University College Dublin

School of Electrical and Electronic Engineering

EEEN40290 MEngSc Project

Student Name: Mohammad Naser.

Student Number: 21207944.

Project Title: Spur Free Digitally Controlled Oscillator.

Supervisor: Professor Peter Kennedy.

Declaration of Authorship

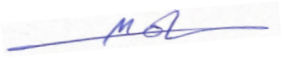
I declare that all of the following are true:

1) I fully understand the definition of plagiarism.

2) I have not plagiarised any part of this project and it is my original work.

3) All material in this report is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

I acknowledge the contribution of the following post-graduate students / post-doctoral fellows / researchers or technicians to work detailed in this report:



Signed: Date: Sep 2nd, 2022

Abstract

Indirect frequency synthesis is extensively used to generate defined frequency signals in modern communication systems and computer hardware applications. This includes Radio Frequency (RF) carrier signal generation used for channelization in wireless communication systems, and high-speed clock generation for hardware clocking applications. The indirect synthesis technique employs Phase Locked Loops (PLLs) to generate such signals. The heart of the PLL is a Digitally Controlled Oscillator (DCO). The output spectrum of the PLL is not a pure signal, meaning that the PLL does not produce a pure single tone signal due to the inherent phase noise of the DCO and the physical noise associated with the components of the PLL. The output of the PLL typically consists of the fundamental tone plus DCO's phase noise skirt component and other spurs (tones). The DCO is driven by a Digital Delta-Sigma Modulator (DDSM) which is inherently causing the tones to appear in the output spectrum when the output of the DDSM interacts with the nonlinearity in the DCO. In this work, we discuss spur mitigation methods in the presence of nonlinearity in the DCO and we describe a novel DDSM that does not produce spurs when it drives a DCO.

Acknowledgement

I would like to express my appreciation to everyone that has supported and helped me throughout my master's degree journey at University College Dublin. I would like to express my appreciation and gratitude to my supervisor Professor Peter Kennedy who has always a supporting and a patient mentor over the master's project period.

I would like to thank my family from the bottom of heart for their support they provided me to pursue a master's degree despite the difficulty. I would like to express my deepest love and admiration to my mother who did not graduate from high school, yet she has always encouraged me to get the highest education level possible saying that "*Education is most admirable way to guarantee a better future*".

At University College Dublin, I have the privilege to take courses under the supervision of highly qualified faculty that I would like to thank for their efforts and time they spent throughout my master's degree. At University College Dublin, I have significantly developed my knowledge and expertise in the field of electronic and computer engineering, mainly integrated circuits design which I think qualifies me to start my professional career and be a successful engineer in the future.

Contents

Acronyms

ADC	Analog-to-Digital Converter
ADPPL	All-Digital Phase Locked Loop
ADSM	Analog Delta-Sigma Modulator
CMOS	Complementary Metal Oxide Semiconductor
CO	Controlled Oscillator
DAC	Digital-to-Analog Converter
DCO	Digitally Controlled Oscillator
DDSM	Digital Delta-Sigma Modulator
DSM	Delta-Sigma Modulator
EFM	Error Feedback Modulator
EFM1	First Order Error Feedback Modulator
FSM	Finite State Machine
LFSR	Linear Feedback Shift Register
LPF	Low Pass Filter
LSB	Least Significant Bit
MASH	Multi stAge noise SHaping
MOS	Metal Oxide Semiconductor
MSB	Most Significant Bit
NTF	Noise Transfer Function
PD	Phase Detector
PLL	Phase Locked Loop
PVT	Process/Voltage/Temperature
RF	Radio Frequency
SQNR	Signal to Quantization Noise Ratio
STF	Signal Transfer Function
VCO	Voltage Controlled Oscillator

Chapter 1. Introduction

Delta-Sigma Modulators (DSM) have been extensively studied. DSMs can be classified into two categories: Analog Delta Sigma Modulators (ADSM) and Digital Delta-Sigma Modulators (DDSM). Analog DSMs are used in applications such as Analog-to-Digital Converters (ADCs). Less attention has been given to the DDSMs that are mainly used in the context Digital-to-Analog Converters (DACs) and fractional-N frequency synthesis. The motivation of this work is due to the wide use the DDSMs in wireless application as it is considered an important block in fractional-N frequency synthesizers producing carrier signals for channelization. In addition to that, DDSMs are used in high-speed hardware clocking applications. In this work, we aim to study different popular architectures of DDSMs, namely the Error Feed-back Modulator (EFM), and the Multi-stAge noise SHaping (MASH) topology. One important aspect of a DDSM is the spurious tones. These arise due to the inherent periodicity of the output of the DDSM when the input is constant and will be analyzed extensively in this work.

The inherent periodicity in the output of a DDSM comes from the fact that a DDSM is a Finite State Machine (FSM). A DDSM is implemented using finite precision arithmetic units that have a finite number of states. When the input of the DDSM is constant, a DDSM follows a deterministic rule for transitioning between the states, meaning that the DDSM visits each state once before repeating. In fact, the input of a DDSM must always be constant or periodic.

Therefore, the output of the DDSM is a periodic output signal. Furthermore, the output quantization error signal is also periodic.

The DDSM output period (cycle) determines how high in power the spurious tones are; when the cycle length is short, the average quantization noise power is spread over a small number of discrete tones, which according to Parseval's theorem produces high power tones. These tones are undesired and called spurious tones. However, if the cycle is sufficiently large, the quantization noise power is spread over a higher number of discrete tones, thus a smooth shaped output quantization noise is obtained that has no high-power tones. There are two main methods for cycle length maximization in DDSMs: "stochastic" and deterministic. The stochastic approach basically increases the cycle length by dithering the input of the DDSM in a pseudo-random manner. This effectively breaks the periodicity in the output of the DDSM due to the constant input. The deterministic approach increases the cycle length by modifying the architecture of the DDSM. In this work, we consider mainly the dithering method for cycle length maximization.

In Chapter 2, we review the fundamentals of indirect frequency synthesizers, namely the Phase Locked Loop (PLL) frequency synthesizer. We study Integer-N and fractional-N modes of a PLL and discuss the issues associated with each one. Furthermore, how a DDSM are used in fractional-N PLLs, and the role of the controlled oscillator.

In Chapter 3, we give a brief introduction to Digitally Controlled Oscillators (DCOs). This includes how digital control of a Voltage Controlled Oscillator (VCO) is realized, the issue of the finite resolution of a DCO due to finite frequency step of the tuning units and how that can be resolved to obtain finer frequency resolution

In Chapter 4, we study DDSMs fundamentals and noise shaping principles, review the classes of DSMs, and present two DDSM architectures, namely the Error feedback Modulator (EFM) and the Multi-stAge noise SHaping (MASH) DDSM.

In Chapter 5, we study how a MASH DDSM can be used to drive a DCO to obtain higher frequency resolution, and study how dithering is implemented to increase the DDSM cycle length and reduce spurs, we plot results of simulated DCO output phase noise, and how the phase noise is affected by nonlinearity in the DCO (tuning units mismatch), we then discuss spurs mitigation method (bit-rotator).

In Chapter 6, we apply a modification to the DDSM used in chapter 4 that implements the INIS-DDSM. We compare the results with the results obtained in chapter 4. Conclusion and suggestions for future work are discussed in chapter 7.

Chapter 2. Phase Locked Loop (PLL)

2.1 Introduction

Phase Locked Loop (PLL) is an indirect frequency synthesis architecture which is the most employed architecture in wireless applications [1] and is also used in hardware applications. In transceivers, the frequency synthesizer is an integral block in the system which generates periodic carrier signals at both ends (receiver and transmitter). The limited bandwidth available for each user in the channel in modern communication systems has made it challenging to obtain signals at defined frequencies with high channel resolution. Therefore, frequency synthesizers play a critical role when it comes to channelization in modern communication systems.

