NTNU

Norwegian University of
Science and Technology

Design of a Linear FMCW Radar Synthesizer with Focus on Phase Noise

Ruben Undheim

Master of Science in Electronics
Submission date: February 2012
Supervisor: Egil Eide, IET
Co-supervisor: Harald Rosshaug, Kongsberg Seatex AS

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Student: Ruben Undheim

Title: Design of a Linear FMCW Radar Synthesizer With Focus on Phase Noise

Text:
Phase noise is the most important performance parameter in phase locked synthesizers and local oscillators of communication equipment and radars. In this exercise, a synthesizer will be constructed which will be used in a range/bearing measurement system based on FMCW radar principles.

Tasks:

- Theoretical study of phase noise in phase locked loops
- Analysis of the importance of the phase noise for the performance of a FMCW based range/bearing measurement system
- A survey of alternative topologies for the synthesizer of an FMCW-based range/bearing measurement system. Possible frequency bands are: 5.46 GHz - 5.64 GHz and 9.2 GHz - 9.3GHz
- Construction of synthesizer based on chosen topology

External supervisor: Harald Rosshaug, Kongsberg Seatex AS

Internal supervisor: Egil Eide

Abstract

The linear FMCW radar has become more and more popular in recent years mainly due to advances in digital signal processing and the good performance of the radar at close ranges. What puts limits to the performance is mainly phase noise. Because transmission and reception happen simultaneously, the phase noise will limit the maximum power that should be used and hence also the ability to detect weak targets. By ensuring during the design process that the phase noise is low, the radar's performance will thus get better. This thesis describes the construction of a FMCW radar frequency synthesizer where the focus is mainly on phase noise. The functionality of the circuit is shown to be successful, but there is more phase noise than what is predicted. Several causes for this are discussed. Important background theory about radars, phase noise and phase-locked loops is presented and several simulations are performed in order to get a better understanding. The conclusion of the work is that it is not very hard to build a synthesizer, but in order to tweak the phase noise performance to be as good as the linear theory tells it to be, careful attention must be paid during all stages of the design.

Preface

This thesis concludes my Master Degree in Electronics Engineering at the Norwegian University of Science and Technology (NTNU). It was carried out during the Winter of 2011/2012 and submitted to NTNU, Trondheim February 20th, 2012. The focus is mainly on how a synthesizer for an FMCW radar can be built, what its performance issues may be, and then getting a thoroughly understanding of phase noise, the PLL and radar systems in general. It involves general circuit design, RF circuit design, PCB layout, microcontroller programming, simulations, soldering, measurements etc.

I would like to thank Kongsberg Seatex AS for letting me work there and for funding the project, and especially my supervisor, Harald Rosshaug, for the assistance I have received during the work with the thesis.

I would also like to thank Egil Eide at NTNU for the questions he has helped me clear out.

Trondheim, February 20th, 2012
Ruben Undheim

Contents

Glossary

AM: Amplitude Modulation

CW: Continuous Wave

DAC: Digital to Analogue Converter

DDS: Direct Digital Synthesis

DUT: Device Under Test

FFT: Fast Fourier Transform

FM: Frequency Modulation

FMCW: Frequency Modulated Continuous Wave

IF: Intermediate Frequency

ISF: Impulse Sensitivity Function

LO: Local Oscillator

LTV: Linear, Time-Variant

MCU: Microcontroller Unit

OCXO: Oven-Controlled Crystal Oscillator

PCB: Printed Circuit Board

PD: Phase Detector

PFD: Phase/Frequency Detector

PI: Proportional-Integral (A regulator)

PLL: Phase Locked Loop

PRF: Pulse Repetition Frequency

PRI: Pulse Repetition Interval

RBW: Resolution Bandwidth

RPC: Reflected Power Canceller

SFDR: Spurious-free Dynamic Range

SMA: SubMiniature version A (coaxial connector)

SSB: Single Sideband

STC: Sensitivity Time Control

TCXO: Temperature-Compensated Crystal Oscillator

VBW: Video Bandwidth

VCO: Voltage Controlled Oscillator

Chapter 1

Introduction

In recent years, there has been a growing interest in the linear FMCW radar [1]. It has some properties that are not seen with the traditional pulse radar or its variety, the pulse-Doppler radar - the main advantage being higher performance at close ranges. This is very useful for radio navigation systems that are used to navigate near obstacles. A typical example is marine vessels. They are often relatively big, and the captain will often struggle to keep every part of the vessel under observation when navigating in narrow areas or in bad weather.

As in other fields of electronics and signal engineering, noise is an important issue. For the FMCW radar, the main worry is more specifically phase noise. This category of noise appears in all systems which employ oscillators. Because transmission and reception happen simultaneously in an FMCW radar, the phase noise will limit the maximum power that it makes sense to use and hence also the ability to detect weak targets. The fascinating and disturbing thing with phase noise is that it is very hard to get rid of when it first has appeared. Therefore it is of high importance to ensure that the oscillators in the circuit are stable and show good phase noise performance at many offset frequencies. Phase noise is not only important in radar systems but also in communication systems. It is the major contributor to undesired phenomena such as interchannel interference, leading to increased bit error rates. In radar- and communication technology, phase noise is regarded in the frequency domain, but in other fields, like digital design, it is regarded in the time domain where it is called jitter. Even here it may be of critical importance.

The aim of the thesis is to get an understanding of radar systems, investigate how they can be implemented and what put limits to their capability. Furthermore, a frequency synthesizer will be implemented which will be used in a linear FMCW radar in the frequency band of 9.2-9.3 GHz. It must provide a very clean sweep in this frequency range and therefore phase noise will be one main consideration. Different theories have appeared during the last decades which attempt to explain what phase noise is and how it appears [2, 3, 4]. There is still not complete agreement on these issues, and it therefore remains somewhat a mystery.

To be able to design a synthesizer for the FMCW radar, knowledge of many kinds of circuits and devices is required. Two circuits that are particularly much used

nowadays are the Phase-locked loop (PLL) and the Direct Digital Synthesizer (DDS). The PLL can be considered a linear control theory system when it is in lock, so that the analysis can be simplified. It consists of several important parts that need to be well understood. The DDS on the other hand, is mostly a digital circuit connected to the world with an on-chip DAC.

The following three chapters present radars, phase noise and phase-locked loops. Further Chapter 5 presents a discussion and some simulations of the degradation effects that appear in FMCW radars. Chapter 6 discusses different topologies that can be used for the design, while Chapter 7 presents the actual design. The thesis sums up with the measurement results of the radar synthesizer and a discussion.

Chapter 2

Radars

Radars use radio waves to estimate the distance, bearing and velocity of *something*. Something can either be one target, multiple targets or all targets in view as for imaging radars.

There are two main-types - the pulse radar and the continuous wave (CW) radar. They both have their advantages. In the beginning of the days of radio technology, the CW radar was the only one that was successfully made. [5] A single oscillator running at a constant frequency could be used to transmit a carrier and the reflected signal from a moving target would then result in a received Doppler shift[1]. This would then be proportional to the velocity of the target. It was first discovered accidentally when big ships near receivers caused changes to the received signal from a distant transmitter. This was a bistatic[2] radar, but later the concept was employed on purpose and the first real radar appeared. With this technology, it was not possible to measure the distance to the target, but moving targets could more or less easily be detected.

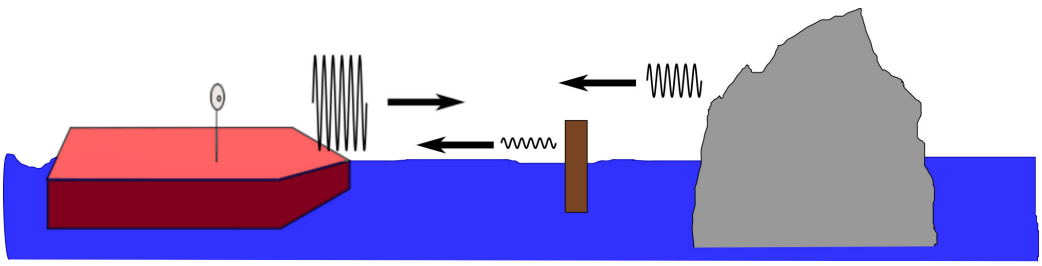


Figure 2.1: *How marine vessels can use radars. In this case a pulse radar.*

Later, when the technology allowed it, the pulse radar was made. Its principle is easy to understand, but at that time, it was not that easy to implement. It was necessary to transmit very short pulses of a clean radio signal, and then be completely silent for a while in order to receive the echo. The development of the pulse radar

[1]This is a change in frequency $\Delta f = 2v/\lambda$ for targets moving with a relative speed v

[2]Bistatic means that the transmitter and the receiver are at different locations as opposed to the normal monostatic radar.

advanced rapidly when the magnetron was invented in 1940 [6]. Much of the major development happened during World War II, when the different countries realized the benefit of detecting airplanes. The pulse radar was able to measure the distance to the targets where the CW radar failed. As time went on, and electronics became more advanced, good pulse radars were developed with very high performance and the continuous wave radars were almost forgotten. With the arrival of the pulse-Doppler radar, it was possible to measure both range and velocity with one single radar.

Radars can be used for two main applications, namely target detection and imaging. Imaging radars try to make an image of the observed area, and they take all echos into consideration. A prime example of this is the Synthetic Aperture Radar. It is mainly used from space to make a mapping of the earth's surface. A target detection radar tries to find the location of a certain or several objects.

This chapter will first present the traditional pulse radar and then the simple CW Doppler radar before proceeding with the explanation of the FMCW radar which the rest of the text will be about.

2.1 Pulse Radar

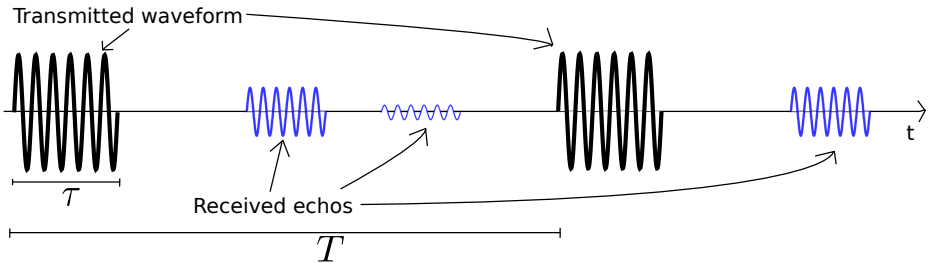


Figure 2.2: *Principle of basic pulse radar*

The basic pulse radar is intuitive to understand. A pulse is transmitted at regular intervals and the received echoes of this pulse define the targets. When we say a pulse, it is actually a sine-wave of very short duration, even though most texts forget to mention and illustrate this. We see that in order to detect and decide the distance to a target far away, the length between the pulses, T(also called PRI), needs to be long enough. The pulse flies at the speed of light, and it has to go back and forth, so the minimum time between each pulse in order to have no ambiguous ranges[3] is given by:

$$T = \frac{2R_{\max}}{c} \tag{2.1}$$

[3]Ambiguous range means that a certain reading on the radar screen can be caused by targets at two different distances quite far apart from each other.

$R_{\max}$ is the maximum range of the radar. Note that in Figure 2.2, the length of τ is overemphasized. In traditional pulse radars, τ is very much smaller than T.

The inverse of T is called the pulse repetition frequency (PRF).

$$PRF = \frac{1}{T} = \frac{c}{2R_{\max}} \tag{2.2}$$

Another fact that can be observed is that the length of each pulse, τ, defines how close targets can be to each other and still distinguish them as two separate targets, and not a big one. This is known as the resolution. It is important to remember that the accuracy of a system is something else than the resolution. The accuracy can be good even though the resolution is bad. The accuracy is mainly degraded by noise. However, the resolution normally influences negatively the way noise degrades the accuracy. So indirectly, bad resolution often causes worse accuracy.

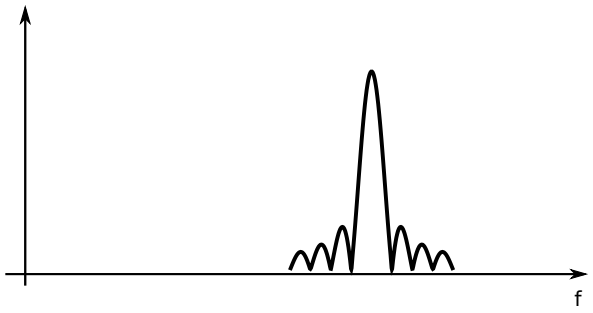


Figure 2.3: *One pulse in frequency domain*

In the frequency domain, a pulse of a sine wave at a certain frequency looks like a sinc-function centered at that frequency. The width of the main-lobe of the sinc-function is defined by the length of the pulse. A longer pulse will have a narrower main-lobe of the sinc-function in frequency domain and will therefore use less bandwidth. There is therefore a relationship between the bandwidth of the pulse and the resolution of the radar.

$$x_r = \frac{c}{2B} \tag{2.3}$$

The spectrum with a single sinc-function is just applicable for the ideal case when only one single pulse is sent. However, when many pulses are sent after each other at regular intervals, the frequency specter will consist of many discrete lines which have an envelope of the sinc-function of the single-pulse case (Figure 2.4). The distance between these discrete lines is given by the PRF. The higher the PRF, the longer distance between every discrete line. More advanced pulse radars, called *pulse-Doppler Radars*, take advantage of the frequency spacing between these discrete lines in order to measure the Doppler shift of the returned pulses. The Doppler shift appears when the target moves relatively to the radar. It is therefore an indicator of velocity. A moving target will shift the entire spectrum either up or down (depending

on whether the target is approaching or not) and the velocity will be measured. In order to measure the velocity it is very important that the radar signal is coherent. That basically means that only the amplifier is turned off between the pulses and not the oscillator. This is hard to do with a magnetron but easy to do with a klystron amplifier. Since the PRF defines the distance between the discrete lines, it will also define how fast targets can be detected without getting an *ambiguous velocity* reading.

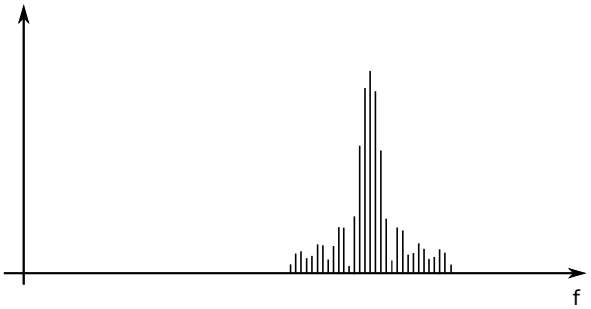


Figure 2.4: *Many pulses in frequency domain*

We therefore have a dilemma, increasing the PRF, improves the velocity ambiguities, while decreasing the PRF, improves the range ambiguities. Because the Doppler shift gets higher if higher carrier frequencies are used, the problem also becomes bigger at higher frequencies. The pulse-Doppler Radar actually often ignores what was said above about that the distance between the pulses must be long enough to achieve no ambiguous range. It makes a compromise between ambiguous range and ambiguous velocity.

There are three different main-types of the pulse-Doppler Radar depending on how high the PRF is [7].

Low PRF: The PRF is kept so low that there will be no range ambiguities but many velocity ambiguities.

Medium PRF: The PRF is kept higher so that there will be both range ambiguities and velocity ambiguities, but not so many of each.

High PRF: The PRF is kept so high that there will be no velocity ambiguities. However, there will be many range ambiguities.

The exact values of the PRFs for the different types above, depend on over which region in range and velocity the radar is supposed to operate.

Many methods have been made to cope with this problem of ambiguities. One solution is to use a variable PRF, often called staggering. It is quite commonly used for high PRF-radars and will solve out the range ambiguities since they will be different for the different PRFs that are used. Normally, at least 3 different PRFs are needed in order to solve all the ambiguities.

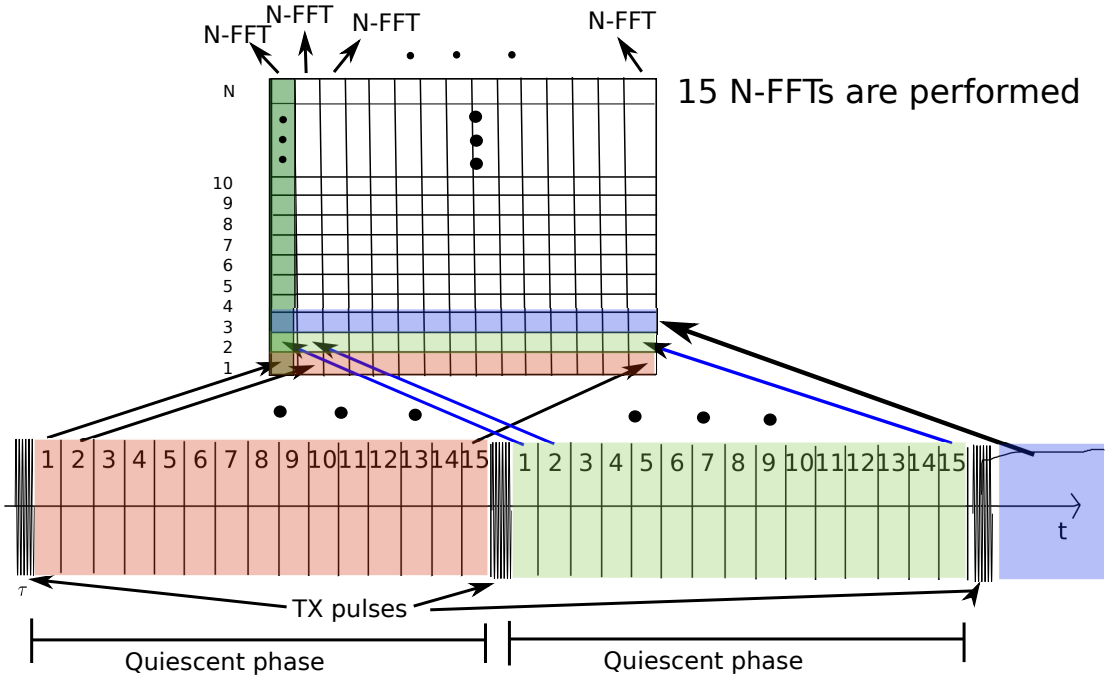


Figure 2.5: *How Pulse-Doppler signal processing can be performed*

2.1.1 Pulse-Doppler signal processing

Pulse-Doppler signal processing is nowadays used for most pulse radars. Here follows a short explanation of how it can be performed [8]. After the radar pulse has been emitted, the receiver monitors the received echoes. The received signal during the time between the pulses is called the quiescent phase. This time is divided into M equally long *bins* corresponding to the desired range resolution - often called range gating in the literature. The received signal is mixed down to an intermediate frequency with a local oscillator whose frequency is derived from the transmitted signal. Following some filters at the IF frequency, the signal is mixed down in quadrature to baseband. One complex sample is thereby taken for each bin and saved in a matrix in memory (see Figure 2.5). All the complex samples for the succeeding bins are saved in the same row in the matrix. When then the next pulse is emitted, the following samples are saved in a new row. After N pulses a full matrix of $M \times N$ complex values are found. An N-FFT is taken for each column in the matrix. The result is multiple Doppler spectra. Each of them correspond to a time bin after one pulse and hence one ambiguous range. If a high PRF is used, each bin corresponds to several distances. The Doppler spectra will reveal the speed of the moving targets in each bin. Before the advent of FFT processors, the processing was performed with analogue filters and the structure was therefore slightly different.

2.2 CW Radar

The CW radar uses a single oscillator running at a constant frequency. By taking advantage of the Doppler shift, moving targets can be detected. The received signal is mixed with the transmitted signal and the *beat signal*[4] will contain the Doppler shifts of the targets. The processing is much simpler than for the pulse-Doppler processing. An FFT can be performed continuosly for the received signal. The main problem is however that since the signal is stationary at a single frequency, the distance to targets can not be found. The radar has been used, and is still used in many different applications. A prime example is the police speed radar. Since the range is not interesting in this case, the basic CW radar does its job well.

One difficulty with the CW radar compared to the pulse radar is that it is receiving at the same time as it is transmitting. This will cause some of the transmitted signal to propagate or leak directly to the receiver. This is certainly the case if the same antenna is used. Two different antennas (quasi-bistatic) are often used but still in this case the leakage of the transmitted signal can be severe. As we will see, this problem is actually the biggest disadvantage of the CW radar, and its relative, the *FMCW radar*, which will be presented now.

2.3 FMCW Radar

The main problem of the CW radar above is that it is unable to measure the distance to the target. In many cases this is exactly what we want. To be able to determine the distance, the signal cannot be stationary. It must change somehow. It is possible to either change the amplitude or the frequency. When the signal then returns from the target the distance can be calculated since it is known how the transmitted signal looks like at all times. Changes of the amplitude are indeed very hard to measure because they are easily overloaded by the transmitted signal, and there is no way to filter out the returned signal. In addition, the amplitude of the returned signal is very variable depending on the exact angle in which the wave hits the target and so on. There are few implementations of an amplitude modulated CW radar. An example is found in [9]. On the contrary, changing the frequency of the carrier has proved to be a successfull thing to do. By frequency modulating the signal, the returned signal can more easily be filtered from the transmitted signal and the variations depending on the illumination angle are much lower. This principle is called Frequency Modulated Continuous Wave Radar (FMCW). There have been attempts to modulate the carrier with different wave forms. The first used were probably sine waves because of its simple implementation. Now, the most popular form is a linear sweep. This kind of radar is referred to as a linear FMCW radar.

The sweep may be from lower to higher frequencies, from higher to lower frequencies or do both successively. The first two are referred to as asymmetrical linear FMCW

[4]The beat signal is the transmitted signal mixed with the received signal. Also called the conversion product.

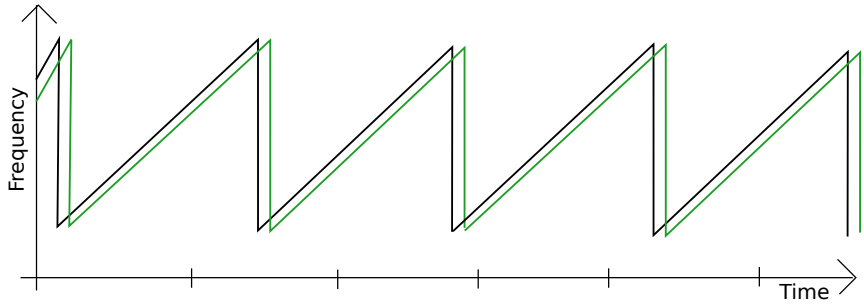


Figure 2.6: *How an asymmetrical linear sweep FMCW radar sweeps*

radars (Figure 2.6), while the third is called a symmetrical linear FMCW radar (Figure 2.7). It has been discovered that the linear waveform has many advantages compared to other waveforms [1]. The most obvious thing is probably that during the sweep, if the transmitted signal is mixed with the received signal, the beat signal will be a signal containing a wave with a frequency that is exactly proportional to the distance to the target. In Figure 2.6 and 2.7 the received signal is also indicated in green. It should be noticed that the frequency of the received signal differs from the transmitted signal at a certain time instant.

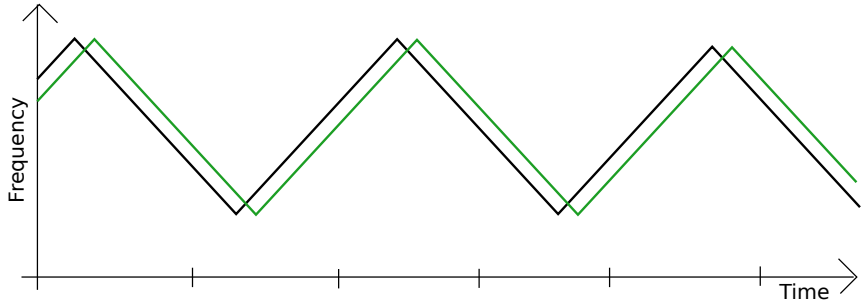


Figure 2.7: *How a symmetrical linear sweep FMCW radar sweeps*

Let us do an analysis of a single sweep - here an up-sweep. If the instantaneous frequency is given by [1]

$$f_t = f_0 + \alpha t$$

where α is the sweep rate, the phase is:

$$\phi = 2\pi \int_0^t f_t dt = 2\pi \left(f_0 t + \frac{\alpha t^2}{2} \right)$$

The transmitted signal can then written

$$s(t) = A_0 \sin 2\pi \left(f_0 t + \frac{\alpha t^2}{2} \right) \tag{2.4}$$

If the signal is delayed by τ from the target, the received signal is:

$$s_{\text{received}}(t) = B_0 \sin 2\pi \left[f_0 (t - \tau) + \frac{\alpha (t - \tau)^2}{2} \right] \tag{2.5}$$

Where B_0 represents the loss in energy. Because of

$$2 \sin x \sin y = \cos(x - y) - \cos(x + y)$$

the beat signal is:

$$c(t) = C_0 \cos 2\pi \left[f_0 \tau + \alpha t \tau - \frac{\alpha \tau^2}{2} \right] \tag{2.6}$$

Here the second component at twice the transmission frequency has been removed (filtered). The delay τ is

$$\tau = \frac{2r}{c}$$

and therefore the beat signal can be written:

$$c(t) = C_0 \cos 2\pi \left[\frac{2 f_0 r}{c} + \frac{2\alpha t r}{c} - \frac{\alpha (2r/c)^2}{2} \right] \tag{2.7}$$

The only time-varying term is the middle one, $2\alpha tr/c$ (for a non-moving target), and thus the only term leading to oscillation. The main frequency component of the beat signal will hence be

$$f_{\text{conversion}} = \frac{2\alpha r}{c} \tag{2.8}$$

Compared to the pulse radar, the FMCW radar has a number of advantages and it is almost surprising that it has not been used more before. Since the FMCW radar transmits continuously, the average power (energy) will be the same as the maximum power. The pulse radar, on the other hand, outputs all its energy in a very short time period, so its average power will be low. The ability to detect targets is related to the average power, so in order to have the same detection ability, the pulse radar needs a very high maximum power. It requires typically expensive amplifiers like klystron amplifiers. An FMCW radar can use solid-state amplifiers which today are much cheaper to produce and can be highly integrated into the design of the radar. The low maximum power provides an advantage in another way too. It is harder to detect - the radar has a low probability of interception. For military use, this is very valuable, because it makes it harder for the enemy to manipulate the radar signal.

These are not the only advantages. Since the radar is receiving at all times, it does not have to wait for the pulse to be transmitted before it starts analyzing the received signal. This causes a remarkable improvement of the minimum distance that can be measured [10]. Furthermore, the range resolution is no longer dependent on how short the pulse can be made, but on the total frequency sweep range that is employed. Since the processing then does not have to sample a very short pulse which requires a high sample rate, its range resolution can easily be made considerably better.

The main issue, however, is the leakage from the transmitter to the receiver that appears from the fact that it is receiving at the same time as it is transmitting. It puts an upper limit to how strong the transmitted signal can be, and therefore also

the maximum range. Apart from this thing, it seems like the FMCW radar almost has only advantages. It is implied that the digital revolution has happened and that the more complicated processing power needed can be satisfied. First of all this concerns an efficient FFT processor, but perhaps also the DDS.

One can conclude that the FMCW radar is very good for detection at near ranges, and the industry has started to discover this by now using it in several commercial products. Some examples are collision detection radars for cars, altitude measurement and near obstacle ship navigation.

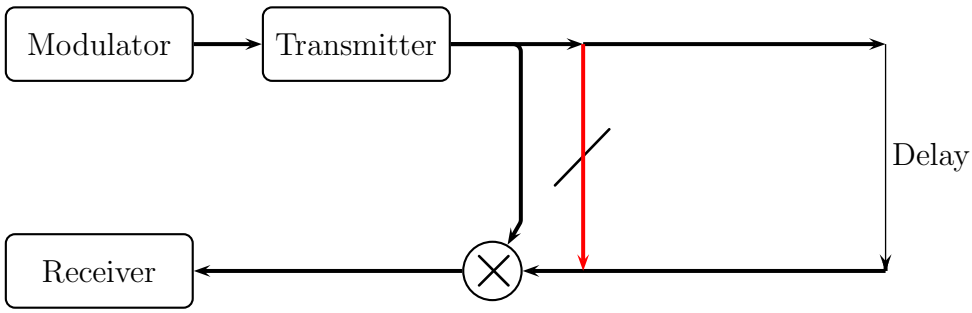


Figure 2.8: *Conceptual diagram of an FMCW Radar. The transmitter leakage is indicated with the red line*

Figure 2.8 is a simple diagram showing how an FMCW radar may be implemented. The modulator generates a sweep, which is transmitted by the transmitter. The same signal is taken to the receiver in order to mix with the reflected signal. A homodyne configuration is used. This means that the signal is mixed directly to baseband (not via an intermediate frequency). The leakage from the transmitter to the receiver is also illustrated in red. In the receiver, there can be an FFT processor which calculates the frequency spectrum of the beat signal. It is of course also possible to use a collection of filters. The receiver is in fact quite the same as for a normal CW radar.

The beat signal is sampled during the sweep. One FFT is normally calculated for each sweep. The number of samples taken during one sweep therefore decides how big the FFT must be. The sample rate defines the maximum beat signal frequency that can be detected. By increasing the sample rate, the FFT must be made bigger, but that will only make it able to represent higher frequencies. Its resolution will not get any better. If the sweep time is increased and the swept bandwidth is kept constant, the frequency sweep rate will decrease. Such a decrease causes the frequency in the beat signal to be lower. The resolution will not get any better in this case either. The only way to improve the resolution is therefore to increase the span of the sweep and it can be shown that the resolution is given by (2.3) for the FMCW radar too.

Figure 2.9 shows another structure that can be used. The difference here is that quadrature mixing is performed. This causes the receiver to be able to distinguish the frequencies above the carrier and below the carrier. This can be useful in some

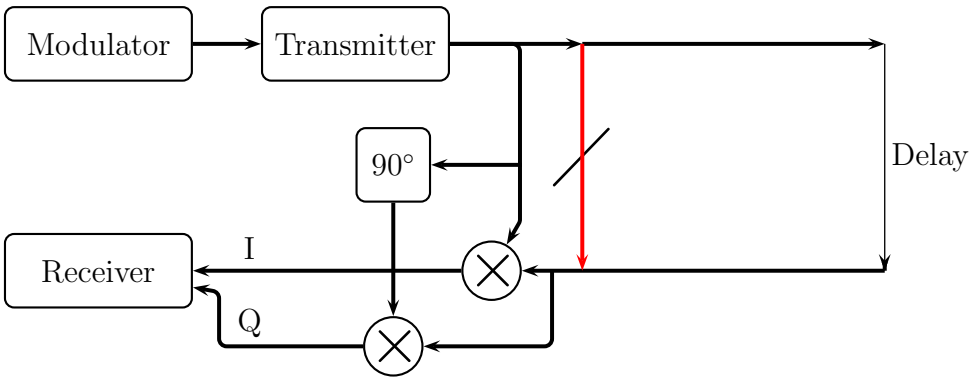


Figure 2.9: *Conceptual diagram of FMCW Radar with complex processing*

radar systems when for example the targets have IDs which are modulated onto the carrier. In order to do the quadrature mixing, the transmitted signal must be splitted in two and one of the signals must be phase-shifted 90°.

2.3.1 Ambiguity

Like the pulse radar, the FMCW also has ambiguities. There are both range ambiguities and range/velocity ambiguities. First of all, the distance to a target is calculated from the received frequency shift. If the target is further away, the frequency shift will be higher. In the case when the target is moving, there will also be a frequency shift because of the Doppler effect. It is easy to imagine that this can make it hard to say if the shift is caused by the range or the velocity. However, it is observed that in the case of Doppler shift, a higher velocity causes greater frequency shift during the up-sweep and smaller frequency shift during the down-sweep. So by exploiting both the up-sweep and the down-sweep, the Doppler shift can be separated from the range-induced shift. This way, the FMCW radar is capable of both range and velocity measurements, but only when a symmetrical sweep is used[5].

Let us see what happens to Equation (2.7) if r is not any more a constant but given by

$$r(t) = r_1 + vt$$

where v is the relative speed between the targets [1].

[5]This is not always true. By doing fourier analysis of the whole signal including the return to start for the assymmetrical sweep, the beat signal will still contain an indication of what is Doppler shift and what is range-induced shift [11]. Most radars do however not take this into consideration.

$$
\begin{aligned}
c(t) &= C_0 \cos 2\pi \left[\frac{f_0 \cdot 2(r_1 + vt)}{c} + \frac{\alpha t \cdot 2(r_1 + vt)}{c} - \frac{\alpha(2(r_1 + vt)/c)^2}{2} \right] \qquad (2.9) \\
&= C_0 \cos 2\pi \left[\frac{2f_0 r_1}{c} + \frac{2f_0 vt}{c} + \frac{2\alpha t r_1}{c} + \frac{2\alpha v t^2}{c} - \frac{2\alpha r_1^2}{c^2} - \frac{4\alpha r_1 vt}{c^2} - \frac{2\alpha v^2 t^2}{c^2} \right] \\
&= C_0 \cos 2\pi \left[\frac{2\alpha r_1 t}{c} \left(1 - \frac{2v}{c}\right) + \frac{2f_0 vt}{c} + \frac{2\alpha v t^2}{c} \left(1 - \frac{v}{c}\right) + \frac{2r_1}{c} \left(f_0 - \frac{\alpha r_1}{c}\right) \right]
\end{aligned}
$$

Thus, the contribution from the range and the velocity can now be seperated. The first term belongs to the range and the second to the velocity:

$$f_{\text{due\_to\_range}} = \frac{2\alpha r_1}{c} \left(1 - \frac{2v}{c}\right) \qquad (2.10)$$

$$f_{\text{due\_to\_velocity}} = \frac{2f_0 v}{c} = \text{Doppler effect} \qquad (2.11)$$

The third term:

$$\frac{2\alpha v t^2}{c} \left(1 - \frac{v}{c}\right)$$

is also affected by the change of distance. As [1] states: *It may either be interpreted as chirp on the range beat, due to the changing rate, or as chirp on the Doppler, due to the changing transmitter frequency.* The last term in (2.9) is a constant phase term and will hence not influence the frequency of the beat signal.

One thing that should be noted is that the frequency offset due to the range (2.10) is no longer equal (2.8). A factor $(1 - 2v/c)$ is added. It will of course not affect the result much as long as the velocity is much lower than the speed of light.

Now let us take a look at the range ambiguities. In the extreme case, the sweep may be so fast, or the target so far away, that when the reflected signal eventually returns, the radar has already started on a new sweep. The returned signal can then be mistaken for being very much closer than it really is. For most practical purposes this is however not the problem. The implementation of the sampling plays a bigger role. Because the further away the target is, the higher the frequency shift will be, the sampling frequency sets the upper limit of the range, and according to Nyquist this is $F_s/2$. If the low-pass filter in front of the AD converter is not good enough, there will be aliasing and targets further away than the maximum Nyquist range can be mistaken for being nearer.

2.3.2 Target ID

The returned signal from non-moving targets will as we have seen be a frequency shifted $\Delta f = 2\alpha r/c$ Hz. If there are many visible targets, it might be hard to distinguish the desired targets. When only a certain set of known targets are to be tracked, it is possible to equip them with onboard oscillators and mixers so that the returned signal will be shifted in frequency. This makes it possible to easier

recognize the known targets. Each of them may also be identified if different mixing frequencies are used. The returned signal will contain two carriers, one on both sides of the original returned signal at a frequency offset equal to the local oscillator of the targets. If the radar signal is given by:

$$s_0(t) = \sin 2\pi \left(f_0 t + \frac{1}{2}\alpha t^2 \right)$$

And the local oscillator at a target j is given by:

$$s_j(t) = \sin(2\pi f_j t)$$

The returned signal will be:

$$\begin{aligned} s(t) &= s_0(t) \cdot s_j(t) && (2.12)\\ &= \sin 2\pi \left(f_0(t-\tau) - \frac{1}{2}\alpha(t-\tau)^2 \right) \cdot \sin(2\pi f_j t) && (2.13)\\ &= \frac{1}{2}\cos 2\pi \left((f_0 - f_j)t - f_0\tau + \frac{1}{2}\alpha(t-\tau)^2 \right) \\ &\quad - \frac{1}{2}\cos 2\pi \left((f_0 + f_j)t - f_0\tau + \frac{1}{2}\alpha(t-\tau)^2 \right) && (2.14) \end{aligned}$$

Thus, two carriers shifted with a frequency f_j.

One other advantage of having such an oscillator at the targets, is that the returned signal will be shifted away from the worst phase noise of the transmitted signal. Hence, the sensitivity can be made better. Letting the targets have such an ID is one additonal thing that makes the FMCW radar attractive compared to a pulse radar.

2.4 Frequencies

For radars, a general rule is that the range resolution obtained is given by the bandwidth used by the transmitted signal. This attracts the use of high frequencies, because at these frequencies, a high bandwidth is not high proportional to the operating frequency, and the components of the radar won't show big variations over the band. Furthermore, there are more available bands at higher frequencies. A third reason is that the waves do not propagate further than the horizon at these frequencies but more in a straight line. At last, the resolution in bearing can be made better.

There are reasons for using lower frequencies too. First of all, the components used at low frequencies are cheaper and easier to produce. Secondly, the higher frequency used, the greater the Doppler shift will be and the sampling in Doppler domain needs to be higher which implies a higher pulse repetition frequency (PRF) in the

case of a pulse radar. This reduces the maximum unambiguous range. For FMCW radars, this last argument is not applicable. At low frequencies, it is also easier to achieve low phase noise. In this thesis, a synthesizer at 9.2-9.3 GHz is built. This is part of the X-band which is defined as being from 8.0 to 12 GHz. It is a rather high frequency band, but it is chosen because it is among other things reserved for maritime radionavigation [12].

Chapter 3

Phase Noise

Noise is often considered a problem in the design of electronic circuits. It is normally desired to achieve the minimum amount of noise possible. However, one often tends to forget that it is actually the noise that makes it possible to communicate at a location without disturbances from the other side of the globe. Since most noise is gaussian random and often also has a white spectrum, it can be averaged out in many cases. The noise also makes it possible to do remote sensing of objects because of their thermal radiation. In this text, however, noise will be regarded as a problem just as in nearly all texts. In particular it will focus on phase noise, which can be very tricky and most likely never has been considered a positive thing. In order to understand phase noise, a short summary of the types of noise that lead to phase noise is appropriate.

3.1 Internally Generated Noise

3.1.1 Thermal noise

The best known type of noise is thermal noise. It is radiated by all material that has a temperature above 0K, and it is therefore hard to prevent. For the frequencies employed by radio, the thermal noise power is given by:

$$P_n = kTB \tag{3.1}$$

where k is the Boltzmanns' constant. This noise depends only on the temperature and the bandwidth so that in order to reduce it, either of them can be reduced. For many applications, however, it is not possible to reduce the temperature, and we are left with the bandwidth as the only way to reduce the noise.

The noise voltage variance provided by a certain resistor R, is given by:

$$\overline{v}_n = \sqrt{4kTBR} \tag{3.2}$$

Thermal noise often sets the lower limit of the performance of a radio system.

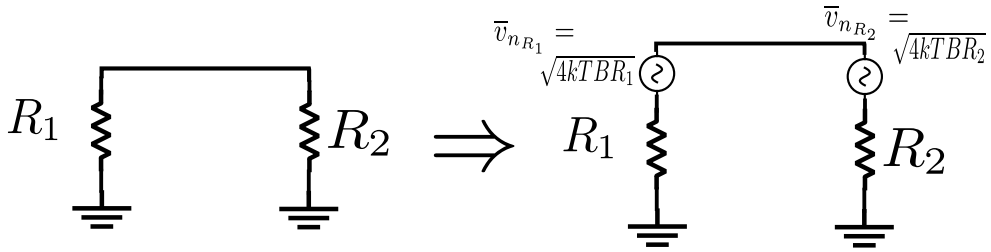


Figure 3.1: *Thermal noise of resistors*

3.1.2 Flicker noise

While thermal noise is generated in all components containing an electrical resistance, the flicker noise appears in all semiconductor based components. The exact physics to explain why it is there, still remains unclear today, but we know that it is there, and it must be taken into consideration. Flicker noise is unalike thermal noise since its spectrum is not white (at radio frequencies). It is however *pink*. This means that its energy is higher at low frequencies. It is actually proportional to $1/f$. Because of this, it actually dominates at low frequencies while at higher frequencies it is overshadowed by the thermal noise. In oscillators, the flicker noise is often modulated up and therefore stays close to the carrier, where it plays one major role in the contribution to the phase noise (see Section 3.4.1 and 3.4.2).

$$S(f) \propto 1/f \tag{3.3}$$

For FMCW Radars, which are normally designed as homodynes, the flicker noise can play a role by itself by degrading the beat siganl. For heterodynes with high IF-frequency and amplication there where the flicker noise is dominated by thermal noise, it does not play that big role.

3.2 External Noise

Thermal noise and flicker noise are types of internally generated noise. There are also sources of noise that can come from the outside. This can for instance be radiation that hits the circuit from outside. At places where there are strong transmitters nearby, this can be a problem. In addition, the 50 Hz (or 60 Hz) AC frequency can jump from the walls and into the circuit. It is however more common that this AC frequency enters through the power supply (as long as it is not battery powered equipment). It is necessary to design the power supply well in order to suppress noise. This is accomplished with good capacitors of different sizes after the rectifier and maybe also an inductor to form a good filter. There might also be active circuits involved in order to get rid of the noise. Some types of power supplies do actually generate much noise themselves. These are the switching mode power supplies. For them it is even more important to design good noise reduction filters in front of the circuits if they at all should be used.

Besides bypassing capacitors in the power supply, it is important to have bypass capacitors on the circuit board also preferably very close to the power supply pins of IC's when such are used.

3.3 What is Phase Noise?

The output of an ideal oscillator is given by:

$$s(t) = A_0 \sin(2\pi f t) \tag{3.4}$$

In the real world, however, this is an impossible signal. A more correct representation of the output of a real oscillator is given by:

$$s(t) = A_0(1 + \varepsilon(t)) \sin(2\pi f t + \phi(t)) \tag{3.5}$$

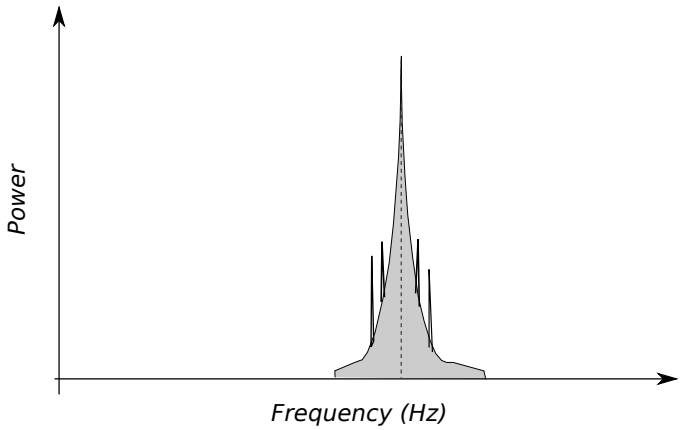


Figure 3.2: *Simple illustration of phase noise in the frequency plane. The peaks illustrate spurious signals*

Here we see two things that were not present in 3.4, $\varepsilon(t)$ and $\phi(t)$. The first is the amplitude noise, while the latter is the phase noise. The amplitude noise is the disturbance of the amplitude of the signal, while phase noise is the disturbance of the phase of the signal. It is also possible to consider the phase noise as frequency noise, since a variation in phase always will be accompagnied by a variation in frequency.[1]

A typical spectrum of the output of a real oscillator is given in Figure 3.2. On both sides of the carrier, there is a skirt of noise. In addition to this, there are some peaks caused by spurs. The noise that we see, is actually composed of both the phase noise and the amplitude noise. In a normal situation with additive noise, the phase noise will be equal to the amplitude noise in magnitude. This is given by the equiparition theorem of thermodynamics [14, p.65]. However, the two categories of noise react to changes in different ways. An imporant example is in a signal limiter. A perfect limiter will remove the amplitude noise completely, but not influence the phase noise

[1] If the phase noise is given by $S_\phi(f)$, the frequency noise is given by $S_y(f) = f^2/f_0 \cdot S_\phi(f)$ [13].

at all. In many circuits, there is unvoluntarily a limiting-effect and we are then left with the phase noise. For instance a frequency multiplier will amplify phase noise, but keep the amplitude noise unchanged or even remove it. Also, an oscillator itself will have a limiting effect and remove the amplitude noise so that the output of an oscillator is dominated by the phase noise. Because of this, we do in most cases only care about the phase noise. The amplitude noise is simply already taken care of. The expression for $s(t)$ then looks like:

$$s(t) = A_0 \sin(2\pi f t + \phi(t)) \tag{3.6}$$

Often it is believed that the skirt around the carrier in Figure 3.2 is the phase noise itself. However, the phase noise is formally defined by IEEE as one half of the spectrum of $\phi(t)$ [13].

$$\mathcal{L}(f) = \frac{S_\phi(f)}{2} \tag{3.7}$$

$\mathcal{L}(f)$ is often given in dBc/Hz. This spectrum will not be equal to what we see in Figure 3.2, but it is possible to find a relation between the two for frequencies a bit away from the carrier. In the phase spectrum, the power of the noise will approach ∞ as the frequency goes towards 0. In the signal spectrum, on the other hand, the power will approach the magnitude of the carrier as we approach 0 offset frequency.

It might be confusing to read about phase noise since it can be defined in two ways. The power of the noise in a 1 Hz bandwidth at a certain offset frequency in the signal spectrum divided by the power of the entire signal is one definition. It is measured in dBc/Hz and here the symbol $S_{\mathrm{RF}}(\Delta f)$ is used for it. The c in dBc means that it is relative to the carrier. This was actually the former official definition by IEEE for phase noise before it was changed to the definition above (3.7) in 1997 [13].

$$S_{\mathrm{RF}}(\Delta f) = \frac{\text{power density in one phase noise modulation sideband per Hz}}{\text{total signal power}} \tag{3.8}$$

The power density of ϕ for a certain frequency offset is measured in $\mathrm{rad}^2/\mathrm{Hz}$, and normally referred to 1 $\mathrm{rad}^2/\mathrm{Hz}$. Its symbol is $S_\phi(f)$. It can be shown that for offset frequencies not too close to the carrier, the value of $S_\phi(f)$ is two times the value of $S_{\mathrm{RF}}(\Delta f)$. This is the reason for that the definition above (3.7) contains a division of 2. When the formal definition was changed they wanted it to remain as similar as before. For all cases when we have no amplitude noise and the frequency offset is not too small, the value of $\mathcal{L}(f)$ might be found just by observing the signal spectrum. $\mathcal{L}$ is in the literature usually referred to as the *Single Sideband* (SSB) phase noise.

