, and N correspondingly large. A large multiplication factor is detrimental because it degrades the output frequency spectrum with closely spaced spurious tones and requires loop filter parameters that result in an increased lock time and poorly suppressed VCO phase noise [10].

To overcome the shortcomings of integer-N PLLs, the divide operation can be modified to perform non-integer division. To achieve such fractional-N frequency synthesis, the frequency division factor in the feedback path is made to vary between two different division factors, typically done using a dual-modulus divider. One method, called pulse swallowing, switches the division factor at a regular interval. The regular switching in a pulse swallowing fractional-N design, however, introduces a spurious tone in the frequency synthesizer output. These spurious tones can be minimized by using a higher-order delta-sigma modulator that randomizes the switching of the division factor [28].

The relative simplicity of PLLs makes them a common choice for frequency synthesis applications, and with fractional-N synthesizers, a wide range of closely spaced frequencies can be generated. Phase noise and jitter measures are dependent on the type of PLL components used in the implementation, with the loop filter parameters being of particular importance to the nature of the PLL synthesizer output. In terms of frequency capability, the upper limit for PLL frequency synthesis is usually limited by the maximum divider speed [10]. For traditional frequency dividers, this means that the highest speed PLLs can perform frequency synthesis at frequency significantly higher than the fastest DLL or DDS implementations.

### 2.4.3 Hybrid Frequency Synthesis

Although a frequency synthesis system can rely exclusively on direct or indirect synthesis methods, both techniques can also be applied in combination. For instance, a DDS synthesizer can be used as reference frequency source for a PLL, resulting in a frequency synthesizer with better switching times or improved frequency resolution [29, 30]. Unfortunately, the PLL transfers the spurious tones from the DDS to its output. This is avoided in another hybrid frequency synthesizer architecture that mixes the output of a DDS frequency synthesizer into the feedback loop of a PLL [31]. For very high speed frequency synthesizers, one hybrid architecture approach uses a PLL followed by RF upconversion circuits [32]. In summary, hybrid system can be used to avoid some of the drawbacks associated with one particular method. However, due to the combination of multiple techniques hybrid frequency systems tend to be more complex.

## 2.5 Phase-Locked Loop Implementation

When developing a frequency synthesis PLL, the designer is faced with several implementation choices. A PLL can be implemented with purely analog components, digital components, a mix of digital and analog components, and designs can even include software modules. Frequency synthesis PLLs, whether digital or analog, still share a common set of components, including oscillators, loop filters, phase detectors and frequency dividers. This section will provide a brief overview of the most common building blocks found in most PLL frequency synthesizer implementations.

### 2.5.1 Common Building Blocks

Figure 2.6 shows typical components in a frequency synthesis PLL. The function of each component, as well as some common implementations, are described.

#### 2.5.1.1 Oscillator

For a PLL to be of any use, the oscillator that generates the synthesized frequency must be tunable, typically through voltage control. High-speed integrated VCOs are most frequently implemented as a ring oscillator or using a LC circuits. Ring oscillators generally have a wider tuning range and are more compact than LC circuits, but consume more power and have a lower quality factor, Q, than an LC oscillator [33] for the same frequencies. In applications such as wireless communications, where phase noise is important, LC oscillators are typically used. Due to their smaller size, ring oscillators are most often found in frequency synthesizers for digital ICs, where phase noise requirements are less stringent. In some digital circuits, phase noise may even be desirable to ensure that emissions from the clock-triggered digital circuits minimize the electro-magnetic interference. The fastest reported CMOS VCO in the literature runs at 410 GHz and is implemented in a 45 nm process [8]. Push-push VCOs like this have a lower fundamental oscillation frequency, but attenuate the fundamental signal and extract, for instance, the second harmonic in order to generate output frequencies in excess of $f_{max}$.

#### 2.5.1.2 Phase Comparator

The phase comparator's role is to transform a phase difference between the reference clock and feedback signal to a control output for the PLL loop filter. A very simple phase detector can be built using an exclusive-OR gate (XOR) and can be directly connected to an analog low-pass filter. In this configuration an XOR gate functions as a phase detector, but the maximum phase difference detectable is $\pm\frac{\pi}{2}$. A more common phase detector topology is a phase-frequency detector (PFD). PFDs are capable of discerning a phase error of $\pm 2\pi$, and can also be said to detect frequency errors. A basic PFD [34], can be built from two flip-flops and an AND gate, configured as seen in Figure 2.7. A PFD produces either an $Up$ or $Down$ pulse that directs the VCO to speed up or slow down, where the duration of the pulse corresponds to the phase error between the reference and feedback signals. Usually a PFD is coupled to a charge pump that converts the $Up$ and $Down$ pulses to signals processable by a loop filter. The charge pump does this by pushing current into or pulling current out of a low-pass filter while the $Up$ or $Down$ signals are asserted, respectively.

Another phase detector that is worth a brief mention in the context of an alias-locked loop is the binary phase detector, also known as bang-bang phase-detector [35]. A bang-bang phase detector only detects which signal is leading or lagging,

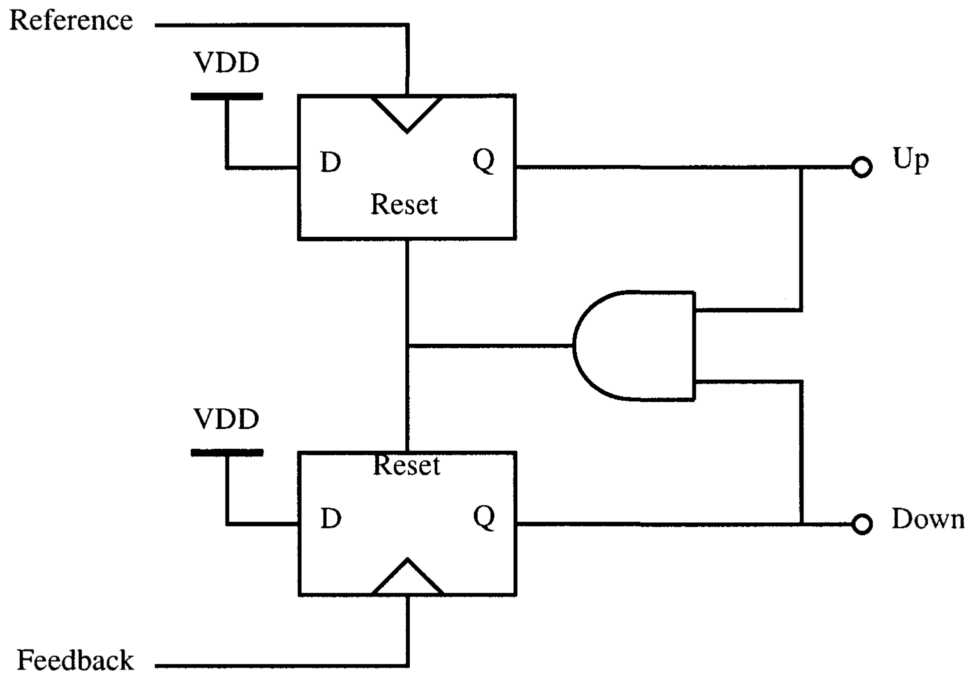

Figure 2.7: Phase-frequency detector circuit

and its output signal does not contain any information on the magnitude of the phase error. The inputs are evaluated once per input clock cycle, and the output value is the sign of the phase error between the feedback signal and reference clock. The bang-bang PLLs and ALL architecture are similar in that both systems produce a phase error signal that is discretized. Further detail on the discrete nature of the phase error signal in the ALL architecture is described in chapter 3. Bang-bang phase detectors typically find application in systems that utilize very high-speed reference signals [36].

#### 2.5.1.3 Loop Filter

PLL loop filters are low-pass filters that act on the error signal produced by the phase detector. The filter bandwidth and frequency response will determine the PLL loop bandwidth and affects parameters like lock time and phase noise suppression. From an oscillator perspective, the PLL acts as a high pass filter, controlling low-frequency (long-term) deviations from the reference frequency and keeping the output phase locked, while passing high frequency noise from the oscillator. Loop filters can be implemented in the analog domain using only passive components such as capacitors and resistors, or they can include active components that may reduce the area required for large passives. It is also possible to perform the same filtering operation digitally, using digital signal processing (DSP) techniques.

#### 2.5.1.4 Feedback Divider

Frequency synthesis PLLs need some sort of frequency reduction in the feedback path in order to perform frequency multiplication. A simple frequency divider can be built using toggling flip-flops connected in series to form an asynchronous counter. The asynchronous counter operates on the VCO output, and each flip-flop performs a divide by two operation on the frequency. Such counters works well for low frequencies, but the fastest static CMOS flip-flops consume significant power and can toggle at only a fraction of $f_{max}$. Rather than using static flip-flops, the fastest frequency synthesis PLLs published [37] use different frequency divider circuits for different frequency ranges. High-speed frequency divide circuits can also be referred to as "prescalers".

## 2.6 Aliasing

As alluded to earlier, aliasing can be used to perform frequency reduction in the feedback path of a PLL-like frequency synthesizer. If a periodic signal is sampled at a rate that satisfies the Shannon-Nyquist theorem, that signal can be fully reconstructed from the sampled data. The plots in Figure 2.8 show such a process, where the original signal with a frequency of 1 Hz is sampled at 4 Hz. For illustration, the sampled data is graphically interpolated using the cubic spline method, resulting in a reconstructed signal with the same frequency as the original. Figure 2.9 shows a repeat of the same experiment, but now sampling the 1 Hz signal at 0.4 Hz. The subsampling process results in aliasing, with the lowest frequency alias signal having a frequency of 0.2 Hz.

To determine the frequency of the subsampled signal, the sampling theorem

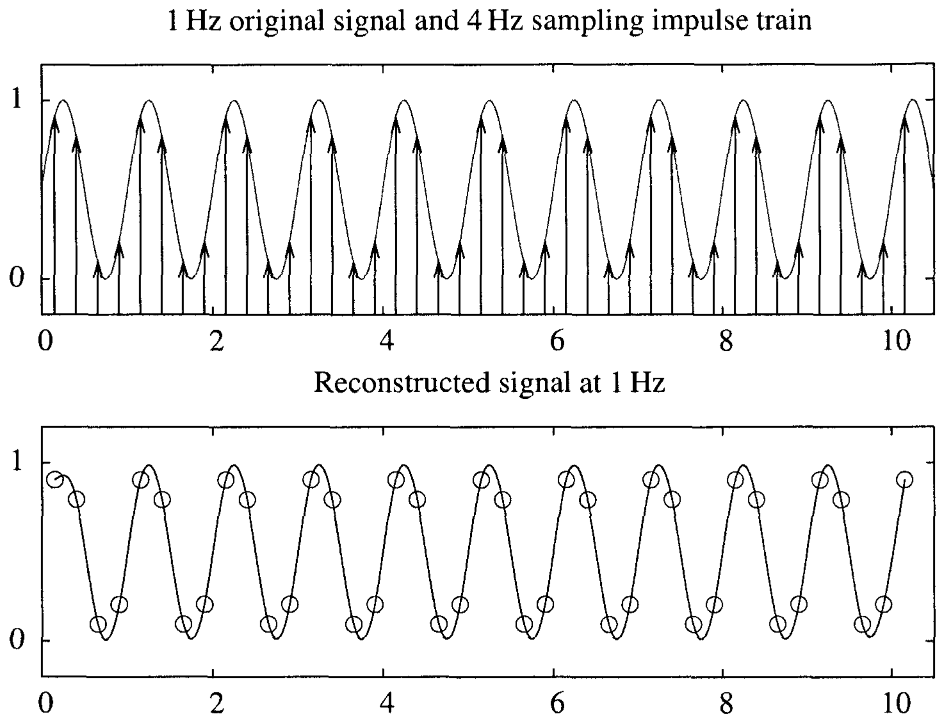

Figure 2.8: Graphical illustration of oversampling, showing the original signal, double oversampling impulse train and reconstructed signal

can be used to arrive at an equation that predicts the alias frequency. We define a time-domain signal $x(t)$ with a Fourier transform of $X(f)$. If this signal $x(t)$ is uniformly sampled at a frequency of $f_{sample}$, the Fourier transform of the sampled waveform $X_s(f)$ will be given by [38]:

$$X_s(f) = \sum_{K=-\infty}^{\infty} X\left(f - K \cdot f_{sample}\right) \tag{2.1}$$

Assume that $x(t)$ is sinusoidal VCO signal with a frequency of $f_{VCO}$. Then the Fourier transform of the VCO signal is given by $X(f) = \delta(f \pm f_{VCO})$ [38], with frequency components at $\pm f_{VCO}$. The sampled signal is thus given by:

$$X_s(f) = \sum_{K=-\infty}^{\infty} \delta\left(f \pm f_{VCO} - K \cdot f_{sample}\right) \tag{2.2}$$

From this equation, the alias frequencies, $f_{alias}$, for a uniformly sampled sinusoidal signal that fit the sampled data are given by:

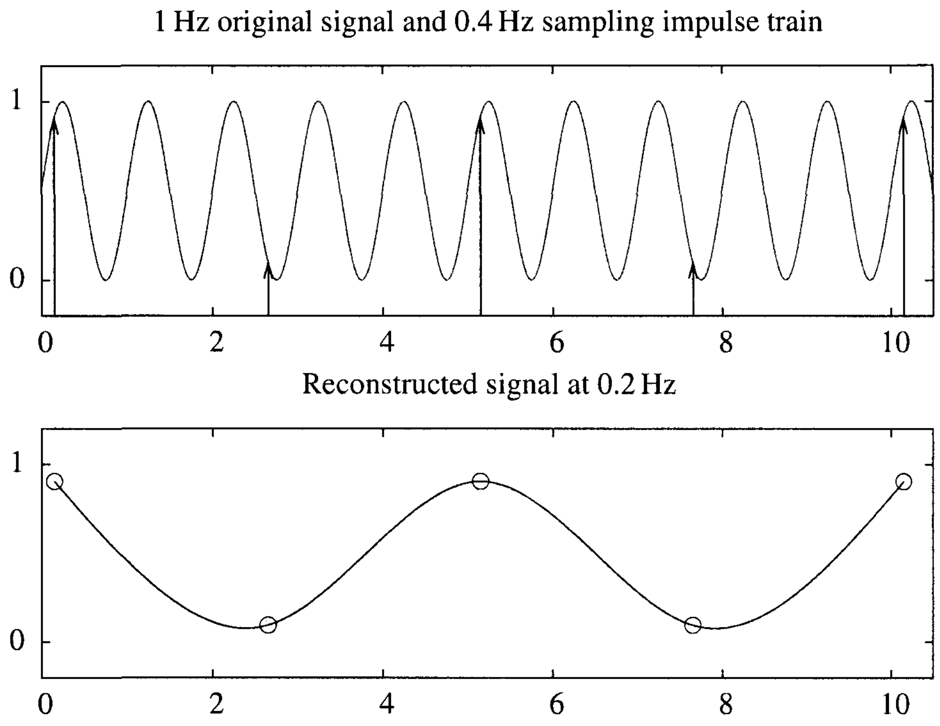

Figure 2.9: Graphical illustration of subsampling, showing the original signal, subsampling impulse train and reconstructed signal

$$f_{alias} = f_{VCO} - f_{sample} \cdot K \tag{2.3}$$

where $K$ is any integer, with the special case $K = 0$ resulting in the original signal. The lowest alias frequency generated is given by the difference between $f_{VCO}$ and the nearest harmonic of $f_{sample}$:

$$f_{alias} = f_{VCO} - f_{sample} \cdot round\left(\frac{f_{VCO}}{f_{sample}}\right) \tag{2.4}$$

where $round(x)$ rounds $x$ to the nearest integer value. The lowest frequency produced by the aliasing operation therefore falls in the following range:

$$-\frac{f_{sample}}{2} \leq f_{alias} \leq \frac{f_{sample}}{2} \tag{2.5}$$

Examining the relationship between the alias frequency and the VCO frequency, it can be recognized that subsampling can be used to generate an aliased signal with a frequency that is significantly lower than the VCO signal. Thus, a sampler operated below the Nyquist rate can be inserted in the feedback path of a PLL to perform frequency reduction, resulting in a system that can perform frequency synthesis.

### 2.6.1 Previous Subsampling Architectures

The notion of subsampling the VCO signal in the feedback loop to perform frequency reduction was first introduced by Amr N. Hafez and M. I. Elmasry in [39], further described in [40] and patented in [41, 42]. In these previously reported designs the reasons for introducing a subsampling circuit in the feedback path are cited as lower power consumption and improved phase-noise performance. Because the designs in the cited papers use analog sample-and-hold circuits that subsample VCO, any harmonics in the VCO signal are also present as problematic low-frequency harmonics in the subsampled output signal. Using sample-and-hold circuits to perform subsampling therefore requires a separate low-pass filter at the output of the sample-and-hold unit to filter out problematic harmonics. Another invention disclosed in a patent by G. E. Von Dolteren Jr. [43] replaces the analog sample-and-hold circuit by an ADC converter that samples below the Nyquist rate. The subsampled output of this ADC is digitally processed to find the phase error, which subsequently filtered by DSP techniques after which a DAC converts the loop filter value back to a voltage that steers the VCO. Unfortunately, the approaches reported previously in literature and patents are not targeted toward higher frequency operation.

## 2.7 Current High-Frequency Synthesis Methods

Although PLLs are very popular as frequency synthesizers for current wireless standards that go up to about 10 GHz, frequency sources for millimetre frequencies have traditionally been the domain of direct analog synthesis or hybrid synthesis methods.

### 2.7.1 Direct Analog and Hybrid Systems

Much published work on mm-wave frequency sources describes systems that use compound semiconductor microwave circuits. Integrated microwave circuits are known as monolithic microwave integrated circuits (MMICs). These MMICs usually consist of microstrip networks and amplifiers that form dielectric resonator oscillators (DROs), stubs, mixers and frequency doublers. These systems are usually not locked to a lower reference frequency, and their tuning capability depends on the oscillator type, relying on mechanically tuned oscillators for example [44].

Hybrid approaches using a PLL to produce a stable frequency at one-third [45], one-half [32] or other large fraction [46] of the desired output frequency. Then, cascading frequency triplers, doublers, or mixers generate the desired output frequency. The hybrid approach with a PLL and upconversion stage improve over the approaches outlined in the previous paragraph by simplifying the tuning process, which can be done by changing the PLL reference clock frequency or using a programmable divider.

### 2.7.2 PLL Only

In recent years, a number of papers [47, 48] have been published that describe CMOS-based high-speed frequency dividers for millimetre wave PLLs, with the fastest CMOS frequency divider operating at 95 GHz [49]. A handful of fully integrated CMOS PLL frequency synthesizers for millimetre wave frequencies have appeared as well, with some PLLs operating around 60 GHz [50, 51] and the fastest fully-integrated PLL at 75 GHz [37]. These high-speed PLLs use cross-coupled differential LC VCOs, and specialized prescalers. Because many of these high-speed prescalers only operate over a narrow range of frequencies, a number of divider topologies are cascaded. For instance, the PLL in [37] uses injection-locked dividers for the first divider stage, a Miller (also know as regenerative) divider for the second stage, and static dividers for the remaining stages. The maximum frequency achieved in these PLLs is clearly much slower than the fastest reported VCOs, and Table 2.1 shows the frequency discrepancy between the fastests reported frequency dividers and VCOs for current state-of-the-art CMOS processes. Of note for this table is that no standalone high-speed VCOs faster than the design in [49] have been reported. One problem with the high-speed PLLs in the literature is that the frequency division ratio is fixed. A programmable division factor is required when a fixed reference frequency prevents multichannel tuning of the PLL. Frequency synthesis for multiple channels is a requirement in most wireless systems. In most cases a programmable divider relies on a fractional-N synthesis approach, which requires precise control of the division modulus of the first frequency prescaler stage. The delay of the modulus control signal that alters the modulus, however, is too long at the oscillation frequencies in question [32].

Table 2.1: Comparison of fastest VCOs and frequency dividers in the literature, given as maximum operating frequency (GHz)

| technology | 90 nm | 65 nm | 45 nm |
|---|---|---|---|
| VCO | 324 [9] | 95 [49] | 410[8] |
| divider | 75 [37] | 95 [49] | — |

## 2.8 Summary

This chapter has provided a brief overview of the history behind frequency synthesizers, described some quantitative measurements used to compare frequency synthesizers. Several frequency synthesis methods have been presented, as well as the basic building blocks of a frequency synthesis PLL. Finally, a brief overview of the state of the art in high-frequency synthesis methods has been described, with a special focus on high-speed CMOS PLL designs. Although the latest PLLs discussed can achieve very high frequencies indeed, the highest VCO frequencies reported still exceed the capabilities of fastest frequency dividers by a significant margin, limiting the maximum frequency that can be synthesized. In the next chapter, we show a solution that addresses the discrepancy between the maximum frequencies of VCOs and frequency dividers.

# Chapter 3

# Alias-Locked Loop Architecture

## 3.1 Overview

This chapter presents the modified PLL architecture that uses subsampling in the feedback loop to create an ALL and discusses the impact of this modification on PLL design and behaviour. The attributes required for the subsampling unit are then presented together with a method to maintain the performance of subsampling circuits when suffering from switching threshold offset problems.

## 3.2 Proposed Architecture

The central idea presented in this thesis concerns itself with the frequency reduction performed in the feedback path of the PLL. While frequency synthesis PLLs perform frequency division in the feedback path as shown in Figure 2.6. We instead propose that a sampling circuit in the feedback path can be used to build a functional frequency synthesis system. An ALL frequency synthesizer is largely identical to a traditional PLL, and only differs in the feedback path.

As mentioned in chapter 2, frequency synthesizers relying on aliasing have been published previously, this thesis departs from earlier works in two ways. The primary reason for using a subsampling architecture in the previously cited works has been to improve phase-noise properties of the synthesized signal. The reason why subsampling is explored in this work is to investigate new architectures that could help in building high-speed frequency synthesizers, potentially for frequencies near or above $f_{max}$. The second way in which this work departs from previously published work is in a redefinition functionality required from the sampling circuit. While previous papers and patents use sample-and-hold circuits or ADCs that provide an accurate sampling of the voltage level of the input signal at the sampling instant, this thesis instead suggests that a sampling latch circuit with a binary output not only suffices but also relaxes the design constraints in a fashion that allows for higher frequency operation. Thus, rather than providing an analog or multi-level digital representation of the subsampled VCO signal, a regenerative circuit in the latch creates a decision circuit that evaluates the VCO signal at the sampling instant to either a logic-high or logic-low value. The benefit of using a static latch, including most conventional digital latches, is that they store a binary value using internal positive feedback or regeneration. Such regenerative sampling latches can take a small signal near the switching threshold on their input, and over time amplify the input signal to a high or low logic level. Using a regenerative sampling latch in this fashion eliminates problematic low-frequency harmonics and requires a much shorter sampling time than sample-and-hold circuits because only a small differential signal on the input is needed by the internal regererenerative circuits. Furthermore, a properly designed sampling latch can consistently sample fast changing inputs and can thus be used to develop high-speed frequency synthesis systems.

An architecture with a sampling latch, which essentially behaves like an edge-triggered flip-flop, is shown in Figure 3.1. This architecture is the same as that of Figure 2.6, but with the divider replaced by a sampling latch that is operated by a sample clock.

### 3.2.1 Alias-Locked Loop Frequency Synthesis

Subsampling, whether with a sample-and-hold circuit or with a sampling latch, results in aliasing of the VCO frequency around the sampling frequency. A normal frequency divided VCO signal produces an output frequency that is related to the VCO frequency as follows:

$$f_{div} = \frac{f_{VCO}}{N} \tag{3.1}$$

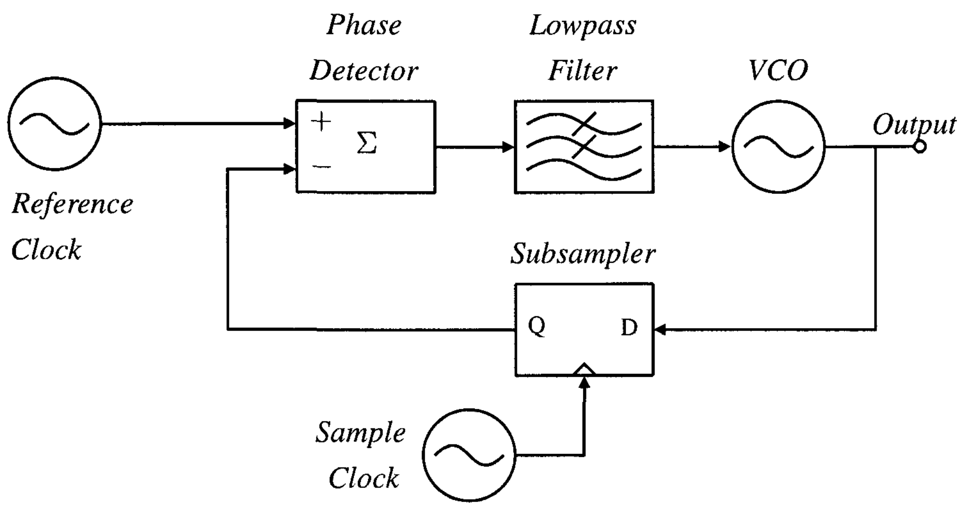

Figure 3.1: Block diagram of a frequency synthesis alias-locked loop

where $f_{div}$ is the frequency of signal at the divider output and $N$ the division ratio of the frequency divider. For a PLL that is locked to the reference signal and has a division ratio of $N$, the output frequency can be computed directly from the division ratio and reference clock frequency, $f_{ref}$:

$$f_{VCO} = f_{ref} \cdot N \tag{3.2}$$

When using a subsampling circuit, the output frequency from the aliasing circuit, $f_{alias}$, is instead determined by the sampling frequency $f_{sample}$, and is given by equation 2.4. An interesting result is that for an ALL system in the locked state, the output frequency of the VCO cannot be uniquely determined from a single sampling rate and reference frequency at the phase detector. For a VCO with a tuning range greater than the sampling frequency, there are in fact multiple VCO frequencies at which the ALL could lock. To overcome this ambiguity in VCO frequency, an approach with multiple sampling clocks is discussed in subsection 3.2.3.2. Referencing equation 2.4, when the ALL is implemented like a PLL and the feedback signal provides negative feedback the VCO frequency is related to the sampling rate and reference frequency by:

$$f_{VCO} = K \cdot f_{sample} + f_{ref} \tag{3.3}$$

where $K$ is some unknown positive integer, and the minimum and maximum frequency of $f_{VCO}$ are bounded by the frequency range of the VCO. Relating the VCO frequency to the reference clock, it is possible to determine an effective multiplication factor. A locked ALL frequency synthesizer multiplies the reference clock by an effective frequency multiplication factor, $L$, of:

$$L = round\left(\frac{f_{VCO}}{f_{sample}}\right) \cdot \frac{f_{sample}}{f_{ref}} + 1 \tag{3.4}$$

It is also possible for the VCO to operate at frequencies where the alias frequency would match the reference clock:

$$f_{VCO} = K \cdot f_{sample} - f_{ref} \tag{3.5}$$

but, in such a configuration the negative feedback would prevent the system from locking. For instance, with reference to equation 2.4, consider the VCO oscillating at a frequency given by 3.5. At this frequency, an increase in $f_{VCO}$ will decrease $f_{alias}$ and result in feedback that would increase the VCO frequency even further, steering the VCO away from operating at the frequency given by equation 3.5. If the ALL is built with positive feedback instead, the system would lock at VCO frequencies given by equation 3.5 and fail to lock at VCO frequencies given by 3.3.

## 3.2.2 Implications

It is immediately obvious from Figure 3.1 that an additional clock source is required to operate the sampling circuit, and, in total, two clock sources are required to operate the system. However, there is not necessarily a need for two stable and independent clock signals. The reference clock frequency is necessarily lower than the sampling clock, as per equation 2.5. It is therefore possible for the reference clock to be derived from the sample clock through for example frequency division. This would eliminate the requirement for two separate clock signals. Appropriate frequency ratios and phase relationships between the sample and reference clock are further investigated in chapter 5.

### 3.2.2.1 Feedback Signal Time Discretization

Equation 2.4 precisely describes the frequency of the signal produced by the analog output of a sample-and-hold circuit. Using a sampling latch with binary outputs, however, digitizes the sampler output and produces a square wave signal with clock edges discretized in time. Because the sampler changes output in response to the sample clock, these clock edges are necessary aligned with the sampling clock. Figure 3.2 shows a 1 Hz sampled by a regenerative latch with a sampling rate of 0.4 Hz, resulting in a 0.2 Hz square wave signal.

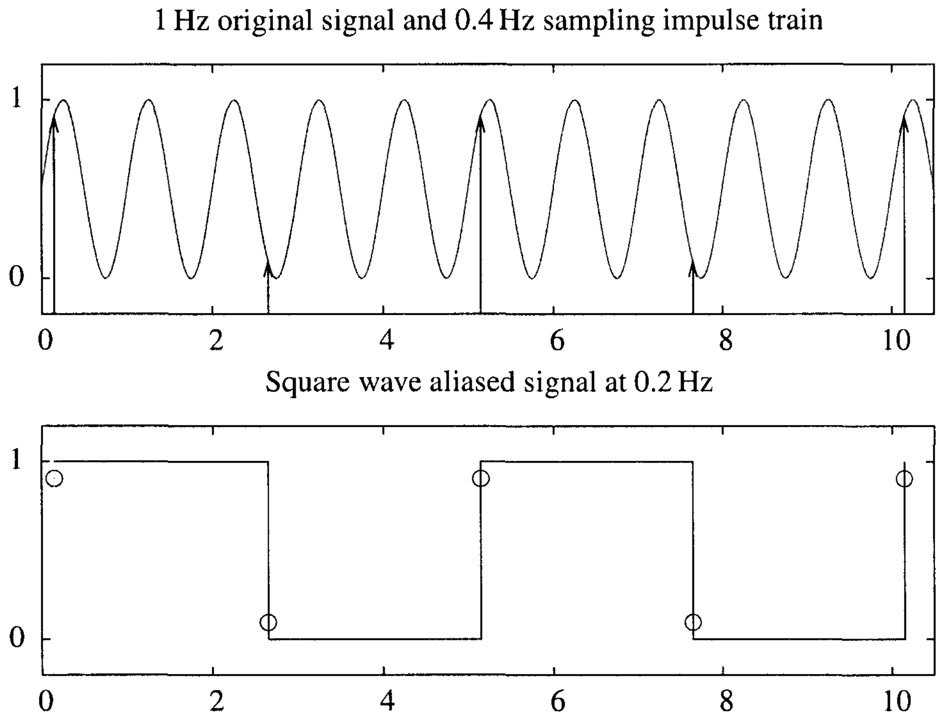

Figure 3.2: Graphical illustration of binary sampling, showing the original signal, subsampling impulse train and binary aliased signal

When the VCO is sampled by a latch with a binary output, the period of the aliased signal must be an integer multiple of the sampling period, since the output of the sampler only changes its output when it is activated by the sample clock. As a result, the instantaneous frequency of the aliased signal does not necessarily match the frequency given by equation 2.4. Although the square wave in Figure

3.2 is precisely the frequency predicted by equation 2.4, Figure 3.3 shows a case where the aliased signal is not the exact frequency predicted. In this figure, it can be observed that the period of the alias signal alternates between 4 s and 6 s, which does not match the predicted 5 s alias period.

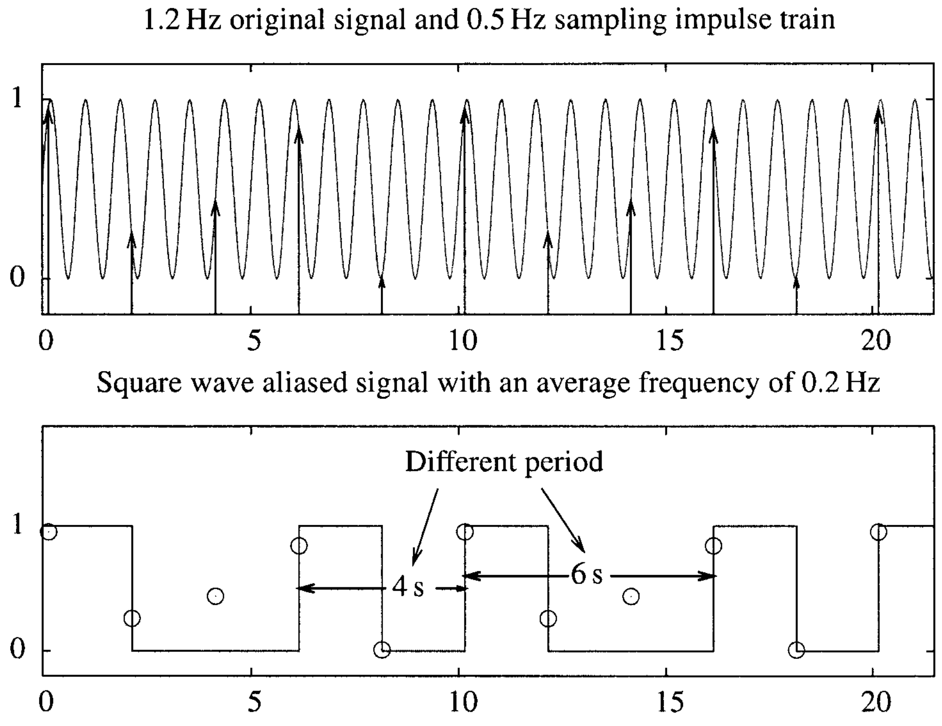

Figure 3.3: Graphical illustration of binary sampling, showing the original signal, subsampling impulse train and binary aliased signal

The long-term average frequency of the aliased signal does, however, match equation 2.4, assuming a constant sampling rate and steady VCO frequency. For the special case in Figure 3.2, where the period of the aliased signal is a multiple of the sampling period, the instantaneous frequency will not vary. From observation of a variety of alias signal and sampling periods, it appears that in the general case the frequency of the aliased signal will alternate proportionally between two frequencies, governed by the sampling period. This proportion appears to be such that the average frequency produced by subsampling will match equation 2.4.

Analyzing Figure 3.3 more closely, a 1.2 Hz signal is sampled at 0.5 Hz, where every sample with a value below 0.5 evaluates to 0, while those samples at 0.5

or larger evaluate to 1. The lowest predicted alias frequency is 0.2 Hz, which is equivalent to a clock period of 5 s. The sampling frequency is 0.5 Hz, however, which implies that the period of the signal at the sampler output must be a multiple of 2 s and cannot generate a clock period of 5 s. The resultant waveform at the sampler in fact alternates evenly between clock periods of 4 s and 6 s, in such a proportion that the clock period averages out to 5 s.

