A Frequency Synthesizer for a 2.4 GHz Spread Spectrum Communications Application

by

Hanoz Gandhi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1997

Author .... Hanoz Gandhi ....
Department of Electrical Engineering and Computer Science
August 26, 1996

Certified by .... [illegible] ....
Anantha Chandrakasan
Assistant Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Theses

A Frequency Synthesizer for a 2.4 GHz Spread Spectrum Communications Application

by

Hanoz Gandhi

Submitted to the Department of Electrical Engineering and Computer Science on August 26, 1996, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering

Abstract

The thesis incorporates the formulation, design, construction, and simulation of a low power frequency synthesizer which operates in the ISM band between 2.402GHz to 2.485GHz. The synthesizer specifications are targeted to the IEEE 802.11 specification for a frequency hopping wireless Local Area Network (LAN)[1]. The focus of the work is placed on achieving fast frequency hopping using a phase locked loop (PLL) type synthesizer, to meet the requirements for accuracy, phase noise, and spurious signal noise. The conventional Fractional-N divider topology is compared with two novel topologies to overcome slow slewing and settling during frequency hops. Functional and circuit simulations are used to verify and characterize performance.

IBM Supervisor: Dr. B. Jeffrey Gross

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor

Acknowledgments

To my parents, who have footed the bill at about $800 per page, my love and thanks. To my brother, who has never failed to call me up and tell me what a good time he is having while I am working, and who has always made me laugh, my love and thanks. To my uncle, Rusi, my grandparents, Pesi and Nergis, my aunts, uncles, cousins and the rest of my extended family abroad, thanks for the moral support.

To Jeff, thanks for all the help - I have learned a great deal. Especially thanks for making the time even though your work load changed enormously. To Prof. Chandrakasan, thanks for taking me on as your student and making the thesis process a relatively simple and painless one.

Thanks to my dear friend Uzma, who provided bed and board while in Boston and who spent hours proof reading this document.

My sincerest thanks to all the people here in the Analog and Mixed Signal group. From the network support (Andrew), to the entertainment and humor (Ray). To Tony, Sue, Kay, Jim, John, Mo, Annette and all the others, thanks for your help.

And finally, a very special thank you to my machine for not crashing (today)!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Communication systems are fundamentally designed around a requirement to move data from one location to another. The data can be transferred in one of two ways: with wires or without wires. It is the latter technology which shall be discussed in this thesis with a focus on Radio Frequency (RF) systems.

In wireless communications today, the use of the electromagnetic spectrum is regulated by the Federal Communications Commission (FCC). Typically a license is needed to operate a wireless communications channel. There are three frequency bands currently available, however, for unlicensed operation. These bands and their respective bandwidths are: 900MHz (902–928MHz), 2.4GHz (2.402–2.483GHz), and 5.7GHz (5.725–5.85GHz). Within these ISM (Industrial, Scientific, and Medical) bands, the FCC requires the use of either direct sequence or frequency hopping spread spectrum communications protocol[1] with an output power of not more than 1Watt. While the availability of unlicensed spectrum is welcomed by the commercial applications market, the spread spectrum requirement adds new technical challenges for the RF designer; most notably with regard to carrier frequency generation.

The end application for the system described here is for a wireless Local Area Network (LAN) card. Currently LAN connections are made using a variety of wired mediums, e.g. coaxial cable, fiber optic cable, etc. While all these mediums can support communication bandwidths upwards of 10Mbits per second, they require a large wire based infrastructure. Such an infrastructure is not conducive for portable style applications and here is at least one need for the wireless communications paradigm. Today, wireless LAN cards, which employ spread spectrum communications technology, are manufactured using mainly discrete components. This translates directly into higher costs and power consumption. The aim is to integrate this design onto a chip to achieve lower costs, lower power, and better performance.

1.1 Background

Central to any RF communication systems is the frequency synthesizer. A simple model of an RF communications system is shown below. The transmitter modulates the communication signal onto a carrier frequency which is generated by the frequency synthesizer. This hybrid signal is acquired by the receiver which in turn demodulates the desired signal from the carrier. In order to recover this signal, the receiver needs to synchronize its own local oscillator with that of the transmitter. In this model, synchronization is achieved by tuning the local oscillator of the demodulator to the same frequency as the carrier and eliminating any phase error between the local oscillator and the transmitted carrier with a frequency synthesizer. In a frequency hopping spread spectrum communications system, where the carrier frequency no longer remains fixed, but instead changes in a pre-determined manner, the frequency synthesizer becomes an integral component.

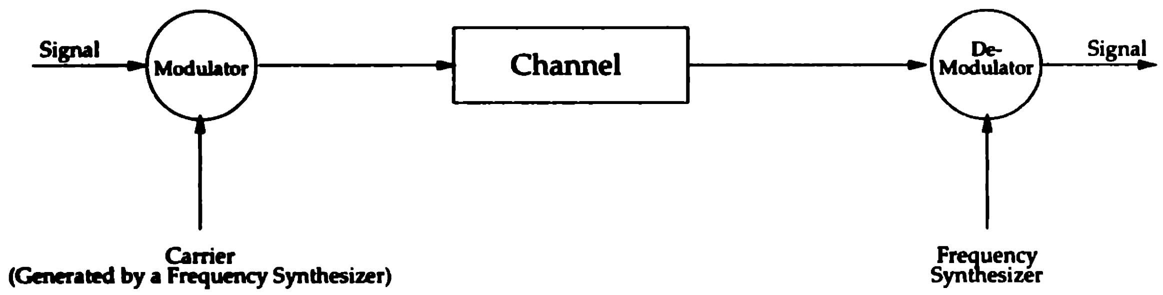

Figure 1-1: Simple model of an RF communications system.

Frequency synthesizers can be divided into two groups. The first group consists of 'direct' synthesizers. These synthesizers are distinguished by being open-loop in structure and generate an output frequency by multiplying a reference clock (see figure 1-2. As a result of its open-loop nature, a device of this type can have a high bandwidth, but tends not to be able to reject disturbances well. In addition, the multipliers needed in such a topology consume large amounts of power. The other group consists of 'indirect' synthesizers. Unlike direct synthesis, indirect synthesis is a closed-loop operation. The simplest kind of an indirect frequency synthesizer is the phase locked loop (PLL).

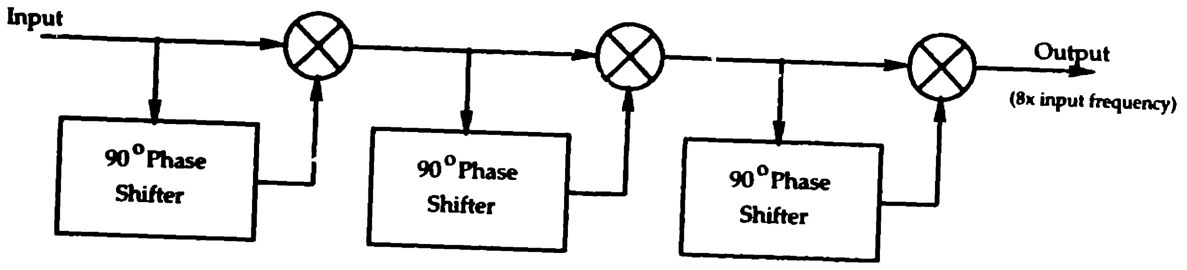

Figure 1-2: Simple model of an RF communications system.

The phase locked loop is a device which can be divided into three fundamental parts. These are: a phase discriminator, a low-pass filter, and a voltage controlled oscillator (VCO). Its operation is as follows: a reference signal $v_{\text{in}}(t)$ is provided at the input port to the synthesizer. The discriminator generates a voltage proportional to the difference between $v_{\text{in}}(t)$ and the output of the VCO, $v_{\text{vco}}(t)$. This output is then filtered to remove any high frequency components, after which it is passed into the VCO block (see Chapter 3 for a more detailed analysis).

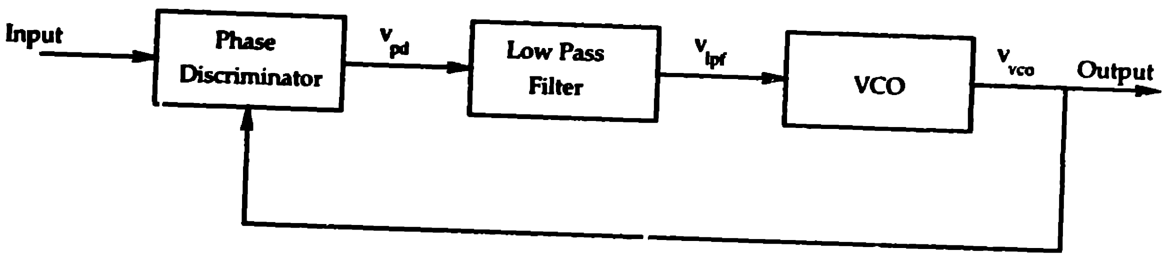

Figure 1-3: Block Diagram of a simple Phase Locked Loop.

Three major constraints that frequency synthesizers have to meet are: fast frequency settling time, low phase noise, and minimum power consumption. For the 2.4GHz ISM band, these are respectively: less than 300μseconds, $\pm$25ppm around the transmitter center frequency, and satisfy the need for portability[1]. Currently, frequency synthesizers that operate in the gigahertz portion of the spectrum are implemented in a number of different ways. One method is to perform Direct Digital Synthesis (DDS)[2]. Here a reference clock is multiplied N times to produce the desired output frequency. This is a 'direct' type synthesizer which tends to have a much faster frequency slewing time, but requires a large amount of power which makes it unattractive. Another type of frequency synthesizer known as a Fractional-N synthesizer, employs a variable divider ratio in the feedback path of the PLL[3]. This is an 'indirect' type synthesizer and is prone to excessive phase noise which results from having a divider that is not an exact divisor of the frequency at the output of the VCO. A third method to improve frequency slewing time employs a non-linear phase detector[4] within the Fractional-N type topology. While this method speeds up the closed loop response of the system, the non-linear element leaves some question as to the stabilit and a potential increase in the complexity of the system.

1.2 Proposal

The goal of this thesis is to formulate, design, and simulate a low power frequency synthesizer. The synthesizer will be capable of operating in the frequency range between 2.402GHz and 2.485GHz, with 1MHz steps, and will slew between frequencies within a few cycles of its reference clock. The synthesizer specifications will be targeted to the IEEE 802.11 specification for frequency hopping wireless LAN[1]. Given the past work done to develop such synthesizers, this thesis will generate a hybrid synthesizer which improves upon the technology introduced by others. This is accomplished by a combination of a Fractional-N synthesizer and some non-linear building blocks to speed-up hopping transitions.

Chapter 2

Synthesizer Topologies

In any radio frequency system, the central component is the frequency synthesizer. Both the transmitter and receiver must be able to select the same frequency, so that signal energy transmitted at a particular frequency by the former, can be picked up by the latter. To this end, there are three critical aspects of the synthesizer which characterize its performance.

- *Accuracy* of a synthesizer relates to how well it can select a particular frequency. The more precise it is, the more of its output energy it can focus at a given frequency.

- *Phase Noise* in a synthesizer determines how the energy put out by it is shaped at a particular frequency. The sharper and narrower the energy diagram, the less noise is introduced by the synthesizer to corrupt the transmitted signal.

- *Spurious Sideband* noise is the appearance of energy at other frequencies of the spectrum that are often image frequencies. This is usually a result of using a mixer or like type device in the process of synthesizing the frequency. Again the fewer the sidebands, the less noise is introduced by the synthesizer to corrupt the transmitted signal.

2.1 Types of Synthesizers

All frequency synthesizers require a reference clock. The simplest high accuracy reference clock is a crystal oscillator. An AC voltage applied to crystalline material causes it to vibrate at a fixed natural frequency. This output is subsequently buffered and amplified for use. All frequency synthesizers operate by referring their output frequency to a high accuracy reference clock. With this as the basis, frequency synthesizers fall into three categories.

2.1.1 Direct – Analog

Direct Analog synthesis is formed by using an array of crystal oscillators and changing frequency by simply using a switch to index into this array. Frequencies that are not uniquely contained within the array can be created. Given an array which has an adequate basis set of crystals, crystal outputs can be added, multiplied and filtered to produce a new frequency.

This type of system requires large amounts of physical space to create an adequate array of basis oscillators from which to created the requisite frequencies; the degenerate case being one oscillator for each frequency required. The switching speed of such a system is primarily limited by how quickly the switch can change its current state, therefore, for the sake of comparison it can be considered to have a 'fast' switching speed. Since crystals tend to oscillate at a unique frequency, they have very pure spectral content which generates minimal phase noise as a result. Direct

Analog systems however, suffer from large amounts of spurious phase noise. This noise manifests itself from image frequencies as well as voltage noise from the control frequency resulting in harmonics which can be substantial in magnitude.

The consideration from a consumer driven market standpoint, makes such a topology impractical for two major reasons – monetary cost and power consumption. Large crystal arrays would not only occupy valuable space, but would also increase the cost substantially. Similarly, power consumption for such an array would also be high, given the number of oscillators which would be required. These concerns have a particular emphasis, more so now with the growth of the portable consumer market – a market which is constantly demanding leaner, cheaper, and more compact initiatives.

2.1.2 Direct – Digital

Direct Digital synthesis operates from a platform similar to its analog counterpart, but its array takes a different form. In direct digital synthesis a ROM or RAM look-up table contains the data words describing one period of a sinusoidal waveform. The output of this table is passed through a Digital to Analog Converter (DAC), which decodes the data words into corresponding voltages, and then through a low-pass filter to generate a smoothly varying sinusoidal waveform. The input to the look-up table is an index whose increment step size varies depending upon the frequency desired at the output. Finally the entire system is synchronized using a clock (see figure 2-1. This reference clock is a digital switching waveform which can be generated by a simple circuit; for example a ring oscillator. Other than the requisite output frequency, there are no stringent performance specifications which it must meet.

As a synthesizer, the Direct Digital solution has exceptional command over the three critical parameters that characterize it. The accuracy of this topology can be excellent since a system of this type is immune to much of the interference that couples into an analog system. Phase noise for such a system is also minimized as it is determined by jitter in the system clock frequency and the resolution of the Digital to Analog Converter; this can be also be removed by the low-pass filter. Similarly, spurious sideband noise is also at a minimum since no mixing process is involved.

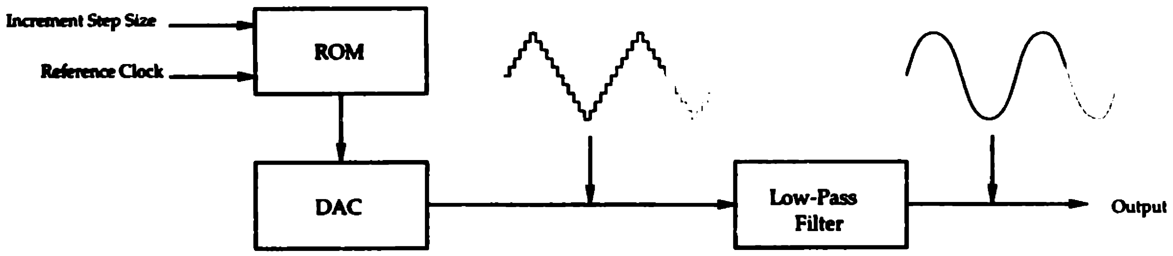

Figure 2-1: Direct Digital Synthesizer.

Changing from one frequency to another can also be accomplished quickly while maintaining continuous phase.

Perhaps the biggest outright disadvantage to such a topology is the need to have a reference clock that is at least two times the maximum frequency generated by the synthesizer. The reference clock in this case is a digital switching waveform, which aside from the frequency requirement, does not have constraints on phase noise or accuracy. This requirement comes from the Nyquist Sampling theorem which states that a waveform must be sampled at a frequency which is at least twice the greatest frequency that comprises the waveform. Not only is this a difficulty in itself for high frequency synthesis, it also makes this topology a very power hungry solution for frequency synthesis.

Synthesizers of this topology have been suggested with even higher frequency switching times [5], however they are geared to military applications where performance is the primary concern. Another means of using such a topology would involve using it at a lower frequency (IF frequency) and then mixing the output with a fixed frequency oscillator at a higher frequency. This would reduce the need for a very high frequency reference oscillator while simultaneously providing fast switching and high accuracy. The cost however, would be an increase in phase noise and spurious sideband noise as a result of the mixed-in high frequency oscillator and the mixer respectively.

2.1.3 Indirect

Indirect synthesizers are closed loop in nature. A typical topology is a phase locked loop. It is described in detail within the next chapter but a brief summary of its operation follows. The loop takes in a reference frequency and compares it with the output of a voltage controlled oscillator. The error signal between the two is filtered and passed to the VCO where it is used to shift the phase and frequency at the output of the oscillator. If the oscillator is operating at a frequency different from that of the reference, a divider is inserted in the path between the oscillator and the point where the comparison is performed.

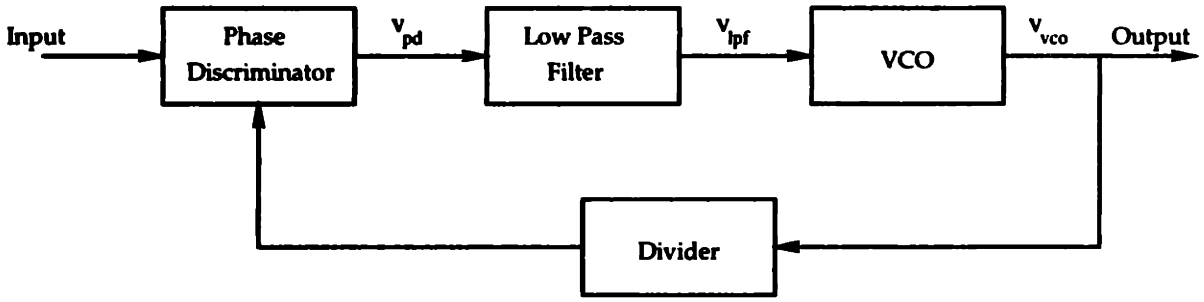

Figure 2-2: Frequency Synthesizer.

The accuracy of such a system is limited by the voltage controlled oscillator. Phase noise is a particular problem for such a system as a result of noise which lies within the pass band of the filter and the noise injected into the loop after the filter [6]. Furthermore, the time required to settle is drastically increased from the other topologies discussed, due to the need for a low-pass filter whose cut-off frequency is well below the operating frequency of oscillator.

It is this topology, however, which permeates 99% of all synthesizer topologies [7], because it is simple, dissipates the least power, and can utilize a reference frequency well below the output frequency. These luxuries come at the price outlined above and also include the need for complex logic to achieve frequency synthesis [6].

2.2 Indirect Topology Variations

The biggest problem with the indirect topology is the latency involved in changing frequency. This latency is usually in the neighborhood of 2$m$seconds and can be much larger [8]. As a consequence of this latency, 'fast' slewing between frequencies is not trivial.

A number of solutions have been suggested to improve the settling time of the loop. Perhaps the most direct one is to simply increase the cut-off frequency of the low-pass filter. This solution and its consequences are discussed in the next chapter.

Another solution on a similar vein involves the use of multiple voltage controlled oscillators [8]. This method while affording better results comes at the cost of increased hardware, noise, power consumption, and circuitry.

Replacing the divider with a direct frequency measurement block is another solution [9], however while this may be feasible at low frequencies it cannot be done reasonably at higher frequencies.

Another method proposes the use of digital signal processing techniques to calculate the phase difference without introducing strong spurs or harmonics thereby removing the need for the low-pass filter [10].

Yet another solution dynamically alters the loop parameters (i.e. low-pass filter cut-off frequency) thereby contracting and expanding loop bandwidth [11]. (The effect of loop bandwidth on the settling time of a synthesizer is discussed in detail in Chapters 2 and 3.)

All of these solutions are aimed at the problem of slew and settling time latency, but they do not approach the problem from its cause. While it is true the low-pass filter does limit how fast the loop settles, the more important questions are: Why is an error signal generated and what can be done to eliminate it, or minimize it during frequency hop? The answer is: the error signal is clearly due to the divider in the circuit. The divider state is often not maintained between changes; thereby introducing large errors. Even if the state is maintained partially, the resulting error signal is not continuous in nature and therefore the periods where there is feedback are limited by the phase detector topology.

Looking at frequency synthesis from this point of view, a new solution is proposed that will aim to nip the problem of large settling time between frequency hops before it manifests. However, before presenting the solution a review of synthesizer operation follows.

Chapter 3

Fundamental Frequency Synthesizer Operation

In order to gain a detailed understanding of the operation of a frequency synthesizer, let us first take a look at how a phase locked loop operates.

3.1 Phase Locked Loop

A phase locked loop has three essential pieces. These are: phase discriminator, low-pass filter, and a voltage controlled oscillator. The topology of the loop is pictured below:

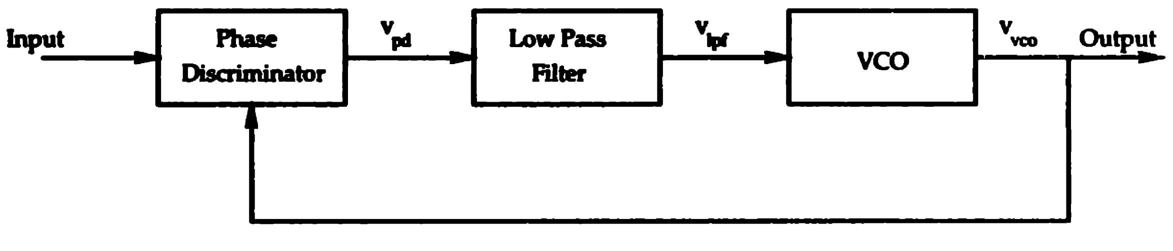

Figure 3-1: Phase Lock Loop topology.

3.1.1 Phase Discriminator

The phase discriminator compares the frequency of the input signal with the frequency of the output signal from the phase-locked loop. The discriminator generates an output voltage that is related to the phase difference between the two.

$$v_{pd}(t) = f(v_{in}(t), v_{vco}(t)) \tag{3.1}$$

The function f can be almost any function of the input signals, however most often it is simply a constant multiplying the voltages – that is because for small phase ϕ

$$cos(\omega t + \phi)\ cos\ \omega t \ = \ \frac{cos\phi}{2} + \frac{cos(2\omega t + \phi)}{2}, \tag{3.2}$$

which, when low-pass filtered becomes

$$cos(\omega t + \phi)\ cos\ \omega t \ \approx \ \frac{\phi}{2} \tag{3.3}$$

3.1.2 Low Pass Filter

The output voltage of the phase discriminator is then passed through a low-pass filter, whose pass band lies well below the output frequency of the voltage controlled oscillator. The purpose of this filter is to attenuate high frequency components of the output from the phase discriminator.

$$v_{lpf}(t) = k_{lpf} \int_0^t e^{\frac{\tau - t}{k_{lpf}}} v_{pd}(\tau) d\tau \tag{3.4}$$

Essentially the low-pass filter provides a fixed gain for all frequencies below the ω_{cf} and above this frequency acts as an integrator. The frequency domain representation of a low-pass filter is:

$$H_{lpf}(s) = \frac{k_{lpf}\omega_{cf}}{s + \omega_{cf}} \tag{3.5}$$

3.1.3 Voltage Controlled Oscillator

The voltage controlled oscillator (VCO) is a type of voltage to frequency converter. It takes at its input a voltage and generates an output waveform at a frequency proportional to it. This is expressed as

$$v_{vco}(t) = sin(v_{lpf}(t)\alpha t), \tag{3.6}$$

where α is a proportionality constant. Most voltage controlled oscillators have a free running frequency. That is with a 0V input voltage, the VCO generates an output sinusoid at a frequency ω_{f_r}. This is of importance, as most VCO do not have linear relationship between input voltage and output frequency. Equation 3.6 can be rewritten as

$$v_{vco}(t) = sin(\omega_{f_r}t + v_{lpf}(t)\alpha t) \tag{3.7}$$

This equation does not however accurately describe the behavior of the VCO with respect to phase. According to equation 3.7, if v_{lpf} is 0V for all time except some interval Δt, then

$$v_{vco}(t < \Delta t) = sin(\omega_{f_r}t) \tag{3.8}$$

and

$$v_{vco}(t > \Delta t) = sin(\omega_{fr}t), \tag{3.9}$$

where $t < \Delta t$ and $t > \Delta t$ represent time before and after the interval Δt, over which v_{lpf} is non-zero. In fact, what actually occurs when v_{lpf} is non-zero over the interval Δt is that the output at v_{vco} shifts its phase by ψ radians where

$$\psi = k_{vco} \int_0^{\infty} v_{lpf}(t)dt \tag{3.10}$$

The phase shift depends on both the duration of the interval Δt and the v_{lpf} over that interval. The correct equation to describe the behavior of the VCO is therefore

$$\begin{aligned} v_{vco}(t) &= \psi(t) & (3.11) \\ &= sin[\omega_{fr}t + k_{vco} \int_0^t v_{lpf}(t)dt] & (3.12) \end{aligned}$$

and the corresponding output waveforms from the VCO given a v_{lpf} described previously would be

$$v_{vco}(t < \Delta t) = sin(\omega_{fr}t) \tag{3.13}$$

and

$$v_{vco}(t > \Delta t) = sin(\omega_{fr}t + \psi) \tag{3.14}$$

The input to the VCO can be classified into four categories: constant zero, constant non-zero, linear, non-linear. Here is a brief summary describing the output of the VCO to these four groups of inputs. In the third column, is the frequency of the output sinusoid.