In this chapter, we will study two architectures of a PLL frequency synthesizer. Firstly, we will discuss the basic principles of operation of the integer-N frequency synthesizer. Secondly, we study the principles of fractional-N frequency synthesizer used to obtain finer frequency resolution. We will also introduce the spurious tones issue that arises in fractional-N synthesizers.

2.2 Integer-N Frequency Synthesizers

A Phase Locked Loop (PLL) is a feedback control system. A simple PLL comprises a Phase Detector (PD), a Low Pass Filter (LPF), and a Voltage-Controlled Oscillator (VCO). Fig 2.1 shows a PLL architecture with a frequency divider in the feedback path.

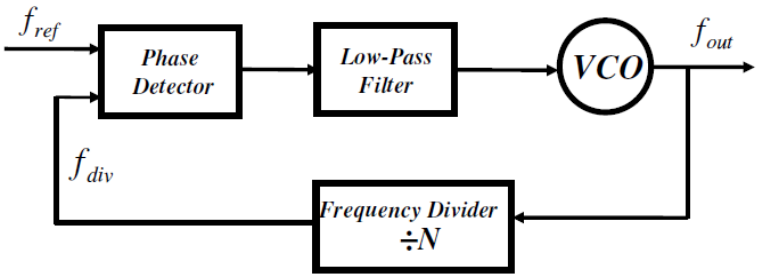


Fig. 2.1: PLL block diagram.

In the architecture in Fig. 2.1, the output signal at frequency f_{out} is obtained by multiplying the reference signal at frequency f_{ref} by an integer number (N). This can be achieved by the virtue of frequency division of the output signal in the feedback path using a frequency divider; this divides the frequency of the output signal by an integer number (N) so that the divided signal is at frequency f_{div}. The phase detector produces a signal that is proportional to the phase difference between its two inputs at frequencies f_{ref} and f_{div}. The high frequency components in the output of the phase detector are then removed by the loop filter to obtain a smooth control signal that drives the VCO. The loop is said to be locked when f_{div} is equal to f_{ref}, which implies that the phase difference between the refence and the divided signal is constant, and the control signal of the VCO is also constant. This mode of operation is called the integer-N mode, where the output signal frequency is an integer multiple of the refence frequency. When the PLL is locked, the VCO output frequency is given by Eq. (2.1).

$$f_{out} = N \cdot f_{ref} \quad (2.1)$$

Hence, the output frequency resolution in integer-N frequency synthesizers is dictated by the reference frequency, i.e. the minimum frequency separation that can be obtained at the output is f_{ref}. However, higher resolution can be obtained by employing a fractional-N synthesizer. The two architectures are distinguished by how the divider controller is driven and how it is implemented [2].

The main advantage of the integer-N synthesizer is the simplicity of implementation which has made it widely used in communication systems [1]. However, the integer-N architecture suffers from multiple drawbacks. The limited frequency resolution is considered the main drawback, meaning that the channel spacing obtained is necessarily integer multiples of the reference frequency, hence frequency spacings in fractions of the reference frequency cannot be obtained. A simple solution to obtain higher frequency resolution in integer-N synthesizers is to use a reference with a lower frequency. However, this leads to severe problems. In PLLs, it is required that the loop bandwidth is limited to one tenth of the reference frequency to guarantee loop stability [3, 4]. Therefore, decreasing the reference frequency requires decreasing the loop bandwidth to maintain stability, which in turn increases the settling time of the loop. Moreover, with lower loop bandwidth, less suppression of VCO's inherent phase noise is achieved [2]. Furthermore, a critical problem that arises with a low frequency reference is that to obtain high frequencies at the output, large values of the division ratio are needed. The noise contribution of each block (except for the VCO) in a PLL is multiplied by the division ratio [5], thus the in-band noise increases. Therefore, reduction in loop bandwidth is required to suppress the in-band noise introduced, which increases the settling time of the loop. We can conclude that, in integer-N synthesizers, there is a trade-off

between the simplicity of implementation and on one hand, settling time and phase noise on the other hand. To obtain higher resolution, without sacrificing the dynamic performance, spectral purity and stability, fractional-N synthesizers are employed which will be discussed in the next section.

2.3 Fractional-N Frequency Synthesizers

2.3.1 Principles of Fractional-N Frequency Synthesizers

Unlike the integer-N frequency synthesizer, where only integer multiples of the reference frequency can be obtained at the output, in a fractional-N synthesizer, fractional multiples of the reference frequency can be synthesized and thus provide finer frequency resolution. Thereby, a higher reference frequency can be used, and a lower division ratio in the feedback path is required, thus lower phase noise level and better spectral purity are achieved. Consequently, higher bandwidth loop filter can be used as the noise suppression requirement of the loop filter is less stringent. This leads to a faster settling time of the loop, which is critical in modern communication systems standards [4].

The architecture of the fractional-N frequency synthesizer is almost the same as the integer-N frequency synthesizer, meaning that it comprises the same main blocks, which are the phase detector, the loop filter, the VCO and the frequency divider. However, the fractional-N synthesizer is fundamentally distinguished by the control scheme of the frequency divider in the feedback path. Fig 2.2 shows the block diagram of a fractional-N frequency synthesizer with a digital accumulator that controls the divider value. A digital accumulator can be perceived as a first order digital sigma delta modulator which we will study later. For simplicity, the digital accumulator used to control the frequency divider in Fig. 2.2 is an n_0-

bit adder. The carry bit of the accumulator controls the frequency divider such that it divides the output frequency by N_0 or (N_0+1). The accumulator input X has n_0 bits. Hence, the input takes values from 0 to ($2^{n_0} - 1$). With a constant input X, the carry bit takes the value '0' ($2^{n_0} - X$) times and the value '1' X times every 2^{n_0} generated samples [2].

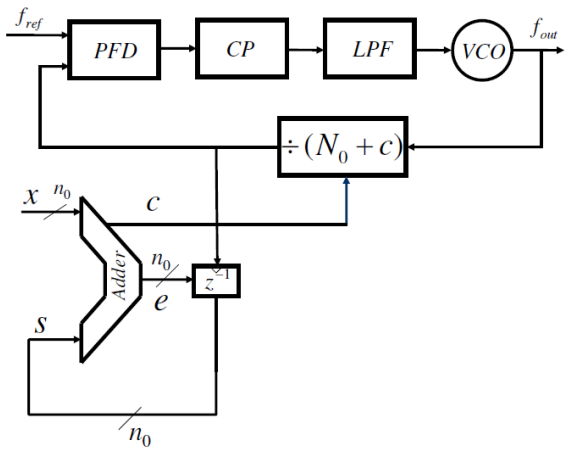


Fig. 2.2: Fractional-N PLL block diagram.

Therefore, over 2^{n_0} generated samples, the output frequency will be divided X times by (N_0+1), and ($2^{n_0} - X$) times by N_0, which results in a mean division ratio N_{mean} given by Eq. (2.2).

$$N_{mean} = N_0 + \alpha \tag{2.2}$$

Where $\alpha \in \left\{0, \frac{1}{2^{n_0}}, \frac{2}{2^{n_0}}, \ldots, \frac{2^{n_0}-1}{2^{n_0}}\right\}$. Therefore, the output frequency is given by Eq. (2.3).

$$f_{out} = (N_0 + \alpha) f_{ref} \tag{2.3}$$

Thus, a finer frequency resolution Δf is obtained, where Δf is given by Eq. (2.4).

$$\Delta f = \frac{f_{ref}}{2^{n_0}} \tag{2.4}$$

As can been seen in Eq. (2.4), the finer frequency step obtained depends directly on the accumulator's word length. The finer resolution is obtained by changing the divider moduli. Modulating the divider moduli causes a jitter signal in the feedback path, this signal is produced by the difference between the actual division modulus (N_0 *or* $N_0 + 1$) and the desired modulus ($N_0 + \alpha$), thus the jitter signal gets injected into the PLL [2], which causes spurious tones, also called fractional tunes which we will discuss in the next section.