The side effects of discretization in time of the feedback signal also merit some consideration. When using a sampler with analog output levels, like the sample-and-hold circuits used in [39], the phase of the sampler output varies continuously. This contrasts with a sampling latch that produces a digital output with clock edges that are aligned on with the sampling clock. While a PLL with integer frequency division eventually reduces the phase error between the feedback signal and the reference clock to a value that is close to zero, the ALL proposed in this thesis never achieves true phase lock at the PFD. Although the phase error can be zero when the sampling clock is in phase with the reference clock, the next increment in phase error that would arise from a drift in the VCO frequency will be at least one sample clock period in magnitude. This means that the phase error will jump in increments of one sample clock period, causing constant corrections to the VCO frequency that show up as spurious tones in the output spectrum. This periodic correction behaviour, however, is not unlike that exhibited by bang-bang phase detectors or fractional-N synthesizers. Behaviour of bang-bang PLLs is analyzed in [36, 35], while a further analysis of the ALL as proposed in this thesis is provided in chapter 5.

#### 3.2.2.2 VCO Frequency Determination

Another interesting result, and potentially problematic, that arises when subsampling in the feedback path is that the exact frequency of the VCO signal is not uniquely determined by frequencies of the reference and sample clocks. Again, referencing equation 2.4 it is clear that multiple VCO frequencies, separated in frequency by integer multiples of $f_{sample}$, can give rise to the same alias frequency at which the ALL can lock. The number of possible frequencies at which the ALL locks will generally be limited by the tuning range of the oscillator. The next subsection present some potential solutions to uniquely determine the oscillating frequency of the VCO during operation of the system.

#### 3.2.2.3 Loop Filter Design

It can be noted that if traditional control system analysis were to be performed, the feedback division factor for an alias-locked loop would approximate to unity. This follows from equation 2.4, from which it is obvious that when the VCO frequency changes by some small amount, the frequency of the subsampler output will change by the same small amount. Intuitively, this also makes sense, since the frequency and phase of the VCO are not scaled in a sampling operation. Although the phase of the subsampler output is discretized, any phase change in the VCO signal that is larger than the smallest phase increment in the subsampler output results in an equivalent phase change in this subsampled output. This result contrasts with typical frequency dividers, where VCO phase and frequency are scaled proportionally to the division ratio. Since the phase of the VCO is not divided down in an ALL, calculations done for the purpose of determining the loop filter parameters should assume a frequency division factor of unity in the feedback path.

One additional parameter in loop filter design that requires some consideration is the location of reference spurs in the frequency spectrum that arise from the discretization of the phase error. If spurious tones are a concern, loop-filter design principles for bang-bang PLLs and fractional-N synthesizers can be used as a guideline. Both bang-bang PLLs and fractional-N synthesizers have the same discrete steps in phase-error as an ALL.

### 3.2.3 Variations

Several modifications can be made to the basic architecture presented above. Perhaps there is some benefit in combining a subsampling circuit with a frequency divider in the feedback path. And, as alluded to earlier, multiple sampling frequencies can be used to disambiguate the VCO frequency. Finally, many of the variations applied to standard PLL designs can also be applied to the ALL. In particular, the discrete nature of subsampling makes the ALL architecture a good candidate for digital filtering of the phase error.

#### 3.2.3.1 Sampler and Divider in Feedback Path

Although the basic architecture relies on just a sampling latch in the feedback path, if the alias frequency is still too high for a given sample frequency, frequency dividers can be inserted after the sampling unit. Such an arrangement is shown in Figure 3.4. One benefit introduced by a frequency divider is that it may reduce spurs in the output spectrum, since the regular repetition of two clock periods at the subsampler output can be eliminated if the modulus of the divider is an integer multiple of the period of alternation of the subsampler output. For instance, if the output of a subsampler remains at one frequency for three clock periods and produces a second frequency for the fourth cycle, a divide by four operation on the output of the subsampler would eliminate the phase error jump that would occur every fourth cycle without a divider in place.

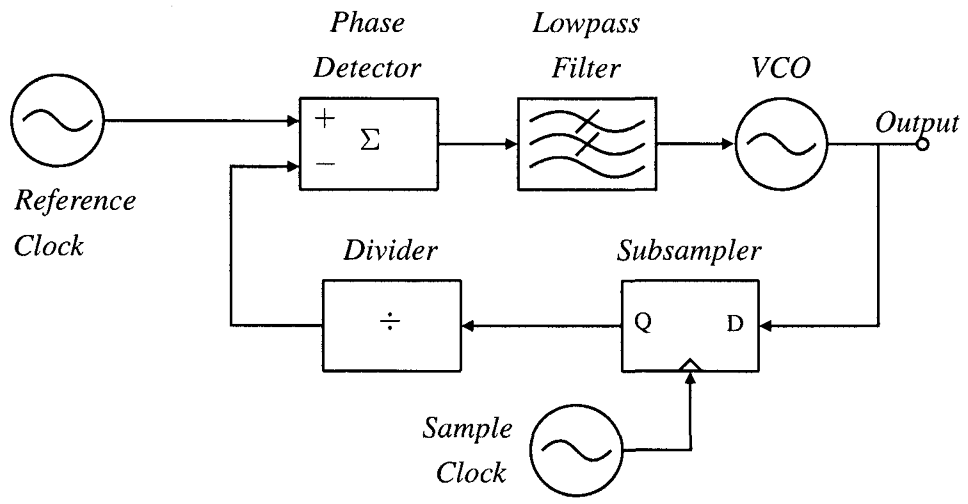

Figure 3.4: Block diagram of a modified frequency synthesis alias-locked loop

#### 3.2.3.2 Multiple Sample Frequencies

As mentioned in the previous subsection, the VCO frequency is not uniquely determined from the frequencies of the sample and reference clock. However, if the ability to change the sample clock is present, one approach that could be used to reduce the number of frequencies at which the VCO might be oscillating is to change the sample clock to another frequency. For instance, if a VCO signal is sampled at 1 Hz and the reference clock is 0.2 Hz, the possible VCO frequencies that could be synthesized, as per equation 3.3, are: 1.2 Hz, 2.2 Hz, 3.2 Hz, 4.2 Hz, etc. Now, if the reference frequency is kept the same and the sample frequency is changed to 0.6 Hz, the possible VCO frequencies are now: 0.8 Hz, 1.4 Hz, 2.0 Hz, 2.6 Hz, 3.2 Hz, etc. Combining the two sampling frequencies, the number of possible VCO frequencies are now 3.2 Hz, 6.2 Hz, 9.2 Hz, 12.2 Hz, etc. Having knowledge of the VCO tuning range and the reduced number of possible VCO frequencies possible with two sampling frequencies. The VCO frequencies at which the ALL locks with two different sampling frequencies, $f_{sample_1}$ and $f_{sample_2}$, and a reference frequency, $f_{ref}$, is given by:

$$f_{VCO} = K \cdot lcm(f_{sample_1}, f_{sample_2}) + f_{ref} \tag{3.6}$$

where $K$ is any positive integer and the function $lcm()$ finds the lowest common multiple of its arguments. Choosing two sample frequencies with a large common multiple such that there is only one solution to equation 3.6 within the tuning range can ensure that the VCO locks at the desired frequency. Of course, more than two sample frequencies can be used as well, and equation 3.6 can be extended for more than two sample frequencies.

#### 3.2.3.3 Digital Phase-Error Processing

Many of the modifications made to the basic architecture of frequency synthesis PLLs can also be applied to the basic frequency synthesis ALL outlined in this chapter. Because the ALL architecture proposed herein operates with discretized phase errors, a logical method of processing discretized data is to work in the digital domain. So, like many bang-bang PLLs [36], processing the phase error using digital filters may be worth examining. For instance, in an architecture with phase aligned reference and sampling clocks the discretized phase errors are limited to a small set of possible values. These phase errors can be kept in the digital domain and processed by a digital loop filter that controls a DAC connected to the VCO.

## 3.3 Sampling Circuit Requirements

The distinguishing component in an ALL architecture is the circuit that performs the subsampling operation in the feedback path. The analog sample-and-hold or ADC circuits described in previous papers and patents works well for low-frequency applications, but to sample high-frequency VCO outputs, a sampling latch with internal generation and a binary output is simpler and can generally operate at higher frequencies.

The desired behaviour for the sampling latch is like that of a D-type flip-flop: the sampling latch evaluates its input signal to a logic-low or logic-high value on a transition of a clocking signal and subsequently propagates this logic value to the latch output. Although much has been written on flip-flop circuits in the literature, most of these flip-flops are used as storage elements in digital circuits. Storage elements in digital circuits are optimized for figures-of-merit such as the power-delay product and circuit size, which are not of particular relevance for a single high-speed sampling unit. The design criteria that are imposed on high performance D-latch circuits found in retiming circuits of high-speed interfaces such as the serializer-deserializer chips in fibre optic and backplane communications systems, however, align more closely with the constraints placed on a high-speed sampling latch for an ALL. In D-latch retiming circuits power consumption and latency are less of a concern and greater emphasis is placed on high-speed operation. In the literature, retiming circuits have been reported capturing data at rates of 75 Gbit/s using high electron mobility transistor (HEMT) technology in a 0.13-μm III-V process [52] and 40 Gbit/s in 90-nm CMOS technology [53].

### 3.3.1 Sampling Latch Properties

Although the sampling latch should behave as a D-type flip-flop, several constraints normally placed on flip-flops are relaxed for the sampling latch in an ALL. Specifically, the clock-to-output propagation delay through the sampling latch does not need to be minimized. Furthermore, the reset time during which the latch is prepared for the next sampling operation can take most of one period of the sample clock. This means that, unlike in a frequency divider or prescaler and depending on the frequency of the sample and input signals, the clock-to-output and reset time of the latch circuit can take multiple cycles of the VCO signal. Furthermore, with a periodic input signal from the VCO, the setup and hold time for the sampling latch are irrelevant, but the latch circuit must consistently and predictably sample the input signal from cycle to cycle in such a fashion that the phase of the input signal is properly represented at the latch output. In fact, the sampling window of the latch can span more than one VCO cycle, as long as the VCO phase is consistently passed to the latch output. Finally, any DC offset in the sampler that biases the sampler toward a particular logic level is not problematic, as long as the sampler outputs both ones and zeros and is still periodic and representative of the periodicity of the input signal. These severely loosened restrictions on the latch behaviour allow for more design freedom with a focus on the capability to sample very fast changing inputs.

Because the sampling latch is not operated at the frequency of the input signal, but rather at the frequency of the sampling signal, power consumption will be reduced compared to traditional high-speed prescaler and divider circuits. To estimate power consumption, the activity factors of the signals in a frequency divider can be compared with a sampling latch. A series of toggling flip-flops in a counter-based frequency divider toggle at 1/2, 1/4, 1/8, 1/16, etc. of the VCO frequency, adding up to an average activity factor of 1. For a sampling latch, the activity factor is driven by the sampling clock instead. Assuming that the sampler produces the highest alias frequency it is capable of, the worst case activity factor is 1/2 of the sampling frequency. Since the sampling frequency can be an order of magnitude lower than the VCO frequency, the power consumption for the feedback circuits in ALL is indeed much lower than that of a frequency divider based PLL for equivalent VCO frequencies.

### 3.3.2 Duty Cycle Correction

One problem in high-speed sampling that can be detected and potentially corrected is a DC offset in the sampling latch that biases the sampler output to a particular logic value, or in the worst case, causes the sampler output to be stuck at a logic low or logic high value. Many high speed systems use differential circuits, but any imbalance or mismatch in the symmetry of the circuit will cause such DC offsets. Careful consideration of geometry choices and their impact on device matching during the layout of a circuit does aid in minimizing the effects of device and parasitic mismatch. But, as device geometries shrink, practically unavoidable random variations are becoming more pronounced. In an attempt to correct latch circuits with threshold problems, a feedback mechanism is proposed that adjusts the input switching threshold of a circuit, counteracting any DC offsets at the input.

The detection mechanism used to adjust circuit input thresholds relies on a deviation of the duty cycle of the latch output. In a perfectly balanced sampling latch, the subsampling operation on the VCO signal should result in a square wave output signal with a duty cycle that tends to 50 %. Several methods for duty cycle measurement have been reported [54], using analog, digital, or mixed digital/analog approaches. Most of the methods for measuring the duty cycle in previously published work, however, rely on complex circuits that are needed for systems where the frequency of the duty-cycle signal being measured is on the same order or higher than the system clock frequency. When measuring the duty cycle of the latch output signal, however, the sampling clock can be used as the relevant clock, because the frequency of latch output signal is lower than the system (sampling) clock. This greatly simplifies measuring the duty cycle, since the sample clock can then be used to sample the output values of the latch.

#### 3.3.2.1 Duty Cycle Measurement Algorithm

The proposed method to measure the duty cycle calculates a running sum of the output values produced by the sampling circuit. Logic-high values at the sampler output increment this running sum, while logic-low values decrement the running sum. If the duty cycle of the sampler output is 50 %, the running sum will tend to zero. Any DC bias that increases the probability of logic-high output will cause the running sum to increase over time, while a DC bias toward logic-low output causes the running sum to decrease. The rate of increase or decrease will be proportional to the severity of the DC bias. Over a single period of the alias signal at the sampler output the running sum will go up and down, so small changes in the running sum value should be discarded. In a binary representation of the running sum, this would mean discarding a number of low-order bits in the sum. The high-order bits can then be used to adjust the threshold levels for the sampling circuit. The mechanism through which to achieve this is dependent on the specific circuit used. In a differential circuit, for instance, an extra biasing transistor could be inserted in each differential branches of the circuit. These biasing transistors could then be individually controlled to alter the symmetry in the circuit.

The proposed duty cycle correction scheme creates a feedback system that continually adjusts the biasing transistors so that the duty cycle is as close to 50 % as the DAC quantization permits. However, as with any feedback system, such an arrangement may be unstable or oscillate, as it attempts to achieve a 50 % duty cycle. This target duty cycle is not achievable under all conditions, however, since the duty cycle of the signal is determined by the number of sampling impulses per alias clock period. If the number of sampling impulses per alias clock period is odd, a perfect 50 % duty cycle is not possible, and the threshold level will continually vary, and thus continually alter the phase relationship between the sampler input and output signals. This is undesirable, as it affects the ability of an ALL to maintain lock. An improved duty cycle correction system with hysteresis will only adjust the input threshold of the sampler if the duty cycle deviates by more than a specified margin, e.g. $\pm 10$ %. To implement this acceptable margin, the biasing transistors only need adjusting when the rate of increase or decrease of the running sum exceeds a specified amount. Specific circuit solutions, such as those in Figure 4.16, to achieve this type of duty cycle control mechanism are discussed in chapter 4.

## 3.4 Summary

This chapter has provided an overview of the operating principles that allow an ALL to perform frequency synthesis. The mathematical relationships between input and output frequencies of an ALL system have been presented. An overview of a sampling latch based architecture has been provided, along with several possible variations that could be made to this architecture. The predicted effects of subsampling in the feedback path has been discussed, along with specific considerations that must be taken when designing an ALL frequency synthesizer. Finally, the constraints and desired behaviour expected from the subsampler are described, together with a method to tune out any DC bias in a sampling latch.

Replacing the high-speed frequency dividers by a sampling latch relaxes the constraints on the circuits in the feedback path. This has the potential to allow for the development of faster frequency synthesizers than what is possible with current state-of-the-art frequency dividers. An additional benefit of using a subsampling latch is that the power consumption is significantly reduced when compared to a frequency divider. One side-effect of using a sampling latch that discretizes the phase error is that measures of jitter and phase noise spectra for ALL-based frequency synthesizers are, although similar to bang-bang PLLs or fractional-N synthesizers, most likely worse than integer-N frequency divider PLLs.

# Chapter 4

# Sampler Circuit Design and Simulation

## 4.1 Overview

The operation that limits the maximum frequency of modern high-speed frequency synthesis PLLs is the frequency division that needs to be performed in the feedback path [10]. In frequency dividers and prescalers, the first stage division circuit is connected directly to the oscillator and must operate at very high frequencies. In contrast, the sampling latch generates a new output signal at the frequency of the sampling clock, rather than half the frequency of the VCO clock. Although it must still consistently sample a very high speed input signal, the loosened constraints outlined in chapter 3 allows for more freedom in design trade-offs. Section 4.2 presents a number of potential sampling latch circuits and evaluates their suitability as subsampling unit in a frequency synthesis ALL through simulations in a modern CMOS process. Section 4.3 concludes with a description of threshold correction circuits for sampling latches, again providing simulation demonstrating their functionality.

## 4.2 Latch Circuits

In this section, several latch circuits will be evaluated for their suitability as a high-speed subsampling latch. Each latch circuit is presented and subsequently evaluated through analysis and simulation. The suitability of a sampler is determined by verifying that the candidate circuit produces the correct aliased output signal when subsampling a periodic input signal.

## 4.2.1 Latch Performance Evaluation Methodology

To evaluate the suitability of each latch, two simulations are performed on the candidate circuits. The first simulation consists of a test bench that aims to determine the maximum input frequency at which the sampling latch still produces the correct alias frequency. The second method for evaluating latches defines an input sensitivity function for each latch by modelling the latch as a linear correlation followed by a binary decision. This analysis provides some insight on the size of the sampling window, and provides a relative basis for comparison of minimum detectable pulse widths and that the sampling window need not be shorter than the input cycle time.

### 4.2.1.1 Maximum Frequency Test

The first evaluation strategy aims to replicate the conditions that a real sampling circuit operates under. To do so, a simulation test bench has been configured with a sinusoidal input that models the VCO signal and a square wave sample clock that triggers the sampler. The time between rising edges at the sampler output is plotted versus time, and this period is compared to the alias frequency predicted by 2.4. A block diagram of the test bench is shown in Figure 4.1.

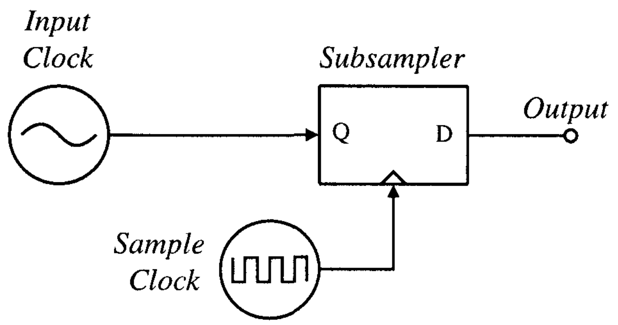

Figure 4.1: Block diagram of a test bench for a sampling latch

All simulations are done at the transistor level in a 90-nm general purpose (GP)

CMOS process using Spectre as the simulator. The simulation parameters, design kit version and software simulator version are specified in appendix A. A simulation to determine the performance of the logic SPICE models in the design kit shows that the $f_T$ is optimistically high at about 200 GHz, while publications with measured data indicate that a 90-nm GP CMOS process has an $f_T$ and $f_{max}$ of about 120 GHz and 170 GHz, respectively [53]. This is not unexpected, since the design kit is not intended for RF circuit design, and the device models do not include effects such as gate resistance. Nonetheless, simulations can still be used to relatively compare the merits of sample circuit architectures, even if the predicted performance is likely optimistic when compared to actual implementations.

To determine the maximum frequency that can be sampled by a latch, the input clock frequency is increased until the latch fails, meaning that the period at the output of the latch no longer matches the alias frequency predicted by equation 2.4. Since a latch is likely to fail around its input switching threshold, failures are best identified by verifying that the sampler functions deterministically around the input switching threshold. When the phase relationship between the input clock and sample clock has been found at which the sampler operates near the switching threshold, it is possible to run a parametric sweep that varies the phase by small increments across the phase at which the input switch threshold lies. This type of parametric sweep, however, does not take into account hysteresis effects that may be present in the latch. Instead of a parametric sweep, the input clock and sample clock are configured to ensure a large number of sample points across the whole phase of the input clock signal. To achieve this, the predicted alias frequency is made small in comparison to the sample clock frequency, thereby forcing a large number of samples to be taken across each period of the alias signal, which corresponds to the same number of samples taken across one cycle (a phase of $2\pi$) of the input clock. Although a latch can be similarly indeterministic around the input switching threshold for alias frequencies that are close to the sampling frequency, it is much harder to identify these failures in simulation.

Using a low alias frequency, however, is not useful to determine any problems arising from hysteresis or an input switching threshold that is high or low. A low input switching threshold, for instance, will cause the duty cycle of the output signal to be high. When the alias frequency is close to the sample frequency, there are only a small number of sample points per alias period and a low input threshold may in fact cause the output to be permanently stuck high. For this reason, the samplers are evaluated at a low and high alias frequency. The high alias frequency detects problems with high or low switching threshold problems and hysteresis, while a low alias frequency will find problems related to output signal indeterminism around the input switching threshold.

For the simulations in this section the sampling frequency has been set to 1 GHz and the input frequency is configured to produce a low alias frequency of 1 MHz and a high alias frequency of 100 MHz, which implies an input frequency that is given by equation 3.3 or 3.5 and therefore must satisfy $K \pm 0.001\,\text{GHz}$ or $K \pm 0.1\,\text{GHz}$, where $K$ is any positive integer. The circuit is simulated for at least 1010 µs or 20 ns, depending on the alias frequency, and is sufficient to observe one clock period of the alias signal. To replicate real-world conditions more closely, the transients analysis is run with transient noise enabled. The noise parameters are based on recommendations in [55], although loosened to keep simulation times reasonable at high frequencies. The transient noise maximum frequency is set to 1 THz, with the minimum noise frequency at 10 MHz, and includes both white (thermal) noise and flicker noise. These noise parameters are specified along with the simulation parameters in appendix A.

#### 4.2.1.2 Latch Sensitivity Function Analysis

Although the maximum frequency simulation is a good predictor of performance for various latch circuits, the determination of the maximum frequency requires long-duration simulations at high frequencies and is very time consuming. In a second method we attempt to define an sensitivity function for each latch around the sampling instant by assuming the latch responds as a linear correlation of the input and the latch sensitivity function (LSF).

D. F. Williams et al. [56] described a similar approach for evaluating sample-and-hold circuits of oscilloscopes. In their paper they describe a method in which they treat their sample circuit as a linear time-invariant system, apply a series of input signals that mimics an impulse response and measure the output signal. Using the output signal they define an *impulse response* function for the sampling operation that appears to work reasonably well in predicting the circuit response for a variety of input signals.

To evaluate sampling latches we assume that the latch circuit is linear, and attempt to define a latch sensitivity function. For a latch with an input switching threshold (logic threshold) of $V_{LT}$, input function $V(t)$, and LSF function $LSF(t)$, the output $Q$ of the latch is given by

$$Q = sign\left(\int_{-\infty}^{+\infty} (V(t) - V_{LT}) \cdot LSF(t)dt\right) \tag{4.1}$$

where the output of the $sign()$ function can be interpreted to evaluate to a logic low or logic high value normally seen at the output of a latch circuit.

Unfortunately the non-linearity of the $sign()$ operation prevents a simple reconstruction of the LSF from the output signal. Instead, to determine the LSF, a large number of simulations are run, and this data is combined to reconstruct the LSF. Two different plots are generated. For the first plot the latch circuit is stimulated by a sinusoidal pulse at the input. The time offset between the centre of the pulse and sample clock are varied, and for each offset step the minimum width pulse required to trigger the latch is found. Plotting the inverse of the pulse width at the switching threshold versus the offset time produces something akin to the LSF. With the LSF defined for each sampling latch, a relative comparison between latches can be made on the basis of the LSF. One drawback of this approach is that these plots do not give a clear indication of the sensitive time window. This is because a sufficiently wide pulse at a far offset will still overlap the sampling window of the latch. The generated plots such as the one shown in Figure 4.10 therefore show sensitivity reaching to very large offset values.

To address this shortcoming, a second sampling window sensitivity plot that is more indicative of the sampling window width is generated. The data for this plot is collected in a similar manner. Rather than using sinusoidal pulses, square wave pulses are used instead. These square pulses have rise and fall times that are equivalent to the slew rates of signals at frequencies on the order of the $f_{max}$ of the process. The simulation data is then analyzed to find the minimum pulse width required to trigger the latch. For the offsets at which the minimum pulse triggers the latch, data points with values inversely proportional to the minimum pulse width are used. To determine the sensitivity of the latch with offsets outside the peak sensitivity, the minimum pulse is delayed by one unit of the plotting time step. At this delayed offset using the minimum pulse width will cause the latch to no longer capture the pulse. The pulse width is then increased by maintaining the offset of the rising edge of the pulse but delaying the falling edge of the pulse. The pulse width is increased until the latch registers a logic high again. For this offset increment, the inverse of the triggering pulse width is plotted. This is repeated until an offset is reached at which there is no pulse width wide enough to trigger the latch. This point defines the edge of the sampling window after which no data is captured. The same steps are repeated in reverse to generate the data points prior to peak sensitivity, advancing the offset by the plotting time step and finding the minimum pulse width required to trigger the latch at that particular offset.

### 4.2.2 Standard Cell D Flip-Flop

To get a baseline performance figure for latch circuits, a master-slave D flip-flop from the digital standard cell library included in the design kit is simulated.

#### 4.2.2.1 Circuit Design

The standard cell D flip-flop from the 90-nm design kit used for simulations is *FD1QSVTX1*. The *FD1QSVTX1* cell is a rising-edge triggered D flip-flop with Q output, uses standard threshold transistors and has a normalized output drive of one. The design is a standard master-slave flip-flop built with inverters and transmission gates, and is shown in figure 4.2

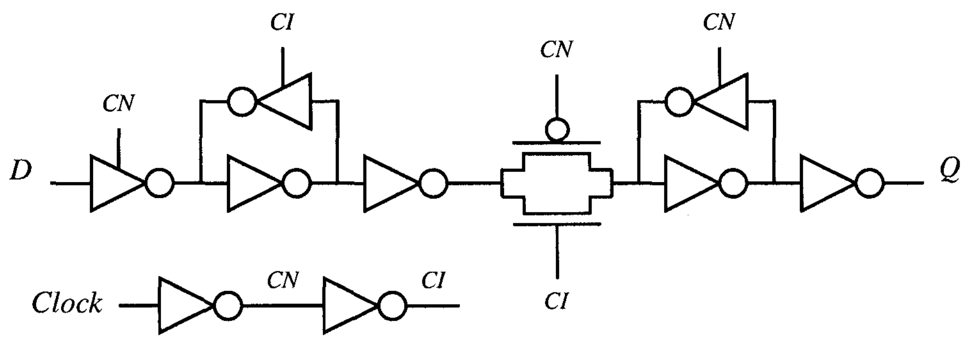

Figure 4.2: Standerd cell master-slave D flip-flop circuit

### 4.2.2.2 Circuit Simulation

Simulating the standard cell D flip-flop and increasing the input frequency beyond 4 GHz shows that the duty cycle of the alias signal begins deviating from 50 %. At 14.001 GHz the input signal has the proper alias frequency, albeit with a very high duty cycle. At 14.1 GHz, however, the circuit produces a steady logic high value at the output. Thus, the baseline figure for comparison for other latch architectures is approximately 13 GHz for the standard cell D flip-flop. The simulation results with the high and low frequency alias signals are shown in Figures 4.3 and 4.4.

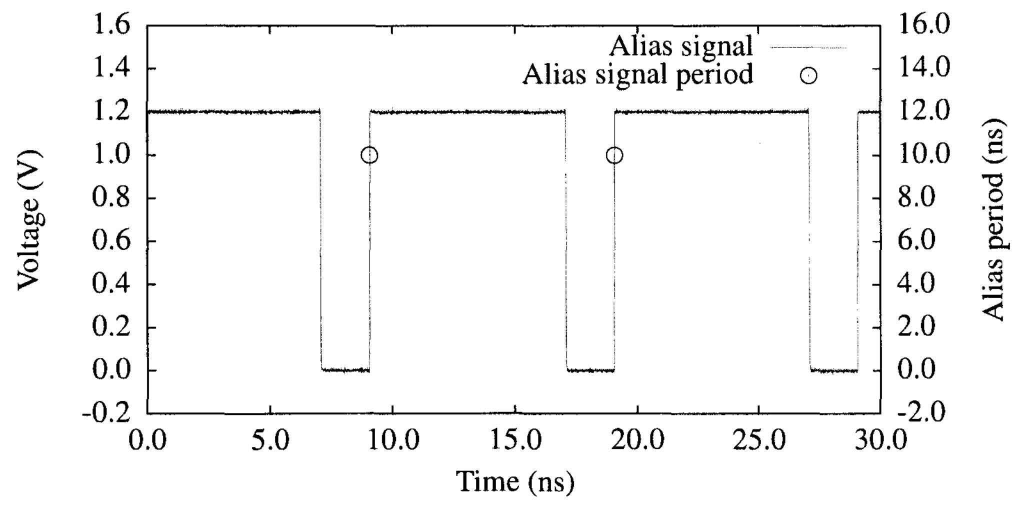

Figure 4.3: Alias signal for standerd cell D flip-flop subsampling simulation with a 13.1 GHz sinusoidal signal sampled at 1 GHz

To determine the LSF for the standard cell D flip-flop, the first step is to deter-

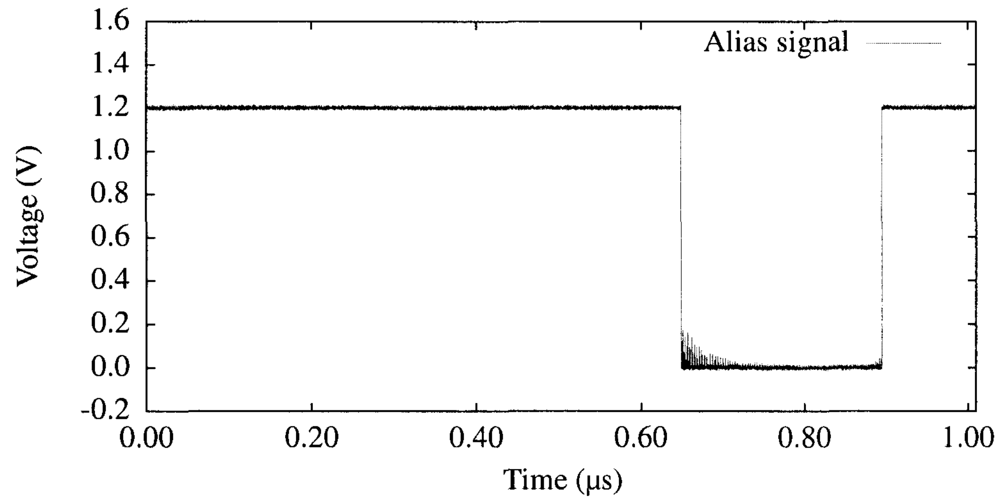

Figure 4.4: Alias signal for standard cell D flip-flop subsampling simulation with a 13.001 GHz sinusoidal signal sampled at 1 GHz

mine the switching threshold and the minimum pulse width that can be detected. Simulations reveal that for a supply voltage of 1.2 V the input switching threshold is approximately 0.553 V. A short parametric sweep of input pulse widths and offsets reveals that the minimum pulse width required to trigger the flip-flop is 23 ps, with the centre of the pulse arriving with an offset between -7 ps and -5 ps before the rising edge of the sample clock. The parameter space is swept for pulse widths up to 50 ps, and we find that the minimum and maximum offsets of the centre of the pulse for which the latch still captures data varies from -46 ps to 9 ps. The LSF plot is shown in Figure 4.5 and reveals peak sensitivity around 6 ps before the sampling clock edge arrives. The latch sensitivity drops sharply after peak sensitivity in the latch.

Plotting the sampling window sensitivity shows that any data on the standard cell D flip-flop input after -2 ps is not captured. Peak sensitivity for a square input pulse is found for a pulse width of 20 ps between -5 ps and -4 ps. The sampling window cut-off is not very sharp prior to peak sensitivity, with wide input pulses affecting the latch output up to 20 ps ahead of peak sensitivity. The sampling window sensitivity plot is shown in Figure 4.6.

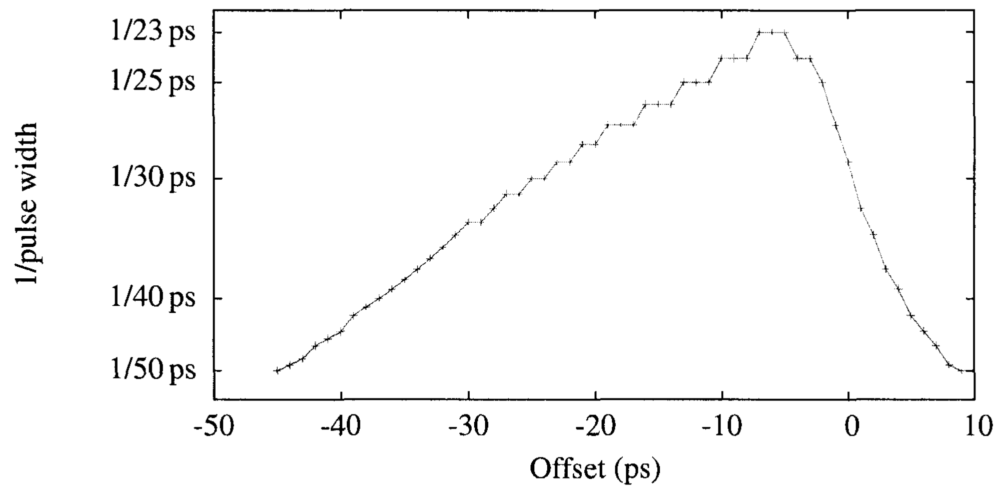

Figure 4.5: Latch sensitivity plot for standard cell D flip-flop

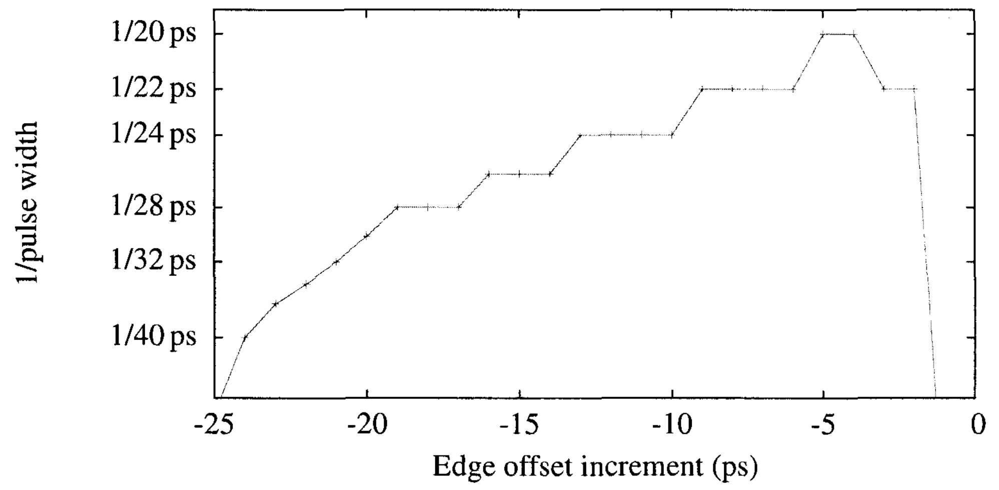

Figure 4.6: Latch sampling window plot for standard cell D flip-flop

### 4.2.3 Differential Pass-Transistor Pulsed Latch

An improvement on the single-ended standard cell D flip-flop can be made by using a differential sampling latch design. Inspired by the sense-amplifier circuits used in computer memories, the differential pass-transistor pulsed latch (DPTPL) [57] only requires a small input swing on the differential signal, which is then amplified internally and produces a rail-to-rail voltage at the sampler output.