To show the relation between the two definitions, we take advantage of Bessel functions. Let's consider $\phi(t)$ given as a sine with angular frequency p. This is the same

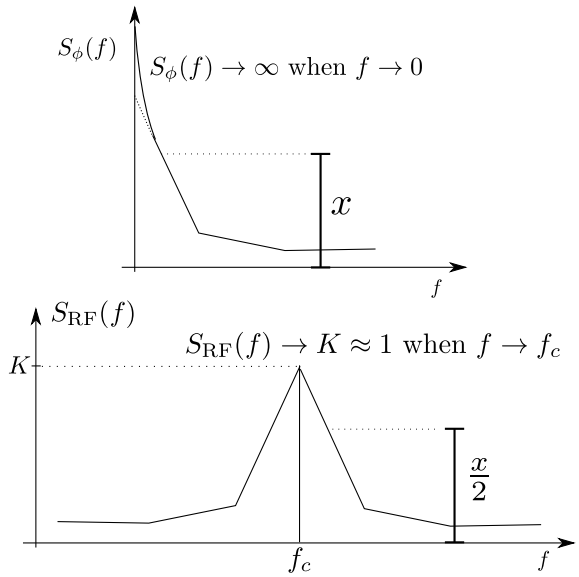


Figure 3.3: *How the signal spectrum is related to the phase spectrum for the case with no amplitude noise.*[a]

[a]That K is equal to 1 is not ecactly true, but it is what is observed with spectrum analyzers. In theory, it is the entier integrated spectral density that is equal to 1. The exact value of K is only known for some specific cases like in Equation (3.23)

as saying that the carrier is phase-modulated. The oscillator output of a carrier with the amplitude equal to 1 can then be written [15, p.10].

$$s(t) = \sin(2\pi f t + \theta \sin pt) \tag{3.9}$$

This may also be written:

$$s(t) = \sin(2\pi f t)\cos(\theta \sin pt) + \cos(2\pi f t)\sin(\theta \sin pt) \tag{3.10}$$

There are two Bessel function properties [16]:

$$e^{jx\sin\phi} = \sum_{n=-\infty}^{\infty} J_n(x)e^{jn\phi} \tag{3.11}$$

$$J_{-n}(z) = (-1)^n J_n(z) \tag{3.12}$$

Using these, we find two expressions:

$$\cos(x\sin\phi) = J_0(x) + 2[J_2(x)\cos 2\phi + J_4(x)\cos 4\phi \cdots] \tag{3.13}$$

$$\sin(x\sin\phi) = 2[J_1(x)\sin\phi + J_3(x)\sin 3\phi + \cdots] \tag{3.14}$$

With them, we can get a new expression for $s(t)$:

$$\begin{aligned} s(t) =& \sin(2\pi ft)[J_0(\theta) + 2J_2(\theta)\cos(2pt) + 2J_4(\theta)\cos(4pt) + \cdots] \\ &+ \cos(2\pi ft)[2J_1(\theta)\sin(pt) + 2J_3(\theta)\sin(3pt) + \cdots] \end{aligned} \tag{3.15}$$

Simplifying this and we get:

$$\begin{aligned} s(t) = J_0(\theta)\sin(2\pi ft) + J_1(\theta)\sin((2\pi f + p)t) - J_1(\theta)\sin((2\pi f - p)t) \\ +J_2(\theta)\sin((2\pi f + 2p)t) + J_2(\theta)\sin((2\pi f - 2p)t) + \cdots] \end{aligned} \tag{3.16}$$

The $J_\alpha(\theta)$-values are given by:

$$J_\alpha(\theta) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} \left(\frac{1}{2}\theta\right)^{2m+\alpha} \tag{3.17}$$

When θ is very small which is normally the case for random noise (at least at greater offsets), this can be simplified to:

$$\begin{aligned} J_0(\theta) &\simeq 1 \\ J_1(\theta) &\simeq \frac{\theta}{2} \\ J_2(\theta) \simeq J_3(\theta) \simeq J_4(\theta) \cdots &\simeq 0 \end{aligned} \tag{3.18}$$

Now (3.16) can be written

$$s(t) \simeq J_0(\theta)\sin(2\pi ft) + J_1(\theta)\sin((2\pi f + p)t) - J_1(\theta)\sin((2\pi f - p)t) \tag{3.19}$$

$$\simeq \sin(2\pi ft) + \frac{\theta}{2}\sin((2\pi f + p)t) - \frac{\theta}{2}\sin((2\pi f - p)t) \tag{3.20}$$

For real noise, we get an infinite number of such sine waves next to each other. If the total phase deviation is small enough, it can be shown based on this that the noise density has the following relationship:

$$S_{\mathrm{RF}}(\Delta f) \approx \frac{S_\phi(f)}{2} = \mathcal{L}(f) \tag{3.21}$$

for big enough Δf.

What we can recognize above is that the phase noise causes sidebands of equal magnitude at each side of the carrier. Their sign differ however. The spectrum will always be antisymmetrical around the carrier as long as there is only phase noise. For amplitude noise it can be shown that the spectrum always will be symmetrical

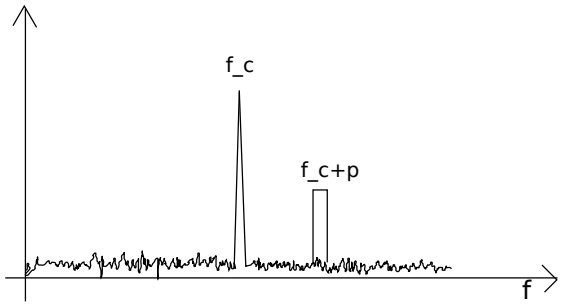


Figure 3.4: *Carrier with added noise in a 1 Hz bandwidth*

around the carrier. So in the case when there is noise only at one of the sides, we know that there must be a combination of phase noise and amplitude noise.

Consider Figure 3.4. There is noise only at one side of the carrier. This implies that there is both amplitude noise and phase noise. One can say that the amplitude noise cancels the phase noise at the lower side. Let's say that we then let the signal go through a limiter so that the amplitude noise disappears. What will happen is that we suddenly get noise on both sides of the carrier. So by introducing additive noise at one of the sidebands, we have suddenly created noise on both sides of the carrier by putting it through a limiter [15, p.20-24].

At small frequency offsets, the approximation shown earlier (3.21) does not hold. This is caused by the fact that $S_\phi(f)$ gets very big as the offset gets smaller (in fact it approaches infinity in a normal oscillator). It has been shown [4] that if $\mathcal{L}(f) = h_2/f^2$, the relation is the following :

$$\mathcal{L}(f) = \frac{S_\phi(f)}{2} = \frac{h_2}{f^2} \Rightarrow$$
$$S_{\mathrm{RF}}(f) = \frac{h_2}{(\pi h_2)^2 + \Delta f^2} \tag{3.22}$$

This is the Lorentzian shape - the shape of the squared magnitude of a one-pole lowpass filter. It ensures that the carrier does not get infinite power even though the phase noise at 0 frequency offset in the phase spectrum is infinite. It has a value of

$$S_{RF}(0) = \frac{1}{h_2 \pi^2} \tag{3.23}$$

at zero offset. Some other properties of this shape is that the total power of S_{RF} is equal to 1 and that it approaches (3.21) as the frequency offset gets big.

Typical phase noise of an oscillator

An oscillator will typically have a phase noise spectrum that consists of several terms:

$$\mathcal{L}(f) \approx \frac{h_4}{f^4} + \frac{h_3}{f^3} + \frac{h_2}{f^2} + \frac{h_1}{f} + h_0 \tag{3.24}$$

The most important is the h_2/f^2 term. Nearly all oscillators will have such a slope at some important offsets, and many approximations of the phase noise are calculated based on an "ideal" oscillator with such a phase noise spectrum. The h_4/f^4 term is often ignored. It appears mainly in the spectra of precision frequency standards at very low frequencies (below 1 Hz) [17].

When plotting dB-values, the terms in equation 3.24 are correspondingly slopes of -40 dB/decade, -30 dB/decade, -20 dB/decade, -10 dB/decade and constant.

3.4 Why Phase Noise?

Phase noise is the consequence of the contribution from all the different noise components in the oscillator. The resistors generate thermal noise, the transistors both thermal noise and flicker noise, the power supply provides ripple. These can all contribute noise which will be transformed into phase noise at the output.

At first we will look into how phase noise appears in a simple LC oscillator. Leeson was one of the first that tried to make a theory about this in 1966 [2]. His theory has been referred to a lot in the literature afterwards. However, as we will see, the theory has its limitations, and therefore during the last decades there have been attempts to make even better foundation theories for understanding the phase noise in simple oscillators.

3.4.1 Leeson's equation

Leesons's equation is an expression which can be used to estimate the phase noise of an LC oscillator. [14, p.64-67]

Consider an LC tank in parallel with a resistor and a perfect noiseless energy restoring element. The resistor will provide a noise current density of

$$\frac{\overline{i_n^2}}{\Delta f} = 4kTG = \frac{4kT}{R} \tag{3.25}$$

If this current is multiplied with the impedance, the voltage is given. The perfect noiseless energy restoring element will cancel the resistance in the resistor, and hence the impedance is just the impedance of the LC tank. At a small offset frequency $\Delta\omega$, from the center frequency, ω_0, the impedance may be approximated by

$$Z(\omega_0 + \Delta\omega) \approx j\frac{\omega_0 L}{2\frac{\Delta\omega}{\omega_0}} \tag{3.26}$$

The Q factor is given by

$$Q = \frac{R}{\omega_0 L}$$

and hence the impedance may be written:

$$|Z(\omega_0 + \Delta\omega)| \approx R\frac{\omega_0}{2Q|\Delta\omega|} \tag{3.27}$$

The squared mean noise voltage density is therefore given by:

$$\frac{\overline{v_n^2}}{\Delta f} = \frac{\overline{i_n^2}}{\Delta f}|Z|^2 = 4kTR\left(\frac{\omega_0}{2Q\Delta\omega}\right)^2 \tag{3.28}$$

The estimated phase noise in dB is hence given by

$$\mathcal{L}(\Delta\omega) = 10\log\left(\frac{\overline{v_n^2}}{2P_{\text{sig}}R\Delta f}\right) = 10\log\left[\frac{2kT}{P_{\text{sig}}}\left(\frac{\omega_0}{2Q\Delta\omega}\right)^2\right] \tag{3.29}$$

It basically says that in order to minimize the phase noise, the Q factor and the total signal power should be increased. Expression (3.29) does not show the whole truth. In reality, the region proportional to $1/(\Delta\omega)^2$ is larger than predicted by it. There is also an unavoidable noise floor at sufficiently high offsets. At very small offsets there is typically also a region proportional to $1/(\Delta\omega)^3$. To take all these extra considerations into account, the full Leeson's equation is given by:

$$\mathcal{L}(\Delta\omega) = 10\log\left[\frac{2FkT}{P_{\text{sig}}}\left\{1 + \left(\frac{\omega_0}{2Q\Delta\omega}\right)^2\right\}\left(1 + \frac{\Delta\omega_{1/f^3}}{|\Delta\omega|}\right)\right] \tag{3.30}$$

Figure 3.5 shows the difference between what Equation (3.29) (grey) and (3.30)

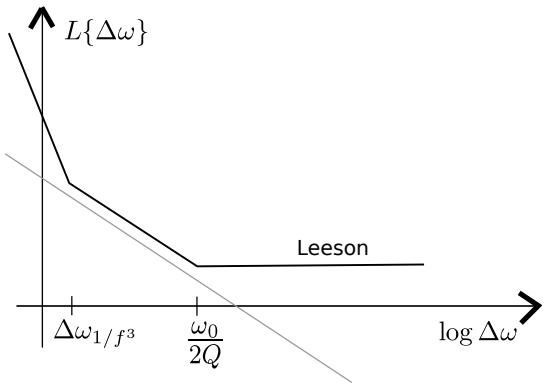


Figure 3.5: *Leeson's equation*

(black) represent. A factor F has been introduced together with a lower noise floor and a $1/f^3$ factor. One of the main problems is that this method does not provide any way to actually calculate the factor F and the value of $\Delta\omega_{1/f^3}$.

3.4.2 LTV model

Since Leeson's model contains empirical parameters without showing where they come from, Lee and Hajmiri developed a new model in 1998 [3]. It considers an oscillator as a time-varying system, and by doing this, much higher insight is achieved into how the phase noise appears. Earlier texts claimed that to explain different effects of the phase noise development in oscillators, it was necessary to consider the system as nonlinear and they therefore normally skipped any explanation since it would become quite complicated. This new model, however, shows that it is not strictly necessary to consider it as a nonlinear system. The time-variant model, goes far in showing that up-mixing of flicker noise to near-carrier phase noise and that down-mixing of near-harmonics noise contributes to the $1/f^2$ phase noise, is fully possible even when the oscillator is considered a linear system as long as the time-variation is taken into consideration.

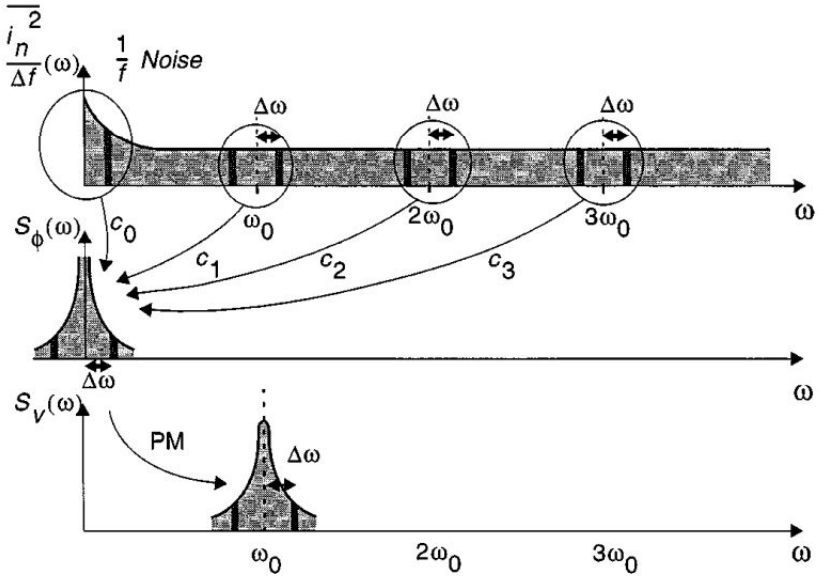


Figure 3.6: *Evolution of circuit noise into phase noise (Borrowed from [18, p.331])*

Figure 3.6 shows how noise from different frequency bands adds to the phase noise. The flicker-noise (upper Figure to the left) becomes $1/f^3$ phase noise in the phase spectrum. The noise near every harmonic of the carrier frequency moves near the carrier and adds to the $1/f^2$ noise. How can it be explained that the noise shifts frequency if nonlinearity is not taken into consideration?

The key to understanding this can be found in Figure 3.7. This is the response of a current impulse in an LC tank that is already oscillating. What can be observed is that the response of the impulse is different depending on where in the cycle it appears. If it appears when the amplitude is maximum (upper), it will not influence the phase anything. However, when it appears at the same time as the sine is at the zero-crossing (lower), the phase will be shifted. This basically says that the system is time variant and theories not including time-invariance will fail to explain much

(like Leeson). It means that in order to minimize the phase noise, it will be good to add the lost energy (in the oscillator) when the amplitude is at its maximum.

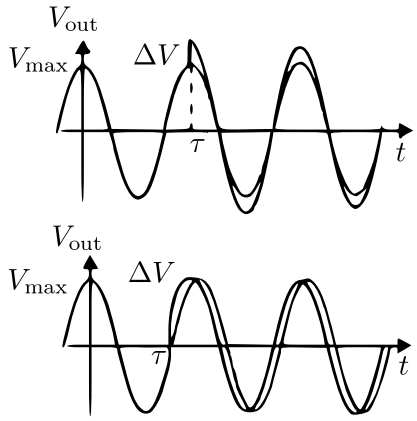


Figure 3.7: *Impulse response of LC tank [18, p.329]*

The impulse response to the phase of one such pulse can be written:

$$h_\phi(t,\tau) = \frac{\Gamma(\omega_0\tau)}{q_{\max}} u(t-\tau) \tag{3.31}$$

$$\tag{3.32}$$

so that the phase response is equal too:

$$\phi(t) = \int_{-\infty}^{\infty} h_\phi(t,\tau) i(\tau) d\tau \tag{3.33}$$

We see that h_ϕ has two arguments. The τ actually means at what time the pulse is inserted and takes care of the time-invariance. The Γ is a function that will define when the impulse makes the biggest influence to the phase. It is called the impulse sensitivity function (ISF). It will be periodic. Since it is periodic it can be expressed as a Fourier series:

$$\Gamma(\omega_0\tau) = \frac{c_0}{2} + \sum_{n=1}^{\infty} c_n \cos(n\omega_0\tau + \theta_n) \tag{3.34}$$

Setting in for (3.33), we get:

$$\begin{aligned} \phi(t) &= \int_{-\infty}^{\infty} h_\phi(t,\tau) i(\tau) d\tau = \frac{1}{q_{\max}} \int_{-\infty}^{t} \Gamma(\omega_0\tau) i(\tau) d\tau \\ &= \frac{1}{q_{\max}} \left[\frac{c_0}{2} \int_{-\infty}^{t} i(\tau) d\tau + \sum_{n=1}^{\infty} c_n \int_{-\infty}^{t} i(\tau) \cos(n\omega_0\tau) d\tau \right] \end{aligned} \tag{3.35}$$

If then a sinusoidal current is added which is near an integer multiple m of the oscillation frequency,

$$i(t) = I_m \cos[(m\omega_0 + \Delta\omega)t] \tag{3.36}$$

it is found that only the terms where $n = m$ contribute considerably and (3.35) can be written

$$\phi(t) \approx \frac{I_m c_m \sin(\Delta\omega t)}{2q_{\max}\Delta\omega} \tag{3.37}$$

This is interesting because it says that injected noise at frequencies near harmonics of the oscillator frequency, ω_0, also contribute to the near-carrier phase noise. This was earlier claimed to be because of nonlinearities, but we see that the time invariance can cause this behaviour. We see that the influence is scaled by the c_m factor. The m is the number of the harmonic, so that if we know all the c_m factors for $2 \leq m < \infty$, we will be able to say how much each of the near-harmonics noise bands influence the phase spectrum. Another quite interesting thing is that c_0 actually says how much of the near-0Hz noise that will be upmixed, and this is mainly the flicker noise. At last the c_1 says how much of the near carrier noise that will be added to the carrier phase noise. Figure 3.6 shows how the noise becomes phase noise. Just one problem remains, how do we find the c_m values - or in other words, how do we find the ISF ($\Gamma(\omega_0\tau)$). In [3] some methods are explained. It is possible to measure the ISF by injecting pulses and see the response or one may calculate it. It is referred to [3] for details.

3.4.3 Nonlinear models

Even though the LTV method is able to explain how the noise from different frequencies adds the phase noise, it is a matter of fact that the oscillator behaviour is nonlinear by nature. So it can be expected that the results obtained will not take everything into account and hence not be completely correct. Some approximations employed in the LTV method has been shown to be false [19], but still may provide good design intuition. There have been several attempts to analyze the phase noise in a nonlinear system too, and perhaps the most acknowledged of these is presented by Demir, Mehrotra and Roychowdhury in [4].

The noise is in their work modelled rather differently. If the periodic response of an unperturbed oscillator is given by $x_s(t)$, the signal with noise (both amplitude and phase) is described as:

$$x_s(t + \alpha(t)) + y(t) \tag{3.38}$$

$\alpha(t)$ contains the time shift (phase shift) experienced after time t. It was stated above that $S_\phi(f)$ approaches ∞ as $f \to 0$. That implies that after a certain time, most likely (in practice always) the signal will have a constant phase shift relative to the beginning. This means that $\alpha(t)$ will in general keep increasing with time. $y(t)$ will on the other hand always remain small because it represents the orbital deviation. Nonlinear differential equations are solved and some important results are derived. One of them is that the spectrum is Lorentzian near the carrier as

given in (3.22). Another thing is that the oscillator behaviour with noise is shown to be stationary despite being intuitively hard to understand. It also points out that what is done in the LTV method is in general false.

The article only considers noise appearing from white noise sources, hence only phase noise with a $1/f^2$ characteristic, and it is therefore not straightforward to use their result in practical desisgn. It mainly attemps to establish a foundation theory for the description of phase noise in nonlinear systems, which has been lacking earlier. Based on the huge number of newer articles citing their work, it may be concluded that they have kind of succeeded with the last thing.

3.5 Propagation in Devices

3.5.1 Mixers

The mixer will preserve the phase noise. It will be added or subtracted depending on which mixing product that is wanted.

Two perturbed carriers are given by:

$$\begin{aligned} s_1(t) &= \sin(2\pi f_1 t + \phi_1(t)) \\ s_2(t) &= \sin(2\pi f_2 t + \phi_2(t)) \end{aligned}$$

Their product is:

$$\begin{aligned} s(t) &= s_1(t) \cdot s_2(t) \\ &= \sin(2\pi f_1 t + \phi_1(t)) \cdot \sin(2\pi f_2 t + \phi_2(t)) \\ &= \frac{1}{2}\cos[2\pi f_1 t + \phi_1(t) - 2\pi f_2 t - \phi_2(t)] \\ &\quad - \frac{1}{2}\cos[2\pi f_1 t + \phi_1(t) + 2\pi f_2 t + \phi_2(t)] \\ s(t) &= \frac{1}{2}\cos[2\pi(f_1 - f_2)t + \phi_{\text{difference}}(t)] \\ &\quad - \frac{1}{2}\cos[2\pi(f_1 + f_2)t + \phi_{\text{sum}}(t)] \end{aligned} \tag{3.39}$$

One of these two terms is then typically selected by a filter since they are at completely different frequencies. For the sum frequency, the phase noise is given by:

$$\phi_{\text{sum}}(t) = \phi_1(t) + \phi_2(t) \tag{3.40}$$

For the difference frequency:

$$\phi_{\text{difference}}(t) = \phi_1(t) - \phi_2(t) \tag{3.41}$$

In FMCW radars, the interest is almost always in the difference frequency, because it is how the beat signal is made. The phase noise will therefore be given by $\phi_1(t) - \phi_2(t)$ and as we will see in Chapter 5, this can be made an advantage.

3.5.2 Frequency multipliers

The effect of frequency multiplication is also considered [15, p.76]. If a perturbed carrier is given as:

$$s_1(t) = \sin(2\pi f_1 t + \theta \sin pt) \tag{3.42}$$

the phase and the instantaneous frequency of the signal is

$$\theta_{\text{sig}} = 2\pi f_1 t + \theta \sin pt \tag{3.43}$$

$$f_1(t) = \frac{1}{2\pi}\frac{d\theta_{\text{sig}}}{dt} = f_1 + \frac{1}{2\pi}(p\theta \cos pt) \tag{3.44}$$

If the frequency is multiplied by n

$$\begin{aligned} f_2(t) = nf_1(t) &= nf_1 + \frac{1}{2\pi}(np\theta \cos pt) \\ &= f_2 + \frac{1}{2\pi}(np\theta \cos pt) \end{aligned} \tag{3.45}$$

The phase of this is:

$$\begin{aligned} \theta_{\text{sig}} &= \int 2\pi \left(f_2 + \frac{np\theta}{2\pi} \cos pt \right) dt \\ &= (2\pi f_2 t + n\theta \sin pt) + C \end{aligned} \tag{3.46}$$

Neglecting the constant C and the new signal is given by:

$$s_2(t) = \sin(2\pi f_2 t + n\theta \sin pt) \tag{3.47}$$

Hence, the phase noise has been scaled by the frequency multiplication factor n.

$$\phi_2 = n\phi_1 \tag{3.48}$$

In dB, the relation is

$$\mathcal{L}_{\text{out}}(f) = \mathcal{L}_{\text{in}}(f) + 20\log n = \mathcal{L}_{\text{in}}(f) + 20\log\frac{f_{\text{out}}}{f_{\text{in}}} \tag{3.49}$$

- an expression which may be very useful.

3.6 Measurement Techniques

The most obvious way to measure phase noise is to use a spectrum analyzer directly on the signal. The noise observed will have to be scaled according to the bandwidth used when measuring with the formula:

$$\text{Phase noise in 1 Hz} = \text{Noise in RBW} - 10\log\text{RBW} \tag{3.50}$$

where RBW is the resolution bandwidth used while measuring. In addition, a correction factor which takes into account the implementation of the RBW, the logarithmic display mode and the detector characteristic should be added [20]. Many spectrum analyzers are capable of doing this scaling automatically. The noise will then be given in noise power per Hz. By dividing this by the total carrier power, an estimate of $\mathcal{L}(f)$ is found. This corresponds to the old definition, S_{RF}, in Equation (3.8) which is measured in dBc/Hz.

However, using a spectrum analyzer directly on the signal has its limitations. First of all, as stated above, the skirt around the carrier is composed of both the phase noise and the amplitude noise. We therefore need to make sure that the amplitude noise is eliminated. In some cases it can also be assumed that the phase noise and amplitude noise are equally strong and half of the observed power can be attributed to each of them. Another problem is that if the phase noise is low, the spectrum analyzer needs a very high dynamic range in order to see the phase noise. This is caused by the fact that the carrier itself is observed at the same time. The carrier furthermore makes it harder to measure near-carrier phase noise. Because of all this, it is natural that other methods have been developed in order to measure the phase noise [21].

3.6.1 Down-conversion and filtering of carrier

One possibility is to down-convert the signal to an IF and pass it through a sharp IF filter which removes the carrier while keeping the noise around it. The signal can then be measured with a spectrum analyzer with the carrier removed and the constraints on the dynamic range are reduced a lot. Also the noise introduced by the mixing with the spectrum analyzer's local oscillator is reduced. The quality of the oscillator used for the down-conversion must of course be very good, preferably better than the oscillator under test (DUT).

One drawback is that because of the IF filter, it is hard to measure noise closer to the carrier than 10 kHz. Compared to the previous method it is also slightly harder to calibrate the measurement. This is because the carrier has been removed. At first, the down-conversion oscillator (LO) must be tuned to a frequency so that the carrier falls into the passband of the filter and then measure the level of it. Afterwards, the LO must be tuned so that the carrier is removed and then the noise can be measured.

3.6.2 Quadrature method

A different method can measure $S_\phi(f)$ directly. In order to do this, another stable oscillator with exactly the same frequency is needed. If the two oscillators are mixed, the result after a low-pass filter will be the phase spectrum. It is compulsary that the two oscillators are in quadrature - 90° degrees phase shifted to each other. It might be quite hard to achieve this, and in cases with a drifting oscillator it is practically impossible. However, if the two oscillators are very stable and controlled

by a synthesizer, it is possible to adjust them exactly to get them in quadrature. In other cases it is possible to use a Phase-locked loop in order to lock the phase of the second oscillator to the first.

This method can measure phase noise really well. It is therefore used to measure high performance oscillators, like for instance crystal oscillators. One problem with this method is that it requires a more complicated calibration than the previous methods.

3.6.3 Delay line discriminator

A third method measures the frequency noise. It is performed by splitting the oscillator signal in two and then delay one of them 90° and mix them together. After a low-pass filter we have frequency demodulated the signal. It is only possible to measure noise up to a certain offset frequency, because the discriminator has a bandwidth of $\text{sinc}(\pi T f_m)$ where the T is the delay of the delay line. This method works well for drifting oscillators and it eliminates the amplitude noise just like the quadrature method. However it is not so useful for high-offset noise and for noise very close to the carrier.

It is also possible to design such an FM detector at an IF frequency. Then the discriminator can be made better since it is designed for only one frequency. The signal must first be mixed down to the IF frequency.

3.7 Jitter

The jitter is the perturbation of the phase seen in time domain. There are several definitions of jitter, but the most common are *absolute jitter* and *period jitter* [17].

The absolute jitter is defined as the total time deviation from where a clock transition were supposed to be without jitter after time nT_0 where T_0 is the nominal period $T_0 = 1/f_0$. The standard deviation of the absolute jitter is:

$$\sigma_{j\_\text{abs}}(T_0) \approx \frac{1}{2\pi f_0}\sqrt{\int_0^\infty S_\phi(f)df} = \frac{1}{\pi f_0}\sqrt{\frac{1}{2}\int_0^\infty \mathcal{L}(f)df} \tag{3.51}$$

The period jitter is the time difference of a single clock period and the ideal clock period. It is often more useful than absolute jitter - at least in digital circuits with high clock speeds. The period jitter is often also referred to as cycle jitter. The standard deviation of the period jitter is:

$$\sigma_j(T_0) \approx \frac{1}{\pi f_0}\sqrt{\int_0^\infty 2\sin^2(\pi f T_0)\mathcal{L}(f)df} \tag{3.52}$$

In most cases it does not converge. Then other integration limits can be set, f_1 and f_2. The lower limit f_1 is often given as 10 Hz because everything below 10 Hz is

regarded as wandering, and everything above is considered jitter. The upper limit must be chosen depending on the application and actual phase spectrum. In some cases when f_1 is chosen as 10 Hz and f_2 kept at ∞, it converges.

A special case is when the phase noise displays a -20 dBc/Hz slope (it is proportional to $1/f^2$). Then the period jitter can be related to the phase noise by:

$$\sigma_j = \sqrt{\frac{f^2 \mathcal{L}(f)}{f_0^3}} \tag{3.53}$$

If $\mathcal{L}(f) = h_2/f^2$, this is the same as:

$$\sigma_j = \sqrt{\frac{h_2}{f_0^3}} \tag{3.54}$$

The unit of σ_j is seconds.

In the case with real phase noise which displays a characteristic of -30 dBc/Hz slope at low offsets, Equation (3.53) is only approximate. It is kind of paradoxal because Equation (3.52) in theory diverges when a -30 dBc/Hz slope is present, but still crystal oscillators are specified with a small period jitter [17]. This is probably because the -30 dBc/Hz slope component causes such a slow variation that it will not be observed in a short time, and hence the lower integration limit is set to a finite value above 0 in order to find a more practical value of the period jitter.

Chapter 4

Phase Locked Loops

The Phase Locked Loop is an important part of most of modern day radio equipment. It enables oscillators which are capable of frequency changes (typically VCOs) to be stable and to be easily controlled digitally. The concept consists of using an oscillator with a control input which will tune the frequency and a phase detector which will compare the phase of a reference signal and try to adjust the tunable oscillator so that its phase is synchronized to the phase of the reference signal. When it is, we say that the phase is locked to the reference oscillator, thereby the name Phase-Locked Loop.

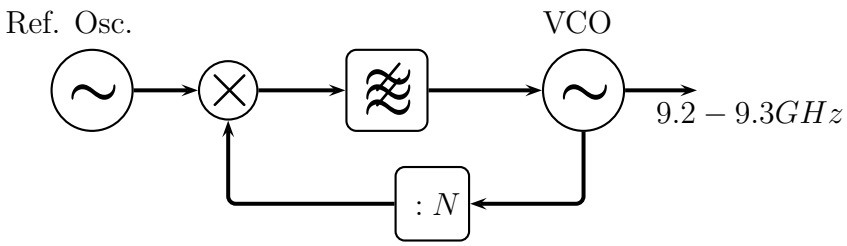


Figure 4.1: *Block diagram of PLL*

It is also possible to let the VCO be locked to a reference oscillator running at another frequency if a divider is used before the phase detector. In Figure 4.1, this is included with the block $: N$. In addition, a divider can be placed after the reference oscillator in order to get a fractional relation between the two oscillator frequencies.

When the PLL is in lock, it can be considered a linear control theory system where the variable through the loop is phase [17]. In this case, it is possible to use common linear control theory techniques in order to analyse the system. The reason for that we need to use phase and not amplitude as the variable, is that it is the difference of the phases that is compared in the feedback. The phase varies more or less linearly while the amplitude follows a sine curve in this part of the loop. Using the amplitude as the variable would therefore make little sense.

A linearized model of the PLL in lock is presented in Figure 4.2. By looking at this,

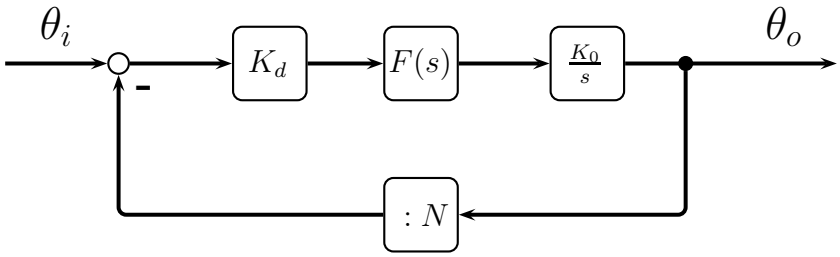


Figure 4.2: *Mathematical block diagram of PLL*

an expression for the output phase θ_o of the closed loop-system can be found to be:

$$\theta_o(s) = \frac{K_d K_0 F(s)}{s}\left(\theta_i - \frac{\theta_o(s)}{N}\right) \tag{4.1}$$

Using this, the transfer function of the closed-loop system is found:

$$\theta_o(s)\left(1 + \frac{K_d K_0 F(s)}{sN}\right) = \frac{K_d K_0 F(s)}{s}\theta_i \tag{4.2}$$

$$\theta_o = \frac{K_d K_0 F(s)}{s + K_d K_0 F(s)/N}\theta_i \tag{4.3}$$

$$H(s) = \frac{\theta_o(s)}{\theta_i} = \frac{K_d K_0 F(s)}{s + K_d K_0 F(s)/N} \tag{4.4}$$

It is important to discuss what the different variables represent. K_d is the amplification factor of the phase detector. The difference in phase measured in radians is multiplied by this factor before entering the next stage in the block diagram which is a filter $F(s)$. The filter is needed in order to shape the dynamics of the loop and the loop would be quite useless without it. A secondary effect of the filter is to low-pass filter noise and unwanted products coming from the phase detector. The filter is discussed further in Section 4.2. The oscillator is represented by the block $\frac{K_0}{s}$. Division by s in the Laplace domain is the same as integration in the time domain. This is exactly what is needed in order to get from frequency to phase. The phase is the integration of the frequency and since we are working with phases we need to divide by s. The constant K_0 is the tuning ramp of the VCO. It is normally measured in rad/s· V. VCOs normally do not have completely linear tuning response, but near the operating frequency, it might still be linearized and we get the constant K_0.

4.1 Phase Detectors

There are different types of phase detectors (PD). The most basic is the mixer. It performs a multiplication of the two inputs, and one of the products (after low-pass filtering) is the sine of the phase difference. We therefore have a sinusoidal response of the phase difference, but as long as the difference is small, the sine is a good approximation of a linear function.

If the reference oscillator signal $v_r(t)$ and the oscillator signal $v_o(t)$ are given by,

$$v_r(t) = A_1 \sin(\omega t + \theta_r) \tag{4.5}$$
$$v_o(t) = A_2 \cos(\omega t + \theta_o) \tag{4.6}$$

the output of the mixer with gain K_m is given by

$$\begin{aligned} v_d(t) &= K_m v_i(t) v_o(t) \\ &= \frac{1}{2} K_m A_1 A_2 \sin(\theta_r - \theta_o) + \frac{1}{2} K_m A_1 A_2 \sin(2\omega t + \theta_r + \theta_o) \end{aligned} \tag{4.7}$$

It can be observed that after lowpass-filtering, the output is the sine of the phase difference scaled with

$$K_d = \frac{K_m A_1 A_2}{2} \tag{4.8}$$

For small phase errors, the sine will be very much like a linear function. Note that for this phase detector, the PLL locks when the inputs are near 90° phase shifted to each other. In many cases it is desired to have the phase shift equal to 0. This can be accomplished by adding a delay even though it will not be perfect for all frequencies.

This mixer phase detector only detects phase difference. If the two oscillators operate at two different frequencies, it will not necessarily be able to lock them to each other. As long as the frequency difference is not too big, it will handle it because of pull-in effects, but the speed that the pull-in effect works at does not get higher when the frequency difference is higher. It will therefore not be good when a big tuning range is wanted. It may need some time to get in lock if it actually gets in lock at all. What is necessary, is some way to also measure the frequency difference and use this in order to lock the oscillators quickly to each other. The most popular method for doing this is with a Phase-Frequency Detector (PFD).

4.1.1 PFD

The Phase-Frequency Detector was most likely first presented in 1971 by Brown J.I [22], and since then it has become very popular and it is used in most frequency synthesizers nowadays. The PFD is a sequential detector. That implies that it is dependent on zero-transitions of the signal in order to track the phase (and frequency) difference. Because of this, it is not good when there is very much noise involved.

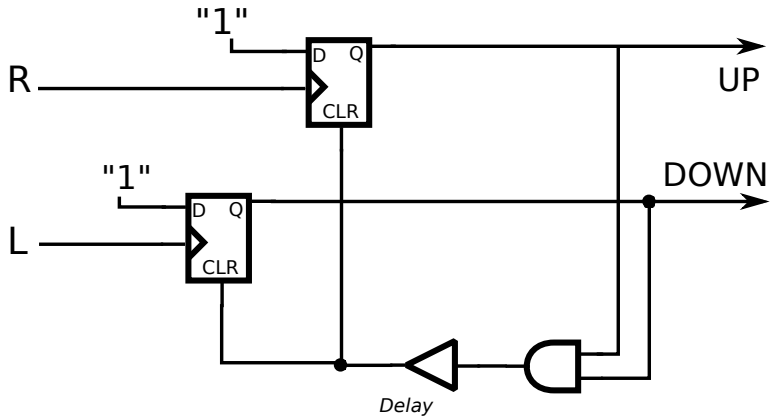


Figure 4.3: *A basic PFD*

However, it works very well when the signals are clean. Figure 4.3 explains how it can be made.

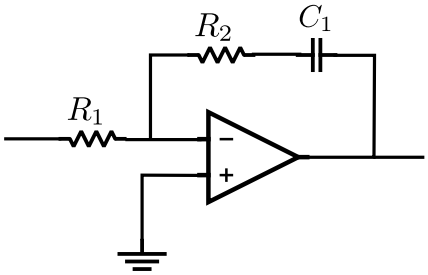
The output of the PFD are an UP-signal and a DOWN-signal. They indicate the direction of the frequency change that is needed. Every time they both are active at the same time, they will both after a very short delay turn off again. When they are off, the state that the PD is in is called a zero-state and it means that no changes of the tuning voltage are needed. It is this third state which enables it to also indicate the direction and magnitude of a frequency difference.

The two digital outputs, UP and DOWN, must be transformed into a voltage since the VCO is controlled by a voltage. This can be accomplished in several ways. One method is to use a charge pump [23]. It is a circuit that lets each of the two outputs control each their current sources. If UP is active, current is sinked to the output, if DOWN is active, current is drawn from the output. If then there is a capacitor attached to the output, there will be a varying voltage over it. Instead of using a charge pump it is possible to let the two outputs be attached to a differential integrator implemented with an op-amp.

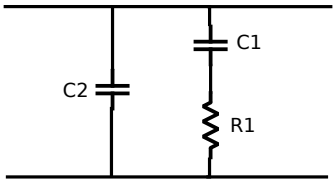
4.2 Filter

For all the different implementations of a phase detector above it is absolutely necessary to have a filter after them in order to control the dynamics of the loop. This filter is referred to as the loop filter. It will also low-pass filter the tuning signal in order to remove noise, but that is just a secondary effect which we will not be so concerned about. PLLs are classified depending on how many integrators they have in the loop. There is always one integrator in the VCO because it changes a phase difference into a frequency. Therefore all PLLs are at least of *Type 1*. Possibly other integrators must be built into the filter. It can be shown that in order to have no static phase difference between the two oscillators, the PLL must at least be of type 2.

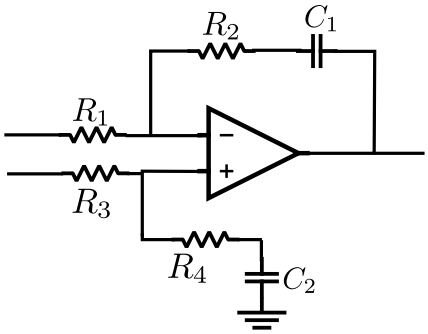
The filter may be either passive or active. What is best to use depends on the PD



(a) An active filter



(b) A filter for a charge pump PFD



(c) An active differential filter

Figure 4.4: *Showing 3 common types of loop filters*

type, the VCO tuning voltage range, and the need for extra integrators in the filter. For example will the charge pump based PD not need an active filter in order to get an extra integrator (see Figure 4.4(b)), while other PD types always will need an active filter in that case. Furthermore, if the tuning voltage of the VCO exceeds the maximum output of the phase detector voltage, there will be a need for an active filter.

Transfer function of an active integrator filter

Consider Figure 4.4(a). The transfer function of this filter is given by:

$$F(s) = \frac{R_2 + \frac{1}{sC_1}}{R_1} = \frac{sR_2C_1 + 1}{sR_1C_1} \tag{4.9}$$

Transfer function of a charge pump filter

For the case with a charge pump, the circumstances are a bit different. Since the charge pump provides a constant current, I_p, over a certain time equal to the phase shift in each cycle, it is necessary to consider the case when the oscillator frequency is much higher than the loop bandwidth. Then the average current supplied to the loop filter is given by [17]:

$$i_d = \frac{I_p \theta_e}{2\pi} \tag{4.10}$$

The PD gain is hence $K_d = I_p/2\pi$. The transfer function of the entire PLL is given by

$$H(s) = \frac{K_0 I_p Z_F(s)/2\pi}{s + K_0 I_p Z_F(s)/2\pi} \tag{4.11}$$

where $Z_F(s)$ is the impedance of the loop filter. By looking at Figure 4.4(b), the impedance can be found to be:

$$Z_F(s) = \frac{\frac{s}{C_2} + \frac{1}{C_1 C_2 R_1}}{s^2 + \frac{s}{C_1 R_1} + \frac{s}{C_2 R_1}} \tag{4.12}$$

In fact, the capacitor C_2 is not strictly needed in order to have the desired integrator effect of the loop filter, but it is good to have since it prevents big spikes reaching the loop filter.

4.3 Full Transfer Function

If the filter with an active integrator is used, the transfer function of the filter is given as:

$$F(s) = \frac{sR_2C + 1}{sR_1C} \tag{4.13}$$

By putting this into the expression for $H(s)$, we get:

$$\begin{aligned} H(s) &= \frac{K_d K_0 \left(\frac{sR_2C+1}{sR_1C}\right)}{s + \frac{K_d K_0}{N}\left(\frac{sR_2C+1}{SR_1C}\right)} \\ &= \frac{K_d K_0 \frac{R_2}{R_1} s + \frac{K_d K_0}{R_1 C}}{s^2 + s\left(\frac{K_d K_0 R_2}{N R_1}\right) + \frac{K_d K_0}{N R_1 C}} \end{aligned} \tag{4.14}$$

This is an expression for the closed-loop response of a Type-2 second-order PLL. By noting that a general expression for a second order transfer function can be given as:

$$H_{\text{general}}(s) = \frac{N\left(2\zeta\omega_n s + \omega_n^2\right)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.15}$$

Expressions for the natural frequency, ω_n, and the damping factor, ζ, are found to be:

$$\omega_n = \sqrt{\frac{K_d K_0}{N R_1 C}} \qquad \zeta = \frac{R_2}{2}\sqrt{\frac{C K_d K_0}{N R_1}} \tag{4.16}$$

The natural frequency in Hz, f_n, later also referred to as the loop bandwidth is given by:

$$f_n = \frac{\omega_n}{2\pi} = \frac{1}{2\pi}\sqrt{\frac{K_d K_0}{N R_1 C}} \tag{4.17}$$

These can be used to predict the behaviour of the PLL.

4.4 Tracking

When the PLL is in lock, changes of the reference oscillation frequency will be tracked by the VCO. As long as the changes are not too abrupt, the PLL will stay in lock and track the changes smoothly.

For a type 2 PLL, the phase error after a frequency step is given by [17]

$$\theta_e(t) = \frac{\Delta\omega}{\omega_n}\left(\frac{1}{\sqrt{1-\zeta^2}}\sin\sqrt{1-\zeta^2}\omega_n t\right)e^{-\zeta\omega_n t} \tag{4.18}$$

for $\zeta < 1$ and

$$\theta_e(t) = \frac{\Delta\omega}{\omega_n}\left(\frac{1}{\sqrt{\zeta^2-1}}\sinh\sqrt{\zeta^2-1}\omega_n t\right)e^{-\zeta\omega_n t} \tag{4.19}$$

for $\zeta > 1$.

What needs to be ensured in order to keep the PLL locked, is that the phase error given by the expression above always stays much smaller than 2π. Exactly how much smaller, depends on what kind of phase detector that is used. By differentation of (4.18) and (4.19), we get the frequency error.

$$\omega_e(t) = \Delta\omega\cos(\sqrt{1-\zeta^2}\omega_n t)e^{-\zeta\omega_n t} - \Delta\omega\left(\frac{1}{\sqrt{1-\zeta^2}}\sin\sqrt{1-\zeta^2}\omega_n t\right)\zeta e^{-\zeta\omega_n t} \tag{4.20}$$

for $\zeta < 1$. And for $\zeta > 1$:

$$\omega_e(t) = \Delta\omega\cosh(\sqrt{\zeta^2-1}\omega_n t)e^{-\zeta\omega_n t} - \Delta\omega\left(\frac{1}{\sqrt{\zeta^2-1}}\sinh\sqrt{\zeta^2-1}\omega_n t\right)\zeta e^{-\zeta\omega_n t} \tag{4.21}$$

Figure 4.5 and 4.6 are plots of $(\Delta\omega - \omega_e(t))/2\pi$ for different values of ω_n. It shows how the VCO will track a frequency step in the reference. The higher the value of ω_n, the faster the VCO will settle at the desired frequency.

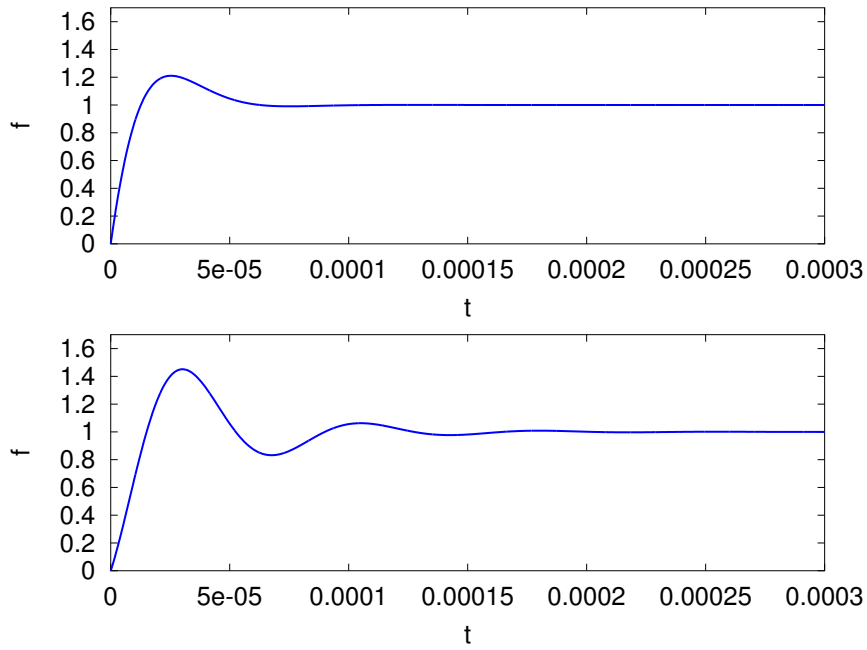


Figure 4.5: *Frequency step with a PLL bandwidth of 14kHz.* $\zeta = 0.7$ *(upper) and* $\zeta = 0.3$ *(lower)*

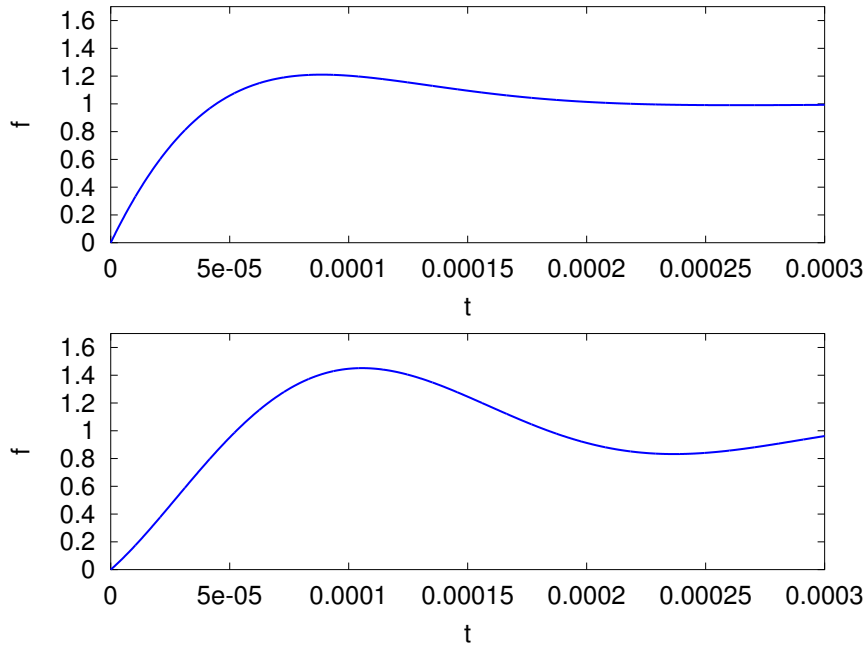


Figure 4.6: *Frequency step with a PLL bandwidth of 4kHz.* $\zeta = 0.7$ *(upper) and* $\zeta = 0.3$ *(lower)*

4.5 Phase Noise in Phase Locked Loops

The phase noise of a phase-locked loop propagates through the loop and its characteristics are changed. The PLL has a tendency to improve the phase noise of the VCO in large parts near the carrier as long as the reference oscillator has a phase noise much better than the VCO near the carrier. Because of this, it is actually possible to use a PLL just in order to improve the phase noise of an oscillator. One might say that the phase noise of a signal coming out of a PLL ressembles the phase noise of the reference oscillator near the carrier, and the phase noise of the VCO further away from the carrier stays more or less the same. The transition area is given by the bandwidth of the loop.