2.3.2 Spurious Tones in Fractional-N Frequency Synthesizers

To qualitatively analyse the spurious tones in a fractional-N synthesizer, consider the fractional-N synthesizer in Fig. 2.2. with accumulator word length n_0 = 3. The accumulator is considered a finite state machine, meaning that it has finite number of states that it can be in. Taking the case where the input X is a constant, the accumulator goes through a finite number of states with a specified period length depending on the accumulator word length n_0 and the input X. Therefore, the carry out signal c exhibits periodicity. In the case of 3-bit word length accumulator, the carry out signal c is determined by the following expression [2]:

$$c = \begin{cases} 0, & X + s < 8 \\ 1, & X + s \geq 8 \end{cases} \tag{2.5}$$

As an example, let us assume that the reference frequency is 1 MHz, X = 2, $n0$ = 3 and N_0 = 100. Assuming the initial state $s[0] = 0$, the carry out signal period (cycle) is then 4 since the carry out sequence over one cycle is [0, 0, 0, 1]. *Note:* the carry out signal's average

over one cycle is $\alpha = \frac{2}{2^3} = 0.25$, which is the fractional part of the division ratio. With these values, the output frequency is calculated from equation Eq. (2.3).

$$f_{out} = \left(100 + \frac{2}{2^3}\right)1 = 100.25\ MHz.$$

The actual value of the divider is either $N_0\ or\ (N_0 + 1)$, which maps to (100 or 101) in the previous example, while the desired divider value is N = 100.25. Therefore, the instantaneous difference between the desired and the actual divider value results in an instantaneous phase modulation at the input of the phase detector, where the phase difference (error) is determined by the nature and the periodicity of the carry out control signal. In other words, the periodic control signal produces a periodic phase error. The periodic phase error signal is smoothed and then applied to the VCO, thus creating the spurious tones in the output spectrum of the synthesizer. The spurious tones are also called fractional tones since they appear at fractional multiples of the reference frequency. The loop filter attenuates the undesired components beyond the loop bandwidth. However, the fractional tones may appear at a frequency very close to the output frequency, thus they appear within the loop bandwidth and are considered as undesired spurs in the output spectrum [2]. These tones can be attenuated by decreasing the loop bandwidth. However, tones that are located very close to the output frequency are not suppressed if they are inside the loop bandwidth. Moreover, reducing the loop bandwidth compromises the settling time of the loop, which negates the fundamental motivation behind using the fractional-N synthesizer, that is obtaining fractional frequency steps while maintaining relatively large bandwidth [2]. Spurs (fractional tones) can be reduced by producing a long period pseudorandom control sequence that approximates the average value of the original sequence. Instead of having a few tones with high power in the case of short period control sequence, when long period control sequence controls the divider, the noise

power is spread over higher number frequencies (tones) rather than a few, producing shaped white noise. Using this method, frequencies higher than the loop bandwidth are attenuated, while the in-band noise is reduced but still exists and could be problematic if the loop bandwidth is high [2]. To solve the problem of in band noise, a randomized noise-shaped control sequence is produced using a digital delta sigma modulator (DDSM). The DDSM produces a pseudorandom control sequence whose power at low frequencies (in-band frequencies) is attenuated and noise power is shaped towards frequencies higher than the loop bandwidth. DDSMs and noise shaping principles will be studied in details in chapter 4.

2.4 Charge Pump PLL (CP-PLL)

Charge Pump PLL (CP-PLL) is one of the most popular analog frequency synthesizers that is widely used in low phase noise applications [6]. The traditional charge pump PLL comprises a Phase Frequency Detector (PFD) for frequency/phase comparison, a loop filter, a Voltage Controlled Oscillator, and a divider in the feedback path. Fig. 2.3 shows the PFD circuit diagram.

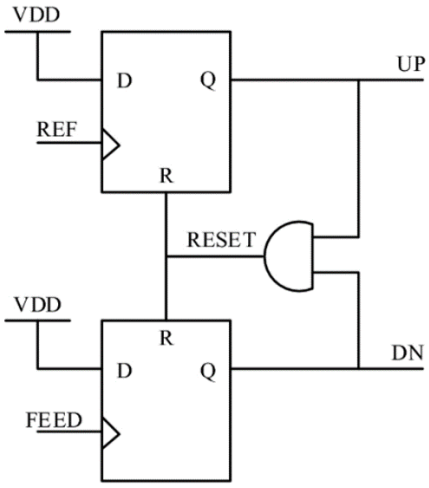


Fig. 2.3: Phase/Frequency Detector (PFD).

The phase frequency detector compares the reference signal and the feedback signal and generates UP/DN digital signals of different widths that represent the phase/frequency error between the reference and the feedback signal. Fig. 2.4 shows the block diagram of a traditional charge pump PLL.

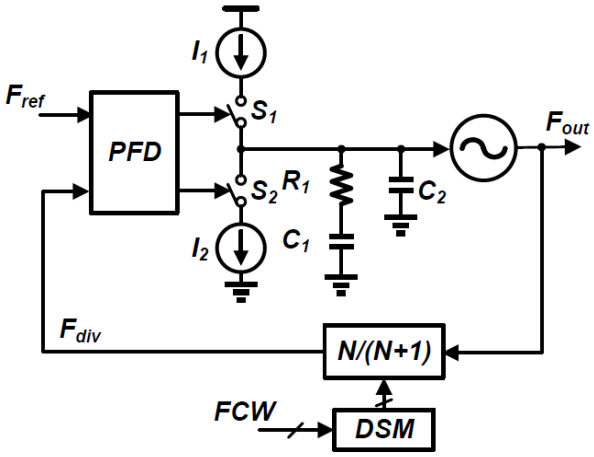


Fig. 2.4: Charge Pump (CP-PLL) block diagram.

The output of the PFD controls a charge pump switches (S_1, S_2) to either charge or discharge the capacitors of the loop filter (C_1, C_2); this translates the phase/frequency error between the reference and the feedback signals into voltage on the capacitor via the loop filter, where the error signal is averaged and smoothed. The voltage-controlled oscillator output frequency is controlled by the voltage across the loop filter capacitors, where the output frequency is corrected until the reference and the divided feedback signal are running at the same frequency. In the integer-N mode, the divider value is set to N and kept unchanged. However, in the fractional-N mode, the divider value is dithered between N and (N+1), where the divider is controlled via Delta-Sigma Modulator (DSM) with fractional input control word (FCW).

With recent advances in CMOS technology that provides high level of scaling, higher transistors' intrinsic bandwidth is obtained, better area efficiency, and lower voltage supply. However, in a CP-PLL, the analog circuitry such the charge pump makes it more difficult to achieve high linearity considering the short channel effects and the low voltage headroom. In addition, the analog components of the loop filter (C_1, C_2, R_1) occupy large area and difficult to reconfigure [6]. Due to the limitations and linearity challenges of the analog implementation of the CP-PLL, All Digital PLLs (ADPLLs) are extensively studied and analysed [6].

2.5 All-Digital PLL (ADPLL)

All Digital PLLs (ADPLLs) emerged due to the CMOS process scaling advantage which attract more research in the field of frequency synthesizers [7]. Fig 2.5 shows the block diagram of a divider-based ADPLL which uses similar structure to the CP-PLL.

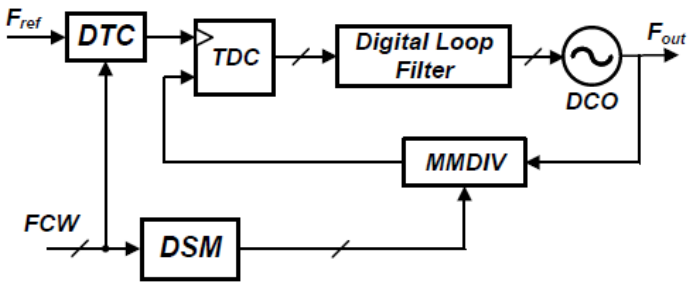


Fig. 2.5: All Digital PLL (ADPLL) block diagram.