#### 4.2.3.1 Circuit Design

The DPTPL topology, shown in Figure 4.7, is built on two cross-coupled inverters that provide regenerative feedback using transistors M1 through M4. The circuit shown in Figure 4.8 is used to generate a pulse that creates a brief sampling window by enabling transistors M7 and M8. The inverters in the pulse generator circuit are sized to minimize the pulse length while ensuring that the sampling circuit remains functional across PVT variations. During the pre-charge phase, when the clock signal is low, transistors M5 and M6 pull both internal nodes to $V_{DD}$. A gated set-reset (SR) latch at the output of the sampler holds the values when the internal nodes of the cross-coupled inverter go to $V_{DD}$ during the pre-charge phase. When the clock signal goes high transistor M9 is enabled and the pulse generation circuit creates a sampling window with a duration that is on the order of 20 ps in 90-nm technology. The differential signal between the D and $\overline{\text{D}}$inputs is then amplified through the cross-coupled inverters. All transistors drawn in Figure 4.7 are the smallest size that does not give rise to dog-bone shaped transistors, which corresponds to a drawn width-over-length (W/L) gate dimensions of 0.2 μm / 0.1 μm in the 90-nm process used for simulations.

#### 4.2.3.2 Circuit Simulation

To simulate the DPTPL circuit the test bench from Figure 4.1 is modified to produce a differential signal, producing a rail-to-rail swing of two sinusoidal input signals that are 180 degrees out of phase. When configured to produce an alias frequency of 100 MHz, the DPTPL circuit functions correctly at up to 179 GHz. However, changing the input signal for an expected alias frequency of 1 MHz, the circuit produces a noisy result near the switching threshold that requires filtering at input frequencies higher than 26 GHz. This failure mode is demonstrated in Figure 4.9. From this figure it can be observed that the latch does not sample consistently when it is operated near the switching threshold. Instead of a single edge as the input crosses the switching threshold, the latch output toggles between logic high and logic low values before it settles.

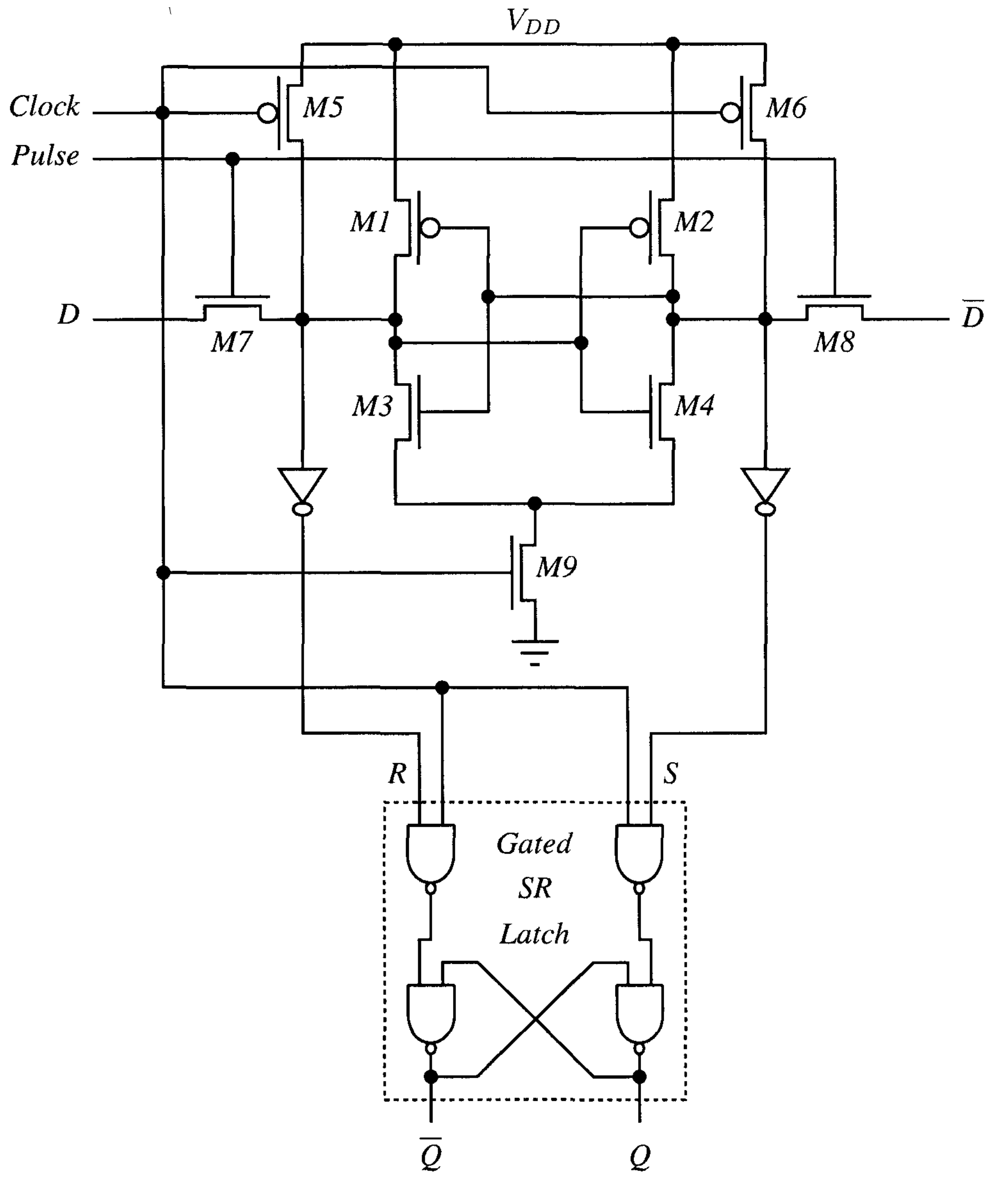

Figure 4.7: Differential pass-transistor pulsed latch circuit

Simulating the DPTPL circuit with transient noise and a target alias frequency of 100 MHz, the maximum frequency at which the circuit samples correctly is reduced to 100.1 GHz. Simulating with an alias frequency of 1 MHz, however, reveals that the DPTPL circuit produces noisy results near the switching threshold at frequencies as low as 4 GHz when transient noise is enabled, and again with a failure behaviour that is similar to that depicted in Figure 4.9.

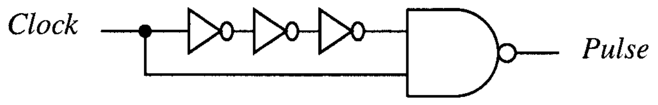

Figure 4.8: Pulse generator circuit for pulsed latch

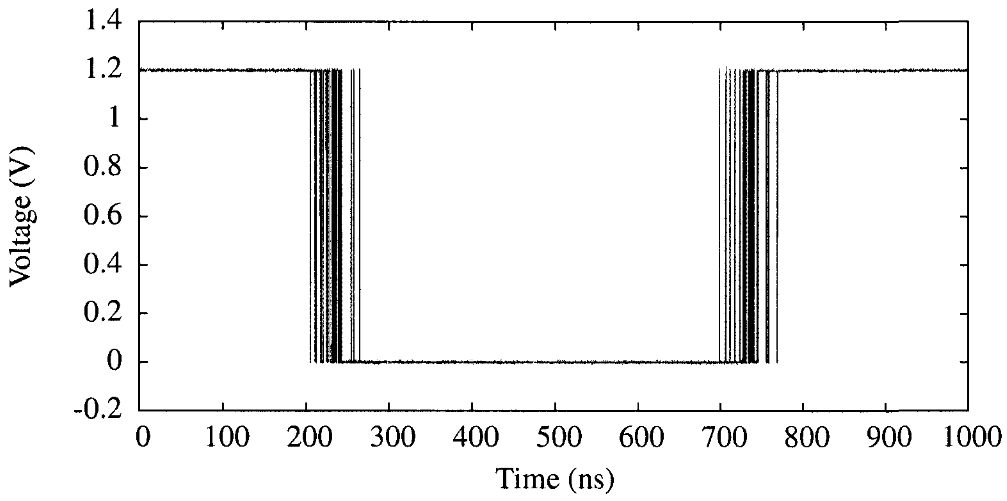

Figure 4.9: Toggling alias signal at the switching threshold of a DPTPL subsampling circuit, simulated with transient noise and a 58.001 GHz sinusoidal input signal sampled at 1 GHz

As a differential circuit, the DPTPL does not have switching threshold like single-ended circuits do, and even the smallest difference between the input signals should consistently evaluate to a specific logic level. To perform the LSF analysis for the DPTPL, the same approach as for the standard cell D flip-flop is taken. Initially the input differential signal is at $-V_{DD}$, resulting in a logic low output. Two differential cosine shaped pulses that reach a peak-to-peak value of $V_{DD}$ on the latch input should then cause the latch to produce a logic high value when sampled. A parametric sweep reveals that the minimum detectable pulse width is 9 ps, with the centre of the input pulse offset from the rising edge of the sampling clock between 27 ps and 32 ps. The DPTPL is thus sensitive to pulse widths much narrower than the standard cell D flip-flop. Repeating the same parametric sweep for pulse widths up to 50 ps it is found that the minimum and maximum offsets at which the DPTPL still captures the pulse are 5 ps and 54 ps, respectively. The LSF plot is shown in Figure 4.10, and shows a much more symmetric function, with a broad peak sensitivity around 29 ps. The large positive offset for peak sensitivity is due to the fact that there is a considerable delay between the arrival of the rising edge of the sampling clock and the generation of a sampling pulse that enables the pass transistors in the latch.

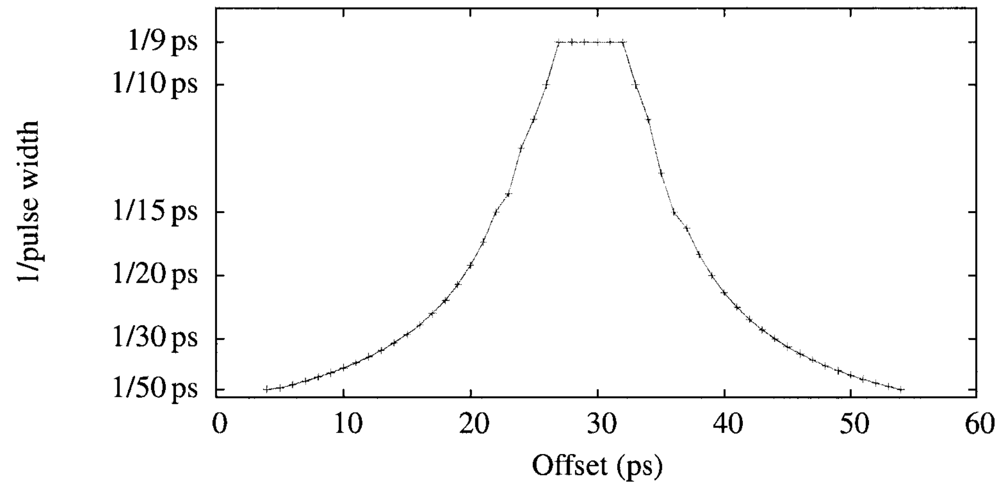

Figure 4.10: Latch sensitivity plot for DPTPL

The sampling window sensitivity, shown in Figure 4.11, for the DPTPL is much narrower and sharper than the standard cell D flip-flop, extending from approximately 26 ps to 33 ps. Peak sensitivity lies between 30 ps and 31 ps, with the smallest square pulse detected at 8 ps. The latch is slightly more sensitivity to input arriving prior to peak sensitivity.

#### 4.2.3.3 Latch Hysteresis Buffer

To overcome the toggling behaviour at the output of the DPTPL at low sample to alias period ratios a circuit has been designed that eliminates the high frequency noisy toggling at the output. Since the frequency of this toggling behaviour is much higher than the alias frequency, a low-pass filtering approach that eliminates these high frequency components can be used. Such a low-pass filter could be implemented using an analog approach or could the filtering can be done in the digitally

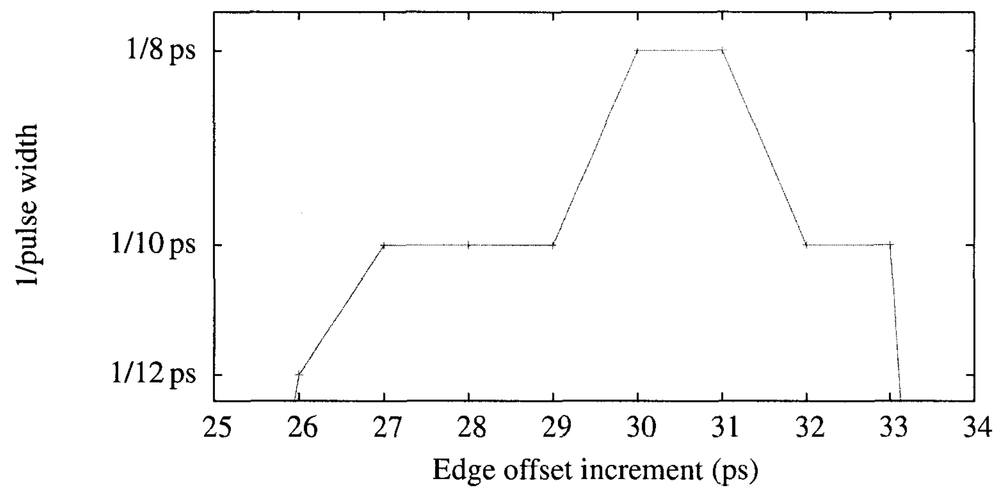

Figure 4.11: Latch sampling window plot for DPTPL

using digital signal processing techniques.

In a simple but effective implementation for this low-pass filtering operation, we designed a hysteresis buffer that maintains a history of previous sampling circuit outputs and compares them with the current sampler output. If these values are in agreement, the current sampler value is stored in the output buffer, otherwise the output buffer value remains unchanged. The choice for the number of previous sampler output values stored and compared with the current value determines the maximum duration for a toggle before it affects the output. From simulations the pulse duration of toggles is generally one or two sample clock periods, thus a reasonable number of previous samples stored in the history for a sampling to reference clock frequency ratio of 1000 is two. A more generic circuit for an arbitrary history length, and a larger range of alias frequencies, is depicted in Figure 4.12.

Although the hysteresis buffer circuit presented works well for low alias frequencies, it can prevent proper operation at higher frequencies if the buffer history is too long. A separate frequency detector could be used that measures long term frequency trends of the aliased signal and selectively adapts the history length of the hysteresis buffer to a level appropriate for a given alias frequency.

With the hysteresis buffer connected to the sampler circuit output the test bench

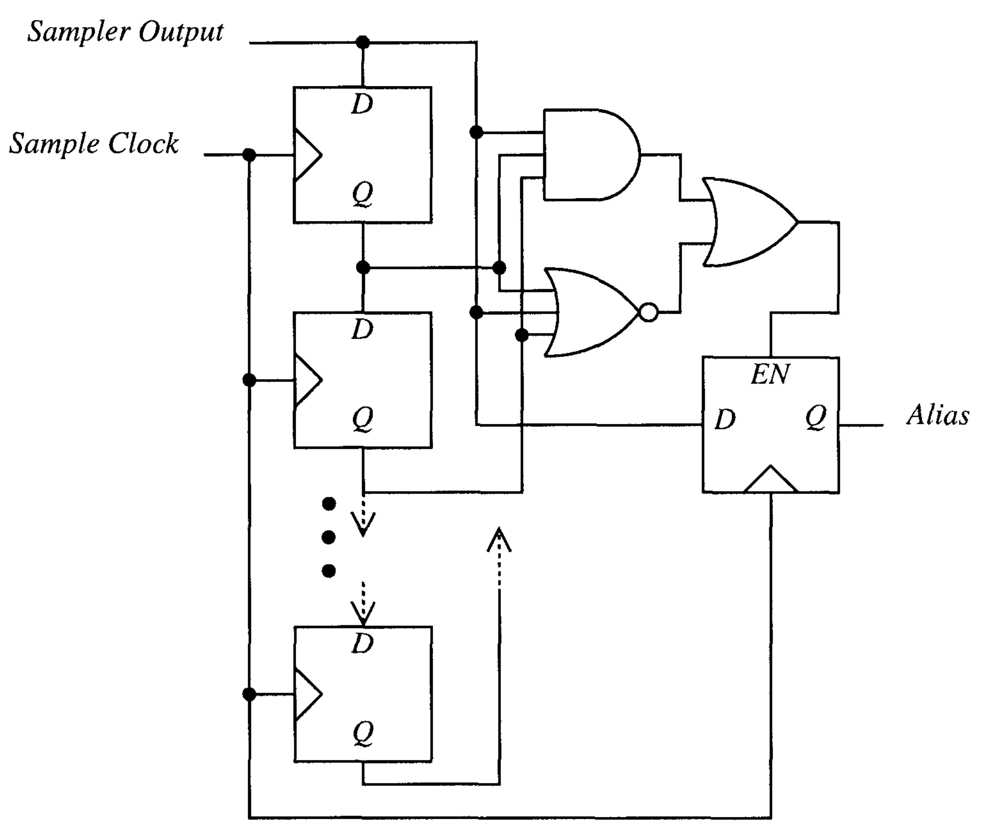

Figure 4.12: Hysteresis buffer for sampling latches.

is simulated again to determine the maximum input frequency with the hysteresis buffer. With an expected alias frequency of 1 MHz, the maximum input frequency is approximately 180 GHz. Adding transient noise, however, reduces the maximum input frequency to about 58 GHz.

A more sophisticated approach could likely be used extend the maximum frequency, depending on the behaviour of the latch output. The hysteresis buffer proposed suffices for oscillations that are rapid enough to be eliminated by the latch buffer. More advanced signal processing could be used to extract the clock period from different toggling or noisy behaviour. For instance, the duty cycle of the toggling behaviour may be dependent on the phase of the alias frequency. The high speed latches we evaluated tend to fail at high frequencies by generating a toggling signal at half the sample frequency, leaving no useful information for the extraction of the alias frequency. Other latch topologies, however, may have high frequency behaviour that still allows for the extraction of the alias frequency through a sufficiently sophisticated signal processing algorithm.

### 4.2.4 Sense-Amplifying Latch Circuit

To avoid relying on a pulse generating circuit, which is difficult to design for consistent behaviour across PVT variations a sense-amplifying flip-flop (SAFF), which was first proposed in [58], is evaluated.

#### 4.2.4.1 Circuit Design

Again, like the DPTPL, this latch uses the regenerative property brought about by internal positive feedback to amplify a small differential swing on the input to a rail-to-rail voltages at the output. Similar to the DPTPL design, the S and R outputs of this sampler connect to a gated SR latch that holds the values when the clock goes low. Unlike the DPTPL design, however, a SAFF does not use pass transistors controlled by pulse generation circuit. Instead, the inputs are connected to the gates of transistors M7 and M8. Using the transistor gate rather than the drain provides a higher input resistance. Again, pull-up resistors M5 and M6 pull the internal nodes to $V_{DD}$, while the drain voltages of transistors M7 and M8 are equalized by transistor M10, which remains enabled at all times [59]. The data sampling occurs when clock goes high, causing M9 to turn on and allowing transistors M7 and M8 to amplifying the differential input signal that in turn causes the latch to capture the input signal. Once the internal nodes have settled, changing input values applied to transistors M7 and M8 no longer affect the output of the latch.

#### 4.2.4.2 Circuit Simulation

Simulating the latch with an expected alias frequency of 100 MHz, the maximum input frequency of the latch is approximately 200 GHz. Changing the input signal to produce an alias frequency of 1 MHz, however, reduced the maximum input frequency to 50 GHz. Simulating the SAFF with transient noise and a 1 MHz predicted alias frequency reduces the maximum input frequency to 3 GHz.

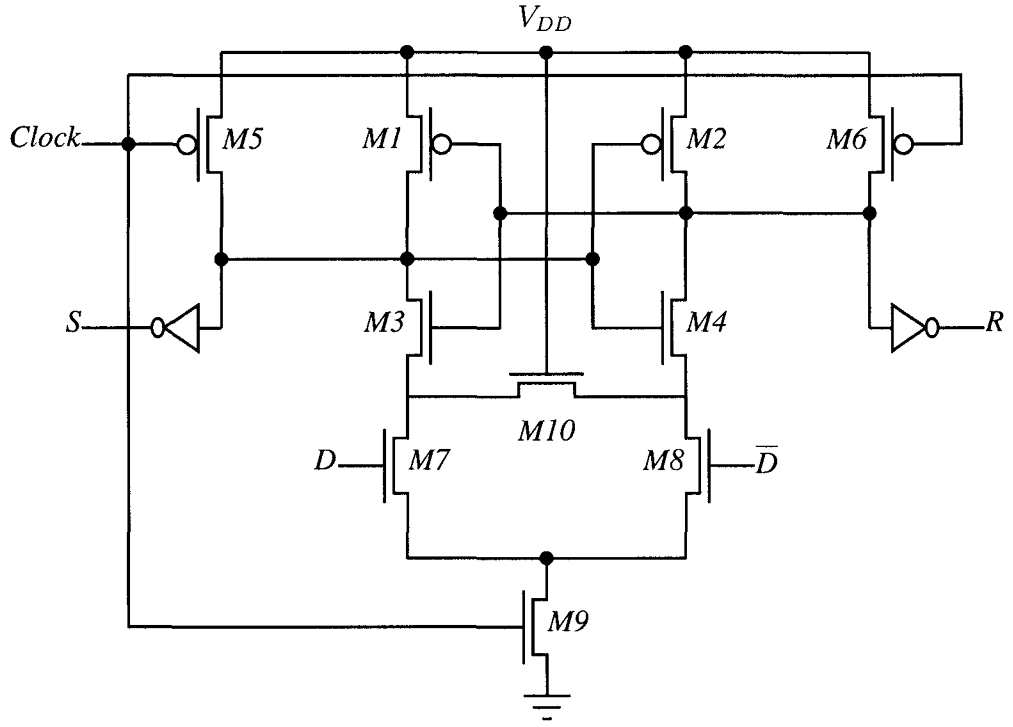

Figure 4.13: Sense-Amplifying Flip-Flop

Again, using the hysteresis buffer at the SAFF output improves the performance. When simulated with an input frequency chosen to produce a 1 MHz alias frequency, the sampling latch operates as expected on input frequencies as high as 230 GHz. Adding transient noise reduces the maximum input frequency to 76 GHz.

The LSF simulation test bench for the SAFF is the same as the DPTPL. Like the DPTPL, the SAFF circuit also has a minimum detectable pulse width is 9 ps, but with a much narrower window where this pulse is detected. A 9 ps pulse is only detected with the input pulse offset from the rising edge of the sampling clock by 12 ps. Repeating the parametric sweep for pulse widths up to 50 ps we find that the minimum and maximum offsets at which the SAFF still captures the pulse are -7 ps and 43 ps, respectively. The LSF plot is shown in Figure 4.14. The peak sensitivity is not centred in the sampling window like the DPTPL, since peak sensitivity lies at an offset of 12 ps but the centre of the sampling window is at approximately 18 ps. The LSF for the SAFF is also less symmetric than the DPTPL, and is more sensitivity after the peak than before.

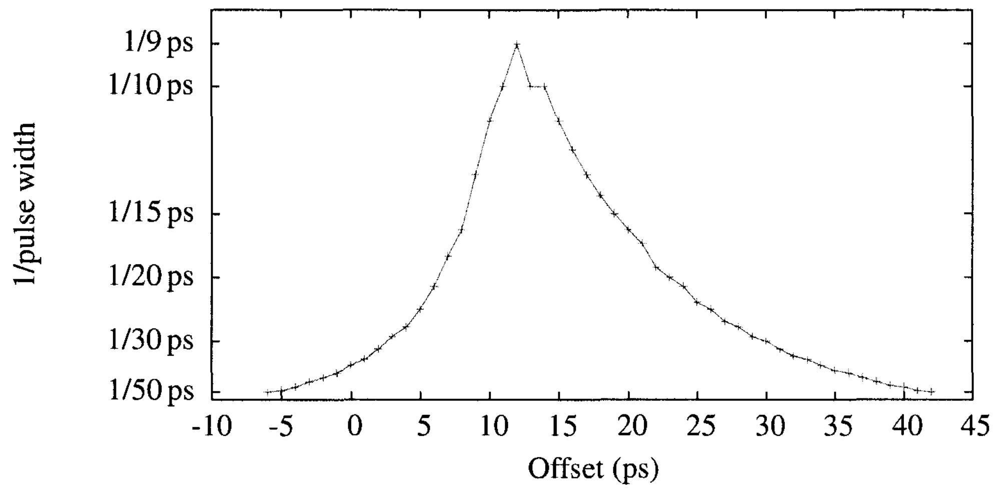

Figure 4.14: Latch sensitivity plot for SAFF

The sampling window sensitivity for the SAFF is shown in Figure 4.15. Like the DPTPL, the narrowest square pulse detectable at peak sensitivity is 8 ps between offsets of 10 ps and 11 ps. Unlike the DPTPL, however, the SAFF has a very sharp cut-off prior to peak sensitivity and instead remains sensitive much longer after the peak sensitivity. The sampling window for the SAFF is a few picoseconds narrower than the DPTPL, which in theory results in better high frequency performance.

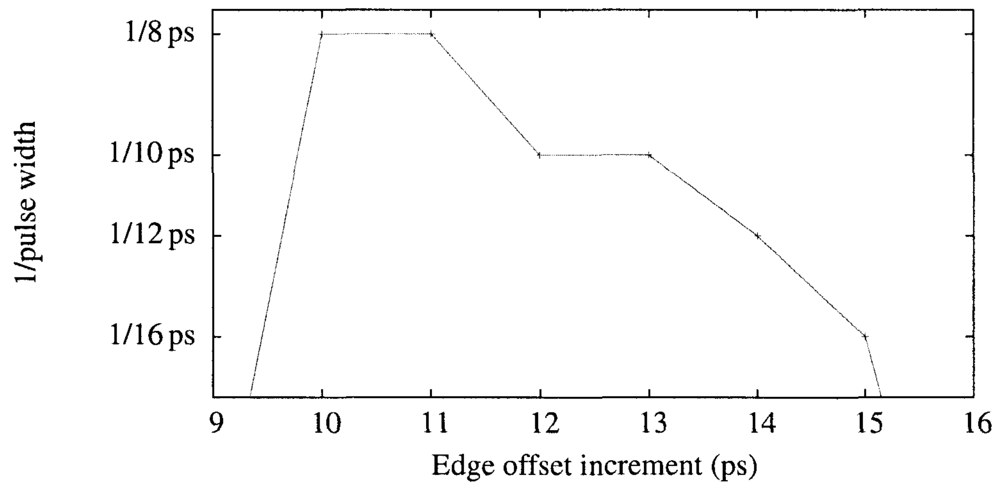

Figure 4.15: Latch sampling window plot for SAFF

### 4.2.5 Latch Simulation Results Summary

The maximum input frequencies under various operating conditions are tabulated in table 4.1.

Table 4.1: Maximum input frequencies for latch designs

| Alias Frequency | D flip-flop | DPTPL | SAFF |
|---|---|---|---|
| 100 MHz | 14.100 GHz | 179.100 GHz | 215.100 GHz |
| 1 MHz | 13.001 GHz | 26.001 GHz | 50.001 GHz |
| 1 MHz with noise | | 3.001 GHz | 3.001 GHz |
| 1 MHz with hysteresis | | 180.001 GHz | 230.001 GHz |
| 1 MHz with hysteresis and noise | | 58.001 GHz | 76.001 GHz |

Comparing the LSF for the different latches, it is clear that the ability to detect a narrower pulse translates to a higher frequency capability. Although the DPTPL and SAFF have the same peak sensitivity for both square and sinusoidal pulses, the maximum frequency simulations show the SAFF to have a higher frequency capability. This is perhaps due to the narrower range of offsets for the SAFF at which the latch reaches peak sensitivity or the narrower sampling window during which the SAFF can capture data. The latch sensitivity analysis indicates that the smallest pulse width that can be captured by the differential designs is on the order of 8 ps. These latches might then seem appropriate for sampling clock signals at up to 2.5 GHz, which corresponds to a signal with a 16 ps period and a 50 % duty cycle. In maximum frequency simulations, however, these latches can actually sample periodic signals at up to 230 GHz.

## 4.3 Input Threshold Correction

As seen in simulations with the standard cell D flip-flop design and discussed in chapter 3, an offset in the input threshold can cause the latch output to be biased to a certain logic level and cause the latch to fail at higher frequencies. A similar offset can also arise in differential circuits, which are sensitive to mismatch between what should be symmetric branches. In this section, an offset correction circuit for differential circuits is presented and simulated. The duty cycle detection circuits apply equally well to single-ended circuits, but would require a different offset cancellation mechanism inside the latch.

### 4.3.1 Circuit Design

The duty cycle measurement system proposed uses a single up/down counter that stores a running sum in two's complement. The up/down mode input of the counter is evaluated on the rising edge of the sample clock and determines whether the counter increments or decrements for the given sample clock period. The up/down mode for the counter is driven directly by the latch output, so while the latch output is low the count value will decrement by one each sample clock period, and conversely, increment the count value by one when the latch output is high. When operated in this manner, the counter generates a running sum of output values for the latch. For an ideal 50 % duty cycle, the long term average value stored in the counter will thus tend to zero. Deviations from the ideal duty cycle will lead to an increasing or decreasing value in the counter, with the rate of change of the counter value dependent on the degree of deviation from a 50 % duty cycle. The value stored in the up/down counter is then used to program DAC circuits that control transistors within the latch that counteract asymmetries and correct the duty cycle of the latch output. A block diagram of such a correction system is shown in Figure 4.16.

A modification to the DPTPL circuit to support duty cycle correction is shown in Figure 4.17. Two transistors, M9 and M10, are independently controlled by tuning the voltage on the *Left* and *Right* pins. These transistors can then be used counteract mismatch by biasing one side of the differential circuit.

The bit-width of the counter and the DAC precision will depend on a number of factors. The DAC should have a sufficiently high resolution to control the threshold correction transistors finely enough to correct the duty cycle. For different latch topologies, the required precision will differ, dependent on the sensitivity of the latch to threshold correction transistor input levels. The total bit-width of the counter is determined by the required precision of the DAC and the ratio between

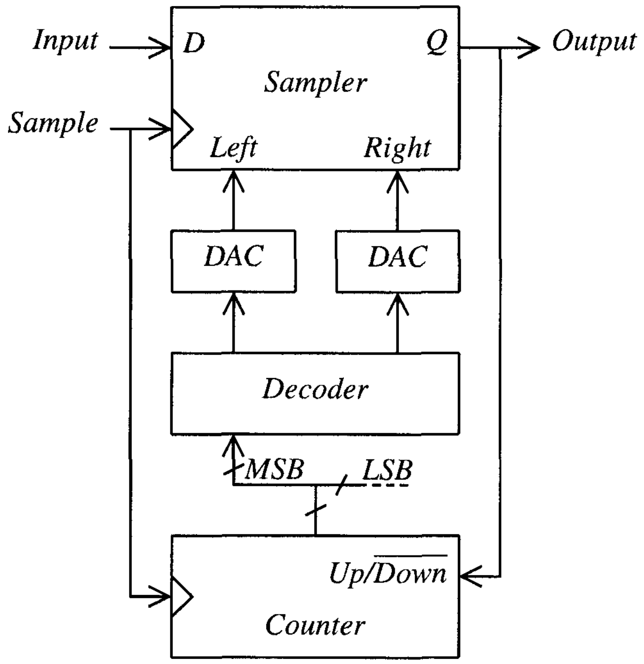

Figure 4.16: Block diagram of basic duty cycle correction system

the period of the latch output signal and the period of the sampling clock. Once the desired DAC precision has been determined, the next factor that determines the bit-width of the counter is the fact that the low-order bits in the counter change continuously as the counter up/down mode input alternates from up to down during a single period of the latch output. The value of these low-order bits during a single clock period should have no effect on the DAC output voltage, since they do not constitute a long term deviation from a 50 % duty cycle. For instance, if the sampling clock has a period of 1 ns and the alias clock period is 32 ns with a 50 % duty cycle, the up/down counter may increment by as much as 16 before decrementing by 16. Clearly, in this example, at a minimum, the 4 least-significant bits in the counter should be ignored by the DAC. In general, the number of bits to be ignored should practically be at least:

$$log_2\left(\frac{f_{sample}}{f_alias}\right) \tag{4.2}$$

As the number of low order bits ignored by the DAC is increased the number of sample clock periods that are required before the DAC output value is affected goes

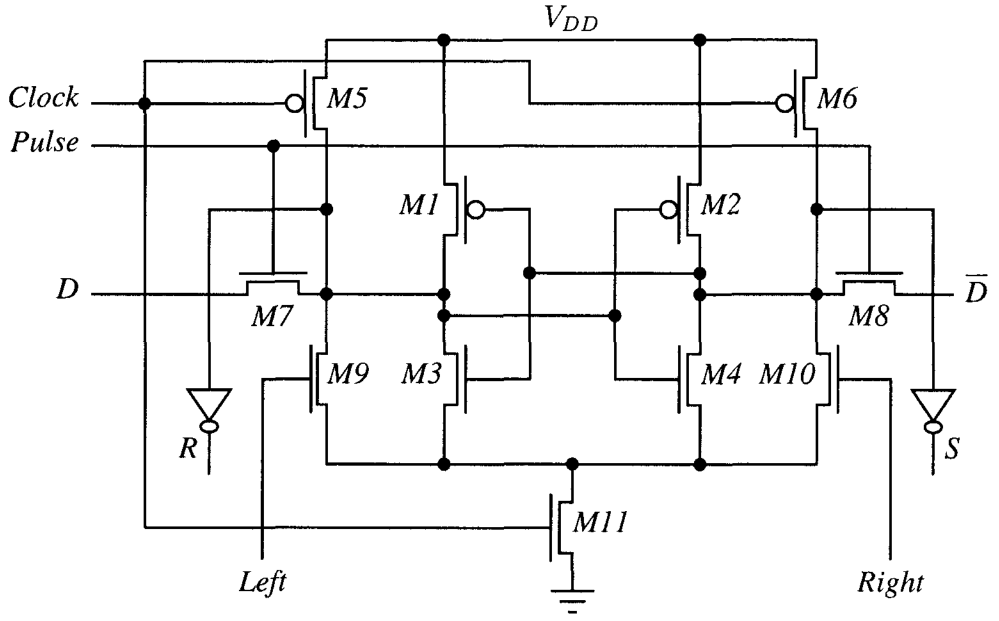

Figure 4.17: Differential pass-transistor pulsed latch circuit with input threshold correction transistors

up, causing the response time of the duty cycle correction system to increase.