The PLL is in the following approximated to be linear so that the analysis can be made simpler. The phase noise can then quite easily be estimated at the output and the effects of the different components can be found.

The input phase θ_i is composed of the phase of a perfectly clean reference and the phase noise of the reference:

$$\theta_i = \theta_r + \theta_{ni} \tag{4.22}$$

The phase noise of the reference will hence be multiplied by the same transfer function as the oscillator phase of the reference. On the other hand, the phase noise of the VCO does not propagate through all the components of the PLL, and will therefore not undergo the same transfer function:

The phase noise of the output of the PLL may be written:

$$\theta_o(s) = \frac{K_d K_0 F(s)\theta_i + s\phi_o}{s + K_d K_0 F(s)/N} \tag{4.23}$$

This can be expressed with $H(s)$ and $E(s)$.

$$\theta_o(s) = H(s)\theta_i + E(s)\theta_o \tag{4.24}$$

where $H(s)$ is given in (4.4) and $E(s)$ is given by:

$$E(s) = \frac{s}{s + K_d K_0 F(s)/N} \tag{4.25}$$

The phase noise of the output of the PLL can be approximated by

$$\mathcal{L}(j\omega) = |H(j\omega)|^2 \mathcal{L}_c(j\omega) + |E(f)|^2 \mathcal{L}_o(j\omega) \tag{4.26}$$

where $\mathcal{L}_c(j\omega)$ is the phase noise of the reference and $\mathcal{L}_o(j\omega)$ is the phase noise of the open-loop VCO. The noise that appear in the divider and the phase detector will add to the phase noise of the reference signal and hence also be multiplied by $|H(j\omega)|$. When working with dB, these expressions can be used:

$$\mathcal{L}_{\text{due-to-reference}}(f) = \mathcal{L}_{\text{ref}}(f) + 20\log|H(f)| \tag{4.27}$$

$$\mathcal{L}_{\text{due-to-open-loop-vco}}(f) = \mathcal{L}_{\text{vco}}(f) + 20\log|E(f)| \tag{4.28}$$

The loop filter noise will add to the phase noise like this [17]:

$$\mathcal{L}_{\text{due\_to\_loop\_filter}} = \frac{|E(f)|^2 K_0^2 S_{n\_\text{filter}}}{(2\pi f)^2} \tag{4.29}$$

where S_{n_filter} is the one-sided spectrum of the loop filter noise[1].

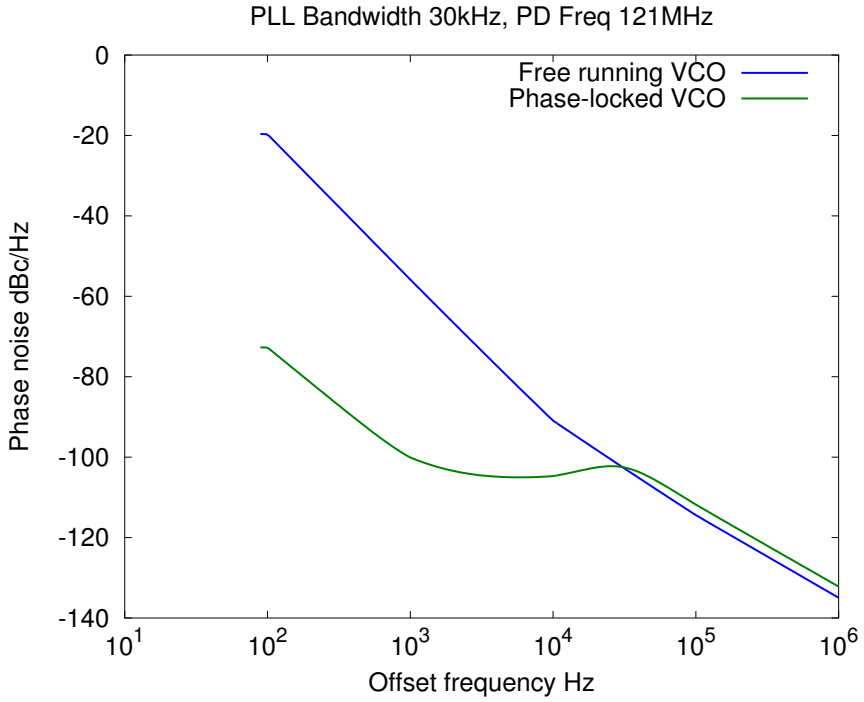


Figure 4.7: *How the PLL will modify the phase noise of a VCO*

Figure 4.7 presents how the phase noise of the output of a PLL typically will change when the reference oscillator is very much cleaner than the VCO near the carrier. The open-loop VCO is in blue, and the closed-loop VCO in green.

[1] This is noise and not phase noise

Chapter 5

Degradation Effects in FMCW Radars

5.1 Phase Noise

The phase noise is the most important degradation effect for FMCW Radars and it plays a much bigger role than for pulse radars. It influences the performance in several ways. The main issue is that the radar transmits and receives simultaneously. The signal will therefore leak from the transmitter directly to the receiver. This will happen through space, and perhaps even more severely if the transmitter and the receiver share the same circuit board, through the substrate. The leakage signal[1] may be so strong that it may saturate either the low-noise amplifier or the mixer. However, a more common problem is that the phase noise that also leaks will have a tendency to mask out targets since the targets are detected at frequency offsets of the carrier. The same is the case for strong nearby clutter.

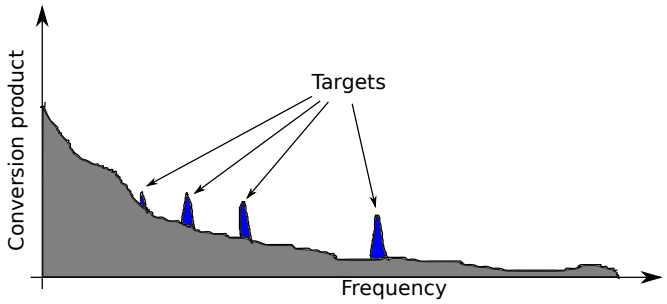


Figure 5.1: *How targets can be masked by the phase noise*

In the mixer of the receiver, the transmit signal is injected so that it can be mixed with the received signal. It can be seen that the phase noise of the transmitted signal and the received signal are to some degree correlated, and that this effect can be exploited in order to improve the performance. This is especially true for the direct

[1]The leakage signal is sometimes called spillover.

leakage at the radar site. In fact, most radars based on FMCW principles would not be very sensitive if this correlation were not exploited.

The correlation can be found by setting up an expression for the phase noise difference $\Delta\phi(t)$ [24]. t_d is the time back and forth to the target:

$$\Delta\phi(t) = \phi(t - t_d) - \phi(t) \tag{5.1}$$

The autocorrelation is then:

$$\begin{aligned} R_{\Delta\phi}(\tau) &= E\{[\phi(t+\tau-t_d) - \phi(t+\tau)][\phi(t-t_d) - \phi(t)]\} \\ &= E[\phi(t+\tau-t_d)\phi(t-t_d)] - E[\phi(t+\tau)\phi(t-t_d)] \\ &\quad - E[\phi(t+\tau-t_d)\phi(t)] + E[\phi(t+\tau)\phi(t)] \\ &= R_\phi(\tau) - R_\phi(t_d+\tau) - R_\phi(\tau-t_d) + R_\phi(\tau) \\ &= 2R_\phi(\tau) - R_\phi(t_d+\tau) - R_\phi(\tau-t_d) \end{aligned} \tag{5.2}$$

By taking the fourier transform of the autocorrelation, the spectrum density is found.

$$\begin{aligned} S_{\Delta\phi}(f) &= \mathcal{F}\{R_{\Delta\phi}(f)\} = 2S_\phi(f) - S_\phi(f)e^{-j2\pi f t_d} - S_\phi(f)e^{j2\pi f t_d} \\ &= 2S_\phi(f)(1 - cos(2\pi f t_d)) \\ &= 2S_\phi(f)(1 - cos(2\pi l f/c)) \end{aligned} \tag{5.3}$$

$$S_{\Delta\phi}(f) = 4S_\phi(f)sin^2(\pi l f/c) \tag{5.4}$$

As can be seen, the spectrum density of the phase noise difference is given as $4\sin^2(\pi l f/c)$ times the spectrum density of the phase noise of the transmitted signal. The factor l is the path length between the two signals. It is only the phase noise difference that plays a role because the receiver mixes the transmitted signal with the received signal, and if the phase noise difference between the two signals is zero, the beat signal will contain no phase noise. If for example the path length l is zero, the phase noise will cancel completely. This can be exploited by making the delay of the transmitter leakage exactly as long as the delay from the transmitter to the mixer. It should also be noted that the effect is greater at low frequency offsets, f. From Equation (5.4) a cancellation factor, C, can be defined as is done in [1].

$$C = 4\sin^2(\pi l f/c) \tag{5.5}$$

Note that this cancellation will only affect the phase noise - not the amplitude noise. In addition, even if there were complete cancellation ($C = 0$), there would be nonlinear elements that transform amplitude noise into phase noise [1] and hence the phase noise will not be completely eliminated. It should also be noted that the factor is periodical along the frequency offset axis. At some offsets it will reduce the phase noise, while at others it will make it higher because of the 4.

If there are targets with strong reflection, the phase noise that is also reflected will have a tendency to mask out weaker targets. This could also cause problems. It is therefore very important to make the phase noise of the transmitter as low as possible.

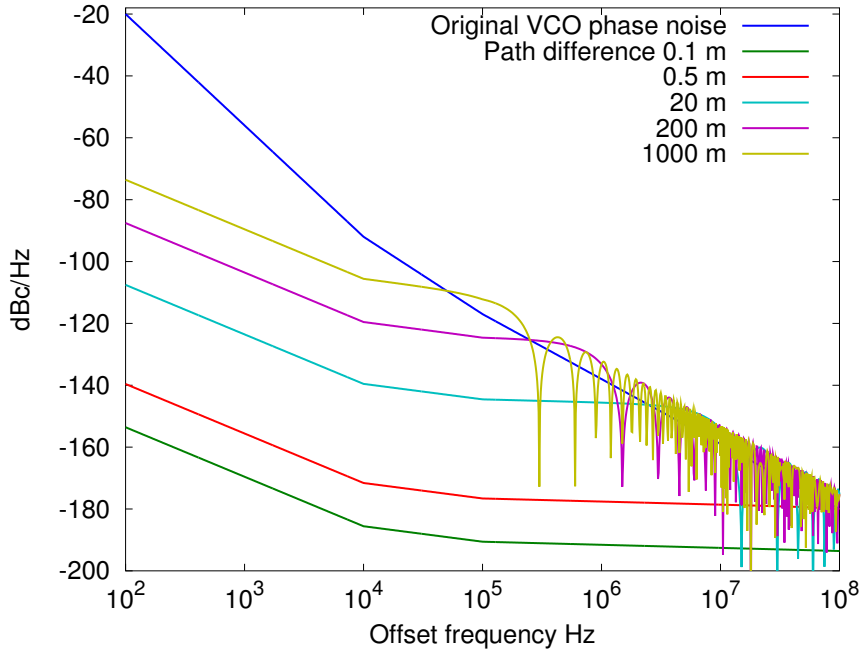


Figure 5.2: *Cancellation effect of phase noise because of correlation. The oscillator is the open-loop VCO used in the design (Table 6.2 page 61).*

Figure 5.2 presents the cancellation effect for different path length differences. It can be seen that the periodicity of the cancellation factor is first really having an effect for targets far away. If the path length error in the transmitter is 10 cm, the phase noise is very much cancelled as indicated by the green line and it may even be so good that the noise spectrum will be dominated by thermal noise (not indicated in the figure).

5.1.1 Contribution to the unaccuracy of the distance estimation

In addition to having a tendency to mask out the signal, the phase noise will also contribute to the unaccuracy of the signals that are detected. This will lead to a uncertainty in the range and bearing measurements that can be expressed as a variance. The frequency estimation error of an FFT when there is a systematic phase error has been shown to be [25]:

$$\delta\omega \cong \frac{\sum_{n=0}^{N-1} \delta\varphi[n] \cdot (\Delta t \cdot n - t_0) w(\delta t \cdot n - t_0) \Delta t}{\int_{-\infty}^{\infty} t^2 w(t) dt} \tag{5.6}$$

This is based on a deterministic phase error, and must hence be adapted to the stochastic case with noise variance. The FFT window function is represented by the

$w(t)$. It is expected that the same cancellation effect, C, will play a role also for the accuracy. That means, that at closer ranges, the phase error will not be that big as expected from the phase noise spectrum.Clearly, there are many factors influencing the measurement accuracy and not just phase noise. The receiver noise performance is also among them.

When the target is not moving, the distance error due to phase error can be found from (5.6) and (2.8) to be:

$$\delta r \cong \frac{c \sum_{n=0}^{N-1} \delta\varphi[n] \cdot (\Delta t \cdot n - t_0) w(\delta t \cdot n - t_0) \Delta t}{2\alpha \int_{-\infty}^{\infty} t^2 w(t) dt} \tag{5.7}$$

where α is the sweep rate and c the speed of light.

5.2 Linearity and Quantization of Sweep

The sweep that is generated should be very linear and clean. If it does not follow a linear curve, the range resolution will be reduced as pointed out in [26]. When a DDS is employed to generate the sweep, the main issue is however not nonlinearity, but quantization. The sweep consists of many small steps. These will in some cases influence the capability of the radar, but as we will see, if everything is done correctly, they will play no big role. [11, p. 42-45]

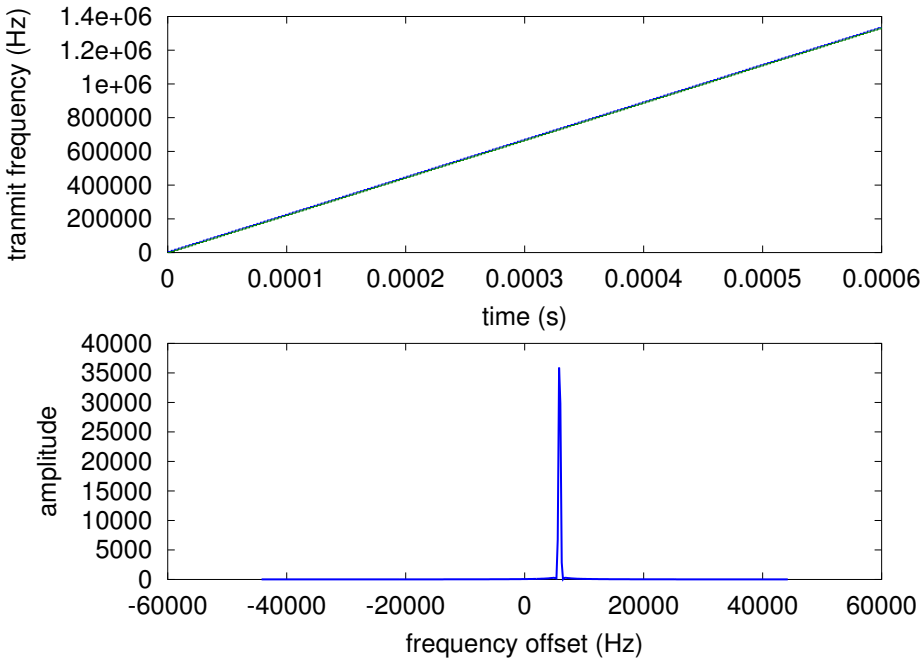


Figure 5.3: *Beat signal when using unquantized sweep*

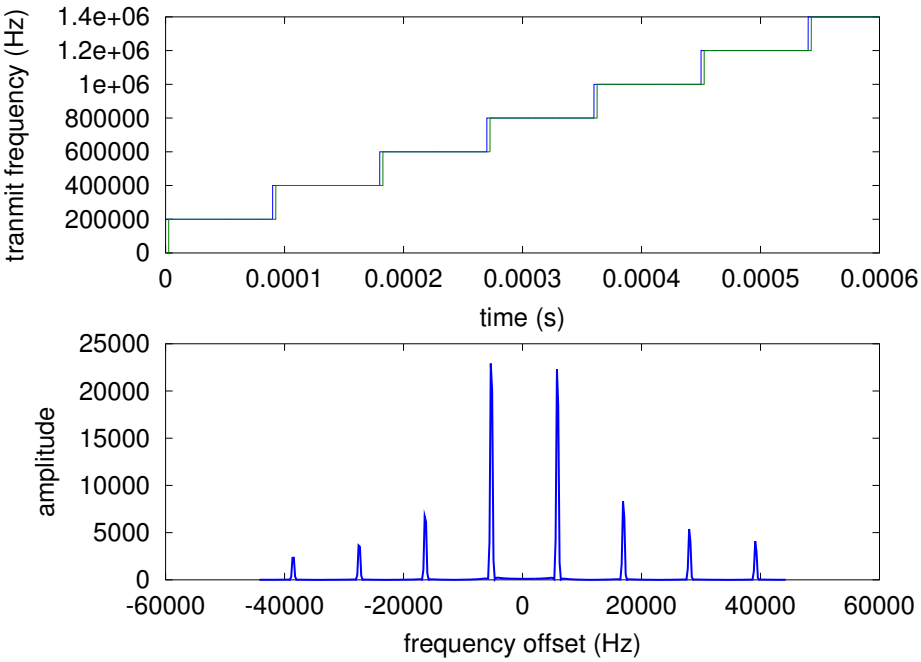


Figure 5.4: *Beat signal when using big steps*

A simulation is performed. A sine wave is swept from 2 MHz and up to 12 MHz with a sample rate of 33.33 MHz (Appendix C.4). Quadrature mixing with a delayed version of the sweep is performed. The beat signal is then low-pass filtered and passed through an FFT. By taking the absolute value of the FFT, the consequence of the step quantization is investigated. Figure 5.3 presents the case then we have a perfect sweep with no quantization. A single peak appears in the spectrum and it will correspond to the signal delay and hence the ”distance”.

If clearly visible steps are made while sweeping, the spectrum looks completely different (figure 5.4). There will be many peaks at regular intervals and it is observed that there can be ambiguities. If the delay is increased, all the peaks will move to the right. At a certain delay, one of the ”false” peaks takes the role of the main peak, and we will not know which of them correspond to a real distance. So by introducing steps in the sweep, there will be a maximum unambiguous range. If the steps are made just half as long, the distance between the peaks doubles as can be observed in Figure 5.5. The reason for that the spectrum is non-symmetrical around 0 Hz, is because quadrature mixing is performed. In the case with normal mixing, the two sides of the spectrum will be mirrored. That means that the first peak at negative frequencies around -16000 Hz will appear at positive frequencies too. This would cause the effective maximum unambiguous range to be just the half. Half of the peaks would move to the left as the delay increased, and half of them to the right.

It can be seen that the distance between every peak is given by the step rate. So in for example Figure 5.4 there is one step every $90\mu s$. This corresponds to a step rate of $f_{\text{step\_rate}} = 1/(90 \cdot 10^{-6}) = 11.11kHz$ which is the distance between every peak in the spectrum.

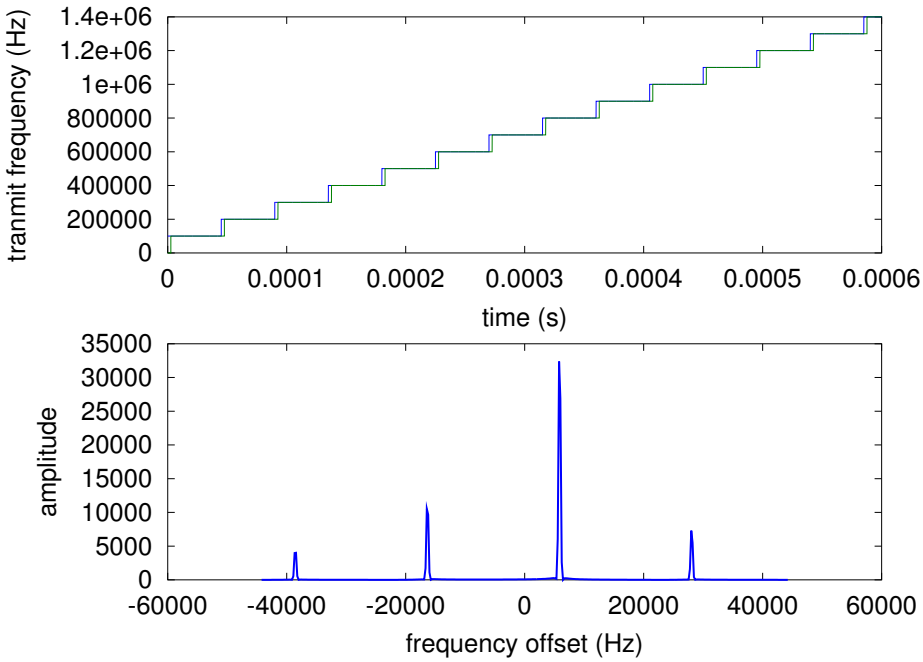


Figure 5.5: *Beat signal when using smaller steps*

$$f_{\text{distance}} = f_{\text{step\_rate}} \tag{5.8}$$

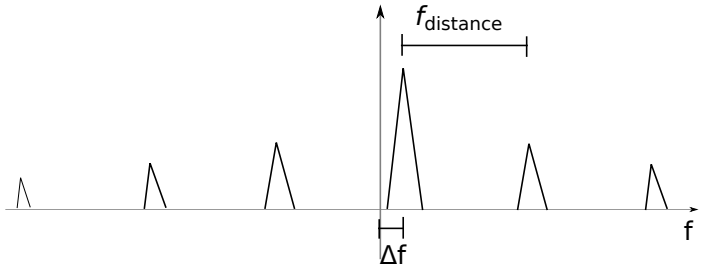


Figure 5.6: *How the step rate influences the spectrum of the beat signal*

In the case of an oscillator in a PLL, it has been observed that the step response will normally have an overshoot before it settles. A simulation of a sweep with that type of *un-ideal steps* was also performed. The result can be seen in Figure 5.7. It looks very similar to the result for ideal steps in Figure 5.4, but the amplitude of the peaks for positive frequencies are higher than for negative frequencies.

The simulations above were performed with an FFT running at a quite high sample rate. This gives us the complete spectrum. If however, the FFT calculates with samples of a much lower sample rate, it will be observed that the peaks that are not wanted fall outside the sampled spectrum. There will typically be a low-pass filter in front of the AD converter which removes the unwanted peaks. In the case when one sample is taken for each step in the sweep, it will be seen that the maximum

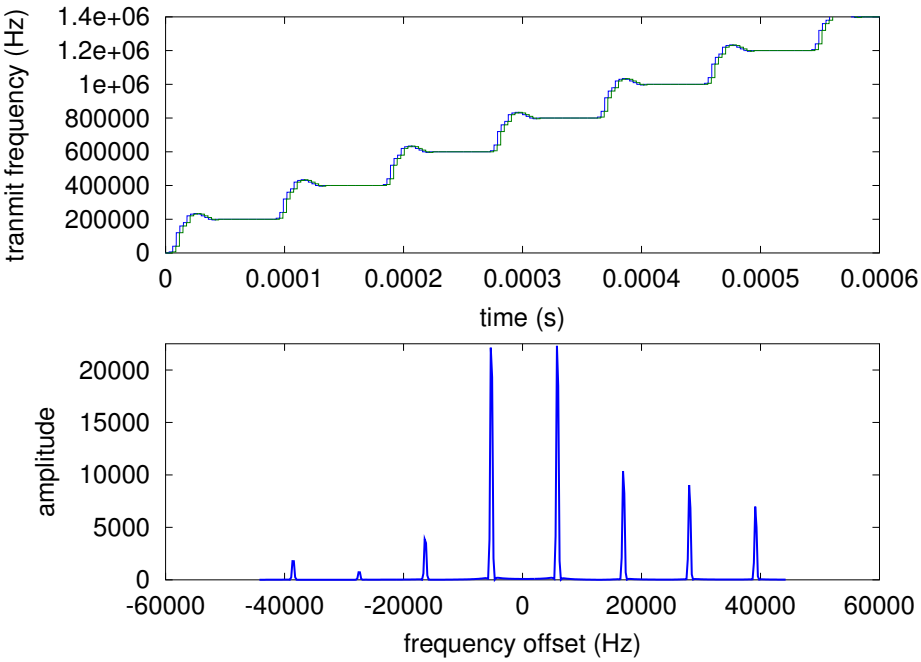


Figure 5.7: *Beat signal when using practical steps*

unambiguous range as a consequence of the low-pass filter before the sampling is the same as the maximum unambiguous range because of the stepping. So the conclusion will be that the steps play no big role as long as not more samples are taken than the number of steps.

However, there is one thing that should be noticed. The not-wanted peaks steal some of the energy from the ideal peak. In Figure 5.3, the amplitude is about 35000, while it is just 23000 in Figure 5.4. It basically says that the radar sensitivity can be a bit reduced if this is not taken into consideration.

Chapter 6

Topology

Different topologies can be used for the synthesizer of a linear FMCW-radar signal. What is needed is a linear sweep from lower frequencies to higher frequencies. This can be implemented with a VCO. If a linear ramp-generator is made and connected to the tuning input of a VCO, we have what is needed (Figure 6.1).

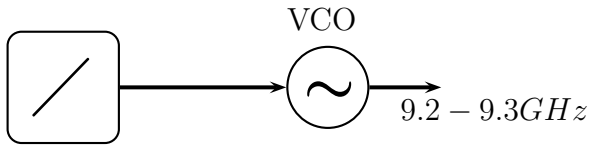


Figure 6.1: *Simple illustration of the concept for generating a sweep*

6.1 Possible Solutions

6.1.1 Open loop

In order to make the ramp, an analogue circuit can be constructed with an op-amp and a trigger circuit. It is also possible to use a digital to analogue converter (DAC) so that the ramp can be controlled digitally. One considerable problem with this method is that the voltage-to-frequency characteristic of the VCO is never completely linear. It will often be steep at low parts of the tuning range and get more flattened as the voltage increases. This will lead directly to nonlinearities in the frequency sweep when the voltage sweep is linear. It is possible to cope with this problem when a DAC is used. If the inverse of the tuning response nonlinearities is programmed into the sweep, the result will be linear. This is a kind of predistortion. Still, there remains a problem which is that this is an open-loop configuration. What actually happens at the output is not monitored and consequenctly if changes happen to the circuit or the environment (as will happen), like for instance a minor temperature change, the sweep will no longer be linear. For example, some feedback

could be added which monitored the linearity and then adjusted the predistortion algorithm accordingly. However, this is not any fast and still not very elegant. There are methods which proove to be much more elegant and which work very much better.

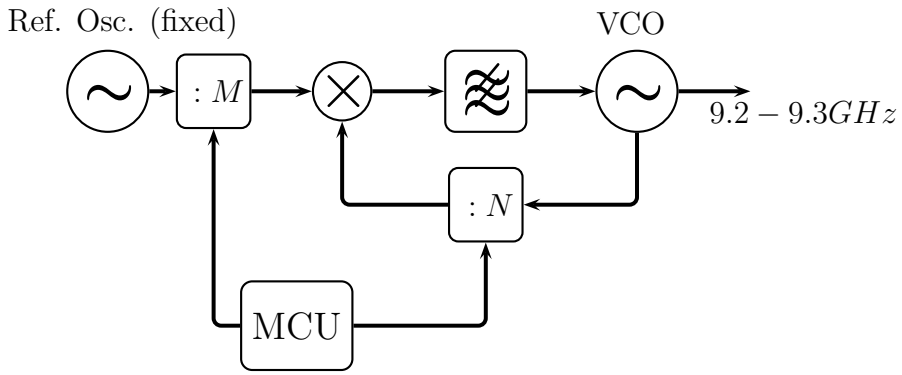


Figure 6.2: *A sweep generator using a PLL with variable prescaler*

6.1.2 PLL with variable prescaler

First of all, by employing a PLL, it was seen in Chapter 4 that phase noise can be reduced while the carrier is locked to a reference oscillator. If this is employed, we will be able to stabilize the output very well at the same time as the phase noise is reduced. This is also a feedback system, but instead of that the feedback generates updates only now and then like for the previous method, the feedback now provides continuos monitoring of the output signal and its frequency. The main issue now is to decide how the frequency changes needed for the sweep can be generated.

Consider Figure 6.2. This is a PLL with division of both the reference oscillator and the VCO. By doing it this way, it is possible to adjust the output frequency in steps as large as the phase-detect frequency. By keeping M fixed and adjust N repetedly, a sweep will be made. This configuration normally requires a low phase-detect frequency so that the steps can be small. It was seen in Chapter 5 that the length of the steps directly influences the maximum range.

One problem with this topology is that such a small phase-detect frequency is needed, and if the operating frequency is very high (like 9GHz in our case), the divisor N must be very high. This can make it harder to achieve good phase noise. Another consideration is that after each step, the phase will not be continuous, and this will lead to that basically the PLL falls out of lock and quickly has to reacquire lock at each step. Since the frequency steps are not that big, it is not very complicated to reacquire the lock. It will happen fast, but the process leads to more noise appearing.

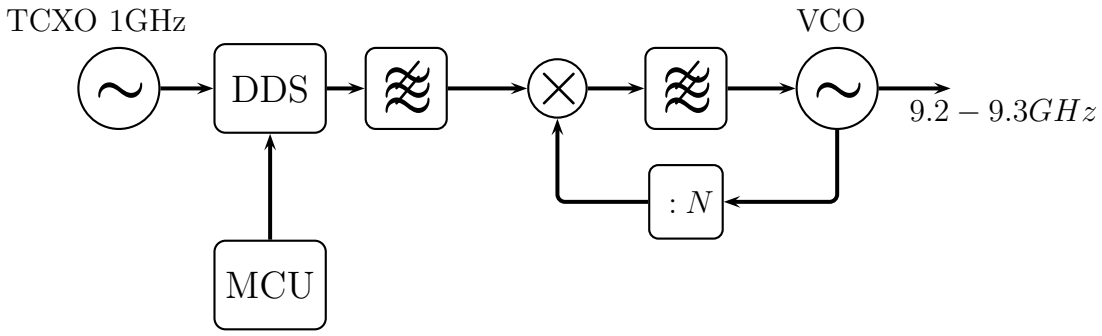


Figure 6.3: *Using a DDS as the reference to the PLL. The frequency of the VCO output is divided while the DDS output is only filtered.*

6.1.3 DDS as the reference oscillator for the PLL

Figure 6.3 shows another very interesting configuration. Here a DDS (Direct Digital Synthesis, see Appendix D) circuit is employed. The need for a very linear sweep is very often nowadays accomodated with the use of a DDS. It lets us output all the frequencies we want up to a frequency of about 40% of the frequency of its reference oscillator, and it can be controlled directly digitally. An MCU will set the configuration parameters of the DDS, and the output signal will then be used as the reference of the PLL. This provides a number of advantages. DDS circuits normally have continuous phase which implies that the frequency steps can be accepted by the PLL without loosing lock. The step response will be more or less exactly as calculated with the Equations in Section 4.4. Furthermore, it is possible to make the DDS produce very small steps and adjust them precisely.

The output frequency will be given by N times the DDS generated frequency. The size of the steps at the output of the DDS will correspondingly be scaled by N. Still the steps at the VCO can be made remarkably small as the DDS steps may be set as small as a fraction of a Hz!

The transfer function of the PLL in this configuration is the same as in Chapter 4 because only the VCO frequency is followed by a divider.

$$H(s) = \frac{\theta_o(s)}{\theta_i} = \frac{K_d K_0 F(s)}{s + K_d K_0 F(s)/N} \tag{6.1}$$

As we saw in that same chapter, the phase noise near the carrier will mainly be given by the frequency of the reference oscillator scaled by N, and that the phase noise further away will be given by the VCO itself. The near-carrier phase noise will therefore normally be made better than when the VCO runs freely. The focus on choosing a VCO with low near-carrier phase noise, is therefore not that important, but the phase noise further away from the carrier must be good (the two normally follow each other though). At the same time, the output of the DDS must be

provided with very low near-carrier phase noise. This is often the main issue. The noise will be scaled by N which makes it even harder. All in all, this configuration is a very much employed one.

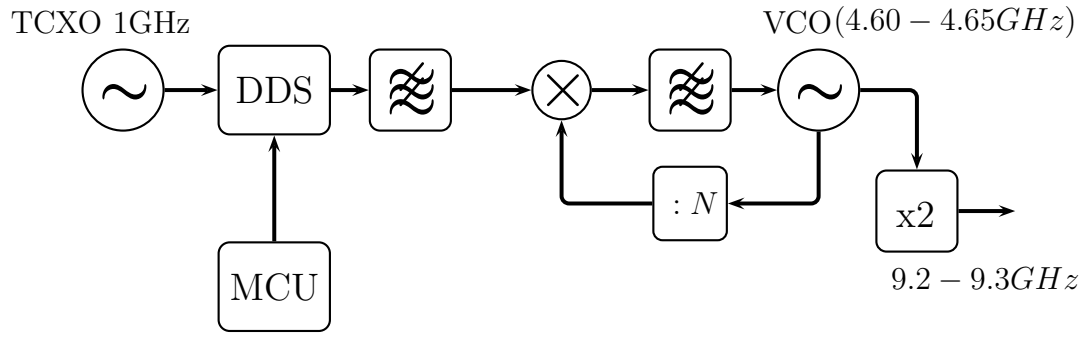


Figure 6.4: *Equal to the previous one just with a frequency doubler at the output.*

With frequency doubler

Figure 6.4 shows another topology that can be used. It is different because it uses an additional frequency doubler after the VCO. The topology is not very much different. Some VCOs are even implemented with a frequency doubler internally. It must be used if the operating frequency is so high that good oscillators cannot be found. When a frequency doubler is used afterwards, it must be ensured that the doubler does not add much more noise. The phase noise will, as we have seen earlier, be scaled by the factor N for frequency multiplication, and since here N is 2, the phase noise will be scaled by 2, or $20 \log 2$ dB.

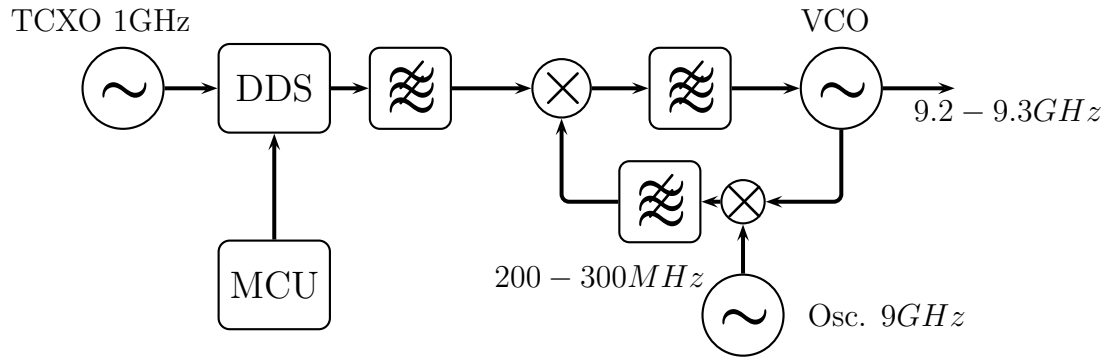


Figure 6.5: *PLL having frequency conversion in the loop.*

PLL with frequency conversion within the loop

Let us have a look at some other topologies too. Figure 6.5 is a variation of the former topology in that it uses down-conversion in the loop [15]. Yet another fixed frequency oscillator is needed for this. The result of frequency-conversion is different from the prescaler. The phase noise will not be scaled down, but it will be added just as if nothing happened to it.

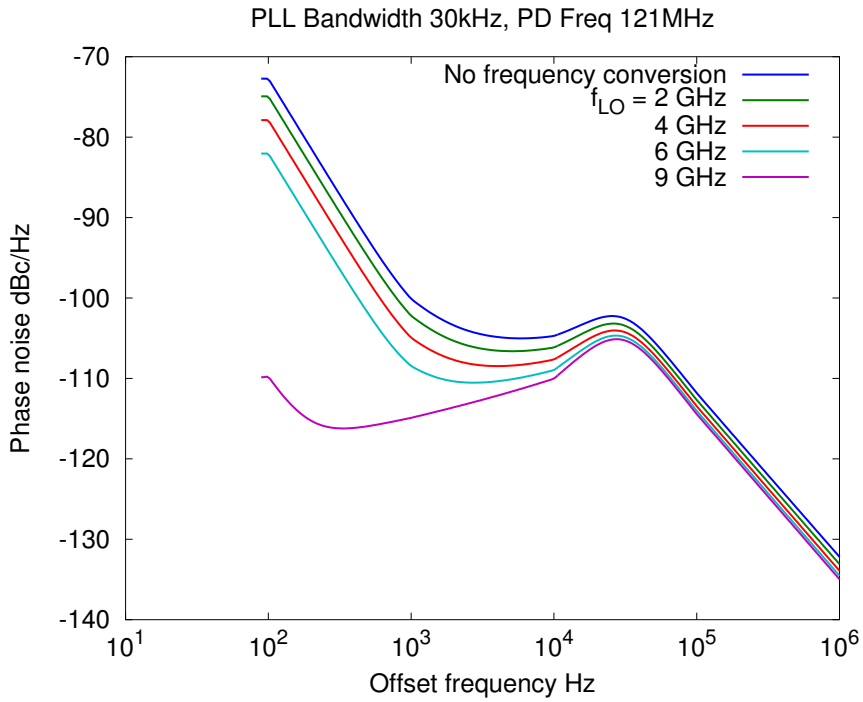


Figure 6.6: *Predicted phase noise for the topology with frequency conversion in the loop and an ideal oscillator*

A Laplace expression can be made for this case to see how it differs. To make the case more general, a prescaling factor, N, is included between the mixer and the phase detector.

$$\theta_o(s) = \frac{K_d K_0 F(s)}{s}\left(\theta_i - \frac{\theta_o(s) + \theta_{\mathrm{LO}}}{N}\right) \tag{6.2}$$

$$\theta_o(s)\left(1 + \frac{K_d K_0 F(s)}{sN}\right) = \frac{K_d K_0 F(s)}{s}\theta_i - \frac{K_d K_0 F(s)}{sN}\theta_{\mathrm{LO}} \tag{6.3}$$

$$\theta_o = \frac{K_d K_0 F(s)}{s + K_d K_0 F(s)/N}(\theta_i - \frac{\theta_{\mathrm{LO}}}{N}) \tag{6.4}$$

The transfer function for the two different sources of noise will be given by

$$H(s) = \frac{\theta_o(s)}{\theta_i} = \frac{K_d K_0 F(s)}{s + K_d K_0 F(s)/N} \tag{6.5}$$

$$H_{\mathrm{LO}}(s) = \frac{\theta_o(s)}{\theta_{\mathrm{LO}}} = \frac{K_d K_0 F(s)/N}{s + K_d K_0 F(s)/N} \tag{6.6}$$

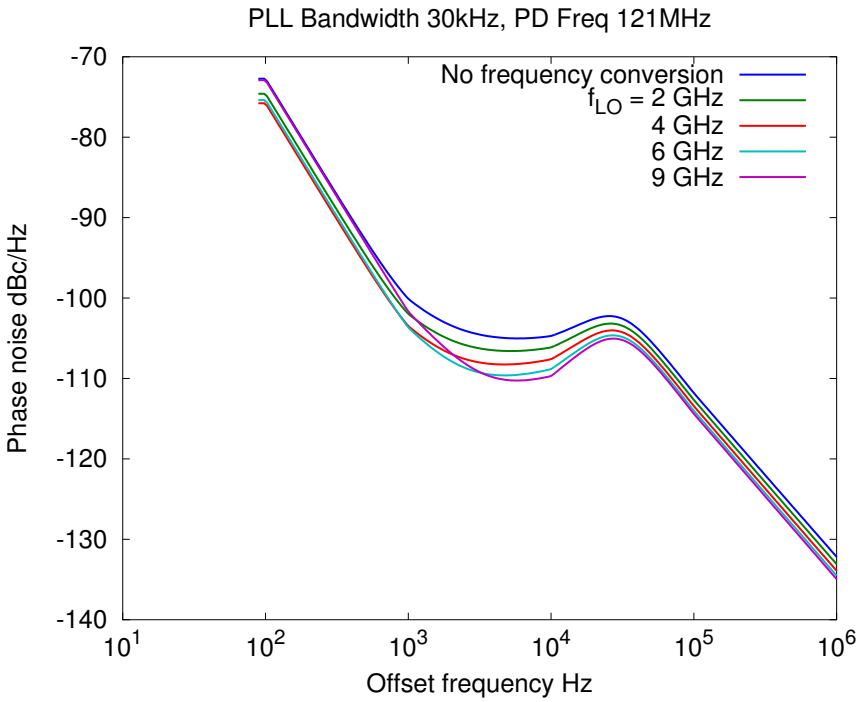


Figure 6.7: *Predicted phase noise for the topology with frequency conversion in the loop with an oscillator with equal phase noise performance as the TCXO*

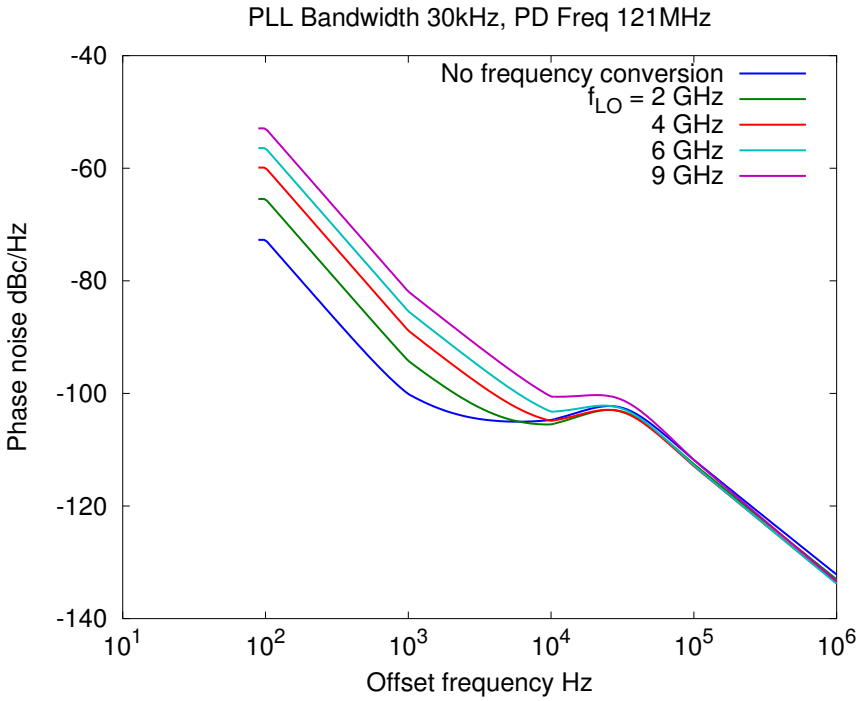


Figure 6.8: *Predicted phase noise for the topology with frequency conversion in the loop with an oscillator 20 dB worse than the TCXO*

This means that the the noise from the second oscillator will add directly to the output. It will not be scaled by the N factor as the reference signal will. On the other hand, since the N factor will be much lower than for the previous topology, the noise from the reference will no longer be scaled as much up. This will be an advantage. It has to be weighed against the contribution of the noise from the second oscillator. All in all, this topology can be very good. The main disadvantage is probably that another high quality oscillator is needed along with a good mixer and a low-pass filter. This makes it quite a bit more expensive. Furthermore, the DDS must be swept over a larger bandwidth since its steps are not scaled by a big N, and consequently it might be harder to eliminate spurs from it.

Figure 6.6 presents a comparison between the topology with frequency conversion in the loop and without (for code see Appendix C.1). Equation (6.5) and (6.6) are used in the calculation. In this case, the local oscillator has been considered ideal, thus not contributing any noise. It can be seen that the total phase noise of the PLL decreases as the frequency conversion frequency gets higher. It is observed that there is definitely something to be gained by using frequency conversion. If the oscillator is no longer considered ideal, as in figure 6.7 , the phase noise reduction becomes much lower. In this case, a local oscillator with the same phase noise performance as the TCXO (which is very good) scaled by $f_{\mathrm{LO}}/f_{\mathrm{TCXO}}$ (see Section 3.5) has been used. In the case when the local oscillator, or the combination of the local oscillator and the mixer, have 20 dB more phase noise than the TCXO for all offset frequencies, the phase noise of the output becomes much worse when frequency conversion is added (Figure 6.8).

All in all it may be concluded that the gain achieved with frequency conversion depends heavily on the performance of the oscaillator, and it must be judged if it is worth the cost.

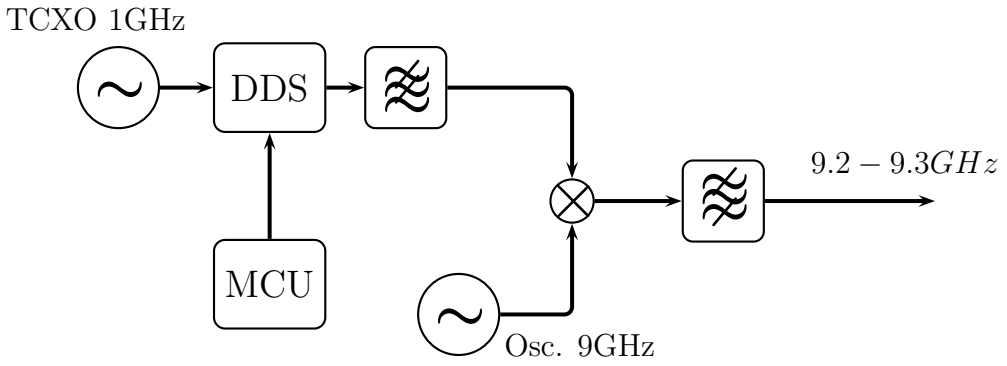


Figure 6.9: *Direct mixing of the DDS signal with a fixed frequency oscillator.*

6.1.4 DDS output mixed directly with fixed oscillator

The topology seen in Figure 6.9 is different from the previous because it does not necessarily use a PLL. The output of the DDS is mixed directly with the output of a fixed frequency oscillator. In this case, the phase noise of the output will be given as the sum of the phase noise of the two oscillators. Since they both can be made very good, the total phase noise can be made very good. A very important consideration for this topology is that the bandpass filter following the mixer must be made very good. If it is not, the image frequency will also propagate through. That is, if a 9 GHz fixed oscillator is used and the DDS outputs at 200MHz, there will be an output at both 8.8 GHz and 9.2 GHz. The filter must have a very high attenuation at 8.8 GHz if the desired output is at 9.2 GHz. The mixer must also be made good so that it does not add much phase noise.