The loop filter of the ADPLL is implemented in digital logic that benefits from the scaling advantage of the advanced CMOS technology. The DCO output is divided by multi modulus divider controlled by (FCW). The divided signal is then compared with the Digital to Time Converter (DTC)-modulated reference signal through a Time to Digital Converter (TDC). The output of the TDC is then fed to the digital loop filter to control the DCO output frequency.

Chapter 3. Digitally Controlled Oscillator (DCO)

3.1 Introduction

Indirect frequency synthesis is extensively used to generate defined frequency signals in modern communication systems and computer hardware applications. This includes Radio Frequency (RF) carrier signal generation used for channelization in wireless communication systems, and high-speed clock generation for hardware clocking applications. The indirect synthesis technique employs Phase Locked Loops (PLLs) to generate such signals. At the heart of a PLL is the Controlled Oscillator (CO), either Digitally Controlled Oscillator (DCO) or a Voltage Controlled Oscillator (VCO). In this chapter, we study principles of the DCO and how it realized.

3.2 Voltage Controlled Oscillator's Core

The controlled oscillator considered in this work is an LC tank oscillator shown in Fig. 3.1. The frequency of oscillation is proportional to $\frac{1}{\sqrt{LC}}$. Since the Inductor value cannot be changed easily due to integration and size limitations, L is held constant, and the frequency of oscillation is adjusted by controlling the capacitance value C. The capacitor is realized using Complimentary Metal-Oxide Semiconductor (CMOS) varactor, where the value of the

varactor's capacitance is controlled in an analog fashion by applying an appropriate voltage value to the CMOS varactor. Therefore, tuning the frequency of oscillation.

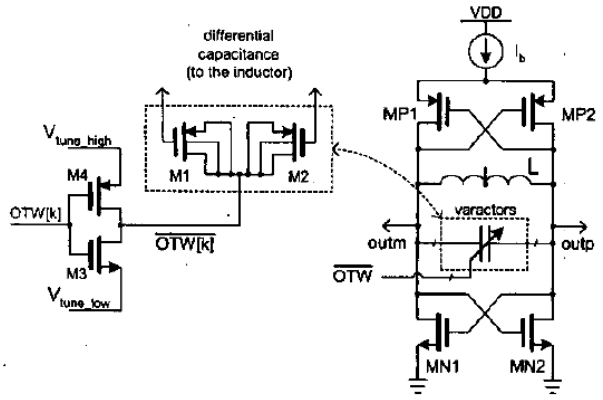


Fig. 3.1: LC tank oscillator's core.

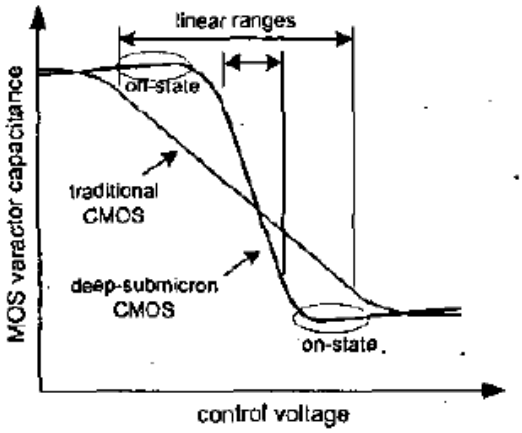


Fig. 3.2: MOS varactor's C-V characteristic.

3.3 CMOS Varactor's (C-V) Characteristics

Frequency tuning of a low-voltage deep-submicron CMOS oscillator is quite a challenging task due to its highly nonlinear frequency-voltage characteristics and low voltage headroom [8]. Fig. 3.2 shows a MOS varactor's capacitance vs. control voltage (C-V) characteristics of standard and deep submicron processes. In the deep submicron CMOS process, the linear region of varactor's capacitance is narrower than in the case of standard CMOS; or equivalently, the varactor's capacitance has become more sensitive to the control voltage in the case of CMOS deep submicron process ($KVCO = \frac{\Delta f}{\Delta V}$) is higher where $KVCO$ is the VCO's frequency sensitivity constant measured in (Hz/volt), f is the frequency of oscillation, and v is the control voltage. This makes the varactor's capacitance tuning more difficult and vulnerable to noise and operating point offsets [8].

3.4 Digital Control of CMOS Varactors

CMOS varactors in deep submicron process have high sensitivity and are vulnerable to noise in the linear region of operation on the C-V curve. Making continuous frequency tunning extremely difficult. Therefore, CMOS varactors can be operated either in the flat off-state or the flat on-state, where the varactor capacitance is less susceptible to control voltage variations and noise. Thus, varactor capacitance takes discrete values of either high or low capacitance with the highest difference between them [8]. When the oscillator varactors are controlled digitally and switched between two discrete values, either high or low capacitance, the oscillator is then called a Digitally Controlled Oscillator (DCO). DCOs fundamentally have limitations on the frequency resolution that can be obtained since the varactors are controlled digitally and switched between only two discrete values. Fig 3.3 shows an LC based oscillator structure with a switched capacitor bank [8].

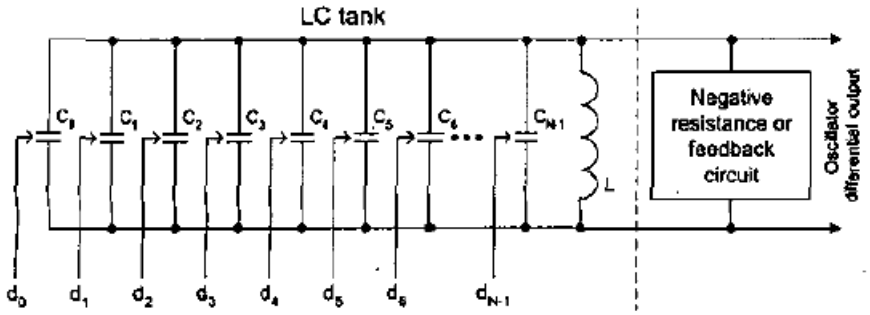


Fig. 3.3: LC based oscillator with switched capacitor bank.

Referring to Fig. 3.3, Controlling the oscillator frequency can be done by digitally controlling the binary weighted switchable capacitors, realized using CMOS varactors. Since the varactors are binary weighted, the most significant bits give a coarse step, while the least

significant bits give a fine step. Therefore, the frequency resolution is dictated by the difference between the high and low capacitance value of the least significant bit varactor. One proposed way to achieve a finer frequency resolution is to employ hybrid control scheme of the CMOS varactors [8] where the MSBs are controlled digitally giving the coarse steps by switching between high or low varactor capacitance value, while the LSB is controlled in an analog fashion, meaning that the LSB varactor is operated in the linear region so that continues capacitance values are obtained. However, this solution requires a DAC to provide and analog control signal. An alternative solution to obtain a finer resolution is LSB(s) high speed dithering. In this work, the varactors are only operated in the flat off or on state, where the capacitance is less dependent on the control voltage. Dithering the LSB(s) allows one to control the time average value of the varactor capacitance, thus, achieving finer frequency resolution. Fig. 3.4 shows the DCO discrete capacitance with high-rate dithering [8].

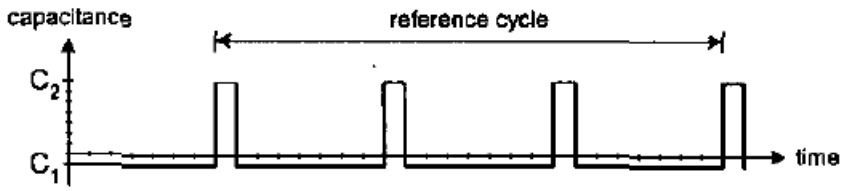


Fig. 3.4: High speed dithering of CMO capacitance.