Another consideration is the total required bit-width of the counter to avoid overflow of the counter. As the counter value changes and begins to alter the duty cycle of the latch, the latch output duty cycle will approach 50 %. As a 50 % duty cycle is approached, the rate at which the counter will change decreases. When a 50 % duty cycle has been reached, the control bits for the DAC should stop changing altogether. Overflow may still occur if the counter is saturated and the DAC control values have saturated but the counter value is still changing. In this case the duty cycle is not correctable by the system, and an eventual overflow or saturation of the counter is unavoidable, independent of the bit-width of the counter.

A possible configuration of the counter and decoder is shown in Figure 4.18. The counter value is stored in two's complement, and so bit 15, the most significant bit (MSB), is used by the decoder to determine whether the duty cycle is below 50 % (MSB is low) or above 50 % (MSB is high). When the MSB is high the next 8 bits (bits 14 through 7) are complemented and activate the left DAC control outputs $D_L7$ through $D_L0$ while the right DAC control outputs will remain low. When the

MSB is low, the right DAC is activated with the count value in bits 14 through 7 of the counter, while the left DAC control outputs are low.

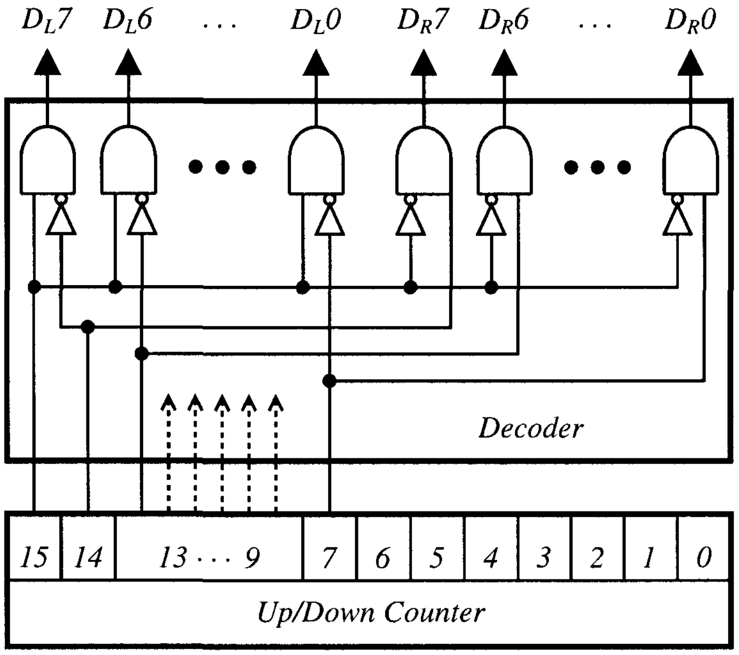

Figure 4.18: Block diagram of decoder circuit with up/down counter

#### 4.3.1.1 Duty-Cycle Tolerance Circuit

The system depicted in Figure 4.16 creates a feedback system that continually adjusts the voltage on the balance transistors in the differential latch circuit so that the duty cycle is exactly 50 %. However, as with any feedback system, this system may be unstable or oscillate. Because the adjustment steps are discrete, oscillations in the duty cycle will occur. As mentioned in chapter 3, the duty cycle correction circuit should only adjust the latch if the duty cycle deviation exceeds a specified value. To prevent duty cycle correction near 50 %, a certain number of low order bits in the counter can be reset periodically by a second counter. Such a duty cycle detection mechanism is shown in Figure 4.19.

Demonstrating by example, assume for instance, that the lower five bits of an up/down counter are reset to a known value every 300 cycles of the sampling clock. The known value loaded into the counter depends on the MSB of the up/down

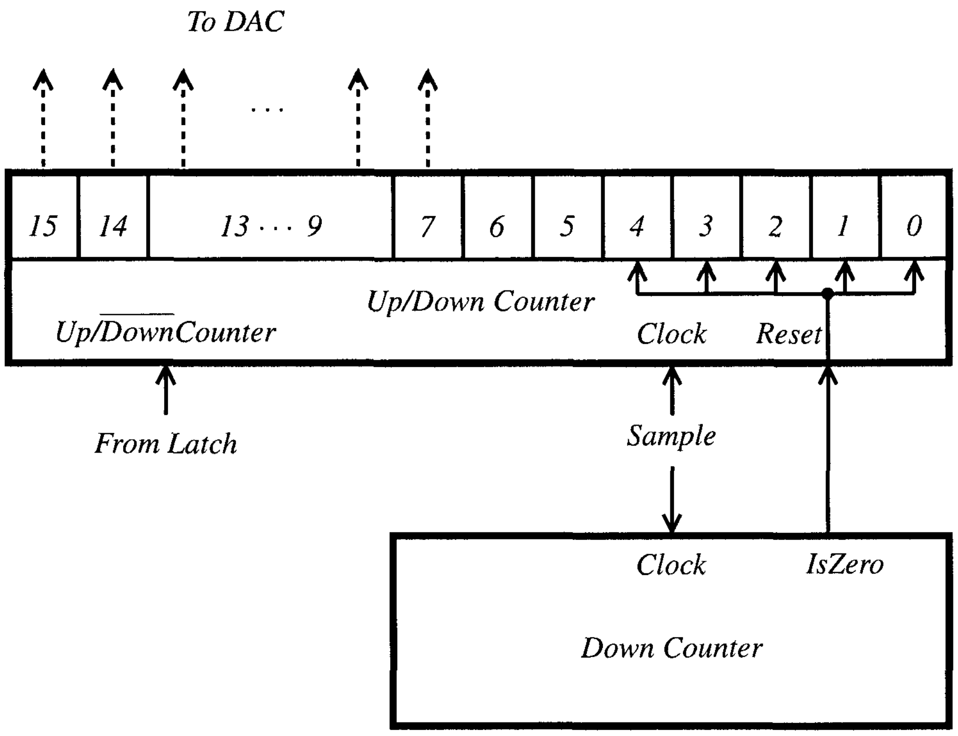

Figure 4.19: Block diagram of duty cycle detector with periodic reset

counter. If the MSB is high (1), and the counter value is thus negative, the lower five bits are reset to a two's complement value representation of -1 (11110 for a reset of the lower five bits). If the MSB is low (0), the lower five bits are set to 1 (00001 for a reset of the lower five bits). Using 300 cycles for a five bit reset implies that the up/down count increment or decrement during the 300 cycle period must exceed 31 to affect the counter value. The difference between the number of samples that are high ($highCount$) and low ($lowCount$) during this cycle must thus exceed 31. Assuming our duty cycle exceeds 50 %, $highCount$ will exceed $lowCount$, giving:

$$highCount - lowCount >= 31 \tag{4.3}$$

Their sum during a 300 cycle period is:

$$highCount + lowCount = 300 \tag{4.4}$$

Thus:

$$highCount - (300 - highCount) >= 31 \tag{4.5}$$

So

$$highCount >= 331/2 \tag{4.6}$$

Thus $highCount$ must be at least 166, leaving $lowCount$ at 134. With a 300 cycle reset period for the five lowest bits in the up/down counter the duty cycle that is greater than:

$$DutyCycle = 166/300 \cdot 100\,\% = 55.3\,\% \tag{4.7}$$

is not detected and the input-threshold adjustment controls are not altered by the system. Similarly, if the duty cycle is less than 44.7 %, the input-threshold adjustment value is not altered. More generally, for a reset period $resetPeriod$ and reset bit-width of $numResetBits$, the duty cycle range in which the threshold circuitry is not activated is given by:

$$DutyCycle = 50\,\% \pm \frac{2^{(numResetBits-1)}}{resetPeriod} \cdot 100\,\% \tag{4.8}$$

Some care must be taken to ensure that the low-order bit reset period is not a multiple of the predicted period of the output latch signal. If these periods are harmonically related, unfortunate low-order bit reset timing can cause the counter to count faster than intended. This risk can be minimized by ensuring that the number of low-order bits reset is sufficiently large that changes in the count value require multiple periods of the latch circuit.

### 4.3.2 Circuit Simulation

To demonstrate the feasibility of using the duty cycle detection and correction circuitry, the DPTPL from Figure 4.17 is used as a design with tunable offset correction. The addition of transistors M9 and M10 does reduce the maximum operating frequency for a 100 MHz alias frequency somewhat, as simulations shows proper functionality with input signals up to about 164.1 GHz, rather than the 179.1 GHz for the circuit in Figure 4.7.

Although various transistor mismatch mechanisms can arise in modern CMOS processes, for the purpose of verifying the duty cycle correction circuit, only the transistor dimensions are modified. Specifically, for the circuit in Figure 4.17, the circuit is made asymmetric by increasing the drawn length of transistor M1 from 100 nm to 130 nm. The asymmetric DPTPL fails to operate correctly when simulated with a 164.1 GHz signal, and produces an unchanging output signal, and this remains true for input signals as low as 64.1 GHz. With the input frequency set to 63.1 GHz, the mismatched latch produces an output signal with the proper alias period, albeit with a duty cycle of 80 %. Reducing the input frequency further down to 44.1 GHz causes the output alias signal to return to a 50 % duty cycle.

Demonstrating the restoration of higher frequency operation, the voltage on the $Right$ pin is increased to 0.45 V, and the mismatched DPTPL function correctly at frequencies up to 164.1 GHz, a reduction from the perfectly matching circuit, but a big improvement over the mismatched DPTPL without tuning voltages applied. The tuning voltage on the $Right$ pin under which the mismatched DPTPL functions correctly at a reduced 150.1 GHz ranges from approximately 0.440 V to 0.455 V, implying that the DAC output step size should be less than 15 mV to correct the latch up to frequencies of 150 GHz and requiring a seven bit DAC for a 1.2 V supply voltage.

To verify that the duty cycle detection circuit functions as expected a simulation test bench that generates a configurable duty cycle input and periodic low-order bit reset signal is created. Using the circuit with a configurable duty cycle tolerance, as shown in Figure 4.19, this system is simulated with a 1 ns sample clock and a 300 ns low-order bit reset signal. The DAC outputs are connected to a small capacitor, and the simulation is run with a duty cycle of 20 %, 42 %, and 47 %. With the number of reset bits at 5, according to equation 4.8, the DACs should not produce a voltage for any duty cycle that is greater than 044.7]% or less than 55.3 %. The output voltages for the left DAC output are plotted in Figure 4.20. The right DAC output remains at zero for the duration of the simulation. As expected, the 47 % duty cycle signal does not affect the output, while those signals with duty cycles less than the threshold cause the DAC output voltage to change.

The previous results demonstrate that a duty cycle correction circuit can detect DC bias, mismatch or offsets that result in a duty cycle that deviates from 50 %,

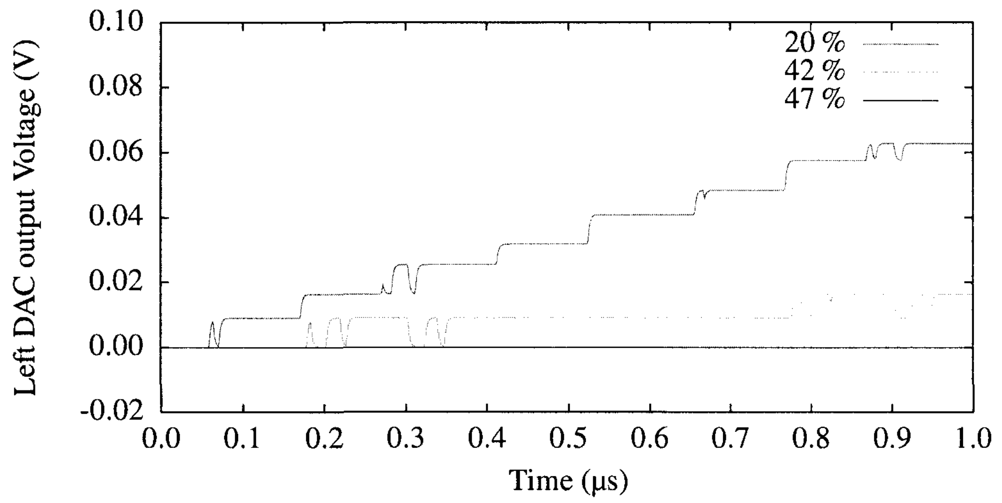

Figure 4.20: Left DAC output voltage versus time for various duty cycle input signals

and, in response, produce control voltages that correct these problems. Similarly, the simulation with the mismatched DPTPL demonstrates that a biased output can be tuned out by introducing tuning transistors that counteract the effects from problems such as mismatch. Ideally, to verify the complete system, the mismatched DPTPL circuit is combined with the duty cycle correction circuit and simulated to verify that the tuning voltages are indeed adjusted until the duty cycle is within a predefined margin of 50 % and the alias period is as expected. Unfortunately, such a simulation, which incorporates hundreds of transistors and combines an oscillator circuit operating in excess of 150 GHz with a duty cycle correction mechanism that takes on the order of microseconds to change the tuning voltages, requires weeks of simulation of time. Instead, to confirm that it is indeed possible to correct any offset in the input threshold, the mismatched DPTPL latch is simulated to generate a plot that shows the duty cycle versus the correction voltage, demonstrating that as the correction voltage monotonically increases, the duty cycle approaches and exceeds 50 %. This plot is shown in Figure 4.21. These results, combined with the information in Figure 4.20 shows that it is indeed possible to perform offset correction on a DPTPL with severe mismatch.

Although some of the above simulations are specific to a DPTPL circuit with

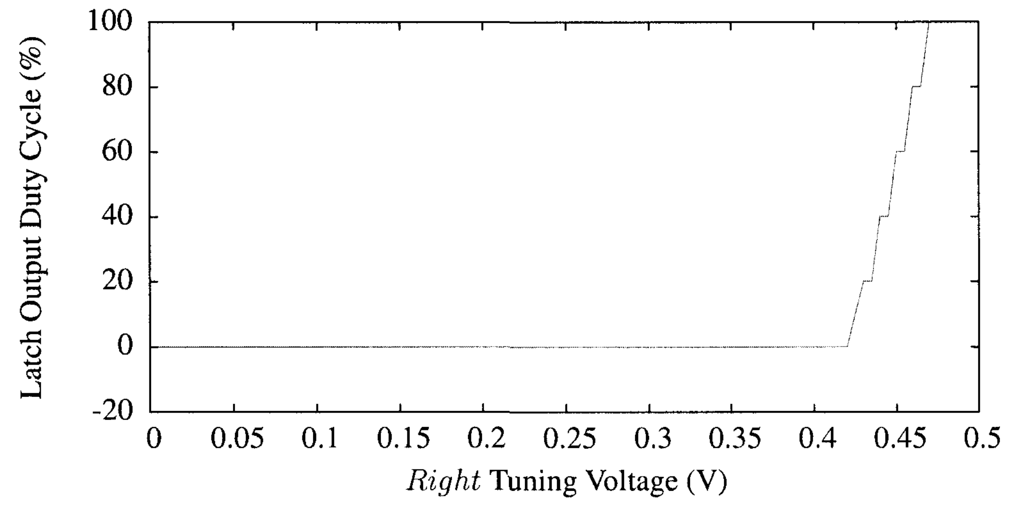

Figure 4.21: Mismatched DPTPL duty cycle versus applied offset correction voltage with a 150.1 GHz input signal

differential mismatch, similar offset correction mechanisms can be conceived for a variety of latches. The duty cycle detection and output circuitry that drives the offset correction mechanism can still be used in the same manner.

## 4.4 Summary

This chapter presented three sampling latch circuits and evaluated their suitability as aliasing unit for the ALL architecture. The loosened constraints placed on the behaviour of the latch allows for design optimizations that would not normally apply to standard latch designs in other digital systems, allowing for the possibility of building sampling circuits that could operate on input signals that exceed the $f_{max}$ of a process. The suitability of each latch architecture is compared by simulation of each design in a 90-nm CMOS process under the same conditions that the latch is expected to operate in. To establish a baseline performance figure a single-ended D flip-flop from a standard cell library is simulated first, and showing that this design fails to operate correctly at frequency above 14 GHz.

Simulating two differential circuits, the DPTPL and SAFF, reveals much higher input frequencies can be achieved. The maximum input frequency, however, depends on the alias frequency. We find that at lower alias frequencies the latch output is indeterministic when the input signal is near the switching threshold of the latch. At alias frequencies that are one-thousand times lower than the sampling frequency, the maximum input frequencies under which the DPTPL and SAFF operate correctly are 26 GHz and 50 GHz, respectively. In order to overcome the indeterminism, a hysteresis buffer is cascaded at the latch output. This improves the maximum input frequency to 180 GHz and 230 GHz for the DPTPL and SAFF, respectively. To further accurately replicate real-world conditions the simulations are repeated with thermal and flicker noise sources that are included through transient noise. At low alias frequencies this reduces the maximum input frequency at which the differential designs operate correctly to less than 5 GHz. With the hysteresis buffer cascaded, however, these figures improve to 58 GHz and 76 GHz for the DPTPL and SAFF, respectively.

Finally, this chapter concludes with a set of circuits that can tune out any DC bias, mismatch or other imbalances that alter the input threshold of the latch and cause the duty cycle of the sampler output to deviate from 50 %. A detection circuit with a programmable error margin for duty cycle measurement is presented and simulated. This circuit, combined with some output circuitry and latch with offset correction transistors, is then shown, through simulations, to be capable of detecting and correcting a deliberately and severely mismatched differential DPTPL circuit. Through this duty cycle correction and detection mechanism the maximum input frequency of the DPTPL is restored to levels close to those observed in the perfectly symmetric and matched DPTPL. Although not further explored, the same duty cycle detection and correction circuits can be applied with minor modifications to different latch designs.

# Chapter 5

# Analysis and System Simulation Results

## 5.1 Overview

To verify that an ALL system can function as a frequency synthesizer, the system must be able to achieve lock through the normal PLL pull-in process and maintain lock subsequently. For theoretical validation of the behaviour in the locked state a mathematical model that describes the locked behaviour of an ALL with ideal components has been derived. To further verify that the system can obtain lock and still functions with non-ideal components, a transistor level ALL circuit is designed and simulated in 90-nm CMOS technology.

## 5.2 Non-Linear Model

Because the aliased signal is discretized in time, and will typically vary between two different clock periods, the system never achieves exact phase lock at the PFD. In an ordinary PLL the VCO control voltage eventually settles on a fixed value, but an ALL that locks with binary sampling can behave like a fractional-N PLL [28] or bang-bang PLL (BBPLL) [35]. The ALL is similar to a BBPLL in that it never achieves true phase lock and cannot be accurately described by traditional Laplace domain analysis. Instead, like the BBPLL, the dynamics of the ALL are governed by limit cycles in phase-space diagrams that relate the loop filter voltage and phase error. Using a non-linear time-domain model, the period jitter due to a lack of exact phase locking can be observed and the ALL behaviour under varying sample clock and reference clock frequencies can be examined. Both the frequency ratio and relative offset between the sample and reference clocks influence the trajectory of the limit cycles observed.

### 5.2.1 Model Assumptions

To properly analyze the ALL, a time domain model for the system operating in locked state has been developed, similar to the analysis presented in [35], but adapted for an ALL frequency synthesizer. The time-domain description uses idealized models for all components. We assume that the input reference clock and sample clock are free of jitter. The VCO frequency does not drift and its instantaneous frequency is given by the equation:

$$F_{VCO} = (F_0 + K_v \cdot V_{LPF}) \tag{5.1}$$

where $F_{VCO}$ is the VCO period, dependent on the base frequency $F_0$, VCO gain $K_v$ and the low-pass filter (LPF) voltage $V_{LPF}$ that controls the VCO. For the PFD and charge pump an idealized model is used that will produce a positive or negative current for a duration that corresponds linearly to the phase difference between the two input signals of the PFD. The LPF is modelled as a series-connected RC filter, which essentially provides a proportional and integrating path. The sampler is assumed to be ideal; that is the sampler acts as a Dirac comb function applied to the VCO signal. To ensure that an ALL still functions with finite speed transistors and higher-order effects, a transistor-level simulation is described in section 5.4.

### 5.2.2 Model Description

To model the ALL, a set of finite-difference equations are found that describe the next state of the ALL as a function of the previous state in time. The state of the system is updated every reference clock cycle and can be described by three variables:

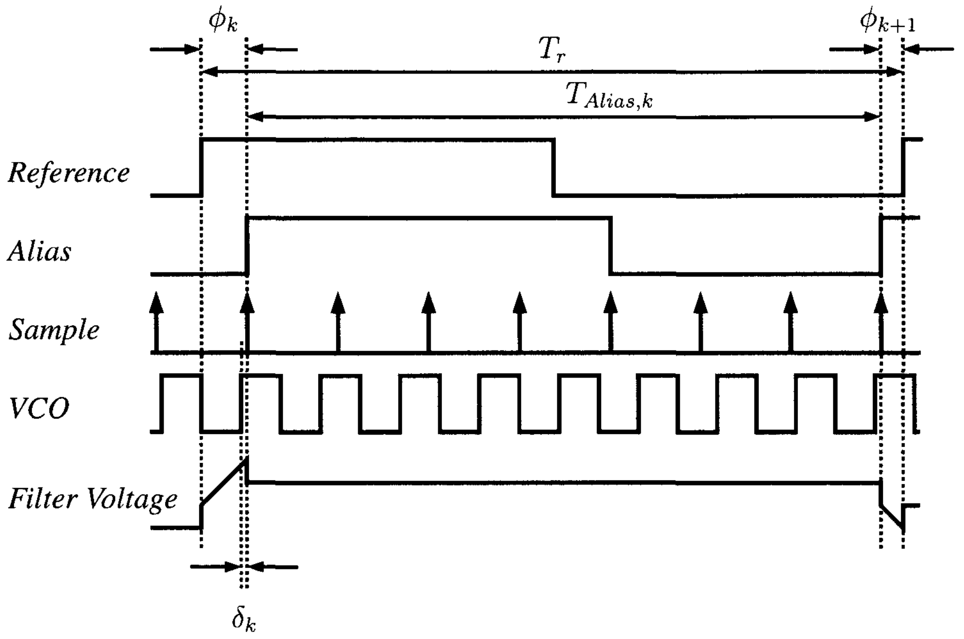

Figure 5.1: Alias-locked loop timing for non-linear model

- $\delta$: the offset between the rising edge of the VCO clock and the sampling signal
- $\phi$: the offset between the rising edge of the reference clock and the rising edge of the aliased signal, that is, the phase error at the PFD inputs
- $V_{LPF,ss}$: the steady-state voltage on the LPF

The relation between $\delta$ and $\phi$ is shown in Figure 5.1. The state variable subscript $k$ corresponds to the $k$-th rising edge on the reference clock.

The initial timing relationship between the reference and sample clocks is defined by the state variable $\phi$, which represents the PFD input phase error. Knowing the initial $\phi$, the reference clock period $T_r$ and the sample clock period $T_s$ further define how $\phi$ evolves over time. Since the quantization step size for the PFD phase error is always a multiple of the sampling clock period, the initial $\phi$ value determines the future PFD phase error offset values possible. For instance, if $T_s$ is 1 ns, $\phi_0$ is 0.5 ns and $T_r$ is an integer multiple of $T_s$, then all future $\phi$ values satisfy $0.5 \pm l$ ns, where $l$ is any integer value.

In earlier revisions of the idealized model, $\delta$ was used to determine the duration of the next alias clock period, $T_{alias,k}$. This method relied on determining the average VCO frequency during a given reference clock period and assuming that a voltage change in response to a phase error at the PFD was initiated by the rising edge of the reference clock, even if the rising edge of the alias clock arrives earlier. The average VCO frequency was then determined by averaging the injected charge and new steady-state loop filter voltage over the reference clock period. Other equations, however, assumed that the average filter voltage applied over the alias clock period. This model was abandoned due to unrealistic modelling and inconsistencies.

The updated model presented below corrects these inconsistencies and other problems. Instead of using $\delta$ and an average VCO frequency for a given reference clock period, the different regions (charge injection and steady state) of the loop filter state are stored in a table and saved for subsequent phase error calculations. The alias period is instead calculated by integrating the VCO frequency from every sample point after the rising edge that starts the alias period. The integration of the frequency yields a phase value, and the falling edge of the alias clock is detected by a transition, starting from below, across 0.5 for the fractional part of the phase value. When the fractional phase value for previous sample point is at or above 0.5 and the next sample point yields a fractional phase that is below 0.5 again the rising edge of the next alias clock has arrived and this sample point defines the period of the current alias signal. To cascade a standard frequency divider after the sampling latch, like in Figure 3.4, a number of alias clock periods are counted to produce the divided down signal. The process is more formally described in Algorithm 1.

The $Fraction()$ function returns the fractional part of its argument, while the $Integrate(Signal, StartTime, StopTime)$ function integrates the frequency of the signal in the first argument between the time between the second and third argument. In the context of Algorithm 1, a time value of 0 as the $StartTime$ implies that the $Integrate()$ function integrates the frequency from the start of the alias period. Since the rising edge of the VCO clock does not necessarily align with

**Algorithm 1** Calculate the period of the feedback signal

```
$AliasPeriod \leftarrow 0$
**for** $i = 1$ to $DivideFactor$ **do**
  **repeat**
    $AliasPeriod \leftarrow AliasPeriod + SamplePeriod$
    $Phase \leftarrow OldPhase + Integrate(VCOFrequency, 0, AliasPeriod)$
  **until** $Fraction(Phase) \geq 0.5$
  $AliasPeriod \leftarrow AliasPeriod - SamplePeriod$
  **repeat**
    $AliasPeriod \leftarrow AliasPeriod + SamplePeriod$
    $Phase \leftarrow OldPhase + Integrate(VCOFrequency, 0, AliasPeriod)$
  **until** $Fraction(Phase) < 0.5$
**end for**
**return** $AliasPeriod$
```

the rising edge of the alias clock (it is offset by some amount $\delta$), the phase of the VCO signal does not start at zero, and starts off at the fractional part of the phase remaining from the previous alias clock cycle.

Having calculated the period of the alias signal, $T_{alias}$, and knowing the previous PFD phase error $\Phi_k$ and reference clock period, $T_r$, the next PFD phase error $Phi_{k+1}$ can be found. To find the next phase error at the PFD, $\phi_{k+1}$, the difference between the reference clock period and the alias signal period is added to the previous PFD phase error:

$$\phi_{k+1} = \phi_k + T_{alias} - T_r \tag{5.2}$$

where the reference clock period, $T_r$, is subtracted from the alias period, $T_{alias}$.

Having found the PFD phase error, the voltage on the low-pass filter can be calculated. Using an RC filter results in a waveform like that shown in Figure 5.2, with the instantaneous VCO frequency given by the loop filter voltage, these voltages are used to calculate the next alias period. With an RC filter, the charge pump current into the resistor gives rise to an immediate step in output voltage, given by the charge pump current, $I_{cp}$, and the resistance value in the RC filter, $R$. The charge pump remains enabled for a duration of $\phi$, charging the capacitance $C$ at a rate of $I_{cp}/C$. When the charge pump is turned off, the voltage step across the resistor drop disappears, and the new steady-state voltage of the loop filter has

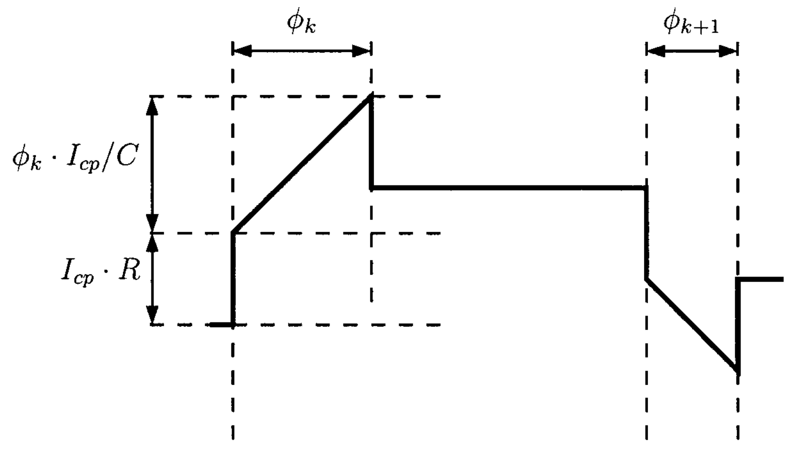

Figure 5.2: Low-pass Filter Waveform

changed by $\phi \cdot I_{cp}/C$.

## 5.3 Non-Linear Model Simulation

With the equations that relate the loop filter voltage and alias signal period defined, the loop is closed, and the ALL model can be simulated. The actual implementation of the model is done in C, and the complete source code is available in Appendix B. For all subsequent simulation results, the parameters used for the simulations are identical, except when otherwise noted. The intention of these simulations is to compare the impact of different parameters on the behaviour of the ALL, rather than demonstrating the process for a particular choice of settings. An interesting plot that can be generated is a phase-space diagram shows the evolution of the PFD phase error, $\phi$, and the steady-state loop filter voltage, $V_{LPF,ss}$ over time. For the results presented in subsequent simulations the baseline ALL is configured with a target VCO frequency of 10.1 GHz and a sample frequency of 1 GHz. To demonstrate the locking behaviour, this ALL configuration is simulated with an initial PFD phase error $\phi$ of -9.5 ns. The resultant phase-space plot with $\phi$ and $V_{LPF,ss}$ is shown in Figure 5.3. The discrete steps in the phase-error $\phi$ are clearly visible, and the plot demonstrates that the PFD phase error between the alias signal and reference signal is reduced as the ALL locks.

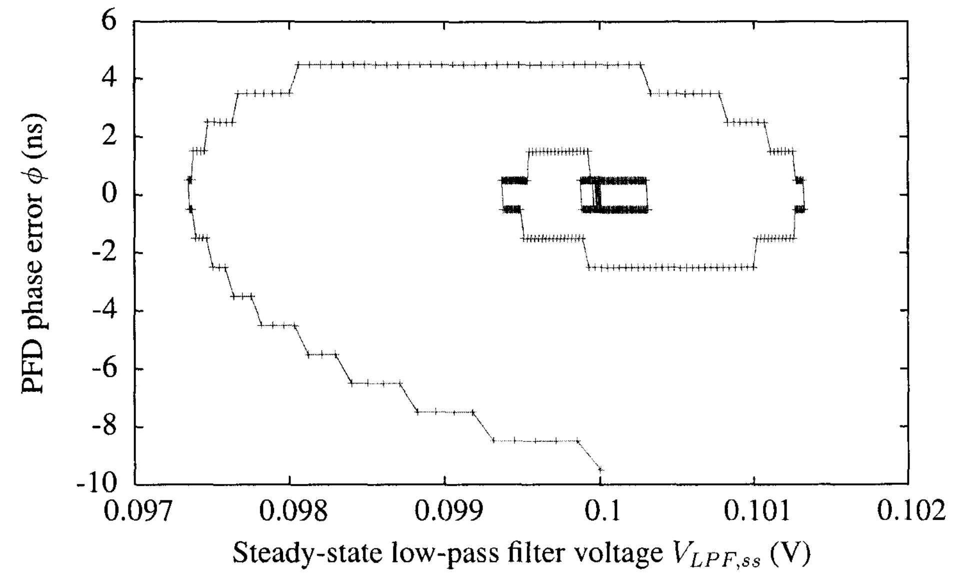

Figure 5.3: Phase-space diagram of PFD phase error $\phi$ and the steady-state loop filter voltage with a -9.5 ns initial phase-error

When setting the initial PFD phase error $\phi$ to -0.5 ns, the phase-space plot in Figure 5.4 shows that the PFD phase error remains bounded, and a four-point trajectory is traced out. Although not clearly visible in Figure 5.3, the same trajectory is evident when examining the centre of that plot. The bounded orbit indicates that the phase error at the PFD is stable. The initial PFD phase error choice of -0.5 ns, which exactly half of the sample period, results in a PFD phase error that jumps between -0.5 ns and 0.5 ns.

Changing the initial PFD phase error $\phi$ to -0.1 ns gives rise to a more complicated trajectory in the phase-space plot of Figure 5.5. This change in offset between the sample clock and reference clock changes the number of steps in the orbit. With a sampling period of 1 ns, the PFD phase error for a locked ALL is either -0.1 ns or 0.9 ns, but the steady-state voltage on the loop filter varies over a larger range than the simple orbit in Figure 5.4.