Another thing to mention with this topology is that the sweep from the DDS must be made much wider. It will not be scaled anything as in the previous configurations. To get a sweep from 9.2 to 9.3 GHz, the DDS must sweep from 200 MHz to 300 MHz. For the topology in Figure 6.3 using a division factor of for instance 100, the sweep would be only from 92 MHz to 93 MHz.

To get a very good phase noise performance of the last configuration, the fixed frequency oscillator should be phase locked to an external reference. A TCXO has a very good near-carrier phase noise, but it is often worse than a VCO at offsets further away from the carrier. By locking the VCO to an external TCXO, we will get the best from the two devices as we have seen in the previous chapter. It can for example be locked to the same TCXO as is used as the reference of the DDS in order to save money. This implies that the VCO should run at an integer-multiply frequency of the DDS reference.

6.2 Chosen Topology

It was decided to implement the topology with a DDS as a reference and without frequency conversion (Figure 6.3). This was done because it has many good properties and still the design is pretty straight forward. No expensive filters or mixers are needed. The phase noise performance is quite good, but it will depend on the frequency conversion frequency that is used.

The VCO must by itself provide low phase noise. This is especially true for offsets far away from the carrier since there the noise will not be attenuated by the PLL. It is also important that the VCO has the desired tuning range, and that the tuning range is not too large for the application. If it is, the noise coming in through the tuning input will play a bigger role. The VCO should also have a reasonable strong output so that not too much amplification is needed afterwards which will further degrade the phase noise. Table 6.1 presents different VCOs capable of outputting 9.2 GHz directly. Their performance were compared, and the oscillator HMC510 from Hittite Microwave Technologies was chosen. Its phase noise performance is presented in Table 6.2.

Company	**Name**	**Freq**	V	V_{tune}	**dBm**	**10kHz**	**100kHz**
Hittite	HMC734LP5	8600-10200	5.0	1-13	18	-70	-100
Hittite	**HMC510LP5**	**8450-9550**	**5.0**	**0-15**	**13**	**-92**	**-116**
Hittite	HMC511LP5	9050-10150	5.0	2-13	13	-88	-115
Hittite	HMC587LC4B	5000-10000	5.0	0-18	5	-65	-95
RFMD	RFVC1834	9000-10200	5.0	2-12	8	-90	
RFMD	RFVC1833	8300-9700	5.0	2-12	8	-90	
RFMD	RFVC1800	8000-12000	5.0	0-13	4	-66	
RFMD	RFVC1801	5000-10000	5.0	0-18	4	-72	
RFMD	RFVC1800C	8000-12000	5.5	0-13	4	-66	
Synergy	DXO900965-5	9000-9650	5.0	0-12	-3	-80	-100

Table 6.1: *Comparison of different VCOs capable of 9.2-9.3 GHz*

Frequency offset	**dBc/Hz**
100Hz	-22
1kHz	-55
10kHz	-92
100kHz	-115
1MHz	-135

Table 6.2: *SSB Phase noise of the HMC510 from Hittite Microwaves [27]*

Frequency offset	**dBc/Hz**
100Hz	-92
1kHz	-121
10kHz	-141
100kHz	-147
1MHz	-150

Table 6.3: *SSB Phase noise of the VFTX210 from Valpey Fisher [28]*

Frequency offset	**dBc/Hz**
1kHz	-147
10kHz	-150
100kHz	-152

Table 6.4: *SSB Phase noise of the AD9858 DDS at f_{out} = 103 MHz [29]*

To be able to use a high phase detector frequency, a DDS capable of outputting a signal of some hundred MHz was needed. The AD9858 from Analog Devices (Appendix E.3) is one such device. It uses a maximum reference oscillator of 1 GHz, which means that it can output signals up to 400-450 MHz (40-45% of the reference). The phase noise contribution of the DDS is given for an output frequency of 103 MHz (Table 6.4), and it is scaled to the frequency of interest like this:

$$\mathcal{L}_{\text{dds}} = \mathcal{L}_{103\text{ MHz}} + 20\log\frac{f}{103\text{ MHz}} \tag{6.7}$$

according to Equation (3.49) page 30.

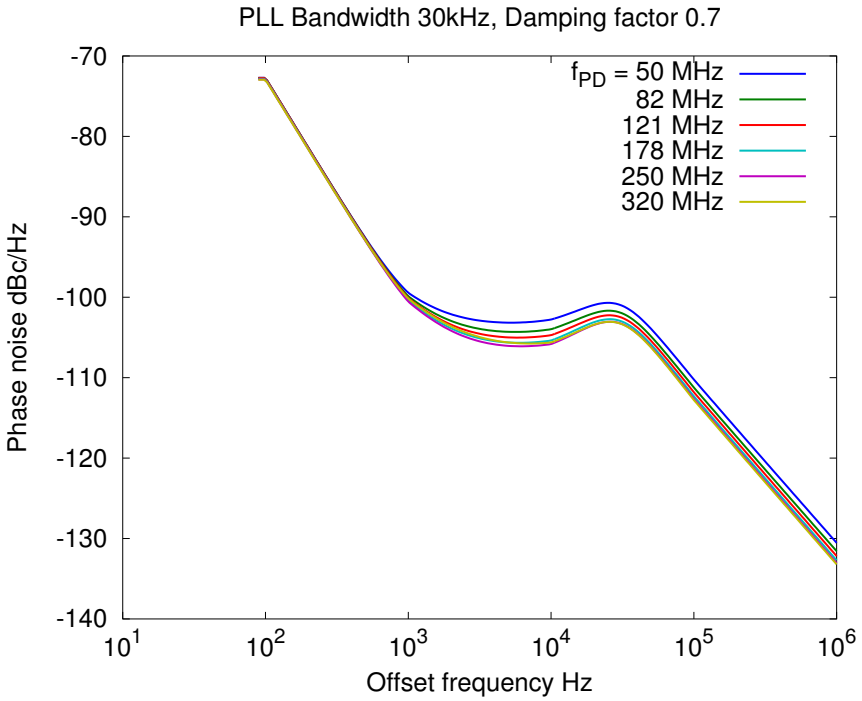


Figure 6.10: *Predicted phase noise for different phase detector frequencies*

The phase noise of the frequency reference sets the lower limit of the phase noise inside the loop bandwidth. Because of this, it was important to find a good one. There are quality oscillators using different technologies. One technology is the Temperature Compensated Crystal Oscillator (TCXO). They typically employ a thermistor network in order to generate a correction voltage which reduces the frequency variations over temperature. A more expensive type is called an Oven Controlled Crystal Oscillator (OCXO). This one uses heating in order to keep the oscillator at a constant temperature. It requires a warm-up period since it operates at temperatures above room-temperature. However, it drifts less than the TCXO.

The DDS may use a reference clock frequency up to 1 GHz, and because the full span of the DDS was desired, a reference oscillator of 1 GHz was needed. A TCXO from Valpey Fischer called VFTX210 was chosen as the reference. The main reason for this was that such an oscillator was already available and could be borrowed from Kongsberg Seatex AS. High performance crystal oscillators are quite expensive.

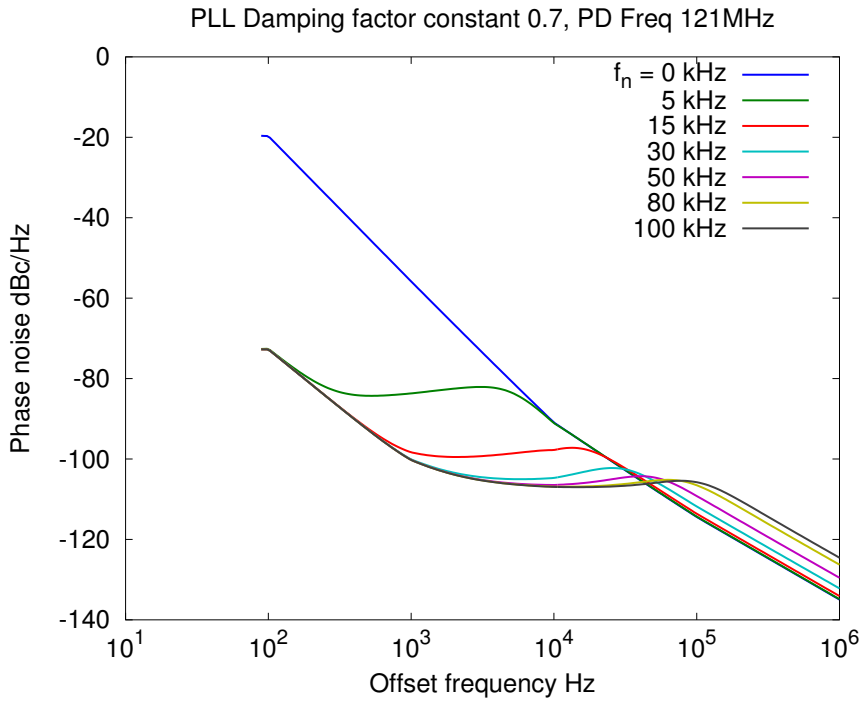


Figure 6.11: *Predicted phase noise for different loop bandwidths*

6.2.1 Phase noise predictions

Using the equations in Section 4.5 and the data for the phase noise of the different components, the phase noise was plotted for different situations. All the plots include phase noise effects from the VCO, the TCXO, the DDS, the phase detector, and the loop filter.

First figure 6.10 presents different phase detector frequencies. It is observed that the total phase noise is reduced as the frequency gets higher, or N lower. There is a great improvement from 50 MHz to 100 MHz, so lower frequencies should not be used. It was decided to use a divisor of 76, giving an output of 121.053 MHz to 122.368 MHz. The images and harmonic spurs of the DDS for these output frequencies are given in Table 6.5.

Number	MHz
1	877.6 - 878.9
2	755.3 - 757.9
3	632.9 - 636.8
4	510.5 - 515.8
5	388.2 - 394.7
6	265.8 - 273.7
7	143.4 - 152.6
8	21.0 - 31.6

Table 6.5: *Main image frequencies and harmonic spurs of the DDS with an output at 121 MHz. (see Appendix D)*

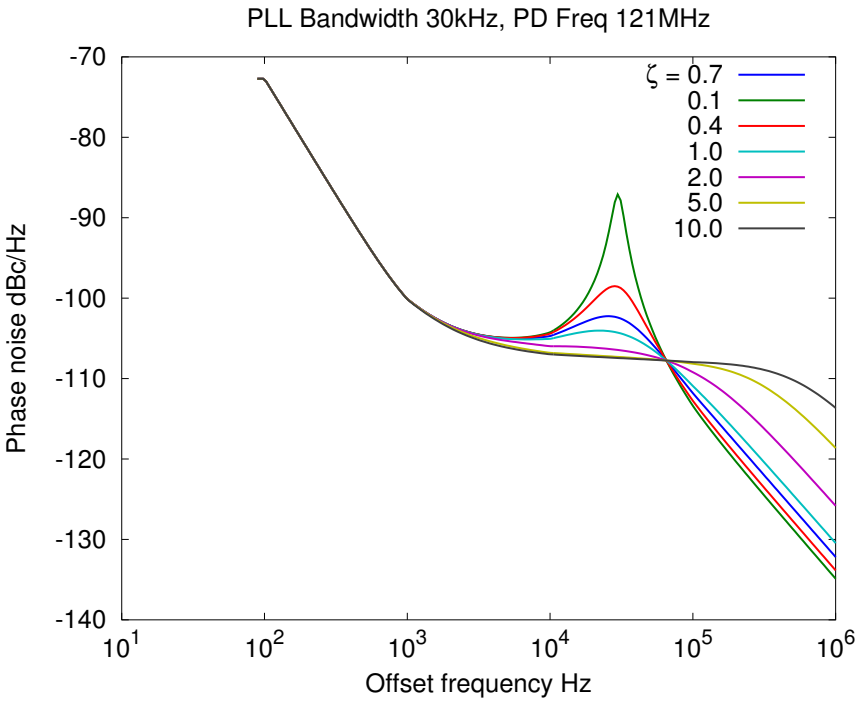


Figure 6.12: *Predicted phase noise for different damping factors*

Figure 6.11 present the effect of changing the loop bandwidth. It is observed that it influences the phase noise spectrum quite much. The damping constant is kept constant, while the loop bandwidth is changed from 0 to 100 kHz. When it is 0, the phase noise spectrum is only affected by the VCO, and hence the spectrum is the same as for the VCO in an open-loop condition (Table 6.2). It is observed that for very high loop bandwidths, the phase noise inside the loop bandwidth is lower, but it decreases more slowly at higher offsets. The point where the phase noise of the reference scaled by N is equal to the phase noise of the VCO, all the curves almost pass through. By choosing the loop bandwidth as a frequency near this point, both the phase noise inside the loop bandwidth and outside the bandwidth can be made low. A loop bandwidth of about 20 - 30 kHz seems to be a good choice for a total low phase noise.

For the case when the bandwidth is kept constant and the damping factor is adjusted, the phase noise influence can be seen in Figure 6.12. The lower the damping factor, the higher peak there will be near the loop bandwidth frequency. This is not desired, so the damping factor must be chosen high enough. When the damping is very high, the phase noise outside the loop bandwidth decreases much more slowly. It is concluded that there is a lot to be gained from the inside bandwidth phase noise by going from $\zeta = 0.1$ to $\zeta = 1.0$, but at increasing values of ζ, there is not much of a decrease. The ideal choice seems to be between 0.7 and 1.5.

6.2.2 Bode diagram

The Bode diagram of the open-loop transfer function of the PLL with loop bandwidth $f_n = 24$ kHz, damping factor $\zeta = 0.78$ and phase detector frequency $f_0 = 121$ MHz is shown in figure 6.13. Here the phase margin can be found to be about 65°. The Bode diagram of the closed-loop transfer function is presented in figure 6.14. This is $|H(jw)|$, what the frequency noise of the reference oscillator is scaled with. It is observed that at low frequencies, the amplification is about 38 dB. That is the same as $20 \log N$, where N is the divisor in the loop.

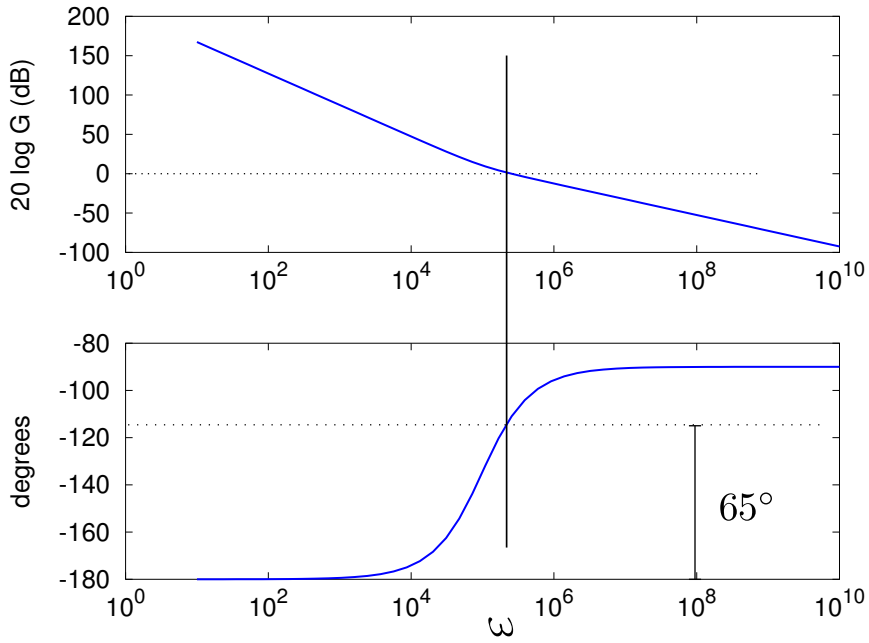


Figure 6.13: *Bode diagram of the open loop response of the PLL*

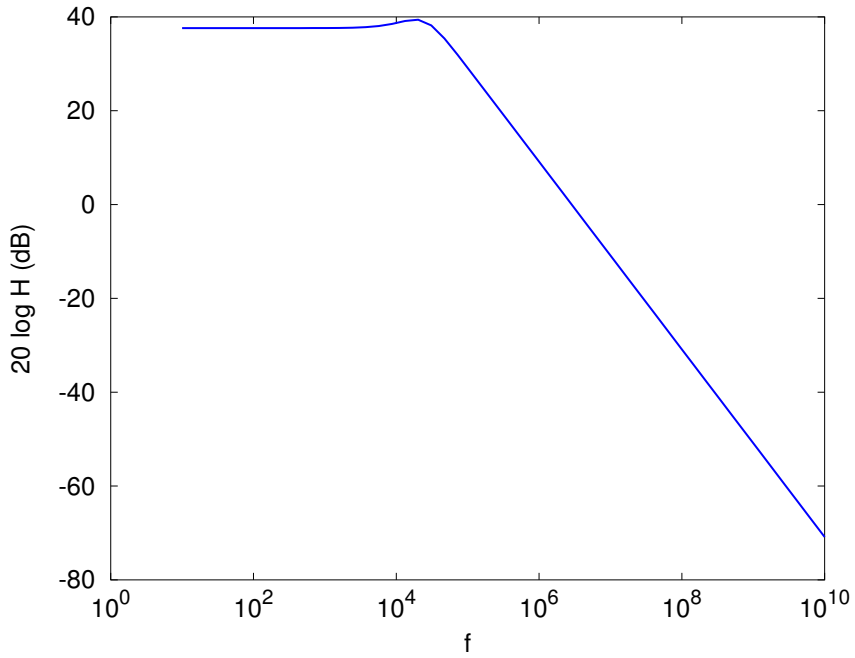


Figure 6.14: *Bode diagram of the closed loop response of the PLL*

Chapter 7

Circuit Design

A synthesizer for a linear FMCW radar in the 9.2-9.3 GHz band was to be designed and built. The requirements were mainly low phase noise and the capability of a linear sweep. Figure 7.1 presents the chosen topology - now with component names included. The choice of the VCO, the DDS and the VCO was presented in the previous chapter. This chapter presents the other components and how the whole circuit was designed.

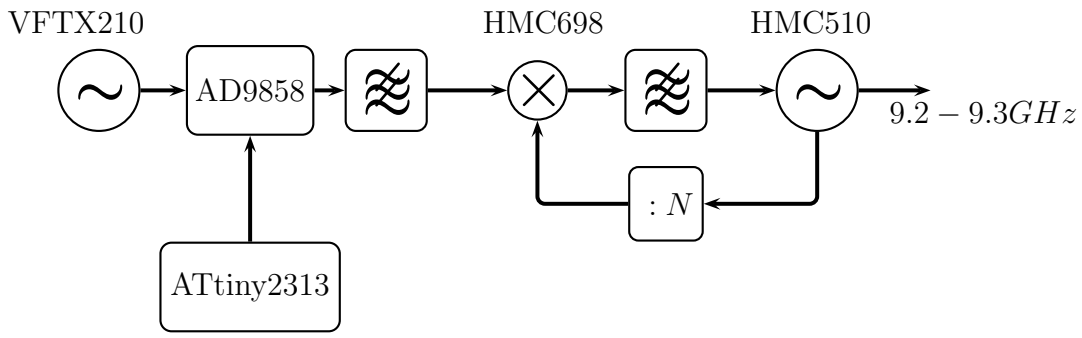


Figure 7.1: *Block diagram of the design*

7.1 Components

7.1.1 VCO

The VCO (HMC510) comes in a 32 lead 5x5 mm SMT package [27]. It is powered with 5 V and draws nominally 315 mA. The power output is claimed to be between +10 and +15 dBm at the operating frequency 8.45 - 9.55 GHz. There are also a divide-by-2 and divide-by-4 output. Only the divide-by-4 was used to provide

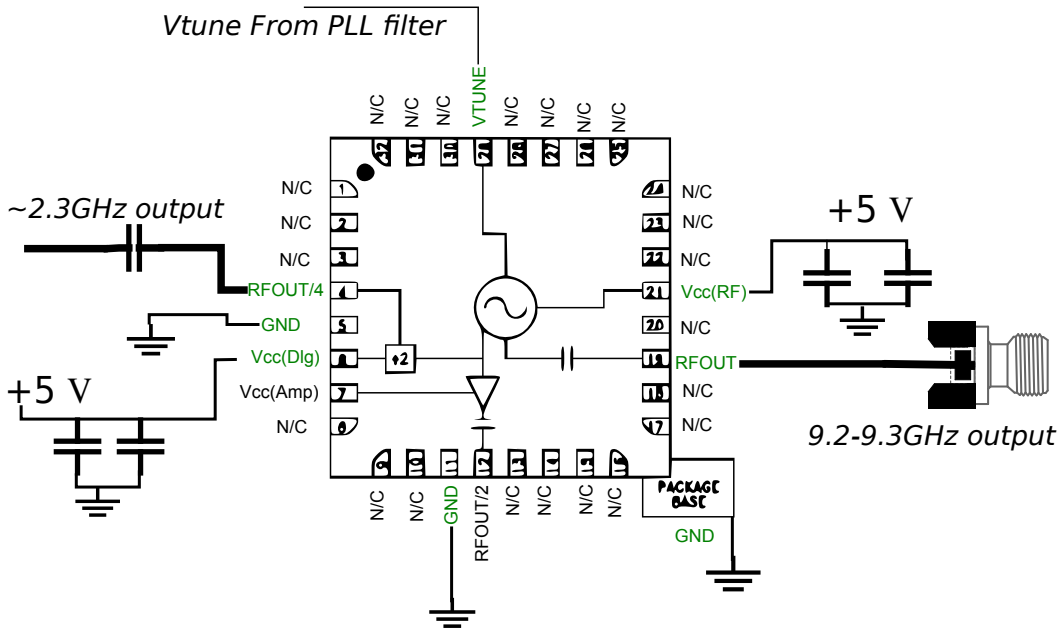


Figure 7.2: *Connections of the VCO (HMC510)*

a reference signal for the phase detector. Leaving the power supply pins for the divide-by-2 amplifier unconnected (pin 7 Figure 7.2) simply deactivated this output. The divide-by-4 output provides -8 to -4 dBm output which is enough for the phase detector that was used. It needs a tuning voltage of up to +15 V (pin 29) and it was therefore necessary to use an OP-AMP to increase the tuning voltage level from the phase detector.

A picture of the PCB layout for the VCO is seen in Figure 7.19. As can be noted, there is pad underneath that serves as a ground connection as well as a thermal conductor. The 5 V $\cdot$ 315 mA = 1.575 W power is a lot for a small 25 mm^2 package and the thermal conduction ability is therefore very important. The pad was connected to the ground plane with several vias. All the outputs are matched to 50 Ω, and the lines connected to them were therefore also designed for 50 Ω.

7.1.2 DDS

Max ref freq.	1 GHz
Power dissipation (PFD, mixer and CP off)	Max 2 W
PN 1 kHz (at 103 MHz)	-147 dBc/Hz
PN 10 kHz –	-150 dBc/Hz
PN 100 kHz –	-152 dBc/Hz
PN 1 kHz (at 403 MHz)	-133 dBc/Hz
PN 10 kHz –	-137 dBc/Hz
PN 100 kHz –	-140 dBc/Hz

Table 7.1: *Properties of DDS*

The DDS comes in a 100 pin TQFP_EP package (See Figure 7.3). This is one of the reasons for that a proper PCB was needed. Most of the pins are actually power supply and grounding. The digital and analogue parts of the chip are separated so different grounds and supplies can be connected. In this design, a common ground was used, but the power supplies were separated so that bypassing could be done for each of them separately.

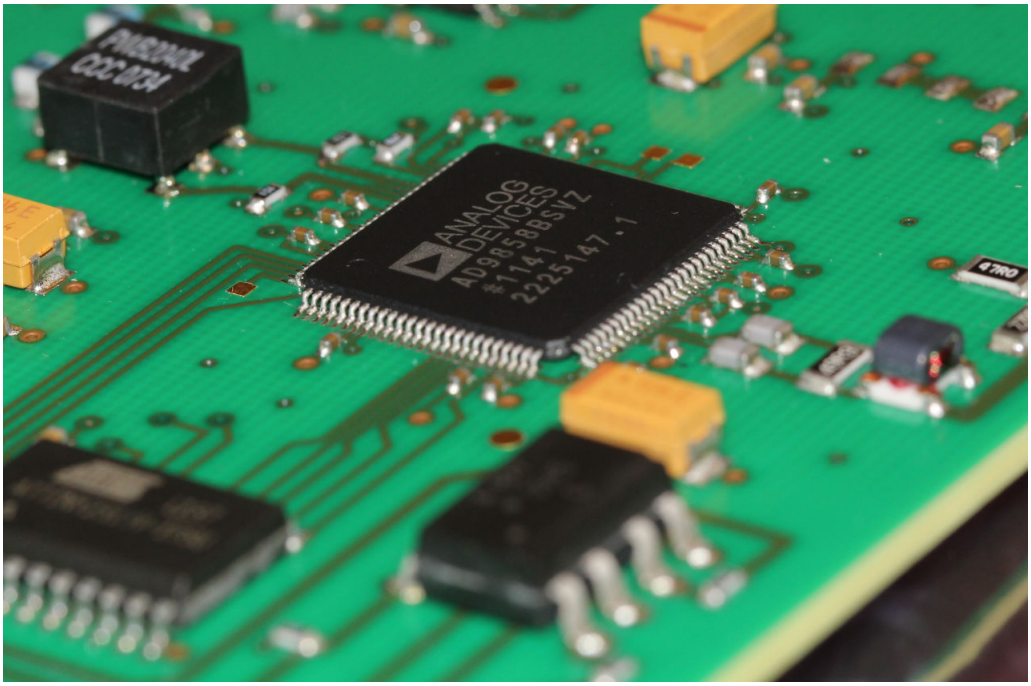
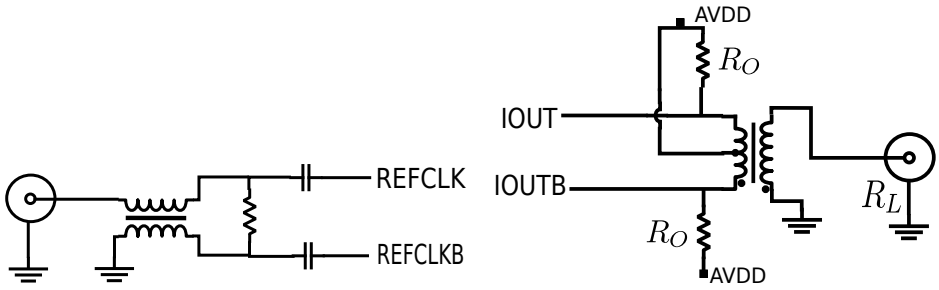


Figure 7.3: *Picture of the DDS*

The steering can be performed either with a parallel interface or a serial interface. This is selected by setting the pin *SPSELECT* either high or low. The parallel interface uses 8 data pins and 5 address pins together with some handshaking pins. This requires a big microcontroller or an FPGA. However, the serial interface can be used with only 3 wires (Chip select included). It permits using a much smaller microcontroller. The maximum speed that can be used to control is given by *SYNCLK* which is the reference oscillator divided by 8. It is obvious that the parallel interface allows higher speed controlling than the serial interface since whole bytes can be clocked at the rate of the SYNCLK instead of just single bits. When the serial interface is used, the register address must be sent separately before the data instead of parallely with its own pins. A last complication is that the serial interface only allows changing whole 32-bit words at a time, while the parallel interface can change the value of 8-bit parts of these words at a time. However, since the parallel interface needs so many more control wires, it was decided to use the serial interface to accommodate the use of a small MCU.

There is also an internal phase detector and mixer in the AD9858. However, even though a phase detector was needed in the design, this one was not used since its specification was not good enough. This caused even more pins of the DDS to remain unconnected.



(a) Reference input (b) DDS output

Figure 7.4: *Two baluns*

In order to connect the reference signal to the DDS, an unbalanced to balanced converter (Balun) and impedance transformer was needed since the input is balanced and most reference oscillators expect to get a 50 Ω load. The same was the case at the output which is also balanced and a single-ended filter was to be used. (See Figure 7.4).

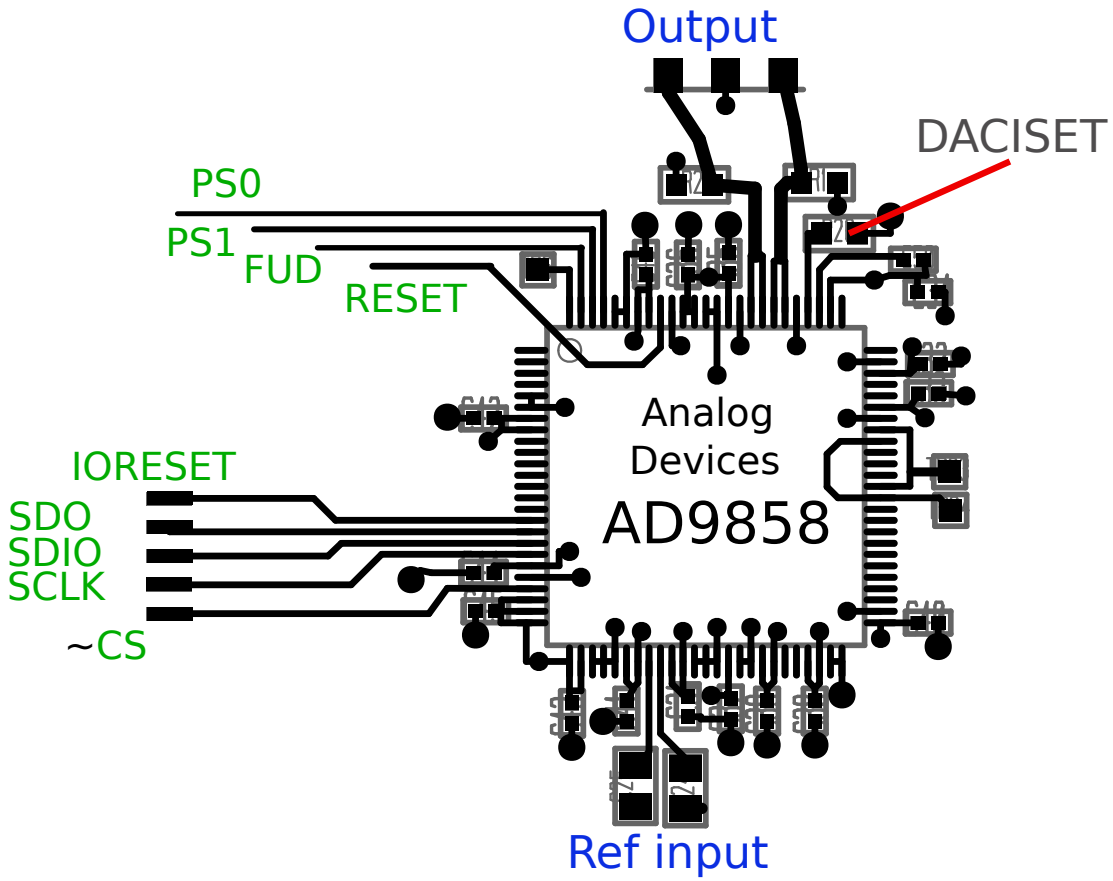


Figure 7.5: *How the DDS was connected.*

The output level is given as a current which can be set by putting different resistors between the pin *DACISET* and ground. The full-scale current is proportional to the

resistor value as [29]:

$$R_{SET} = 39.19/I_{\text{MAX}} \tag{7.1}$$

A resistor of 1.96 kΩ was used. 1.96 $k\Omega$ yields $I_{MAX} = 20$ mA as the maximum current. The DDS is capable of 40 mA output at maximum, but because of reduced noise and spuriosity at lower currents, 20 mA was chosen. [29] states that the best *spurious-free dynamic range* (SFDR) is given at this value.

In order to match to 50 Ω, the resistors R_O and the winding ratio N of the transformer at the output had to be scaled accordingly. N defines the output power like this [30]:

$$P_L = \frac{R_L}{2}\left(\frac{I_{\text{MAX}}}{4N}\right)^2 \tag{7.2}$$

where R_L is the desired output impedance, in this case 50 Ω. For $N = 1/2$, this is:

$$P_L = \frac{50}{2}\left(\frac{20\text{ mA}}{4 \cdot 1/2}\right)^2 = 2.5\text{ mW} \approx +4\text{ dBm} \tag{7.3}$$

It seems like the output power from the DDS can be made infinitely high by choosing N very small, but in addition to having a maximal current output, it also has a maximum peak voltage of 0.5 V. The reference input of the phase detector accepts inputs of -5 to +5 dBm, so more output power than +4 dBm is not needed. To match, the resistors R_O were given values according to [30]:

$$R_O = \frac{R_L}{2N^2} = \frac{50}{2 \cdot (1/2)^2} = 100\ \Omega \tag{7.4}$$

They were connected from $IOUT$ and $IOUTB$ to AVDD as indicated in the datasheet [29].

To avoid mistakes and to get inspiration, the official evaluation board [31] was studied intensively. Among other things, the same input and output transformers were used. The layout of the power plane (Figure 7.16(c)) was also heavily inspired by the design.

The registers in the DDS had to be set to the desired values (see Appendix E.3). First of all, the 2 GHz divider had to be disabled by setting bit 6 in the control register 0x00. For fixed frequency output the frequency is set with the register FTW and given by:

$$f_{OUT} = \frac{(FTW \times SYSCLK)}{2^{32}} \tag{7.5}$$

where $SYSCLK$ in our case is 1 GHz. The value of FTW can thus be calculated like this:

$$FTW = \frac{f_{OUT} \times 2^{32}}{SYSCLK} \tag{7.6}$$

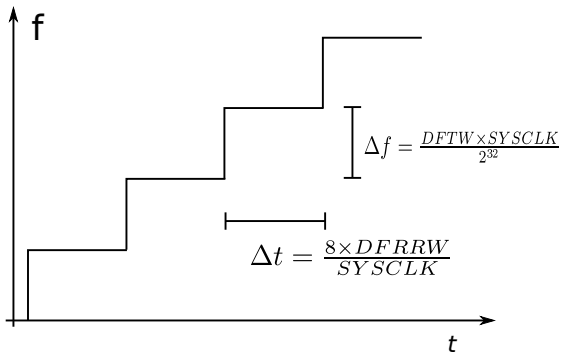


Figure 7.6: *How the registers in the DDS controls the automatic sweep*

When using the automatic sweep function, the frequency steps are set by

$$\Delta f = \frac{DFTW \times SYSCLK}{2^{32}} \tag{7.7}$$

and the time between every step by

$$\Delta t = \frac{8 \times DFRRW}{SYSCLK} \tag{7.8}$$

Consider also figure 7.6 for the registers used when sweeping. In order to start the sweep, bit 15 of the control register is set. See Appendix A for details on how the AD9858 was controlled.

Many power supply pins are present on the AD9858. All of them were individually bypassed with capacitors directly to the ground plane. Figure 7.5 shows them together with all the output and inputs used. The digital connections are green while the RF connections are blue.

7.1.3 Low-pass filter

The phase detector frequency was 121 MHz, and because of the Nyquist images generated (Appendix D), a low-pass filter was needed at the DDS output. The cutoff frequency was set to about 200 MHz. This would then also allow the testing of other phase detector frequencies up to about 200 MHz. A 5-pole Chebyshev 0.1 dB ripple filter referred to 50 Ω was constructed as figure Figure 7.7 presents. This was found using an Internet low-pass filter calculator [32]. It would also be possible to use prototype tables with standard values and adapt to the correct cutoff frequency and circuit impedance.

The frequency response of the filter is plotted in figure 7.8. To make the filter, chip inductors and capacitors were used.

7.1.4 Microcontroller

The microcontroller AVR ATtiny2313 from Atmel was chosen because of its small size and easy programming. It was programmed using an Atmel AVR ISP mkII

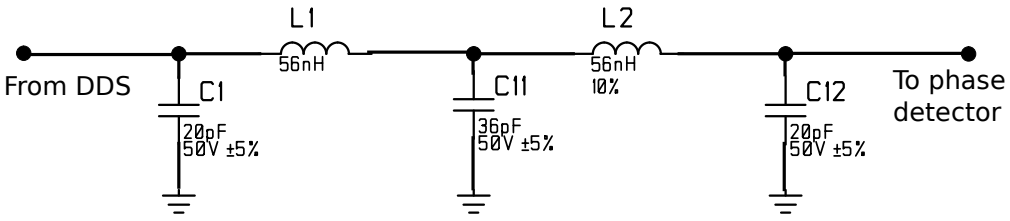


Figure 7.7: *Filter to remove image products from the DDS output*

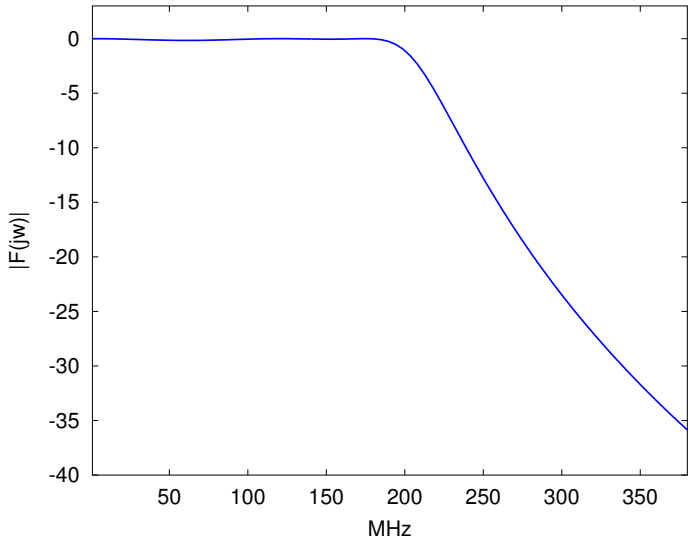


Figure 7.8: *Response of the low-pass filter*

Programmer connected to the computer with USB. The controller code was written in C and compiled using *gcc-avr* and *avr-libc* (Refer to Appendix A for the source code). It was transferred to the circuit using *avrdude*. All of the three are available as open source software in the Ubuntu package repository. The connections that were needed to program the MCU are presented in Appendix B.

The MCU has a built-in clock oscillator at 8 MHz which is divided by 8 inside the chip by default so that the effective clock rate is 1 MHz. It was decided to use serial programming of the DDS and hence not so many pins of the MCU were used.

A file *ad9858control.c* was developed (see Appendix A.1) with many functions for the serial communication It was desired to let the synthesizer be controlled with a

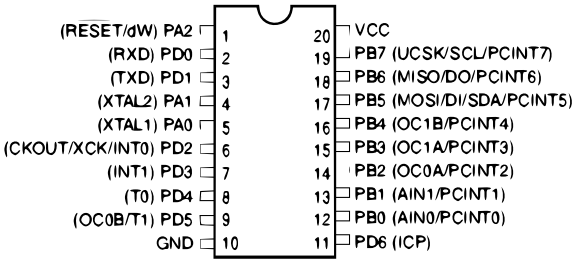


Figure 7.9: *Pinout of AVR ATtiny2313 [33]*

computer. Since the computer can cause much noise to be transferred to the board, it was decided to use an optocoupler for the communication. By using an optocoupler, galvanic isolation can be achieved and the noise from a connected computer can be eliminated to some degree.

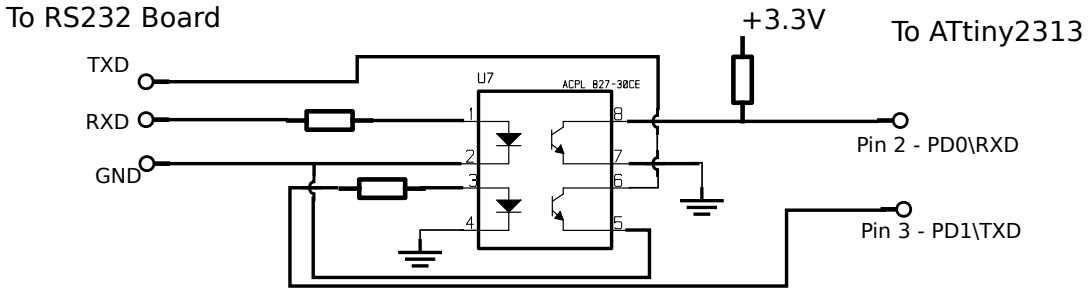


Figure 7.10: *How the galvanic isolation was achieved with an optocoupler*

7.1.5 Phase detector

Max ref freq.	1.3 GHz
PN 10 kHz square wave clock 100MHz	-153 dBc/Hz
PN 10 kHz sine wave clock 100MHz	-148 dBc/Hz
Total current	310 mA

Table 7.2: *Properties of phase detector [34]*

The phase detector from Hittite, HMC698, was chosen because of its high performance. It is a PFD without a charge pump. The output is however a differential voltage. There is a divider inside and the division factor can be set with external jumpers from 12 to 159.

The phase noise floor of the phase detector is given both for sinusoidal reference input and square wave reference input. The square wave is definitely better. However, in our case, the input will be sinusoidal, and hence the phase noise floor is at -148 dBc/Hz at most offset frequencies. This can be converted to the frequency of interest [35] by using the formula:

$$\mathcal{L}_{\text{from\_PLL}} = \mathcal{L}_{\text{PLL\_at\_100MHz}} + 10\log\left(\frac{f}{100\text{ MHz}}\right) \tag{7.9}$$

To choose the divisor, there are 8 pins that must be set either high or low. In order to allow experimentation, a place was made for jumpers on the board. In addition, a LED was connected to the lock-detect output of the phase detector to indicate when the PLL is in lock. This can be very useful.

7.1.6 Loop filter

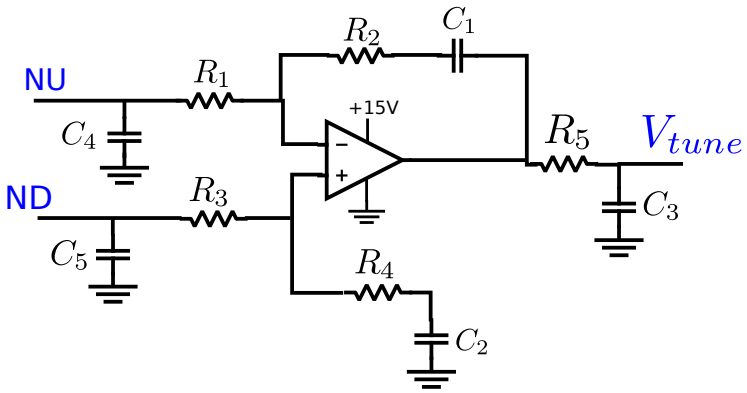


Figure 7.11: *Schematic of the loop-filter*

The filter used was a differential integrator. To the left in Figure 7.11 are capacitors for jump suppression (C_4 and C_5) [23]. To the right is a filter with a pole far outside the bandwidth of the loop. If these are ignored for now, the output voltage of the op-amp is given by.

$$V_{\text{out}} = A_v(V^+ - V^-) \tag{7.10}$$

where V^+ is:

$$V^+ = V_{\text{ND}}\left(\frac{R_4 + \frac{1}{sC_2}}{R_4 + \frac{1}{sC_2} + R_3}\right) \tag{7.11}$$

and:

$$V^- = V_{\text{NU}} \left(\frac{R_2 + \frac{1}{sC_1}}{R_2 + \frac{1}{sC_1} + R_1} \right) + V_{out} \left(\frac{R_1}{R_2 + \frac{1}{sC_1} + R_1} \right) \tag{7.12}$$

If $A_v \gg 1$, the output of the op-amp can be found to be:

$$V_{\text{out}} = V_{\text{ND}} \left(\frac{R_4 + \frac{1}{sC_2}}{R_1} \right) \left(\frac{R_1 + R_2 + \frac{1}{sC_1}}{R_3 + R_4 + \frac{1}{sC_2}} \right) - V_{\text{NU}} \left(\frac{R_2 + \frac{1}{sC_1}}{R_1} \right) \tag{7.13}$$

In the case when $R_1 = R_3$, $R_2 = R_4$ and $C_1 = C_2$, the equation can be written:

$$\begin{aligned} V_{\text{out}} &= V_{\text{ND}} \left(\frac{R_4 + \frac{1}{sC_2}}{R_1} \right) - V_{\text{NU}} \left(\frac{R_2 + \frac{1}{sC_1}}{R_1} \right) \\ V_{\text{out}} &= (V_{\text{ND}} - V_{\text{NU}}) \left(\frac{R_2 + \frac{1}{sC_1}}{R_1} \right) \end{aligned} \tag{7.14}$$

and the transfer function from the differential input to the output of the op-amp is given by:

$$F(s) = \frac{V_{\text{out}}}{V_{\text{ND}} - V_{\text{NU}}} = \frac{R_2 + \frac{1}{sC_1}}{R_1} = \frac{sR_2C_1 + 1}{sR_1C_1} = \frac{R_2}{R_1} + \frac{1}{sR_1C_1} \tag{7.15}$$

Thus, in the case when $R_1 = R_3$, $R_2 = R_4$ and $C_1 = C_2$, the transfer function is the same as for a normal integrator (figure 4.4(a) page 39). From here on, the resistors R_2 and R_3 are referred to as R_2 and assumed to be of equal value. The same is the case with C_1 and C_2, which are referred to as C_1. If the values are not equal, it has been shown [36] that the output will not be constant for zero differential input voltage, which it should be for an integrator.

Op-amp

Whatever op-amp could be chosen. Because the loop filter also will contribute noise that adds to the output phase noise, it was decided to use a low-noise op-amp. The ADA4898 from Analog Devices is such an op-amp which claims to have ultralow noise. It is specified with 0.9 nV/$\sqrt{\text{Hz}}$ as voltage noise, and 2.4 pA/$\sqrt{\text{Hz}}$ as current noise.

Noise contribution from the loop filter

Noise from the loop filter is found like this [37]:

$$\overline{v^2}_{n,R_1} = 2 \cdot 4kTR_1 \left(\frac{R_2}{R_1 + R_2} \right)^2 \tag{7.16}$$

$$\overline{v^2}_{n,R_2} = 2 \cdot 4kTR_2 \left(\frac{R_1}{R_1 + R_2} \right)^2 \tag{7.17}$$

$$\overline{v^2}_{n,\text{opamp}} = \overline{v^2}_{n,\text{opamp-voltage}} + 2 \cdot \overline{i^2}_{n,\text{opamp-current}} \left(\frac{R_1 \cdot R_2}{R_1 + R_2} \right)^2 \tag{7.18}$$

$$\overline{v}_{n,\text{input}} = \sqrt{\overline{v^2}_{n,R_1} + \overline{v^2}_{n,R_2} + \overline{v^2}_{n,\text{opamp}}} \tag{7.19}$$

In the case with $R_1 = 200\ \Omega$ and $R_2 = 15\ \Omega$, the noise is given as about 1.12 nV. 64 % of this appears from the noise of the op-amp, and the remaining 36 % appears from the resistors.