Instead of fixing the capacitance value to C_1 or C_2, the varactor is switched between high and low value multiple times in one reference cycle. This gives an average capacitance value between C_1 and C_2, equivalently, resolution finer than ΔC, where ΔC is the LSB varactor's capacitance difference between the high and low value. In this way, higher frequency resolution is obtained from the DCO. However, the dithering in Fig. 3.4 is not random, which will see later that it produces fractional spurious tones at the output of the DCO.

3.5 DCO Architecture

The high-speed dithering of the LSB varactor is done by a Digital ΔΣ modulator (DDSM). Fig. 3.5 [8] shows the DCO architecture with ΔΣ modulator used to achieve finer frequency resolution.

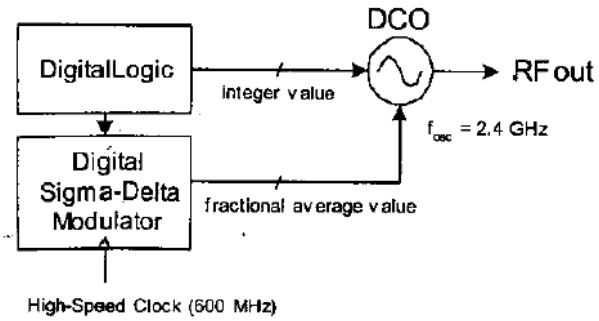


Fig. 3.5: ΔΣ Dithered DCO architecture.

In [8], The ΔΣ modulator implemented was a third order modulator using a Multi stAge noise SHaping (MASH) DDSM structure. The modulator is clocked by a 600 MHz clock. The raw frequency resolution around a DCO center frequency of 2.4 GHz was Δf^T = 23 KHz. The frequency resolution obtained by applying 5 sub LSB dithering was $\Delta f^T{}_{\Delta\Sigma}$ = 23 KHz / 2^5 = 718 Hz.

Chapter 4. Delta-Sigma Modulators (DSMs)

4.1 Introduction

Signal quantization involves resolution reduction of the signal as the quantization process produces quantization noise in the signal band (due to the limited quantizer resolution) that is added to the quantized signal. Delta-sigma modulators are used to improve the resolution of the output signal obtained from the quantization process by means of oversampling and noise shaping [2]. Oversampling as the name suggests is sampling the signal at a rate higher than the Nyquist rate so that the noise power is distributed over higher frequencies, thus reducing the average noise level in the signal band. While the idea of noise shaping is that the quantization noise is suppressed in the signal band and noise power is concentrated at higher frequencies beyond the signal band. Fig. 4.1 shows the block diagram of 1-bit delta-sigma modulator [2].

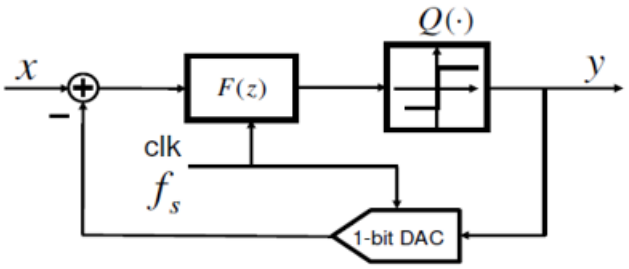


Fig. 4.1: 1-bit Delta-Sigma Modulator block diagram.

The feedback system block diagram in Fig 4.1 comprises a discrete-time filter $F(z)$ and a 1-bit quantizer denoted by $Q(\cdot)$ in the forward path, and a 1-bit Digital-to-Analog Converter (DAC)

in the feedback path. The name “Delta-Sigma Modulator” came from the fact that it is a negative feedback system which has the subtraction block, where the “Sigma” denotes the summation performed by the filter block in the feedforward path [2]. The idea of noise shaping, and resolution improvement mentioned above can be qualitatively described in Fig. 4.2 where the input signal x is sampled at a rate of f_s Hz.

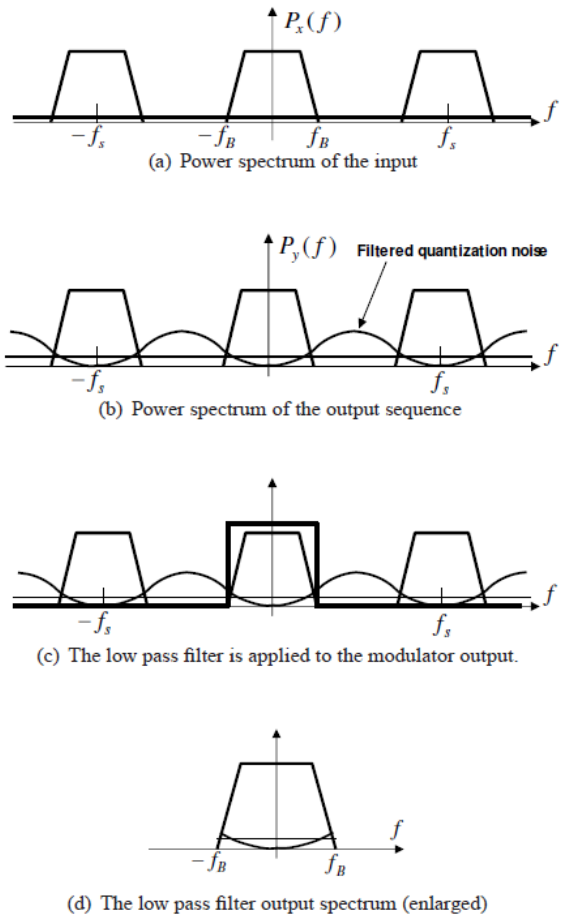


Fig. 4.2: Quantization noise shaping in a low-pass discrete-time delta-sigma modulator: (a) input signal x power spectrum. (b) The output spectrum of the modulator. (c) An ideal continuous-time low pass filter is applied to the modulator output. The filtered output spectrum.

Signal quantization by a stand-alone quantizer introduces quantization noise in the signal band, thus degrading the signal to noise ratio and output signal resolution. However, when a

delta-sigma modulator is used to quantize the signal, the noise is shaped and concentrated out of the signal band. In the case of delta-sigma modulator, the sampled input spectrum is shown in Fig. 4.2 (a), where the spectrum of the high-resolution input signal x has no noise in the signal band f_B. However, the spectrum of the modulator output signal y exhibits filtered quantization noise in and out of the signal band f_B, where the noise is concentrated at $f_s/2$ and minimized in the signal band f_B as shown in Fig. 4.2 (b). Quantizing the high-resolution input into coarse output signal introduces quantization errors, the quantization noise is filtered by the modulator so that the noise is concentrated away from the signal band, thereby the signal to quantization noise ratio is improved in the signal band [2]. Therefore, the high-resolution signal can be recovered by applying an ideal low pass filter to the low-resolution output signal y as shown in Fig. 4.2 (c). If the shaped quantization noise power in the signal band is significantly lower than the signal power, the high-resolution input signal is recovered with negligible signal to noise ratio degradation. In general, the performance of a delta-sigma modulator is evaluated by observing the output power spectral density of the delta-sigma modulator [2].

In this work, Digital Delta-Sigma Modulators (DDSMs) will be considered, where the input is a digital word, and the filter is implemented digitally. A digital delta-sigma modulator is fundamentally a finite state machine, meaning that there is a finite number of states the modulator can be in. When the input of the DDSM is constant, the output is periodic, thus creating periodic spurs that appear in the PSD of the output sequence as shown in Fig. 4.3.

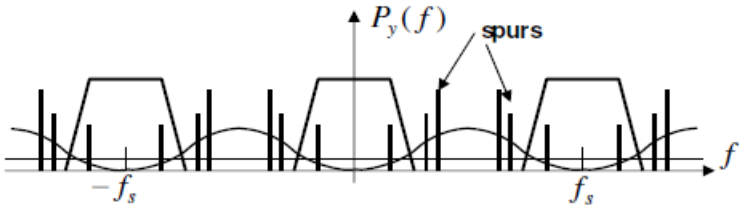


Fig. 4.3: Spurious tones in the output PSD of the modulator.