Input (v_{lpf})	Output (v_{vco})	Frequency (rad)
Constant Zero	$sin[\omega_{fr}t]$	ω_{fr}
Constant Non-Zero	$sin[\omega_{fr}t + k_{vco}v_{lpf}t]$	$\omega_{fr} + k_{vco}v_{lpf}$
Linear	$sin[\omega_{fr}t + \frac{k_{vco}}{2}v_{lpf}t^2]$	$\omega_{fr} + k_{vco}v_{lpf}t$
Non-Linear	$sin[\omega_{fr}t + k_{vco}\int_0^t v_{lpf}(t)dt]$	N/A

Notice that a constant input voltage results in a shift in the output frequency, whereas a linear input voltage results in a frequency which changes in time!

Now with an understanding of each of the building blocks, the whole loop can be analyzed.

3.1.4 Loop Dynamics

In order to use our frequency domain analysis tool box, let us label each box in the loop with its frequency domain representation. We will continue to assume that the phase discriminator is a linear block with a gain k_{pd}, the low-pass filter is as specified in equation 3.5, and the VCO will be treated as an integrator of phase whose transfer function is:

$$H_{vco}(s) = \frac{k_{vco}}{s} \tag{3.15}$$

The VCO is modeled as an integrator, because in the argument of the sin function for the VCO, (refer to equation 3.12), the control voltage is integrated. The instantaneous sum of integrated control voltage and the free running frequency is the instantaneous output frequency of the VCO.

The complete forward loop transfer function is

$$G_f(s) = \frac{k_{pd} k_{lpf} \omega_{cf} k_{vco}}{s^2 + \omega_{cf} s} \tag{3.16}$$

The root locus analysis of $G_f(s)$ diagram shows that for increasing positive values of gain the closed loop system will move towards instability as the natural frequency of the closed loop system will increase while the damping ratio of the system will decrease.

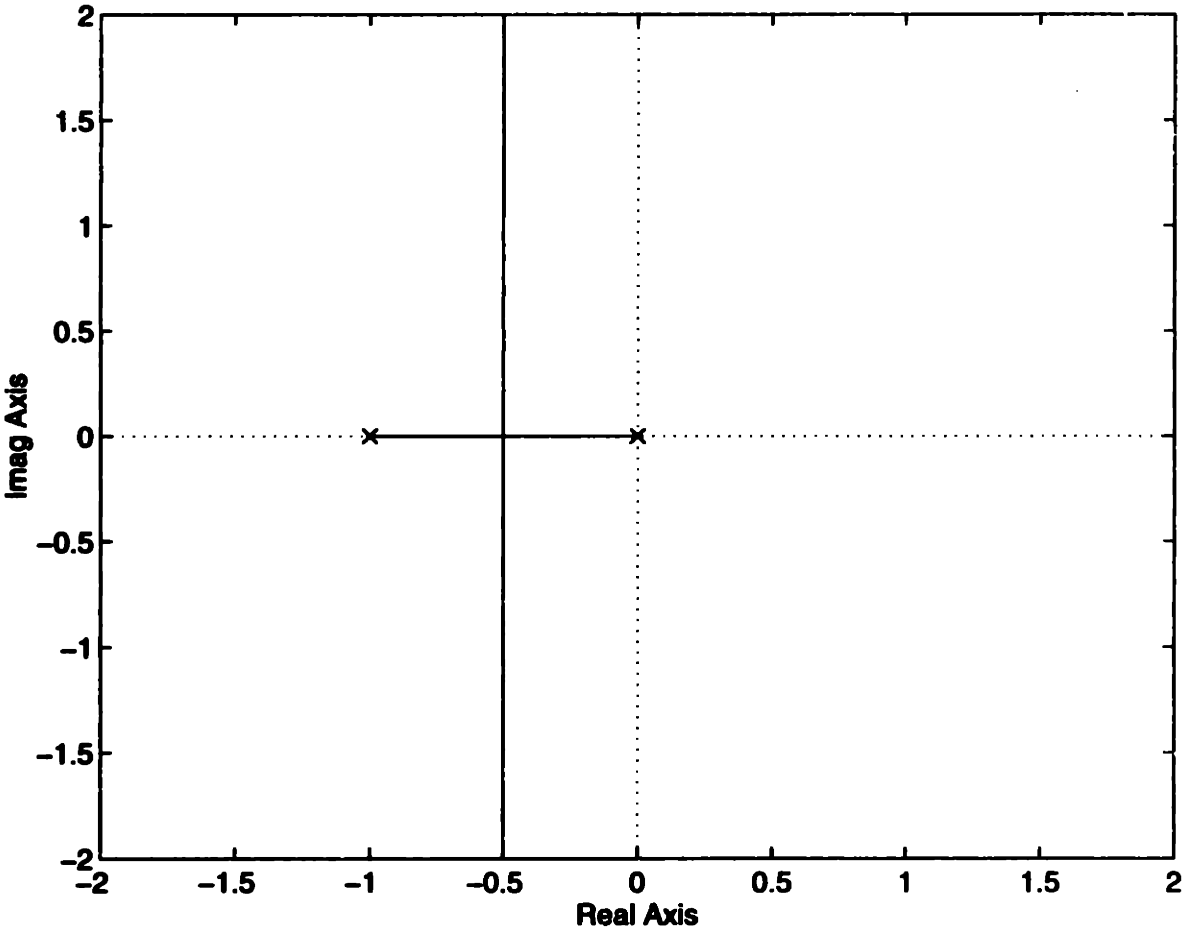

Figure 3-2: Root Locus Diagram of $G_f(s)$.

Equivalently, we can obtain a closed loop transfer function:

$$H_{pll}(s) = \frac{G_f(s)}{1+G_f(s)} \tag{3.17}$$

$$= \frac{k_{pd}k_{lpf}k_{vco}w_{cf}}{s^2+\omega_{cf}s+k_{pd}k_{lpf}k_{vco}\omega_{cf}} \tag{3.18}$$

$$= \frac{w_n^2}{s^2+2\zeta w_n s+w_n^2} \tag{3.19}$$

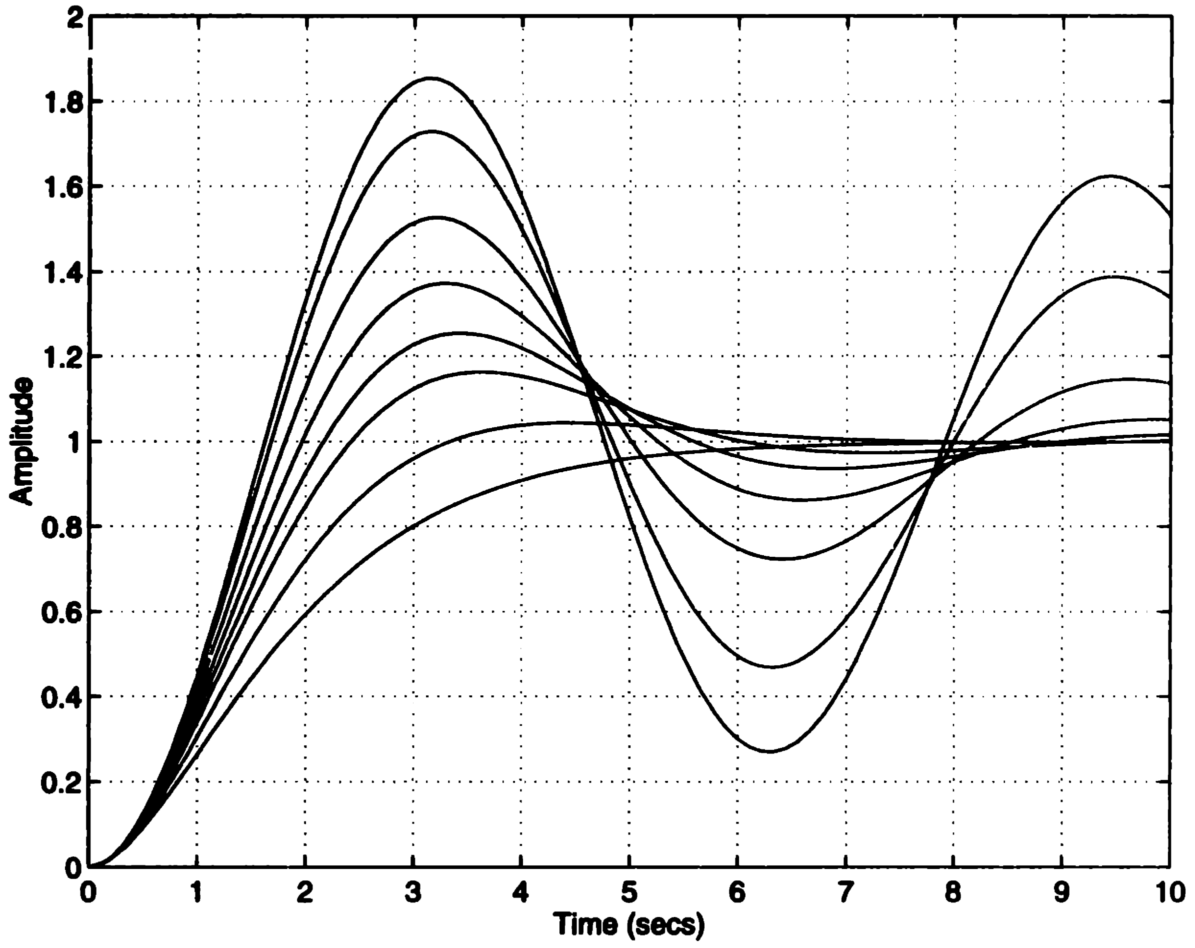

Figure 3-3: Step response of a second order system for varying ζ over the range [.05 .1 .2 .3 .4 .5 .707 1].

From this standard second-order system form, we can obtain both the damping ratio ζ and the natural frequency of the loop ω_n. Figure 3-3 affords a pictorial view of these two quantities. Here, a second order system is provided with a step input and ζ is varied while ω_n is kept constant. Notice, the smaller ζ is, the more lightly damped the system.

$$\begin{aligned} w_n &= \sqrt{k_{pd}k_{lpf}k_{vco}\omega_{cf}} \\ \zeta &= \frac{\omega_{cf}}{2\omega_n} \\ &= \frac{\omega_{cf}}{2\sqrt{k_{pd}k_{lpf}k_{vco}\omega_{cf}}} \end{aligned} \tag{3.20}$$

These two pieces of information characterize the performance of the overall closed loop system. They tell us how quickly the loop will respond to disturbances and how stable the loop is. Increasing the gain (k_{pd}, k_{lpf}, k_{vco}) increases ω_n and decreases ζ. Increasing ω_{cf}, increases ω_n and ζ by the same factor.

Before moving forward and extending this discussion one step further to frequency synthesizers in general, a little more time spent in analyzing this model will reap some practical intuition.

3.1.5 Phase Domain Analysis

The closed loop transfer function $H_{pll}(s)$ derived above for the phase lock loop represents the transfer function between the *input phase* and the *output phase.* What are these *phases*?

If you examine the following expression:

$$v_o = sin(\theta t) \tag{3.21}$$

$$= sin(\rho t + \theta_o) \tag{3.22}$$

ρ is the angular frequency expressed in rad/sec and t represents the variable that is changing in this case, time. As time increases in a linear fashion, the argument θ increases in a similar fashion and its slope is determined by ρ. When referring to

input phase **and** ***output phase*****, what is being referred to are the arguments of the trigonometric functions, θ. We shall see that the analysis of the loop becomes much clearer when discussed in this fashion.**

Going back, $H_{pll}(s)$ is the transfer function between the input and output phases of the phase locked loop.

$$H_{pll}(s) = \frac{\Theta_{out}(s)}{\Theta_{in}(s)} \tag{3.23}$$

In the steady state case, the input and output frequencies are the same so both the time functions, $\theta_{in}(t)$ and $\theta_{out}(t)$, have the same slope.

Let us command a step change in the phase of the input $\theta_{in}(t)$ and examine how $\theta_{vco}(t)$ responds to this step change. Here are the corresponding waveforms for an over-damped system.

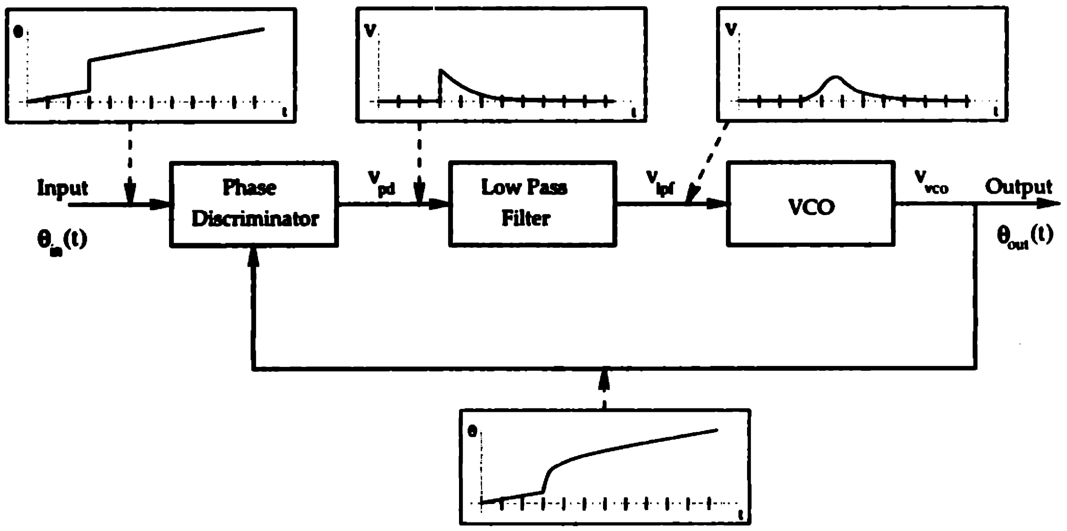

Figure 3-4: PLL system response to a step change in *phase*.

Notice that $\theta_{in}(t)$ and $\theta_{out}(t)$ are a step and the response to a step, respectively, superimposed on a ramp. Therefore, we can reduce our analysis to the step response (in phase) of our second order closed loop system.

$$t_{rise} = \frac{2.2}{\rho_{cf}} \tag{3.24}$$

t_{rise} is the 10% to 90% rise time of the system and it gives a good measure of of the bandwidth of a system. Examining the the step response for a sample well damped PLL in figure 3-5, the 10% to 90% rise time is approximately .2mseconds, which results in a loop bandwidth of about 69KHz. For this example, k_{pd} and k_lpf were both unity gain blocks, the filter cut-off frequency ω_{cf} was 1x10^6, and the VCO gain was 1x10^5.

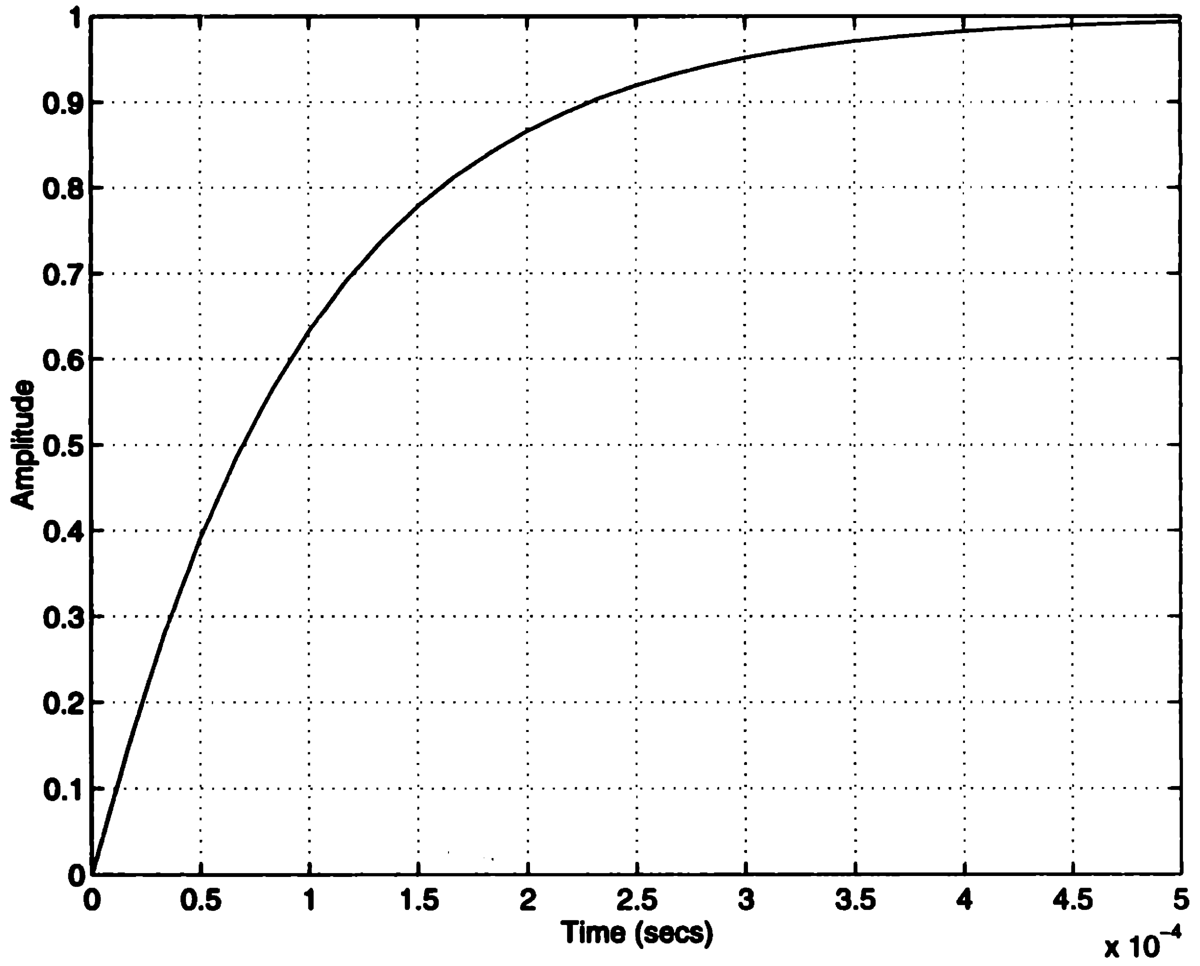

Figure 3-5: Sample step response of a typical PLL.

Since the step response of the system reaches and remains within 10% of this final value in .2mseconds, the settling time of this system is commensurate with the rise time.

Moving onto a frequency synthesizer, all the analysis performed previously is performed again with one minor change – the addition of a divider block in the feedback path.

3.2 Frequency Synthesizer

A frequency synthesizer is a more general class of devices, of which the phase locked loop is a subset. While there are many types of synthesizer topologies (refer to Chapter 2) the focus for this research was limited to one particular genre.

A frequency synthesizer contains all the parts of a phase locked loop and in addition has a *divider* in the feedback path – from the output of the VCO to the input of the phase discriminator (refer to figure 2-2). Since each of the other pieces of a synthesizer is the same as those for the phase locked loop, there is little need to repeat the description here. The only portion left undiscussed is the divider block.

3.2.1 Divider

The divider block does exactly what its name says, it divides the frequency of the output waveform from the VCO to a lower frequency; mainly that of the input received frequency. Since the output of the divider is a waveform whose frequency is equivalent to that of the received frequency and whose phase corresponds to the output of the VCO, the divider in the phase domain can be simply enumerated as a gain block,

$$H_{div}(s) = k_{div} \tag{3.25}$$

A simple example of a divider is a counter. Assume that the input to the divider is a signal oscillating at 32MHz and the desired output frequency is a 2MHz signal. The divider would simply need to divide the input frequency by sixteen. This could be easily accomplished using a 4-bit counter where the most significant bit (MSB) of the counter would be used as the output of the divider.

3.2.2 Synthesizer Operation

Static Analysis

Illustrated below are the waveforms associated with the reference clock v_{in} and the output of the VCO v_{vco}. The reference clock is operating at a frequency of 2MHz and the output of the VCO is operating at the desired frequency of 32MHz. In the feedback path there is a divide-by-16 divider. This implies that the divider will be dividing the output of the VCO by 16 to generate a 2MHz square wave at its output. Notice that v_{vco} goes through 16 cycles in the time v_{in} completes one.

Let us assume, again for simplicity, that the output of the divider is in phase with the reference clock when the VCO is operating at 32MHz. (Note: this is not a matter of concern as we are assuming that this is the free running frequency of our VCO, w_{fr}.) With the output of the divider in phase with the reference clock, the corresponding output from the phase discriminator will be 0V. In steady state therefore, the output of the low-pass filter will be 0V as well.

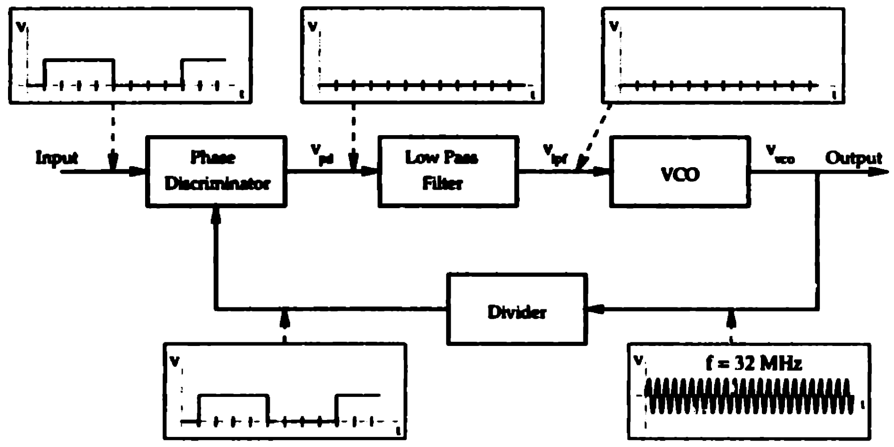

Figure 3-6: Frequency synthesizer operating in *steady state* with an input reference clock of 2MHz and an output frequency of 32MHz.

Depicted in figure 3-6 are the corresponding output waveforms for each element of the synthesizer. Notice the reference signal is a square wave, while the output of the synthesizer is a sinusoid. This does not pose any problem as shall be shown in the next chapter.

Dynamic Analysis

Now let us examine what happens in the dynamic case. Let us take the case where the desired output frequency changes from 32MHz to 30MHz. In order to produce this, the divider will have to divide the output frequency by 15 to generate a 2MHz output waveform. This is accomplished by changing the divide ratio of the divider to 15 from 16.