This noise is referred to the input of the loop filter. It is necessary to multiply it with the filter response:

$$V_{n,\text{output}}(j\omega) = V_{n,\text{input}}(j\omega)|F(j\omega)| \tag{7.20}$$

7.1.7 Frequency reference

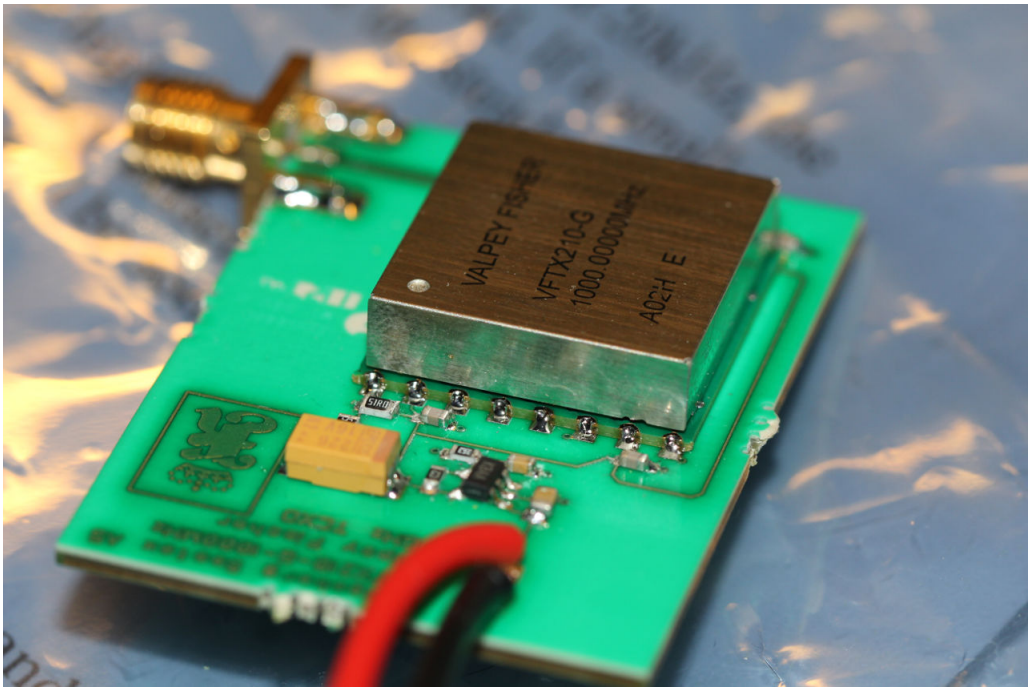


Figure 7.12: *Picture of the 1 GHz reference oscillator*

Valpey Fisher's VFTX210 was chosen as the frequency reference. It is rather big and enclosed in a metal box (see Figure 7.12). A 50 Ω output provides a 1 GHz

signal at a level of about +8 dBm. The DDS accepts an input of -20 to +5 dBm, so a simple T attenuator made of 3 resistors was placed between them. The PCB was accomodated with a place for the frequency reference, but because of its high cost, it was later decided to not mount it on the board, but provide the signal via a coaxial cable from a board where it was already mounted (Figure 7.12).

7.1.8 Voltage regulator

A voltage regulator LT1963AEST-3.3 was used to provide 3.3 V to the DDS, the TCXO and the microcontroller.

7.2 Schematic

The schematic of the entire synthesizer can be seen in Figure 7.13. It was designed with CADSTAR 12.1.

7.3 PCB

Because of the high operating frequency and the surface-mounted components, a proper PCB was needed. The design of the PCB was an important part of the construction. There are different materials used for PCBs. The choice was mainly between the Rogers laminates (RO4003 and RO4350) which are often used at higher frequencies and the much cheaper FR-4 alternative which is used for most other types of circuits. Since this was a design of a 9.2 GHz frequency synthesizer, it would probably sound smart to use a Rogers laminate. However, since a dedicated VCO chip was used, the 9.2 GHz part of the circuit could be confined to a small area of the circuit board. All the critical parts were inside the VCO and only a short line from the VCO to the output at 9.2 GHz was needed. The divide-by-4 output of the VCO was used to connect the VCO to the phase detector and this line carries a signal at 2.3 GHz. The FR-4 laminate was chosen because it is cheaper and still suitable.

Since the AD9858 has so many pins and many of them must be connected to the power supply, it is very useful, and pretty much compulsary, to design a multi-layer PCB. A 4-layer board was chosen. The top layer was used for the main components and connections. Layer 2 was used as a ground plane, layer 3 as a power plane. Different voltages were needed for the different parts of the board so by having a proper layer for the power, the voltages could neatly be distributed. The bottom layer (layer 4) was also used as a ground plane. At places where other signals had to cross each other, they were simply routed via either layer 3 or the bottom layer at places where the ground plane and the power plane were removed.

The entire vertical structure of the PCB is shown below.

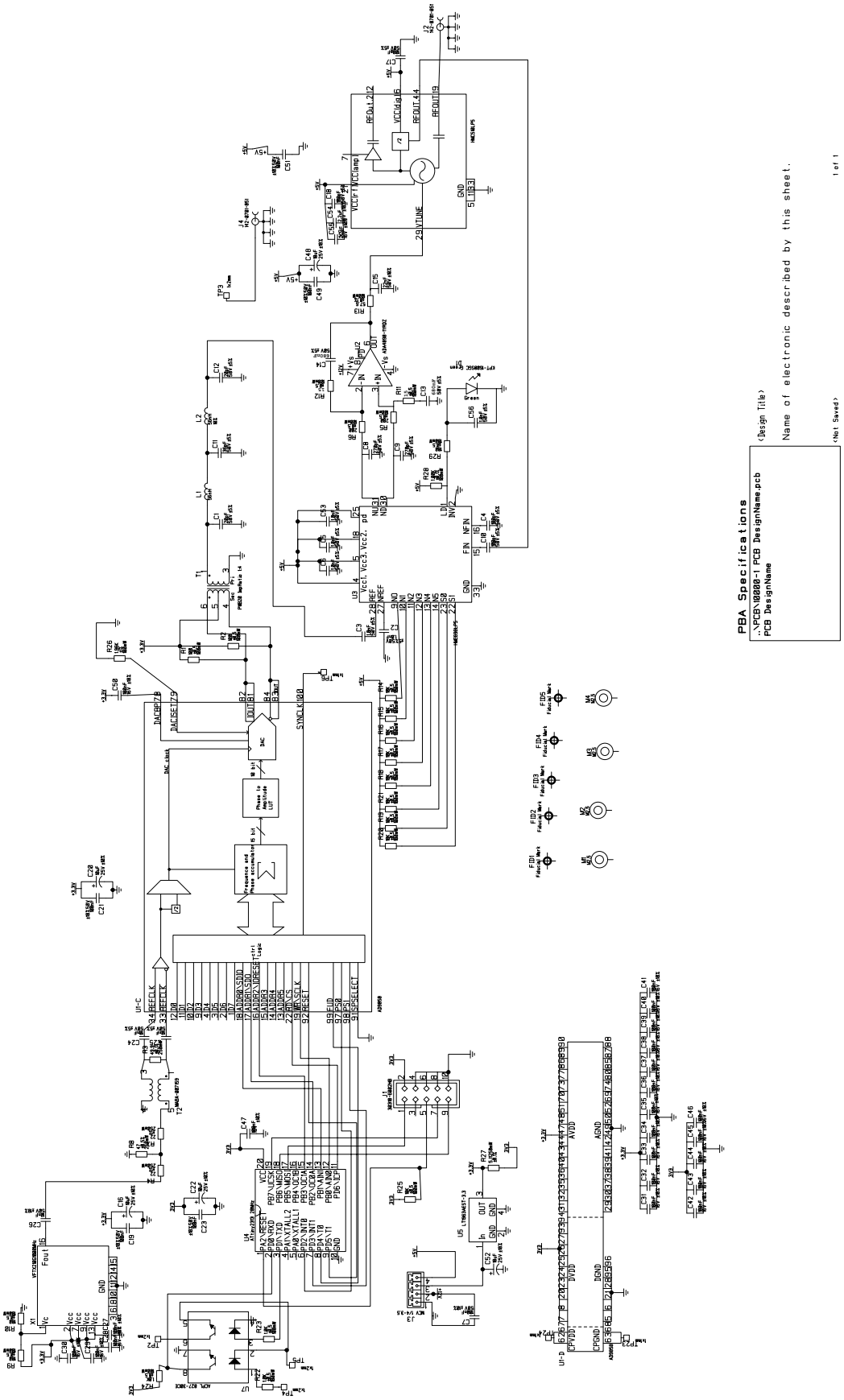


Figure 7.13: *The complete schematic exported from Cadstar*

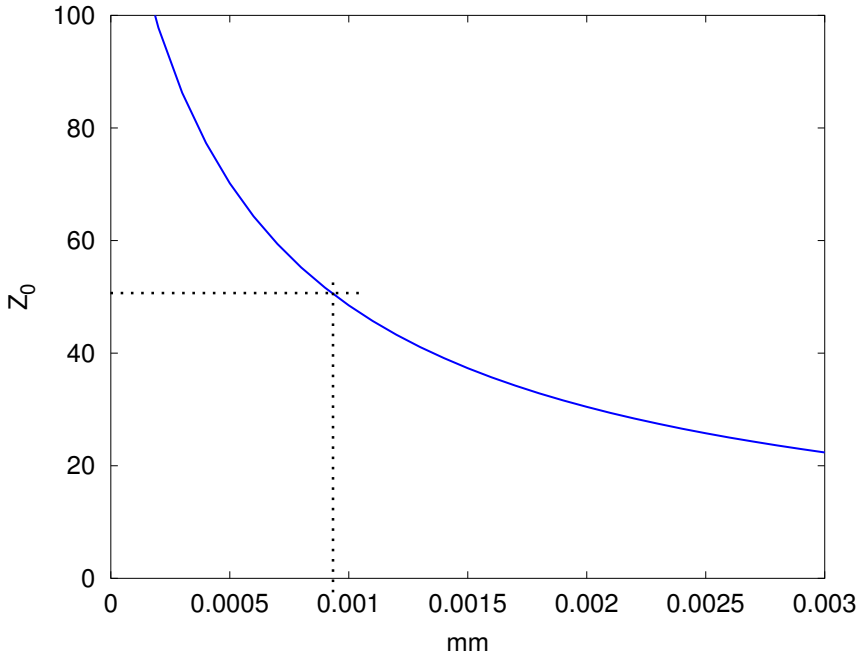


Figure 7.14: *Characteristic impedance for $\varepsilon_r = 4.2$ and different widths*

```
      Blind   Plated
      Pri-L2  Pri-Sec
                                  No Silkscreen
    ~~~~~~~~~~~~~~~~~~~~~  20um  Solder Mask
Pri ======|=======|======  35um  Copper Foil: Resistivity=1.75
    XXXXX | XXXXX | XXXXX 508um  FR4: Er=4.50
L2  ======|====== | =====  35um  Copper Foil: Resistivity=1.75, Embeds DOWNWARDS
    XXXXXXXXXXXX | XXXXX 483um  FR4 Prepreg: Er=4.20
L3  ============ | =====  35um  Copper Foil: Resistivity=1.75, Embeds UPWARDS
    XXXXXXXXXXXX | XXXXX 508um  FR4: Er=4.50
Sec ============|======  35um  Copper Foil: Resistivity=1.75
    ~~~~~~~~~~~~~~~~~~~~~  20um  Solder Mask
                                  No Silkscreen
```

As can be seen, two different types of vias were used. The permittivity of the FR4 is 4.5 - a number that is needed when calculating line widths.

The layout was also designed with CADSTAR 12.1. For the high frequency lines - 4 lines in total, the line widths must correspond to the correct line impedance. All of them were referred to 50Ω and the engineering formula (7.22) was therefore used to find their widths. A plot of the formula is shown in Figure 7.14 and it can be observed that a narrower line causes a higher impedance. For this 4-layer board, the h in the formula was the distance between the top layer and layer 2, thus $508um$.

$$\varepsilon_{\text{eff}} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2\sqrt{1 + (10h/W)}} \tag{7.21}$$

$$Z_c = \frac{120\pi}{\sqrt{\varepsilon_{\text{eff}}}\left(\frac{W}{h} + 1.393 + 0.667\ln\left(\frac{W}{h} + 1.444\right)\right)} \tag{7.22}$$

The width was found to be about 0.95 mm. An impedance calculator found on the

Internet [38] was used to adjust this for the exact frequencies employed. This gave 0.97 mm for the 121 MHz part, and 0.95 mm for the 9.2 GHz part.

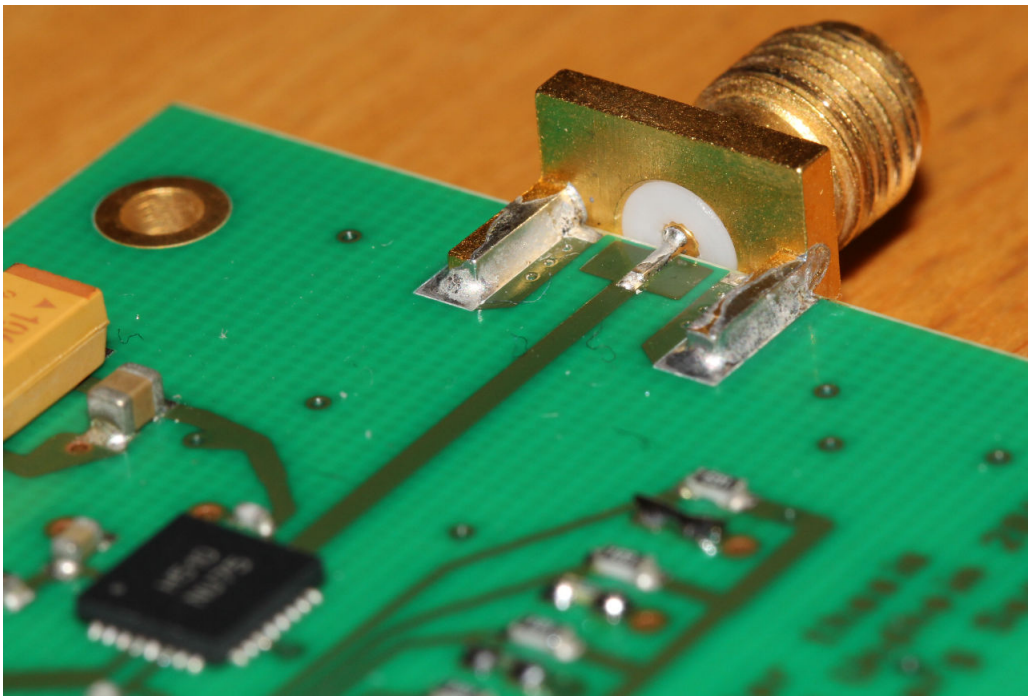


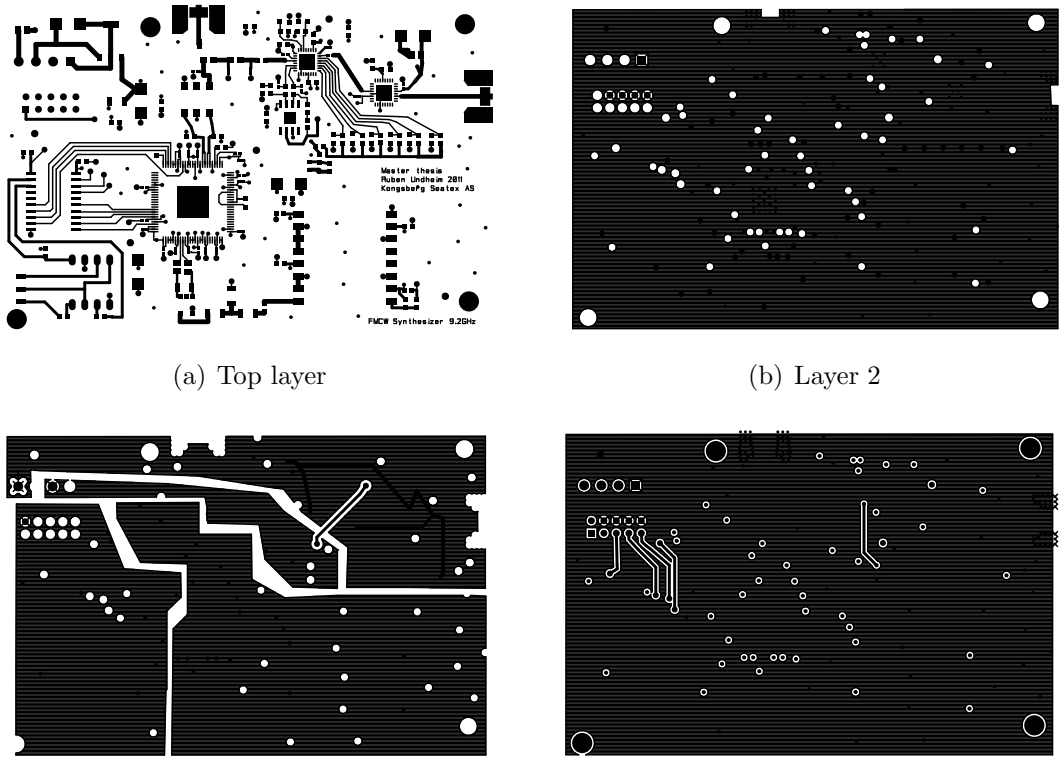
Figure 7.15: *Picture of the VCO and the output SMA connector*

The SMA connectors were of the edge-mounted type (see Figure 7.15). They were designed for a board thickness of 1.57 mm. This was a deciding factor for that the total board thickness was set to this. However, the connectors are also designed for being attached to a microstrip line designed for this board thickness, but in this design the microstrip lines are dimensioned according to the spacing between the top layer and layer 2 (0.508 mm). A trick was done. Just at the point where the centre conductor of the SMA connector touches the board, the microstrip was made wider (consider Figure 7.15). In addition, the copper on layer 2 and layer 3 was removed exactly underneath the wide part of the microstrip. By doing this, the connector was supposed to be matched to the board. At frequencies as high as 9.2 GHz, it is expected that by doing this wrongly, there will rapidly be a big loss at the connection.

7.3.1 Finished design

The PCB layout was finished and fiducial marks were placed on the board (5 in total even though only 3 are stricly needed). Then the design files were exported from Cadstar. The design files consist mostly of Gerber files, but two NC drill files for the different via holes and one readme file were also exported. The readme file contains important data about how the board was desired to be, including PCB stack-up[1], board dimensions and via sizes.

[1]The stack-up contains the type, thickness and spacing of the different layers.



(a) Top layer

(b) Layer 2

(c) Layer 3

(d) Bottom layer

Figure 7.16: *Showing the 4 different layers of the FR-4 PCB*

The following is a complete list of the files sent to the manufacturer. They were added to a zip-archive and sent.

```
PCB Readme.txt
PCB Manuf Output.rep
PCB Master Drawing.pdf
PCB Assembly Drawing.pdf
PCB Solder Paste Drawing.pdf
PCB Primary Silk.gbr
PCB Mechanical Drawing.gbr
PCB Primary Solder Resist.gbr
PCB Secondary Solder Resist.gbr
PCB Primary.gbr
PCB Layer-2.gbr
PCB Layer-3.gbr
PCB Secondary.gbr
PCB Drill Drawing Primary to Layer-2.gbr
PCB Drill Drawing Through Hole.gbr
PCB NC Drill Plated Primary to Layer-2.nc
PCB NC Drill Plated Through Hole.nc
```

As can be seen, Cadstar also exported some *.pdf files which contain the design and these are useful in order to ease the inspection. They are stricly not needed by the manufacturer. The file *.rep is a log-file from the export process of Cadstar and is included for the convenience. Gerber files can be opened with the open source program *gerbv* which is found in the Ubuntu package repository. Figure 7.16 and 7.17 were exported from *gerbv* and present the entire PCB design.

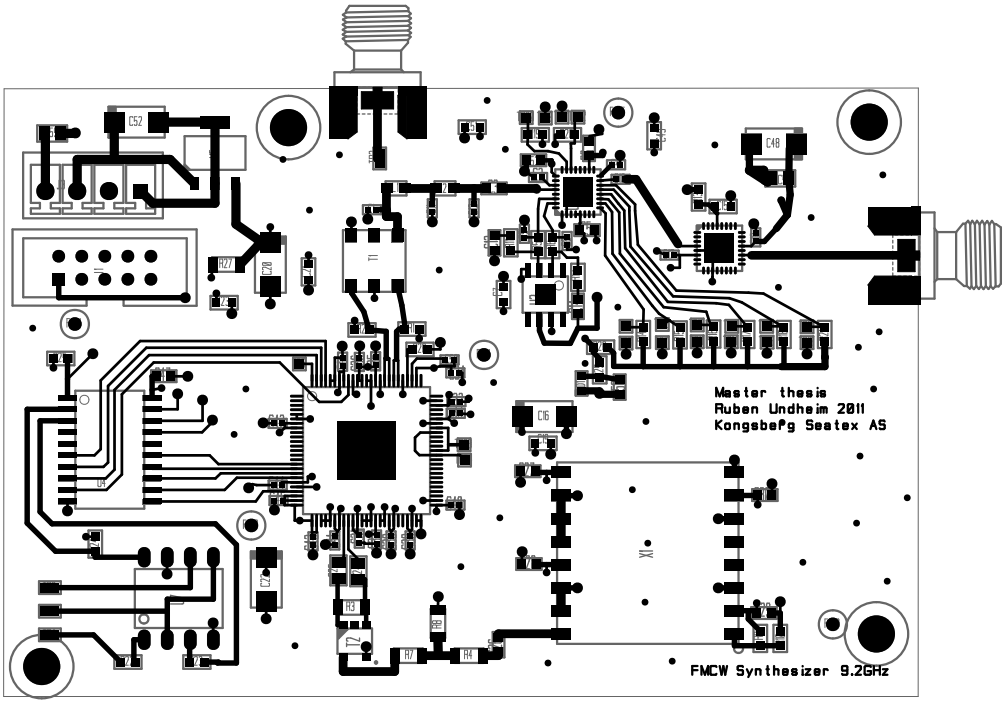


Figure 7.17: *Top layer of the PCB and component placement*

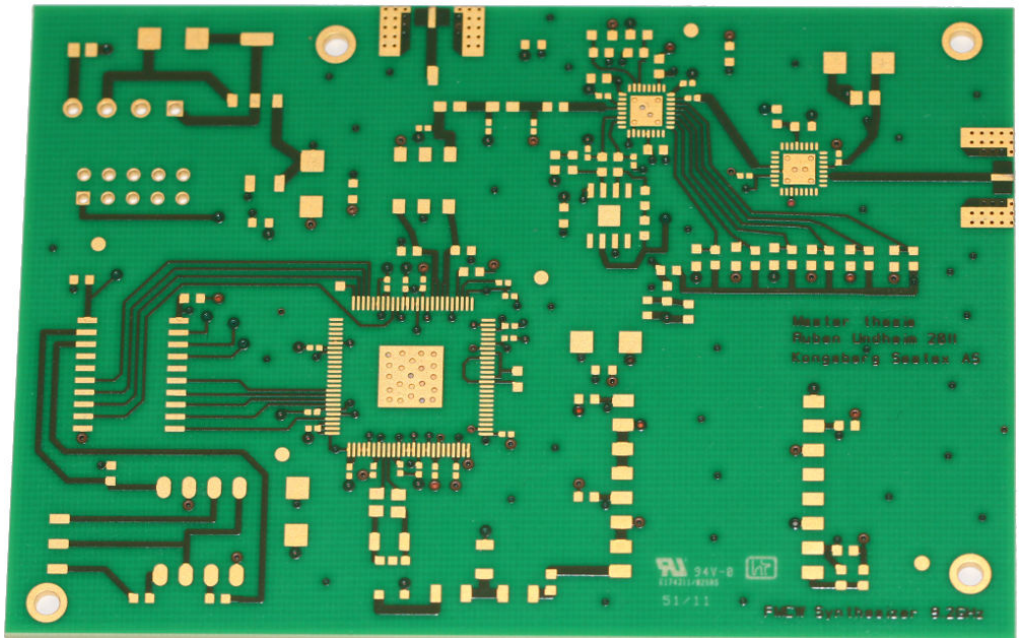


Figure 7.18: *Picture of the PCB without components*

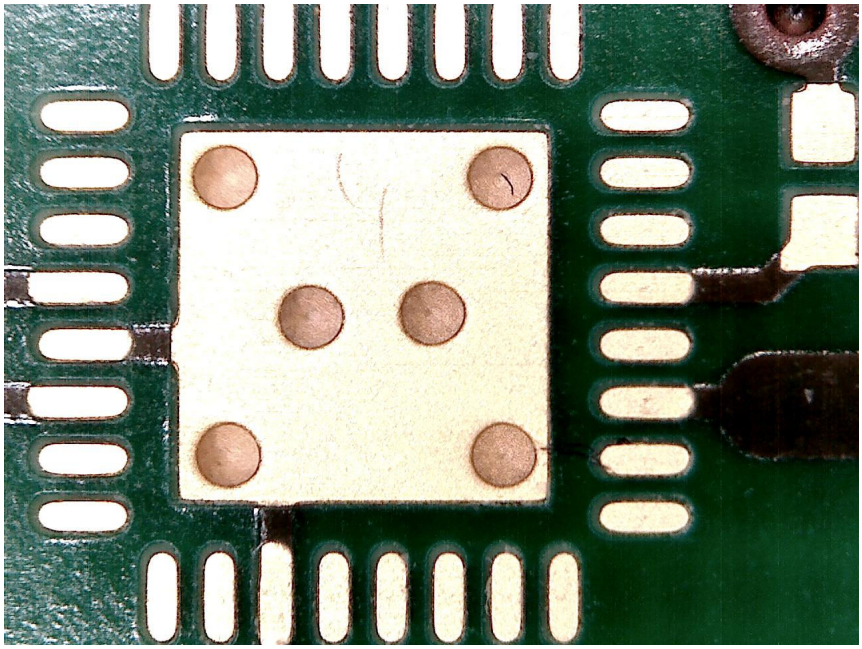


Figure 7.19: *Close-up picture of the pads for the VCO*

7.4 Verification of Board

When the PCB arrived from the manufacturer (figure 7.18), the excitement was great. All the traces were checked in order to see if they were correctly connected inside the board. Luckily, no errors were found. Because of the components from Hittite which have soldering pads underneath, it would not be easy to solder the board by hand. Thanks to Kongsberg Seatex, the board was sent to Simpro, Løkken, in order to be mounted. The files needed for the mounting were:

```
PCB Primary Assembly.gbr
PCB Primary Paste.gbr
PCB Primary Pick and Place.txt
PCB Secondary Paste.gbr
PCB BOM.xls
```

After a couple of days, when the board was ready, the excitement was even greater. Was it going to work?

At first, the DDS was to be checked. The board was powered up with one power supply at 5 V. The power to the VCO and phase detector was not connected at this time. No short-circuits were discovered, and the board was drawing the expected current of about 0.45 A. The MCU was still not programmed, so the board was not doing anything reasonable. The AVR ISP mkII programmer was then connected between the computer and the board and the code for a fixed frequency (see Appendix A.3) was used to program the microcontroller. An Agilent infiniium 54832B DSO oscilloscope was connected to the output of the DDS, and after some time fiddling around, the expected result appeared on the oscilloscope screen! The output was a sine wave at 121.7 MHz. The DDS and MCU part of the board was thus verified, and everything worked perfectly.

Now, the next stage was to check the VCO. The output of the board was connected to the R&S FSP spectrum analyzer and the board was powered with one supply for the phase detector and the VCO. What was observed on the spectrum analyzer after carefully adjusting the frequency span and amplitude controls, was a carrier at some frequency near 8 GHz. The output signal was not very stable. Next, the power to the op-amp was connected, 17 V. Then it was observed that the output frequency was changed when the op-amp voltage was adjusted.

The next stage was to be a bit more tricky. During the design, just some random values for the capacitor and the resistor in the loop filter were selected. This was done in order to find out how important the choice is. The board was therefore not necessarily expected to work before these components were replaced. Still, the complete board was powered with two different power supplies. One at 5 V for the DDS, MCU, VCO and phase detector and one at 17 V for the op-amp. The divisor of the phase detector was configured (using jumpers) to be at 19. Together with the division of 4 at the reference output of the VCO, the total division factor was 76. With the DDS reference frequency programmed above, the output was expected to be $121.7 \cdot 10^6 \cdot 76 = 9.25$ GHz. This was not observed. The output was not fixed at a certain frequency. It jumped around over a frequency band corresponding to a GHz or so. It was back to the calculation board. Based on the components that

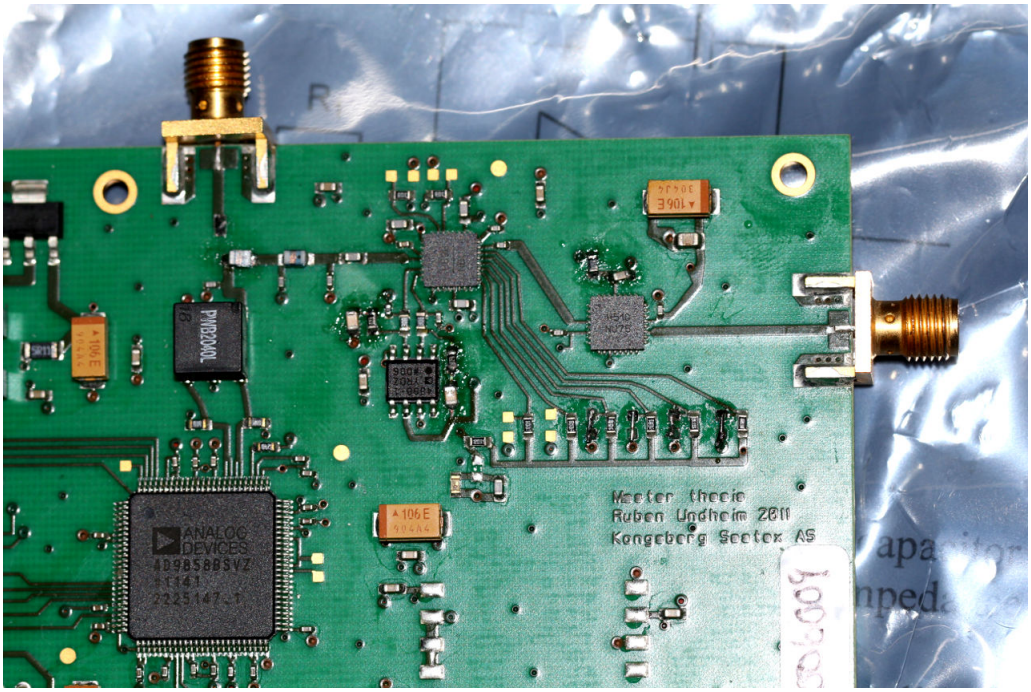


Figure 7.20: *Picture of the PCB showing PLL loop and DDS*

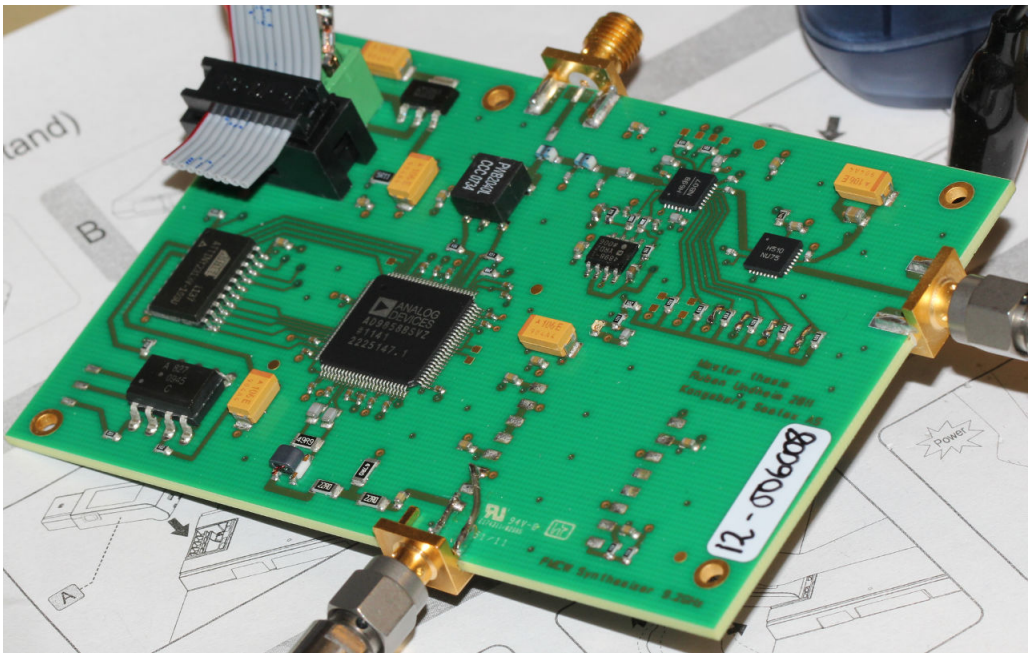


Figure 7.21: *Picture of the complete circuit*

were on the board, the natural frequency of the board was found to be 630 kHz and the damping factor 1.8. These values were consequently not good. According to the simulations performed earlier, the best natural frequency should be about 30 kHz, and the best damping factor about 1.0 (see Figure 6.11 and 6.12 page 63). New components values were calculated to be $C_1 = 470\text{nF}$ and $R_2 = 22\ \Omega$. However, these exact values were not selected because they were not available at the moment. The capacitors were changed to 100 nF and the resistors to 15 Ω. After they were replaced, the board was once again powered. The expected natural frequency was to be 63 kHz, and the expected damping factor, ζ, was to be 1.11. These values were much closer to the calculated ideal values, and the board was this time therefore expected to work.

And it was working! By tuning the spectrum analyzer to 9.25 GHz, it was observed that the output was remarkably stable. The lock had been achieved. It was kind of a surprise. There had not been so much trouble getting it to work than feared, and all the worries could be abandonded.

Chapter 8

Measurements

8.1 Equipment

These are all the instruments that were used for the measurements.

- Tektronix RSA6114A Real-Time Spectrum Analyzer
- Rohde & Schwarz FSP Spectrum Analyzer 9 kHz .. 13.6 Ghz
- Rohde & Schwarz FSQ 40 Signal Analyzer 20 Hz .. 40 GHz
- Anritsu MS2724C Spectrum Analyzer 9 kHz .. 20 GHz
- Agilent infiniium 54832B DSO 1 GHz 1GSa/s Oscilloscope
- Anritsu MG3691B Signal Generator 10GHz
- Topward Dual-Tracking DC Power Supply 6303D
- Rohde & Schwarz NRP-Z21 AVG Power Sensor

8.2 Procedure

At first, the frequency and level of the output signal were measured since all the other measurements depend on them. The level was measured using the NRP-Z21 power sensor. With the loop filter components set to $C_1 = 100$ nF, and $R_2 = 15\ \Omega$, the output spectrum was measured with the R&S FSP spectrum analyzer using a span of 800 kHz and a resolution bandwidth (RBW) of 3 kHz. Then C_1 was changed to 680 nF in order to get nearer the good values calculated in Section 7.4. R_2 was kept at 15 Ω and the output was once again measured.

8.2.1 Phase Noise

Then the phase noise measurements were performed. The spectrum analyzer from Anritsu was used most of the time, but a few tests were also done with the R&S

FSQ40 which has a built-in phase-noise measurement option. By using this option, the other measurements could be verified. The Anritsu spectrum analyzer is portable and does not have a very good phase noise performance itself which may be limiting.

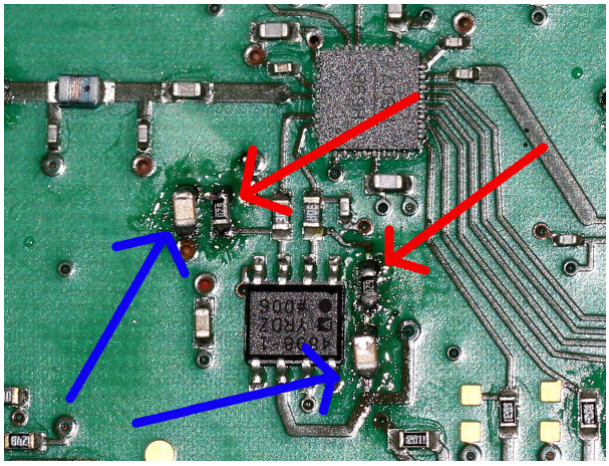


Figure 8.1: *The capacitors and resistors of the loop filter that were changed (How it looked like after about 30 changes). Note that the two capacitors have the same value and are referred to as C_1 while the resistors have the same value and are referred to as R_2*

The span of the spectrum analyzer was set to 200 kHz in order to see the most important part of the phase noise spectrum. Both of the sidebands were observed even though they are expected to be equal to each other (when there is no amplitude noise). Thus, the phase noise was measured from zero to 100 kHz offset. After every measurement, R_2 was changed and a new measurement was performed. C_1 was kept at 680 nF. The measurements were performed with the RBW set to 10 Hz and the VBW set to 3 Hz.

Afterwards, the effect of different capacitor values was tested. C_1 was changed from 680 nF to 470 nF with R_2 equal 22 Ω. Then, R_2 was set to 10 Ω and the same capacitor change was done once again (figure 9.5).

Since it was not certain if the measurements performed were correct because of possibly wrong scaling and related issues, the dedicated phase noise option of the FSQ spectrum analyzer was used to do a control check. The phase noise performance of this instrument is much better and it will therefore not obscure the measurements. It also measures at higher offset frequencies and plots with a logarithmic frequency axis. The loop filter values used in this case were $C_1 = 680$ nF and $R_2 = 33$ Ω. Because of this, it is possible to compare the result directly with the calculated phase noise in Section 6.2. The phase noise measurement option is pretty much automatic, only the carrier frequency and level must be entered before pushing the RUN-button.

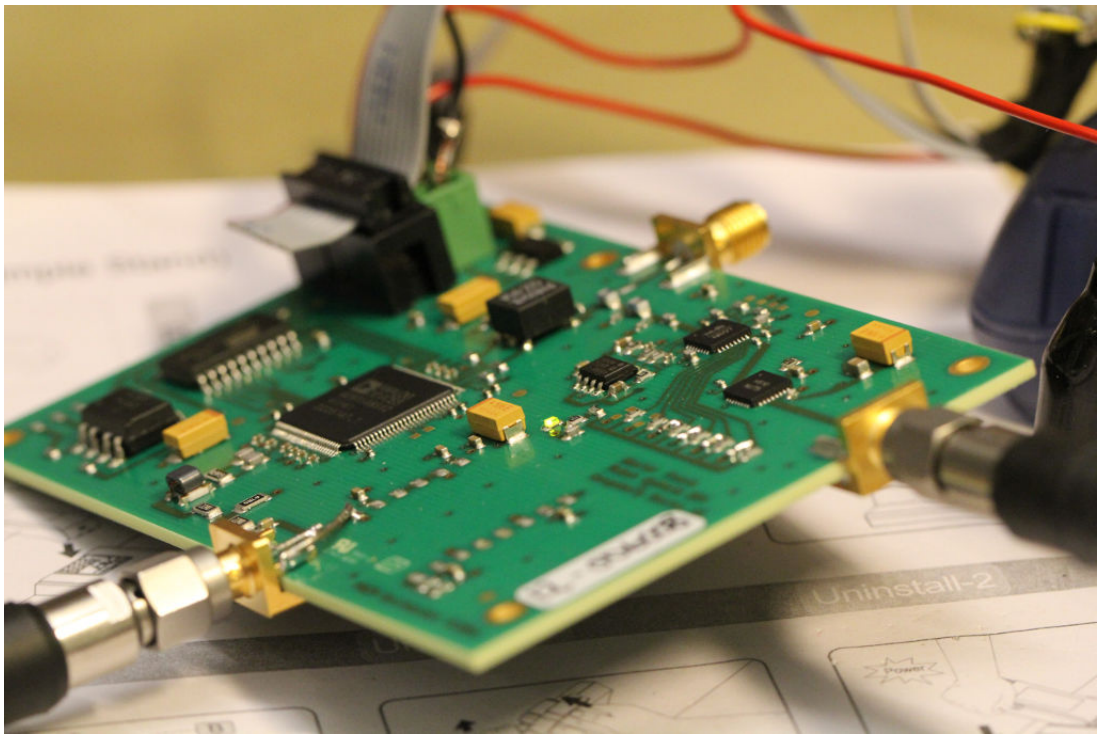


Figure 8.2: *Picture of the board during testing. Notice the green LED in the center which indicates that the PLL is in lock.*

8.2.2 Step Response

Because of the access to a real-time spectrum analyzer, some really interesting measurements could be performed. This type of spectrum analyzer works differently from other spectrum analyzers by first sampling the signal and then calculating the spectrum from the samples. It is therefore capable of showing frequency changes such as a plot of frequency vs time. This can then be used in order to find the bandwidth and the damping factor of the PLL.

The microcontroller was for the step-response measurements programmed with the switch-frequency program (Section A.4). The frequency is switched rapidly between 9.200 GHz and 9.208 GHz. This is accomplished by programming the DDS with two different profiles and then switching between them with the *set_profile()* function. The spectrum analyzer was set to trigger at a frequency of about 9.204 GHz and the frequency span was set to ±10 MHz.

Figure 8.3 is a screenshot from the instrument. Time is on the x-axis, and frequency is on the y-axis. It shows how a typical frequency step looks like.

8.2.3 Sweep

The last thing tested was sweeping. The code for the sweep (Appendix A.5) was programmed into the microcontroller, and the real-time spectrum analyzer was configured for a frequency span of ±20 MHz. Since the spectrum analyzer performs better

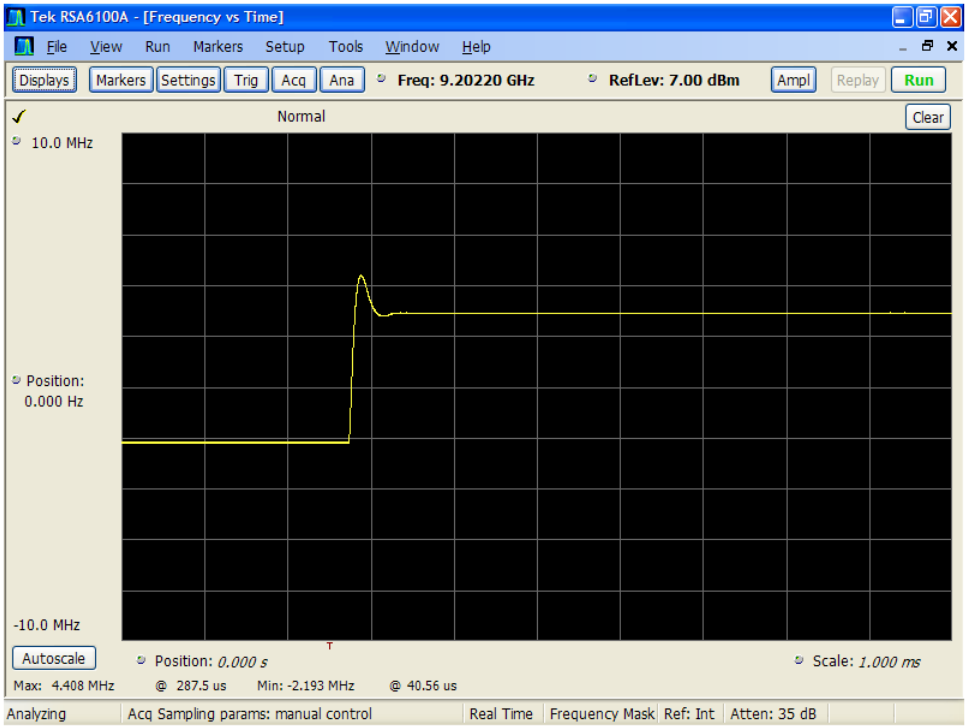


Figure 8.3: *Typical response of a frequency step observed on a real-time spectrum analyzer. Note that time is along the x-axis, and frequency is along the y-axis.*

with narrower spans, it was decided to just let the synthesizer sweep a bandwidth of 25 MHz and not 100 MHz which the operative radar will. It was programmed to sweep from 9.250 to 9.275 GHz. The DFTW register in the DDS was set to 1000 which from equation (7.7) corresponds to a step size of

$$\Delta f = \frac{DFTW \times SYSCLK}{2^{32}} = 233Hz$$

at the reference frequency, or 17.7 kHz at the operating frequency. The time between every step was set to 8 μs by setting DFRRW also to 1000 (7.8). About

$$\frac{25 \text{ MHz}}{17.7 \text{ kHz}} \cdot 8\mu\text{s} = 12 \text{ ms}$$

was then needed to sweep 25 MHz. The sweep rate was 2.21 GHz per second - fairly high.

A symmetrical sweep was also tested. The sweep rate was doubled (DFTW = 2000) to get both the up-sweep and the down-sweep on the same screen using the same scaling as for the asymmetrical sweep. The delay of the microcontroller had to be decreased correspondingly. In order to make a symmetrical sweep, the sweep direction manually had to be changed when the output had reached its maximum frequency. The microcontroller therefore had to time exactly how long the sweep needed. To make the sweep always start from the same frequency for the up-sweep, and another frequency for the down-sweep, the frequency incrementer was reset and the start frequency manually configured after each sweep.