The periodic spurs degrade the performance of the delta-sigma modulator, namely degrading the spurious free dynamic range of the modulator [2]. The period (cycle) of the digital delta-sigma modulator's output sequence depends on the input value, the initial conditions and the DDSM architecture [2]. The spurs at the output of the DDSM can be reduced by *stochastic* and deterministic techniques. Deterministic spur mitigation is mainly achieved by modifying the architecture and the design of the DDSM so that the cycle length is maximized. However, the stochastic approach deals with the statistics and the randomness of the DDSM output. For instance, random dithering of the input which effectively breaks the periodicity of the output sequence, thus lengthening the output cycle. Referring to Parseval's theorem, when the output cycle is lengthened, the average noise power is distributed over more frequency components (tones), thus white noise is obtained rather than tones. In this work, input random dithering will be implemented in chapter 5 to mitigate spurs problem.

4.2 Delta-Sigma Modulation and Noise Shaping principles

As mentioned in the previous section, quantization of signals produces noise the signal band, thus degrading the signal to quantization noise ratio and the signal resolution when the signal is reconstructed. A delta-sigma modulator is employed so that so that the noise is shaped, meaning that the noise is attenuated in the signal band to obtain higher signal to quantization

noise ratio. The delta-sigma modulator is a negative feedback system where the quantizer is placed in the forward path in the loop. In this section, noise shaping and delta-sigma modulation principles will be discussed. To understand the basic principles of quantizer operation, we will consider a stand-alone mid-tread quantizer. Fig. 4.4 (a) illustrates the quantizer block, and the linearized model of the quantizer in Fig. 4.4 (b).

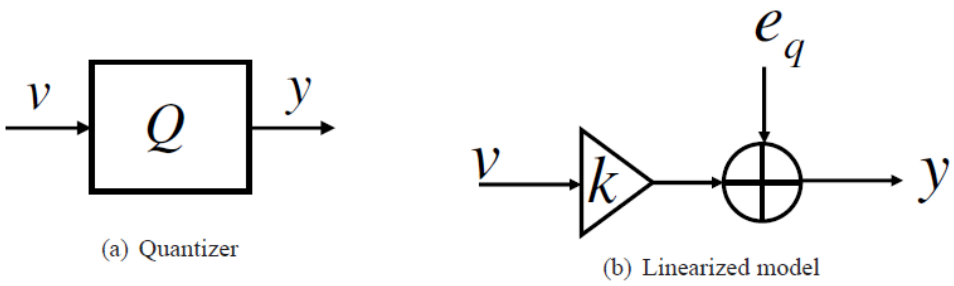


Fig. 4.4: (a) Quantizer's block symbol. (b) Quantizer's linear model.

The quantizer maps the high-resolution input to a low-resolution output that takes discrete values. The quantization process introduces addition of quantization error, which is represented in the linearized model as e_q added to the output of the quantizer, and k is a factor called quantizer gain. The quantization levels are equally separated by one delta (Δ), where Δ is the quantization step. The quantitation error bounds are dictated by Δ value, where the quantization error takes the values between $-\Delta/2$ and $\Delta/2$. Fig. 4.5 (a) shows the input-output mapping of a mid-tread quantizer that has 5 quantization levels and the quantizer gain $k = 1$ [2].

In this example, the quantizer has only 5 quantization levels that the input signal sample is mapped to. The quantizer has 2 ranges of operation: when the input is between $-\frac{5\Delta}{2}$ and $\frac{5\Delta}{2}$, the quantizer is operating in the no-overload range, while when the input is greater than $\frac{5\Delta}{2}$ or less than $-\frac{5\Delta}{2}$, the quantizer is saturated [2].

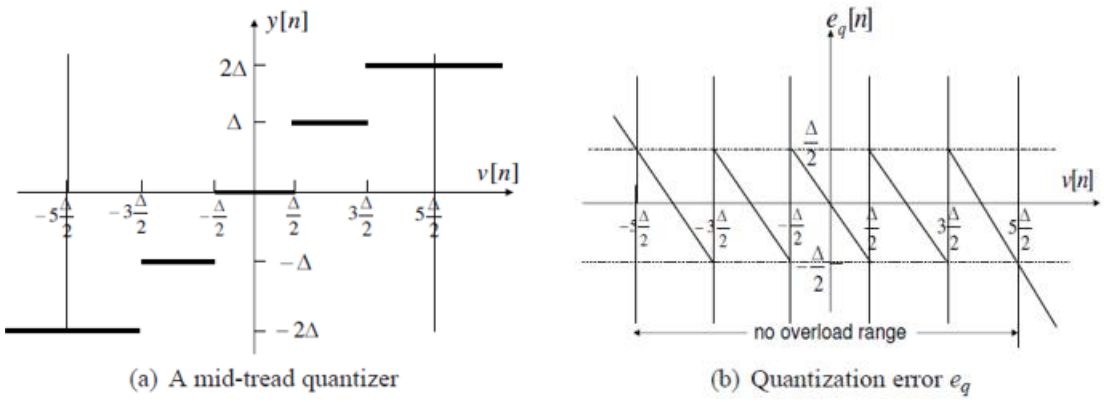


Fig. 4.5: (a) Quantizer's input-output transfer characteristic. (b) Quantization error.

Fig. 4.5 (b) shows the quantization error plot where the quantization error is bounded between $-\frac{\Delta}{2}$ and $\frac{\Delta}{2}$ in the no-overload range. The quantization noise is given by Eq. (4.1).

$$e_q = y - kv \tag{4.1}$$

The quantization noise can be reduced by increasing the number of quantization levels as the quantization step Δ decreases. The Signal to Quantization Noise Ratio (SQNR) is defined as the ratio of signal power to the quantization noise power which is dependent on the quantization step Δ. The quantization noise power P_q is given by Eq. (4.2).

$$P_q = \frac{\Delta^2}{12} \tag{4.2}$$

Therefore, in a stand-alone quantizer, to obtain SQNR values without increasing the signal power, the quantization noise power must be reduced, meaning that the quantized step Δ must be smaller. A delta-sigma modulator and the idea of noise shaping are employed to obtain higher signal to quantization noise ratios without the need to increase the quantizer levels, i.e. decreasing Δ. In a sigma delta modulator, quantizer lies in the forward path of the negative

feed-back system (modulator). Fig. 4.6 (a) shows the block diagram of a digital delta-sigma modulator. The linearized model of the modulator is shown in Fig. 4.6 (b).

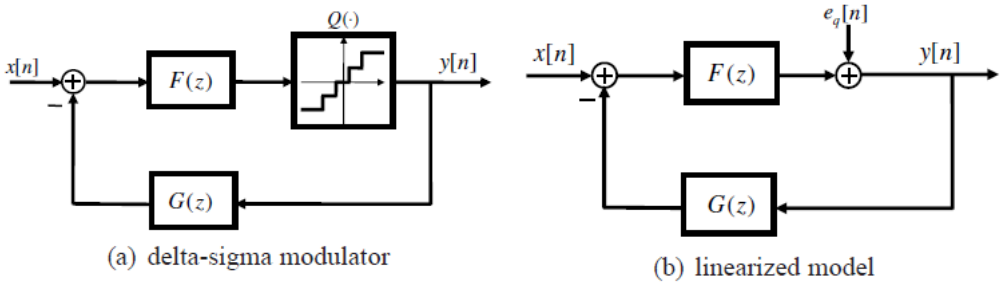


Fig. 4.6: (a) Digital Delta-Sigma Modulator block diagram. (b) Digital Delta-Sigma Modulator linear model.