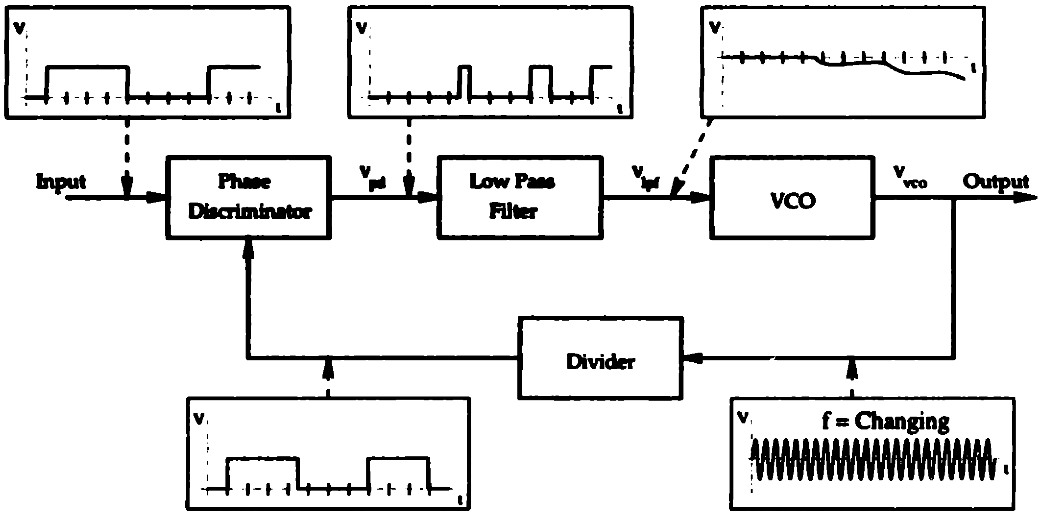

Figure 3-7: Frequency synthesizer operating in the *transition period* from an output frequency of 32MHz to 30MHz.

The result is that the divider output frequency increases which in-turn generates an error signal at the output of the phase discriminator. This error signal is filtered by the low-pass filter and this result is passed to the VCO. Notice that the filter cut-off frequency, ω_{cf} is much lower than the frequency of the reference input. This is necessary to attenuate the high frequency components of the output of the discriminator before passing to the VCO. Should the high frequency components not be filtered out, the VCO would attempt to track this error signal which could corrupt the VCO output and drive it into a region of instability.

Assuming that this closed loop system has the same well damped characteristics exhibited by the PLL in the previous section, as the output frequency of the synthesizer approaches 30MHz the divider output frequency would decrease approaching 2MHz. However, unlike the PLL a finite phase difference would remain between the reference input and the output of the divider. This phase difference would generate a constant signal into the VCO which would maintain its output frequency at 30MHz.

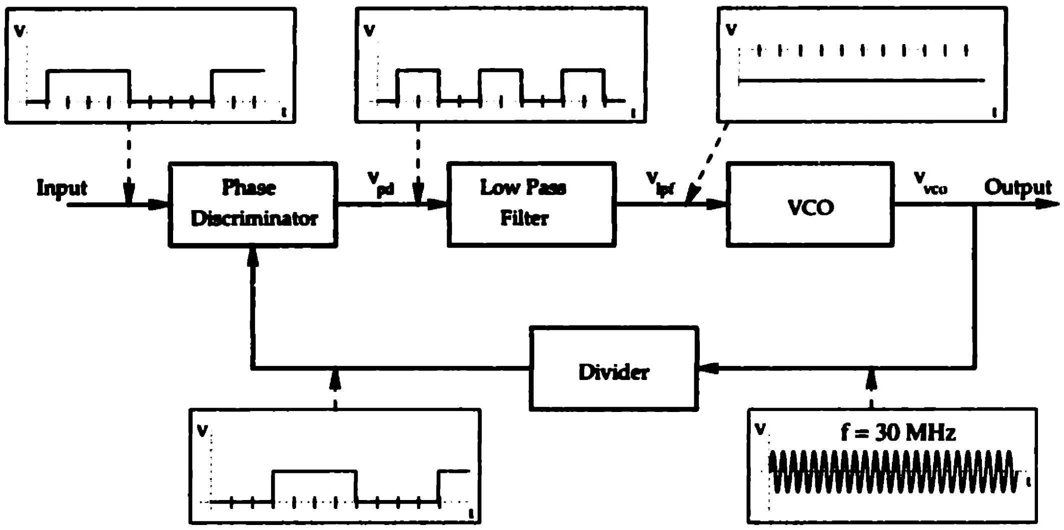

Figure 3-8: Frequency synthesizer operating in *steady state* with an input reference clock of 2MHz and an output frequency of 30MHz.

To characterize the dynamic behavior of the frequency synthesizer, it is necessary to return to frequency domain analysis. Like the PLL, the frequency synthesizer has the same forward loop transfer function

$$G_f(s) = \frac{k_{pd}k_{lpf}\omega_{cf}k_{vco}}{s^2 + \omega_{cf}s} \tag{3.26}$$

The closed loop transfer function however is

$$H_{fs}(s) = \frac{G_f(s)}{1 + G_f(s)k_{div}} \tag{3.27}$$

$$= \frac{k_{pd}k_{lpf}k_{vco}w_{cf}}{s^2 + \omega_{cf}s + k_{pd}k_{lpf}k_{vco}\omega_{cf}k_{div}} \tag{3.28}$$

Placing this equation into the standard second order equation format we calculate ω_n and ζ for the synthesizer.

$$\begin{aligned} w_n &= \sqrt{k_{pd}k_{lpf}k_{vco}\omega_{cf}k_{div}} \\ \zeta &= \frac{\omega_{cf}}{2\sqrt{k_{pd}k_{lpf}k_{vco}\omega_{cf}k_{div}}} \end{aligned} \tag{3.29}$$

Two things are immediately apparent when comparing these expressions for w_n and ζ with those calculated for the PLL. Since k_{div} is less than 1 for systems where the input frequency is smaller than the output frequency, ω_n decreases and ζ increases, for the frequency synthesizer compared with the PLL. Thus with the same system parameters, a frequency synthesizer has less bandwidth and more damping than a PLL. This can be seen directly by comparing the equations in 3.29 with those in 3.20, which were derived earlier for the PLL. This is also demonstrated by comparing the step responses of the two systems in figures 3-5 for the PLL and 3-9 for the synthesizer.

Calculating the 10% to 90% as before we see that it is now .5msec. Using equation 3.24, the bandwidth of the system is approximately 27KHz. This bandwidth however no longer represents just the settling time of the output. In the case of a frequency synthesizer, this bandwidth represents the slewing time between one output frequency and the next, as well. The slewing time between frequencies is equally important in frequency synthesizers as is the settling time. When a frequency hop is initiated, it is the error signal, which propagates to the VCO, that controls slew of the VCO output frequency. The bandwidth of the system is a measure of how quickly

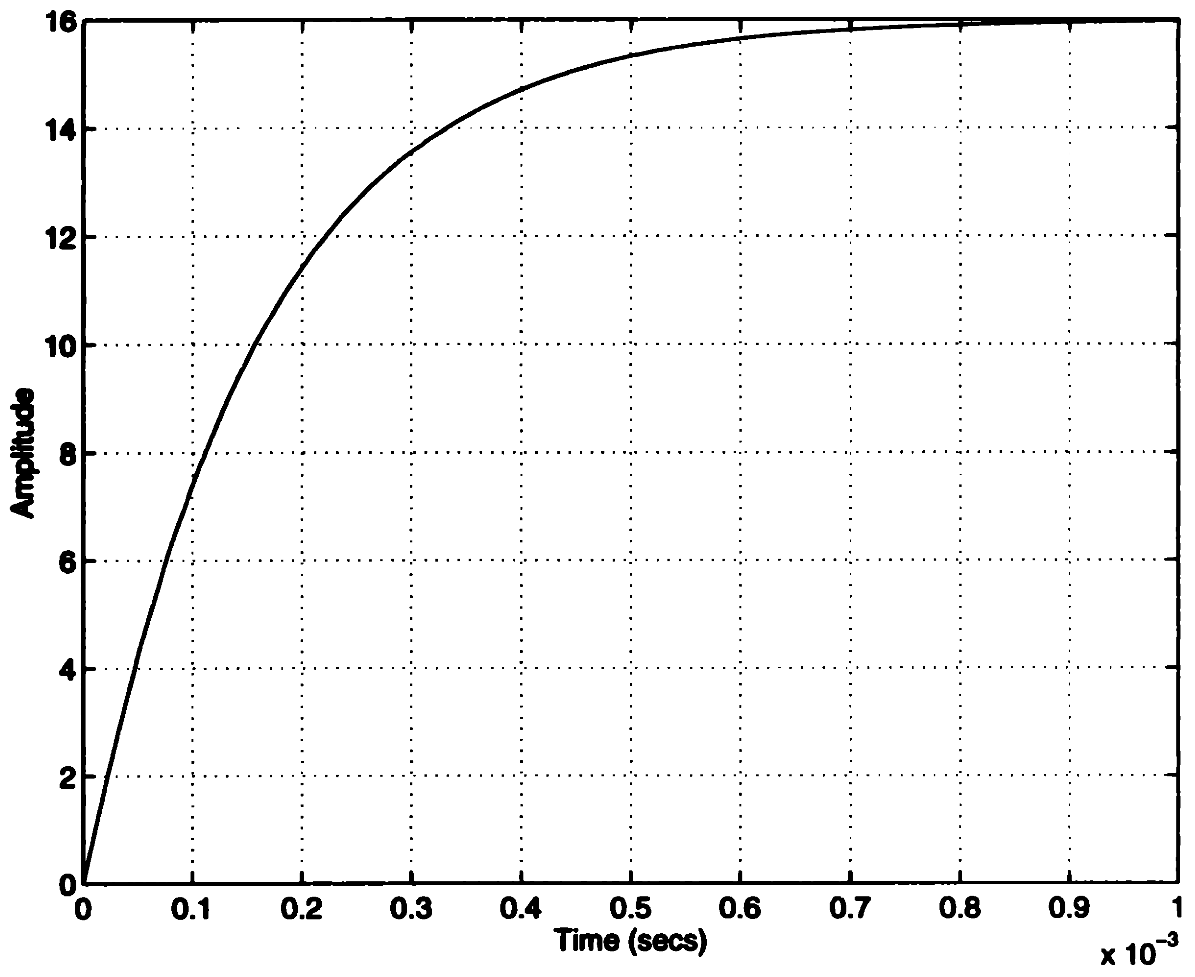

Figure 3-9: Sample step response of a typical Frequency Synthesizer.

the output of the system responds to an error signal.

One additional detail is apparent in this figure. Notice that unlike the PLL, which had a steady state gain of 1, the frequency synthesizers steady state gain is $\frac{1}{k_{div}}$. In the graph in figure 3-9, which represents the response of the synthesizer to a step change in phase, it is therefore apparent that the divider is a divide-by-16 divider.

If instead of requiring either, 30MHz or 32MHz at the output of the synthesizer studied thus far, what if an output frequency of 31MHz was needed? How would this be done?

Fractional-N Divider Dynamics

One way to accomplish this is by dithering the divider between a value of 15 and 16. This idea of dithering between two values is at the heart of a conventional divider topology known as a Fractional-N divider. Additional control logic is used to keep track of when to dither between divide ratios. For the case where 31MHz is the desired output frequency, the dither will have to alternate equally between 15 and 16. The error signal from the phase discriminator would have a period of 1MHz. The error signal period is determined by realizing that when the output frequency of the VCO settles at 31MHz, neither a divide-by-15 cycle, or a divide-by-16 cycle will generate a 2MHz wave at the output of the divider. For the former cycle the waveform from the divider will have a corresponding output frequency of

$$\frac{31\text{MHz}}{15} \approx 2.067\text{MHz}, \tag{3.30}$$

and for the latter cycle the divider will have a corresponding output frequency of

$$\frac{31\text{MHz}}{16} \approx 1.938\text{MHz} \tag{3.31}$$

This leads, respectively, to a negative error signal followed by a positive error signal; and this pattern repeats. The period of the pattern is two cycles of the reference clock which translates to a frequency of 1MHz.

If a frequency of 30.5MHz was needed, the dither pattern would change to be three divide-by-15 cycles followed by one divide-by-16 cycle. However, this pattern would have a frequency component which would lie below 1MHz. This would therefore require a reduction in the low-pass filter bandwidth or the propagation of this component to the VCO.

The assumption that the VCO output frequency does not change during once full cycle of the dither pattern is a valid, because the low-pass filter *must* reject the dither disturbances in order to generate a steady state output frequency – the primary goal of a frequency synthesizer – from the VCO.

All these aspects of the conventional synthesizer topology will be addressed in detail within the next chapter. Now that the dynamics of the analysis are complete, a closer look at output frequency transitions or 'frequency hopping' is necessary. To facilitate this, keep in mind the goal of this research; to develop a synthesizer that is geared for the 2.4GHz ISM band.

Chapter 4

Phase Tracking Frequency Synthesizer

The various synthesizer topologies that have been discussed in chapter 2 can all provide an accurate output when operating in the steady state case. For the dynamic case, encountered during frequency-hop transitions, they can have significant problems with settling. For conventional commercial divider topologies, the dynamic case settling problems are inherent because they maintain the state within the divider between successive frequency hops, and also because of the low loop bandwidth required for low phase noise and good frequency accuracy.

4.1 Loop Settling Times

4.1.1 Fractional-N

A typical Fractional-N type divider relies on an averaged steady state error signal $v_{lpf}(t)$ to calibrate the output frequency of the VCO. As was discussed in the previous chapter, when the synthesizer is required to move to a new frequency, the dither pattern in the divider changes. This change propagates around the loop and the averaged steady state error signal $v_{lpf}(t)$ settles at a new value.

With this method, the settling time of $v_{vco}(t)$ is fundamentally determined by the cut-off frequency of the low-pass filter and the gain of each block (refer to the equations on page 36). From these equations, the rise time for the step response of an ideal synthesizer of the filter to a step input response is

$$t_r = \frac{2.2}{\omega_n}, \tag{4.1}$$

where ω_n represents the natural frequency of the synthesizer. (It should be noted that for a critically damped or over damped system, the rise time would approximate the settling time.) The rise time could be hastened, by increasing the bandwidth of the filter. The larger the increase in the bandwidth of the filter the more phase noise is introduced at the control input of the VCO. This phase noise, recall, would result from the high frequency components of the discriminator's output. Another way to effect the settling time of $v_{vco}(t)$ would require increasing the forward loop gain. While this is possible, there are certain practical limits beyond which stability issues begin to surface which makes this infeasible..

It has been assumed up to now that when a frequency change is commanded, the system immediately responds by changing the divider ratio. This however may not be the case with a Fractional-N type synthesizer. For this type of synthesizer, the result of a frequency hop command, causes the duty cycle of $v_{pd}(t)$ to change. This duty cycle change might not occur until the divider has completed its count, which can be as long as:

$$t_{divider} = \frac{n}{\text{Frequency}[v_{vco}]}K, \tag{4.2}$$

where n is the decimal size of the counter, Frequency$[v_{vco}]$ represents the free-running output frequency of the VCO, and K represents the number of cycles that comprise the dither pattern. For example, in the case discussed in Chapter 3, on page 38, n would be 16, Frequency$[v_{vco}]$ would be 32MHz, and K would be 4. t_r for this example would be 5μseconds, which is equivalent to 4 cycles of the reference clock. The total settling time therefore for a Fractional-N type synthesizer, lies between:

$$t_r \leq t_{settle} \leq t_r + t_{divider} \tag{4.3}$$

4.1.2 Issues Surrounding the Settling Time of the Loop

As was discussed earlier, the fundamental pieces in the frequency synthesizer loop that control the overall loop settling time are the low-pass filter and the divider. If we chose to optimize the divider by itself, the fastest settling time for the overall loop is constrained by t_r. Increasing the bandwidth of the low-pass filter only leads to larger phase noise and greater instability in the system.

Consider what occurs when a frequency hop is initiated. First, the divider ratio is changed, often without regard to the current state. This results in a change to the dither pattern. Now because the dither pattern has changed, the phase discriminator begins to accumulate larger and larger phase errors as the divided down VCO output *phase* no longer matches the reference input *phase*. The changes in the error voltage output from the phase discriminator are band-limited by the low-pass filter. The VCO frequency changes in response to this error voltage. This process continues until the divided down VCO output *phase* matches the reference input *phase*.

With this discussion in mind, we propose another solution. In order to increase the settling time of the synthesizer, we suggest a change that alters the primary control characteristic of the frequency synthesizer loop. Instead of maintaining phase and frequency control by changing the divider ratio (as with a Fractional-N) and generating an error signal, we propose the use of an external controller for frequency control. The controller is matched to the VCO, and its output is summed with the output from the low-pass filter. This sum is now the error voltage which controls the VCO. When a frequency hop is initiated, this controller would provide the error voltage necessary to change the VCO to the new frequency. Simultaneously, the divider ratio would be changed (see figure 4-1). This separates the two tasks of frequency and phase control, into two categories.

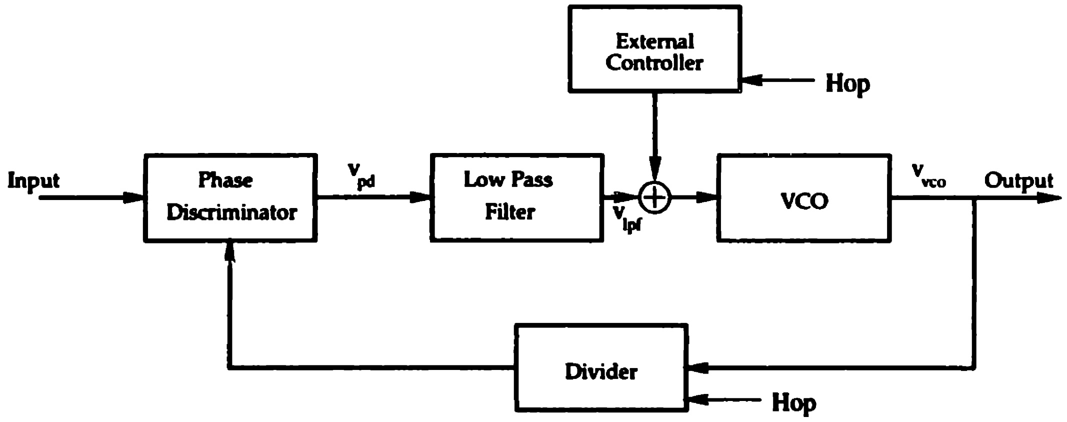

Figure 4-1: New topology proposed for a Frequency Synthesizer.

Coarse frequency tuning would be provided by the external controller, while fine frequency tuning would be provided by the loop. Ideally, $v_{lpf}(t)$ would be regulated to 0V, and the external would provide exact frequency required – this implies zero phase error in the loop between the reference input and the VCO output.

The benefits achieved through the use of such a structure are:

- Frequency adjustment is done externally. This removes the low-pass filter from affecting the slew time of the synthesizer output when engaged in a frequency hops.

- Given a divider which can track its phase state between frequency hops, the loop settling time should be dominated by the step response of the external controller and not the settling time of the loop or the divider.

4.2 New Loop Dynamics

While the dynamics of this new topology are altered from those studied in the previous chapter, the same analysis techniques can be used. To make the analysis clear, the loop is redrawn in figure 4-2.

The forward loop transfer function from the external controller to the output, (of

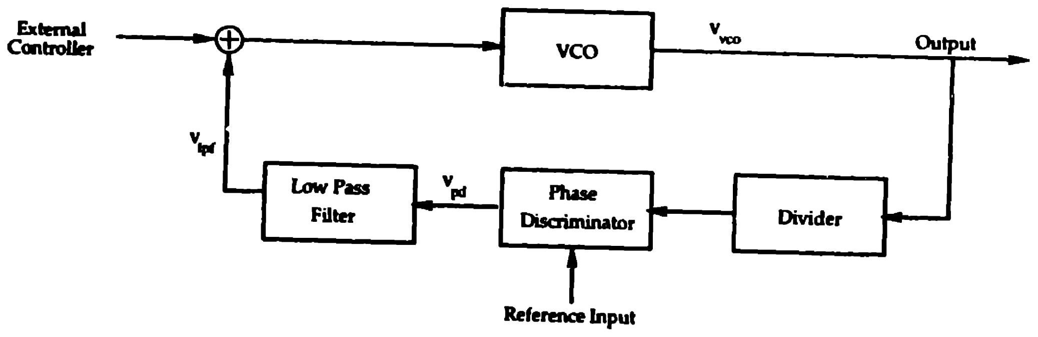

Figure 4-2: Redrawn Loop Topology.

the loop), is

$$G_f(s) = \frac{k_{vco}}{s} \tag{4.4}$$

The closed loop transfer function from the external controller to the output, (of the loop), is

$$\begin{aligned} H_{fs}(s) &= \frac{G_f(s)}{1 + G_f(s)\frac{k_{lpf}k_{div}k_{pd}\omega_{cf}}{s+\omega_{cf}}} && (4.5) \\ &= \frac{k_{vco}(s+\omega_{cf})}{s^2 + \omega_{cf}s + k_{vco}k_{lpf}k_{div}k_{pd}\omega_{cf}} && (4.6) \end{aligned}$$

Since this system is not a standard second order system, the standard second order parameters cannot be extracted directly. To get a measure of the bandwidth of the system therefore, we use Matlab™ to obtain the step response of this system.

The step response is identical to figure 3-9 in Chapter 3. Therefore, the corresponding rise times are identical as well. This analysis belies the performance of the system however. Inherent in this analysis is the assumption that divider block does not maintain state, or if it does the state is not maintained after a frequency hop. This topology however, requires not only additional hardware, but also a divider of a type that will maintain zero phase error. The frequency domain representation of

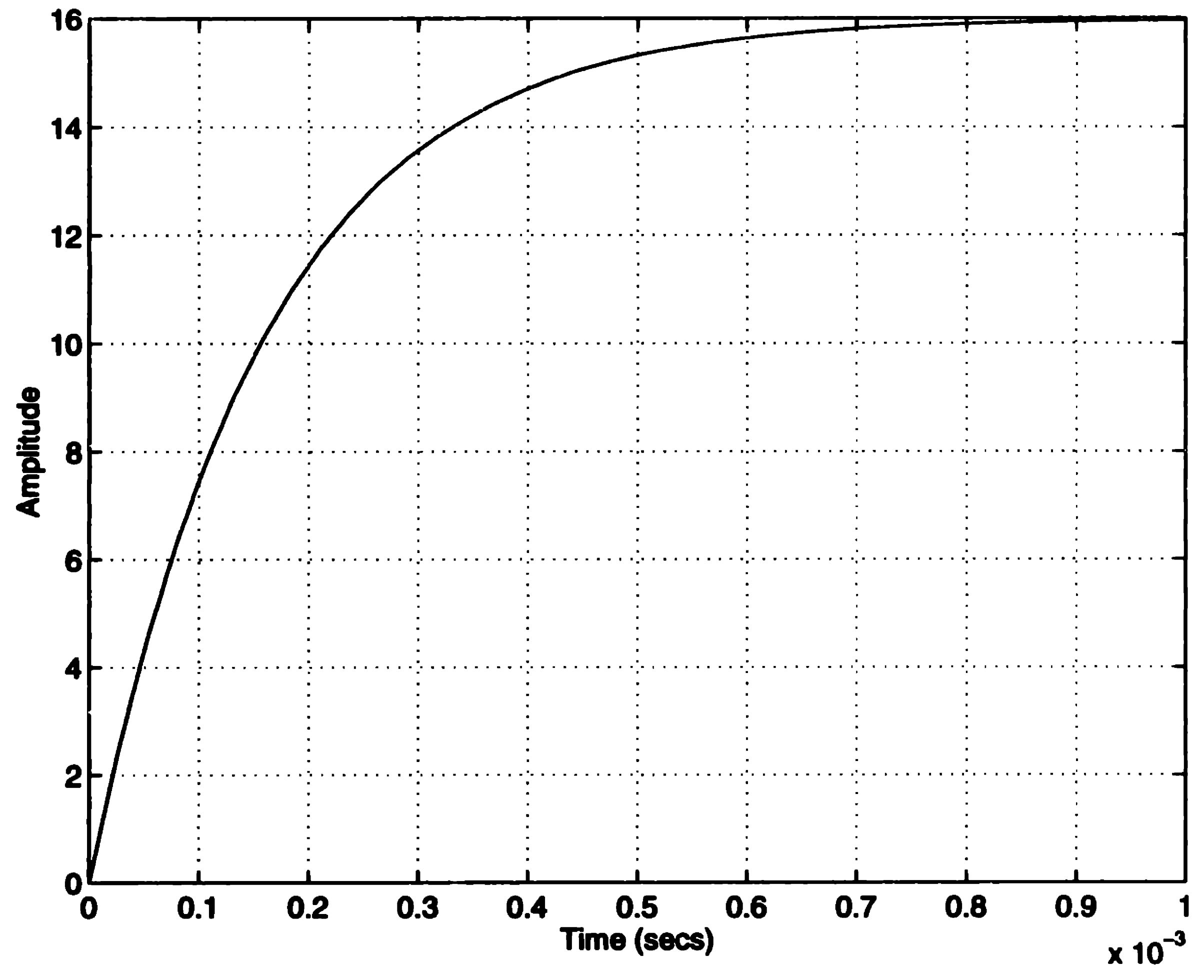

Figure 4-3: Step Response of New Loop Topology.

the output therefore is superposition of two responses, (since this a multiple input system), from equations 3.29 and 4.6, which is

$$Y_{fs}(s) = \frac{k_{vco}(s+\omega_{cf})}{s^2+\omega_{cf}s+k_{vco}k_{lpf}k_{div}k_{pd}\omega_{cf}}X_1(s) \quad + \quad \frac{k_{pd}k_{lpf}k_{vco}w_{cf}}{s^2+\omega_{cf}s+k_{pd}k_{lpf}k_{vco}\omega_{cf}k_{div}}X_2(s), \tag{4.7}$$

$Y_{fs}(s)$ is the frequency domain representation of the synthesizer output. $X_1(s)$ is the frequency domain representation of the external controller input, and $X_2(s)$ is the frequency domain representation of the reference input to the synthesizer.