Chapter 9

Results

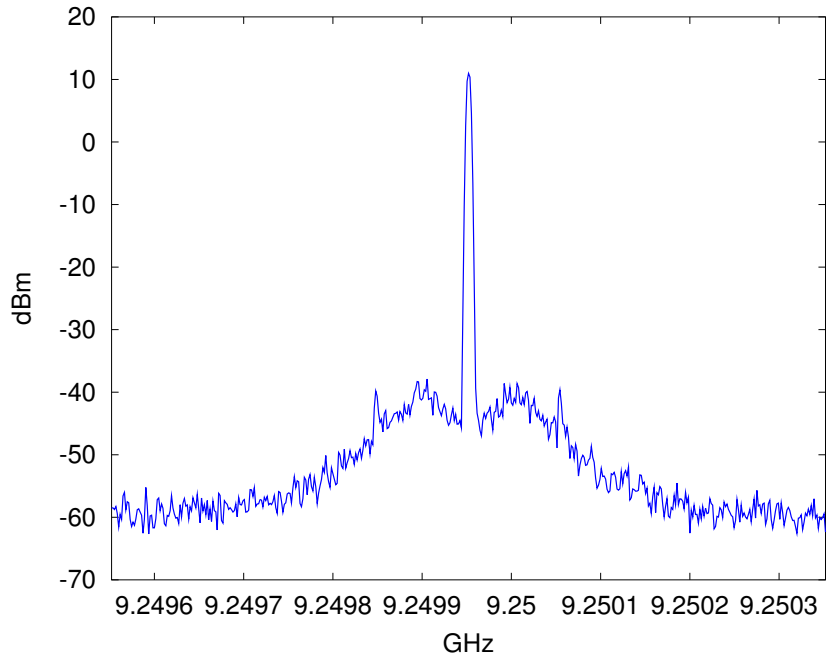


Figure 9.1: *The signal observed scaled in dBm. C_1 = 100 nF, R_2 = 56 Ω, RBW: 3 kHz, VBW: 100 Hz*

The initial results were of the frequency and the level. The frequency was found to be 9.249963 GHz - very close to 9.25 GHz as expected from the DDS settings. Using the power sensor, the level was found to be about +11 dBm. By varying the VCO voltage from 4.5 V to 5.5 V, the output level varied from +9.95 dBm to +11.75 dBm. According to the datasheet (Appendix E.1), it was supposed to be between +10 and +15 dBm so it seemed to be fine. The spectrum of the signal is presented in Figure 9.1 and 9.2 with a span 800 kHz, a resolution bandwidth 3 kHz and a video bandwidth 100 Hz. It should be noted that because of the resolution bandwidth, the noise seen will not correspond to noise power per Hz. If this is desired, it must first be scaled by a factor as explained in Section 3.6. Also, the noise floor is caused

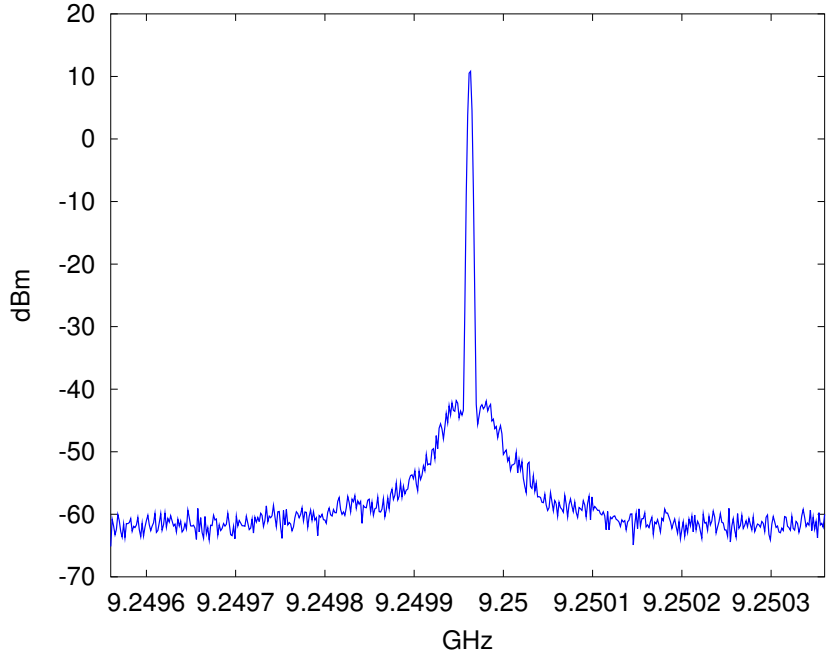


Figure 9.2: *The signal observed scaled in dBm. C_1 = 680 nF, R_2 = 15 Ω, RBW: 3 kHz, VBW: 100 Hz*

by thermal noise in the instrument and is therefore not part of the signal. Figure 9.1 presents the measurement with a bandwidth, f_n, of 63kHz and a damping factor, ζ, of 1.11. f_n is much higher than what was predicted to be the best. The skirt of phase noise is therefore clearly visible and quite bad. It can be noted that the noise power increases somewhat at increasing offsets near the carrier before decreasing again. Some peaks are visible at about 100 kHz offset, and obviously the main peak (the carrier) is visible in the center. The level of the carrier can be observed at about +11 dBm which is the same as what was measured with the power meter. Figure 9.2 presents the same output when the loop filter components are changed. It should correspond to a bandwidth of 24 MHz and a damping factor of 0.7. The skirt of phase noise has been reduced a lot. It may still be seen that the amplitude increases somewhat at increasing offsets, before decreasing again.

9.1 Phase Noise

Different resistance values R_2 in the loop filter

The first phase noise measurements are presented in Figure 9.3. It shows the phase noise for offset frequencies between 0 and 100 kHz on both sides of the carrier. The capacitor values in the loop filter are fixed at 680 nF which should correspond to a bandwidth of 24 kHz. The resistor values have been varied from 10 Ω to 62 Ω which should correspond to a damping factor from 0.15 to 3.2. All the plots for phase noise are scaled according to the noise bandwidth and the signal level and therefore

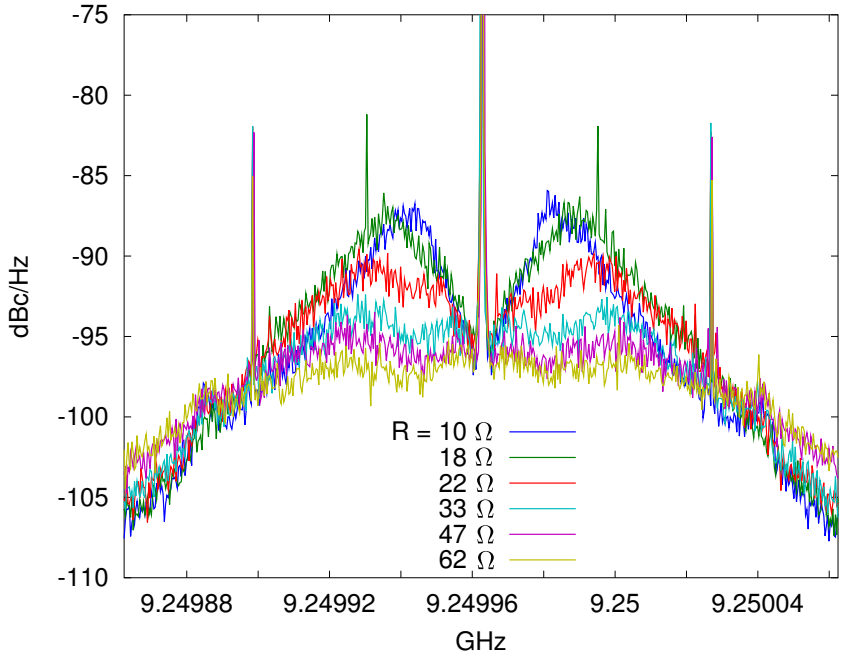


Figure 9.3: *Phase noise with different damping factors. Span 200 kHz.*

present the results in dBc/Hz.

It is noted that when the resistor value is small, there is a visible increase of phase noise at offsets a bit away from the carrier. The maximum is seen at about 19 kHz for the 10 Ω resistor value and at about 35 kHz for the 33 Ω resistor. For resistance values of 47 Ω and higher, there is no such clear increase of phase noise.

The phase noise is in general lower near the carrier for the higher resistor values. At higher offsets, which are near the edge of the plot, the opposite is the case. The higher the resistance is, the more constant the phase noise spectrum seems to be. Undesired spikes are observed at 62 kHz for all of them and at 32 kHz offset for the 18 Ω case. These are most likely caused by spurs.

Different capacitor values in the loop filter

The spectra for different capacitor values with a fixed resistor value of 22 Ω are presented in Figure 9.4. The two capacitor values used are 470 nF and 680 nF. 470nF should give the bandwidth 29 kHz and 680 nF, 24 kHz. The span is in this case also 200 kHz so that it allows the phase noise for offset frequencies up to 100 kHz to be seen. An increase of the phase noise at higher offsets is observed which would be expected as the previous measurement also indicates such an increase with resistor values below 47 Ω. The noise decreases again at offsets higher than 32 kHz. Maximum noise is about -90 dBc/Hz at 32 kHz offset for both of them. It is slightly higher for 680 nF. The spikes at 62 kHz are seen in both cases.

With 470 nF, lower phase noise at offsets below 50 kHz is observed. At greater offsets, the phase noise is more or less the same.

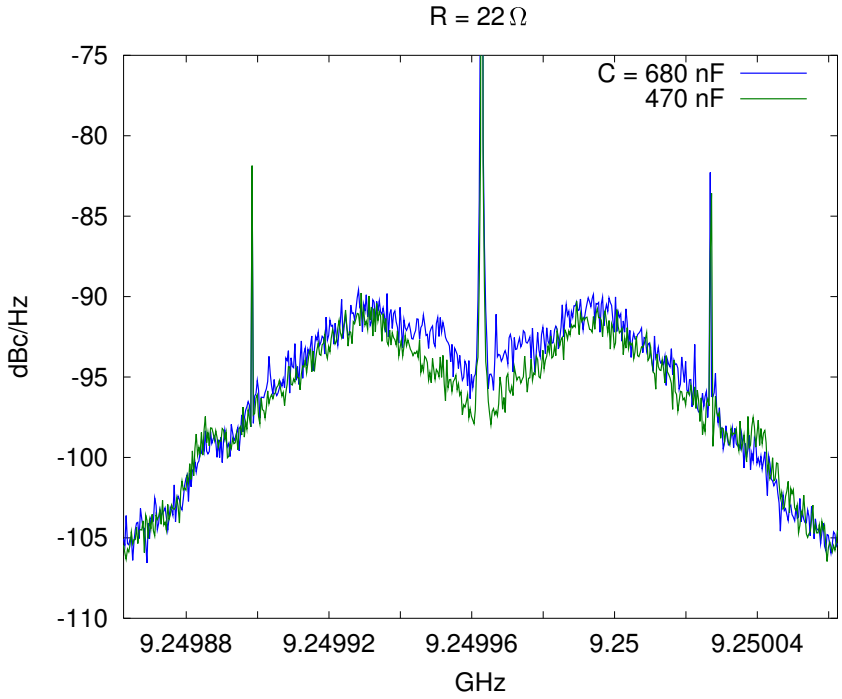


Figure 9.4: *Phase noise for different capacitor values with R_2 constant 22 Ω. Span 200 kHz.*

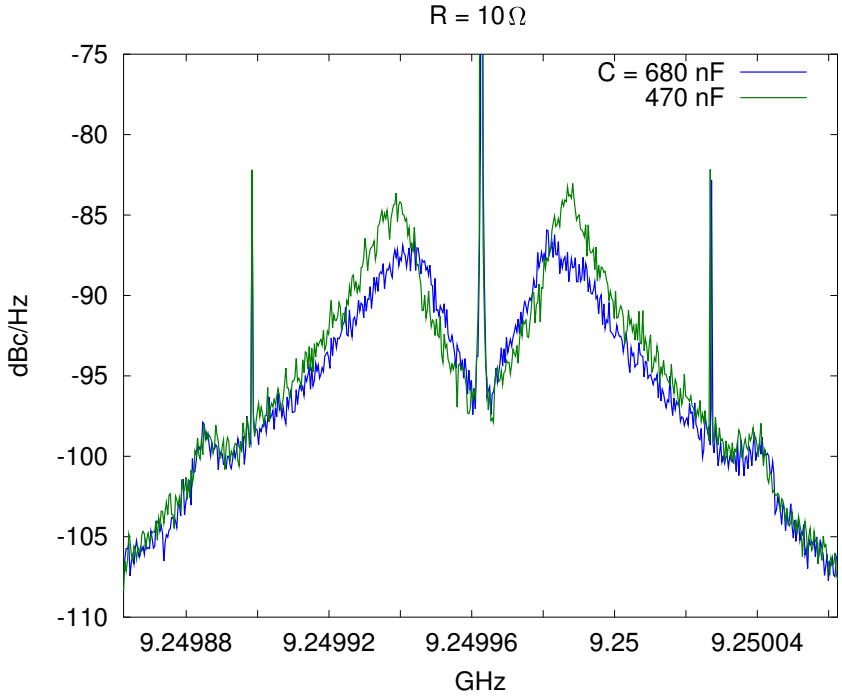


Figure 9.5: *Phase noise for different capacitor values with R_2 constant 10 Ω. Span 200 kHz.*

In Figure 9.5, the result from the measurement with $R_2 = 10\ \Omega$ is presented. The capacitor is also in this case varied between 470 nF and 680 nF. It can be observed that for offsets below 19 kHz, the noise is lower with $C_1 = 470$ nF, while for offsets between 19 kHz and 62 kHz, the noise is lower with $C_1 = 680$ nF. There is a clear peaking of about -84 dBc/Hz at 23.5 kHz for $C_1 = 470$ nF and of about -87 dBc/Hz at 20.5 kHz for $C_1 = 680$ nF. This is much higher than for the previous measurements. The spurs at 62 kHz are still there.

Phase noise option in the R&S FSQ spectrum analyzer

The measurements performed with the phase noise option (Figure 9.6) reveils that the phase noise found and scaled earlier is correct. The expected bandwidth is 24 kHz and the expected damping factor 1.7. It also includes the phase noise at offsets up to 1 MHz. The results are presented with a logarithmic frequency axis and the plot can therefore more or less be compared directly with the predictions in Section 6.2.

It can be noted that the noise remains quite constant from 1 kHz offset to 20-30 kHz offset, before it starts decreasing at a relatively constant rate (logaritmically) from 30 kHz to the edge of the plot. There is a little peak at about 230 kHz, but the spike observed earlier at 62 kHz is gone.

Many more measurements of the phase noise were performed, but only some were selected to be presented here.

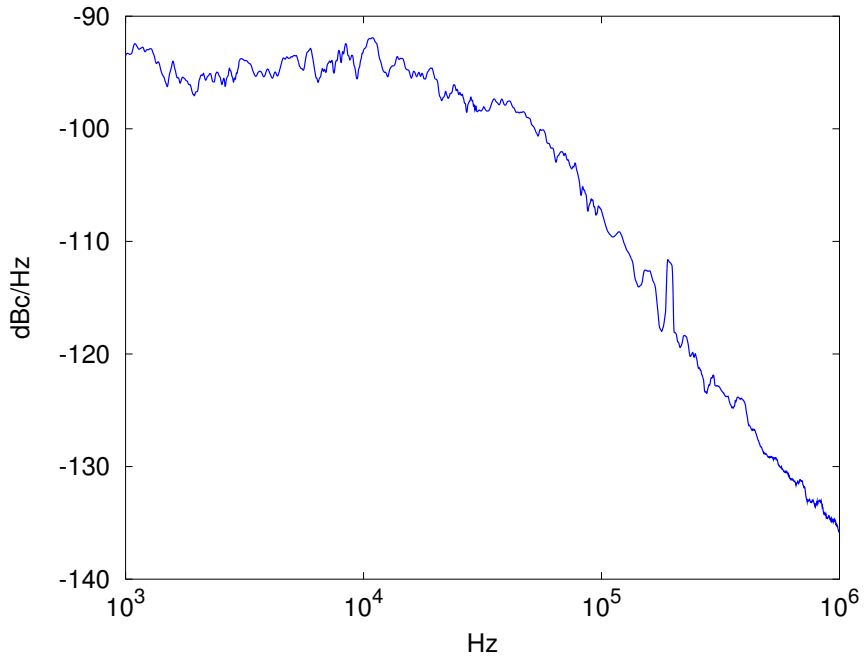


Figure 9.6: *Phase noise measured with phase noise option on FSQ. The capacitor values are here 680 nF and the resistor values 33* Ω.

9.2 Frequency Step

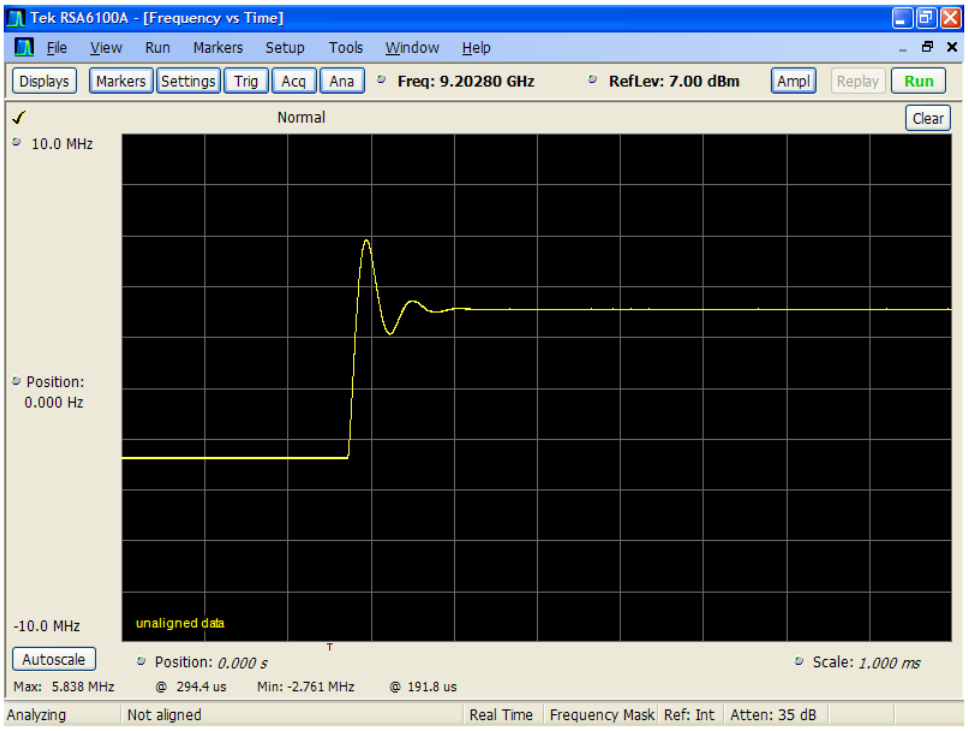


Figure 9.7: *Response of a frequency step observed on a real-time spectrum analyzer, $R_2 = 10\ \Omega$, $C_1 = 470$ nF. Frequency vs time*

Capacitance	Resistance	Bandwidth	Damping
470 nF	10 Ω	23 kHz	0.22
470 nF	22 Ω	24 kHz	0.55
470 nF	33 Ω	24 kHz	0.8
680 nF	10 Ω	19.5 kHz	0.31
680 nF	22 Ω	19.5 kHz	0.67
680 nF	33 Ω	19.5 kHz	1.01

Table 9.1: *Measured bandwidth and damping factors for different values of C and R*

A screenshot of the frequency step response measurement is shown in Figure 9.7.There is clearly a ringing response which indicates that the damping factor is somewhat too low. For this measurement, the bandwidth was expected to be 29 kHz and the damping factor 0.43.

Plots of the measurements with different resistor values are presented in Figure 9.8. It can be observed that with a higher resistor value, the ringing disappears. At 22 Ω there is still a small tendency to ringing, while at 33 Ω there is just one single overshoot before the frequency settles.

The same measurements with another capacitor value are presented in Figure 9.9. What is observed, is more or less the same as in the previous figure, but the response

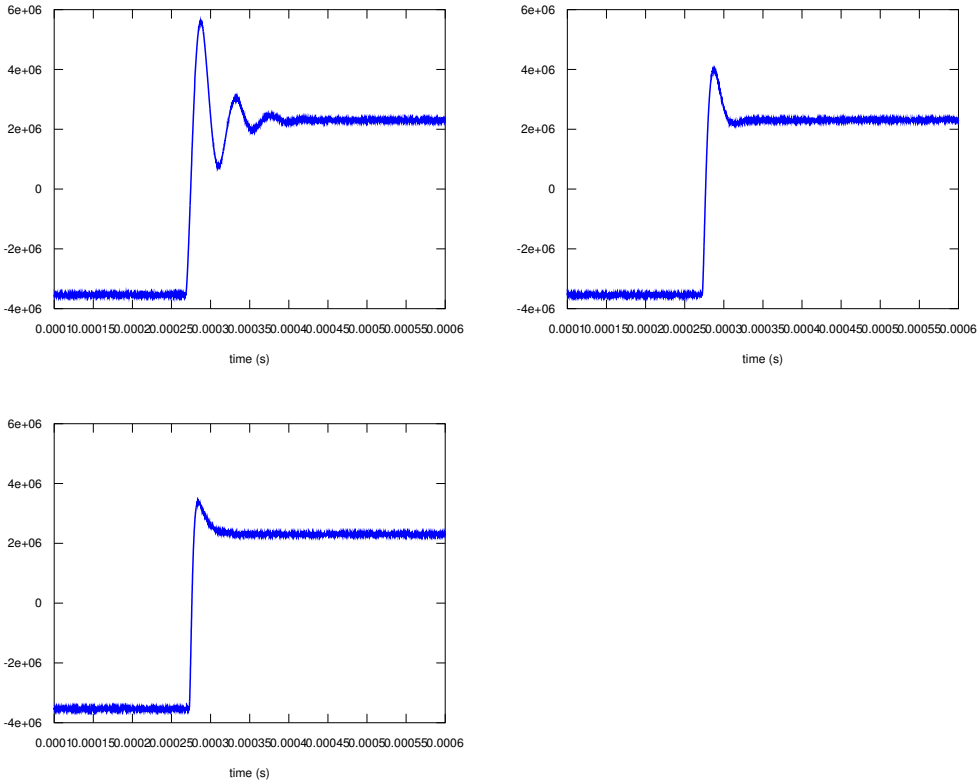


Figure 9.8: *Responses of frequency steps observed on a real-time spectrum analyzer. $C_1 = 470$ nF. For three different values of R_2 - 10 Ω, 22 Ω and 33 Ω*

is slightly more damped. In addition, it is noticed that the ringing is slower - the frequency is lower.

From all these responses, the bandwidth and damping factor could be found. It was done by fitting curves from the expressions in Section 4.4 page 41 with the measurements. The result is presented in table 9.1

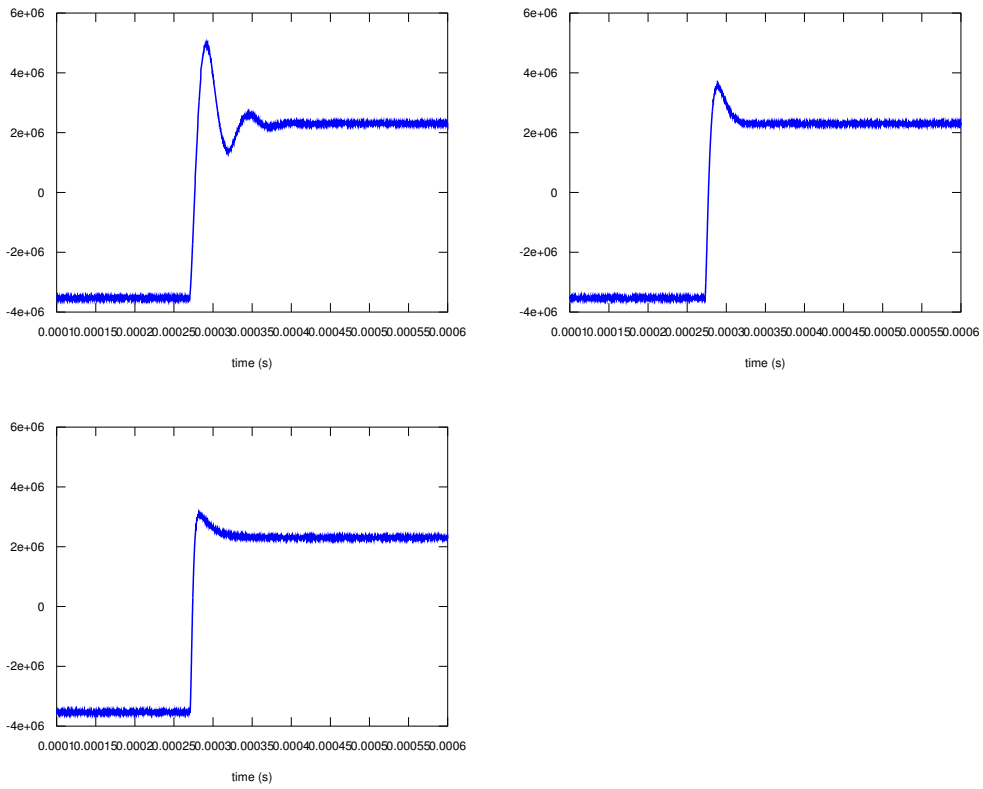


Figure 9.9: *Responses of frequency steps observed on a real-time spectrum analyzer, $C_1 = 680$ nF. For three different values of R_2 - 10 Ω, 22 Ω and 33 Ω*

9.3 Sweep

Asymmetrical sweep

The result of the asymmetrical sweep can be seen in Figure 9.10. Note the little delay after each return-to-start before it starts sweeping again. The delay lasts for 1.7 ms. The sweep all in all lasts for 11.88 ms. Also note the overshoot when the frequency switches back to the start frequency. The sweep bandwidth is 26.9 MHz and the overshoot when returning is 3.2 MHz, 12 % of the total sweep bandwidth. The capacitor value C_1 used here was 680 nF and the resistor R_2, 33 Ω.

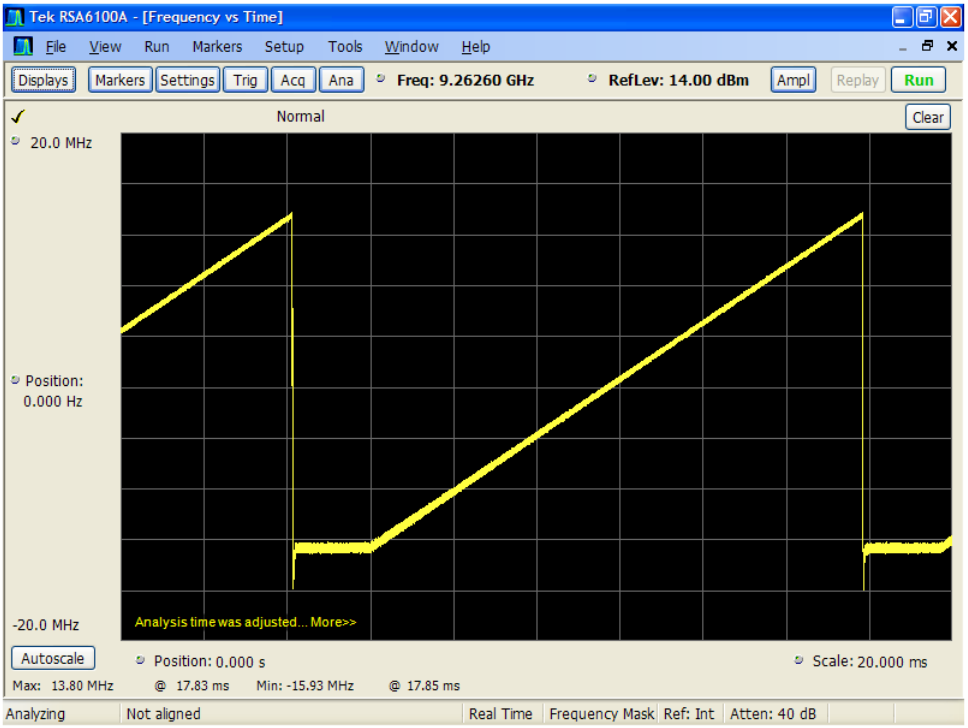


Figure 9.10: *Asymmetrical sweep observed with real-time spectrum analyzer*

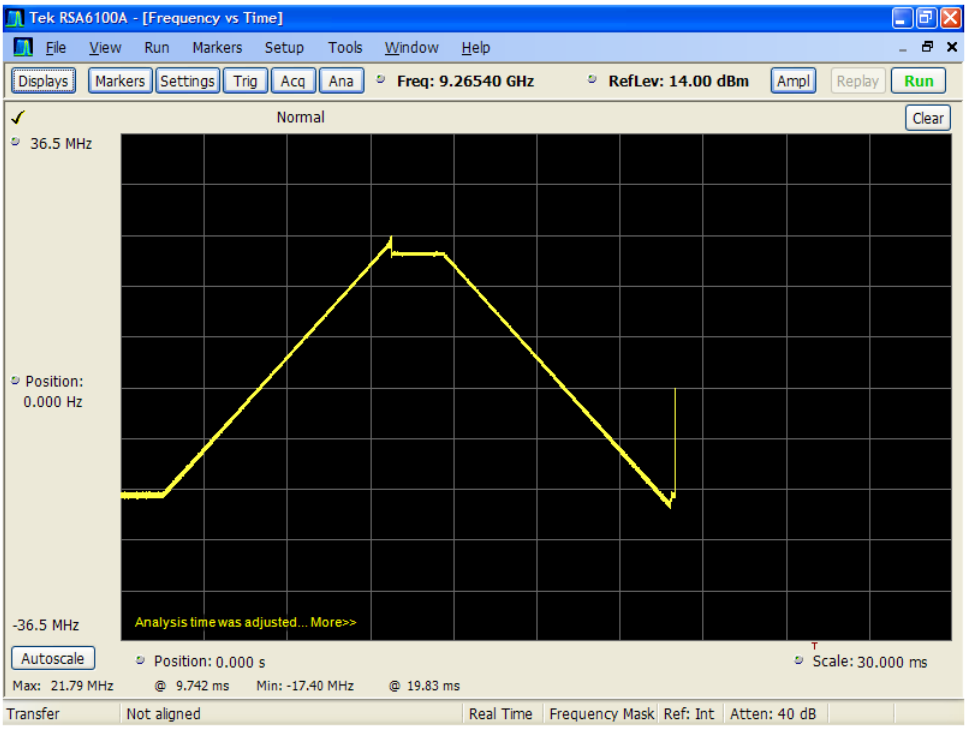


Figure 9.11: *Symmetrical sweep observed with real-time spectrum analyzer*

Symmetrical sweep

Since the return rate and start frequency have to be set manually by the MCU, the first attempt to make a symmetrical sweep ended up like Figure 9.11 presents. The up-sweep reaches a higher frequency than the one set for the beginning of the down-sweep. It therefore looks a bit ugly. The new frequency was adjusted slightly and new tests were performed. In Figure 9.12, the problem is gone. The maximum frequency is however slightly higher than for the previous case.

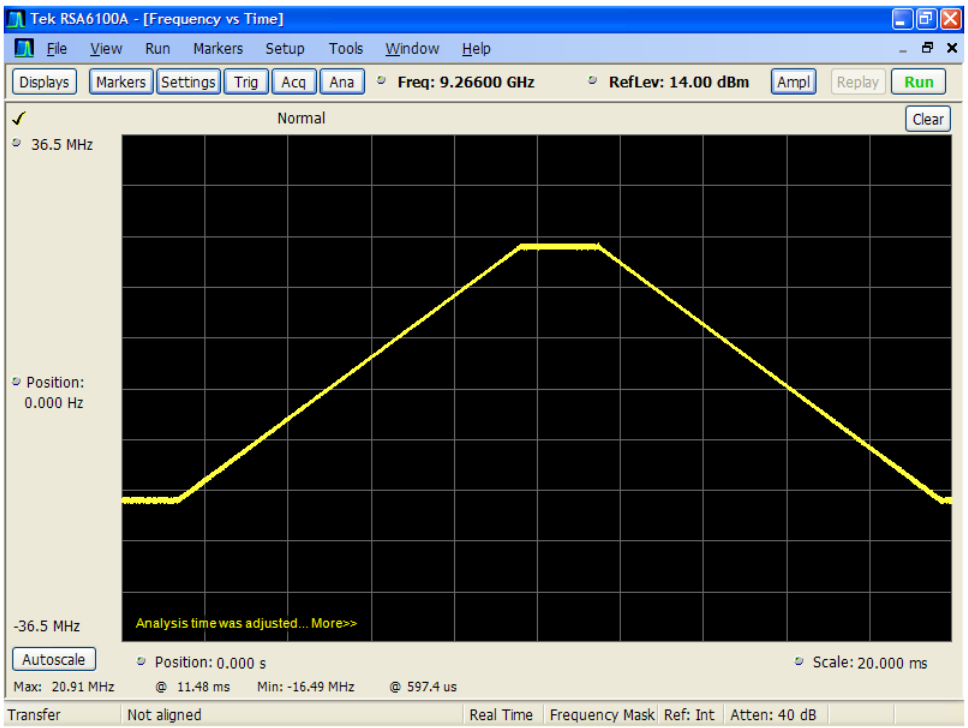


Figure 9.12: *Symmetrical sweep observed with real-time spectrum analyzer*

Chapter 10

Discussion

In the previous chapters, a frequency synthesizer for a linear FMCW radar in the 9.2-9.3 GHz band has been constructed and tested. It has proved to be a successfull implementation that outputs a signal with the desired frequency and amplitude. This chapter discusses the results, mainly from the measurements, and compares them with the expectations.

10.1 Frequency Step Measurements

The frequency step measurement results are the first to be discussed (Section 9.2). It is done because these results provide important data which will be useful later in the discussion. As expected, the frequency step from the DDS caused the VCO to have a ringing step response. The amount of ringing was higher with smaller R_2 in the loop filter. The results are seen in Figure 9.8 and 9.9 page 99 and 100. For the case with the capacitor $C_1 = 470$ nF, the ringing response was observed to be stronger than for $C_1 = 680$ nF. Bandwidths, f_n, and damping factors, ζ, for all the steps were presented in table 9.1. It is repeated here along with the predicted values.

Values	**Expected f_n**	**Measured**	**Expected ζ**	**Measured**
470 nF, 10 Ω	29.1 kHz	23 kHz	0.43	0.22
470 nF, 22 Ω	29.1 kHz	24 kHz	0.95	0.55
470 nF, 33 Ω	29.1 kHz	24 kHz	1.42	0.8
680 nF, 10 Ω	24.2 kHz	19.5 kHz	0.52	0.31
680 nF, 22 Ω	24.2 kHz	19.5 kHz	1.14	0.67
680 nF, 33 Ω	24.2 kHz	19.5 kHz	1.71	1.01

Table 10.1: *Measured bandwidth and damping factors for different values of C and R compared with theoretical predictions*

The difference between theoretical and observed values is rather big. This is especially the case for the damping factors. The theoretical values were found based on the equations in (4.16) page 41. What was assumed when using them was that the

PLL is linear. The equations are rather simple and do not take into account effects of the circuit board for instance.

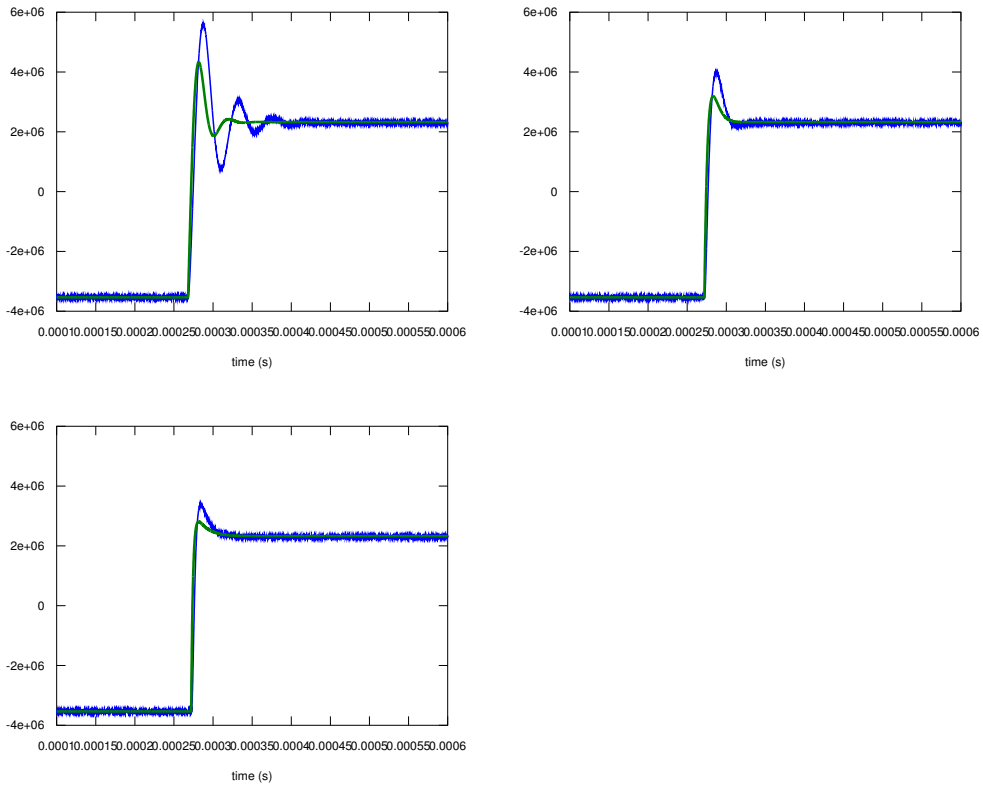


Figure 10.1: *The measured frequency step response compared with the calculated*

In Figure 10.1, the observed response is plotted (blue) together with the calculated response (green) found with Equation (4.20) and (4.21) page 41. The measured step response is clearly more ringing than what was predicted. It also oscillates at a lower frequency.

Based on the bandwidth prediction error, it is suspected that either K_d, K_0 were assumed too high or that the effective value of R_1 was assumed too low. It may also have been caused by some attenuation anywhere in the loop.

The damping factor prediction error is more tricky. It is almost a factor of 2 wrong. One of the few ways to get this behaviour in the formulas for f_n and ζ is to scale the resistor R_2 with a constant.

A correction to the formula was derived by assuming that K_d was wrong and that the effective value of R_2 was smaller than its actual value.

$$K_{d\text{-corrected}} = 0.65 \cdot K_d$$

$$R_{2\text{-effective}} = 0.70 \cdot R_2$$

$$f_n = \frac{1}{2\pi}\sqrt{\frac{K_{d\text{-corrected}}K_0}{NR_1C}} \qquad \zeta = \frac{R_{2\text{-effective}}}{2}\sqrt{\frac{CK_{d\text{-corrected}}K_0}{NR_1}} \tag{10.1}$$

That $R_{2\text{-effective}}$ is smaller than R_2 can be caused by multiple factors. First of all, the loop filter is not perfect. The loop filter receives pulses at a rate of 121 MHz, and the actual design may create capacitors that reduce the effective resistance (impedance). It is also possible that the configuration is not symmetric enough and hence the asymmetry will play a role. The op-amp has been powered with a single voltage supply and not a double-voltage which it usually is. This in itself may lead to changes in the behaviour. The PLL is considered linear in the calculations. Perhaps its nonlinearities play such a big role that they cause big deviations. This should however not cause a systematic error like here for multiple values of ζ. The frequency step is quite small (8 MHz at the operating frequency, 91 kHz at the phase detector frequency), and it should be expected that the PLL operates in its linear region. Note that all this is speculation. It is also possible, under doubt, that the prediction error simply is caused by calculation errors.

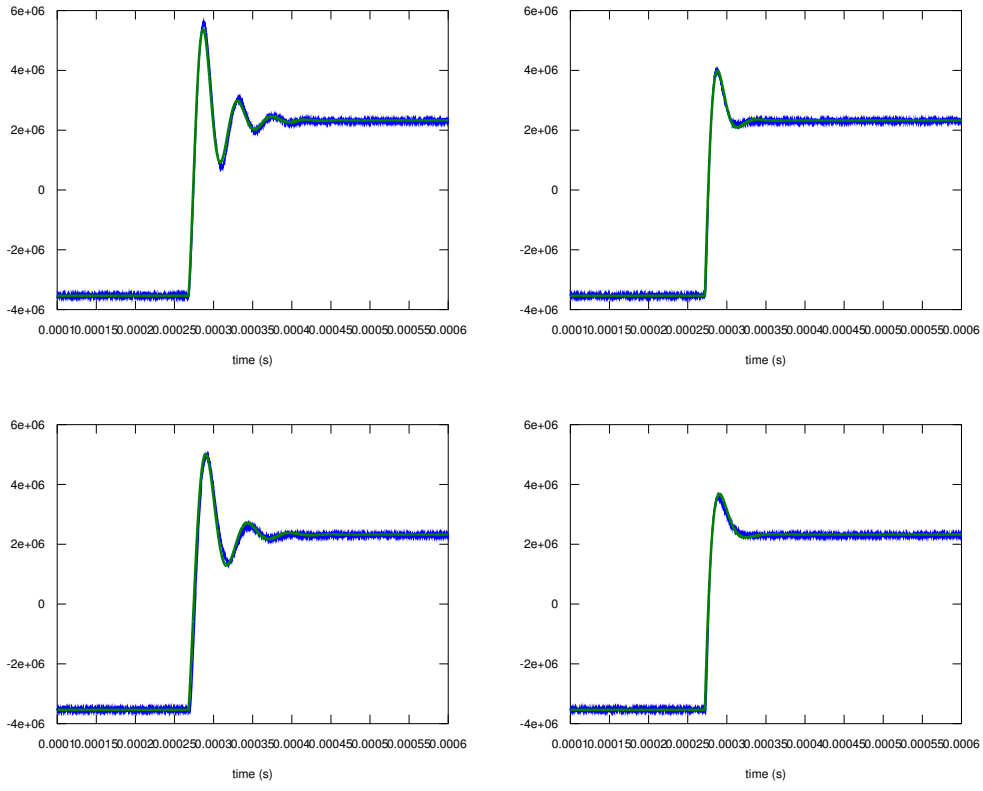


Figure 10.2: *The measured frequency step response compared with predictions with corrected formula for $C = 470$ nF (upper two) and $C = 680$ nF (lower two)*

However, even though the absolute values of f_n and ζ do not correspond directly to the calculated values, the trend certainly matches the theory. It is clearly observed

that by increasing the resistance, the response becomes more damped, while increasing the capacitance gives a lower ringing frequency. By employing the correction formula developed above, the measured values are again compared with calculated values for several different values of C_1 and R_2. The results are plotted in Figure 10.2. It can be observed that the formula fits very well. The trend has therefore been verified.

10.2 Phase Noise Measurements

The effect on the phase noise of changing the resistor value (Figure 9.3 page 95) corresponds well to theory. When the resistor value R_2 increases, and hence also ζ, the phase noise spectrum becomes more and more flat. It corresponds well with the predictions made in Chapter 6.

What is not so good, is that the absolute level of the phase noise is considerably higher than predicted. It was expected to see it around -105 dBc/Hz at its best, but instead it is at about -97 dBc/Hz. One thing that should be noted is that the Anritsu Spectrum analyzer has such a high phase noise, that for the measurements with the highest resistor values, the measurements are most likely influenced by the instrument phase noise. In addition, the bandwidth and damping factors used when predicting the phase noise were in the previous section shown to be wrong.

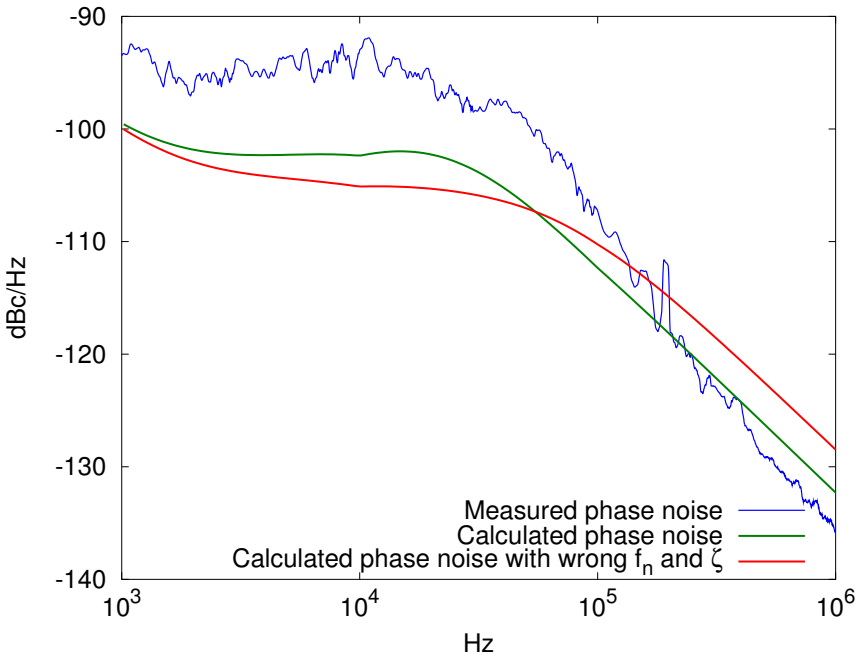


Figure 10.3: *The measured phase noise compared with the expected*

The result from the phase noise option in the FSQ spectrum analyzer is not so easily obscured by instrumentation phase noise. It is compared with calculated values in

Offset	**Estimated** (dBc/Hz)	**Measured** (dBc/Hz)	**Difference** (dBc/Hz)
1 kHz	−99.6	−93.5	+6.1
10 kHz	−102.4	−93.0	+9.4
100 kHz	−112.3	−107.3	+5.1
1 MHz	−132.3	−135.8	−3.5

Table 10.2: *The measured phase noise compared with the estimation*

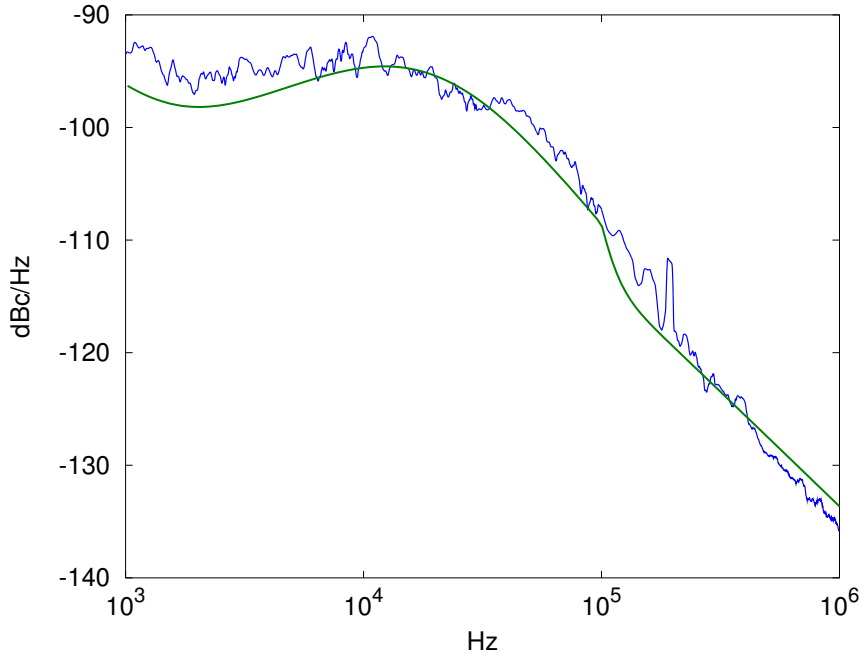


Figure 10.4: *Excessive noise in the loop filter will cause such a behaviour*

Figure 10.3. Two different predictions are plotted. Here the reward of first finding the real PLL bandwidth using frequency step testing is seen (Section 10.1). The red curve is based on f_n and ζ first predicted, while the green curve is based on the new values for f_n and ζ found by analyzing the step response. It can be seen that there is quite a lot more phase noise than calculated. It is however readily seen that the green curve fits the measured phase noise better than the red curve. It starts decreasing at about the same offset.

Table 10.2 summarizes the deviation between predicted values (with correct bandwith and damping factor) and the measured value. It is observed that at low offsets, the prediction is better than achieved, but at offsets approaching 1 MHz, the measurements are slightly better. It is not good that there is so much more phase noise than expected. It should be noted however, that the calculated phase noise mostly corresponds to the best obtainable phase noise. This may seem strange since the phase noise at high offsets got better. The last may be caused by the fact that the low-pass filter following the loop filter (Figure 7.11 page 75) was not included when the predictions of the phase noise were made.

In order to try to find out what could cause the additional phase noise, new calculations were done. By adding 14 dB to the phase noise of the loop filter at offset up to 500 kHz, decreasing the noise slightly at 1 MHz and above, as well as adding 10 dB to the noise from the reference, a new plot was made.

The result is seen in Figure 10.4. The curve seems to fit well. Based on this discovery, it might be suspected that the main problem is noise from the loop filter. By holding a screwdriver against a metal point in the loop filter, it was observed that the phase noise could easily be increased 20-30 dB more. It is therefore obvious that what happens regarding noise in the loop filter can be extremely devastating for the performance.