Although the PFD phase error $\phi$ is interesting, it does not show how well the VCO signal tracks the reference clock, that is, the phase error between the VCO signal and the reference clock is not shown. This VCO phase error, $\psi$, provides

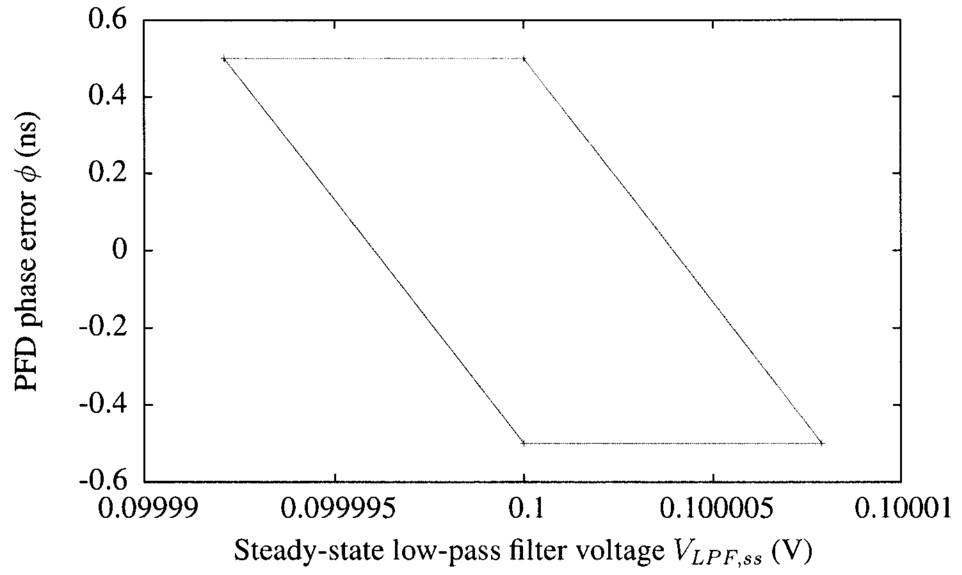

Figure 5.4: Phase-space diagram of PFD phase error $\phi$ and the steady-state loop filter voltage with a -0.5 ns initial phase-error

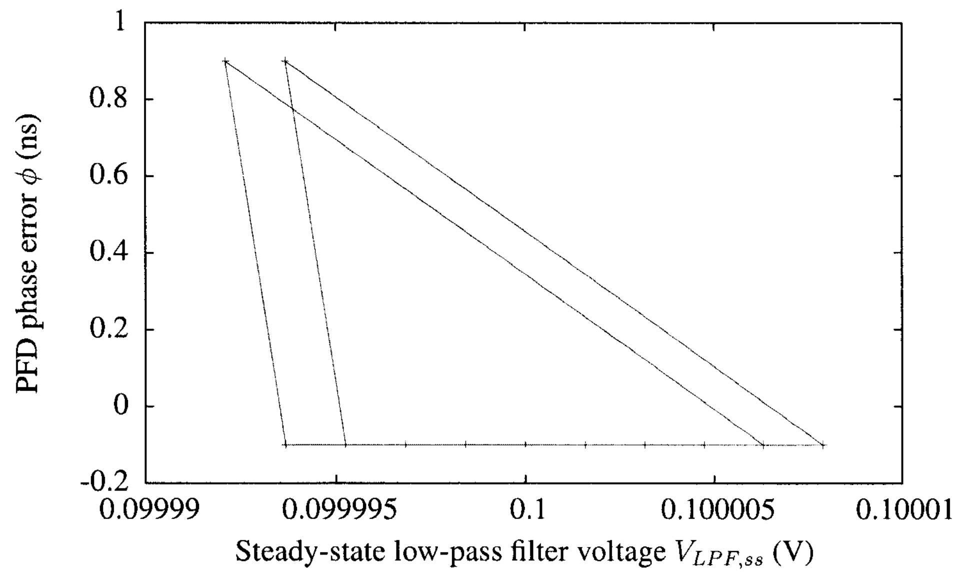

Figure 5.5: Phase-space diagram of PFD phase error $\phi$ and the steady-state loop filter voltage with a -0.1 ns initial phase-error

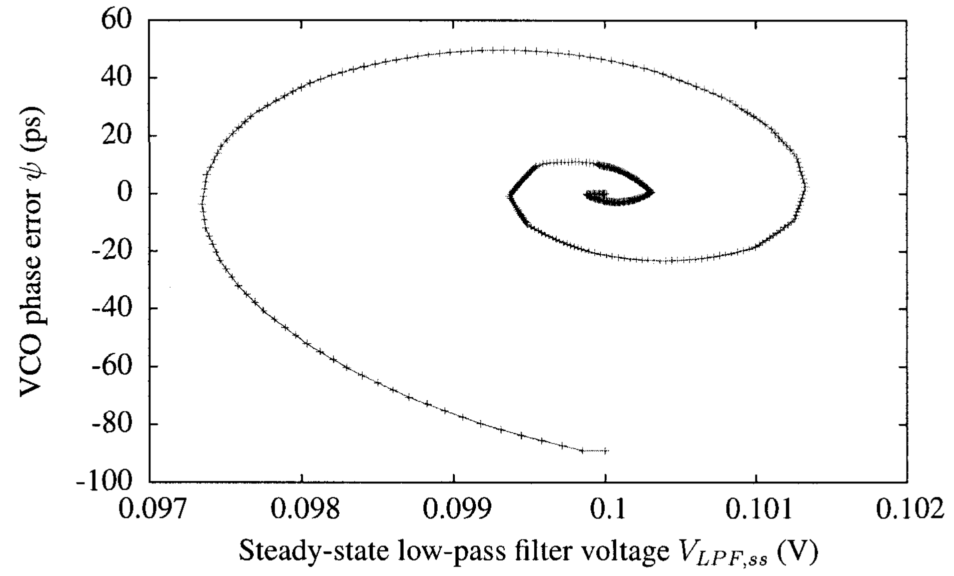

Figure 5.6: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -9.5 ns initial phase-error

information on the statistics of the output signal produced by the VCO and can be used to predict jitter and phase noise for different sample and reference frequencies. In order to calculate $\psi$, at the start of the simulation the initial PFD phase error $\phi$ is scaled by the effective frequency multiplication factor achieved by the system and this value is used as the initial VCO phase error $\psi$. Then, for each reference clock, the VCO phase error $\psi$ is updated by taking the number of ideal VCO clock cycles (which represents the phase) and using this phase value to calculate the time difference between the actual and ideal VCO signals. The actual VCO signal time is based on loop filter voltages previously stored in a table, and which were calculated when $\phi$ was determined. Plotting the data from the simulations used for Figures 5.3, 5.4, and 5.5, but now with $\psi$, the discrete steps disappear and the VCO phase error is reduced to a value close to zero. Figures 5.6, 5.7, and 5.8 show the corresponding phase-space plots with $\psi$ and the steady-state voltage on the loop filter.

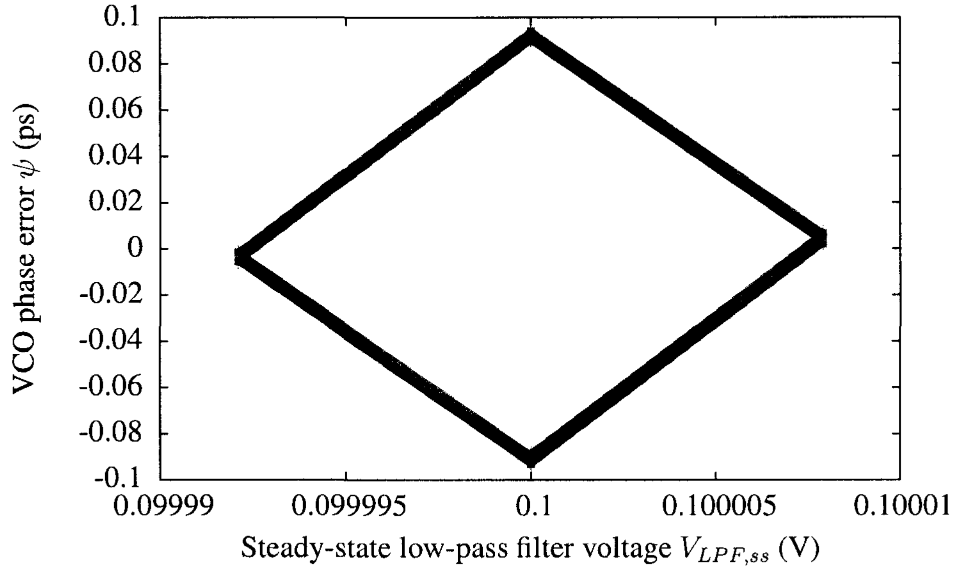

Figure 5.7: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.5 ns initial phase-error

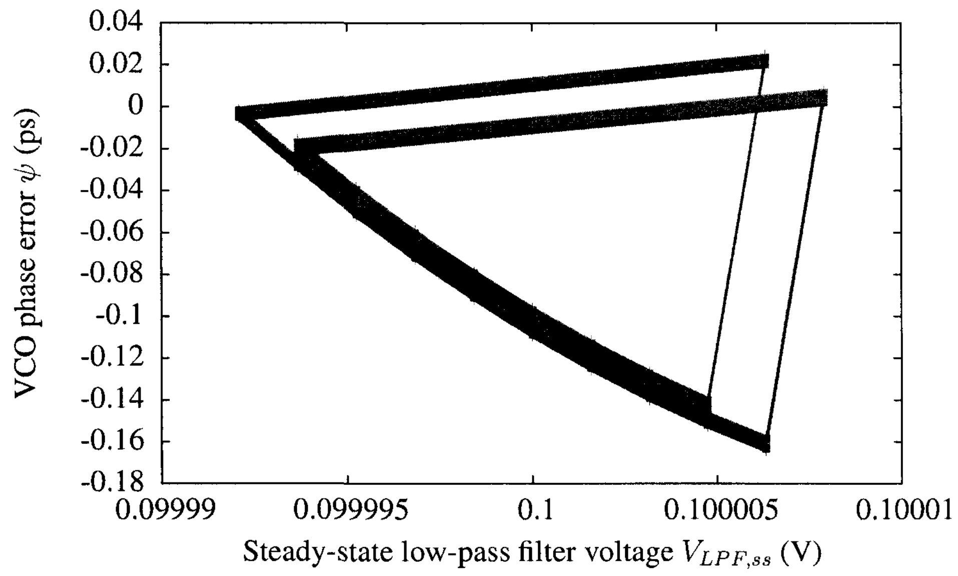

Figure 5.8: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.1 ns initial phase-error

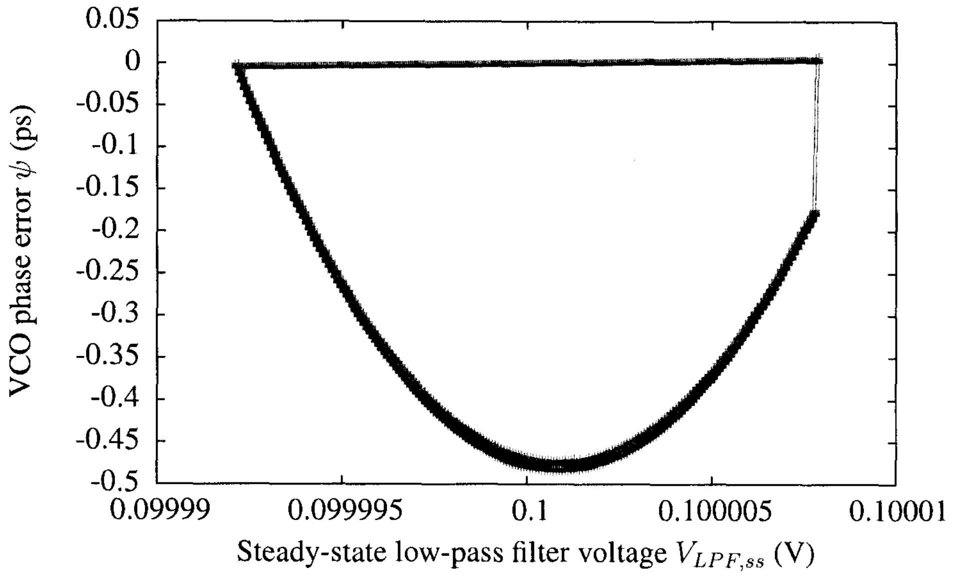

Figure 5.9: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.005 ns initial phase-error

## 5.3.1 Jitter and Phase Noise

Comparing the VCO phase error $\psi$ for Figures 5.7 and 5.8, it is evident that the peak-to-peak period jitter for both cases is roughly 0.2 ps. Repeating the same experiment for a range of initial PFD phase errors $\phi$ from -0.5 ns down to -0.05 ns shows the same peak-to-peak period jitter of 0.2 ps. However, as the initial $\phi$ value is set to values that approach zero from -0.05 ps, the period jitter increases until a new upper-bound is reached, with a peak-to-peak period jitter of approximately 10 ps. Figures 5.9 and 5.10 show the changes in the limit cycle orbits as the initial PFD phase error $\phi$ is decreased toward zero.

Comparing the frequency, as given by the steady-state voltage on the loop filter, for different offsets between the sample and reference clock yield interesting information as well. Generating a histogram plot for the steady-state filter voltages essentially gives an indication of the frequency spectrum of the VCO output signal. One should be reminded that such a histogram of steady-state filter voltages does not include the spurious tones that arise from temporary spikes in the loop filter voltage when the charge pump is enabled. Such spikes are especially prominent in

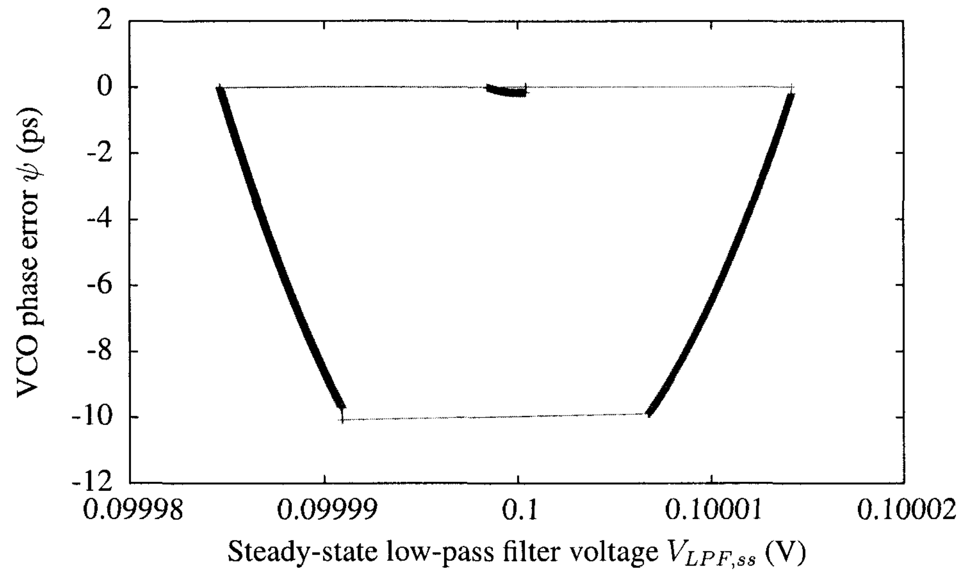

Figure 5.10: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.0005 ns initial phase-error

a simple RC filter, and can be significantly reduced by using a higher-order loop filter. For this histogram, a comparison is made between an initial phase-error $\phi$ of -0.5 ns and -0.001 ns by running the simulation for 10,000 reference clock cycles and counting the steady-state loop filter voltages into one of 100 bins. To demonstrate the effect the loop filter has on the location of the spurious tones, the -0.5 ns simulation is also run with a loop filter with a narrower bandwidth. This loop filter has its RC time constant increased tenfold, scaling the offset frequency of the spurious tones tenfold and decreasing the PLL natural frequency by a factor of $\sqrt{10}$. The resultant histogram is plotted in Figure 5.11. The loop filter bandwidth can be further narrowed to move these spurious tones so close to the carrier frequency that other noise effects dominate over limit cycle effects.

Although previous plots showed that the period jitter was much more significant for a phase-error offset close to zero, from this plot it is evident that the offset between the reference clock and sample clock does not significantly affect the frequency deviation from the nominal output frequency. In this particular configuration, the steady-state filter voltage varies between roughly 0.09999 V and 0.10001 V,

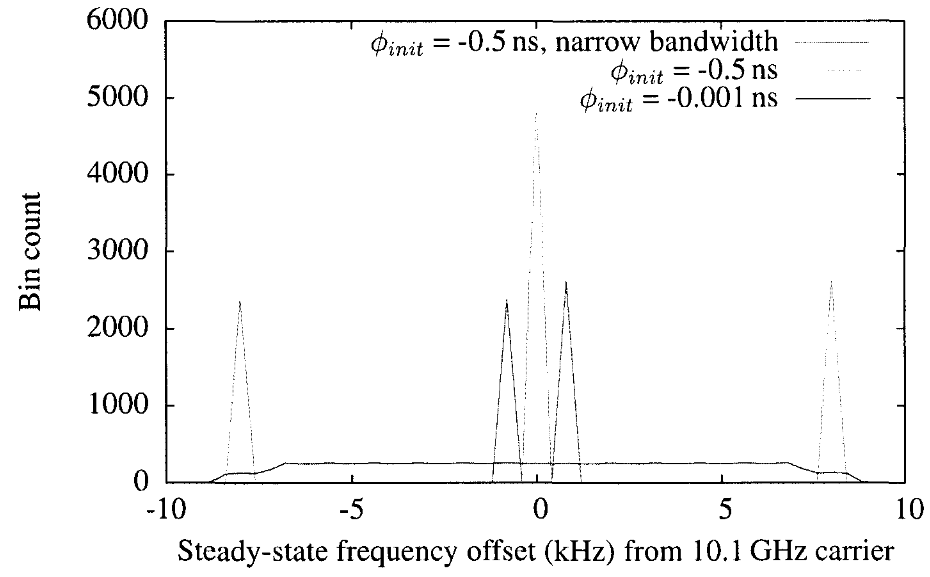

Figure 5.11: Histogram plot comparing the steady-state frequency distribution for initial phase-errors of -0.5 ns and -0.001 ns
and the effect of loop filter bandwidth changes.

corresponding to a frequency deviation of $\pm 10$ kHz. However, while an initial phase offset of -0.001 ns produces a broadly distributed frequency spectrum, using -0.5 ns produces strong spurious tones at offset frequencies of 10 kHz. To eliminate strong-spurs close to the desired frequency in this configuration, an initial phase offset that is closer to zero may produce a more desirable output spectrum. In the more general case, a limit cycle that includes many steps in its orbit will distribute the VCO signal more evenly over the range of frequencies produced. The width of this distribution of frequencies or location of spurious tones can be brought closer to the desired output frequency by increasing the sampling frequency or changing the loop gain of ALL by adjusting the loop filter parameters.

## 5.3.2 Impact of Sample and Reference Signals Choices

A number of choices can be made in the configuration of the sample and reference clock frequencies and their ratio. This subsection explores some of the effects from different selections for the alias and reference clock signals.

#### 5.3.2.1 Increasing the Sampling Clock Frequency

Simulations indicate that increasing the sampling clock frequency benefits both the peak-to-peak period jitter and maximum frequency deviation. Increasing the sampling frequency by a certain factor reduces the peak-to-peak period jitter by the same factor, both for small limit cycles like those in Figure 5.7 and large limit cycles like those in Figure 5.10. In the frequency spectrum the maximum frequency deviation and frequency offset for spurious tones is scaled in the same manner. This is also intuitive, as an increase in the sampling frequency reduces the minimum time step in the phase discretization, and thus reduces the minimum step in correction voltages on the loop filter.

#### 5.3.2.2 Non-Integer Ratio of Sample and Reference Clock Frequencies

When the period of the reference is not a multiple of the sampling period, the alias clock will alternate between two clock periods such that the average alias period matches the reference clock. To compare the behaviour under this situation to the previously simulated system the target VCO frequency is changed slightly from 10.1 GHz and set to 10.0952 GHz, which corresponds to a reference clock period and average alias signal period of 10.5 ns. Figure 5.12 shows the phase-space diagram for the VCO phase-error $\psi$ vs the steady-state loop filter voltage. Although the orbit shape is different from the case where the reference clock is 10 ns, the peak-to-peak period jitter and maximum frequency deviation are identical, at approximately 0.2 ps and $\pm$10 kHz, respectively. A histogram plot for the distribution of frequencies shows the same spectral peaks as Figure 5.11.

Changing the initial PFD phase error $\phi$ from -0.5 ns to -0.001 ns for a reference signal at 10.5 ns shows that the maximum period jitter is approximately 0.3 ps, and thus has not degraded significantly. Setting the initial $\phi$ to -0.25 ns, however, results in a peak-to-peak period jitter of -5 ps.

Generalizing the results of initial PFD phase error offsets further, simulations indicate that for a given selection of reference and sample frequencies certain offsets between the sample and reference signals give rise to a peak-to-peak period

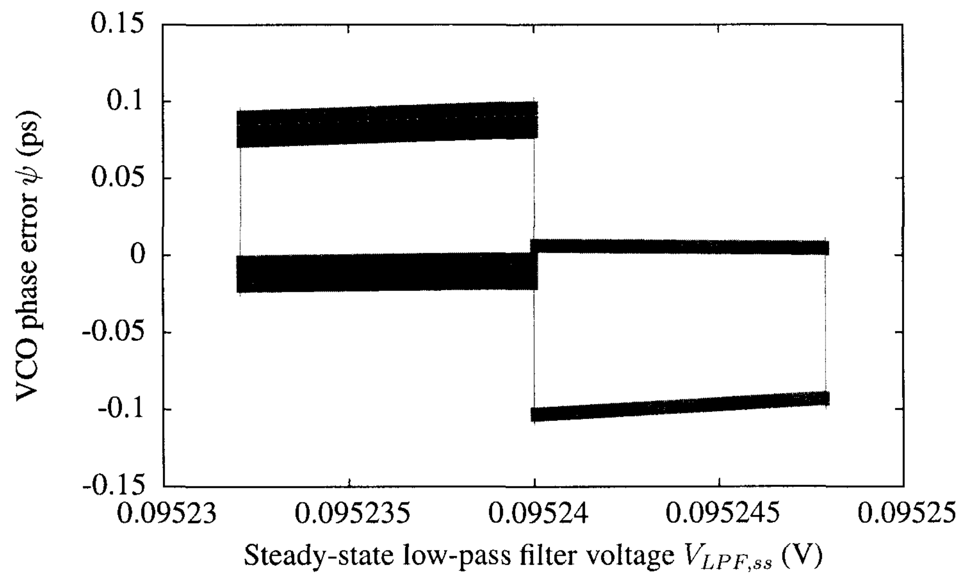

Figure 5.12: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.5 ns initial phase-error for a toggling alias period

jitter that is much larger than the typical case. From observation, these modes of operation with large peak-to-peak period jitter seem to arise around offset frequencies that are near zero. In other cases, a large peak-to-peak jitter is observed when the offset that is near the remainder or half of the remainder obtained when dividing the reference and alias clock periods.

Further simulations with a variety of sample and reference clock period combinations show that generally the larger least common multiple (LCM) of the two clock periods, the larger the peak-to-peak period jitter. From observations, a larger LCM for the two clock periods results in a limit cycle trajectory with more steps and more complicated geometry, typically leading to a larger tracking error between the VCO and reference clock.

#### 5.3.2.3 Subsampling and Frequency Division

In some cases, the alias frequency produced by the sampler can be impractically high. In these situations, a standard frequency divider can be cascaded after the frequency synthesizer. To examine the effect of a subsampler and frequency divider on the behaviour of an ALL system the non-linear model includes a parameter that models an integer-N frequency divider after the subsampling latch.

When maintaining the same reference frequency as earlier simulations (10 ns) and setting the frequency divide factor to two will cause the VCO to lock at 10.2 GHz. Examining the phase-space diagram, the orbit is identical to Figure 5.7, but offset by 0.1 V and centred at 0.2 V. The peak-to-peak period jitter is identical and the spurious tones are of the same magnitude and at the same frequency offset.

When changing the reference frequency to 20 ns and setting the divide factor to two the VCO locks at 10.1 GHz the orbit is again identical to the base case, with the same figures for peak-to-peak period jitter and offset frequencies for the spurious tones.

For cases where the desired VCO frequency requires a ratio of sample and reference clock frequencies that is not integer, a divider can be added to make this so. For instance, in a previous simulation run the desired VCO frequency was 10.0952 GHz. This same frequency is also produced when using a reference frequency of 21 ns and a divide-by-two frequency divider after the subsampling latch. The resultant phase-space orbit that shows $\psi$ and the steady-state loop filter voltage does change from that of Figure 5.12, and instead follows the same trajectory as the system in Figure 5.7, albeit at a different loop filter voltage. Again, the peak-to-peak period jitter and frequency offsets of the spurious tones are unchanged by the addition of a divider in the feedback path.

Generalizing, cascading a frequency divider in the feedback path does not benefit the peak-to-peak period jitter or phase-noise that is attributable to the discretization of the PFD phase error. This is expected, since the frequency division does not change the minimum step size of the phase-error. Consequently, the steps in loop filter correction voltages remain the same as the case where a frequency divider is not present.

## 5.4 90-nm CMOS Implementation

Although the theoretical model described in the previous section is useful in predicting how an idealized ALL system would behave, such a model would be of little use if practical implementations in silicon would not behave in the same manner. To verify that a physical ALL frequency synthesizer can work, a circuit implementation has been developed in a 90-nm process and is simulated with transistor-level SPICE models provided by the foundry. The 90-nm ALL design uses an LC tank oscillator, a DPTPL as a sampling latch, a latch-based PFD, a differential charge-pump, and an ideal RC loop filter. The simulation parameters for the results presented in this section are available in Appendix A and schematic diagrams for the circuits are provided in Appendix C.

### 5.4.1 90-nm Simulation Versus Non-Linear Model

To verify that the non-linear model simulation results can be used to predict the behaviour of physical implementations, the 90-nm design is simulated with initial conditions that closely match those used for the results plotted in Figure 5.3. With the reference frequency set to 100 MHz, the sampling frequency at 1 GHz and the loop filter voltage at a level that ensures that the VCO is oscillating at a frequency close to 10.1 GHz a short simulation reveals that the VCO gain is approximately 1.2 GHz/V. From this short simulation it is also determined that the steady-state loop filter voltage required to produces a 10.1 GHz signal is 1.0655 V. The reference clock signal is then adjusted to produce an initial PFD phase error $\phi$ of -9.5 ns. Measuring the charge-pump current produced at the desired loop filter voltage, however, shows that the magnitude of the current depends on whether the charge-pump is sinking or sourcing current. When sinking current, the measured current is 115 μA, but the sourcing current is -215 μA. Since the non-linear model assumes that these currents are equal some discrepancy can be expected. With the initial $\phi$ set to -9.5 ns, the ALL will initially be sinking current.

The 90-nm design is then simulated for 5 μs, which corresponds to 500 refer-

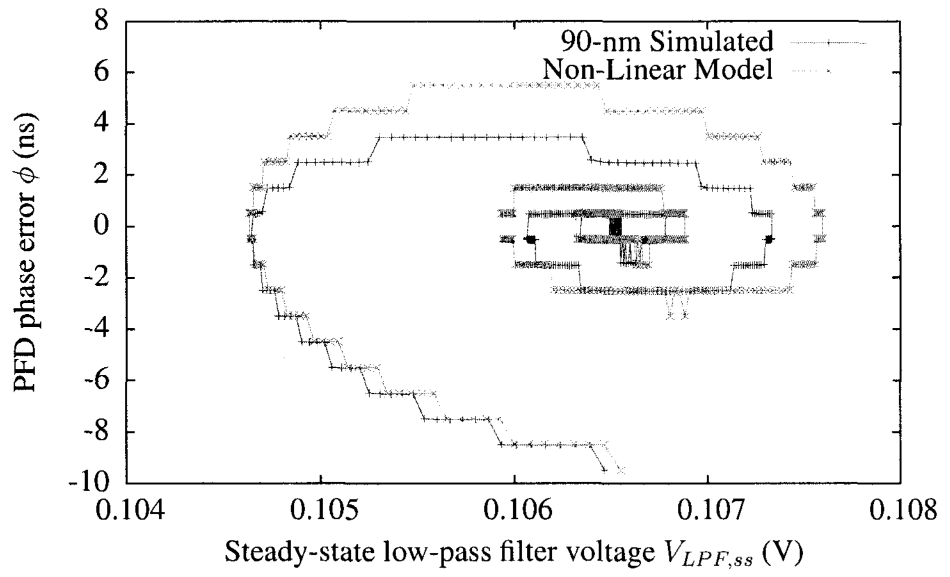

Figure 5.13: Phase-space diagram of PFD phase error $\phi$ and the steady-state loop filter voltage with a -9.5 ns initial phase-error, obtained from a 90-nm transistor-level simulation

ence clock cycles. Extracting the PFD and VCO phase errors $\phi$ and $\psi$ along with the steady-state loop filter voltage allows us to compare the 90-nm data directly with the simulation results of the non-linear model. The non-linear model is simulated with the settings that closely match the 90-nm design. To match the initial operating charge-pump current in the 90-nm circuit simulation the current is also set to $\pm 115\,\mu$A for the non-linear model.

Phase-space diagrams that combine the simulation results of the non-linear model and 90-nm design are plotted in Figures 5.13 and 5.14. It is clear from these plots that the results from non-linear model are in good agreement with simulation data from the 90-nm design for the initial 41 reference clock cycles. After that point, however, the phase-error becomes positive, and the charge-pump current levels of the model and 90-nm design no longer match, with the model assuming a charge pump current of -115 µA while the 90-nm design produces a current of -215 µA. The higher charge-pump current effectively increases the loop-gain for the system in the 90-nm design and reduces the phase-error overshoot when $\phi$ is positive.

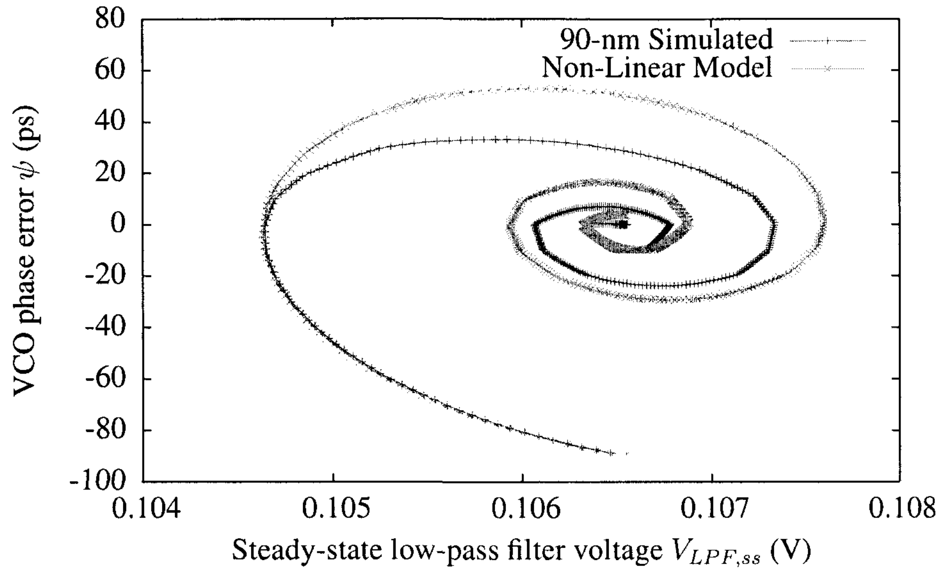

Figure 5.14: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -9.5 ns initial phase-error, obtained from a 90-nm transistor-level simulation

Examining the steady-state behaviour of the 90-nm simulation as it achieves phase-lock reveals that the trajectory followed in phase-space for $\psi$ and the steady-state loop filter voltage does not match the four-point orbit depicted in Figure 5.7. Instead, as shown in Figure 5.15, the trajectory traced out is more like that shown in Figure 5.8. This type of trajectory arises because the sourcing and sinking charge-pump currents are not equal. Thus, while the PFD phase error $\phi$ toggles between -0.5 ns and 0.5 ns, the charge injection into the loop filter is not equal in magnitude for the same phase-error magnitudes. One charge injection at a PFD phase error of 0.5 ns requires multiple charge injections at -0.5 ns to return to the same voltage, thus generating a triangular shape orbit. As expected, the trajectories exhibit more variation and do not follow a tightly bound limit cycle like the ideal models. Several reasons, including limited simulator step size resolution and the inclusion of many higher-order effects that are modelled in a transistor-level simulation, can account for the variation in trajectories. What can be observed and matches predictions by the non-linear model, however, is that the peak-to-peak period jitter is still roughly 0.2 ps. This figure is good in comparison to other frequency synthesis PLLs at this

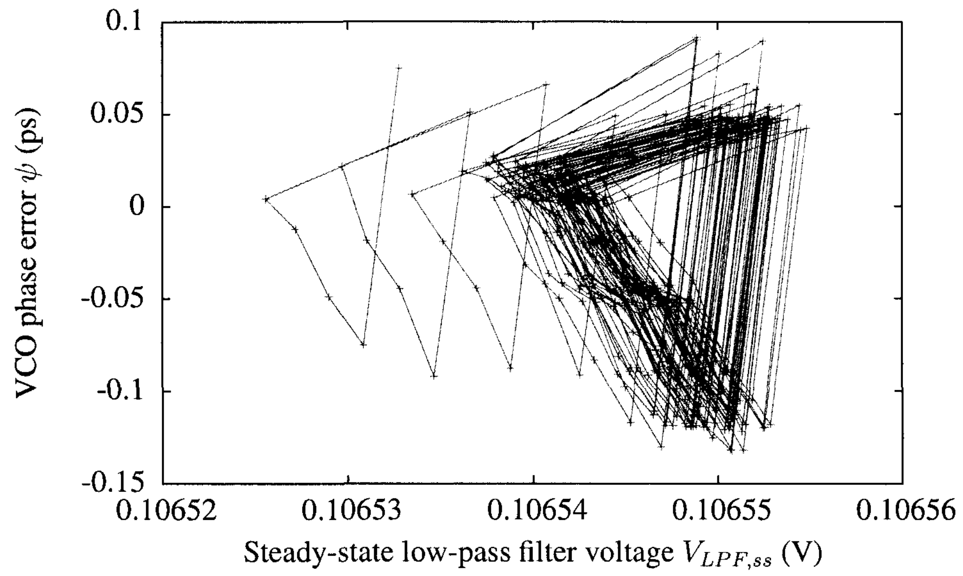

Figure 5.15: Phase-space diagram of VCO phase error $\psi$ and the steady-state loop filter voltage with a -0.5 ns initial phase-error, obtained from a 90-nm transistor-level simulation

frequency, but it must be remembered that the 90-nm CMOS simulations do not include any device noise.

### 5.4.2 90-nm Pull-In Simulation

Because the non-linear model as developed and presented in this thesis is only valid when the phase-error does not exceed one reference clock cycle, the 90-nm design is simulated outside this range. The pull-in process during which PLLs approach the lock-range is typically a slower process because phase-detectors cannot detect phase-errors that exceed one reference clock cycle ($2\pi$). During pull-in, the limited operating range for phase detectors causes cycle-slipping behaviour that temporarily reverses or stalls on the path toward phase-locking. Like typical PLLs, these cycle slips are visible on loop filter voltage against time as short spikes, temporary horizontal steps or decreasing slopes in the loop-filter voltage on an otherwise increasing slope (for cases where the loop filter voltage needs to increase to achieve lock).