4.3 Solutions for Fast Frequency Slewing

Having examined the basic motivation for this work, now the emphasis will be focused on the particular application at hand – a frequency synthesizer for the 2.4GHz ISM Band.

4.3.1 Overall Implementation

The implementation of an external controller for changing the output frequency from the VCO can be realized using an 128-byte RAM and a Digital to Analog Converter (DAC). Each of the 83 unique frequencies, which would be represented by 8-bit words in RAM. While less accuracy could be used for each bit position, more accuracy will allow better coarse tuning and the ability to track the non-linear characteristics of a VCO.

In order to maintain zero phase error between frequency hops, the divider architecture needed to be modified to track the current phase of the divider output. In order to achieve this a RAM look-up table is necessary for this block as well. Like the DAC, an n-bit word is used to by the divider for each of the 83 frequencies. Two divider topologies are developed and a comparison is presented in the forthcoming pages between these and the traditional Fractional-N type divider.

The block diagram for the overall circuit is shown in figure 4-4. The 8-bit words for each frequency, would be determined by first disabling the current feedback path, and in its place placing a separate mechanism which would calibrate each desired output frequency, (from 2.402GHz to 2.485GHz), to a unique control word (see figure 4-5). Ideally, this *self-calibration* would only be necessary once at start-up.

4.3.2 Variable Step Size Divider

A variable step size counter is pictured in figure 4-6. It is made up of one m-bit register, one n-bit register, an adder, and a comparator.

At the input to this divider is a comparator, which transforms the analog sinusoidal waveform from the VCO to a digital square wave to synchronize and drive the

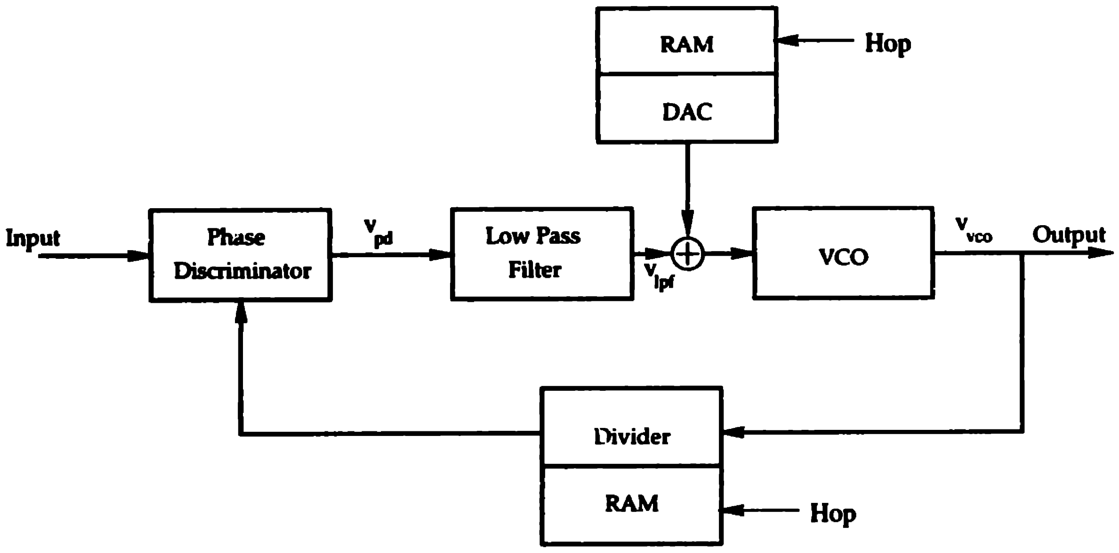

Figure 4-4: Overall Implementation of New Synthesizer Loop.

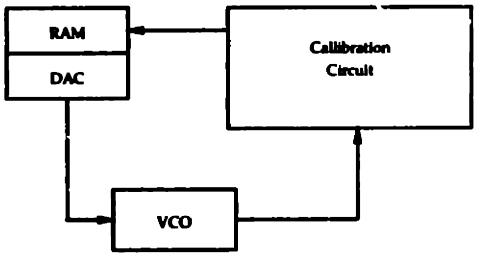

Figure 4-5: Calibration is accomplished by disconnecting the primary feedback loop and inserting in its place a calibration circuit which measures the output frequency of the VCO at each frequency step and loads the RAM table with the corresponding 8-bit word.

remaining digital divider circuitry. The output of the VCO is sensed at the input port of the comparator. In response to this input the comparator generates a **Hi** at its output when the signal at its input is greater than zero and a **Lo** otherwise. This signal is used to increment the *result* register.

The *result* register is an n-bit register which maintains the state of the divider. The output of this register is summed with the output of the *step size* register and this new value is loaded into the *result* register. The step size and values are chosen such that the product of the VCO output frequency and the step size is approximately

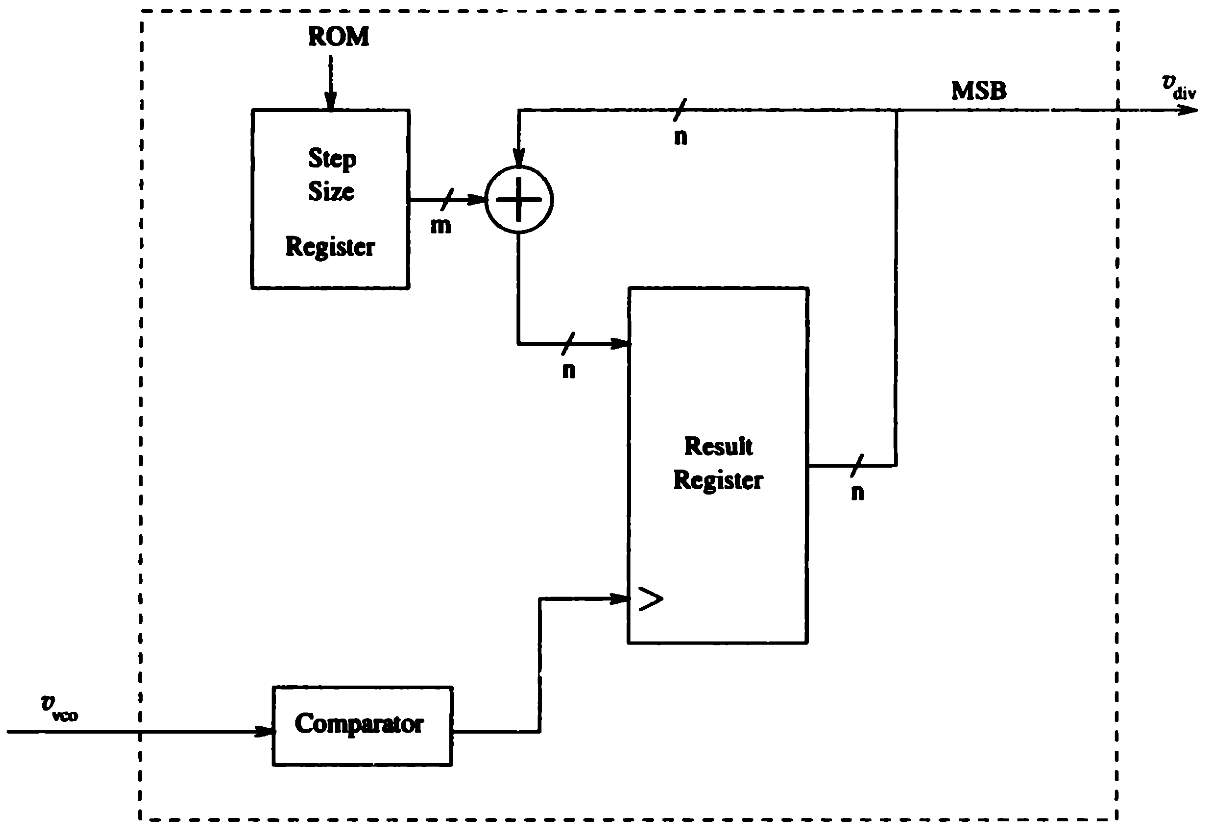

Figure 4-6: Variable Step Size Divider Schematic.

$2^n - 1$; the largest value capable of being represented by the register. The most significant bit (MSB) of the *result* register is used as the divider output signal, because if it takes τ seconds for the result register to go from a value of 0 to a value of $2^n - 1$, then the MSB will toggle back and forth between 0 and 1 every $\frac{\tau}{2}$ seconds. The expression for the value stored the register is

$$result[k] = step\ size + result[k-1] \tag{4.8}$$

When the value in the result register reaches $2^n - 1$, on the next increment, the register will simply wrap around. In this way, the MSB of this register will toggle back and forth between *Hi* (1) and *Lo* (0). The period of the MSB is

$$period[v_{div}] = \frac{(step\ size)(frequency[v_{vco}])}{2^n} \tag{4.9}$$

Each frequency has its own unique step size value, similar to the DAC, which has a unique control word for each frequency. When a frequency hop occurs, the value of the *step size* register is changed to a new value, which corresponds to the new frequency. The *step size* register value for each frequency is determined so that the period of the MSB remains independent of the output frequency from the VCO.

The topology is relatively simple, yet this solution resolves the inability of other types of dividers to accurately maintain the state of the synthesizer. While being simpler in design, this architecture has its share of design challenges. The primary one is the need for a sub-nanosecond n-bit adder[12]. The prospective size of such an adder is outlined in table 5.1. A secondary issue from the perspective of a designer, but a fundamental one from that of a customer, is the increased power dissipation that would accompany such a circuit.

4.3.3 Presetable Divider

The presetable counter pictured in figure 4-7 is comprised of an n-bit presetable counter, two n-bit registers, a block of combinational logic and a comparator.

At the input of this divider, similar to the previous divider, is a comparator. It performs a conversion from the analog VCO output to a digital waveform that is used as the decrement signal for the counter. The *current* register holds the preset value for the current frequency, while the *next* register holds the preset value for the new frequency. In steady state operation, the counter would start at the value preset from the *current* register and count down. Each time the counter reaches 0, the value from the *current* register is again preset into the counter.

When a frequency hop occurs, the current state of the counter is combined with the values from the *current* and *next* frequency registers via some combinational logic and the result is used to preset the counter at a new value. The purpose of

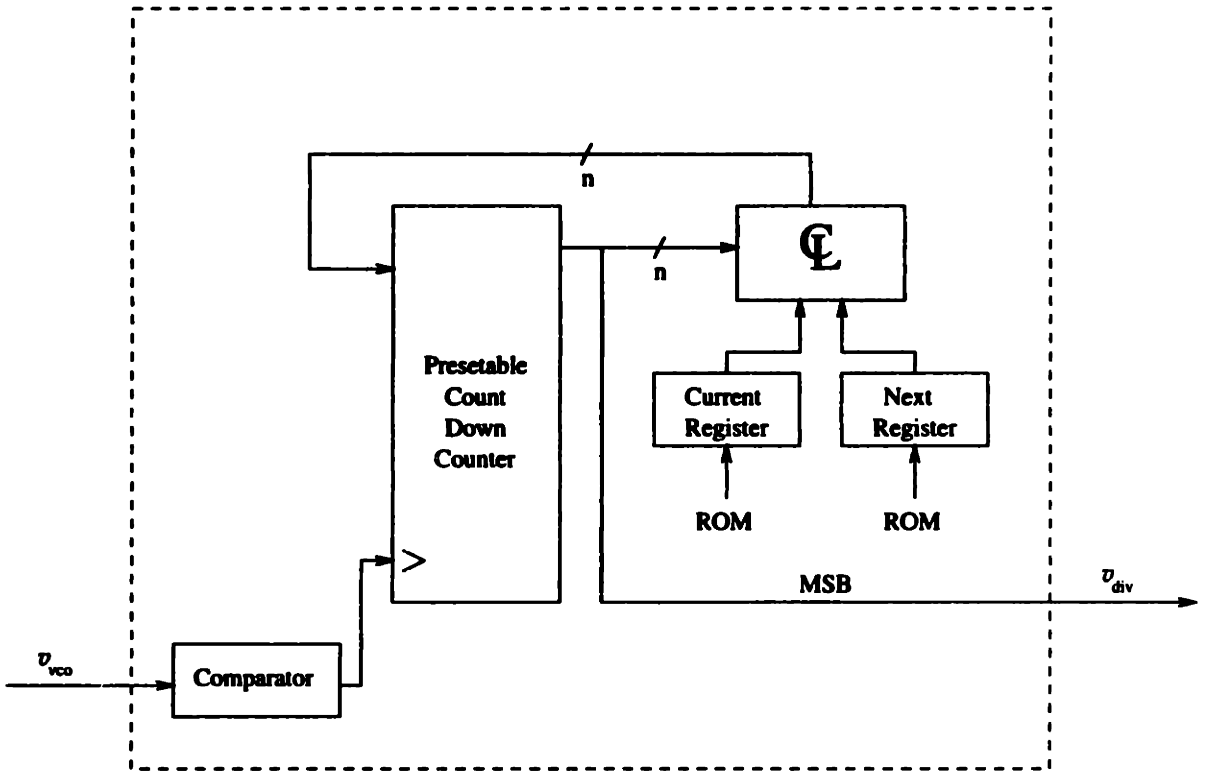

Figure 4-7: Presetable Divider Schematic.

this combinational logic is to change the state of the divider to maintain continuity of phase during a frequency hop. For example, should the divider be half way through its initial preset value from the *current* register, the combinational logic would produce a value that is fifty percent of the *next* register value.

The divider topology has the ability to track phase like the Variable Step Size Divider, however, it does not carry with it the same price in increased power dissipation. Additionally, a simple combinational logic block does not pose the same design challenges of a sub-nanosecond adder which would be necessary for the 2.4GHz band.

In the next chapter, we shall go onto discuss the simulation tests carried out on these topologies and benchmark them against the traditional Fractional-N topology to compare results.

Chapter 5

Functional Simulation, Test, and Results

Functional simulation and testing were conducted at a variety of different levels. Initially, the divider topologies were examined using Matlab and C tests. Following this a series of functional models for each of the synthesizer blocks was constructed using AHDL (Analog Hierarchical Description Language) within the Cadence^{TM} environment. The models were simulated individually and then combined and simulated using the SpectreVerilog Simulator in Cadence^{TM}.

5.1 Initial Divider Analysis

5.1.1 Variable Step Size Divider

For initial characterization tests, the Variable Step Size Divider was examined in Matlab. The program, *delta_table_period* (see Appendix A), was used to calculate the size of the *step size* and *result* registers as well characterize tolerance to jitter in the VCO frequency of ±20ppm[1].

In performing the first set of calculations to determine the size of the registers, constraints were placed on certain parameters. In order to maintain zero phase error, the *step size* register required a unique value for each frequency. Additionally, since the MSB was decoded as the output of the divider,

$$\frac{\textit{step size} \times \textit{frequency}[v_{\mathrm{vco}}]}{\textit{frequency}[v_{\mathrm{div}}]} \approx 2^n, \tag{5.1}$$

where n represents the width of the *result* register. Given the range of output frequencies specified for the 2.4GHz ISM band, the following table enumerates the necessary width (in bits) of the *result* register and *step size* registers. For *result* register sizes where the corresponding *step size* register which has an "N/A" listed as their size, no suitable width could be found for the *step size* register which would satisfy equation 5.1.

Ref. Frequency (MHz)	Result Register Size												
	Step Size Register Size												
1	24	23	22	21	20	19	18	17	16	15	14	13	12
	13	12	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	24	23	22	21	20	19	18	17	16	15	14	13	12
	17	16	15	14	13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
50	24	23	22	21	20	19	18	17	16	15	14	13	12
	19	18	17	16	15	14	13	12	N/A	N/A	N/A	N/A	N/A
100	24	23	22	21	20	19	18	17	16	15	14	13	12
	20	19	18	17	16	15	14	13	12	N/A	N/A	N/A	N/A
250	24	23	22	21	20	19	18	17	16	15	14	13	12
	21	20	19	18	17	16	15	14	13	12	N/A	N/A	N/A
500	24	23	22	21	20	19	18	17	16	15	14	13	12
	22	21	20	19	18	17	16	15	14	13	12	N/A	N/A
1000	24	23	22	21	20	19	18	17	16	15	14	13	12
	23	22	21	20	19	18	17	16	15	14	13	12	N/A

Table 5.1: Register sizes for VSSD calculated for various Reference Frequencies.

From the results shown in table 5.1, a lower limit on the size of the *step size* register is determined to be 12 bits.

This code also created the table of step sizes that would be programmed into RAM for the divider. The data stored in this RAM and the RAM for the DAC, would be used together to hop the synthesizer from one frequency to the next. For a 1MHz reference frequency, the tables in Appendix B provide the values for the *step size* register for each of the 83 frequencies at both *result* register sizes; 23-bits and 24-bits. At this reference frequency, for a 23-bit *result* register, figure 5-1 shows the results of the calculation from equation 5.1. The corresponding phase error that results from the choice of step size at each of the frequencies (from 2.402GHz to 2.485GHz) is shown to be approximately less than .1 degree.

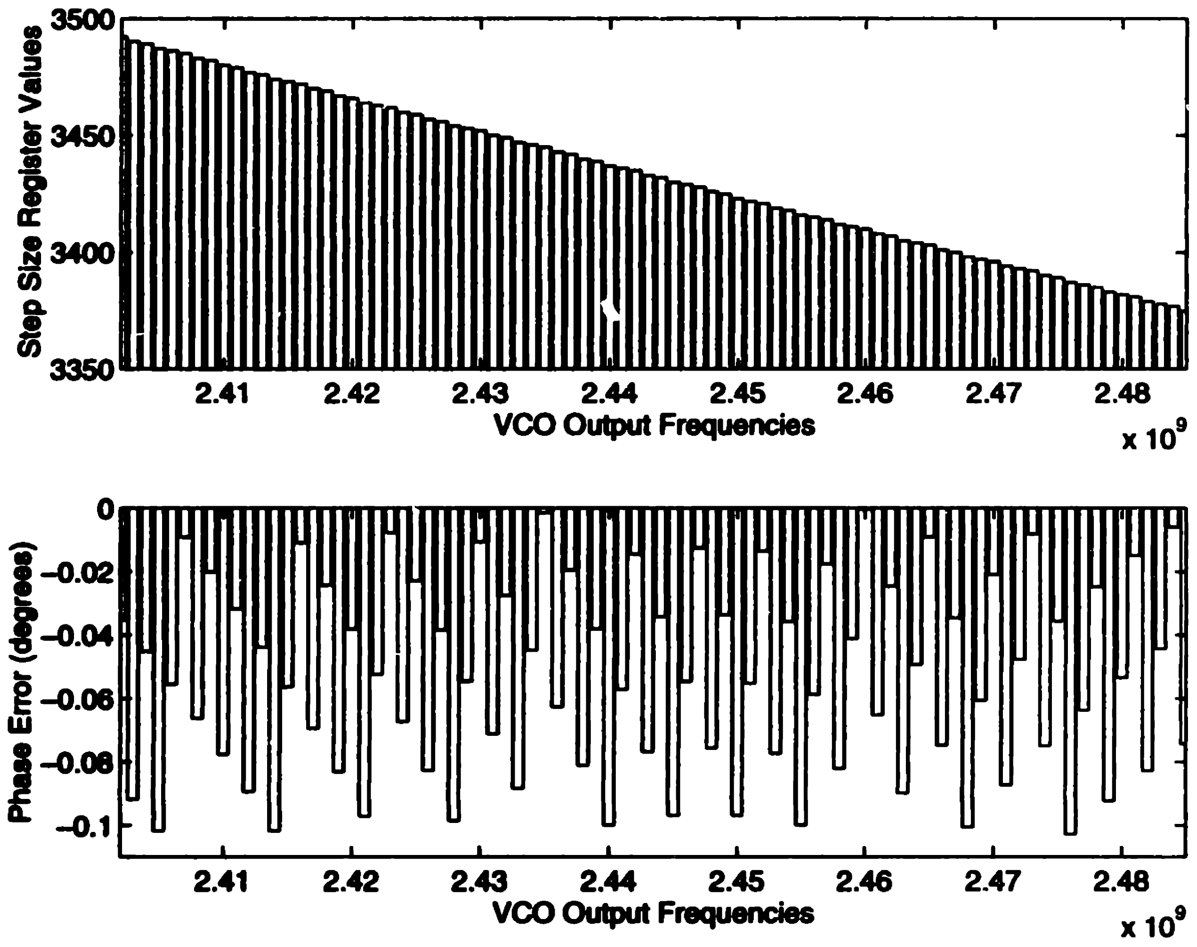

Figure 5-1: Phase Errors introduced by the values chosen for the *step size* register at each of the 83 frequencies.

The benefit of the Variable Step Size Divider structure becomes apparent, when considering the possibilities of calibrating both RAM look-up tables at the time of startup for the synthesizer. This would allow variations in reference frequency and VCO non-linearities to be calibrated out.

In order to determine the tolerance of the divider to frequency jitter, the *delta_table_period* program was modified to measure the sensitivity of the VSSD topology, under the same conditions described previously. Ideally, the divider would not respond to frequency errors within the tolerance window for the synthesizer (± 25ppm in this case[1]).

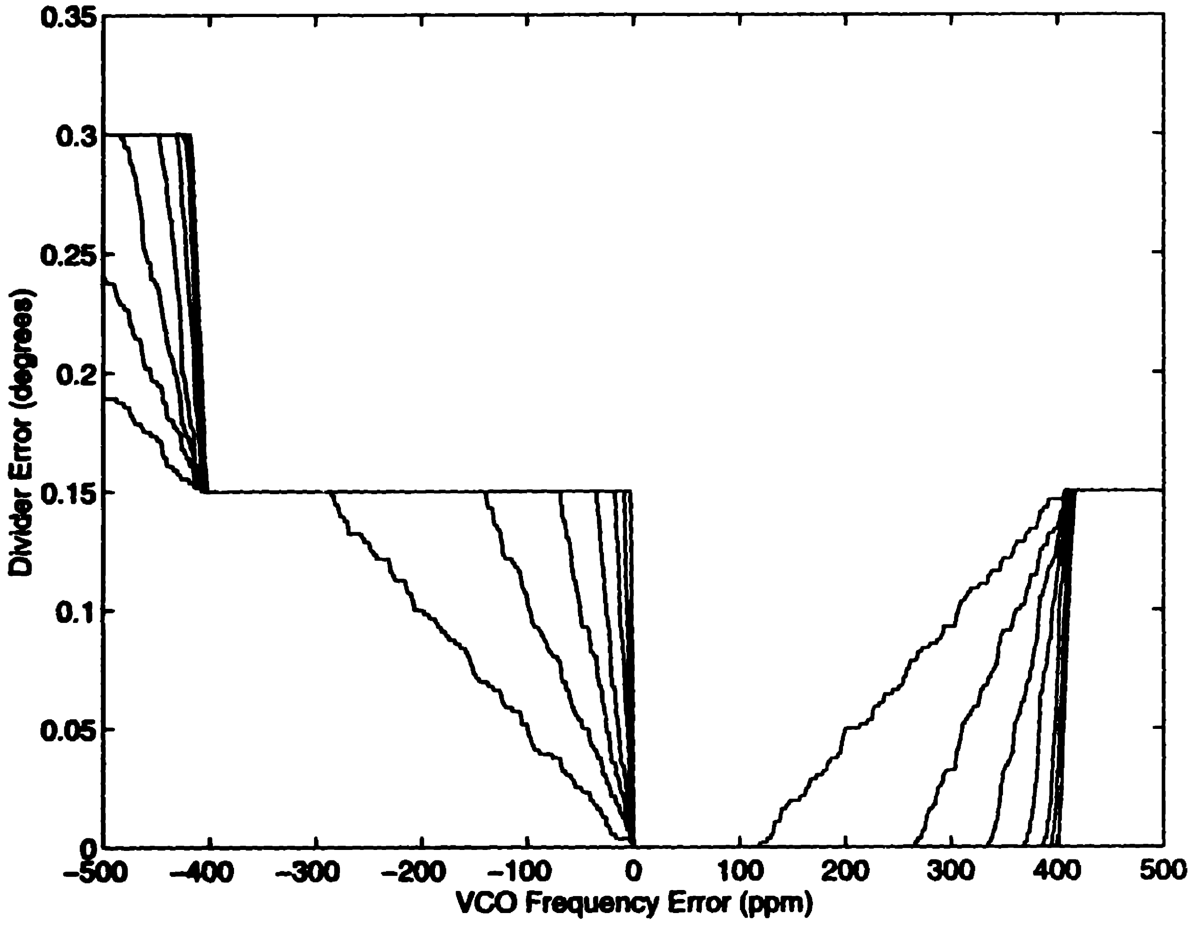

Figure 5-2: Divider Output Error vs. VCO Frequency Error. The *result* register width is stepped from 23 to 32, where the square wave error characteristics correspond to the 32-bit register size.