Different things were attempted in order to find the cause. It was suspected that the power supply could play a role. Figure 10.5 shows what happened when different parts of the board were powered with independent power supplies and also when the power to the op-amp was powered by a battery with short cables. Only one sideband is shown. The result shows that there was not much to be gained by using batteries for the op-amp, but by separating the supply for the VCO from the DDS, the phase noise was actually improved quite much. This just illustrates the fact that the noise may come via the supply voltage or more likely propagate via the supply voltage wiring. One error had probably been made during the design. Every subsection of the board should have had its own low noise voltage regulator with several capacitors of different sizes. Apparently, a power plane and some capacitors did not suffice. Unfortunately, there was not any time left to try this out. There was only time for one attempt of making a synthesizer. It should however be appreciated that it at least worked - in spite of the phase noise not being optimal.

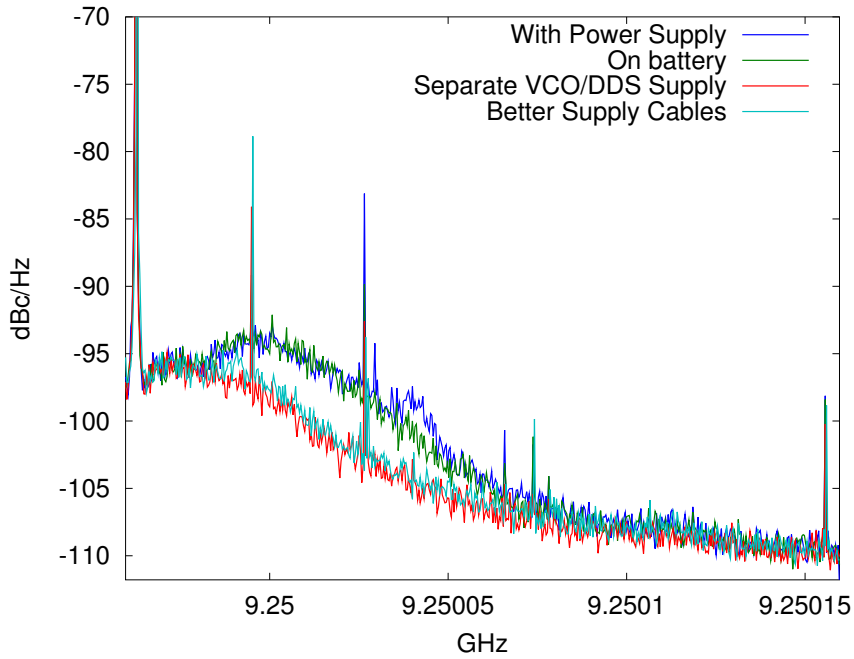


Figure 10.5: *Testing the effect of changing power supplies*

10.3 Frequency Sweep Measurements

The frequency sweep measurements did not reveal so many surprises. The output swept where it was told to sweep. One thing that can be noted is that since the sweep type employed for the measurements was short, the time the microcontroller needed in order to send the control commands to the DDS was a significant part of the period. First of all this caused the delays at the top and bottom of the sweep. Two commands had to be sent for resetting the frequency incrementer. For the case with symmetrical sweep, a new frequency also had to be set at the end of each sweep. Because the AD9858 buffers the instructions and first make them active after a toggle on the *FUD* pin, most of the commands could be sent during the sweep. This obviously caused the microcontroller to not have to wait for such a long time. Almost 2 ms was needed in order to send one command, and for the symmetrical sweep, the sleeping time of the micocontroller had to be set to the total sweep time minus the time for two commands, 4 ms. The slow control commands are a result of that the microcontroller runs at a frequency of 1 MHz. The DDS is capable of receiving instructions at a rate of 10 MHz [29]. The clock divider that is enabled by default inside the ATtiny2313 transforms an internal 8 MHz oscillator into a 1 MHz clock signal. This divider could easily be disabled by a fuse setting, and the command delay would then be 8 times as short.

Another possibility is to run the microcontroller from an external oscillator. This was not included in the design because it slightly complicates it more. The SYSCLK output of the AD9858 could have been divided down and used as the clock oscillator to get 10 MHz. In that case, the microcontroller would have been synchronized

completely to the DDS, and the frequency unaccuracies would be defined by one single very good crystal oscillator instead of an unsynchronized unstable on-chip oscillator.

10.4 Overall Design

One thing that was discovered quite rapidly after the synthesizer was powered up was that it became quite hot. In fact, after a while it was so hot that it could barely be touched. The components generating most of the heat were the two Hittite components - the VCO and the phase detector. Each of them was using 1.5-2 W and they are very small. Fortunately, the ground pads underneath them had been connected to the ground plane of the board with several vias in order to conduct the heat away. But still, the board became very hot. In fact, the temperature of the circuits was measured with an infrared thermometer and seen to approach 80°C once. Its specification says that the operating temperature is from -40 to +85°C, and it was therefore almost too hot. It might be expected that the phase noise performance can be degraded because of this. According to the datasheet for the VCO, the phase noise should just become a couple of dB higher at offsets from 100 Hz to 20 kHz, but remain quite the same for higher offsets. This region is mainly regulated by the PLL and should therefore not be of main concern. All in all, a cooling element / fan should be mounted to the board if it is to operate over a significant amount of time. When the temperature raises so high during inital testing in the lab, it is very probable that it can get even higher when the radar is in the field.

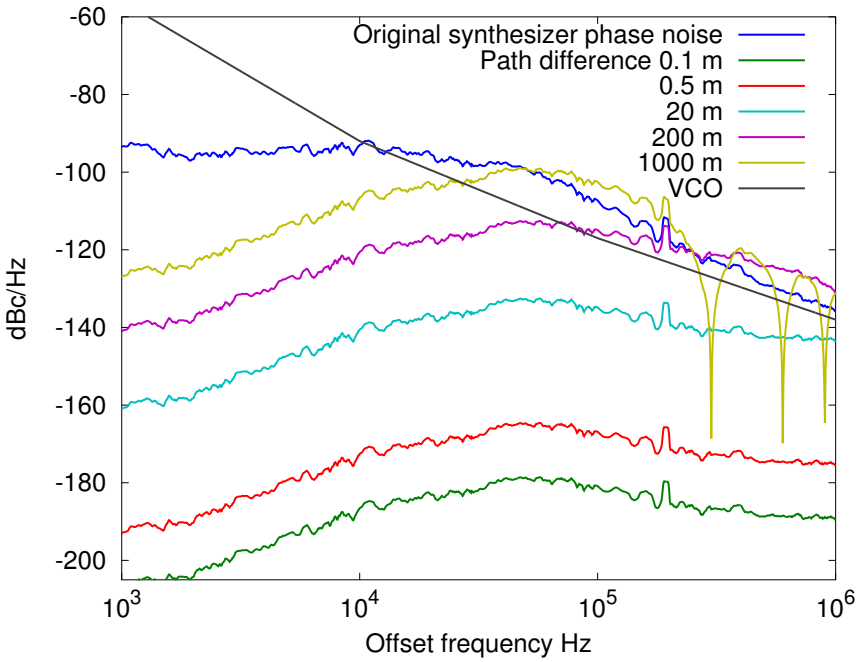


Figure 10.6: *Cancellation because of correlation of measured phase noise*

The output of the synthesizer has a power of +11 dBm ≈ 12.6 mW. This is probably

not enough to be connected directly to the antenna. In order to have a functional radar, an output level in the range of a few watts would be better. Therefore a power amplifier will be needed. This has not been considered in this design, but an amplifier with a gain of about 20 dB would be desired. Its main requirement is obviously that it should not add much phase noise.

The cancellation factor from Chapter 5 was used to calculate the effective radar phase noise for the measured phase noise spectrum of the frequency synthesizer exactly as was done earlier for the open-loop VCO. The result is presented in figure 10.6 for several different path lengths differences. The effective phase noise can be made very low and for very short path lengths, it will fall below the thermal noise floor. The phase noise of the open-loop VCO is included in the plot too.

Chapter 11

Conclusion

In this thesis, there has been a focus on radars, phase noise and PLLs. In addition, a frequency synthesizer in the X-band at 9.2-9.3 GHz has been designed, built and tested. Generally speaking, the results are good. The functionality of the circuit is successful. However, the absolute magnitude of the phase noise is higher than predicted. It has been shown that this can be caused by many factors, but the most likely source is excessive noise in the loop filter of the PLL possibly caused by unstable supply voltage. To tweak the last dB's away from the phase noise, especially care must be taken during the design. All the critical components should have their own low-noise voltage regulators to provide reliable supply voltages.

While working with the thesis, a great amount of knowledge has been acquired in a multitude of different disciplines of electronics engineering. The work has provided experience in PCB design, programming, soldering and spectrum analyzing. Phase noise appears as a comprehensive subject to the author. After studying it for a while, there are still new things and new aspects of it that show up, but in general, everything is now much clearer than before. Understanding how the phase noise spectrum relates to the signal spectrum was especially interesting. An ideal carrier has infinite power at a single frequency, but the phase noise ensures that the signal is realistic and distributes its power over a finite bandwidth. Seeing how phase noise gets canceled because of correlation effects, was another interesting thing. It also has several potential uses. The perhaps most triggering thing was that the circuit that was designed and built worked very well. Unfortunately, there was not time enough to test it so extensively that the direct cause of the phase noise prediction error could be established.

It has been a fun time. Trying out new things and learning new things are always fun. But at the same time, it has been stressful. A lot had to be done to arrive here.

11.1 Future Work

As a future project, a power amplifier should be built, so that the synthesizer can be used in practice. The radar receiver must also be implemented. Since the transmitted

signal is also used as the local oscillator of the receiver, the output can be split to the power amplifier and the receiver. An interesting project may also be to implement the digital signal processing on an FPGA. An FFT processor is the main component.

Regarding the synthesizer designed here, it would be interesting to find out the direct cause of the failure to exactly predict the bandwidth and damping of the PLL. Spending some extra time finding out why there was more phase noise than predicted, is perhaps neither a bad idea.

List of Figures

Bibliography

[1] A.G. Stove. Linear fmcw radar techniques. *Radar and Signal Processing, IEE Proceedings F*, 139(5):343 –350, oct 1992.

[2] D.B. Leeson. A simple model of feedback oscillator noise spectrum. *Proceedings of the IEEE*, 54(2):329 – 330, feb. 1966.

[3] A. Hajimiri and T.H. Lee. A general theory of phase noise in electrical oscillators. *Solid-State Circuits, IEEE Journal of*, 33(2):179 –194, feb 1998.

[4] A. Demir, A. Mehrotra, and J. Roychowdhury. Phase noise in oscillators: a unifying theory and numerical methods for characterization. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 47(5):655 – 674, may 2000.

[5] David K. Barton. *Radars - CW and Doppler Radar*. Artech House, 1980.

[6] Merrill I. Skolnik. *Introduction to Radar Systems*. McGraw-Hill, 3rd edition, 2001.

[7] Merrill I. Skolnik. *Radar Handbook*. McGraw-Hill, 2nd edition, 1990.

[8] W. Holm and J. Echard. Fft signal processing for non-coherent radar systems. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, volume 7, pages 363 – 366, may 1982.

[9] Ole K. Nilsen and W. D. Boyer. Amplitude modulated cw radar. *Aerospace and Navigational Electronics, IRE Transactions on*, ANE-9(4):250 –254, dec. 1962.

[10] Bill Mullarkey. The differences between pulse radars and fmcw ones. *Navigate-us.com*, dec 2008.

[11] Igor V. Komarov and Sergey M. Smolskiy. *Fundamentals of Short-Range FM Radar*. Artech House, 2003.

[12] United States Department of Commerce. United states frequency allocations chart 2011 - the radio spectrum. *http://commons.wikimedia.org/wiki/File:United_States_Frequency_Allocations_Chart_2011_-_The_Radio_Spectrum.pdf*, 2011.

[13] E.S. Ferre-Pikal, J.R. Vig, J.C. Camparo, L.S. Cutler, L. Maleki, W.J. Riley, S.R. Stein, C. Thomas, F.L. Walls, and J.D. White. Draft revision of ieee std 1139-1988 standard definitions of physical quantities for fundamental, frequency

and time metrology-random instabilities. In *Frequency Control Symposium, 1997., Proceedings of the 1997 IEEE International*, pages 338 –357, may 1997.

[14] Michal Odyniec. *RF and Microwave Oscillator Design*. Artech House, 2002.

[15] W.P. Robins. *Phase Noise in Signal Sources*. Peter Peregrinus Ltd, 1982.

[16] Karl Rottmann. *Matematisk Formelsamling*. Spektrum forlag, 11th edition, 2003.

[17] Floyd M. Gardner. *Phaselock Techniques*. Wiley-Interscience, 3rd edition, 2004.

[18] T.H. Lee and A. Hajimiri. Oscillator phase noise: a tutorial. *Solid-State Circuits, IEEE Journal of*, 35(3):326 –336, mar 2000.

[19] G.J. Coram. A simple 2-d oscillator to determine the correct decomposition of perturbations into amplitude and phase noise. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 48(7):896 –898, jul 2001.

[20] Mike Curtin and Paul O'Brien. Phase-locked loops for high-frequency receivers and transmitters - part 2. *Analog Dialogue by Analog Devices*, 1999.

[21] David Owen. Good practice to phase noise measurement. *Pickering Interfaces*, (68), 2004.

[22] J.I. Brown. A digital phase and frequency-sensitive detector. *Proceedings of the IEEE*, 59(4):717 – 718, april 1971.

[23] Floyd M. Gardner. Charge-pump phase-lock loops. *Communications, IEEE Transactions on*, 28(11):1849 – 1858, nov 1980.

[24] Jr. M.C. Budge and M.P. Burt. Range correlation effects in radars. In *Radar Conference, 1993., Record of the 1993 IEEE National*, pages 212 –216, 1993.

[25] M. Pichler, A. Stelzer, P. Gulden, and M. Vossiek. Influence of systematic frequency-sweep non-linearity on object distance estimation in fmcw/fscw radar systems. In *Microwave Conference, 2003. 33rd European*, volume 3, pages 1203 – 1206 Vol.3, oct. 2003.

[26] G.M. Brooker. Understanding millimetre wave fmcw radars. *1st International Conference on Sensing Technology*, Nov. 2005.

[27] Hittite Microwave Corporation. *HMC510LP5 Datasheet*.

[28] Valpey Fisher Corporation. *VFTX210 Datasheet*.

[29] Analog Devices. *AD9858 Datasheet*.

[30] Ken Gentile. Driving a center-tapped transfomer with a balanced current-output dac. *Analog Devices - Application Note AN-912*.

[31] Analog Devices. Ad9858 evaluation board. *http://www.analog.com/en/evaluation/EVAL-AD9858/eb.html*.

[32] Dale Heatherington. Design an l-c low pass or high pass filter. *http://www.wa4dsy.net/filter/hp_lp_filter.html*, May 2008.

[33] Atmel. *AVR ATtiny2313 Datasheet*.

[34] Hittite Microwave Corporation. *HMC698LP5 Datasheet*.

[35] Peter White. Understanding phase noise from digital components in pll frequency synthesizers. *Applied Radio Labs*, DN006, dec. 2000.

[36] T.J. Endres and J.B. Kirkpatrick. Sensitivity of fast settling plls to differential loop filter component variations. In *Frequency Control Symposium, 1992. 46th., Proceedings of the 1992 IEEE*, pages 213 –224, may 1992.

[37] Analog Devices. *ADA4898-1 Datasheet rev C*, 2010.

[38] Microstrip line calculator. *http://www1.sphere.ne.jp/i-lab/ilab/tool/ms_line_e.htm*.

[39] Atmel. In-system programming. *Application Note*, AVR910, 2008.

[40] Analog Devices. A technical tutorial on digital signal synthesis. *1st International Conference on Sensing Technology*, 1999.

Appendix A

Source Code ATtiny2313

A.1 ad9858control.h

```
/*
        Software for ATtiny2313 on
        FMCW Synthesizer 9.2GHz board

        by
        Ruben Undheim
        Master thesis 2012 NTNU
*/

#ifndef INC_AD9858_H
#define INC_AD9858_H

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define DDS_PS0         _BV(PD2)
#define DDS_PS1         _BV(PD3)
#define DDS_FUD         _BV(PD4)
#define DDS_RESET         _BV(PD5)
#define DDS_CS        _BV(PD6)  //inverted!!!

#define DDS_SCLK         _BV(PB0)
#define DDS_SDIO         _BV(PB1)
#define DDS_SDO         _BV(PB2)
#define DDS_IORESET         _BV(PB3)

#define CTRL_PWRDOWN         2
#define CTRL_2GHZ        6
#define CTRL_PWRMIX         4
#define CTRL_PWRPD        3
#define CTRL_SWEEP        15
#define CTRL_CLRFREQ         20
#define CTRL_AUTOCLR         23

#define ACTIVE
#define DELAYBIT 1000
// Clear:
// PORTB &= ~DDS_SCLK;
// Set:
// PORTB |= DDS_SCLK;

// bits transferred at rising SCLK edge
```

```
44  // reading transferred on falling edge of SCLK
45  // LSB first
46
47  void delay_ms(uint16_t ms);
48  void delay_mini(uint16_t mini);
49  void send_byte(uint8_t byte);
50  void write_short(uint8_t address, uint16_t data);
51  void write_long(uint8_t address, uint32_t data);
52  uint32_t read_long(uint8_t address);
53  void ioreset(void);
54  void init(void);
55  void selectchip(uint8_t truefalse);
56  uint8_t receive_byte(void);
57  void set_control(uint32_t long_word);
58  void set_default(void);
59  void set_bit(uint8_t bitnr, uint8_t active);
60  void toggle_fud(void);
61  void set_profile(uint8_t profile);
62
63  extern uint32_t control_reg;
64
65  #endif /* INC_AD9858_H */
```

A.2 ad9858control.c

```
1   /*
2           Software for ATtiny2313 on
3           FMCW Synthesizer 9.2GHz board
4
5           by
6           Ruben Undheim
7           Master thesis 2012 NTNU
8   */
9   #define F_CPU 1000000UL
10
11  #include "ad9858control.h"
12
13  uint32_t control_reg;
14
15  void delay_ms(uint16_t ms) {
16          while ( ms )
17          {
18                  _delay_ms(1);
19                  ms--;
20          }
21  }
22  void delay_mini(uint16_t mini) {
23          while ( mini )
24          {
25                  asm("nop");
26                  mini--;
27          }
28  }
29  void send_byte(uint8_t byte){
30          uint8_t active;
31          uint8_t i;
32          for(i=0;i<8;i++){
33                  active = (0x01 & (byte >> (7-i)));
34                  if(active == 1){
35                          PORTB |= DDS_SDIO;
36                  }
37                  else if(active == 0){
```

```
38                          PORTB &= ~DDS_SDIO;
39                  }
40                  PORTB |= DDS_SCLK;
41                  PORTB &= ~DDS_SCLK;
42          }
43  }
44
45  uint8_t receive_byte(void){
46          uint8_t received = 0x00;
47          uint8_t active = 0x00;
48          uint8_t i;
49          for(i=0;i<8;i++){
50                  delay_ms(1);
51                  PORTB |= DDS_SCLK;
52                  delay_ms(1);
53                  active = ((PINB & DDS_SDIO) != 0);
54                  PORTB &= ~DDS_SCLK;
55                  delay_ms(1);
56                  received &= (active << (7-i));
57          }
58          return received;
59  }
60  void write_short(uint8_t address, uint16_t data){
61          DDRB |= DDS_SDIO;
62          address &= ~0x80;
63  #ifdef ACTIVE
64          selectchip(1);
65  #endif
66          send_byte(address);
67
68          send_byte((uint8_t)((data >> 8) & 0xFF));
69          send_byte((uint8_t)((data) & 0xFF));
70
71  #ifdef ACTIVE
72          selectchip(0);
73  #endif
74  }
75  void write_long(uint8_t address, uint32_t data){
76          DDRB |= DDS_SDIO;
77          address &= ~0x80;
78  #ifdef ACTIVE
79          selectchip(1);
80  #endif
81          send_byte(address);
82
83          send_byte((uint8_t)((data >> 24) & 0xFF));
84          send_byte((uint8_t)((data >> 16) & 0xFF));
85          send_byte((uint8_t)((data >> 8) & 0xFF));
86          send_byte((uint8_t)((data) & 0xFF));
87
88  #ifdef ACTIVE
89          selectchip(0);
90  #endif
91  }
92  uint32_t read_long(uint8_t address){
93          uint32_t read = 0x00;
94          DDRB |= DDS_SDIO; // write read command
95          address |= 0x80;
96  #ifdef ACTIVE
97          selectchip(1);
98  #endif
99
100         send_byte(address);
101
102         DDRB &= ~DDS_SDIO;  // start reading
103         PORTB &= ~DDS_SDIO;
104         read |= (((uint32_t)receive_byte() << 24) & 0xFF000000);
105         read |= (((uint32_t)receive_byte() << 16) & 0x00FF0000);
106         read |= (((uint32_t)receive_byte() << 8) & 0x0000FF00);
107         read |= ((uint32_t)receive_byte() & 0x000000FF);
```

```
108
109 #ifdef ACTIVE
110         selectchip(0);
111 #endif
112         return read;
113 }
114
115 void selectchip(uint8_t truefalse){
116         if(truefalse == 1)
117                 PORTD &= ~DDS_CS;
118         else if(truefalse == 0)
119                 PORTD |= DDS_CS;
120 }
121
122 void ioreset(){
123         PORTB |= DDS_IORESET;
124         delay_ms(1);
125         PORTB &= ~DDS_IORESET;
126         delay_ms(1);
127 }
128
129
130 void set_bit(uint8_t bitnr, uint8_t active){
131         uint32_t mask = ((uint32_t)0x00000001) << (bitnr);
132         if(active == 1)
133                 set_control((control_reg | mask));
134         else
135                 set_control((control_reg & ~mask));
136 }
137
138 void set_control(uint32_t long_word){
139         write_long(0x00,long_word);
140         control_reg = long_word;
141 }
142
143 void set_default(void){
144         set_control(0x00000018);
145 }
146
147 void set_profile(uint8_t profile){
148         if(profile == 1){
149                 PORTD = (PORTD & ~DDS_PS0) & ~DDS_PS1;
150         }
151         else if(profile == 2){
152                 PORTD = (PORTD | DDS_PS0) & ~DDS_PS1;
153         }
154         else if(profile == 3){
155                 PORTD = (PORTD & ~DDS_PS0) | DDS_PS1;
156         }
157         else if(profile == 4){
158                 PORTD = (PORTD | DDS_PS0) | DDS_PS1;
159         }
160 }
161
162 void toggle_fud(void){
163         PORTD |= DDS_FUD;
164         PORTD &= ~DDS_FUD;
165 }
166
167 void init(){
168         control_reg = 0x00000018;
169         DDRB = 0x0f ; /* Enable output on all the port B pins */
170         DDRB &= ~DDS_SDO;
171         DDRD = 0xfe ; /* Enable output on all the port B pins */
172         PORTB = 0x00 ; /* Set them to 0 */
173         PORTD = 0x00;
174         PORTD |= DDS_CS;
175         delay_ms(100);
176         ioreset();
177 }
```

A.3 main-fixedfrequency.c

```
/*
        Software for ATtiny2313 on
        FMCW Synthesizer 9.2GHz board

        by
        Ruben Undheim
        Master thesis 2012 NTNU
*/
#define F_CPU 1000000UL
#define N 13
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "ad9858control.h"


int main(void) {
        init();

        /* Divisor 19 */
        //uint32_t fvalue = 37796000; // very low

        //uint32_t fvalue = 463400000; // 8.2GHz
        //----uint32_t fvalue = 4.8036e8; // 8.5GHz
        //----uint32_t fvalue = 4.9731e8; // 8.8GHz
        //uint32_t fvalue = 5.1427e8; // 9.1GHz
        //uint32_t fvalue = 5.1992e8; // 9.2GHz
        uint32_t fvalue = 5.2274e8; // 9.25GHz
        //uint32_t fvalue = 5.2557e8; // 9.3GHz
        //uint32_t fvalue = 5.3122e8; // 9.4GHz
        //----uint32_t fvalue = 536870912; // 9.5GHz
        //----uint32_t fvalue = 5.4817e8; // 9.7GHz
        //uint32_t fvalue = 5.5382e8; // 9.8GHz


        /* Divisor 13 */
        //uint32_t fvalue = 7.0207e8; // 8.5
        //uint32_t fvalue = 7.1859e8; // 8.7
        //uint32_t fvalue = 6.7729e8; // 8.2
        //uint32_t fvalue = 7.51625e8; // 9.1
        //uint32_t fvalue = 7.5988e8; // 9.2
        //uint32_t fvalue = 7.6401e8; // 9.25
        //uint32_t fvalue = 7.6814e8; // 9.3
        //uint32_t fvalue = 7.764e8; // 9.4
        //uint32_t fvalue = 8.0944e8; // 9.8

        delay_ms(500);
        set_bit(CTRL_2GHZ,1);
        toggle_fud();
        write_long(0x03,fvalue);
        set_profile(1);
        toggle_fud();

}
```

A.4 main-switchfreq.c

```
/*
        Software for ATtiny2313 on
        FMCW Synthesizer 9.2GHz board

        by
        Ruben Undheim
        Master thesis 2012 NTNU
*/
#define F_CPU 1000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "ad9858control.h"


int main(void) {
        init();
        uint32_t fvalue = 5.1992e8; // 9.2GHz
        uint32_t fvalue2 = 5.2025e8;


        delay_ms(500);
        set_bit(CTRL_2GHZ,1);
        toggle_fud();
        write_long(0x03,fvalue);
        write_long(0x05,fvalue2);
        toggle_fud();
        while(1){
                set_profile(1);
                delay_mini(120);
                set_profile(2);
                delay_mini(120);
        }

}
```

A.5 main-sweeper.c

```
/*
        Software for ATtiny2313 on
        FMCW Synthesizer 9.2GHz board

        by
        Ruben Undheim
        Master thesis 2012 NTNU
*/

#define F_CPU 1000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "ad9858control.h"


int main(void) {
        init();
        uint32_t fvalue = 5.2274e8; // 9.25GHz
#define DFTW 1000
```

```
#define DFRRW 1000
        delay_ms(500);
        set_bit(CTRL_2GHZ,1);

        toggle_fud();
        write_long(0x03,fvalue);
        toggle_fud();
        set_bit(CTRL_SWEEP,1);
        toggle_fud();
        write_long(0x01,DFTW);  // frequency step size
        write_short(0x02,DFRRW); // how much time
        toggle_fud();

        while(1){
                toggle_fud();
                set_bit(CTRL_CLRFREQ,0);
                toggle_fud();
                set_bit(CTRL_CLRFREQ,1);
                //delay_ms(45*speed/step);
                delay_ms(10);
        }


}
```

A.6 main-symsweeper.c

```
/*
        Software for ATtiny2313 on
        FMCW Synthesizer 9.2GHz board

        by
        Ruben Undheim
        Master thesis 2012 NTNU
*/

#define F_CPU 1000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "ad9858control.h"


int main(void) {
        init();
        uint32_t fvalue = 5.2274e8; // 9.25GHz
        uint32_t fvalue2 = 5.2480e8; // 9.25GHz
#define DFTW 2000
#define tid 1000
        delay_ms(500);
        set_bit(CTRL_2GHZ,1);

        toggle_fud();
        write_long(0x03,fvalue);
        toggle_fud();
        set_bit(CTRL_SWEEP,1);
        toggle_fud();
        write_long(0x01,DFTW);
        write_short(0x02,tid);
        toggle_fud();

        while(1){
```

```
36                  toggle_fud();
37                  set_bit(CTRL_CLRFREQ,0);
38                  toggle_fud();
39                  set_bit(CTRL_CLRFREQ,1);
40                  write_long(0x03,fvalue2);
41                  write_long(0x01,-DFTW); // Turn direction
42                  delay_ms(3);
43                  toggle_fud();
44                  set_bit(CTRL_CLRFREQ,0);
45                  toggle_fud();
46                  set_bit(CTRL_CLRFREQ,1);
47                  write_long(0x03,fvalue);
48                  write_long(0x01,DFTW);
49                  delay_ms(3);
50          }
51
52
53  }
```

Appendix B

In-System Programming (ISP) of AVR

To make it possible to program the microcontroller on the board, certain considerations must be made [39]. There are 6 pins of the AVR that are used when programming:

- VCC
- RESET
- MISO
- MOSI
- SCK
- GND

The initialization of the programming is performed by pulling *RESET* low. The 3 communication pins *MISO*, *MOSI* and *SCK* are often used for other purposes than programming. Series resistors should be added between them and what they are connected to in order to avoid driver contention (see Figure B.1).

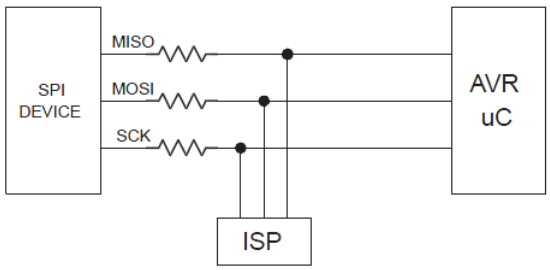


Figure B.1: *How to connect series resistor for ISP programming. [39]*

The circuit is by default not powered via the ISP, but the VCC connection is used to monitor the voltage. Power must therefore be applied externally to the circuit.

All in all, a connector with 6 pins is needed. This is then used to connect the Atmel AVR ISP mkII Programmer to the board.

Appendix C

Source Code Octave

C.1 phasenoise-doall.m

```
%
% Master thesis
% Ruben Undheim 2012
%

%phasenoise(350e6);
[f,pn132_1] = phasenoise1(121e6,2*pi*30e3,0.70);
%set(0,"Defaulttextfontsize",32)
[f,pn132_2] = phasenoise1(121e6,2*pi*30e3,0.1);
[f,pn132_3] = phasenoise1(121e6,2*pi*30e3,0.4);
[f,pn132_4] = phasenoise1(121e6,2*pi*30e3,1.0);
[f,pn132_5] = phasenoise1(121e6,2*pi*30e3,2.0);
[f,pn132_6] = phasenoise1(121e6,2*pi*30e3,5.0);
[f,pn132_7] = phasenoise1(121e6,2*pi*30e3,10);
%[f,pn132_2] = phasenoise2(132e6,2*pi*100e3,0.707,5e9);
%phasenoise(50e6);

figure(1)

set(gca,'fontsize',32)
semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
        f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
        f,pn132_5,'linewidth',5,f,pn132_6,'linewidth',5,...
        f,pn132_7,'linewidth',5);%,f,refuti,'linewidth',2,f,loopvcouti,'linewidth',2);
title('PLL Bandwidth 30kHz, PD Freq 121MHz');
xlabel('Offset frequency Hz');
ylabel('Phase noise dBc/Hz');
legend('\zeta = 0.7','0.1','0.4','1.0','2.0','5.0','10.0');

print -color -deps -F:32 ../thesis/figur/phasenoisesimul-dampfactor.eps

[f,pn132_1] = phasenoise1(121e6,2*pi*0e3,0.70);
[f,pn132_2] = phasenoise1(121e6,2*pi*5e3,0.7);
[f,pn132_3] = phasenoise1(121e6,2*pi*15e3,0.7);
[f,pn132_4] = phasenoise1(121e6,2*pi*30e3,0.7);
[f,pn132_5] = phasenoise1(121e6,2*pi*50e3,0.7);
[f,pn132_6] = phasenoise1(121e6,2*pi*80e3,0.7);
[f,pn132_7] = phasenoise1(121e6,2*pi*100e3,0.7);

figure(2)
set(gca,'fontsize',32)
semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
        f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
```

```
        f,pn132_5,'linewidth',5,f,pn132_6,'linewidth',5,...
        f,pn132_7,'linewidth',5 );%,f,refuti,'linewidth',2,f,loopvcouti,'linewidth',2);
title('PLL Damping factor constant 0.7, PD Freq 121MHz');
xlabel('Offset frequency Hz');
ylabel('Phase noise dBc/Hz');
legend('f_n = 0 kHz','5 kHz','15 kHz','30 kHz','50 kHz','80 kHz','100 kHz');

print -color -deps -F:32 ../thesis/figur/phasenoisesimul-bandwidth.eps

[f,pn132_1] = phasenoise1(121e6,2*pi*30e3,0.7);
[f,pn132_2] = phasenoise2(121e6,2*pi*30e3,0.7,2e9);
[f,pn132_3] = phasenoise2(121e6,2*pi*30e3,0.7,4e9);
[f,pn132_4] = phasenoise2(121e6,2*pi*30e3,0.7,6e9);
[f,pn132_5] = phasenoise2(121e6,2*pi*30e3,0.7,9e9);

figure(3)
set(gca,'fontsize',32)
semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
        f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
        f,pn132_5,'linewidth',5)
title('PLL Bandwidth 30kHz, PD Freq 121MHz');
xlabel('Offset frequency Hz');
ylabel('Phase noise dBc/Hz');
legend('No frequency conversion','f_{LO} = 2 GHz','4 GHz','6 GHz','9 GHz');

print -color -deps -F:32 ../thesis/figur/phasenoisesimul-mix.eps


[f,pn132_1] = phasenoise1(121e6,2*pi*30e3,0.7);
[f,pn132_2] = phasenoise3(121e6,2*pi*30e3,0.7,2e9);
[f,pn132_3] = phasenoise3(121e6,2*pi*30e3,0.7,4e9);
[f,pn132_4] = phasenoise3(121e6,2*pi*30e3,0.7,6e9);
[f,pn132_5] = phasenoise3(121e6,2*pi*30e3,0.7,9e9);

figure(4)
set(gca,'fontsize',32)
semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
        f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
        f,pn132_5,'linewidth',5)
title('PLL Bandwidth 30kHz, PD Freq 121MHz');
xlabel('Offset frequency Hz');
ylabel('Phase noise dBc/Hz');
legend('No frequency conversion','f_{LO} = 2 GHz','4 GHz','6 GHz','9 GHz');

print -color -deps -F:32 ../thesis/figur/phasenoisesimul-mix2.eps


[f,pn132_1] = phasenoise1(121e6,2*pi*30e3,0.7);
[f,pn132_2] = phasenoise4(121e6,2*pi*30e3,0.7,2e9);
[f,pn132_3] = phasenoise4(121e6,2*pi*30e3,0.7,4e9);
[f,pn132_4] = phasenoise4(121e6,2*pi*30e3,0.7,6e9);
[f,pn132_5] = phasenoise4(121e6,2*pi*30e3,0.7,9e9);

figure(5)
set(gca,'fontsize',32)
semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
        f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
        f,pn132_5,'linewidth',5)
title('PLL Bandwidth 30kHz, PD Freq 121MHz');
xlabel('Offset frequency Hz');
ylabel('Phase noise dBc/Hz');
legend('No frequency conversion','f_{LO} = 2 GHz','4 GHz','6 GHz','9 GHz');

print -color -deps -F:32 ../thesis/figur/phasenoisesimul-mix3.eps

[f,pn132_1] = phasenoise1(50e6,2*pi*30e3,0.7);
[f,pn132_2] = phasenoise1(82e6,2*pi*30e3,0.7);
[f,pn132_3] = phasenoise1(121e6,2*pi*30e3,0.7);
[f,pn132_4] = phasenoise1(178e6,2*pi*30e3,0.7);
[f,pn132_5] = phasenoise1(250e6,2*pi*30e3,0.7);
```

```
114 [f,pn132_6] = phasenoise1(320e6,2*pi*30e3,0.7);
115 [f,pn132_7] = phasenoise1(400e6,2*pi*30e3,0.7);
116
117 figure(6)
118 set(gca,'fontsize',32)
119 semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5,...
120         f,pn132_3,'linewidth',5,f,pn132_4,'linewidth',5,...
121         f,pn132_5,'linewidth',5,f,pn132_6,'linewidth',5)
122
123 title('PLL Bandwidth 30kHz, Damping factor 0.7');
124 xlabel('Offset frequency Hz');
125 ylabel('Phase noise dBc/Hz');
126 legend('f_{PD} = 50 MHz','82 MHz','121 MHz','178 MHz','250 MHz','320 MHz');
127
128 print -color -deps -F:32 ../thesis/figur/phasenoisesimul-div.eps
129
130 [f,pn132_1] = phasenoise1(121e6,2*pi*0e3,0.7);
131 [f,pn132_2] = phasenoise1(121e6,2*pi*30e3,0.7);
132
133 figure(7)
134
135 set(gca,'fontsize',32)
136 semilogx(f,pn132_1,'linewidth',5,f,pn132_2,'linewidth',5)
137 title('PLL Bandwidth 30kHz, PD Freq 121MHz');
138 xlabel('Offset frequency Hz');
139 ylabel('Phase noise dBc/Hz');
140 legend('Free running VCO','Phase-locked VCO');
141
142 print -color -deps -F:32 ../thesis/figur/phasenoisesimul-freevco.eps
143
144
145 [f,pn132_1] = phasenoise1(121e6,2*pi*19.5e3,1.05);
146 figure(8)
147 set(gca,'fontsize',32)
148 semilogx(f,pn132_1,'linewidth',5)
149 title('PLL Bandwidth 24kHz, PD Freq 121MHz');
150 xlabel('Offset frequency Hz');
151 ylabel('Phase noise dBc/Hz');
152
153 print -color -deps -F:32 calculated.eps
154 ut = [f(53:size(f,2))' pn132_1(53:size(f,2))'];
155 save -ascii calculated.num  ut
156
157 [f,pn132_1] = phasenoise1(121e6,2*pi*24.2e3,1.71);
158 figure(9)
159 set(gca,'fontsize',32)
160 semilogx(f,pn132_1,'linewidth',5)
161 title('PLL Bandwidth 24kHz, PD Freq 121MHz');
162 xlabel('Offset frequency Hz');
163 ylabel('Phase noise dBc/Hz');
164
165 print -color -deps -F:32 calculated-wrong.eps
166 ut = [f(53:size(f,2))' pn132_1(53:size(f,2))'];
167 save -ascii calculated-wrong.num  ut
168 pause
```

C.2 phasenoise1.m

```
1 %
2 % Master thesis
3 % Ruben Undheim 2011
4 %
```

```
5  function [f,toti] = phasenoise1(pdfrequency,omega,damping)
6
7  f = logspace(1.95,6,200);
8
9
10 tcxox = [10,  100,1e3,10e3,100e3,1e6,10e6 100e6];
11 tcxo = [-92,   -92,-121,-141,-147,-150, -150,  -150]; % Valpey Fisher TCXO
12 %tcxo = [-80,   -100,-102,-106,-109,-116, -118, -150]; % signalgenerator Anritsu
13
14 ddsx = [10,   1e3,10e3,100e3,  100e6];
15 dds103 = [-147,   -147,-150,-152,  -152];
16 dds403 = [-133,   -133,-137,-140,   -140];
17
18 pllx = [10,   1e3,100e3,  100e6];
19 pll = [-228,  -228,-228,  -228];
20
21 vcox = [10,  100,  1e3,10e3,100e3,1e6, 100e6];
22 vco = [-20 , -20 , -56,-92,-117,-138,    -140];
23
24 loopx = [10,   100,1e3,10e3,100e3,1e6,    100e6];
25 loopnV = [31.8  ,  31.8,3.66,1.57,1.50,1.49,  1.49];
26 loop = 20*log10(loopnV.*1e-9*120e6./(sqrt(2).*loopx));
27
28 tcxoi = interp1(10*log10(tcxox),tcxo,10*log10(f),'linear');
29 dds103i = interp1(10*log10(ddsx),dds103,10*log10(f),'linear');
30 dds403i = interp1(10*log10(ddsx),dds403,10*log10(f),'linear');
31 plli = interp1(10*log10(pllx),pll,10*log10(f),'linear');
32 vcoi = interp1(10*log10(vcox),vco,10*log10(f),'linear');
33 loopi = interp1(10*log10(loopx),loop,10*log10(f),'linear');
34
35
36 tcxo2i = tcxoi-20*log10(1000e6/pdfrequency);
37 if pdfrequency > 250e6
38         dds2i = dds403i-20*log10(403e6/pdfrequency);
39 else
40         dds2i = dds103i-20*log10(103e6/pdfrequency);
41 endif
42
43 %dds2i = (dds403i-dds103i)./(403e6-103e6).*(pdfrequency-103e6)+(dds103i);
44 refi = 10*log10(10.^(tcxo2i./10)+10.^(dds2i./10));
45 pll2i = plli + 10*log10(pdfrequency);
46
47 loopvcoi = 10*log10(10.^(vcoi./10)+10.^(loopi./10));
48 %semilogx(f,pll350i,f,pll230i,f,pll130i,f,pll50i);
49
50 %omega = 2*pi*30e3;
51 %damping = 0.907;
52 %N = 24;
53 w= 2*pi*f;
54 E = -w.^2./(-w.^2+2*j*damping*omega*w+omega.^2);
55 HdeltN = (2*damping*omega*j*w+omega.^2)./(-w.^2+2*damping*omega*j*w+omega.^2);
56 %H = N.*HdeltN;
57 Edb = 20*log10(abs(E));
58 HdeltNdb = 20*log10(abs(HdeltN));
59
60 carrier = 9.2e9;
61
62 refuti = refi + 20*log10(floor(carrier ./ pdfrequency)) + HdeltNdb;
63
64 plluti = pll2i + 20*log10(floor(carrier ./ pdfrequency)) + HdeltNdb;
65
66 loopvcouti = loopvcoi + Edb;
67
68 toti = 10*log10(10.^(refuti./10) + 10.^(plluti./10) + 10.^(loopvcouti./10));
69
70
71 end
```

C.3 phasenoise2.m

```
%
% Master thesis
% Ruben Undheim 2011
%
function [f,toti] = phasenoise2(pdfrequency,omega,damping,lofrequency)

f = logspace(1.95,6,200);


tcxox = [10,  100,1e3,10e3,100e3,1e6,10e6 100e6];
tcxo = [-92,   -92,-121,-141,-147,-150, -150,  -150]; % Valpey Fisher TCXO
%tcxo = [-100,   -100,-102,-106,-109,-116, -118, -150]; % signalgenerator Anritsu

ddsx = [10,   1e3,10e3,100e3,  100e6];
dds103 = [-147,   -147,-150,-152,  -152];
dds403 = [-133,   -133,-137,-140,   -140];

pllx = [10,   1e3,100e3,  100e6];
pll = [-228,  -228,-228,  -228];

vcox = [10, 1e2,   1e3,10e3,100e3,1e6, 100e6];
vco = [0 ,-20 ,  -56,-92,-117,-138,    -140];

loopx = [10,   100,1e3,10e3,100e3,1e6,    100e6];
loopnV = [31.8  ,  31.8,3.66,1.57,1.50,1.49,  1.49];
loop = 20*log10(loopnV.*1e-9*120e6./(sqrt(2).*loopx));

mixlox = [10,  100,1e3,10e3,100e3,1e6,10e6 100e6];
mixlo = [-92,   -92,-121,-141,-147,-150, -150,  -150];
mixlo = mixlo+20*log10(lofrequency./1e9); % Valpey Fisher TCXO
%mixlo = ones(1,8)*-110; % Valpey Fisher TCXO

tcxoi = interp1(10*log10(tcxox),tcxo,10*log10(f),'linear');
dds103i = interp1(10*log10(ddsx),dds103,10*log10(f),'linear');
dds403i = interp1(10*log10(ddsx),dds403,10*log10(f),'linear');
plli = interp1(10*log10(pllx),pll,10*log10(f),'linear');
vcoi = interp1(10*log10(vcox),vco,10*log10(f),'linear');
loopi = interp1(10*log10(loopx),loop,10*log10(f),'linear');
mixloi = interp1(10*log10(mixlox),mixlo,10*log10(f),'linear');


tcxo2i = tcxoi-20*log10(1000e6/pdfrequency);
if pdfrequency > 250e6
        dds2i = dds403i-20*log10(403e6/pdfrequency);
else
        dds2i = dds103i-20*log10(103e6/pdfrequency);
endif

%dds2i = (dds403i-dds103i)./(403e6-103e6).*(pdfrequency-103e6)+(dds103i);
refi = 10*log10(10.^(tcxo2i./10)+10.^(dds2i./10));
pll2i = plli + 10*log10(pdfrequency);

loopvcoi = 10*log10(10.^(vcoi./10)+10.^(loopi./10));
%semilogx(f,pll350i,f,pll230i,f,pll130i,f,pll50i);

%omega = 2*pi*30e3;
%damping = 0.907;
%N = 24;
w= 2*pi*f;
E = -w.^2./(-w.^2+2*j*damping*omega*w+omega.^2);
HdeltN = (2*damping*omega*j*w+omega.^2)./(-w.^2+2*damping*omega*j*w+omega.^2);
%H = N.*HdeltN;
Edb = 20*log10(abs(E));
HdeltNdb = 20*log10(abs(HdeltN));

carrier = 9.2e9;
```

```
67
68 refuti = refi + 20*log10(floor((carrier-lofrequency) ./ pdfrequency)) + HdeltNdb;
69
70 plluti = pll2i + 20*log10(floor((carrier-lofrequency) ./ pdfrequency)) + HdeltNdb;
71
72 loopvcouti = loopvcoi + Edb;
73
74 mixlouti = mixloi + HdeltNdb;
75
76 toti = 10*log10(10.^(refuti./10) + 10.^(plluti./10) + 10.^(loopvcouti./10) + 10.^(mixlouti./10));
77
78
79 end
```