Figure 5.16 demonstrates the 90-nm implementation of the ALL completing the

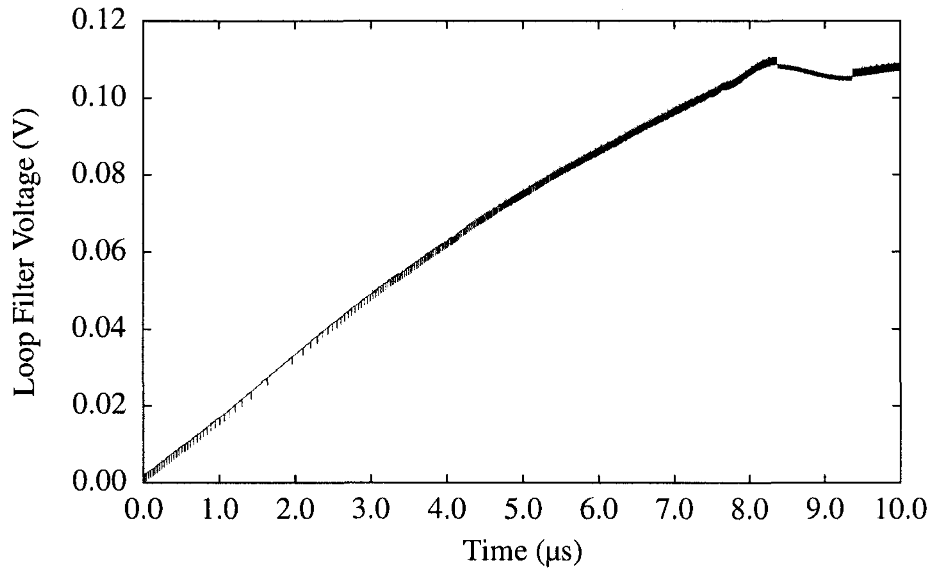

Figure 5.16: Loop filter voltage versus time, showing the ALL pull-in process and locking at 10.1 GHz

pull-in process and locking at 10.1 GHz. As expected, the pull-in and lock process is similar to traditional PLLs, with the effects of cycle slip seen during the pull-in step as the VCO frequency increases to 10.1 GHz.

## 5.5 Summary

This chapter described a non-linear model developed to simulate the ALL architecture. Subsequent simulations of this model demonstrate that an ideal implementation of the ALL architecture can obtain and maintain lock between the reference and VCO signals. A more detailed look at different configurations for the reference and sample clock frequencies, specifically the ratio of these signals and the time offset between them, reveals different limit cycles in phase-space plots of the VCO and PFD phase errors and loop filter voltages. The choice of sample and reference clock offsets generally has little impact on the VCO output signal, with peak-to-peak period jitter mostly unaffected, except for certain specific choices of time offsets between the sample and reference clocks or reference and sample clock periods that only share a large LCM. It is found that the most direct way of reducing period jitter and the offset frequency of spurious tones without altering the loop dynamics of the ALL is to increase the sampling frequency, effectively reducing the PFD phase error quantization step size that results from the sampling operation.

The validity of the results from the non-linear model are corroborated by a 90-nm implementation of an ALL circuit, showing good agreement between simulation results from the non-linear model and the transistor-level circuits of the 90-nm design. To verify the pull-in process of the ALL, the 90-nm implementation is simulated with a large initial frequency error. This simulation reveals that the ALL behaves similar to traditional PLLs during the pull-in process. The simulation results both at the non-linear model level and transistor level reveal that is indeed feasible to perform frequency reduction in the feedback path by subsampling the VCO signal and locking into the lower-frequency alias signal produced by this operation.

# Chapter 6

# Conclusion

In this thesis, an ALL architecture is presented. This novel approach to frequency synthesis replaces the frequency divider in the feedback path between the local oscillator and phase detector by a sampling latch operated well below the Nyquist frequency of the oscillator signal. The subsampling operation produces an alias frequency that is compared to a reference frequency and used to lock the ALL. Using a sampling latch rather than a traditional frequency divider significantly loosens the constraints on the feedback circuit. While a frequency divider's output operates at half the frequency of the oscillator, a sampling latch need only operate at the sampling frequency. Furthermore, a sampling latch in this application is tolerant of reset delays and clock-to-output propagation delays that span multiple cycle of the oscillator. Additionally, the sampling window for these latches can span across multiple cycles of the oscillator, as long as the latch consistently represents the phase of the sampled input signal. The loose constraints on the sampling latch lead to the possibility of building an ALL-based frequency synthesizer at frequencies near or exceeding the $f_{max}$ of a process. Additionally, because the sampling operates at a much lower frequency than a frequency divider, power consumption for sampling latches is lower than frequency dividers operating on equivalent input frequencies.

## 6.1 Accomplishments

This thesis has shown that it is indeed feasible to build a frequency synthesizer that uses a sampling latch for frequency reduction in the feedback path. Simulations with both a theoretical model of the ALL and a transistor-level implementation in 90-nm CMOS technology confirm that aliasing can be used instead of frequency division. Furthermore, the predictions from the idealized theoretical model match up very well with the more detailed and realistic simulation results of the 90-nm implementation of the ALL.

Several candidate sampling latch circuits have been evaluated, and show that appropriate differential latch designs may be able to operate at frequencies near or exceeding the $f_{max}$ of a process. To overcome indeterminism near the switching threshold of these latches a hysteresis buffer can be cascaded to ensure proper operation when the ratio of sample to alias frequency is high. For DC bias, mismatch and asymmetry in a latch that arise from process variations, a duty-cycle detection mechanism has been developed, and has been shown to able to re-adjust the switching threshold of a purposely mismatched differential latch design, restoring high frequency operation with such a correction mechanism enabled. Transient simulations with a SAFF based-latch in 90-nm process technology show proper aliasing at 230 GHz, which exceeds the 170 GHz $f_{max}$ of a typical 90-nm process by 30 %, and it is likely that better optimized latches can be designed.

## 6.2 Future Work

Although this thesis has laid much of the groundwork for the development of an ALL, several of the concepts presented in this work require further investigation. More in depth study of several of the concepts introduced could be carried out. Most importantly to establishing the ALL approach as a feasible method of frequency synthesis, however, is actual verification of this concept through an implementation in silicon. In fact, several chips that test various aspects of ALL frequency synthesizers can be implemented. A reasonable first goal may be to implement a lower frequency ALL that demonstrates the viability of this approach in silicon. Based on the results from this first design, additional designs can make a push for higher frequencies.

For this thesis only two high performance sampling latch circuits were evaluated. It is likely that better performing latch architectures for subsampling exist and it may be interesting to examine a wider variety of latch architectures. Although simulations can give an indication of the performance achievable with these circuits, more reliable verification should be done through actual implementation in silicon. One approach may be to combine an integrated VCO with a number of candidate latch designs onto a chip, fabricate this chip, and perform a side-by-side comparison of various designs. Results from the physical implementation of these latches can then be correlated to the performance figures predicted by simulations.

From an architecture perspective several alternative implementation approaches could be examined as well. Because the phase error is discretized, this design naturally lends itself to further digital processing. Thus, rather than relying on analog charge-pumps, phase detectors and loop filters, the phase error could be digitized and entirely processed by digital means.

One outstanding problem that may arise in an ALL is that the VCO frequency is not uniquely determinable from just the sample and reference frequencies. To overcome this, multiple sampling frequencies could be used to uniquely determine the frequency the VCO is operating at. This would require the development and evaluation of some system that utilizes multiple sampling frequencies to ensure the VCO can only lock at the desired frequency.

Finally, a more in depth analysis of the effect of discretizing phase error may prove useful. Although the simulation model described in this thesis appears to be accurate in predicting the jitter and distribution of frequencies of the VCO signal, it is restricted to a charge-pump based ALL with a simple RC filter. This simulation model could be made more generic and extended to a variety of architectures used in PLLs. For those more mathematically inclined, a more formal analysis of the conditions that give rise to limit cycles in phase-space plots of the loop-filter voltage and phase error might be interesting. A mathematically rigorous analysis could be useful for developing a concise description of the conditions that give rise to optimal output signal statistics. This type of model should provide guidance on appropriate choices for the reference and sample clock frequencies and the timing and frequency relationship between these clocks.

# Bibliography

[1] A. Rokita, "Direct analag synthesis modules for an X-Band frequency source," in *Proc. Int. Conf. Microw. and Radar (MIKON)*, May 1998, pp. 63–68.

[2] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, April 1965.

[3] H. Li, B. Jagannathan, J. Wang, T. chi Su, S. Sweeney, J. J. Pekarik, Y. Shi, D. Greenberg, Z. Jin, R. Groves, L. Wagner, and S. Csutak, "Technology scaling and device design for 350 GHz RF performance in a 45nm bulk CMOS process," in *Proc. Symp. VLSI Technol.*, June 2007, pp. 56–57.

[4] Y. K. Chen, Y. Baeyens, N. Weimann, J. Lee, J. Weiner, V. Houtsma, and Y. Yang, "Recent advances in III-V electronics," in *Proc. Custom Integr. Circuits Conf. (CICC)*, Sep 2006, pp. 687–690.

[5] N. Fong, J. Kim, J.-O. Plouchart, N. Zamdmer, D. Liu, L. Wagner, C. Plett, and G. Tarr, "A low-voltage 40-GHz complementary VCO with 15% frequency tuning range in SOI CMOS technology," *IEEE J. Solid-State Circuits*, vol. 39, no. 5, pp. 841–846, May 2004.

[6] C. Cao and K. K. O, "A 140-GHz fundamental mode voltage-controlled oscillator in 90-nm CMOS technology," *IEEE Microw. Wireless Compon. Lett.*, vol. 16, no. 10, pp. 555–557, Oct 2006.

[7] C. Cao, E. Seok, and K. K. O, "192 GHz push-push VCO in 0.13 μm CMOS," *IEE Electron. Lett.*, vol. 42, no. 4, pp. 208–210, Feb 2006.

[8] E. Seok, C. Cao, D. Shim, D. J. Arenas, D. B. Tanner, C.-M. Hung, and K. K. O, "A 410 GHz CMOS push-push oscillator with an on-chip patch antenna," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, Feb 2008, pp. 472–474.

[9] D. Huang, T. R. LaRocca, L. Samoska, A. Fung, and M.-C. F. Chang, "324 GHz CMOS frequency generator using linear superposition technique," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, Feb 2008, pp. 476–477.

[10] K. Shu and E. Sánchez-Sinencio, *CMOS PLL Synthesizers: Analysis and Design*. Springer, 2005.

[11] M. J. Underhill, "Frequency control in radio-from quartz to direct digital synthesis," in *Proc. Int. Conf. on 100 Years of Radio*, Sep 1995, pp. 167–176.

[12] E. V. Appleton, "The automatic synchronization of triode oscillators," *Proc. Cambridge Philosophical Soc.*, vol. 21, pp. 231–248, 1922-1923.

[13] H. De Bellescize, "La réception synchrone," *L'onde électrique*, vol. 11, pp. 225–240, May 1932.

[14] R. B. Sepe and R. I. Johnston, "Frequency multiplier and frequency waveform generator," U.S. Patent 3 551 826, Dec 29, 1970.

[15] I. Zamke and S. Zamek, "Definitions of jitter measurement terms and relationships," in *Proc. Int. Test Conf. (ITC)*, Nov 2008, pp. 1–10.

[16] D. M. Pozar, *Microwave Engineering*, 3rd ed. John Wiley & Sons, Inc., 2005.

[17] K. Iniewski, *Wireless Technologies: Circuits, System and Devices*. CRC Press, 2007.

[18] J. Tierney, C. M. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Trans. Audio Electroacoust.*, vol. 19, no. 1, pp. 48–57, Mar 1971.

[19] U. J. Lyles, T. Copani, B. Bakkaloglu, and S. Kiaei, "An injection-locked frequency-tracking $\sigma\delta$ direct digital frequency synthesizer," *IEEE Trans. Circuits Syst. II*, vol. 54, no. 5, pp. 402–406, May 2007.

[20] R. Richter and H.-J. Jentschel, "A virtual clock enhancement method for DDS using an analog delay line," *IEEE J. Solid-State Circuits*, vol. 36, no. 7, pp. 1158–1161, Jul 2001.

[21] C. E. Shannon, "Communication in the presence of noise," *Proc. of the Institute of Radio Eng. (IRE)*, vol. 37, no. 1, pp. 10–21, Jan 1949.

[22] K. Elliott, "High speed direct digital synthesis for next generation rf systems," in *Proc. Radio and Wireless Symp.*, Jan 2007, pp. 423–426.

[23] S. Cheng, J. R. Jensen, R. E. Wallis, and G. L. Weaver, "Further enhancements to the analysis of spectral purity in the application of practical direct digital synthesis," in *Proc. Frequency Control Symp. and Exposition*, Aug 2004, pp. 462–470.

[24] B. W. Garlepp, K. S. Donnelly, J. Kim, P. S. Chau, J. L. Zerbe, C. Huang, C. V. Tran, C. L. Portmann, D. Stark, Y.-F. Chan, T. H. Lee, and M. A. Horowitz, "A portable digital DLL for high-speed CMOS interface circuits," *IEEE J. Solid-State Circuits*, vol. 34, no. 5, pp. 632–644, May 1999.

[25] R. Farjad-Rad, W. J. Dally, H.-T. Ng, R. Senthinathan, M.-J. Edward Lee, R. Rathi, and J. Poulton, "A low-power multiplying DLL for low-jitter multigigahertz clock generation in highly integrated digital chips," *IEEE J. Solid-State Circuits*, vol. 37, no. 12, pp. 1804–1812, Dec 2002.

[26] G. Chien and P. R. Gray, "A 900-MHz local oscillator using a DLL-based frequency multiplier technique for PCS applications," *IEEE J. Solid-State Circuits*, vol. 35, no. 12, pp. 1996–1999, Dec 2000.

[27] B. M. Helal, M. Z. Straayer, G.-Y. Wei, and M. H. Perrot, "A highly digital MDLL-based clock multiplier that leverages a self-scrambling time-to-digital converter to achieve subpicosecond jitter performance," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 855–863, Apr 2008.

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

[29] L. Zhai, Y. Jiang, X. Ling, and W. Gao, "DDS-driven PLL frequency synthesizer for X-band radar signal simulation," in *Proc. Int. Symp. Systems and Control in Aerosp. and Astronautics*, Jan 2006, pp. 344–346.

[30] C. Wagner, A. Stelzer, and H. Jäger, "A 77-GHz radar transmitter with parallelised noise shaping DDS," in *Proc. European Radar Conf.*, Sep 2006, pp. 335–338.

[31] A. Bonfanti, F. Amorosa, C. Samori, and A. Lacaita, "A DDS-based PLL for 2.4-GHz frequency synthesis," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 12, pp. 1007–1010, Dec 2003.

[32] J.-Y. Lee, S.-H. Lee, H. Kim, and H.-K. Yu, "A 28.5–32-GHz fast settling multichannel PLL synthesizer for 60-GHz WPAN radio," *IEEE Trans. Microw. Theory Tech.*, vol. 56, no. 5, pp. 1234–1246, May 2008.

[33] Y. A. Eken and J. P. Uyemura, "Multiple-GHz ring and LC VCOs in 0.18 µm CMOS," in *Proc. Radio Frequency Integr. Circuits (RFIC)*, Jun 2004, pp. 475–478.

[34] A. M. Fahim, *Clock Generators for SOC Processors.* Kluwer Academic Publishers, 2005.

[35] N. Da Dalt, "A design-oriented study of the nonlinear dynamics of digital bang-bang PLLs," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 1, pp. 21–31, Jan 2005.

[36] S. Cheng, H. Tong, J. Silva-Martinez, and A. I. Karşilayan, "Steady-state analysis of phase-locked loops using binary phase detector," *IEEE Trans. Circuits Syst. II*, vol. 54, no. 6, pp. 892–832, Jun 2007.

[37] J. Lee, M. Liu, and H. Wang, "A 75-GHz phase-locked loop in 90-nm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1414–1426, Jun 2008.

[38] B. Sklar, *Digital Communications*, 2nd ed. Prentice Hall P T R, 2001.

[39] A. N. Hafez and M. Elmasry, "A novel low power low phase-noise PLL architecture for wireless transceivers," in *Proc. 9th Great Lakes Symp. VLSI*, Mar 1999, pp. 306–309.

[40] ——, "A low power monolithic subsampled phase-locked loop architecture for wireless transceivers," in *Proc. Int. Symp. Circuits and Systems (ISCAS)*, vol. 2, Jul 1999, pp. 549–552.

[41] A. N. Hafez and M. I. Elmasry, "Phase locked-loop using sub-sampling," U.S. Patent 6 463 112, Oct 8, 2002.

[42] ——, "Phase locked-loop using sub-sampling," U.S. Patent 6 614 866, Sep 2, 2003.

[43] G. E. Von Dolteren Jr., "Subsampling digitizer-based frequency synthesizer," U.S. Patent 6 603 362, Aug 5, 2003.

[44] M. Funabashi, T. Inoue, K. Ohata, K. Maruhashi, K. Hosoya, M. Kuzuhara, K. Kanekawa, and Y. Kabayashi, "A 60 GHz MMIC stabilized frequency source composed of a 30 GHz DRO and a doubler," in *Proc. Int. Microw. Symp.*, May 1995, pp. 71–74.

[45] P.-H. Chen, M.-C. Chen, and C.-Y. Wu, "An integrated 60-GHz front-end receiver with a frequency tripler using 0.13-μm CMOS technology," in *Proc. Int. Conf. Electron. Circuits and Systems (ICECS)*, Dec 2007, pp. 829–832.

[46] A. Natarajan, A. Komijani, X. Guan, A. Babakhani, and A. Hajimiri, "A 77-GHz phased-array transceiver with on-chip antennas in silicon: Transmitter and local LO-path phase shifting," *IEEE J. Solid-State Circuits*, vol. 41, no. 12, pp. 2807–2819, Dec 2006.

[47] J. Lee and B. Razavi, "A 40-GHz frequency divider in 0.18-μm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 39, no. 4, pp. 594–601, Apr 2004.

[48] S. P. Voinigescu, R. Aroca, T. O. Dickson, S. T. Nicolson, T. Chalvatzis, P. Garcia, C. Garnier, and B. Sautreuil, "Towards a sub-2.5V, 100-Gb/s serial transceiver," in *Proc. Custom Integr. Circuits Conf. (CICC)*, Sep 2007, pp. 471–478.

[49] E. Laskin, M. Khanpour, R. Aroca, K. W. Tang, P. Garcia, and S. P. Voinigescu, "A 95-GHz receiver with fundamental-frequency VCO and static frequency divider in 65nm digital CMOS," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, Feb 2008, pp. 180–181.

[50] T. Mitomo, R. Fujimoto, N. Ono, R. Tachibana, H. Hoshino, Y. Yoshihara, Y. Tsutsumi, and I. Seto, "A 60-GHz CMOS receiver front-end with frequency synthesizer," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 1030–1037, Apr 2008.

[51] H. Hoshino, R. Tachibana, T. Mitomo, N. Ono, Y. Yoshihara, and R. Fujimoto,

"A 60-GHz phase-locked loop with inductor-less prescaler in 90-nm CMOS," in *Proc. European Solid-State Circuits Conf. (ESSCIRC)*, Sep 2007, pp. 472–475.

[52] T. Suzuki, Y. Kawano, Yasuhiro, S. Yamaura, T. Takahashi, K. Makiyama, and T. Hirose, "A 50-Gbit/s 450-mW full-rate 4:1 multiplexer with multiphase clock architecture in 0.13-μm InP HEMT technology," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 637–646, Mar 2007.

[53] T. Chalvatzis, K. H. K. Yau, R. A. Aroca, P. Schvan, M.-T. Yang, and S. P. Voinigescu, "Low-voltage topologies for 40-Gb/s circuits in nanoscale CMOS," *IEEE J. Solid-State Circuits*, vol. 42, no. 7, pp. 1564–1573, Jul 2007.

[54] R. Z. Bhatti, M. Denneau, and J. Draper, "Duty cycle measurement and correction using a random sampling technique," in *Proc. Midwest Symp. Circuits and Systems*, vol. 2, Aug 2005, pp. 1043–1046.

[55] *Efficient Noise Analysis for Complex Non-Periodic Analog/RF Blocks*, Berkeley Design Automation Inc., 2902 Stender Way, Santa Clara, California, USA 95054, 2008.

[56] D. F. Williams, K. A. Remley, and D. C. DeGroot, "Nose-to-nose response of a 20-GHz sampling circuit," in *ARFTG Conference Digest-Spring*, vol. 36, Dec 2000, pp. 1–7.

[57] M. Kim, I. Jung, Y. Kwak, S. Ahn, and C. Kim, "Differential pass transistor pulsed latch," in *Proc. Int. SOC Conference*, Sep 2005, pp. 295–300.

[58] M. Matsui, H. Hara, Y. Uetani, L.-S. Kim, T. Nagamatsu, Y. Watanabe, A. Chiba, K. Matsuda, and T. Sakurai, "A 200 MHz 13 $mm_2$ 2-D DCT macrocell using sense-amplifying pipeline flip-flop scheme," *IEEE J. Solid-State Circuits*, vol. 29, no. 12, pp. 1482–1490, Dec 1994.

[59] B. Nikolic, V. Stojanovic, V. G. Oklobdzija, W. Jia, J. Chiu, and M. Leung, "Sense amplifier-based flip-flop," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, Feb 1999, pp. 282–283.

# Appendix A

# Spice Circuit Simulation Settings

All transistor-level simulations used the Spectre simulator by Cadence, with a version string of *CDS: spectre version 6.0.2 11/10/2005 17:40 (usimlx109).* The design kit used for the simulations is the CMC Microsystems *cmos90nm.3.0* release of ST Microelectronics' 90 nm General Purpose CMOS design kit. The simulation models in this design kit are not intended for RF design, and do not model certain parameters such as gate resistance, and simulation results at frequencies near or exceeding $f_{max}$ are therefore most likely unrealistically optimistic. The process corners for all circuit simulations is set to nominal for every device type.

## A.1 Sample Circuit Simulations

All simulations available in section 4.2 are done using transient analysis use the simulation parameters set out in table A.2. All these parameters are the default when simulated using $errpreset = conservative$, with exception of the $step$ and $maxstep$ parameters, which have been set manually. These parameters are further described in the document *Virtuoso Spectre Circuit Simulator Reference, Product Version 6.0, November 2005* and is available from Cadence Design Systems Inc. The supply voltage is set to 1.2 V. The simulation parameters and transient noise settings are given in table A.1 and are valid for all circuits, except the standard cell D flip-flop in section 4.2.2.

Table A.1: Latch circuit transient noise simulation settings

| Parameter | Value |
|---|---|
| noiseseed | 1 |
| noisefmax | 1 THz |
| noisescale | 1 |
| noisefmin | 10 MHz |
| noisetmin | default (1/noisefmax) |

Table A.2: Latch circuit transient simulation parameters

| Parameter | Value |
|---|---|
| step | 100 fs |
| maxstep | 200 fs |
| ic | all |
| skipdc | no |
| reltol | 1e-06 |
| abstol(I) | 1 pA |
| abstol(V) | 1 uV |
| temp | 27 C |
| tnom | 27 C |
| tempeffects | all |
| errpreset | conservative |
| method | gear2only |
| lteratio | 10 |
| relref | alllocal |
| cmin | 0 F |
| gmin | 1 fS |
| maxrsd | 0 Ω |
| mos_method | s |
| mos_vres | 50 mV |

## A.2 Duty Cycle Detection Simulation

The data for Figure 4.20 was obtained by simulating the design for 1 μs with a supply voltage of 1 V and with the simulation parameters set out in table A.3. The remaining simulation data provided in section 4.3 are simulated with a supply voltage of 1.2 V and the simulation parameters in table A.2.

Table A.3: Left DAC output voltage transient simulation parameters

| Parameter | Value |
|---|---|
| step | 800 ps |
| maxstep | 80 ns |
| ic | all |
| skipdc | no |
| reltol | 10e-03 |
| abstol(I) | 1 pA |
| abstol(V) | 1 uV |
| temp | 27 C |
| tnom | 27 C |
| tempeffects | all |
| errpreset | liberal |
| method | gear2 |
| lteratio | 3.5 |
| relref | allglobal |
| cmin | 0 F |
| gmin | 1 pS |
| maxrsd | 0 Ω |
| mos_method | s |
| mos_vres | 50 mV |

## A.3 90-nm ALL model

The 90-nm simulation used to generate the plots in Figures 5.13, 5.14 and 5.15 was simulated for 5 µs with standard simulator settings. These parameters are provided in table A.4. The pull-in simulation for figure 5.16 was simulated for 10 µs, with all the same parameters as in table A.4. Unlike simulations of the sampling latch circuits and duty cycle correction circuits, these circuits are simulated with a supply voltage of 1 V.

Table A.4: Simulation parameters for generation of phase-space orbit and pull-in plots

| Parameter | Value |
|---|---|
| step | 5 ns |
| maxstep | 100 ns |
| ic | all |
| skipdc | no |
| reltol | 1e-03 |
| abstol(I) | 1 pA |
| abstol(V) | 1 uV |
| temp | 27 C |
| tnom | 27 C |
| tempeffects | all |
| errpreset | moderate |
| method | traponly |
| lteratio | 3.5 |
| relref | sigglobal |
| cmin | 0 F |
| gmin | 1 pS |
| maxrsd | 0 Ω |
| mos_method | s |
| mos_vres | 50 mV |

# Appendix B

# Source Code

## B.1 Alias-Locked Loop Simulation Program

```
/* Copyright 2008 Leendert van den Berg
 * C-based implementation of alias-locked loop non-linear model
 * Uses the reference clock time to recalculate the state each cycle
 *
 * Compile with:
 * gcc -o phaseAll -lm phaseAll.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_ARGS (13)
#define NDIV_START (0)
#define VC_TABLE_LEN (64)
#define MSG_LEN (256)
#define MODR(x,y)     ((x) - round((x)/(y))*(y))
#define INC_VC(var)   (var = (((var)+1) % VC_TABLE_LEN))
#define NO_PHI (999.0L)

typedef struct {
    long double phaseRem;
    long double timeRem;
    long double phi;
    long double Vc;
} Vc_history;

int main(int argc, char *argv[])
{
    int num_iter, i, j, Ndiv, numVco, numRef, Div;
    long double vcoPhase, newPhi, sqrt_a, sqrt_b, sqrt_c;
    long double Tv, Ts, Delta;
    long double Tr, Phi, Iin, C, R, Kv, F0;
    long double Vc, VcEff;
    long double current_time;
    long double Psi, Ttarget;
    long double VcOld;
    long double phase, newPhase;
    long double Talias, intPart;
    long double secTime;
    char msg[MSG_LEN];
    Vc_history vc_table[VC_TABLE_LEN];
    int vc_head = 0, vc_tail=0;
    if (argc != NUM_ARGS) {
        printf("Format: %s <number of sample iterations> <Sample period (Ts)> "
            "<Sample delay from VCO (Delta0)> <Reference period (Tr)> "
            "<Reference lead from sample (Phi0)> <Current (I)> <Loop Cap. (C)> "
            "<Loop Res. (R)> <VCO gain (Kv)> <VCO base (F0)> <Filter Voltage "
            "(Vc)> <Divider ratio (Div)>\n", argv[0]);
        return 1;
    } else {
        num_iter = atoi(argv[1]);
        Ts = strtold(argv[2], NULL);
        Delta = strtold(argv[3], NULL);
        Tr = strtold(argv[4], NULL);
        Phi = strtold(argv[5], NULL);
        Iin = strtold(argv[6], NULL);
        C = strtold(argv[7], NULL);
        R = strtold(argv[8], NULL);
        Kv = strtold(argv[9], NULL);
        F0 = strtold(argv[10], NULL);
        Vc = strtold(argv[11], NULL);
        Div = atoi(argv[12]);
        printf("# Running %d iterations, Ts = %Le, Delta0 = %Le\n",
                num_iter, Ts, Delta);
        printf("# Tr = %Le, Phi0 = %Le, Iin = %Le, \n",Tr, Phi, Iin);
        printf("# C = %Le, R=%Lf, Kv = %Le, F0 = %Le\n", C, R, Kv, F0);
        printf("# Vc = %Lf, Div=%d\n", Vc, Div);
        Tv = 1/(F0 + Kv * Vc);
        /* Calculate target frequency based on closes current frequency that
         * matches sampling + reference frequencies:
         * Ttarget = 1 /
         *     ( ratio of sample/vco frequencies + Div * reference frequency)
         */
        Ttarget = 1/((round(Ts/Tv)/Ts) + Div/Tr);
        printf("# Fref = %Le, Fsample = %Le, Fvco = %Le\n", 1/Tr, 1/Ts, 1/Tv);
        printf("# Target Tv = %11.7Le (%Le)\n", Ttarget, 1/Ttarget);
        printf("# modr = %0.8Le; remainderl = %0.8Le\n", MODR(Ts, Tv),
                remainderl(Ts,Tv));
    }
    current_time = 0;
    /* Initial phase error between VCO
     * = number of sample periods in phi, scaled by the ratio of sample
     * frequency to reference frequency times the ideal VCO clock, minus the
     * small time difference given by delta.
     *
     * ==> Round up Phi up to nearest multiple fraction of reference period
     * (fraction as in number of samples per reference clock). Times the target
     * VCO clock (i.e. if Phi is 10% of, VCO phase error is 10 % of the VCO
     * clock, less the offset between VCO clock and sampler). This fraction is
     * further corrected for the division factor
     */
    Psi = (ceill(Phi/Ts)*Div*Ts/Tr) * Ttarget - Delta;
    numVco = 0;
    numRef = 0;
    vcoPhase = 0;

    /* Calculate initial phase offset between sample clock and VCO using
     * initial VCO frequency
     */
    Tv = 1/(F0 + Kv * Vc);
    /* Initial phase before new voltage takes effect from delta */
    phase = Delta/Tv;

    newPhase = 0.0L;

    printf("#ind  2 Time    3 Ndiv   4 Delta       5 phi            6 Vc"
           "                7 Tv                            10 Psi\n");
    printf("# Psi initial = %10Le , Phi/Tr = %Lf (ceil=%Lf)\n", Psi, Phi/Tr,
        ceill(Phi/Ts)*Ts/Tr);
```

```
/* Find the alias period for the first section of the waveform (no charging)
 * => Phi = 0 */
Ndiv = NDIV_START;
for (j=0; j < Div; j++) {
    do {
        Ndiv++;
        Talias = Ndiv*Ts;
        if (Phi - Tr + Talias < 0 ) {
            /* Still in flat section of Vc */
            newPhase = phase + (F0 + Kv * Vc) * Talias;
        } else {
            /* Must keep track of charging section now, as reference edge
             * has passed and Vc is changing */
            newPhi = Phi - Tr + Talias;
            newPhase = phase + (F0 + Kv * Vc) * Talias +
                newPhi * Iin * Kv * ( R + fabsl(newPhi)/C/2);
        }
    } while (modfl(newPhase, &intPart) < 0.5L);
    Ndiv--;
    do {
        Ndiv++;
        Talias = Ndiv*Ts;
        if (Phi - Tr + Talias < 0 ) {
            /* Still in flat section of Vc */
            newPhase = phase + (F0 + Kv * Vc) * Talias;
        } else {
            /* Must keep track of charging section now, as reference edge
             * has passed and Vc is changing */
            newPhi = Phi - Tr + Talias;
            newPhase = phase + (F0 + Kv * Vc) * Talias +
                newPhi * Iin * Kv * ( R + fabsl(newPhi)/C/2);
        }
    } while (modfl(newPhase, &intPart) >= 0.5L);
}

phase = modfl(newPhase, &intPart);

/* Add initial vc table entry based on no charging */
vc_table[vc_head].phi = NO_PHI;
vc_table[vc_head].Vc = Vc;
if (Phi - Tr + Talias > 0 ) {
    vc_table[vc_head].timeRem = Tr - Phi + Delta;
} else {
    vc_table[vc_head].timeRem = Talias + Delta;
}
vc_table[vc_head].phaseRem = vc_table[vc_head].timeRem * (F0 + Kv * Vc);
INC_VC(vc_head);


i = 0;
printf("%3d %11.5Le % 4d % 9.7Le % 9.5Le %14.12Le %9.5Le % 41d % 4d "
    "% 14.12Le\n", i, current_time, Ndiv, Delta, Phi, Vc, Tv,
    lround(i*Tr/Ttarget) - numVco, numRef, Psi);

/* calculate new Phi */
Phi = Phi - Tr + Talias;

msg[0] = '\0';
/* Loop through all iterations requested */
for (i=1; i <= num_iter+1; i++) {
    current_time += Tr;

    /* Prepare for next iteration */
    VcOld = Vc;
    Vc = Vc + Iin * Phi/C;
```

```
/* Estimate effect of resistor & charging ___/‾‾‾‾‾‾‾‾‾\_____  */
VcEff = Vc + Iin * (R*Phi/Tr - fabsl(Phi)*Phi/(Tr*C*2));
/* Initial estimate of VCO period to start phase calculation */
Tv = 1/(F0 + Kv * VcEff);

/* Calculate duration of next alias period (including any frequency
 * division factor)
 * Update for actual voltage wave form shape
 * Steps: goes through sloping section, then flat section, then if
 * next Phi < 0, new sloping section */
Ndiv = NDIV_START;
for (j=0; j < Div; j++) {
    do {
        Ndiv++;
        Talias = Ndiv*Ts;
        if (Phi - Tr + Talias < 0 ) {
            /* Still in flat section of Vc */
            newPhase = phase + (F0 + Kv * Vc) * Talias;
        } else {
            /* Must keep track of charging section now, as reference
             * edge has passed and Vc is changing */
            newPhi = Phi - Tr + Talias;
            newPhase = phase + (F0 + Kv * Vc) * Talias +
                newPhi * Iin * Kv * ( R + fabsl(newPhi)/C/2);
        }
    } while (modfl(newPhase, &intPart) < 0.5L);
    Ndiv--;
    do {
        Ndiv++;
        Talias = Ndiv*Ts;
        if (Phi - Tr + Talias < 0 ) {
            /* Still in flat section of Vc */
            newPhase = phase + (F0 + Kv * Vc) * Talias;
        } else {
            /* Must keep track of charging section now, as reference
             * edge has passed and Vc is changing */
            newPhi = Phi - Tr + Talias;
            newPhase = phase + (F0 + Kv * Vc) * Talias +
                newPhi * Iin * Kv * ( R + fabsl(newPhi)/C/2);
        }
    } while (modfl(newPhase, &intPart) >= 0.5L);
}

phase = modfl(newPhase, &intPart);
numVco += intPart;

/* Charging section of Vc waveform */
vc_table[vc_head].phi = Phi;
vc_table[vc_head].Vc = VcOld;
vc_table[vc_head].timeRem = fabsl(Phi);
vc_table[vc_head].phaseRem = (F0 + Kv * VcOld) * fabsl(Phi) +
    Phi * Iin * Kv * ( R + fabsl(Phi)/C/2);
INC_VC(vc_head);
if (vc_head == vc_tail) {
    fprintf(stderr, "Iteration %d: No more VCO table entries "
        "available\n", i);
    break;
}

/* Flat section of Vc waveform */
vc_table[vc_head].phi = NO_PHI;
vc_table[vc_head].Vc = Vc;

if (Phi > 0 && Phi - Tr + Talias > 0) {
    vc_table[vc_head].timeRem = Tr-Phi;
} else if (Phi < 0 && Phi - Tr + Talias > 0) {
        vc_table[vc_head].timeRem = Tr;
    } else if (Phi > 0 && Phi - Tr + Talias < 0) {
        vc_table[vc_head].timeRem = Talias;
    } else {
        /* Phi < 0 && Phi - Tr + TAlias < 0
         * Talias - |Phi| = Talias + Phi */
        vc_table[vc_head].timeRem = Talias + Phi;
    }

    vc_table[vc_head].phaseRem = vc_table[vc_head].timeRem * (F0 + Kv * Vc);

    INC_VC(vc_head);
    if (vc_head == vc_tail) {
        fprintf(stderr, "Iteration %d: No more VCO table entries "
            "available\n", i);
        break;
    }

    /* Calculate number of new cycles for the ideal VCO */
    vcoPhase = numRef;
    numRef = lround(i*Tr/Ttarget) - lround((i-1)*Tr/Ttarget);

    /* Calculate new phase error */
    Phi = Phi - Tr + Talias;

    if (fabsl(Phi) > Tr || fabsl(Phi) > Talias) {
        fprintf(stderr, "Talias = %Le, phase=%Le, numVco = %i\n", Talias,
            phase, numVco);
        fprintf(stderr, "Iteration %d: Phi > Tr, Talias (%9.5Le > %9.5Le,"
            "%9.5Le)\n", i, fabsl(Phi), Tr, Talias);
        break;
    }

    /* VCO period to calculate an approximate delta */
    Tv = 1/(F0 + Kv * Vc);
    Delta = phase * Tv;


    /* Print current phase error */
    if (i > 1) {
        /* Calculate the time difference between the ideal and actual VCO
         * after numRef cycles, add ideal VCO time, minus actual time */
        Psi -= vcoPhase * Ttarget;


        /* Loop through a phase value of vcoPhase until we hit a table entry
         * that has more phase remaining then the amount we wish to take
         * Each table entry's amount of phase and time occupied is taken off
         * from the vcoPhase and VCO phase error Psi */
        while ((vc_head != vc_tail) &&
                (vc_table[vc_tail].phaseRem < vcoPhase)) {
            vcoPhase -= vc_table[vc_tail].phaseRem;
            Psi += vc_table[vc_tail].timeRem;
            INC_VC(vc_tail);
        }
        if (vc_head == vc_tail) {
            fprintf(stderr, "Iteration %d: No more VCO data available\n",
                i-1);
            break;
        }

        /* Now use the remaining phase to calculate the time difference
         * between the ideal VCO and actual VCO. This also requires
         * updating of the VC table entry from which we are taking phase
         * and time */
        if (vc_table[vc_tail].phi >= NO_PHI) {
                /* Easy to calculate the time taken if the voltage waveform is
                 * flat: time = phase / vco frequency in the flat section */
                secTime = vcoPhase / (F0 + Kv* vc_table[vc_tail].Vc);
                vc_table[vc_tail].timeRem -= secTime;
                vc_table[vc_tail].phaseRem -= vcoPhase;
                Psi += secTime;
            } else {
                /*
                 * must solve square root equation here, solving for time
                 * which satisfies phase equation is given above as:
                 * newPhase = phase + (F0 + Kv * VcOld) * Talias +
                 *            Phi * Iin * Kv * ( R - fabsl(Phi)/C/2) +
                 *            Talias * Kv * Iin * Phi/C;
                 *
                 *         Charging part, initial phase = 0:
                 *         == (Fo + Kv * VcOld) * Phi +
                 *            (R + (Phi / C /2) * Iin * Kv) * Phi
                 *
                 * Rewriting as a x^2 + b x + c for the sloping part,
                 * newPhase = vcoPhase and phi is given by secTime, assume
                 * Phi > 0, move newPhase to right-hand side
                 *
                 *  (Kv * Iin/C/2) * secTime^2 +
                 *  ( F0 + Kv * Vc + Kv * Iin * R) * secTime - vcoPhase
                 *
                 *  Instead of Phi, we solve for time (secTime)
                 */
                if (vc_table[vc_tail].phi < 0) {
                    sqrt_a = Kv * Iin/C/2;
                    sqrt_b = F0 + Kv * vc_table[vc_tail].Vc + Kv * Iin * R;
                    sqrt_c = -1*vcoPhase;
                    secTime = (-sqrt_b +
                        sqrtl(sqrt_b*sqrt_b - 4 * sqrt_a * sqrt_c))/2/sqrt_a;
                } else {
                    /* The case for Phi < 0 is almost the same, with some signs
                     * reversed for discharging */
                    sqrt_a = -1*Kv * Iin/C/2;
                    sqrt_b = F0 + Kv * vc_table[vc_tail].Vc - Kv * Iin * R;
                    sqrt_c = -1*vcoPhase;
                    secTime = (-sqrt_b +
                        sqrtl(sqrt_b*sqrt_b - 4 * sqrt_a * sqrt_c))/2/sqrt_a;
                }
                /* Update the last entry in the table with the time we have
                 * used in the current cycle */
                vc_table[vc_tail].timeRem -= secTime;
                vc_table[vc_tail].phaseRem -= vcoPhase;
                Psi += secTime;
            }

            printf("%s % 14.12Le\n", msg, Psi);
        }
        snprintf(msg, MSG_LEN,
            "%3d %11.5Le % 4d % 9.7Le % 9.5Le %14.12Le %9.5Le % 4ld % 4d",
            i, current_time, Ndiv, Delta, Phi, Vc, Tv,
            lround(i*Tr/Ttarget) - numVco, numRef);

    }

    return 0;
}
```