The simulation results in figure 5-2 show that for increasing width sizes of the *result* register, at a reference frequency of 1MHz, the divider responds to negative frequency error which is greater than 0ppm, but does not respond to positive frequency error until it is greater than 100ppm. The resulting phase error at -25ppm, however, is still less than .01 degrees.

5.1.2 Presetable Divider

The presetable divider topology was simulated first using a C model. (This code can be found in Appendix A, under the name *preset_div*.) This was done primarily to speed up execution time of the code. The preliminary tests were run to demonstrate the potential of such a divider, and to obtain a quantitative measure of the error sustained from each type of preset strategy. Preset strategy is the generic name used to identify the various schemes that could be implemented within the combinational logic block to preset the adder at the time of a frequency hop.

The *preset_div* program has a register which initially starts preset with the value 2402, corresponding to a VCO frequency of 2.402GHz. The counter counts down on each cycle of the loop and upon reaching 0, it resets to the value stored in the *current* register – initially this is 2402. The frequency hopping is implemented in a random fashion only at the occurrence of a particular random number. The remaining part of the program implements the combinational logic with a particular preset strategy and the corresponding divider error is written to a file.

The following are a number of the preset strategies that were tested along with a plot displaying the distribution of the phase errors for the strategy.

Immediate Preset with New Value

This preset strategy loads the new frequency divider value immediately into the counter upon a hop command. It does not take into account the current counter state and as a result does not preserve phase between hops. This is the base case against which we measure other strategies. The distribution of the phase error is essentially uniform, with a mean of 177 degrees.

Delayed Preset with New Value

The delayed preset strategy does not respond to a step change change in frequency until the counter reaches 0, at which time it loads the new frequency counter value. As the results in figure 5-4 indicate, this method does not maintain 0 phase error at

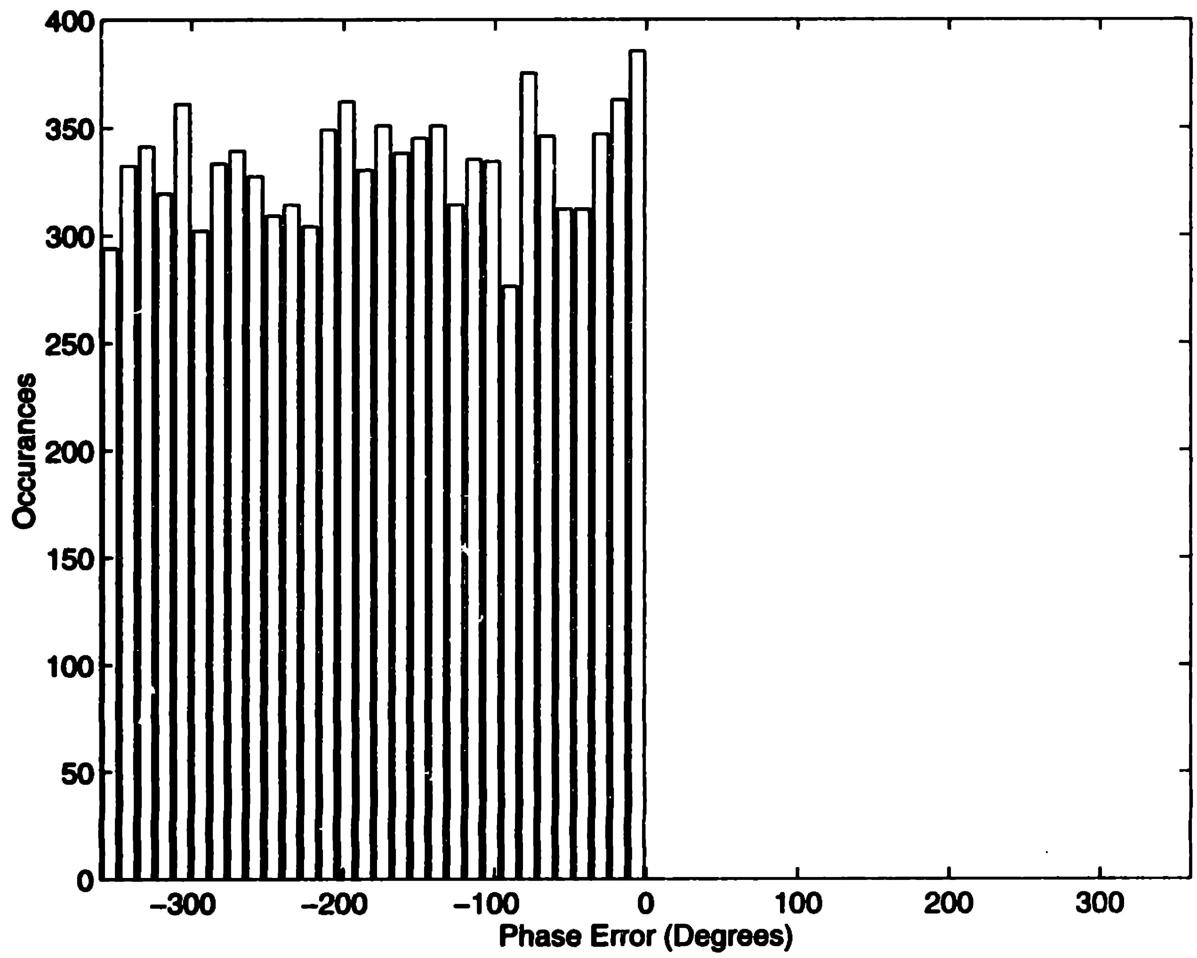

Figure 5-3: Histogram of Distribution of Phase Error for Immediate Preset method.

hop time, but it does maintain state better than the Immediate Preset strategy. The maximum phase error that results from this method is ±11 degrees, and the mean phase error is approximately -.02 degrees.

Immediate Preset with Exact Value

This strategy depicts the ideal case, where both the state of the divider is maintained and where phase error is minimized. With this method, when a hop command is issued, the current state of the divider is used to determine what fraction of the old count has been completed. Then the same fraction of the new count is preset into the counter.

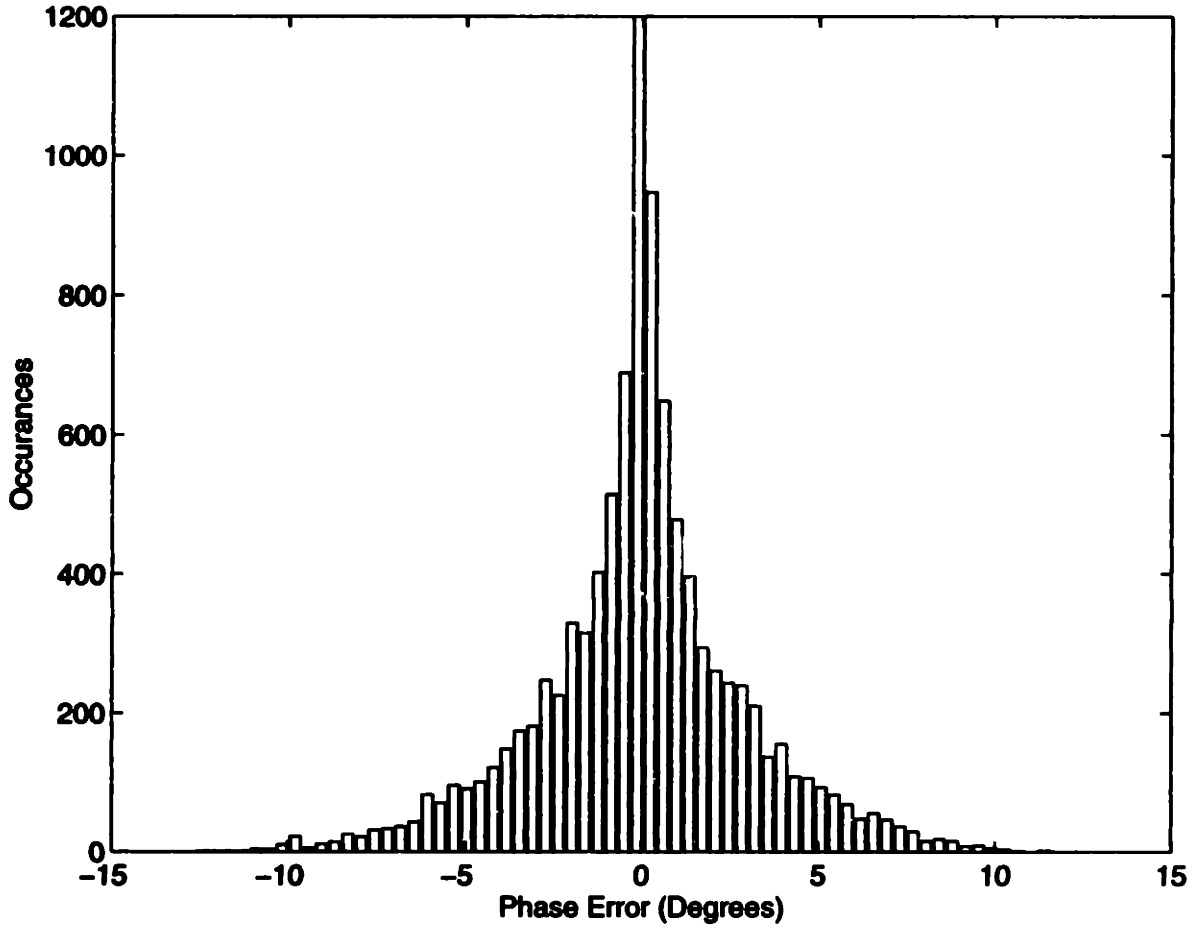

Figure 5-4: Histogram of Distribution of Phase Error for Delayed Preset method.

$$\frac{counter}{current} \times \; next = preset\ value \tag{5.2}$$

Here current and next represent the registers *current* and *next* which are articulated in the preset divider topology illustrated in Chapter 4, figure 4-7.

Other Strategies

Other strategies were tried to generate better results than the delayed preset method which did not require the computational overhead of the immediate preset strategy. A variation on the delayed preset method were tested, which, did not wait for the

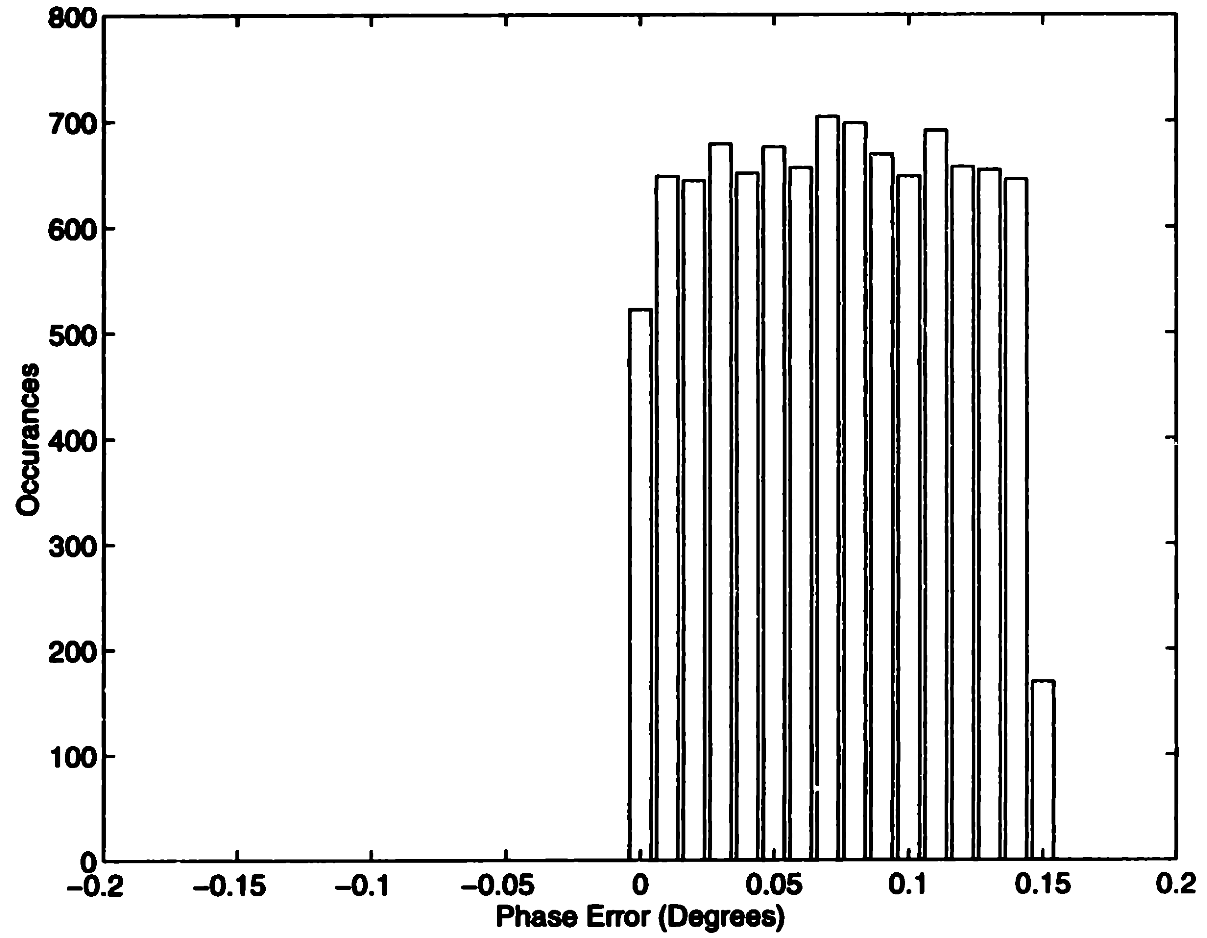

Figure 5-5: Histogram of Distribution of Phase Error for Exact Preset method.

counter to reach 0 before loading it after a hop command was issued. Instead, higher threshold values were chosen: 1,2, ... ,128. These tests demonstrated that, for low threshold values (< 64), the mean phase error remains the same however, the standard deviation progressively increases.

Other strategies involved using bits from the counter to shift the result of the *next* register, thereby simulating a divide. None of these proved better than the delayed preset strategy.

5.2 Power and Timing Issues

5.2.1 Variable Step Size Divider

To obtain an understanding of some of the consequences of this divider topology, two tests were performed to characterize power dissipation and timing considerations.

The first of these tests was a simple power dissipation calculation for this topology. Since total power dissipation is proportional to the number of bit changes plus a constant, the *power_calc* program (see Appendix A) kept track of the number of bits that changed for one cycle of the reference clock. This was done for each frequency and the value returned was an average number of bit changes for this divider topology. The result was compared with the number of bit changes that a standard Fractional-N divider performs for the same period of one cycle of the reference clock. The comparison test indicated that the VSSD topology consumed approximately eight times more power than the Fractional-N topology. A more rigorous test was not performed using Spice or a similar program because the circuit for this topology was not implemented.

A timing analysis of the VSSD topology revealed a need for very aggressive performance. The timing equation for the critical data path is given in equation 5.3 for the VSSD

$$t_{clk-q} + t_{prop\_adder} + t_{setup} \leq 402ps \tag{5.3}$$

These numbers will be very difficult to achieve with current 0.35μm CMOS technology. They could be achieved with bipolar, ECL or with 0.25μm CMOS technology. The most pressing design challenge, however, would be implementing a sub-nanosecond 12-bit adder.

5.2.2 Presetable Divider

The presetable divider has the distinction of capturing the benefits of the Variable Step Size Divider – mainly the ability to maintain phase state information and produce zero phase error – without the added design challenges of that topology.

Since the presetable topology is implemented using a counter as the fundamental central block, it is no different from a Fractional-N type divider. In fact, these dividers often tend to be quite complex and require sophisticated circuits for control of the divider dither algorithms. The presetable divider requires supporting logic, but it is far less complicated to achieve a max of ± 10 degrees with the Delayed Preset strategy than with a Fractional-N, dither type divider.

Since the power consumption along the data path through the combinational logic is only present at hop time, the average power consumption is determined by the counter. Similarly, timing issues only restrict the complexity of the combinational logic path. The use of a pipe-lined architecture, however, could eliminate this with a minimal introduction of phase error.

This topology is clearly simple to implement while providing good phase tracking, satisfying the requirements of a divider specified in Chapter 4. As it is the topology of choice, it shall be incorporated in the circuit design of the frequency synthesizer to demonstrate its potential. The power and timing issues of this topology will be investigated in further detail in the next chapter.

5.3 AHDL Simulations with Cadence

Following the initial tests of both power and timing, a functional model of a frequency synthesizer was built using AHDL within Cadence. (The models have been included in the Appendix B for examination.) The model simulated incorporates a DAC type device providing the frequency command, and uses the presetable divider topology in the feedback path.

A number of static, single frequency, and dynamic, frequency hop, simulations were run to demonstrate the performance of the frequency synthesizer.

These simulations not only verified the initial predictions of the various topologies, but also served to highlight the consequence of phase error on the loop performance. These results therefore support and validate the emphasis of the work so far to minimize phase error within the synthesizer.

Chapter 6

Synthesizer Circuit Design and Simulation

The frequency synthesizer circuit is comprised of 5 blocks: the phase discriminator, a low-pass filter, the divider, a multiplier and a DAC; the VCO is not included for two reasons. First, the inclusion of a VCO on the same piece of silicon with the synthesizer would prove to be difficult, given the strenuous performance requirements in the present silicon technology. Secondly, the use of an off-chip VCO would allow test and characterization of this design at lower frequencies.

6.1 Phase Discriminator

The skeleton of the phase discriminator circuit is based on a typical differential receiver topology with one additional enhancement. Typically, phase discriminators have a dead-band region (refer to figure 6-2) near the zero-crossing point, where the phase difference between the inputs is approximately zero[13].

The differential voltage used to charge or discharge a capacitor, in a such a topology, is provided directly from the signal source; in this case this would be the divider output $v_{div}(t)$ and the reference input $v_{in}(t)$. For the enhanced charge pump type phase discriminator in figure 6-3, the differential pair is no longer directly controlled by the signal source. Instead, these input signals drive a logic block, the discrimina-

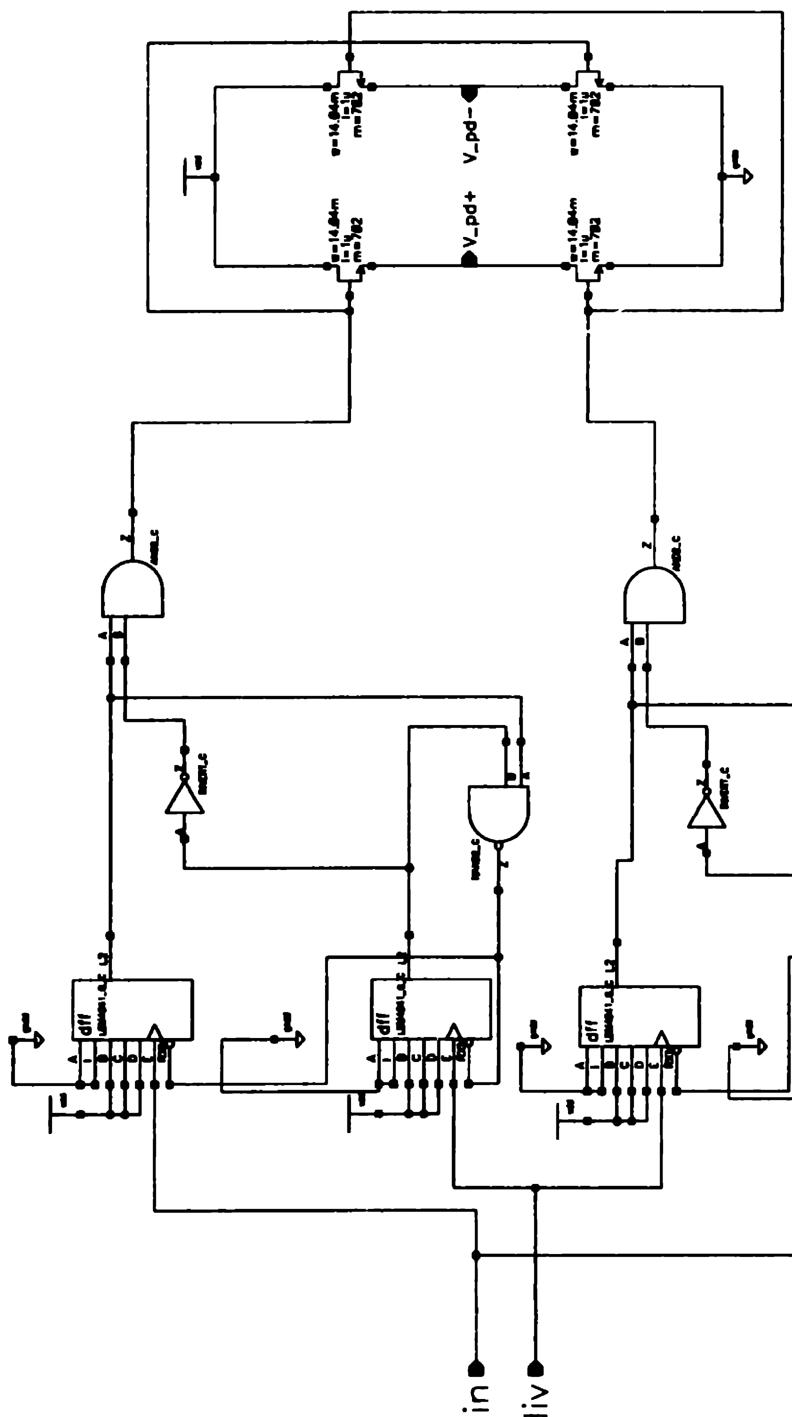

Figure 6-1: Detector Logic Block.

tor, which generates two output signals. These two signals, *charge* and *discharge*, are then used to control a charge pump.

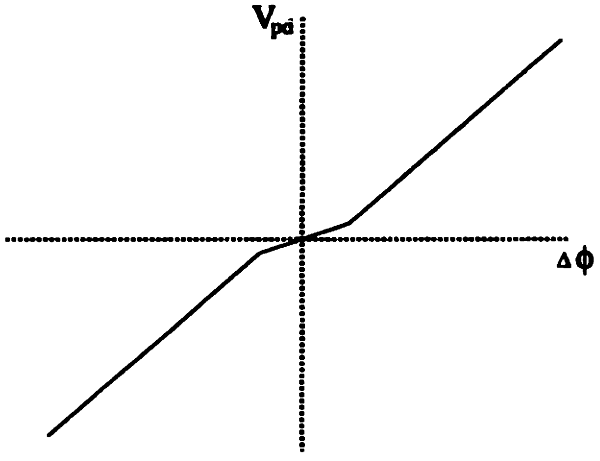

Figure 6-2: Output voltage of the phase discriminator vs. phase difference of input signals – dead-band region near zero phase difference.

The logic pictured in figure 6-3, is one half of the discriminator circuit. The signals A and B enter this block and both are used to latch a logic value of 1 into individual registers; these are initially started in a reset state. If there is a rising edge on signal A before there is one on signal B, the output signal *charge* transitions from a 0 to a 1. The feedback reset circuit to the DFFs (D-flip-flops) remains at its present output value 1. The *charge* output will also remain at its present value indefinitely until a rising edge occurs on signal B.