C.4 sweep.m

```
1  %
2  % Master thesis
3  % Ruben Undheim 2011
4  %
5  function sweep(meter)
6  s = ([100 103 120 160 180 190 210 215 217 215 210 205 200 198 200 200 200 ...
7  200 200 200 200 200 200 200 200 200 200 200 200 200]-100)/100;
8  s= horzcat(s,s+1,s+2,s+3,s+4,s+5,s+6,s+7,s+8,s+9);
9  s= horzcat(s,s+10,s+20,s+30,s+40);
10
11 for i=1:size(s,2)
12         for j=1:100
13                 sentinterp2((i-1)*100+j) = s(i);
14         end
15 end
16 samplerate = 33.33e6;
17 delay = meter/(3.0e8/2.0);
18 frekshift = 1953.1e6*delay
19 Ndelay = floor(delay*samplerate)
20 sentinterp = horzcat(sentinterp2,ones(1,Ndelay)*50);
21 backinterp = horzcat(zeros(1,Ndelay)*0,sentinterp2);
22
23
24 n=[1:size(sentinterp,2)];
25 faseteller = 0;
26 signalsent = zeros(1,size(sentinterp,2));
27 for i=1:size(sentinterp,2)
28         signalsent(i) = complex(cos(faseteller),sin(faseteller));
29         faseteller = faseteller + 2*pi*((sentinterp(i)*0.2e6)+2e6)./samplerate;
30 end
31
32 faseteller = 0;
33 for i=1:size(sentinterp,2)
34         signalback(i) = cos(faseteller);
35         faseteller = faseteller + 2*pi*((backinterp(i)*0.2e6)+2e6)./samplerate;
36 end
37 b = [-0.00014192343223839998, -0.00036639833706431091, -0.00049512332770973444,...
38  -0.00048182642785832286, -0.00030939906719140708, -1.7783308976504486e-06, 0.00037053102278150618,...
39  0.00069792440626770258, 0.00085626530926674604, 0.00074630836024880409, 0.00033762818202376366,...
40  -0.00029959198592543602, -0.00099160766694694757, -0.001498736790381372,  -0.0015877059195190668,...
41  -0.0011220205342397094, -0.00013765363837592304, 0.001128686941228807, 0.0022810851223766804,...
42  0.0028757203835994005, 0.0025748340412974358, 0.0012941312743350863, -0.00071316101821139455,...
43  -0.0028796552214771509, -0.0044669308699667454, -0.0047986195422708988, -0.0035150840412825346,...
44  -0.00076127308420836926, 0.0027718171147731781, 0.0059924973174929619, 0.0077209821902215481,...
45  0.0070871044881641865, 0.0038878121413290501, -0.0012267252895981073, -0.0068306461907923222,...
46  -0.011079289019107819, -0.012268241494894028, -0.0094373384490609169, -0.0028241074178367853,...
```

```
 0.0060075060464441776, 0.014455742202699184, 0.01955750398337841, 0.018895231187343597,...
 0.011483190581202507, -0.0016435199650004506, -0.01735694520175457, -0.030970720574259758,...
 -0.037300385534763336, -0.032007057219743729, -0.01287740096449852, 0.019319223240017891,...
 0.060665227472782135, 0.10471757501363754, 0.14380654692649841, 0.17071092128753662,...
 0.1802975982427597, 0.17071092128753662, 0.14380654692649841, 0.10471757501363754,...
 0.060665227472782135, 0.019319223240017891, -0.01287740096449852, -0.032007057219743729,...
 -0.037300385534763336, -0.030970720574259758, -0.01735694520175457, -0.0016435199650004506,...
 0.011483190581202507, 0.018895231187343597, 0.01955750398337841, 0.014455742202699184,...
 0.0060075060464441776, -0.0028241074178367853, -0.0094373384490609169, -0.012268241494894028,...
 -0.011079289019107819, -0.0068306461907923222, -0.0012267252895981073, 0.0038878121413290501,...
 0.0070871044881641865, 0.0077209821902215481, 0.0059924973174929619, 0.002771817147731781,...
 -0.0007612730842083696, -0.0035150840412825346, -0.0047986195422708988, -0.0044669308699667454,...
 -0.0028796552214771509, -0.0007131610182113945 5, 0.0012941312743350863, 0.0025748340412974358,...
 0.0028757203835994005, 0.0022810851223766804, 0.0011286869412288 07, -0.0001376536383759230 4,...
 -0.0011220205342397094, -0.0015877059195190668, -0.0014987367903813 72, -0.0009916076669469475 7,...
 -0.0002995198592543602, 0.0003376281820237636 6, 0.0007463083602488 0409, 0.0008562653092667460 4,...
 0.0006979244062677025 8, 0.0003705310227815061 8, -1.7783308976504486e-06, -0.0003039906719140708,...
 -0.0004818264278583228 6, -0.0004951233277097344 4, -0.0003663983370643109 1, -0.0001419234322383999 8 ];


conversion = signalsent(Ndelay:(size(signalsent,2)-Ndelay)).*...
     signalback(Ndelay:(size(signalback,2)-Ndelay));
y = filter(b,[1],conversion);
figure(1)
subplot(2,1,1)
set(gca,'FontSize',32)
plot(0:1/33.33e6:19999/33.33e6,sentinterp(1:20000)*0.2e6,...
        0:1/33.33e6:19999/33.33e6,backinterp(1:20000)*0.2e6)

axis([0 0.6e-3 0e6 1.4e6])
xlabel('time (s)');
ylabel('tranmit frequency (Hz)');

c = hamming(size(y,2))'.*y;

C = fft(c);
subplot(2,1,2)
set(gca,'FontSize',32)
plot(-199*222.21:222.21:199*222.21,horzcat(abs(C(size(C,2)-198:size(C,2))),...
        abs(C(1:200))),'linewidth',5)
xlabel('frequency offset (Hz)');
ylabel('amplitude');
end
```

C.5 tracking-step.m

```
%
% Master thesis
% Ruben Undheim 2012
%

wstep = 2*pi;%100e6/1024 * 2*pi / 32 /4
wn = 14e3*2*pi;
zeta = 0.7;

t = [0:0.0000001:0.0004];
perror = wstep./wn .* (1./(sqrt(1-zeta.^2)).*sin(sqrt(1-zeta.^2).*wn.*t)) .* exp(-zeta.*wn.*t);
ferror = wstep-(diff(perror)./diff(t));
perror3 = wstep./wn .* (1./(sqrt(zeta.^2-1)).*sinh(sqrt(zeta.^2-1).*wn.*t)) .* exp(-zeta.*wn.*t);
ferror3 = wstep-(diff(perror3)./diff(t));
ferror4 = wstep-(wstep.*cosh(sqrt(zeta.^2-1).*wn.*t).*exp(-zeta.*wn.*t)-...
        wstep.*sinh(sqrt(zeta.^2-1).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(zeta.^2-1)));
```

```
17  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
18          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
19
20  %figure(1)
21  %set(gca,'fontsize',32)
22  %plot(t,perror,'LineWidth',5);
23  %axis([0 0.0001 -0.05 0.4]);
24  %xlabel('t');
25  %print -deps -color -F:32 grafer/fasefeil14khz.eps
26  figure(1)
27  set(gca,'fontsize',32)
28  plot(t(1:size(t,2)-1),ferror/(2*pi),'LineWidth',5);
29  ylabel('f');
30  xlabel('t');
31  axis([0 0.0003 0 1.4]);
32  print -deps -color -F:32 grafer/frekvensfeil14khz.eps
33
34
35  wn = 14e3*2*pi;
36  zeta = 0.7;
37  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
38          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
39
40  figure(2)
41  subplot(2,1,1)
42  set(gca,'fontsize',32)
43  plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
44  ylabel('f');
45  xlabel('t');
46  axis([0 0.0003 0 1.7]);
47
48  wn = 14e3*2*pi;
49  zeta = 0.3;
50  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
51          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
52
53  subplot(2,1,2)
54  set(gca,'fontsize',32)
55  plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
56  ylabel('f');
57  xlabel('t');
58  axis([0 0.0003 0 1.7]);
59
60
61  print -deps -color -F:32 grafer/frekvensfeil14khz.eps
62
63  wn = 4e3*2*pi;
64  zeta = 0.7;
65  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
66          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
67
68  figure(4)
69  subplot(2,1,1)
70  set(gca,'fontsize',32)
71  plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
72  ylabel('f');
73  xlabel('t');
74  axis([0 0.0003 0 1.7]);
75
76  wn = 4e3*2*pi;
77  zeta = 0.3;
78  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
79          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
80
81  subplot(2,1,2)
82  set(gca,'fontsize',32)
83  plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
84  ylabel('f');
85  xlabel('t');
86  axis([0 0.0003 0 1.7]);
```

```
87  print -deps -color -F:32 grafer/frekvensfeil4khz.eps
88
89
90
91  wstep = 2*pi;%100e6/1024 * 2*pi / 32 /4
92  wn = 4e3*2*pi;
93  zeta = 0.7;
94
95  t = [0:0.0000001:0.0004];
96  perror = wstep./wn .* (1./(sqrt(1-zeta.^2)).*sin(sqrt(1-zeta.^2).*wn.*t)) .* exp(-zeta.*wn.*t);
97  ferror = wstep-(diff(perror)./diff(t));
98  ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
99          wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
100
101
102 figure(3)
103 set(gca,'fontsize',32)
104 plot(t,perror,'LineWidth',5);
105 axis([0 0.0001 -0.05 0.4]);
106 xlabel('t');
107 print -deps -color -F:32 grafer/fasefeil4khz.eps
108
109
110 figure(9)
111 wn = 14e3*2*pi;
112 zeta = 0.5;
113 ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
114         wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
115
116 set(gca,'fontsize',32)
117 plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
118 ylabel('f');
119 xlabel('t');
120 axis([0 0.0003 0 1.4]);
121 print -deps -color -F:32 grafer/frekvensfeil14khzstor.eps
122
123
124
125
126
127
128 wstep = 2*pi;%100e6/1024 * 2*pi / 32 /4
129 wn = 30e3*2*pi;
130 zeta = 0.5;
131
132 t = [0:0.0000001:0.0004];
133 perror = wstep./wn .* (1./(sqrt(1-zeta.^2)).*sin(sqrt(1-zeta.^2).*wn.*t)) .* exp(-zeta.*wn.*t);
134 ferror = wstep-(diff(perror)./diff(t));
135 ferror2 = wstep-(wstep.*cos(sqrt(1-zeta.^2).*wn.*t).*exp(-zeta.*wn.*t)-...
136         wstep.*sin(sqrt(1-zeta.^2).*wn.*t).*zeta.*exp(-zeta.*wn.*t)./(sqrt(1-zeta.^2)));
137
138 figure(5)
139 set(gca,'fontsize',32)
140 plot(t,perror,'LineWidth',5);
141 axis([0 0.0001 -0.05 0.4]);
142 xlabel('t');
143 print -deps -color -F:32 grafer/fasefeil4khz.eps
144 figure(6)
145 set(gca,'fontsize',32)
146 plot(t(1:size(t,2)-1),ferror/(2*pi),'LineWidth',5);
147 ylabel('f');
148 xlabel('t');
149 axis([0 0.0003 0 1.4]);
150 print -deps -color -F:32 grafer/frekvensfeil30khz.eps
151 figure(7)
152 set(gca,'fontsize',32)
153 plot(t(1:size(t,2)),ferror2/(2*pi),'LineWidth',5);
154 axis([0 0.0003 0 1.4]);
155
156
```

```
157
158
159
160 pause
```

C.6 cancellation.m

```
1  f = logspace(2,8,1000);
2  c = 3e8;
3
4  vcox = [10,  100,  1e3,10e3,100e3,1e6, 100e6];
5  vco = [-20 , -20 , -56,-92,-117,-138,  -180];
6
7  vcoi = interp1(10*log10(vcox),vco,10*log10(f),'linear');
8
9  L = 10.^(vcoi./10);%0.01./f.^2;%+100000000./f.^3;
10
11 l = 0.100;
12 Lut1 = 4.*L.*(sin(pi.*l.*f./c)).^2;
13 l = 0.500;
14 Lut2 = 4.*L.*(sin(pi.*l.*f./c)).^2;
15 l = 20;
16 Lut3 = 4.*L.*(sin(pi.*l.*f./c)).^2;
17 l = 200;
18 Lut4 = 4.*L.*(sin(pi.*l.*f./c)).^2;
19 l = 1000;
20 Lut5 = 4.*L.*(sin(pi.*l.*f./c)).^2;
21
22 set(gca,'fontsize',32)
23
24 semilogx(f,10*log10(abs(L)),'linewidth',5,f,10*log10(abs(Lut1)),'linewidth',5,...
25         f,10*log10(abs(Lut2)),'linewidth',5,f,10*log10(abs(Lut3)),'linewidth',5,...
26         f,10*log10(abs(Lut4)),'linewidth',5,f,10*log10(abs(Lut5)),'linewidth',5);
27 axis([100 1e8 -200 -18]);
28 legend('Original VCO phase noise','Path difference 0.1 m','0.5 m','20 m','200 m','1000 m');
29 xlabel('Offset frequency Hz');
30 ylabel('dBc/Hz');
31 print -deps -color -F:32 grafer/cancellation.eps
32
33
34
35
36 figure(2)
37 f = logspace(3,6,1000);
38 load ddstcxofelles.num
39
40 vcoi = interp1(10*log10(vcox),vco,10*log10(f),'linear');
41 Lvco = 10.^(vcoi./10);
42
43 measured = interp1(10*log10(ddstcxofelles(:,1)'),ddstcxofelles(:,2)',10*log10(f),'linear');
44 L = 10.^(measured./10);%0.01./f.^2;%+100000000./f.^3;
45 semilogx(f,measured);%10*log10(abs(L)))
46 figure(3)
47
48 l = 0.100;
49 Lut1 = 4.*L.*(sin(pi.*l.*f./c)).^2;
50 l = 0.500;
51 Lut2 = 4.*L.*(sin(pi.*l.*f./c)).^2;
52 l = 20;
53 Lut3 = 4.*L.*(sin(pi.*l.*f./c)).^2;
54 l = 200;
55 Lut4 = 4.*L.*(sin(pi.*l.*f./c)).^2;
```

```
56 l = 1000;
57 Lut5 = 4.*L.*(sin(pi.*l.*f./c)).^2;
58
59 set(gca,'fontsize',32)
60
61 semilogx(f,10*log10(abs(L)),'linewidth',5,f,10*log10(abs(Lut1)),'linewidth',5,...
62         f,10*log10(abs(Lut2)),'linewidth',5,f,10*log10(abs(Lut3)),'linewidth',5,...
63         f,10*log10(abs(Lut4)),'linewidth',5,f,10*log10(abs(Lut5)),'linewidth',5,...
64         f,10*log10(abs(Lvco)),'linestyle','--','linewidth',5);
65 axis([1000 1e6 -205 -60]);
66 legend('Original synthesizer phase noise','Path difference 0.1 m','0.5 m','20 m','200 m','1000 m','VCO');
67 xlabel('Offset frequency Hz');
68 ylabel('dBc/Hz');
69 print -deps -color -F:32 grafer/cancellation2.eps
```

Appendix D

Direct Digital Synthesis (DDS)

A direct digital synthesizer is a combined digital and analogue circuit. It is basiscally a numerical oscillator connected to a D/A converter. It contains a phase counter register that is incremented according to a frequency control register value. This phase register is used as the input in a sine lookup table. Thus, the phase is transformed into a sinusoid. The sinusoid is then given to the D/A converter which makes the sinusoid analogue. All in all, it is just a register that chooses which frequency that appears at the output.

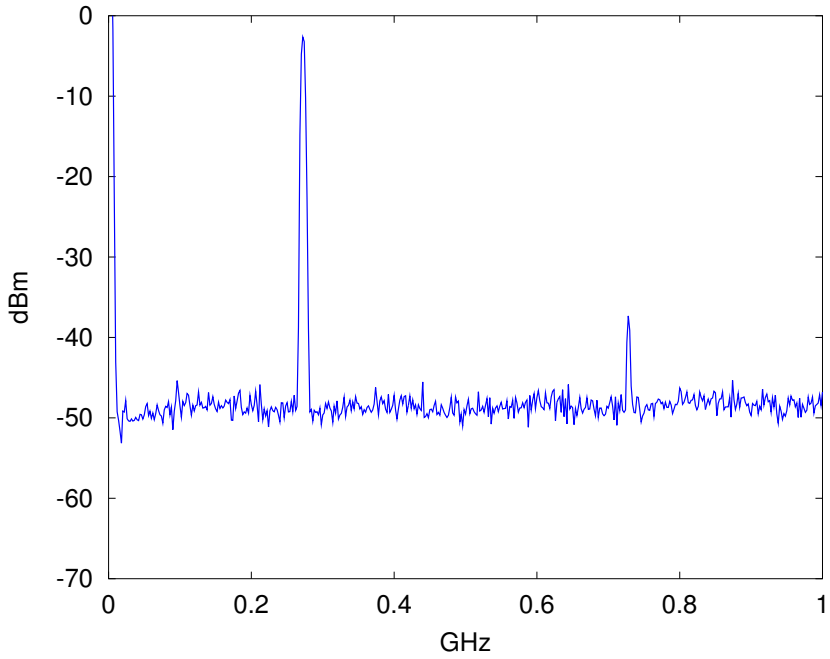


Figure D.1: *Output of DDS with images*

Since the DDS is a sampled system, the sinusoid will not be clean. It will also have a strong Nyquist image at $F_{\text{clk}} - F_{\text{out}}$. In addition, there will be components (spurs)

at all harmonics of both f_{clk} and f_{out}. Hence the spurs are found at

$$N \cdot f_{\text{clk}} \pm M \cdot f_{\text{out}} \tag{D.1}$$

Because of this, it is important to have a low-pass filter at the output. In some cases, it may be useful to have a sharp bandpass filter in order to isolate higher harmonics of the output.

Figure D.1 and D.2 show the measured spectrum from the output of the DDS from 0 to 1 GHz with the low-pass filter removed. The reference clock is at 1 GHz. The output at 250 MHz causes a Nyquist frequency at 1 GHz - 250 MHz = 750 MHz. Its magnitude is quite low, which partly is caused by the transformer at the output which in itself will have a lowpass effect. In Figure D.2, the output is at 410 MHz. The image is in this case at 580 MHz and it is much stronger. A smaller peak of -40 dBm can also be observed at the the difference between the two main peaks. It is at 170 MHz. The harmonics will we shaped by a sin(x)/x rolloff response because of the sampled nature [40]. The first zero will be at the clock frequency. This is also one of the reasons why the image output power is lower in Figure D.1 than in D.2.

There will also be noise appearing from the quantiziation error of the D/A converter, and the jitter in the digital circuit. The DDS will normally have a very good phase noise performance - much depending on the quality of the reference oscillator, but all the spurs and images may still make it a bit hard to deal with.

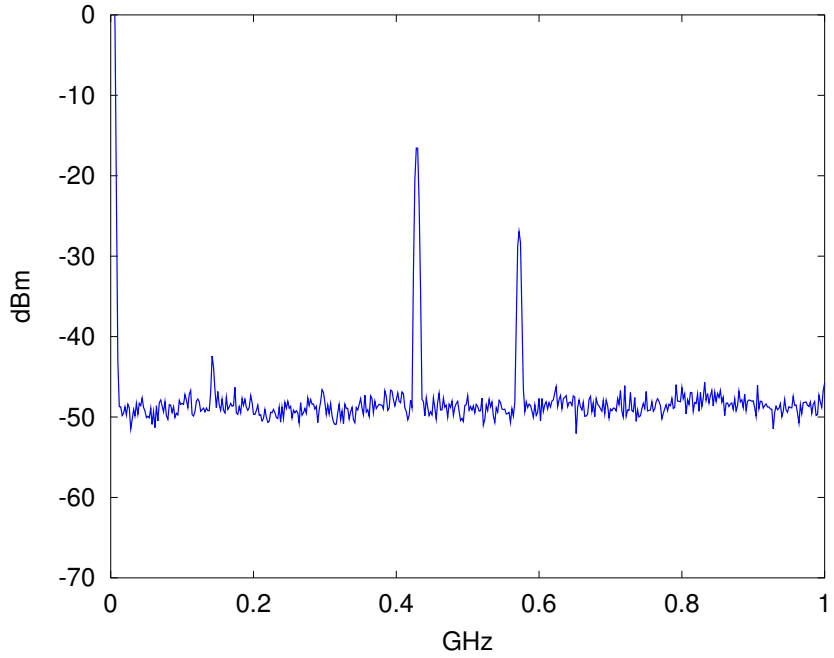


Figure D.2: *Output of DDS with images*

Appendix E

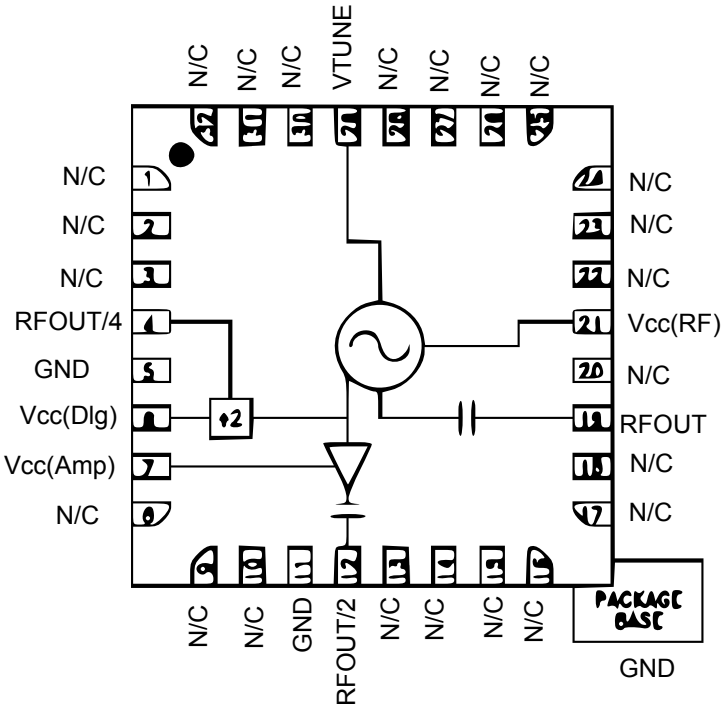
Datasheets

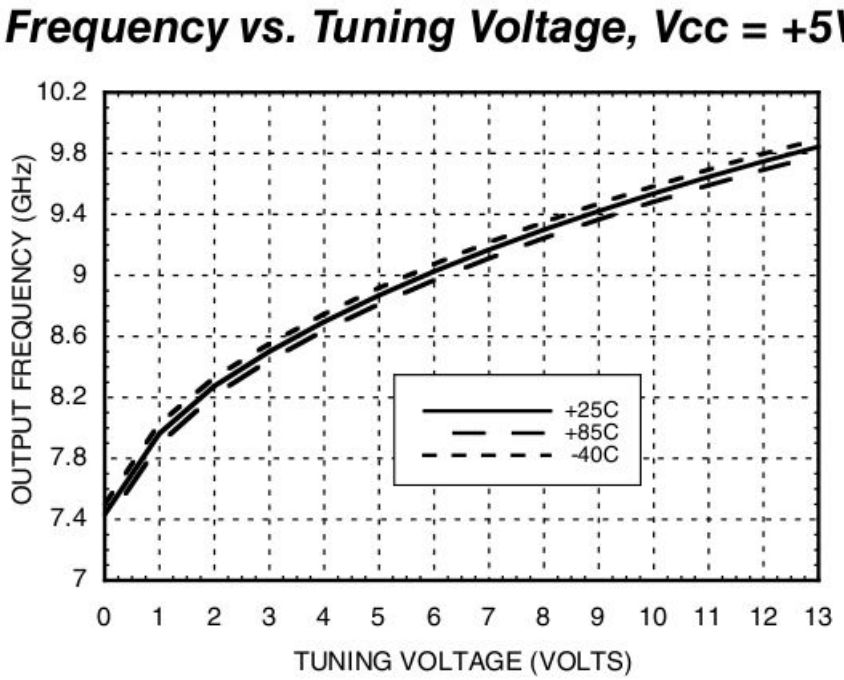
E.1 HMC510LP5

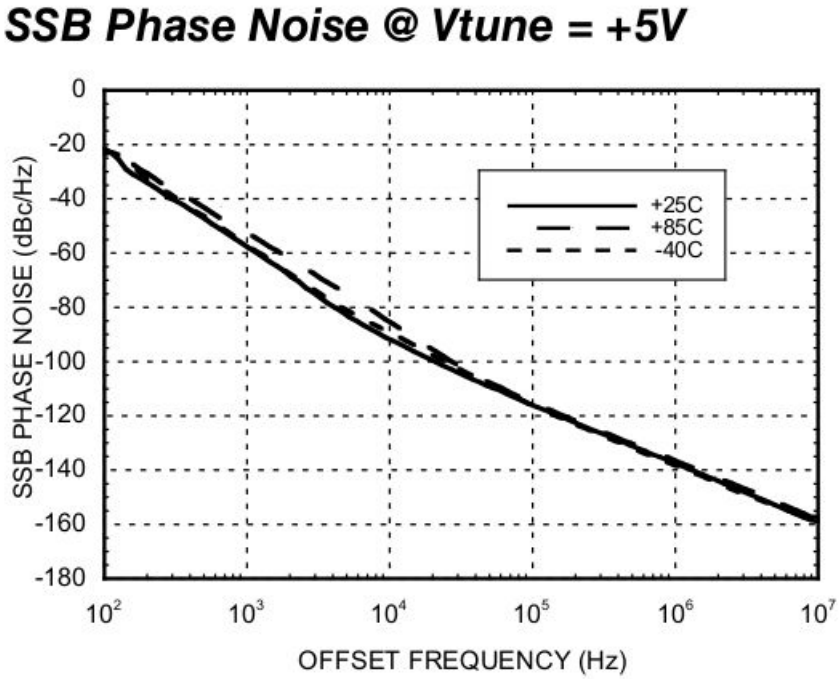
[27] Hittite MICROWAVE CORPORATION

Electrical Specifications, T_A = +25° C, Vcc (Dig), Vcc (Amp), Vcc (RF) = +5V

Parameter		Min.	Typ.	Max.	Units
Frequency Range	Fo Fo/2		8.45 - 9.55 4.225 - 4.775		GHz GHz
Power Output	RFOUT RFOUT/2 RFOUT/4	+10 +8 -8		+15 +14 -4	dBm dBm dBm
SSB Phase Noise @ 100 kHz Offset, Vtune= +5V @ RFOUT			-116		dBc/Hz
Tune Voltage	Vtune	2		13	V
Supply Current	Icc(Dig) + Icc(Amp) + Icc(RF)	270	315	360	mA
Tune Port Leakage Current (Vtune= 13V)				10	µA
Output Return Loss			2		dB
Harmonics/Subharmonics	1/2 2nd 3rd		40 15 40		dBc dBc dBc
Pulling (into a 2.0:1 VSWR)			6		MHz pp
Pushing @ Vtune= 5V			20		MHz/V
Frequency Drift Rate			0.8		MHz/°C


N/C
N/C
N/C
VTUNE
N/C
N/C
N/C
N/C
N/C
N/C
N/C
RFOUT/4
GND
Vcc(Dlg)
Vcc(Amp)
N/C
N/C
N/C
Vcc(RF)
N/C
RFOUT
N/C
N/C
÷2
N/C
N/C
GND
RFOUT/2
N/C
N/C
N/C
N/C
PACKAGE BASE
GND


Frequency vs. Tuning Voltage, Vcc = +5V
OUTPUT FREQUENCY (GHz)
10.2
9.8
9.4
9
8.6
8.2
7.8
7.4
7
0
1
2
3
4
5
6
7
8
9
10
11
12
13
TUNING VOLTAGE (VOLTS)
+25C
+85C
-40C



E.2 HMC698LP5

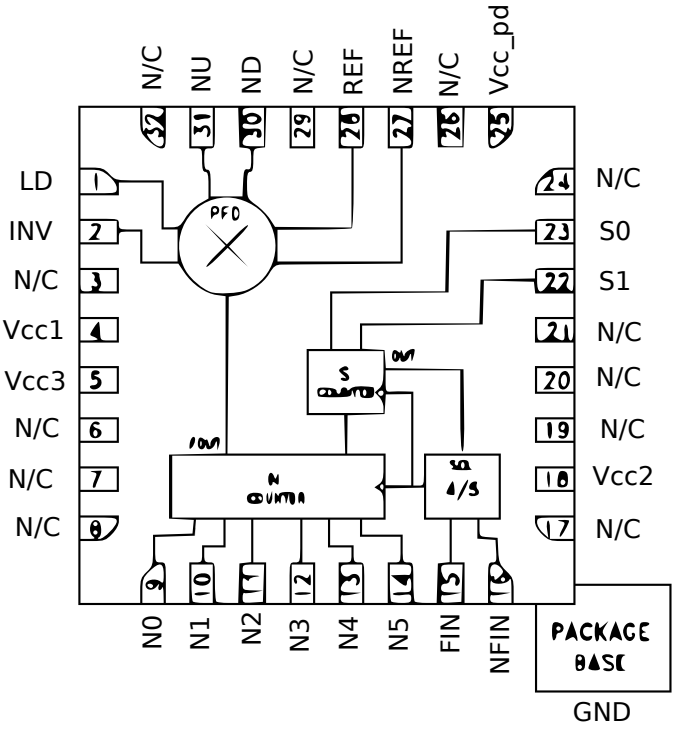
[34] Hittite MICROWAVE CORPORATION

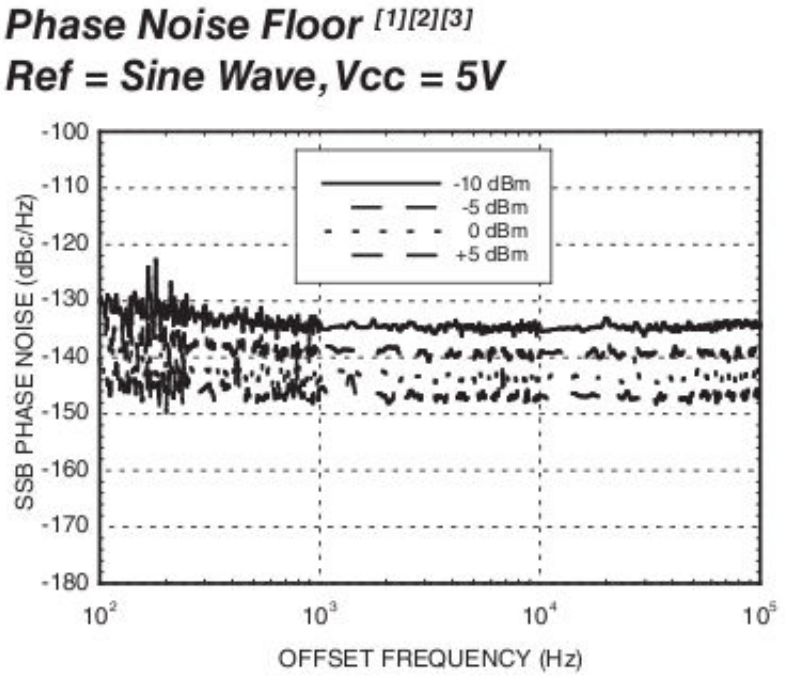
Electrical Specifications, T_A = +25° C, Vcc = Vcc1 = Vcc2 = Vcc3 = Vcc_pd = 5V

Parameter	Conditions	Min.	Typ.	Max.	Units
Maximum Ref. Input Frequency	Sine or Square Wave Input [1]	1300			MHz
Minimum Ref. Input Frequency	Square Wave Input [2]			10	MHz
Reference Input Power Range	100 MHz Frequency	-5		+5	dBm
Maximum VCO Input Frequency		7000			MHz
Minimum VCO Input Frequency				80	MHz
VCO Input Power Range	100 MHz Input Frequency	-10		+5	dBm
PFD Output Voltage			2000		mV, Pk - Pk
PFD Gain	Gain = Vpp / 2π Rad.		0.32		V/Rad.
SSB Phase Noise	@ 10 kHz Offset @ 100 MHz Square Wave Ref. Input Pin= 0 dBm		-153		dBc/Hz
Total Supply Current			310		mA

[1] Maximum frequencies may be limited by available counter division ratio.

[2] Square wave input achieves best phase noise at lower ref. frequency (see sine & square wave comparison plots)


N/C
NU
ND
N/C
REF
NREF
N/C
Vcc_pd
32
31
30
29
28
27
26
25
LD
INV
N/C
Vcc1
Vcc3
N/C
N/C
N/C
1
2
3
4
5
6
7
8
PFD
24
23
22
21
20
19
18
17
N/C
S0
S1
N/C
N/C
N/C
Vcc2
N/C
9
10
11
12
13
14
15
16
N0
N1
N2
N3
N4
N5
FIN
NFIN
PACKAGE BASE
GND


Phase Noise Floor [1][2][3]
Ref = Sine Wave, Vcc = 5V
-10 dBm
-5 dBm
0 dBm
+5 dBm
SSB PHASE NOISE (dBc/Hz)
-100
-110
-120
-130
-140
-150
-160
-170
-180
10^2
10^3
10^4
10^5
OFFSET FREQUENCY (Hz)

HMC698LP5(E) Programming Truth Table

Division Ratio n	N Counter	N Counter Decimal Set	Swallow S Counter	Swallow S Decimal Set	(LSB) N0	N1	N2	N3	N4	N5	(LSB) S0	S1
12	3	2	0	0	0	1	0	0	0	0	0	0
13	3	2	1	1	0	1	0	0	0	0	1	0
14	3	2	2	2	0	1	0	0	0	0	0	1
15	3	2	3	3	0	1	0	0	0	0	1	1
16	4	3	0	0	1	1	0	0	0	0	0	0
17	4	3	1	1	1	1	0	0	0	0	1	0
18	4	3	2	2	1	1	0	0	0	0	0	1
19	4	3	3	3	1	1	0	0	0	0	1	1
20	5	4	0	0	0	0	1	0	0	0	0	0
21	5	4	1	1	0	0	1	0	0	0	1	0
22	5	4	2	2	0	0	1	0	0	0	0	1
23	5	4	3	3	0	0	1	0	0	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
252	63	62	0	0	0	1	1	1	1	1	0	0
253	63	62	1	1	0	1	1	1	1	1	1	0
254	63	62	2	2	0	1	1	1	1	1	0	1
255	63	62	3	3	0	1	1	1	1	1	1	1
256	64	63	0	0	1	1	1	1	1	1	0	0
257	64	63	1	1	1	1	1	1	1	1	1	0
258	64	63	2	2	1	1	1	1	1	1	0	1
259	64	63	3	3	1	1	1	1	1	1	1	1

N = INT (n/P)
S = MOD (n/P)
Where: n = Desired division ratio
P = Prescaler value = 4
N = Counter N value (counter decimal set is N - 1)

E.3 AD9858

[29] ANALOG DEVICES

ELECTRICAL SPECIFICATIONS

Unless otherwise noted, V_{DD} = 3.3 V ± 5%, CPV_{DD} = 5 V ± 5%, R_{SET} = 2 kΩ, CPISET = 2.4 kΩ, reference clock frequency = 1 GHz.

Table 1.

Parameter	Temp	Test Level	Min	Typ	Max	Unit
REF CLOCK INPUT CHARACTERISTICS[1]						
Reference Clock Frequency Range (Divider Off)	Full	VI	10		1000	MHz
Reference Clock Frequency Range (Divider On)	Full	VI	20		2000	MHz
Duty Cycle at 1 GHz	25°C	V	42	50	58	%
Input Capacitance	25°C	V		3		pF
Input Impedance	25°C	IV		1500		Ω
Input Sensitivity	Full	VI	–20		+5	dBm
DAC OUTPUT CHARACTERISTICS						
Resolution	Full			10		Bits
Full-Scale Output Current	Full		5	20	40	mA
Gain Error	Full	VI	–10		+10	% FS
Output Offset	Full	VI			15	µA
Differential Nonlinearity	Full	VI		0.5	1	LSB
Integral Nonlinearity	Full	VI		1	1.5	LSB
Output Impedance	Full	VI		100		kΩ
Voltage Compliance Range	Full	VI	AV_{DD} – 1.5		AV_{DD} + 0.5	V
OUTPUT PHASE NOISE CHARACTERISTICS (AT 103 MHz I_{OUT})						
At 1 kHz Offset	Full	V		–147		dBc/Hz
At 10 kHz Offset	Full	V		–150		dBc/Hz
At 100 kHz Offset	Full	V		–152		dBc/Hz
OUTPUT PHASE NOISE CHARACTERISTICS (AT 403 MHz I_{OUT})						
At 1 kHz Offset	Full	V		–133		dBc/Hz
At 10 kHz Offset	Full	V		–137		dBc/Hz
At 100 kHz Offset	Full	V		–140		dBc/Hz

FUNCTIONAL BLOCK DIAGRAM

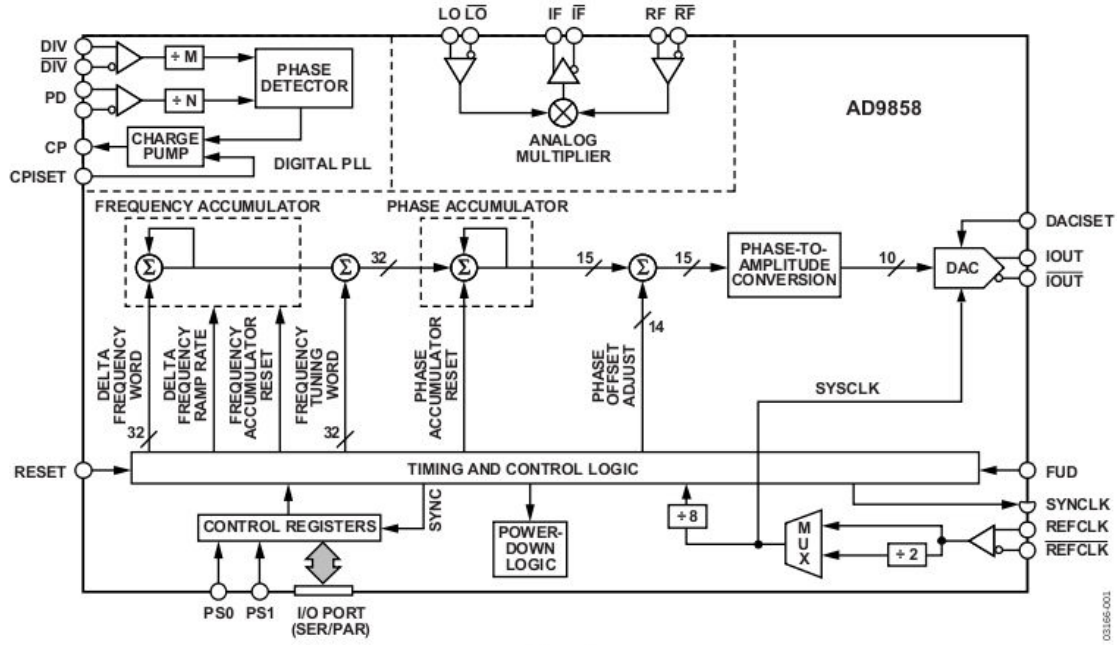
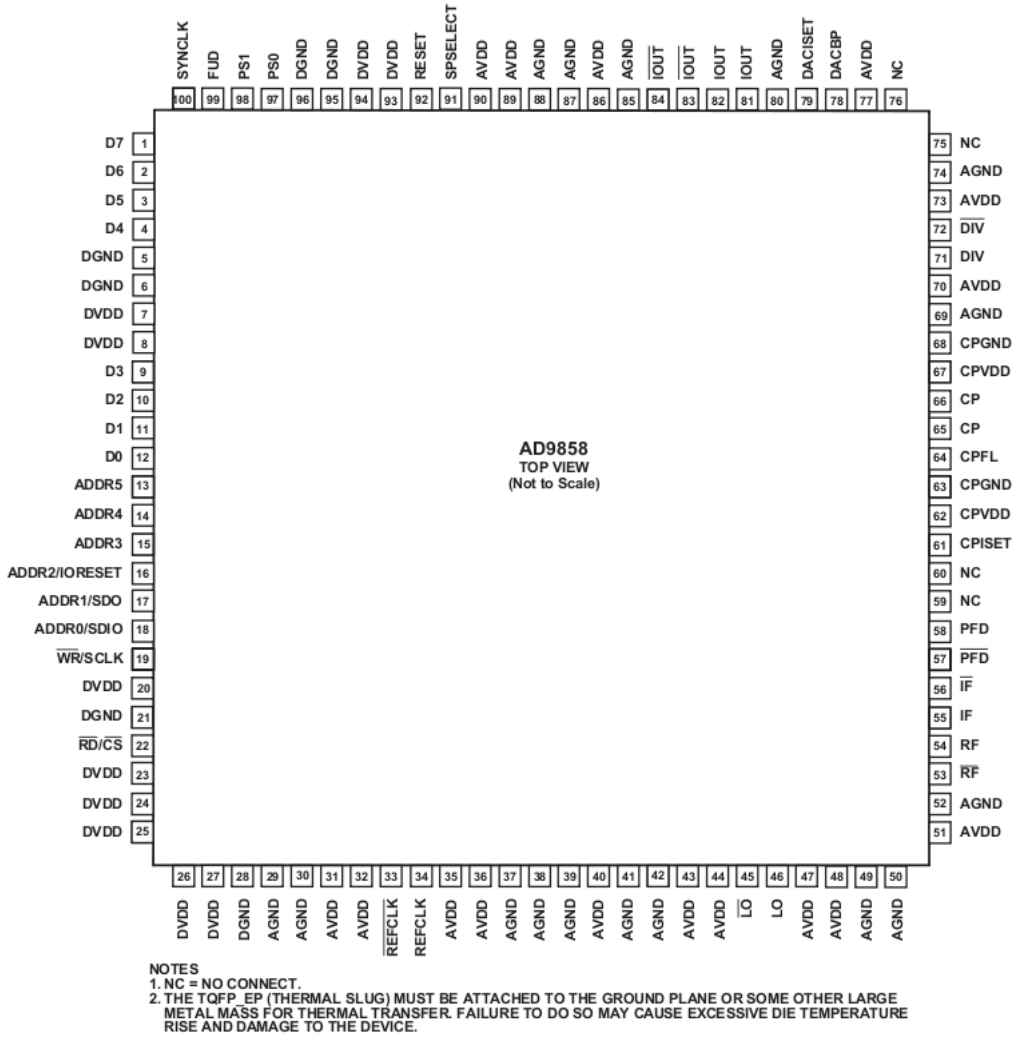


Figure 1.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS



NOTES
1. NC = NO CONNECT.
2. THE TQFP_EP (THERMAL SLUG) MUST BE ATTACHED TO THE GROUND PLANE OR SOME OTHER LARGE METAL MASS FOR THERMAL TRANSFER. FAILURE TO DO SO MAY CAUSE EXCESSIVE DIE TEMPERATURE RISE AND DAMAGE TO THE DEVICE.

Figure 2. Pin Configuration

REGISTER MAP

The registers are listed in Table 6. The serial address and parallel address numbers associated with each of the registers are shown in hexadecimal format. Square brackets [] are used to reference specific bits or ranges of bits. For example, [3] designates Bit 3, and [7:3] designates the range of bits from 7 down to 3, inclusive.

Table 6.

Register	Address		(MSB)							(LSB)	Default	
Name	Ser	Par	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value	Profile
Control function register (CFR)	0x00	0x00 [7:0]	Not used	2 GHz divider disable	SYNCLK disable	Mixer power-down	Phase detect power-down	Power-down	SDIO input only	LSB first	0x18	N/A
		0x01 [15:8]	Freq. sweep enable	Enable sine output	Charge pump offset	Phase detector divider ratio (N) (see Table 10)		Charge pump polarity	Phase detector divider ratio (M) (see Table 11)		0x00	N/A
		0x02 [23:16]	Auto Clr freq. accum	Auto Clr phase accum	Load delta freq timer	Clear freq accum	Clear phase accum	Not used	Fast lock enable	FTW for fast lock	0x00	N/A
		0x03 [31:24]	Frequency detect mode charge pump current (see Table 7)		Final closed-loop mode charge pump current (see Table 8)			Wide closed-loop mode charge pump current (see Table 9)			0x00	N/A
Delta freq. tuning word (DFTW)	0x01	0x04	Delta Frequency Word[7:0]									N/A
		0x05	Delta Frequency Word[15:8]									N/A
		0x06	Delta Frequency Word[23:16]									N/A
		0x07	Delta Frequency Word[31:24]									N/A
Delta frequency ramp rate (DFRRW)	0x02	0x08	Delta Frequency Ramp Rate Word[7:0]									N/A
		0x09	Delta Frequency Ramp Rate Word[15:8]									N/A
Frequency Tuning Word 0 (FTW0)	0x03	0x0A	Frequency Tuning Word 0[7:0]								0x00	0
		0x0B	Frequency Tuning Word 0[15:8]								0x00	0
		0x0C	Frequency Tuning Word 0[23:16]								0x00	0
		0x0D	Frequency Tuning Word 0[31:24]								0x00	0
Phase Offset Word 0 (POW0)	0x04	0x0E	Phase Offset Word 0[7:0]								0x00	0
		0x0F	Not used		Phase Offset Word 0[13:8]						0x00	0
Frequency Tuning Word 1 (FTW1)	0x05	0x10	Frequency Tuning Word 1[7:0]									1
		0x11	Frequency Tuning Word 1[15:8]									1
		0x12	Frequency Tuning Word 1[23:16]									1
		0x13	Frequency Tuning Word 1[31:24]									1
Phase Offset Word 1 (POW1)	0x06	0x14	Phase Offset Word 1[7:0]									1
		0x15	Not used		Phase Offset Word 1[13:8]							1
Frequency Tuning Word 2 (FTW2)	0x07	0x16	Frequency Tuning Word 2[7:0]									2
		0x17	Frequency Tuning Word 2[15:8]									2
		0x18	Frequency Tuning Word 2[23:16]									2
		0x19	Frequency Tuning Word 2[31:24]									2

Register	Address		(MSB)							(LSB)	Default	
Name	Ser	Par	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value	Profile
Phase Offset Word 2 (POW2)	0x08	0x1A	Phase Offset Word 2[7:0]									2
		0x1B	Not used		Phase Offset Word 2[13:8]							2
Frequency Tuning Word 3 (FTW3)	0x09	0x1C	Frequency Tuning Word 3[7:0]									3
		0x1D	Frequency Tuning Word 3[15:8]									3
		0x1E	Frequency Tuning Word 3[23:16]									3
		0x1F	Frequency Tuning Word 3[31:24]									3
Phase Offset Word 3 (POW3)	0x0A	0x20	Phase Offset Word 3[7:0]									3
		0x21	Not used		Phase Offset Word 3[13:8]							3
Reserved	0x0B	0x22	Reserved, do not write, leave at 0xFF								0xFE	N/A
		0x23	Reserved, do not write, leave at 0xFF								0xFF	N/A

E.4 VFTX210

[28]

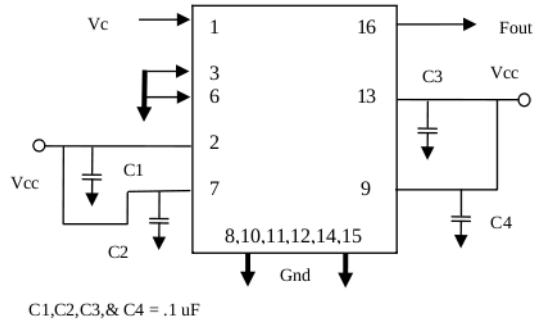
Valpey Fisher Corporation
High Frequency, Low Phase Noise Timing Solutions

Electrical Specifications

Parameter	Symbol	Condition	Min	Typ	Max	Unit	Note
Frequency Range	Fout		200		1,000	MHz	
Frequency Stability	ΔF/F	Vs. Operating Temperature B: 0°C to +70°C G: -40°C to +85°C		0.5 0.8	1.0 1.0	ppm	
		Vs. Supply Voltage Vs. Aging / Year Vs. Aging / 10 Years		± 0.1 ± 1 ± 3		ppm/V ppm ppm	First Year 10 Years
Operating Temperature Range	Ta		0° -40°		+70° +85°	°C	*Order Code B* *Order Code G*
Output		Signal	Sine Wave				
Output Level	Po	**50 Ω Load, Fout > 500 MHz**	6	8		dBm	
		50 Ω Load, Fout < 500 MHz	8	10		dBm	
Subharmonics				-42	-36	dBc	
Voltage Control	Vc		0	1.5	3.0	V	
Input Impedance	Zin		50 Ω + 1000pf // 15KΩ				
APR			± 5			ppm	
Deviation slope	ΔF/ΔVc		*Monotonic positive*				
Modulation BW	MBW			10		Hz	3dB BW

Parameter	Symbol	Condition	Min	Typ	Max	Unit	Note
Supply Voltage	Vcc		3.15	3.30	3.45	V	
Supply Current	Icc	50 Ohm Load		72	85	mA	
Start up time				3		sec	
Phase Jitter	£	12KHz to 20MHz		0.20	0.35	ps	
SSB Phase Noise	Φn	100Hz 1KHz 10KHz 100KHz 1 MHz		-92 -121 -141 -147 -150		dBc/Hz	@ 1000.0 MHz
Setability	Fnom				0.1	ppm	
Setability Voltage	Vc		1.2		1.8	V	

Connection Diagram



C1,C2,C3,& C4 = .1 uF

Pin Assignments

Pin #	Description	Pin #	Description
1	Vc	16	Fout
2	Vcc	15	Gnd
3	Gnd	14	Gnd
4	Do Not Connect	13	Vcc
5	Do Not Connect	12	Gnd
6	Gnd	11	Gnd
7	Vcc	10	Gnd
8	Gnd	9	Vcc

E.5 ATtiny2313

[33]