The gain factor $k = 1$ in the linearized model, $G(z)$ and $F(z)$ are discrete time filters. The design of these filter is critical as they determine the noise and the signal transfer functions of the modulator. The filters are designed such that the noise transfer function is high pass, thus the noise at low frequency is suppressed to obtain higher signal to noise ratio. The sampling frequency of the modulator is assumed much greater than twice the signal bandwidth. Referring to the linearized model of the modulator in Fig. 4.6 (b), talking the z-transform of the signals in the linearized model, the output $Y(z)$ of the modulator can be expressed in terms of the input $X(z)$ and the quantization error signal $Eq(z)$ as in Eq. (4.3) [2].

$$Y(z) = STF(z)\,X(z) + NTF(z)\,Eq(z) \tag{4.3}$$

Where $NTF(z)$ is the quantization noise transfer function, and $STF(z)$ is the input signal transfer function [2]. To obtain $NTF(z)$ and $STF(z)$ expressions, we apply the super-position theorem on the two-input feedback system in Fig. 4.6 (b). The following equations for the noise and the signal transfer functions are obtained as in Eq. (4.4) and Eq. (4.5)

$$NTF(z) = \frac{1}{1 + F(z)G(z)} \tag{4.4}$$

$$STF(z) = \frac{F(z)}{1 + F(z)G(z)} \tag{4.5}$$

As mentioned earlier, the discrete filters with transfer functions $F(z)$ and $Y(z)$ are designed such that the quantization noise is shaped, meaning that the noise at low frequency is highly attenuated which is desired since we consider low pass signals in this work, whereas high frequency noise components are amplified. However, high frequency noise is significantly attenuated by the loop filter of the phase locked loop. In [2], $F(z)$ is chosen as an integrator (digital accumulator), and $G(z)$ is a unit delay.

$$G(z) = z^{-1} \tag{4.6}$$

$$F(z) = \frac{1}{1 - z^{-1}} \tag{4.7}$$

Therefore, plugging the expression of $F(z)$ and $G(z)$ in Eq. (4.4) and Eq. (4.5) to obtain noise transfer function $NTF(z)$ and the signal transfer function $STF(z)$.

$$NTF(z) = (1 - z^{-1}) \tag{4.8}$$

$$STF(z) = 1 \tag{4.9}$$

The signal transfer function $STF(z)$ indicates that input signal is passed with a unity gain, meaning that there is no signal attenuation or amplification. However, the magnitude of the noise transfer function can be obtained by substituting $z = e^{j\omega}$, and deriving the magnitude

response expression. The simplified expression of the magnitude response of the noise transfer function is:

$$\left|NTF\left(e^{j\omega}\right)\right| = \left|(1 - e^{-j\omega})\right| = 2\sin\left(\frac{\omega}{2}\right) \tag{4.10}$$

At low values of ω the noise magnitude can be approximated to $\frac{\omega}{2}$, meaning that the noise magnitude is high pass. Therefore, this Eq. (4.10) confirms how the noise is shaped in the sense that the noise is attenuated at low frequencies.

4.3 Digital Delta-Sigma Modulators (DDSMs)

4.3.1 Introduction

DDSMs are extensively used in modern wireless communication systems, they are mainly used in fractional-N synthesizers and oversampled DACs [2]. The fundamental difference between analogue and digital delta-sigma modulators is that the DDSM is a finite state machine, meaning that for a constant or periodic input, the output of the DDSM is periodic, which causes the periodic tones to appear in the DDSM output spectrum. In [9], it was shown that a short period of DDSM output causes strong spurious tones. The main advantage of the DDSM is that the input word length is reduced, which reduces linearity requirements of the analogue circuitry along the processing chain, and simplifies the processing hardware, thus saving power and area, and increasing speed. Two DDSM architectures will be studied in this section.

4.3.2 DDSM Architectures

4.3.2.1 Error Feedback Modulator (EFM)

In the EFM architecture, rather than the output, the quantization error is fed back to the input. Fig 4.7 (a) [2] shows the block diagram of a 1-bit first order Error Feedback Modulator (EFM1).

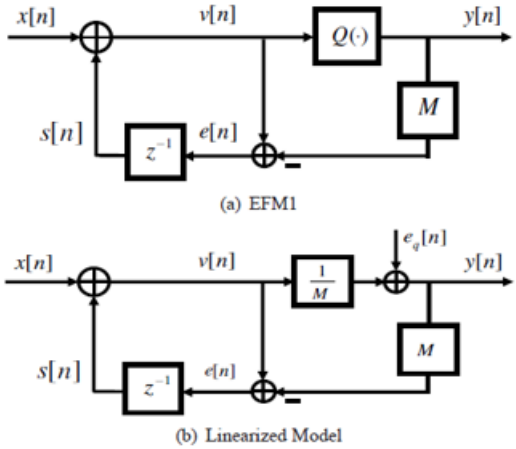


Fig. 4.7: (a) Error Feedback Modulator (EFM) block diagram. (b) EFM linearized model.

The digital filter in the feedback path is a unit delay that has a transfer function $G(z) = z^{-1}$, referring to the linear model of the EFM in Fig. 4.7 (b), where the quantizer is modeled as a gain block $(1/M)$, where M is the quantizer modulus, and an additive quantization noise error depicted by e_q, the output y can be expressed as in Eq. (4.11) [9].

$$y[n] = \frac{1}{M}x[n] + \left(e_q[n] - e_q[n-1]\right) \tag{4.11}$$

Applying z-transform on the Eq. (4.11), Eq. (4.12) is obtained:

$$Y(z) = \frac{1}{M}X(z) + (1 - z^{-1})\,E_q(z) \tag{4.12}$$

Therefore, the signal transfer function $STF(z) = 1/M$, and the noise transfer function $NTF(z) = (1 - z^{-1})$. The noise transfer function shapes the quantization noise, meaning that that the low frequency noise is attenuated, while the high frequency is noise is passed through the filter.

4.3.2.2 Multi-stAge noise SHaping DDSM (MASH DDSM)

The MASH topology allows achieving a higher order noise shaping modulator using multiple lower order modulators combined in cascade, meaning that if first order modulators are available, an l^{th} order MASH DDSM can be constructed if l first order modulators are combined. For instance, a third order MASH DDSM can be built from three first order Error Feedback Modulators (EFM1). In this case the DDSM is called (MASH 1-1-1). A MASH DDSM with 1-bit first order EFMs in the feedforward path is unconditionally stable [10]. Fig. 4.8 shows the block diagram of an l^{th} order MASH DDSM where 1-bit first order error feedback modulators (EFM1) are used in cascade.

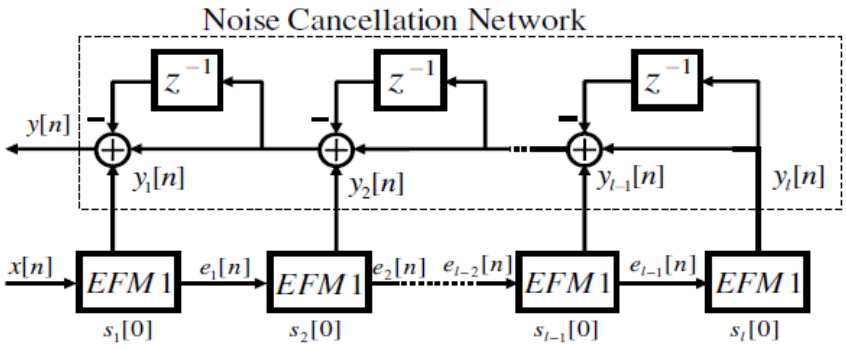


Fig. 4.8: Multi stAge noise SHaping DDSM block diagram.