When a rising edge is observed on signal B, the feedback signal resets both DFFs. The *charge* output then transitions back to 0 and the entire system returns to the reset state.

If signal B transitions first, the *charge* signal remains at 0. The system remains static, and is simply brought back to the reset state when A transitions to 1.

This circuit, therefore, only responds when signal A leads signal B. Using two such circuits where in one circuit signal A is *reference_clock* and B is the *divider_output* and in the other where these two are reversed, provide a means of determining the phase difference and the sign of the phase difference between the two input signals.

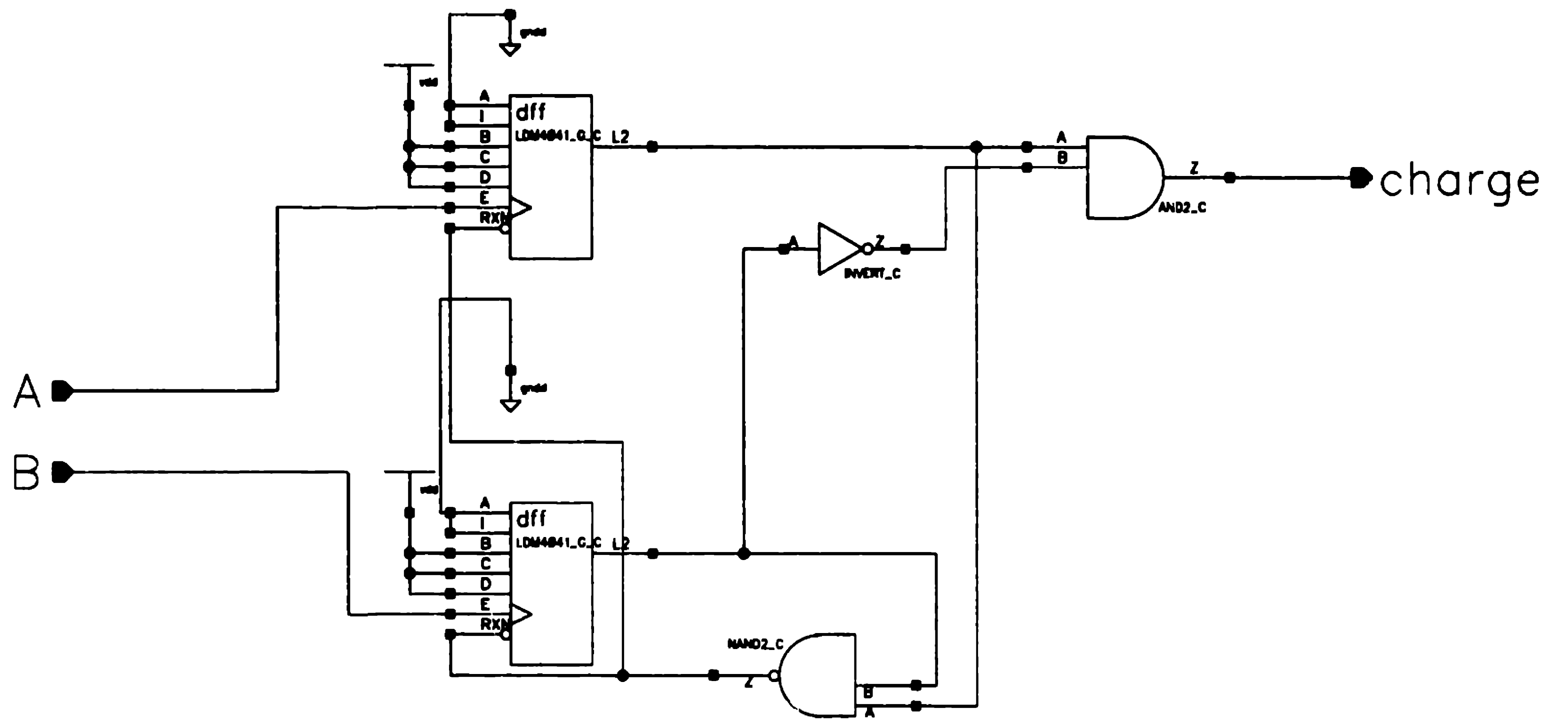

Figure 6-3: Fundamental Building Block of Detector.

The dead-band region is eliminated by insuring that the delay through the inverter from the output of the B DFF, is long enough to allow the *charge* output to rise to a logic level of 1, even if the rising edge of both A and B are coincident. The result is the relationship between input phase difference and v_{pd} shown in figure 6-4.

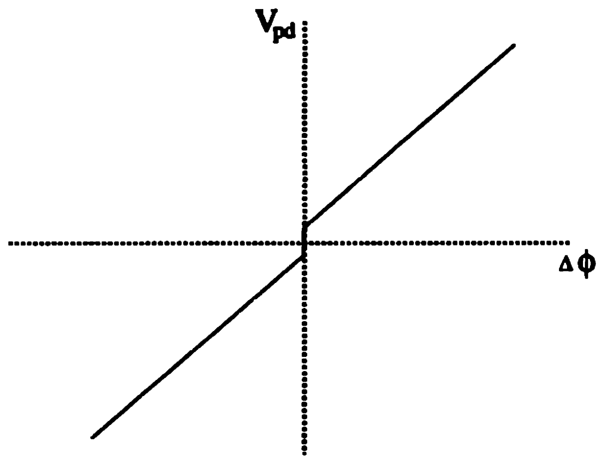

Figure 6-4: Output voltage of the phase discriminator vs. phase difference of input signals with no dead-band region.

Charge Pump

The output of the detector, the *charge* signals are used to drive a charge pump shown in figure 6-5. The pump is comprised of four NFETS which are separated into two pair where the FETs are in the pair stacked in series and the two stacks are in parallel between V_{dd} and Gnd. The gates of these pairs are connected along the diagonal. The output voltage of the phase discriminator will be measured across the the nodes V_{pd+} and V_{pd-} that straddle the center of the pump. One of the the two outputs from the discriminator will charge the capacitor towards a positive voltage while the other will charge the capacitor to a negative voltage. The positive and negative sign will be determined by the manner in which the voltage across the capacitor is measured. Thus one of the detector outputs is the *charge_up* signal while the other is the *charge_down* signal.

The advantage of using the detector can now be readily seen. If the *reference_clock*

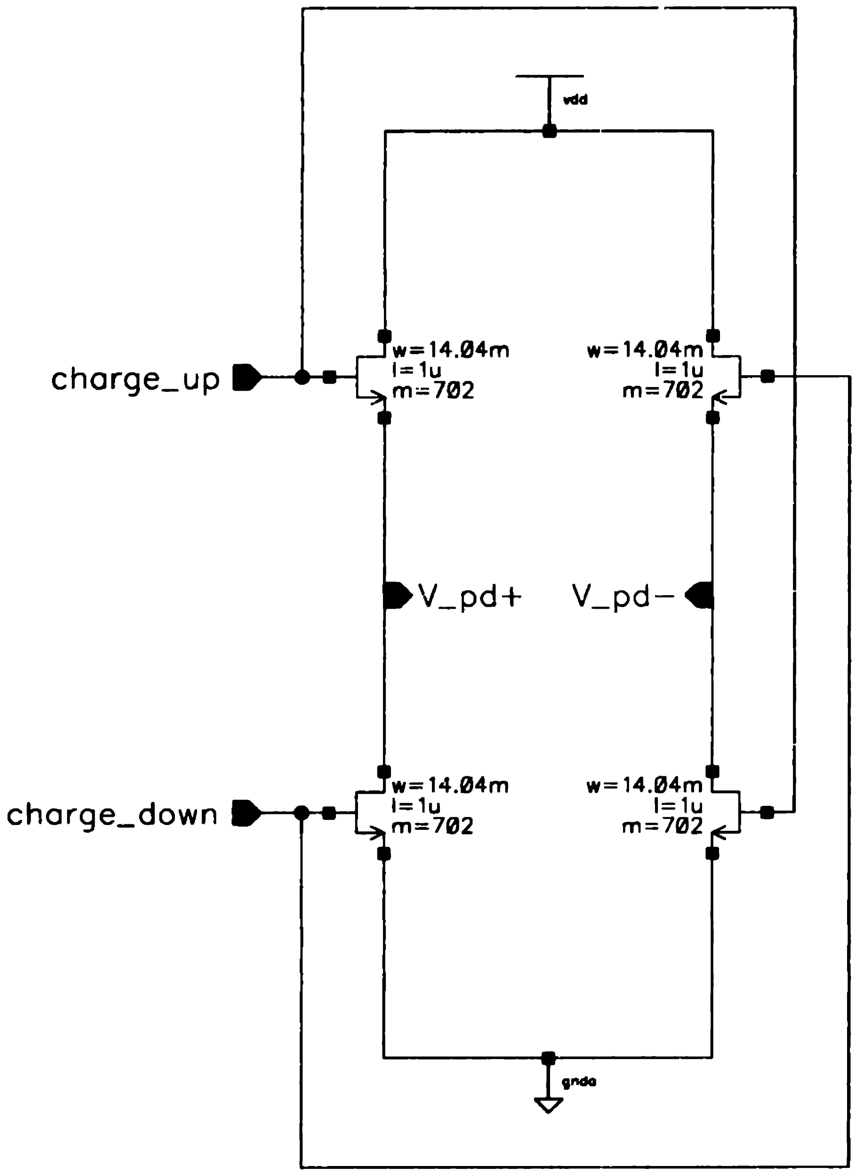

Figure 6-5: Charge Pump.

signal and *divider_output* signal had been driving the charge pump directly, for very small phase errors, the FETs would never turn on completely. The output voltage, v_{pd}, would not track linearly with phase error demonstrating a system response which would look like the plot shown in figure 6-2. After removing the dead-band region, this is still not a completely linear system block. This change, however, improves performance by tracking small phase errors and providing enough feedback gain at small errors to prevent large scale drifting of the VCO output phase and frequency.

6.2 Low Pass Filter

Since the voltage across the terminals of the phase discriminator block is proportional to the phase error between the *reference_clock* and *divider_output*, this output is directly passed to the next block – a single pole differential filter.

The filter is differentially implemented to improve common mode noise rejection in the mixed signal environment. The topology of the filter is given in figure 6-6.

In determining the transfer function for the filter as specified in figure 6-6, it is more accurate to write the transfer function between the input voltage v_{pd} and v_{lpf}. While a charge pump is considered a type of current source, the effective quantity that is being acted upon by the filter is the voltage generated at the nodes v_{pd+} and v_{pd-} by the charge pump.

$$\frac{V_{lpf}(s)}{V_{pd}(s)} = \frac{\frac{1}{Cs}}{\frac{1}{Cs} + 2R} \tag{6.1}$$

The transfer function demonstrates that as the frequency increases, the voltage at the output of the low-pass filter decreases. The cut-off frequency of the transfer function is determined by the choice of R and C. The transfer function can be simplified and written as

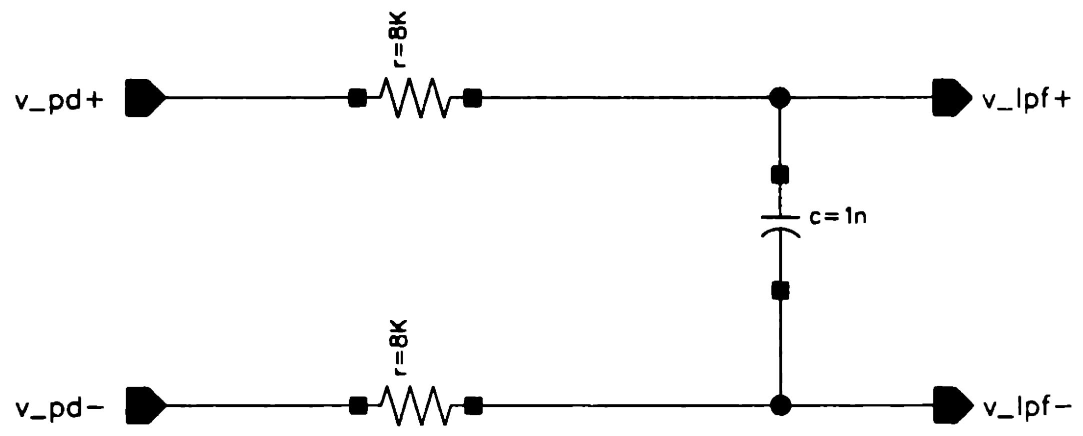

Figure 6-6: Low Pass Filter Circuit.

$$\frac{V_{lpf}(s)}{V_{pd}(s)} = \frac{\frac{1}{2RC}}{s + \frac{1}{2RC}}. \tag{6.2}$$

The filter pole location is at $\omega_{cf} = \frac{1}{RC}$ which corresponds to a cut-off frequency of

$$f_{cf} = \frac{1}{2\pi RC}. \tag{6.3}$$

The output of the low-pass filter would then feed into a summing junction which would combine this control voltage with the output from a DAC.

6.3 Digital to Analog Converter

In figure 6-7 an implementation of a current DAC is presented. A simple current mirror is used to establish a reference current on which the rest of the circuit depends. The output voltage v_{dac} is measured across a sense resistor of value 1KΩ, through which the current is drawn by the DAC.

The reference current i_{ref} is thus set so that when the input to the DAC is 0, v_{dac} is close to 3.6V, and when the input to the DAC is 127, v_{dac} is about .2V.

The second condition is the one which will determine the bias current for the circuit. Since the least significant bit of the DAC (0-bit) is identified with an NFET, which is the same size as the NFET used to generated the reference current, we label the current passing through, when turned-on, to be equal to i_{ref}. Each NFET for the following significant bits is twice the width of the previous. The total current drawn through the sense resistor by the DAC and the bias is

$$\begin{aligned} i_{ref} &= \frac{3.6V - .2V}{128x1K\Omega} && (6.4) \\ &= 26.5\mu A, && (6.5) \end{aligned}$$

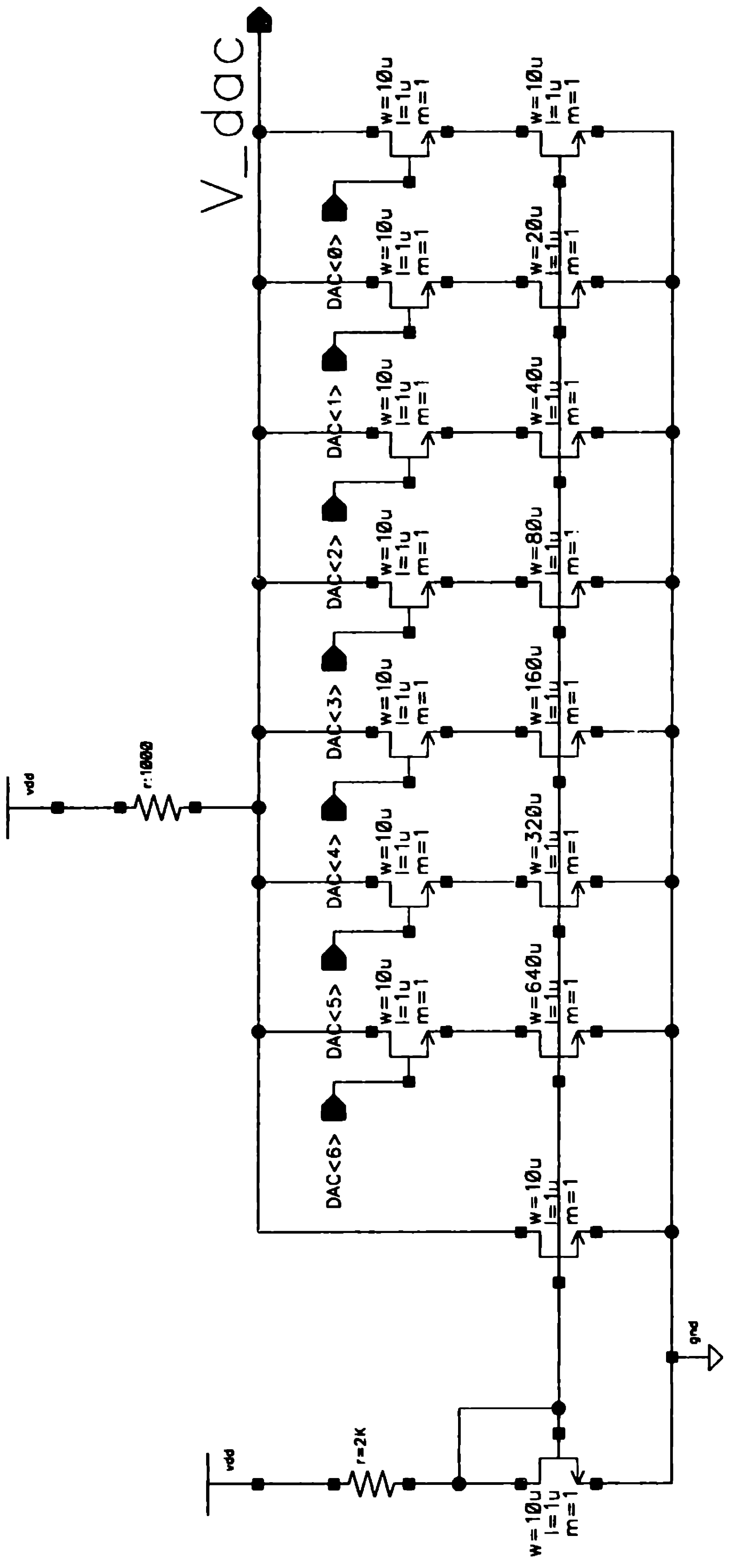

Figure 6-7: Digital to Analog Converter (DAC) Circuit.

where 3.6V represents the rail voltage V_{dd}. Applying the first condition now, when the input to the DAC is 0, the output voltage is

$$v_{dac} = 3.6V - (1K\Omega x 26.5\mu A) \tag{6.6}$$

$$= 3.57V \tag{6.7}$$

While the circuit performs under the given conditions, an increase or decrease in the i_{ref} current can be accommodated with a change in the sense resistors. The widths of the NFET gates could also be changed to provide a non-linear control of the current.

The output v_{dac} is summed with v_{lpf} and is provided as the control voltage to the VCO.

6.4 Divider

The divider circuit is separated into two parts. An analog input stage, and a digital core. The input sinusoidal VCO signal passes through a comparator which transforms the signal to CMOS levels. The input voltage swing to the comparator is restricted to be between ± 150mV. The comparator topology shown in figure 6-8 has as its first stage a high input impedance device. Without this, the VCO output would need to be able to source and sink a large amount of current, which would alter its output waveform and frequency.

The VCO output is coupled via a high-pass filter into a common-collector stage which is biased to level shift the signal. This output then drives a CMOS inverter to form the drive signal for the remaining divider circuitry.

The remaining circuitry implements the rest of the presetable divider topology. The two twelve bit registers, *current* and *next* are stacked one above the other and their outputs are driven into a multiplexer. Whenever a hop command is issued the *hop* bit toggles state thereby changing which register's value (*current* or *next*) is

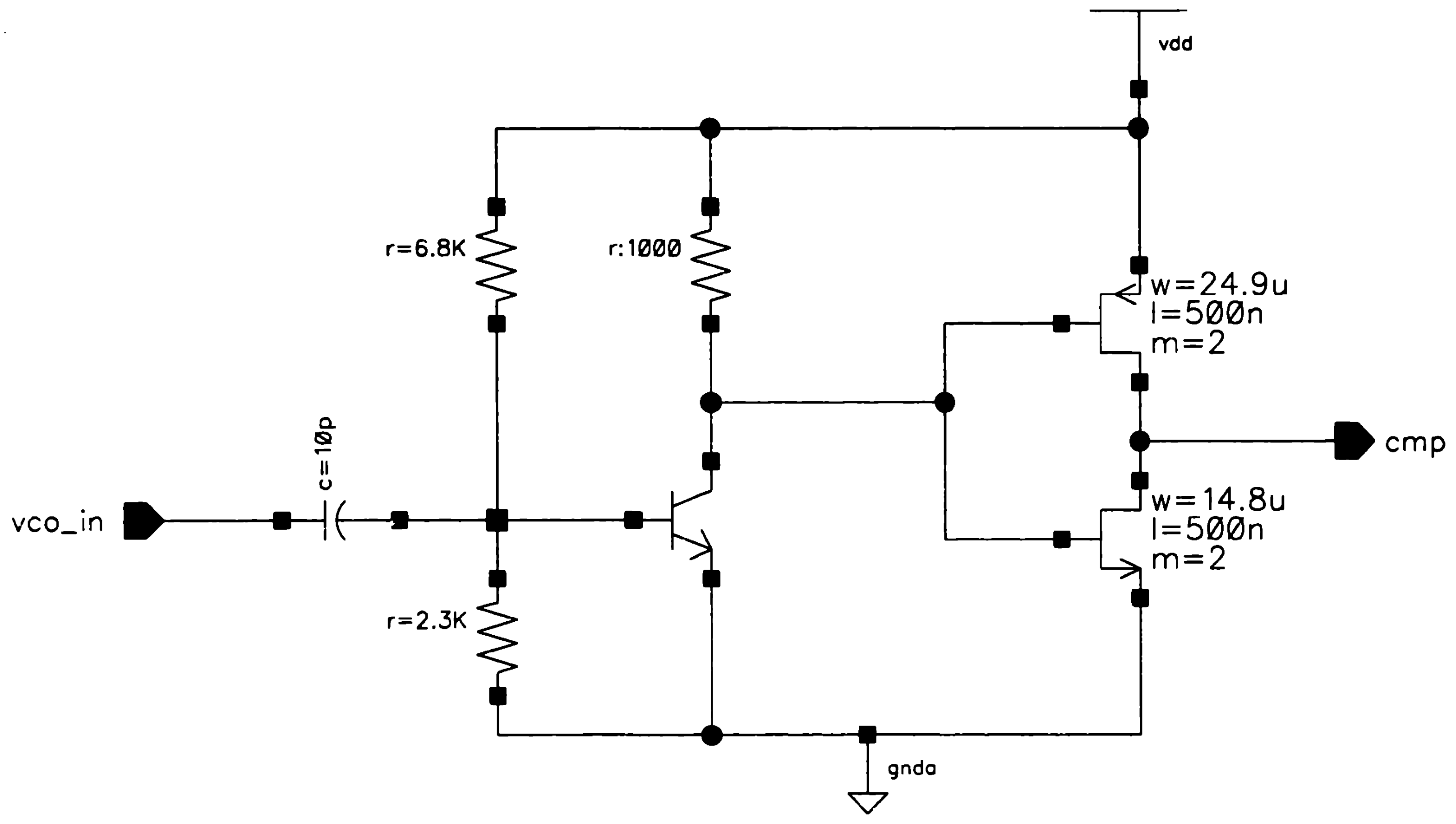

Figure 6-8: Comparator Circuit.

present at the multiplexer output. The output of this multiplexer is the input to a circuit which sets or presets a given bit (depending on the output of the multiplexer), when the counter reaches zero.

The counter is implemented in a synchronous pipe-lined manner, because, at the needed speed of operation, a ripple through couuter has too large a latency. The counter operates by having each bit keep track of the state of all the bits less significant than it. When all of these less significant bits are 0, the given bit toggles its state. When all the bits are 0, the counter reloads itself with the value at the output of the multiplexer set by the *current* and *next* registers. In figure 6-9, one cell of the divider is shown. This cell is further separated in figures 6-10 and 6-11. The full divider is shown in figure 6-12.

In the full divider notice, that the circuit has only a 9-bit counter. The schematic shown can be scaled up to larger sizes without an effect on the function of the circuit. Care must be taken to buffer any signals that have too large a fan out.

6.5 Power Consumption

Estimates of the power consumed by each circuit block were readily available by running each circuit through a simulation at typical operating parameters. The average product of the voltage and current, drawn by the circuit in the simulation, is entered in the following table. The simulator used to generate these results is Spectre.

Circuit	Power Consumed (mWatts)
Phase Discriminator (Analog)	$1.65x10^{-2}$
Phase Discriminator (Digital)	$5.47x10^{-1}$
Low Pass Filter	$5.87x10^{-3}$
DAC	10.9
Divider (Analog)	12.3
Divider (Digital)	182

Table 6.1: Maximum average power consumed by each block.