## B.2 Alias-Locked Loop Histogram Generation Program

```
/* Copyright 2008 Leendert van den Berg
 * C-based implementation of histogram counter. Hardcoded to take the psi value
 * produced by output from the phaseAll.c program
 *
 * Compile with:
 * gcc -o histo -lm histo.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_ARGS (4)
#define SCAN_STRING ("%d %Le %d %Le %Le %Le %Le %ld %d %Le")
#define BUF_LEN (256)


int main(int argc, char *argv[])
{
    int numBins, binIndex, i;
    long valsRead=0;
    long *bins;
    long double minVal, maxVal, binSize;
    char readBuf[BUF_LEN];

    int iter, Ndiv;
    long double current_time, Delta, Phi, Vc, Tv;
    long cycleLen;
    int refCount;
    long double Psi;


    if (argc != NUM_ARGS) {
        fprintf(stderr,"Format: %s <number of bins> <minimum bin value> "
            " <maximum bin value>\n", argv[0]);
        return 1;
    } else {
        numBins = atoi(argv[1]);
        minVal = strtold(argv[2], NULL);
        maxVal = strtold(argv[3], NULL);
    }
    if (numBins < 0) {
        fprintf(stderr, "Number of bins (%d) should be greater than 0\n",
            numBins);
        exit(1);
    }
    if (minVal > maxVal) {
        fprintf(stderr, "Minimum bin value (%14.12Le) should be less than "
            "maximum bin value (%14.12Le)\n", minVal, maxVal);
        exit(1);
    }
    binSize = (maxVal - minVal)/numBins;

    bins = calloc(sizeof(long),numBins);
    if (bins == NULL) {
        fprintf(stderr, "Could not allocate memory for binning\n");
        exit(1);
    }


    while (1) {
        if (fgets(readBuf, BUF_LEN, stdin) == NULL) {
            break;
        }
        if (readBuf[0] == '#') {
            continue;
        }
        sscanf(readBuf, SCAN_STRING, &iter, &current_time, &Ndiv, &Delta, &Phi,
            &Vc, &Tv, &cycleLen, &refCount, &Psi);
        valsRead++;
        if (Vc < minVal || Vc > maxVal) {
            fprintf(stderr,"Iteration %ld value outside bin range %14.12Le\n",
                valsRead, Vc);
        } else {
            binIndex = (Vc - minVal)/(maxVal - minVal) * numBins;
            bins[binIndex]++;
        }
    }

    for (i=0; i < numBins; i++) {
        printf("%14.12Le %ld\n", minVal + i*binSize + binSize/2, bins[i]);
    }

    free(bins);
    bins = NULL;

    return 0;
}
```

## B.3 Latch Sensitivity Plot Generation Program

```
/* Copyright 2008 Leendert van den Berg
 * C-based program to collect data from sensitivity analyis
 * Expects data file with x,y pairs for each pulse width simulated
 *
 * X value should be centre location of pulse, y value should be the latch
 * output value
 *
 * Generates a series of x,y pairs for time offset and sensitivity function
 * at that value
 *
 * Compile with:
 * gcc -o sensMinPulse -lm sensMinPulse.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_ARGS (7)
#define SCAN_XY1 ("%Le,%Le")
#define SCAN_XY (",%Le,%Le")
#define HALF_VDD (0.6)
#define LARGE_VAL (1e15)


int main(int argc, char *argv[])
{
    int numPw, numTimeBins, i, j;
    long valsRead=0;
    long double *bins;
    long double pwMinVal, pwMaxVal, pwSizeStep;
    long double timeMinVal, timeMaxVal, timeBinSize;

    long double xval, yval;

    if (argc != NUM_ARGS) {
        fprintf(stderr ,"Format: %s <number of time offsets> <minimum time "
            " offset> <maximum time offset> <number of pulse widths> "
            "<minimum pulse width value> <maximum pulse width value>\n",
            argv[0]);
        return 1;
    } else {
        numTimeBins = atoi(argv[1]);
        timeMinVal = strtold(argv[2], NULL);
        timeMaxVal = strtold(argv[3], NULL);
        numPw = atoi(argv[4]);
        pwMinVal = strtold(argv[5], NULL);
        pwMaxVal = strtold(argv[6], NULL);
    }
    if (numTimeBins < 1) {
        fprintf(stderr, "Number of time bins (%d) should be greater than 0\n",
            numTimeBins);
        exit(1);
    }
    if (numPw < 1) {
        fprintf(stderr, "Number of pulse width (%d) should be greater than 0\n",
            numPw);
        exit(1);
    }
    if (timeMinVal > timeMaxVal) {
        fprintf(stderr, "Minimum time value (%14.12Le) should be less than "
            "maximum time value (%14.12Le)\n", timeMinVal, timeMaxVal);
        exit(1);
    }
    if (pwMinVal > pwMaxVal) {
        fprintf(stderr, "Minimum pulse width value (%14.12Le) should be less "
            " than maximum pulse width value (%14.12Le)\n", pwMinVal, pwMaxVal);
        exit(1);
    }
    timeBinSize = (timeMaxVal - timeMinVal)/(numTimeBins-1);
    pwSizeStep = (pwMaxVal - pwMinVal)/(numPw-1);


    bins = calloc(sizeof(long double),numTimeBins);
    if (bins == NULL) {
        fprintf(stderr, "Could not allocate memory for binning\n");
    }

    /* Initialize to largest bin value possible */
    for (i=0; i< numTimeBins; i++) {
        //bins[i] = LARGE_VAL;
        bins[i] = pwMaxVal;
        /*
        printf("%14.12Le %Le\n", timeMinVal + i*timeBinSize, bins[i]);
        printf("Bins[% 2d] centred at %Le\n", i, timeMinVal + timeBinSize*i);
        */
    }
/*
    for (i=0; i< numPw; i++) {
        printf("Pw[% 2d] size %Le\n", i, pwMinVal + pwSizeStep*i);
    }
*/

    for (i=0; i < numTimeBins; i++) {
        for (j=0; j < numPw; j++) {
            if (j == 0) {
                if (scanf(SCAN_XY1, &xval, &yval) != 2) {
                    break;
                }
            } else {
                if (scanf(SCAN_XY, &xval, &yval) != 2) {
                    break;
                }
            }
            valsRead++;
            if (xval < timeMinVal || xval > timeMaxVal) {
                fprintf(stderr,"Iteration %ld value outside time bin range "
                        " %14.12Le\n", valsRead, xval);
            } else if ( (xval - (timeMinVal + i * timeBinSize)) >
                        timeBinSize/1000) {
                fprintf(stderr, "Unexpected time value not at time bin centre. "
                    "Line %d, entry %d. Value %Le (expected %Le)\n",
                    i, j, xval, timeMinVal + i* timeBinSize
                    );
            } else {
                /*printf("Read %Le, %Le\n", xval, yval); */
                /*
                if (yval < HALF_VDD) {
                    bins[i] -= 1/(j * pwSizeStep + pwMinVal);
                } else {
                    bins[i] += 1/(j * pwSizeStep + pwMinVal);
                }
                */
                if (yval > HALF_VDD && bins[i] >  j*pwSizeStep + pwMinVal) {
                    bins[i] = j*pwSizeStep + pwMinVal;
                }
            }
        }
    }

    for (i=0; i < numTimeBins; i++) {
        printf("%14.12Le %Le\n", timeMinVal + i*timeBinSize, bins[i]);
    }

    free(bins);
    bins = NULL;

    return 0;
}
```

## B.4 Latch Sampling Window Sensitivity Plot Generation Program

```
/* Copyright 2008 Leendert van den Berg
 * C-based program to collect data from sensitivity analyis
 * Expects data file with x,y pairs for each pulse width simulated
 *
 * X value should be centre location of pulse, y value should be the latch
 * output value
 *
 * Generates a series of x,y pairs for edge time offset and sensitivity
 * function at that value. Assumes square wave signal. Finds minimum pulse width
 * for each edge increment away from central sensitivity
 *
 * Compile with:
 * gcc -o sqPulse -lm sqPulse.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NUM_ARGS (7)
#define SCAN_XY1 ("%Le,%Le")
#define SCAN_XY (",%Le,%Le")
```

```
#define HALF_VDD (0.6)
#define GRIDSIZE (256)


int main(int argc, char *argv[])
{
    int numPw, numTimeBins, i, j, k;
    long valsRead=0;
    long double pwMinVal, pwMaxVal, pwSizeStep;
    long double timeMinVal, timeMaxVal, timeBinSize;
    int dataGrid[GRIDSIZE][GRIDSIZE];
    int curMinPw = GRIDSIZE, minPwStart, minPwStop;

    long double xval, yval;

    if (argc != NUM_ARGS) {
        fprintf(stderr,"Format: %s <number of time offsets> <minimum time "
            " offset> <maximum time offset> <number of pulse widths> "
            "<minimum pulse width value> <maximum pulse width value>\n",
            argv[0]);
        return 1;
    } else {
        numTimeBins = atoi(argv[1]);
        timeMinVal = strtold(argv[2], NULL);
        timeMaxVal = strtold(argv[3], NULL);
        numPw = atoi(argv[4]);
        pwMinVal = strtold(argv[5], NULL);
        pwMaxVal = strtold(argv[6], NULL);
    }
    if (numTimeBins < 1) {
        fprintf(stderr, "Number of time bins (%d) should be greater than 0\n",
            numTimeBins);
        exit(1);
    }
    if (numPw < 1) {
        fprintf(stderr, "Number of pulse width (%d) should be greater than 0\n",
            numPw);
        exit(1);
    }
    if (timeMinVal > timeMaxVal) {
        fprintf(stderr, "Minimum time value (%14.12Le) should be less than "
            "maximum time value (%14.12Le)\n", timeMinVal, timeMaxVal);
        exit(1);
    }
    if (pwMinVal > pwMaxVal) {
        fprintf(stderr, "Minimum pulse width value (%14.12Le) should be less "
            " than maximum pulse width value (%14.12Le)\n", pwMinVal, pwMaxVal);
        exit(1);
    }
    if (numTimeBins > GRIDSIZE || numPw > GRIDSIZE) {
        fprintf(stderr, "Number of pulsewidths or offset to large >%d\n",
                GRIDSIZE);
        exit(1);
    }
    timeBinSize = (timeMaxVal - timeMinVal)/(numTimeBins-1);
    pwSizeStep = (pwMaxVal - pwMinVal)/(numPw-1);



    for (i=0; i < numTimeBins; i++) {
        for (j=0; j < numPw; j++) {
            if (j == 0) {
                if (scanf(SCAN_XY1, &xval, &yval) != 2) {
                    break;
                }
            } else {
                    if (scanf(SCAN_XY, &xval, &yval) != 2) {
                        break;
                    }
                }
                valsRead++;
                if ((xval - timeMinVal) < -1*timeBinSize/100 ||
                    (xval - timeMaxVal) > timeBinSize/100) {
                    fprintf(stderr,"Iteration %ld value outside time bin range "
                            " (%14.12Le)\n", valsRead, xval);
                } else if ( (xval - (timeMinVal + i * timeBinSize)) >
                        timeBinSize/1000) {
                } else if ( (xval - (timeMinVal + i * timeBinSize)) >
                        timeBinSize/1000) {
                    fprintf(stderr, "Unexpected time value not at time bin centre. "
                        "Line %d, entry %d. Value %Le (expected %Le)\n",
                        i, j, xval, timeMinVal + i* timeBinSize
                        );
                } else {
                    if (yval > HALF_VDD) {
                        dataGrid[i][j] = 1;
                        if (curMinPw > j) {
                            curMinPw = j;
                            minPwStart = i;
                            minPwStop = i;
                        } else if (curMinPw == j) {
                            if (i < minPwStart) {
                                minPwStart = i;
                            } else if (i > minPwStop) {
                                minPwStop = i;
                            }
                        }
                    } else {
                        dataGrid[i][j] = 0;
                    }
                }
            }
        }
/*      printf("Found minimum pulse of %.1Lf ps in range (%.1Le ps,%.1Le ps)\n",
            curMinPw * pwSizeStep + pwMinVal, minPwStart*timeBinSize+timeMinVal,
            minPwStop*timeBinSize+timeMinVal);
*/

        for (i = numTimeBins; i>=minPwStop; i--) {
            k = curMinPw;
            for (j=i; ; j++, k+=2) {
/*
                printf("Check %.3Le %.1Lf (%.3Lf .. %.3Lf)\n",
                    j*timeBinSize + timeMinVal,
                    k*pwSizeStep + pwMinVal,
                    1e12*(j*timeBinSize+timeMinVal) - 5e-1*(k*pwSizeStep+pwMinVal),
                    1e12*(j*timeBinSize+timeMinVal) + 5e-1*(k*pwSizeStep+pwMinVal)
                    );
*/
                if (k >= numPw || j >= numTimeBins) {
//                  printf("Outside delay/pulsewidth range\n");
                    break;
                }
                if (dataGrid[j][k]) {
                    printf("%.3Le, %.1Lf\n",i*timeBinSize + timeMinVal,
                        k * pwSizeStep + pwMinVal);
                    break;
                }

            }
        }
```

```
    for (i=minPwStop-1; i > 0; i--) {
        k = curMinPw;
        for (j=i; ; j--, k+=2) {
/*              printf("Check %.3Le %.1Lf (%.3Lf .. %.3Lf)\n",
                  j*timeBinSize + timeMinVal,
                  k*pwSizeStep + pwMinVal,
                  1e12*(j*timeBinSize+timeMinVal) - 5e-1*(k*pwSizeStep+pwMinVal),
                  1e12*(j*timeBinSize+timeMinVal) + 5e-1*(k*pwSizeStep+pwMinVal)
                  );
*/
            if (k >= numPw ) {
                break;
            }
            if (k >= numPw || j < 0) {
                break;
            }
            if (dataGrid[j][k]) {
                printf("%.3Le, %.1Lf\n",i*timeBinSize + timeMinVal,
                    k * pwSizeStep + pwMinVal);
                break;
            }

        }
    }

    return 0;
}
```

# Appendix C

# Circuit Diagrams for 90-nm Implementation

## C.1 Latches and Latch Test Benches

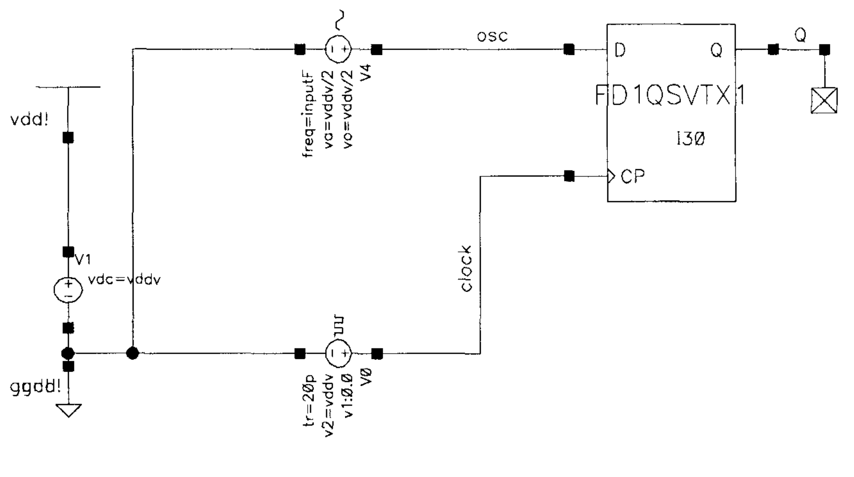

Figure C.1: dflipflop_tb: D flip-flop maximum frequency test bench

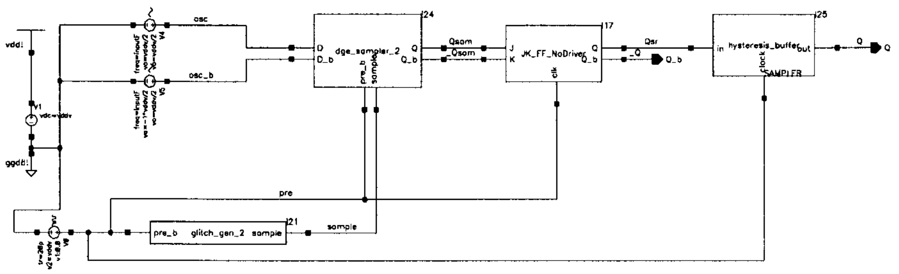

Figure C.2: dge_sampler_no_offset_tb: DPTPL maximum frequency test bench

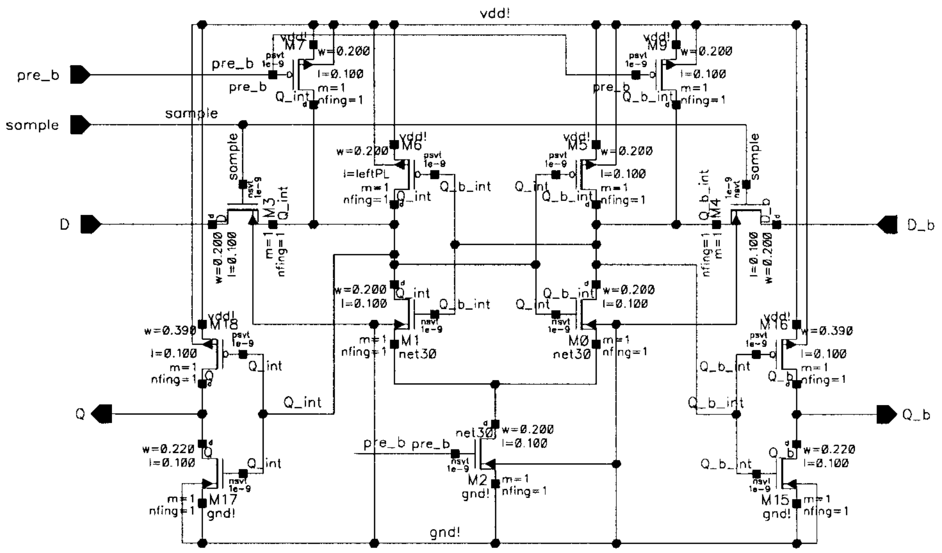

Figure C.3: dge_sampler_no_offset: DPTPL schematic

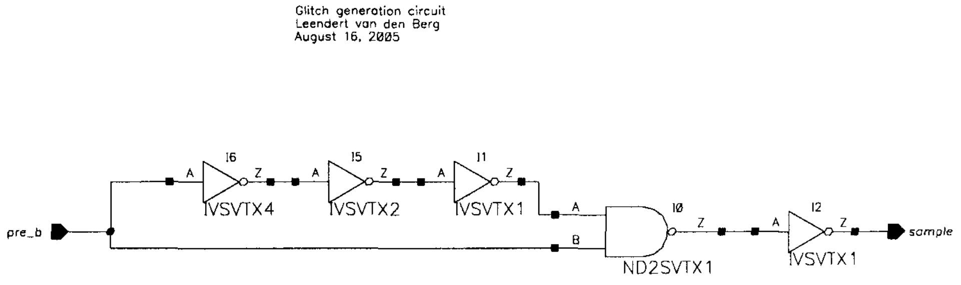

Figure C.4: pulse_gen: DPTPL pulse generator circuit

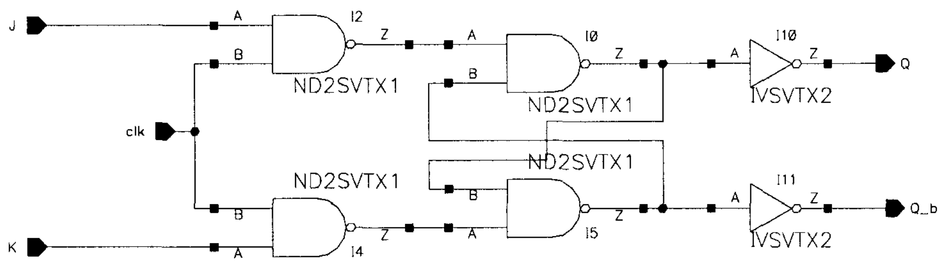

Figure C.5: JK_FF_NoDriver: JK flip-flop, cascaded after the SAFF and DPTPL circuits to hold the output during during precharge of the latch

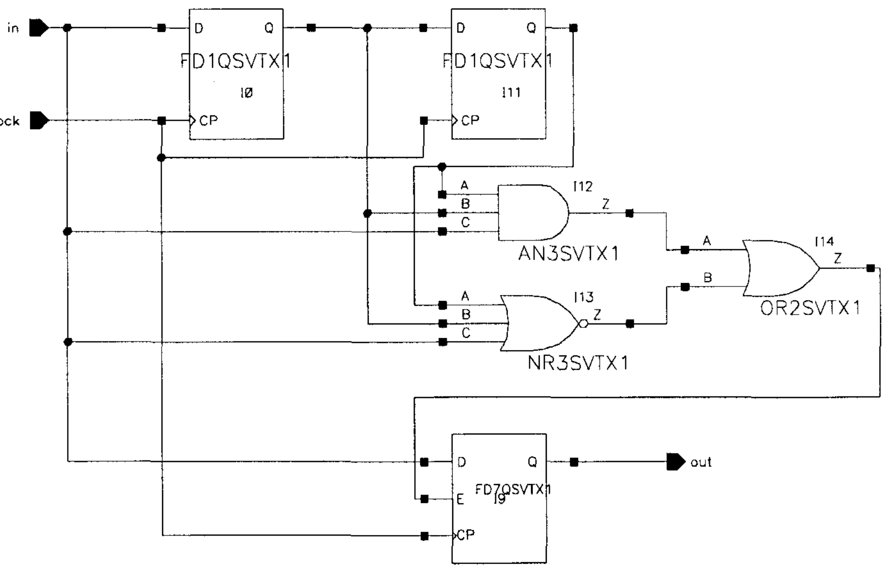

Figure C.6: hysteresis_buffer: two memory-element hysteresis buffer for DPTPL and SAFF

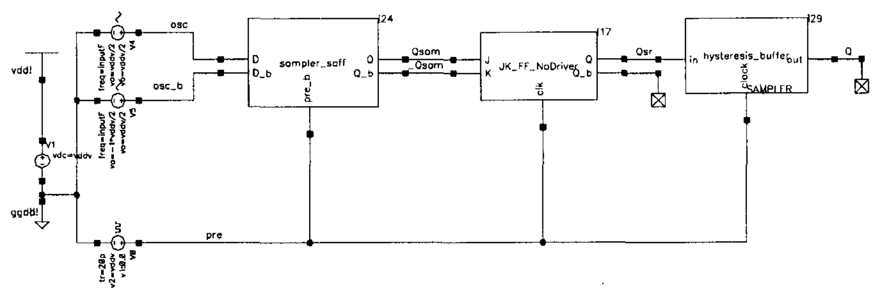

Figure C.7: sampler_saff_tb: SAFF maximum frequency test bench

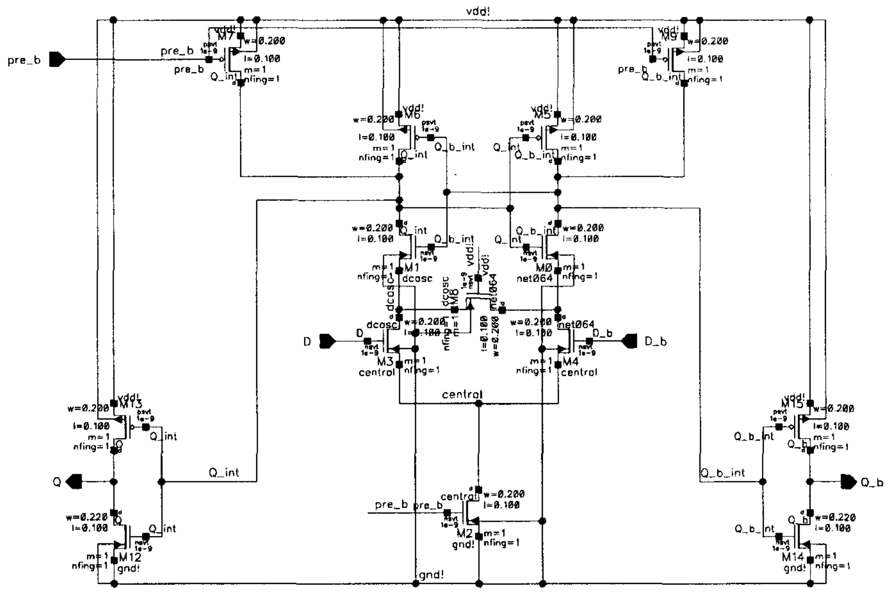

Figure C.8: sampler_saff: SAFF schematic

## C.2 Latch Threshold Correction and Detection Circuits

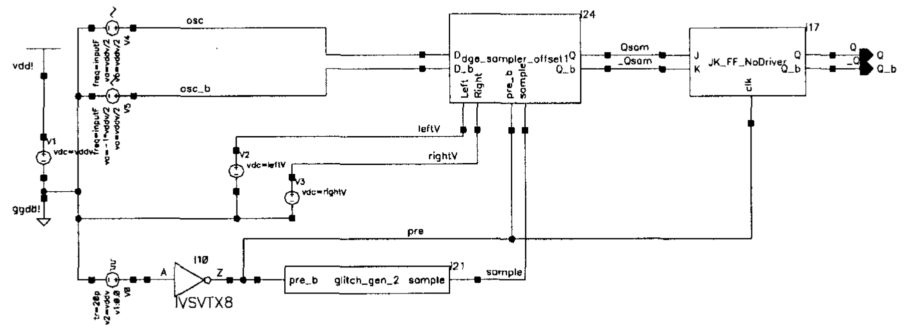

Figure C.9: dge_sampler_offset_test: DPTPL offset transistor test bench

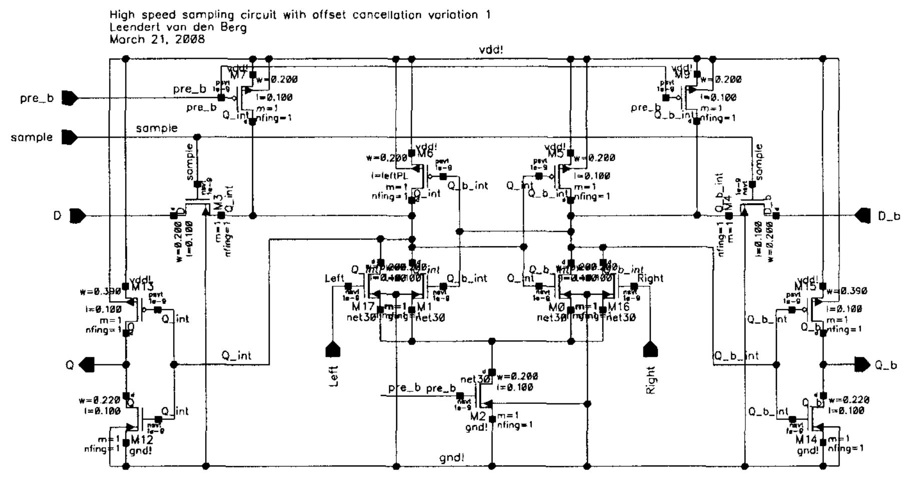

Figure C.10: dge_sampler_offset: DPTPL with offset transistors

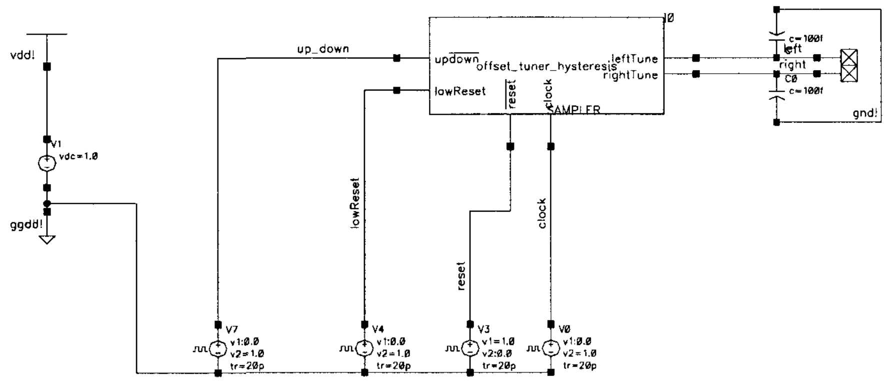

Figure C.11: offset_tuner_test: Test bench for duty-cycle based latch threshold correction circuit