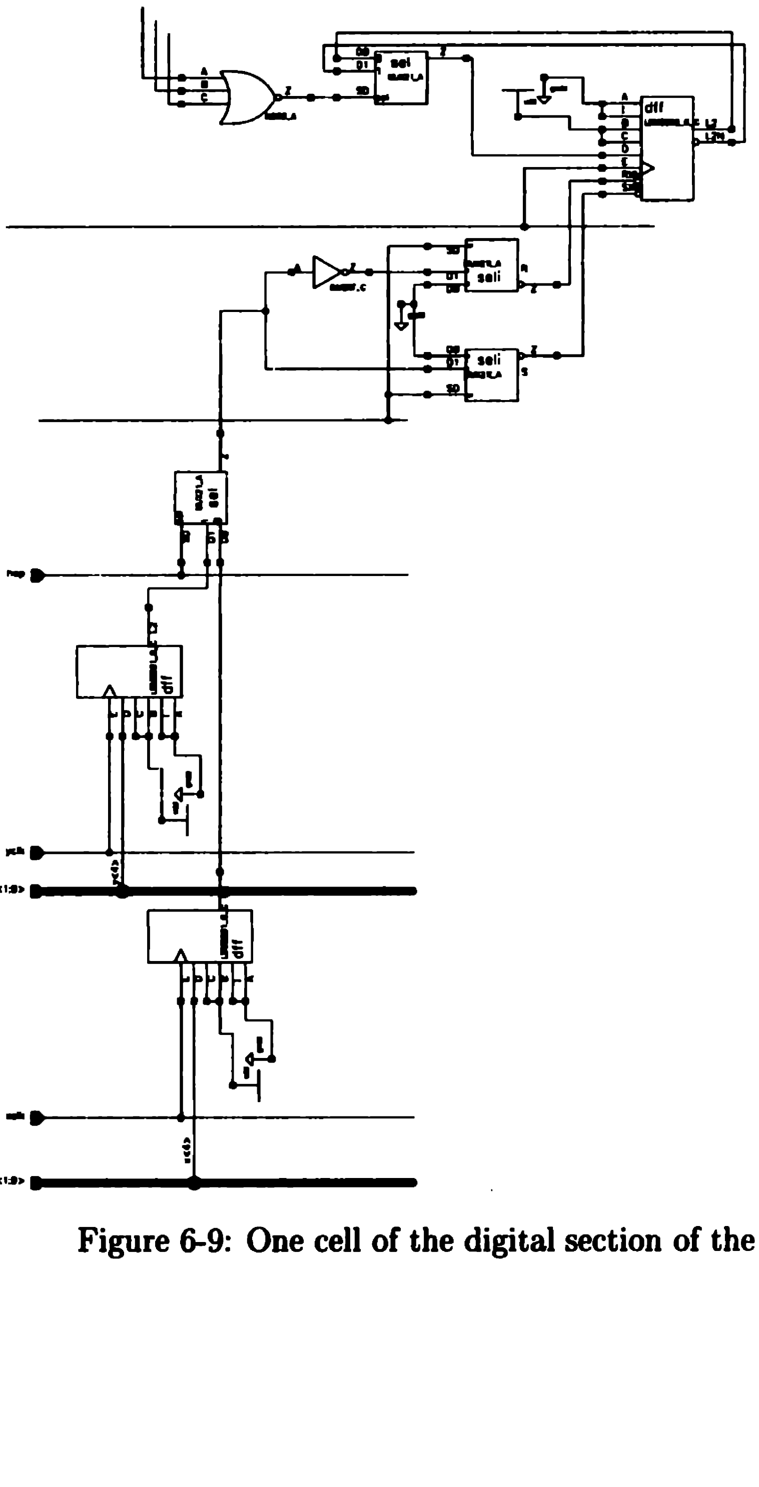

Figure 6-9: One cell of the digital section of the divider.

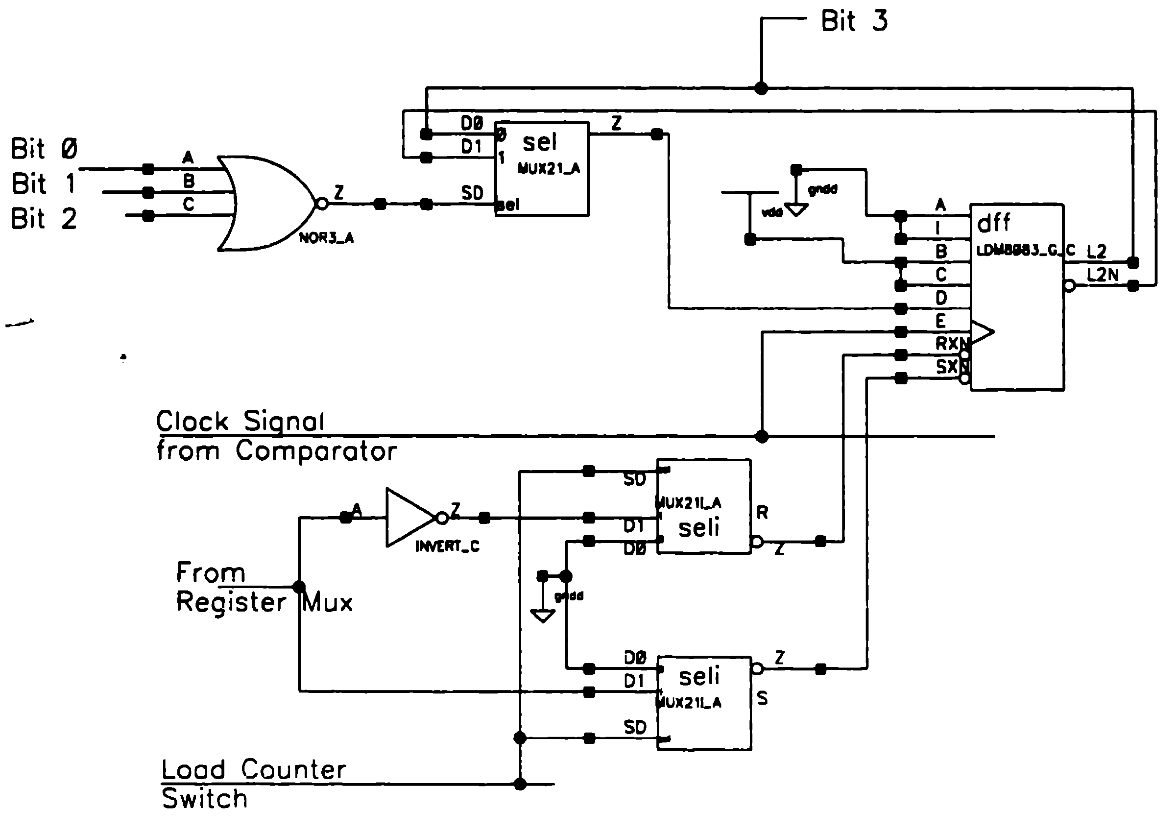

Figure 6-10: One cell of the counter in the digital divider circuit.

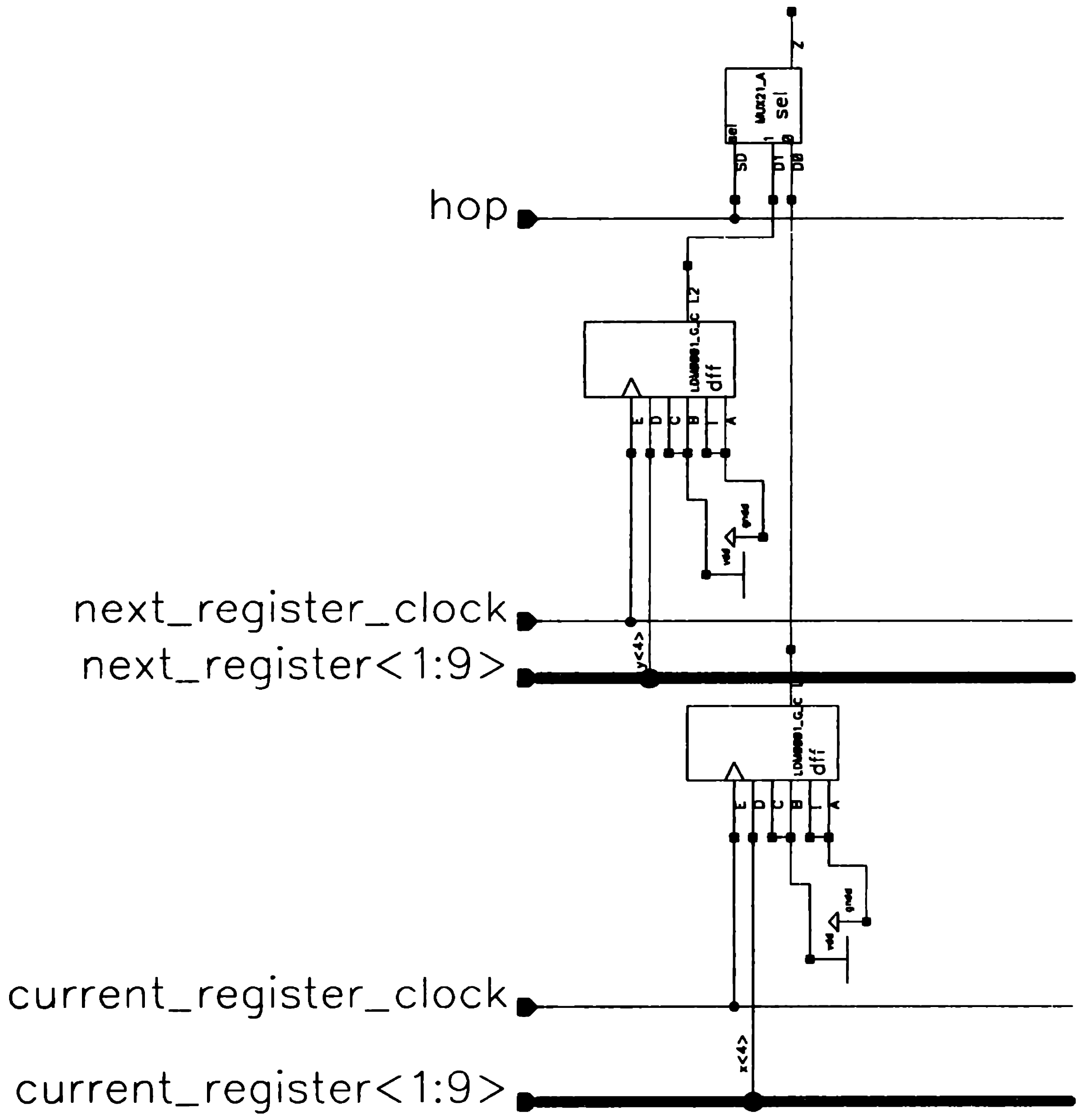

Figure 6-11: One cell of the register interface of the divider circuit.

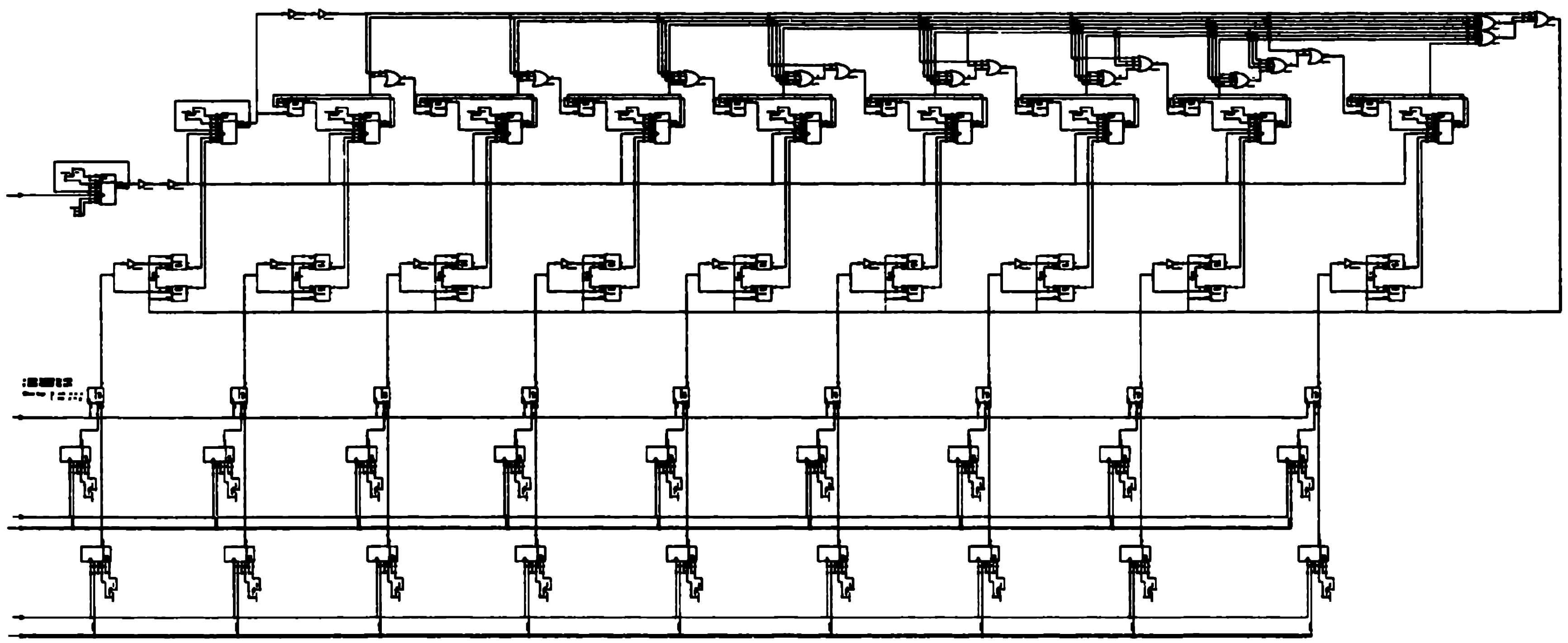

Figure 6-12: Full schematic of digital divider.

6.6 Frequency Synthesizer Operation

The circuit blocks for the phase discriminator and the low-pass filter were connected together along with a functional model of a VCO and a functional model of the divider, to validate the proposed solution discussed in this thesis.

The entire synthesizer was simulated, using the circuit simulator Spectre, which was used previously to characterize and verify the performance of each of the building blocks. The following plot in figure 6-13 demonstrates the operation of the synthesizer in steady state. Included, are plots of the reference clock, the divider output, the phase discriminator output, the filter output, and the VCO output frequency error are all vs. time.

In the next plot (figure 6-14) the dynamic behavior of the synthesizer is demonstrated. A frequency hop from 2.402GHz to 2.485GHz is commanded, but to simulate errors that might manifest at the DAC or the corresponding look-up table, the DAC only steps the control voltage only as far as 2.483GHz. The cut-off frequency of the low-pass filter in this circuit is set at 500KHz.

In figure 6-15, the same circuit is simulated again with the low-pass filter cut-off frequency set at 100KHz.

In the last figure, 6-16, the same circuit is once again simulated with the low-pass filter cut-off frequency set at 10KHz.

From the three transient responses of this circuit, what can be garnered is the effect of changing bandwidth on the settling time of the loop. In the first two instances, frequency settling is observed to be well within the 300μsec limit for the 802.11 IEEE specifications. On the third however, this does not seem to be a likely possibility. Most importantly, however, it should be noted that this represents the worst case performance of the synthesizer. Frequency hops that are smaller will settle in much smaller amounts of time.

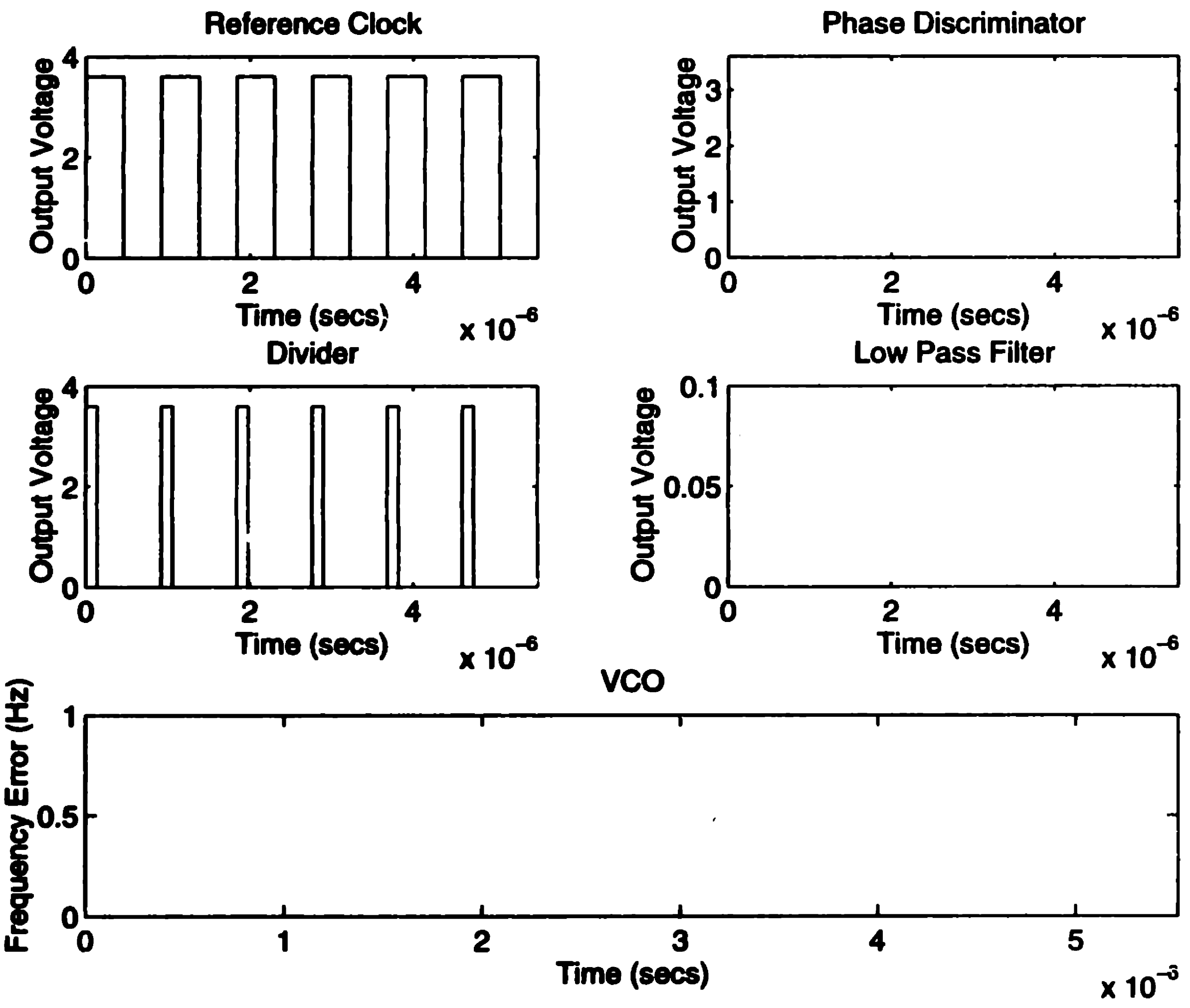

Figure 6-13: Frequency synthesizer circuit response in steady state.

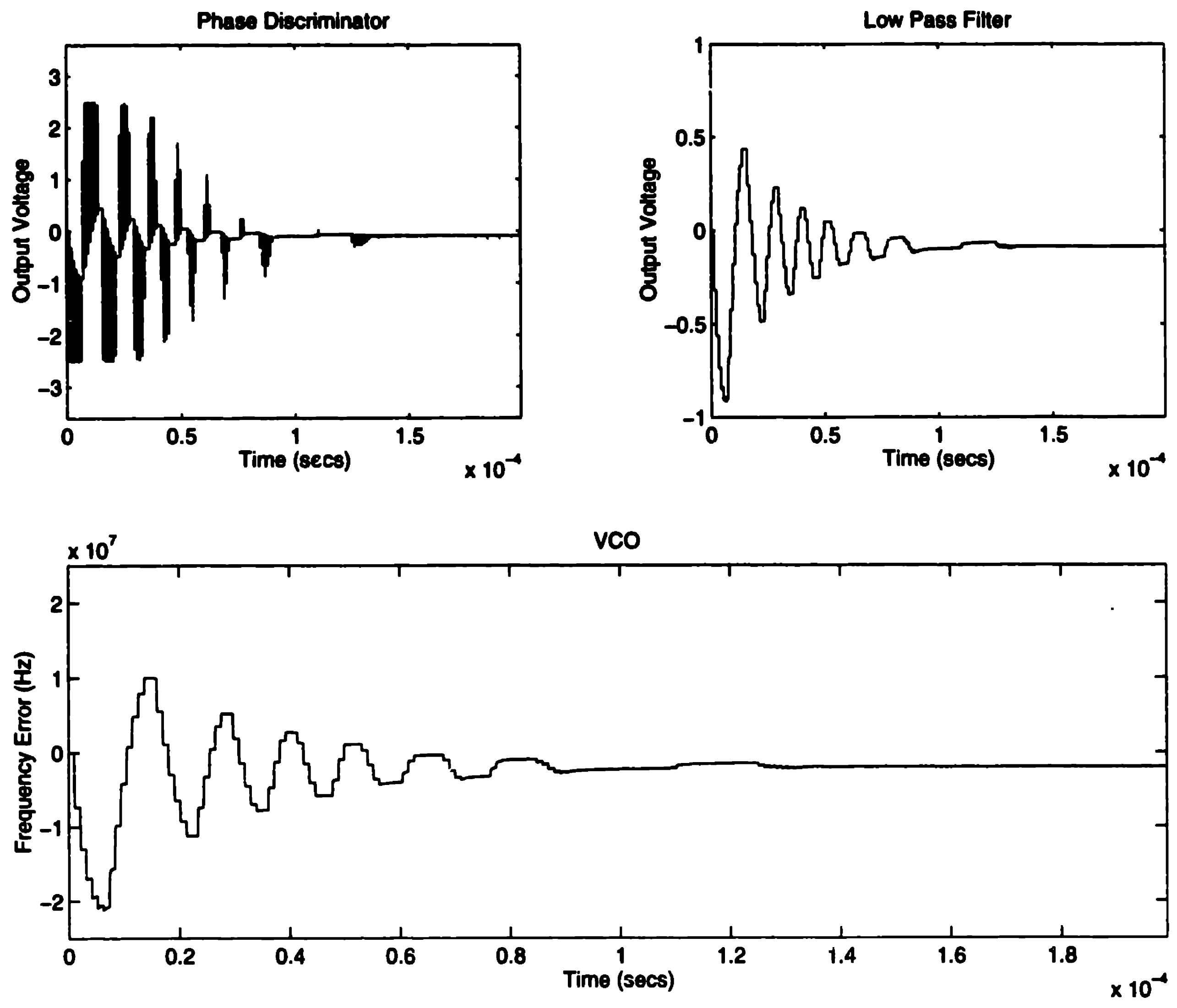

Figure 6-14: Frequency synthesizer circuit response to frequency hop (f_{cf} = 500KHz).