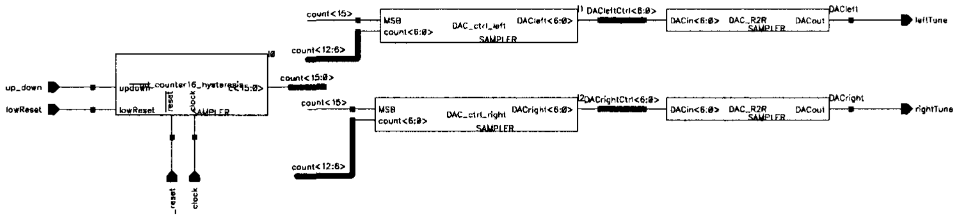

Figure C.12: offset_tuner_hysteresis: Duty cycle detection and correction with hysteresis

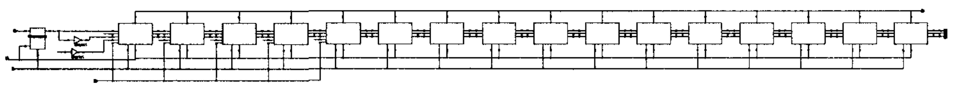

Figure C.13: ud_counter16_hysteresis: 16-bit up/down counter with separate LSB reset (hysteresis)

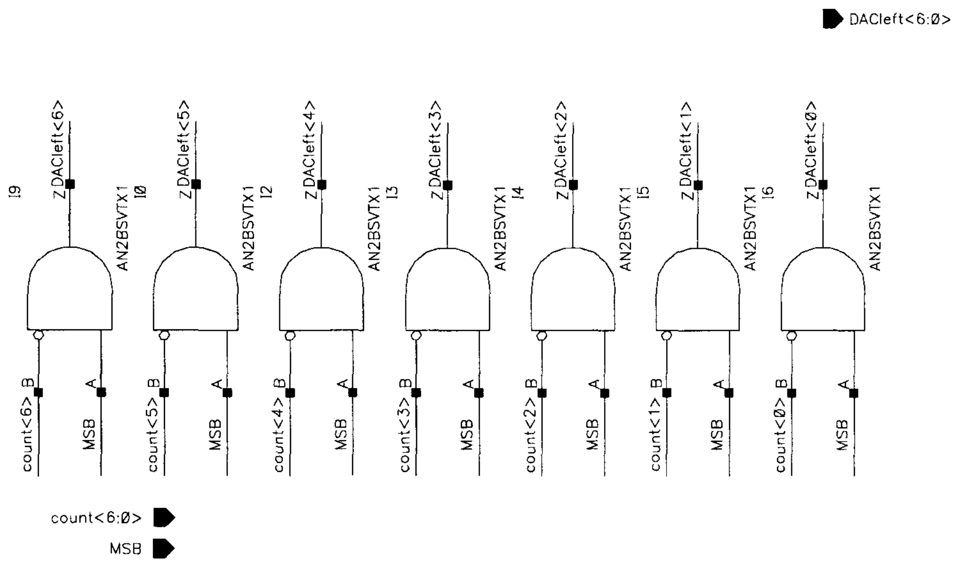

Figure C.14: DAC_ctrl_left: Control signal generation for DAC for left tuning signal

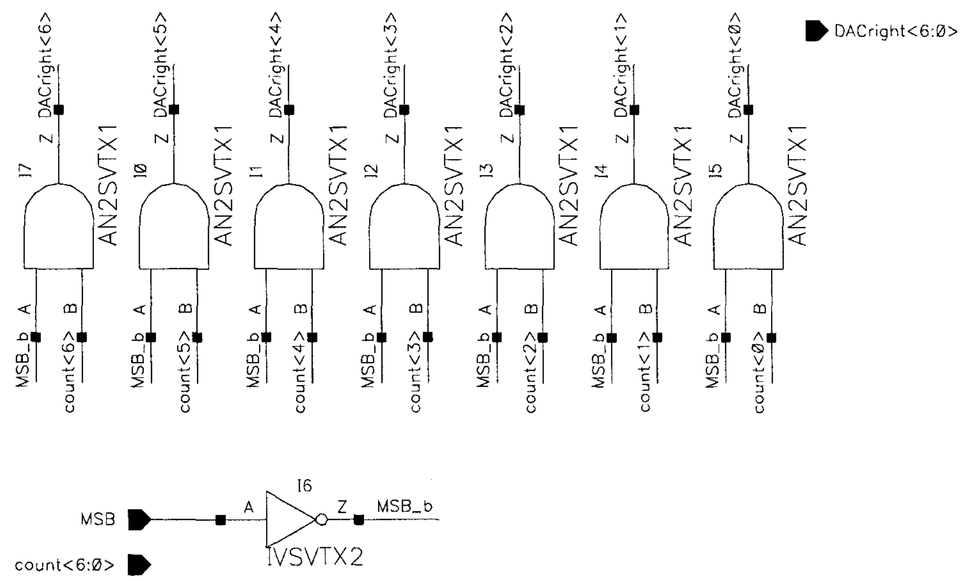

Figure C.15: DAC_ctrl_right: Control signal generation for DAC for right tuning signal

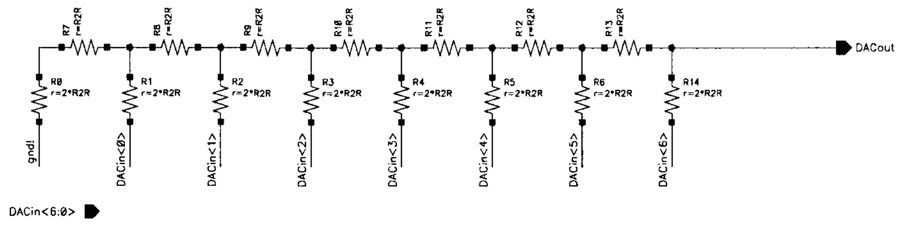

Figure C.16: DAC_R2R: DAC based on R-2R ladder network

## C.3 90-nm CMOS ALL Circuit Implementation

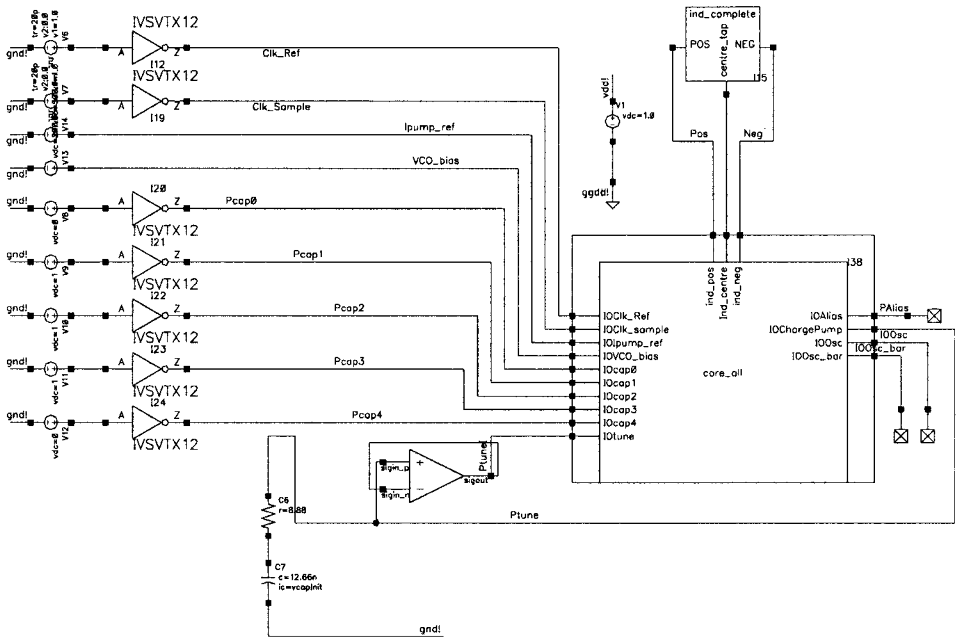

Figure C.17: ALL_core_only_test: Test bench for 90-nm ALL circuit implementation

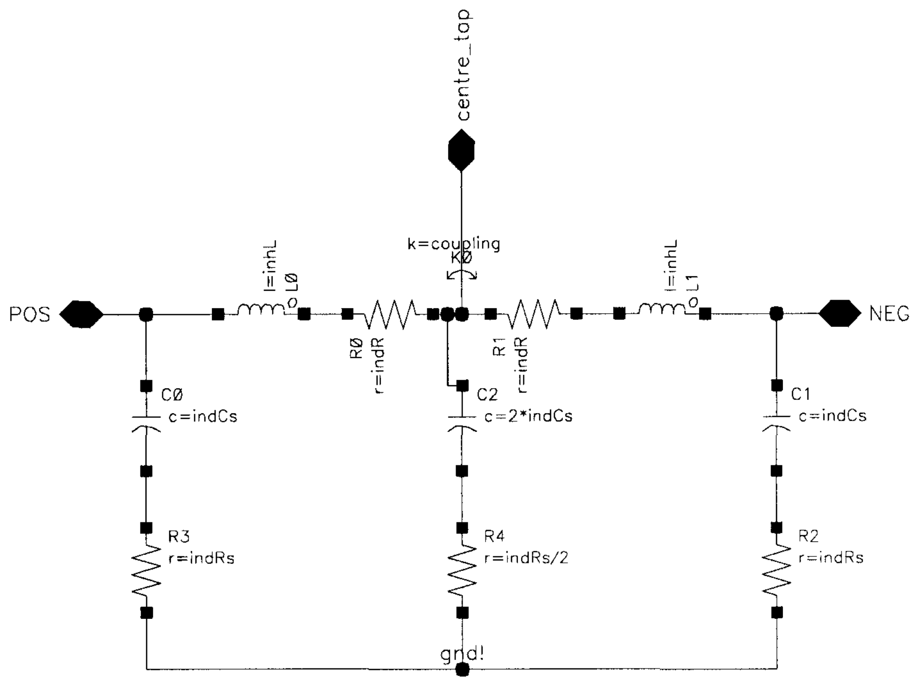

Figure C.18: ind_complete: ASITIC Pi model for a 0.9 nh inductor. Parameters: indRs = 1.8 $k\Omega$; indC = 29 pF; inhL = 285 pH; indR = 0.5 $\Omega$; k = 0.495

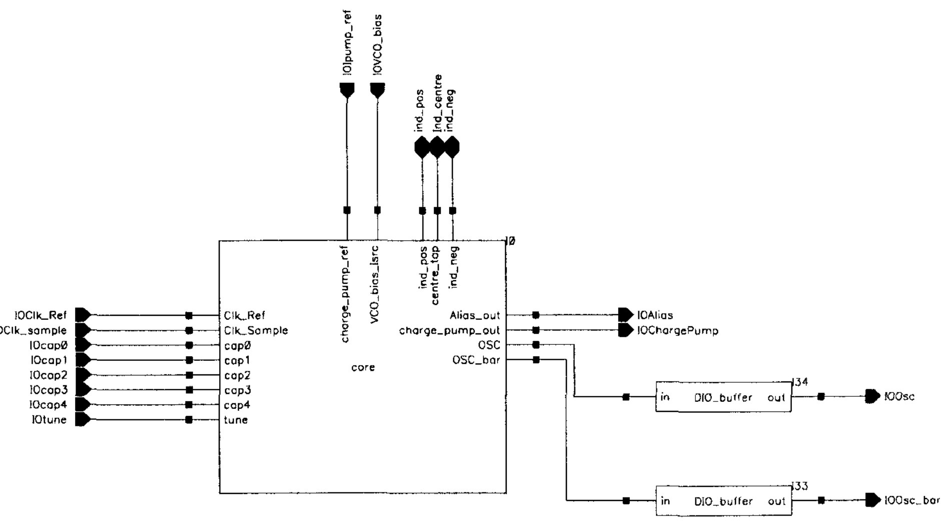

Figure C.19: core_all: Top level core circuit for 90-nm ALL circuit

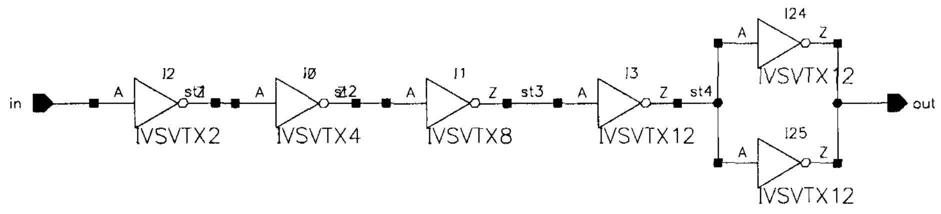

Figure C.20: DIO_buffer: Driver circuit for output pad

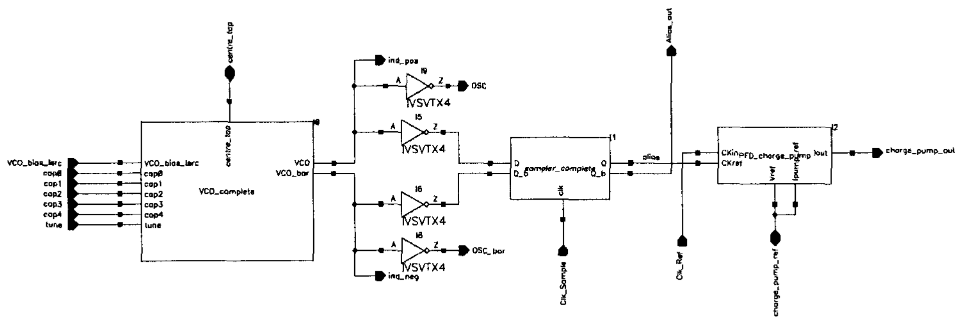

Figure C.21: core: Top level circuit without IO drivers

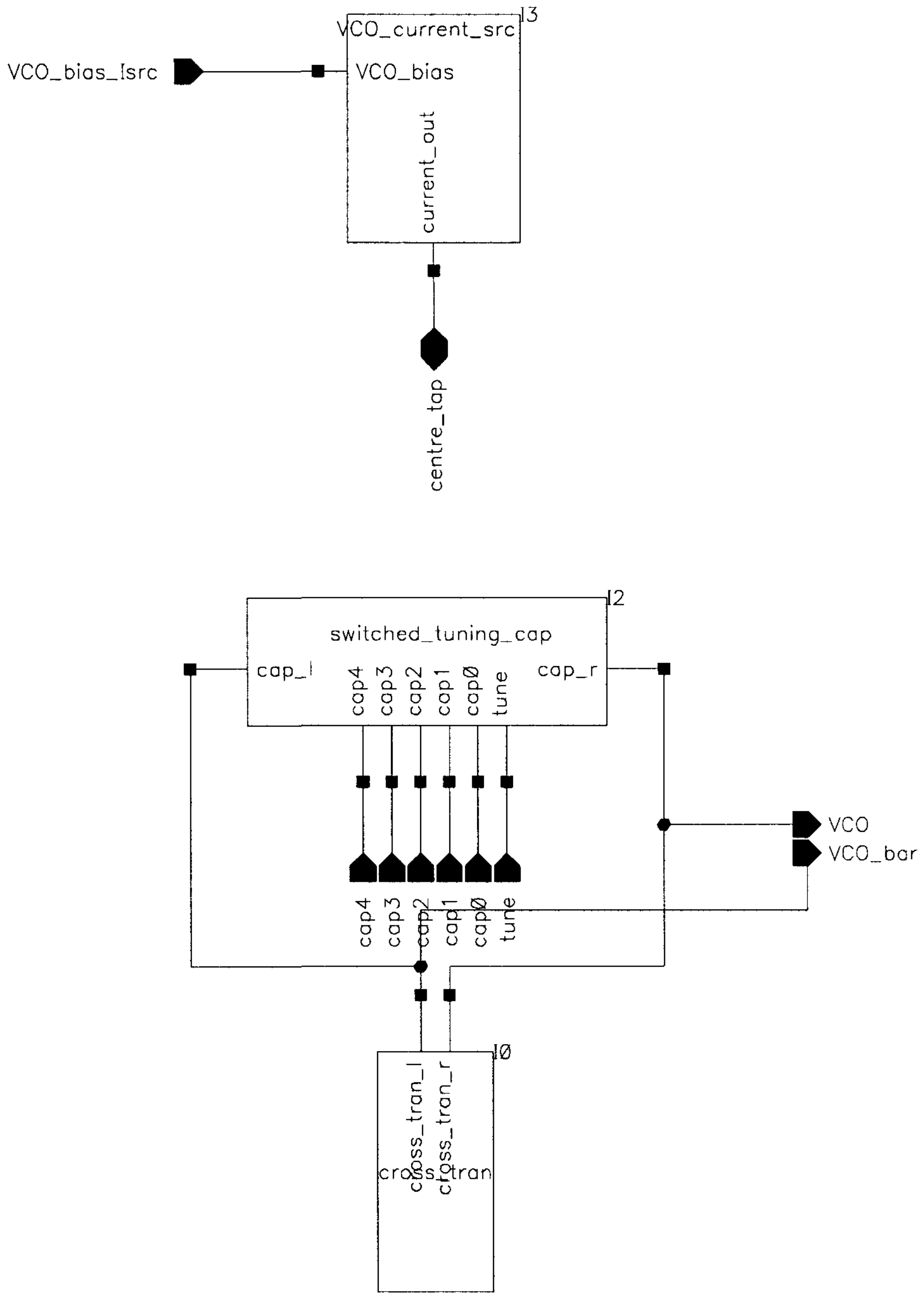

Figure C.22: VCO_complete: Differential VCO based on oscillating LC tank

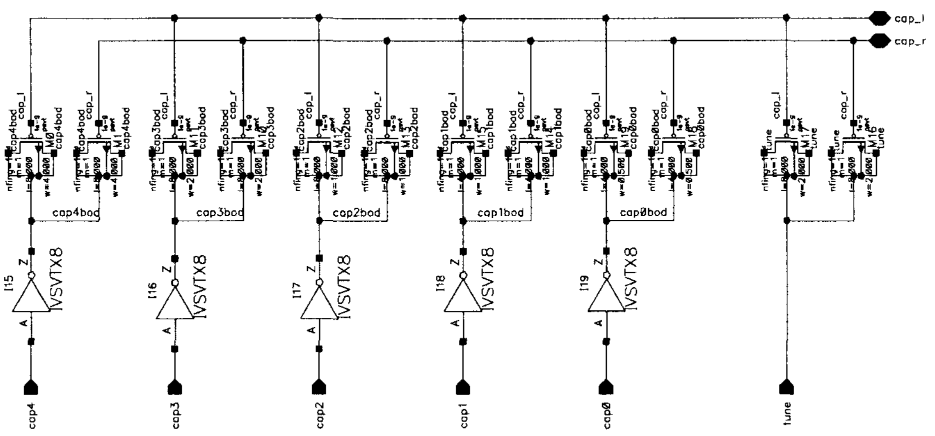

Figure C.23: switched_tuning_cap: Switched varactor bank and tuning varactor for VCO frequency control

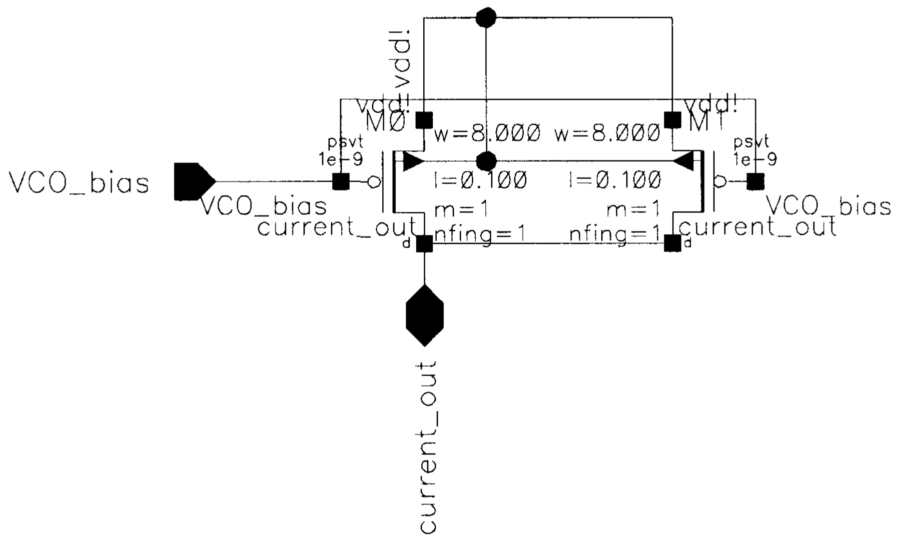

Figure C.24: VCO_current_src: PMOS current source for the VCO

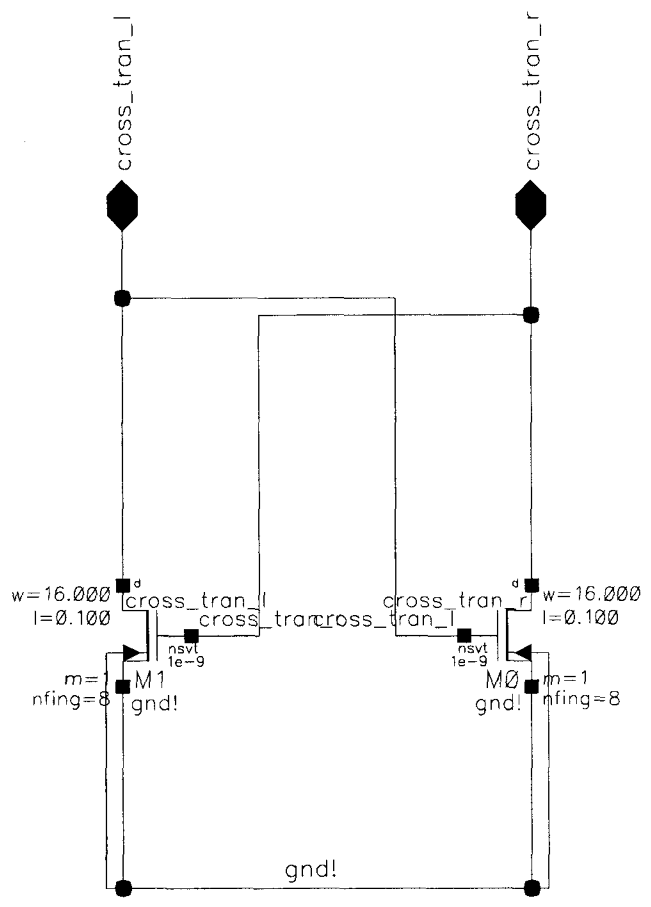

Figure C.25: cross_tran: Cross-coupled transistors driving the VCO

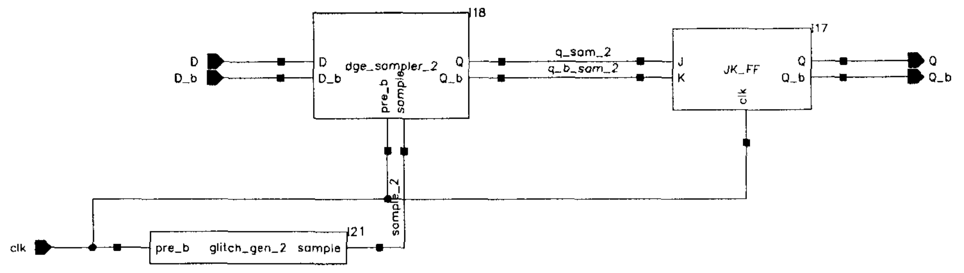

Figure C.26: sampler_complete: DPTPL based sampling latch

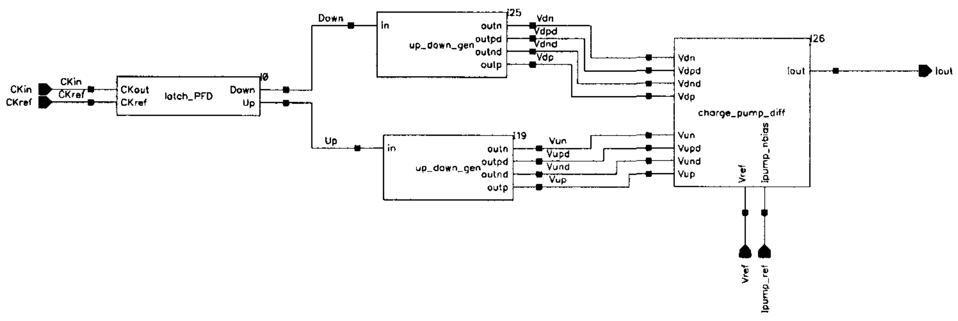

Figure C.27: PFD_charge_pump: Phase-frequency detector and charge pump

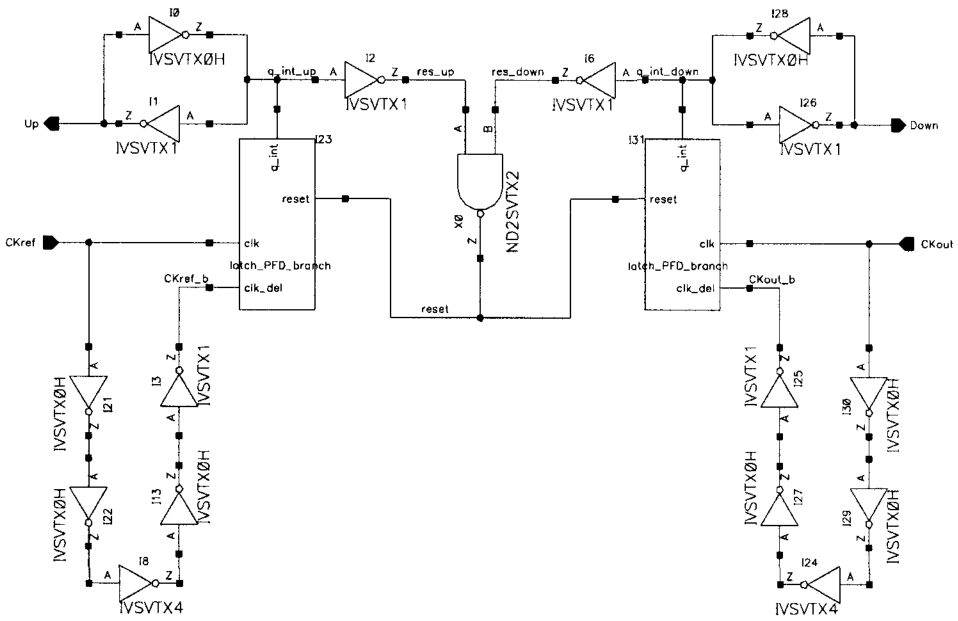

Figure C.28: latch_PFD: Latch-based phase-frequency detector

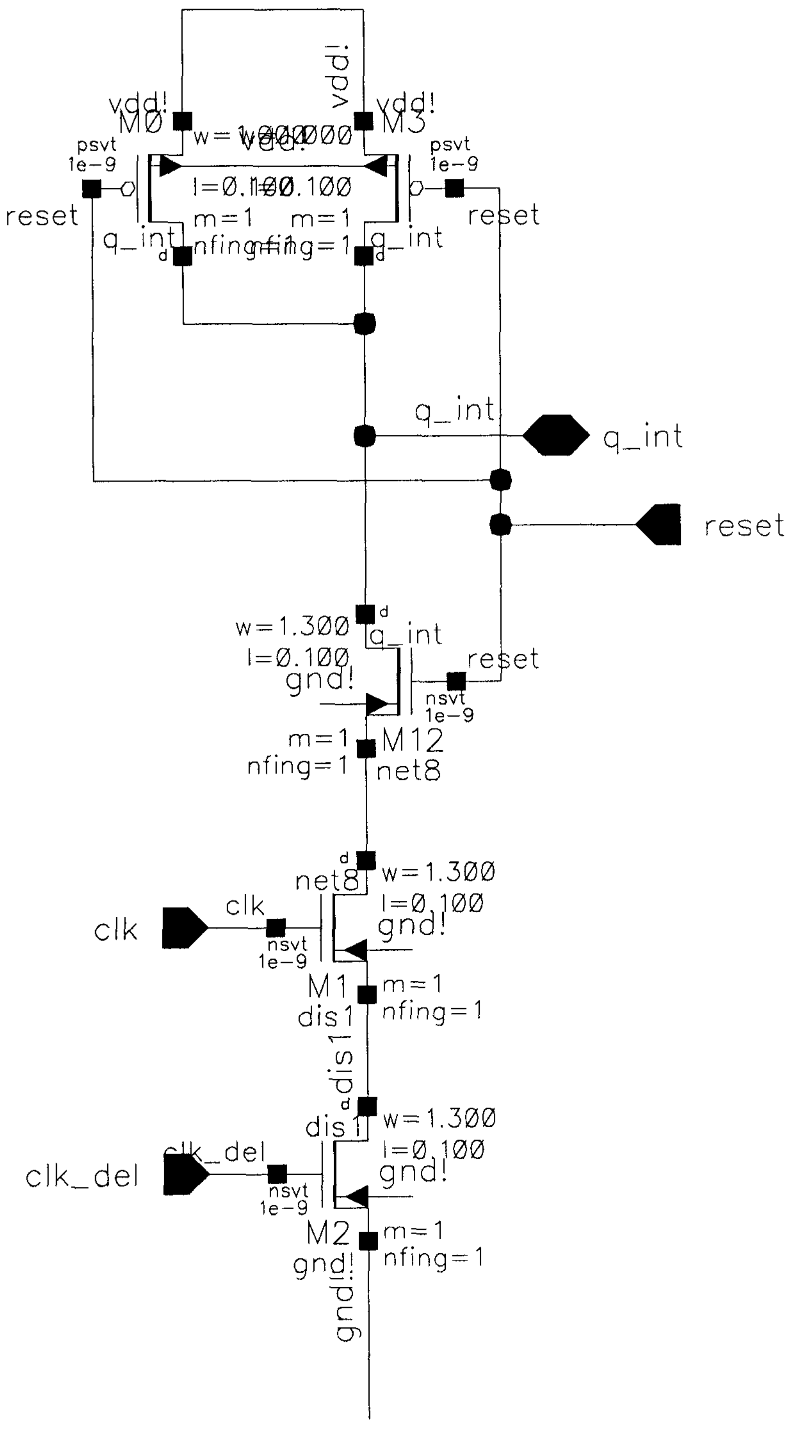

Figure C.29: latch_PFD_branch: Transistor branch inside the latch-based PFD

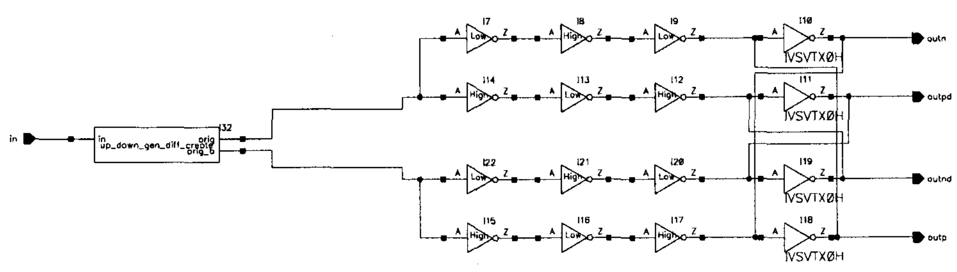
Figure C.30: up_down_gen: Differential control signal generator for charge pump

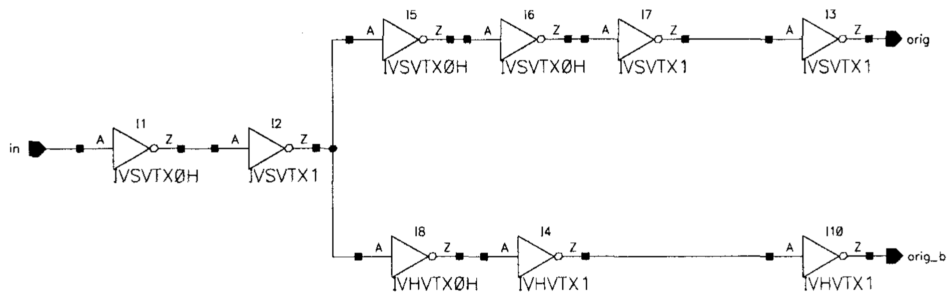
Figure C.31: up_down_gen_diff_create: Single-ended to differential signal generation for charge pump

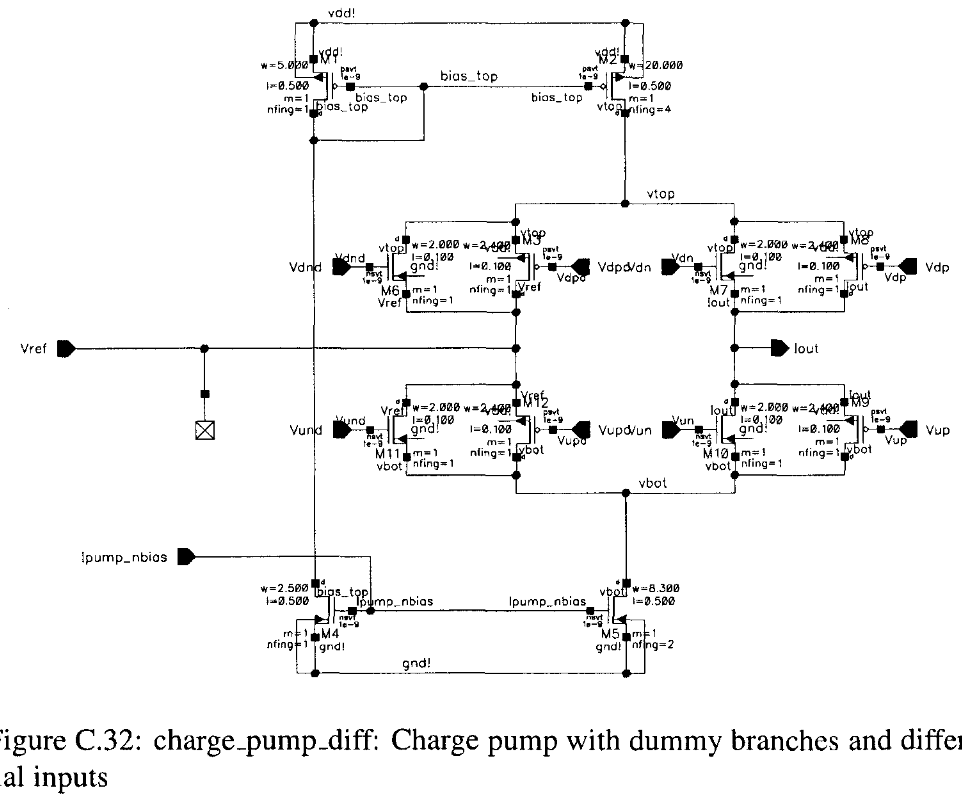
Figure C.32: charge_pump_diff: Charge pump with dummy branches and differential inputs