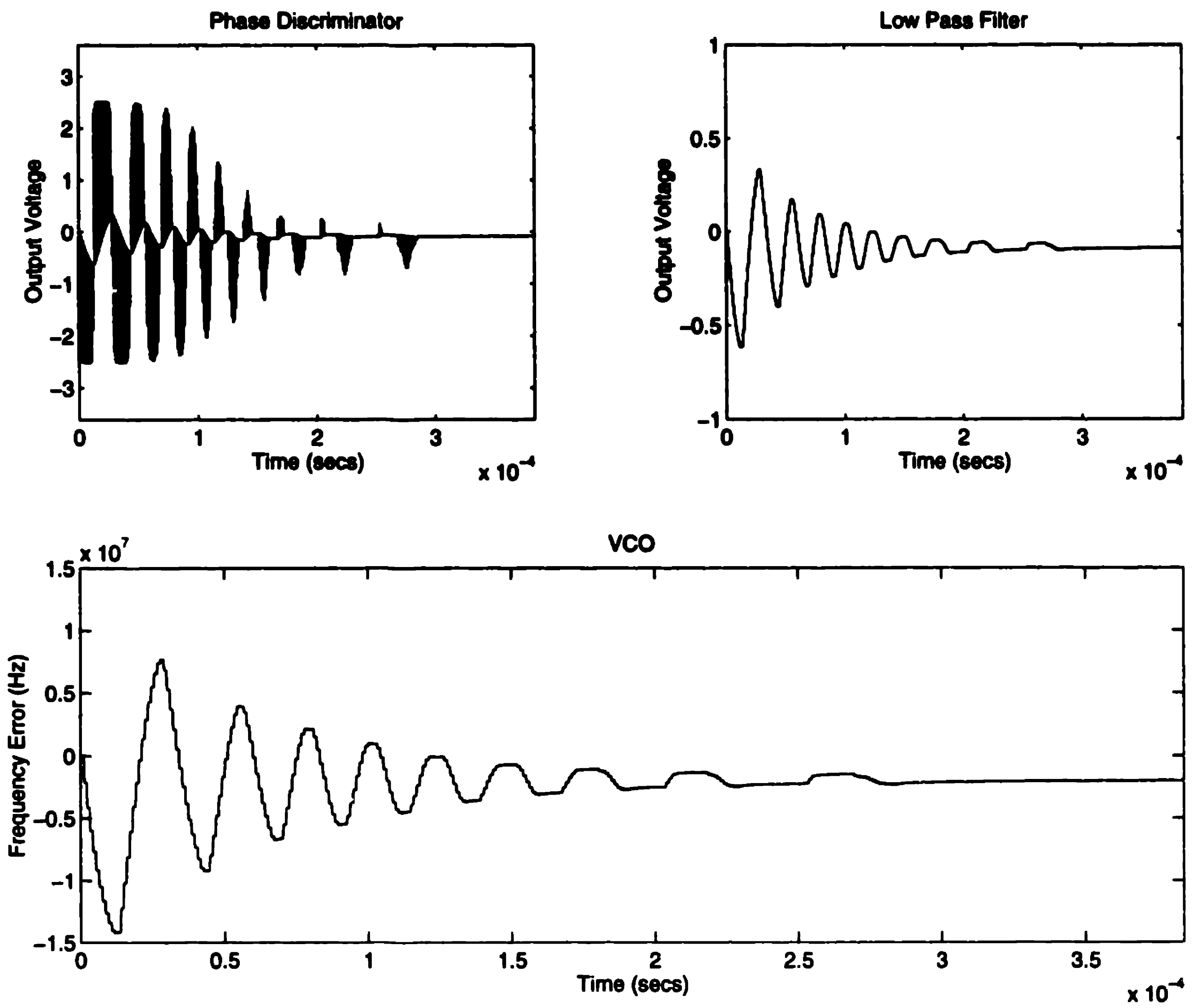

Figure 6-15: Frequency synthesizer circuit response to frequency hop (f_{cf} = 100KHz).

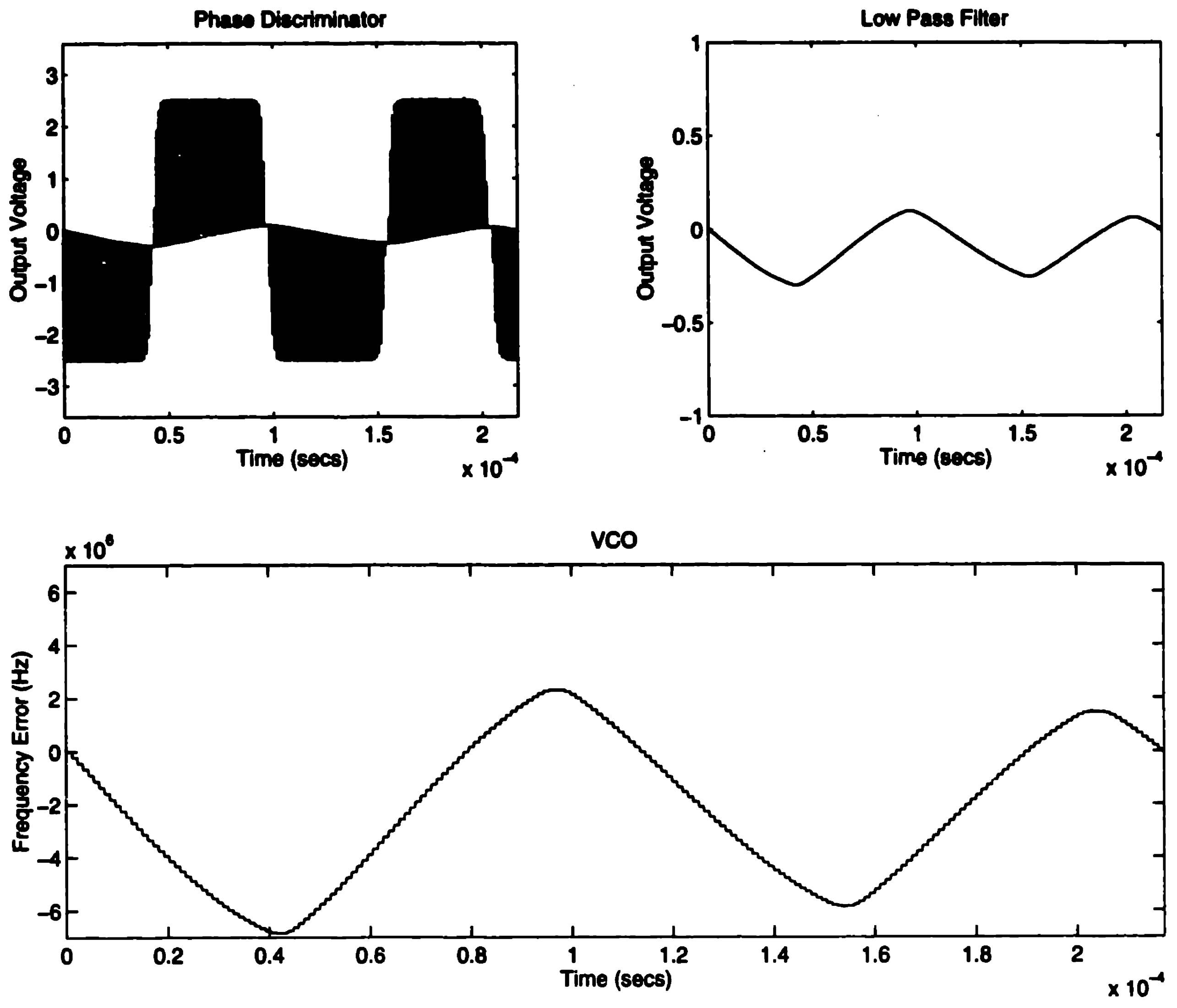

Figure 6-16: Frequency synthesizer circuit response to frequency hop (f_{cf} = 10KHz).

Chapter 7

Conclusion

Given the variety of synthesizer topologies available, the goal of this thesis was to develop a fast slewing synthesizer and demonstrate this capability within the 2.4GHz ISM frequency band.

Compared with the typical Fractional-N topology, where an error signal is used to generate a shift in output frequency, the solution in this thesis combines the methodology of direct frequency synthesis with the benefits of indirect frequency synthesis and demonstrates a viable alternative solution.

While a variety of synthesizer topologies were investigated, a detailed analysis of phase lock loop and synthesizer performance, focused the research on optimizing loop performance via the divider.

The resulting solution required an external DAC to provide the appropriate bias for each frequency, while the loop was used as a fine tuning mechanism. This required the redesign of a key component of the synthesizer – the divider. While a couple of solutions were considered, one was finally settled upon for its accuracy and its practicality - the presetable divider using a delayed preset strategy.

The alternate divider topology – the VSSD divider – which was not implemented, could in fact offer an increase in performance over the presetable divider. This is clearly demonstrated when you compare the figures in Chapter 5, figure 4-6 and figure 5-5.

While the VSSD topology is the ideal solution, it is impractical at present to implement. This research chose a simple divider topology to implement. This was done in part to validate the overall method for achieving better loop performance. A good direction for future work would involve finding a divider topology which provides better accuracy than the delay preset divider while still being more practical to implement than the VSSD topology.

While the entire circuit was not implemented because of the limitations of the current CMOS technology, these circuits could be implemented using bipolar or ECL technology.

Further work in these directions may lead to topologies that have even shorter settling times.

Appendix A

Code

A.1 Variable Step Size Divider

delta_table_period

```
% Calculate the delta step for each frequency
%
% ----------------------------------------

bits_array = 0;                          % sets counter for unique step size
                                         %       bit values.

extra_pulse = zeros(1,84);               % defines array to determnine at WHICH
                                         %       frequencies an extra pulse occurs.

extra_pulses = 0;
ppm = 1000/1e6;                          % jitter fraction

%for joe = -500:1:500
%       ppm = joe/1e6;
for bits = 24:1:24                       % this loops is cycled through for
%       bits = blah;                     %       for various bit size counters
                                         %       Min with a 1 Mhz ref clock
                                         %       is 12 -> 4096 counts.
                                         %       Used to determine for which counter
                                         %       bit sizes will there be unique
                                         %       increment_by steps.

        a = 0;                           % Variable to keep track
                                         %       of whether increment_by
                                         %       array is unique.
                                         %       a = 0 -> unique.
                                         %       a!= 0 -> not unique.

        extra_pulses = 0;                % counter for NUMBER of
                                         %       frequencies at which
                                         %       an extra_pulse occurs.
```

```
result_size = 2^bits;               % counter size
ref_per = 1/1e6;                    % ref frequency
vco_freq = [2.402e9:1e6:2.485e9];% vco frequency array
vco_per  = 1./vco_freq;             % vco period array
vco_per_jitterp    = vco_per + ppm.*vco_per;% vco period +25ppm jitter
vco_per_jitterm = vco_per - ppm.*vco_per;% vco period -25ppm jitter

number_cnts = ref_per./vco_per; % To measure effects of
                                %       jitter in vco output
                                %       change:  vco_per
                                %       to: vco_per_jitterp
                                %       or: vco_per_jitterm

increment_by = result_size./(ref_per./vco_per);        % increment step
                                                       %       IDEAL

increment_by = ceil(increment_by) - 1;                 % increment step
                                                       %       REAL

pulse_count = increment_by.*number_cnts;               % counter value
                                                       %       after corresp
                                                       %       number of cnts.

pulse_count_e = pulse_count - result_size*ones(1,84);        % error between
                                                       %       expected value
                                                       %       of counter and
                                                       %       actual value.

er_percentage = pulse_count_er./increment_by;          % percentage of
                                                       %       error from
                                                       %       increment_by

for n = 1:1:84                                         % Loop determines
        if(er_percentage(n) < 0)                       %       freq at which
        extra_pulse(n) = ceil(er_percentage(n));%              extra pulse
```

```
                extra_pulses = extra_pulses + 1;        %       occurs and
                                                        %       total number
                else extra_pulse(n) = 0;                %       of such freqs.
                end

                if(er_percentage(n) > 0)
                extra_pulse(n) = ceil(er_percentage(n));
                end
        end


        for n = 1:1:83

                if(increment_by(n) == increment_by(n+1))% Loop determines
                a = a + 1;                              %       uniqueness
                end                                     %       of increment_by
                                                        %       array.
        end

        if( a == 0)                                     % Test to see if
        bits_array = [bits bits_array];                 %       increment_by
        end                                             %       array unique.

end

%pulsyn(blah-22,joe+501) = sum(abs(extra_pulse));

%end

%end
%bits_array
```

delta_table_period2

```
% To to generate a unique set of step sizes
%   all of which are k*2^n where k and n
%   can be different for each frequency
%
%--------------------------------------------


freq            = 2402:1:2485;
constant        = 11532254;
n                       = 8;
integer_list = 1.1;

while(any(integer_list ~= floor(integer_list)))

        if min(integer_list) < 1
                constant        = constant + 1;
                n                       = 0;
        else
                n                       = n + 1;
        end

        integer_list = constant./freq./2^n;

end
```

pwr_calc

```
% Try to view adder state in steady state to
%         measure power consumption.
%
%----------------------------------------------

for freq_idx = 1:1:84

for n = 1:1:numb~r_cnts(freq_idx)

          adder_state(n) = n*increment_by(freq_idx);

end

          adder_state = adder_state';

for n = 1:1:number_cnts(freq_idx)

          for m = 23:-1:1

                    toggle_map(n,m)  =        floor(adder_state(n)/2^(m-1));
                    adder_state(n)   =        adder_state(n) - toggle_map(n,m)*2^(m-1);

          end

end

for m = 23:-1:1

          toggles(m)          = 0;

          for n = 1:1:number_cnts(freq_idx)-1

                    if(toggle_map(n,m) ~= toggle_map(n+1,m))
                              toggles(m)          = toggles(m) + 1;
```

```
                else toggles(m)    = toggles(m);

                end

            end

            fmap_bittoggles(freq_idx,m)         =        toggles(m);
        end

        toggles_per_freq(freq_idx)          =        sum(toggles);

        freq_idx

        end

        mean(toggles_per_freq)
```

A.2 Preset Divider

preset_div

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>

#define pi 3.1415926

FILE *fpi, *fpo;

main()
{

  int counter = 0x962;
  int old_counter, counter_index = 1;
  int ci_cur, n, sample_size, counter_array[100];
  int trial;
  float ph_cur, ph_nxt , phase_error[10000];
  float phrase1, phrase2;

  char p[9];

  time_t *tp;



  srandom(time(tp));

  counter_array[0] = 0;

  for(n = 1; n <= 100; n = n + 1)
   {
    counter_array[n] = 0x962 + (n - 1);
   }
```

```
srandom(time);

for(trial = 0x0; trial <= 0x12c; trial = trial + 1)
 {
  sprintf(p,"pdist%x",trial);
  printf("%s\n",p);
  fpi = fopen(p,"w");
  sample_size = 0;

  while(sample_size < 10000)
     {
      if((counter == trial) && (phrase1 == 1))
       {
        counter = counter_array[counter_index];
        phrase1 = 0;
       }
      else if(counter == 0x0)
       counter = counter_array[counter_index];
      else
       counter = counter - 1;

      if(random()%50000 != 1.0)
       sample_size = sample_size;
      else
       {
        sample_size = sample_size + 1;
        ph_cur = ((float) counter/counter_array[counter_index]);
        ci_cur = counter_index;
        old_counter = counter;
        counter_index = ((int) (1 + 83.0*(1 + (random()%100))/100));
        phrase1 = 1;
        phrase2 = old_counter;
        counter = ((int) phrase2);

        ph_nxt = ((float) counter/counter_array[counter_index]);
```

```
            phase_error[sample_size] = 2*pi*(ph_cur - ph_nxt);
            fprintf(fpi,"%f\t%d\t%f\t%d\t%f\t%f\n",phase_error[sample_size], counter, phrase1,old_counter, ph_cu

            /*
               printf("%f\t%d\n",phase_error[sample_size], counter);
               */

           }
         }

      fclose(fpi);

    }
  }
```

A.3 Spectre AHDL Models

VCO

```
module ideal_vco(Vvco, Vp, Vhlf, Vlpf_p, Vlpf_n, Vdac)(fmin,pi,fgain)
   node [V,I] Vvco;
   node [V,I] Vp;
   node [V,I] Vhlf;
   node [V,I] Vlpf_p;
   node [V,I] Vlpf_n;
   node [V,I] Vdac;

   parameter real fmin = 2.402e9;
   parameter real pi = 3.1415926;
   parameter real fgain = .2777778*83e6;

{
         real p;

         analog{

                  p = fgain*(V(Vlpf_p) - V(Vlpf_n));

                  V(Vvco) <- .150*cos(2*pi*fmin*$time() + 2*pi*integ(fgain*V(Vdac),0) + 2*pi*integ(p,0));

                  V(Vp)   <- p*1e-12;

                  V(Vhlf) <- .150*cos(.125*(2*pi*fmin*$time() + 2*pi*integ(p,0)));

                  }
}
```

Divider

```
module ideal_divider(Vdiv, Vvco, hop)(current_freq,next_freq)
   node [V,I] Vdiv;
   node [V,I] Vvco;
   node [V,I] hop;


         parameter integer current_freq = 2402;
         parameter integer next_freq = 2483;


{

         integer counter, current_reg;


         initial {
                    counter = 2;
                   }



         analog{

                  if (V(hop) > 1.8)
                           {
                                    current_reg = next_freq;
                           }
                  else
                           {
                                    current_reg = current_freq;
                           }



                  if ($threshold( V(Vvco), 1))
                           {
                                    counter = counter - 1;
                           }
                  else
```

```
                {
                        counter = counter;
                }


        if (counter == 0)
                {
                        counter = current_reg;
                }
        else
                {
                        counter = counter;
                }

        if (counter < 2048)
                {
                        V(Vdiv) <- 0;
                }
        else
                {
                        V(Vdiv) <- 3.6;
                }

    }
}
```

Appendix B

Simulation Data

B.1 Variable Step Size Divider - *step size* tables

Frequency (GHz)	*step size* Register Value
2.402	3492
2.403	3490
2.404	3489
2.405	3487
2.406	3486
2.407	3485
2.408	3483
2.409	3482
2.410	3480
2.411	3479
2.412	3477
2.413	3476
2.414	3474
2.415	3473
2.416	3472
2.417	3470
2.418	3469
2.419	3467
2.420	3466
2.421	3464
2.422	3463
2.423	3462
2.424	3460
2.425	3459
2.426	3457
2.427	3456
2.428	3454
2.429	3453
2.430	3452
2.431	3450
2.402	3449
2.433	3447
2.434	3446
2.435	3445
2.406	3443
2.437	3442
2.438	3440
2.439	3439
2.440	3437
2.441	3436
2.442	3435
2.443	3433

Table B.1: Step sizes for a 23-bit result register (part 1).

Frequency (GHz)	*step size* Register Value
2.444	3432
2.445	3430
2.446	3429
2.447	3428
2.448	3426
2.449	3425
2.450	3423
2.451	3422
2.452	3421
2.453	3419
2.454	3418
2.455	3416
2.456	3415
2.457	3414
2.458	3412
2.459	3411
2.460	3410
2.461	3408
2.462	3407
2.463	3405
2.464	3404
2.465	3403
2.466	3401
2.467	3400
2.408	3398
2.469	3397
2.470	3396
2.471	3394
2.472	3393
2.773	3392
2.474	3390
2.475	3389
2.476	3387
2.477	3386
2.478	3385
2.479	3383
2.480	3382
2.481	3381
2.482	3379
2.483	3378
2.484	3377
2.405	3375

Table B.2: Step sizes for a 23-bit result register (part 2).

Frequency (GHz)	*step size* Register Value
2.402	6984
2.403	6981
2.404	6978
2.405	6975
2.406	6973
2.407	6970
2.408	6967
2.409	6964
2.410	6961
2.411	6958
2.412	6955
2.413	6952
2.414	6949
2.415	6947
2.416	6944
2.417	6941
2.418	6938
2.419	6935
2.420	6932
2.421	6929
2.422	6927
2.423	6924
2.424	6921
2.425	6918
2.426	6915
2.427	6912
2.428	6909
2.429	6907
2.430	6904
2.431	6901
2.432	6898
2.433	6895
2.434	6892
2.435	6890
2.436	6887
2.437	6884
2.438	6881
2.439	6878
2.440	6875
2.441	6873
2.442	6870

Table B.3: Step sizes for a 24-bit result register (part 1).

Frequency (GHz)	*step size* Register Value
2.443	6867
2.444	6864
2.445	6861
2.446	6859
2.447	6856
2.448	6853
2.449	6850
2.450	6847
2.451	6845
2.452	6842
2.453	6839
2.454	6836
2.455	6833
2.456	6831
2.457	6828
2.458	6825
2.459	6822
2.460	6820
2.461	6817
2.462	6814
2.463	6811
2.464	6808
2.465	6806
2.466	6803
2.467	6800
2.468	6797
2.469	6795
2.470	6792
2.471	6789
2.472	6786
2.473	6784
2.474	6781
2.475	6778
2.476	6775
2.477	6773
2.478	6770
2.479	6767
2.480	6765
2.481	6762
2.482	6759
2.483	6756
2.484	6754
2.485	6751

Table B.4: Step sizes for a 24-bit result register (part 2).

Bibliography

[1] N. Silberman, "IEEE 802.11 Wireless Access Methods and Physical Layer Specifications," Draft Proposal for a Frequency Hopping and Direct Sequence Spread Spectrum PHY Standard 2.2, IEEE, November 8 1993.

[2] A. Yamagishi, M. Ishikawa, T. Tsukahara, and S. Date, "A 2-V, 2-GHz Low-Power Direct Digital Frequency Synthesizer Chip Set for Wireless Communication," *Proceedings of the Custom Integrated Circuits Conference*, pp. 319–322, 1995.

[3] T. Saba, D.-K. Park, and S. Mori, "Fast-Acquisition PLL Synthesizer Using a Parallel N-Stage Cycle Swallower with Low Power Consumption and Low Phase Noise," *IEEE Transactions on Vehicular Technology*, vol. 44, pp. 296–303, May 1995.

[4] P. Larsson, "Reduced Pull-in Time of Phase-Locked Loops Using a Simple Nonlinear Phase Dector," *IEEE Proceedings on Communications*, vol. 142, pp. 221–226, August 1995.

[5] W. A. C. Jr., "High-Speed Frequency Synthesizer for Spread Spectrum Communication Systems," *Tactical Communications Conference*, vol. 1, pp. 275–282, May 1994.

[6] J. A. Crawford, *Frequency Synthesizer Design Handbook.* 685 Canton Street, Norwood, MA 02062: Artech House Inc., 1994.

[7] B.-G. Goldberg and H. Eisenson, "Frequency Synthesizer Strategies for Wireless," *Microwave Journal*, vol. 36, pp. 22,26,31,34,36,39,40, June 1993.

[8] K. Seki, M. Mizoguchi, and S. Kato, "Low Power Consumption and Fast Settling Frequency Synthesizer for TDMA-TDD Systems," *Proceedings of the 43rd IEEE Vehicular Technology Conference*, pp. 281–284, 1993.

[9] W. Zhou, "A New Technique of Frequency Synthesis," *Proceedings of the 1993 IEEE International Frequency Control Symposium*, pp. 251–254, October 1993.

[10] A. Kajiwara and M. Nakagawa, "High Speed PLL Frequency Synthesizer for Mobile Communications," *SUPERCOMM/International Conference on Communications*, vol. 1, pp. 486–490, 1992.

[11] S. R. Al-Araji, A. J. Al-Dweik, and M. M. S. Abu-Rajab, "Fast Switching Frequency Synthesizer using Adaptive PLL Operating in the Pseudo Linear Region," *Proceedings of the IASTED International Conference*, pp. 258–261, 1995.

[12] S. Naffziger, "A Sub-Nanosecond 0.5μm 64b Adder Design," *IEEE International Solid-State Circuits Conference*, pp. 362–363,290–291,475–476, 1996.

[13] B. Razavi, ed., *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, pp. 1–40. 345 East 47th Street, New York, NY 10017-2394: The Institute of Electrical and Electronics Engineers, Inc., 1996.

[14] M. Leonard, "PCMCIA-sized Radio Links Portable WLAN Terminals," *Electronic Design*, pp. 45,46,48,50, August 5 1993.

[15] M. Donlin, "An Ocean's Not a Problem for a Wireless LAN Team," *Computer Design*, pp. 89,90,92, November 1993.

[16] J. L. Jim Dunning, Gerald Garcia and E. Nuckolls, "An All-Digital Phase-Locked Loop with 50-Cycle Lock Time Suitable for High-Performance Microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 412–422, April 1995.

[17] T. Iritani, J. Kuge, and T. Oie, "Fast Convergence PLL Synthesizer with Initial Phase Difference," *Electronics and Communications in Japan*, vol. 78, pp. 79–89, July 1995.

[18] K. Ishii, T. Yamamoto, M. Shigaki, H. Hongo, and M. Iwatsuki, "An On-board 47-GHz Phase-Locked Oscillator Using FET Gate Bias Control for Low Phase Noise," *23rd European Microwave Conference Proceedings*, vol. 1, pp. 767–769, September 1993.

[19] J. Kuge, T. Iritani, and T. Oie, "Fast Hopping Frequency Synthesizer with Small Frequency Error," *IEEE International Conference on Selected Topics in Wireless Communications. Conference Proceedings*, pp. 215–218, 1992.

[20] D. E. Phillips, "PLL Settling Time: Phase VS Frequency," *Tactical Communications Conference - Proceedings*, vol. 1, pp. 283–288, 1994.

[21] K. Itoh and A. Iida, "Wideband Phase-Locked Loop Synthesizer Using Linear Frequency Variation Direct Digital Synthesizer as Reference Oscillator," *Electronics and Communications in Japan*, vol. 78, pp. 79–90, September 1995.

[22] T. J. Endres, R. B. Hall, and A. M. Lopez, "Design and Analysis Methods of a DDS-Based Synthesizer for Military Spaceborne Applications," *IEEE International Frequency Control Symposium*, pp. 624–632, 1994.