An l^{th} order MASH DDSM can be constructed if l first order modulators are connected in cascade, where x is the input of the first stage EFM1, and the quantization error of each stage is fed as an input to the following EFM1 stage. The output of the stages y_i are applied to a noise cancellation network to eliminate the quantization noise contribution of all stages except the last stage. Thereby, the output of the modulator y contains information about the input signal plus quantization noise from the last stage. Referring to Fig. 4.7 (b) which shows the first order error feedback modulator (EFM1) linearized model to obtain the $NTF(z)$ and the $STF(z)$ of the l^{th} order MASH DDSM, the z-domain output of the l^{th} order MASH DDSM $Y(z)$ in Fig. 4.8 can be is given by Eq. (4.13) [9].

$$Y(z) = \frac{1}{M}X(z) + (1 - z^{-1})^{l} E_{ql}(z) \tag{4.13}$$

The signal transfer function has not changed, $STF(z) = 1/M$. Whereas the noise transfer function is now of order l, thus enhancing the low frequency noise cancellation capability. The quantization noise of the l^{th} MASH DDSM is uncorrelated to the input x. The magnitude of the envelop of the quantization noise power at the output is given in Eq. (4.14) [9].

$$\begin{aligned} N_q(f) &= \left|(1 - z)^{-l} E_q(z)\right|^2_{z=e^{j2\pi f/f_s}} \\ &= \left|2 \sin\left(\frac{\pi f}{f_s}\right)\right|^{2l} \end{aligned} \tag{4.14}$$

assuming 1/12 is the variance of the quantization noise e_q, when $f \ll f_s$, $\sin\left(\frac{\pi f}{f_s}\right) \approx \left(\frac{\pi f}{f_s}\right)$,

$$N_q(f) \approx \frac{1}{12}\left(\frac{2\pi f}{f_s}\right)^{2l} \tag{4.15}$$

Fig. 4.9 [2] shows a comparison between the PSD of the output of a third order mash DDSM and the EFM (accumulator). The initial value of the register of the EFM is set to 1, while the initial values of the MASH registers were set to 1, 0, 0 for the first, second and the third stage. The MASH DDSM randomizes the output sequence, and the quantization noise power is distributed over large number of tones, thus a smooth shaped noise plot is obtained. The plot confirms that the power spectral density grows at rate of 60 dB/dec in the case of the third order MASH DDSM. However, the EFM output power spectrum exhibits high power tones due to the lack of randomization. The first tone in the output of the EFM appears at frequency $2x10^{-3}\pi\ rad/sample$, the frequency at which the first tone appears in the output of the EFM will be discussed in chapter 5.

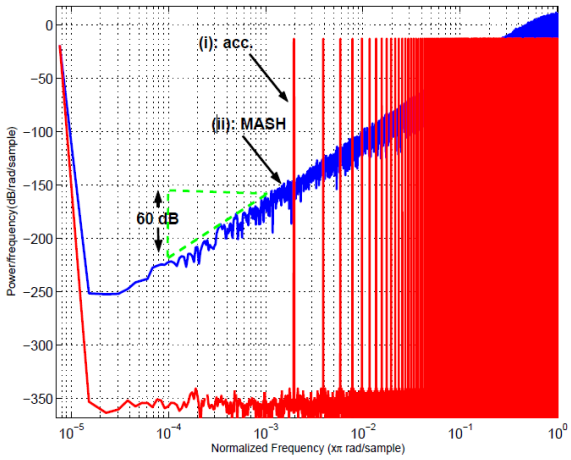


Fig. 4.9: Spectral plots of the output signals of simple accumulator (i) and, a third order MASH DDSM (ii), both with $n_0 = 18$, $X = 256$.

Discrete frequency $\omega = \pi\ rad/sample$ corresponds to continuous frequency $f_s/2$ Hz. This means that the quantization noise power is shaped towards high frequency, and the maximum noise power occurs at half the sampling frequency ($f_s/2$) Hz. To give an idea of how the time

series output of the MASH DDSM looks like, we consider third order DDSM (MASH 1-1-1) with $n_0 = 18$. Fig. 4.10 shows the output sequence of the third order MASH DDSM [2]:

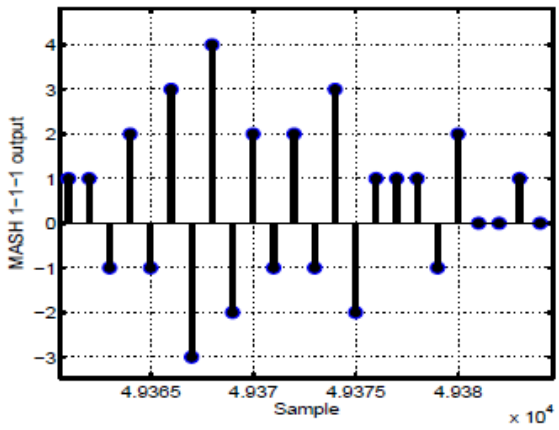


Fig. 4.10: MASH 1-1-1 DDSM output sequence.

Referring to Fig 3.8, the output of a MASH 1-1-1 DDSM is by Eq. (4.16), where the output y takes the values [-3, -2, -1, 0, 1, 2, 3, 4].

$$y[n] = y_1[n] + y_2[n] - y_2[n-1] + y_3[n] + y_3[n-2] - 2y_3[n-1] \quad (4.16)$$

Chapter 5. DDSM Application (DCO Driver)

5.1 Introduction

All-digital phase-locked loops (ADPLLs) are extensively used in frequency synthesis compared to charge pump PLL, the digital implementation of the PLL allows more flexibility, reconfigurability, transfer function stability, settling speed, precise frequency modulation capability, and amenability to integration with digital processors [11]. The fine resolution digitally controlled oscillator at the heart of the PLL considered in this work is an LC tank oscillator, where the capacitors are realized using CMOS varactors as shown in Fig. 3.1. The output frequency of a DCO is proportional to $1/(2\pi\sqrt{LC})$. Since the inductor value cannot be changed easily due to integration and size limitations, frequency tunning of the DCO is realized by keeping L constant, while controlling the capacitor banks connected in parallel with the inductor. In a DCO, the varactors' capacitance value is controlled digitally and switched between discrete values.

5.2 A Digital Delta-Sigma Modulator Driving a DCO

5.2.1 DCO Top level

In this section, we discuss how a DDSM is utilized to drive a digitally controlled oscillator to obtain a fine frequency resolution. Fig. 5.1 [11] shows the block diagram of a DCO top level that has three control varactor banks.

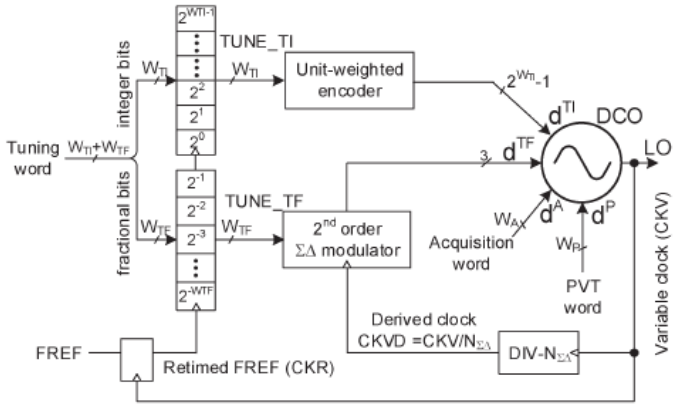


Fig. 5.1: DCO top level (Frequency control using a DDSM).

The varactor banks: *PVT* (coarse) bank controlled by d^P word, the PVT bank places the DCO frequency in the middle of a frequency band [12]. The acquisition bank controlled by d^A control word which used for channel select, and the fine bank which is split into a fractional and integer sub-banks controlled by d^{TF} and d^{TI}, respectively, the fine bank is used during the actual transmit and receive for tracking. The "Tuning word" is split into two words W_{TF} and W_{TI} to control the fractional and the integer banks. The input of the second order digital delta-sigma modulator is the fractional 8-bit control word W_{TF}, the modulator output drives the d^{TF} bits to control the fractional varactor bank of the DCO. In [11], the output of the DCO depicted by "LO" is divided by $N_{\Delta\Sigma}$ to generate the modulator clock "CKVD" (the DDSM is clock at a rate of $f_s = 500\ MHz$). The DDSM is a second order noise shaping MASH DDSM with an 8-bit input, which is used to dither the fine varactors bank between different values so that the time average varactors frequency step is smaller than the raw frequency step of the fine varactor, that is $\Delta f^T = 20\ kHz$.

5.2.2 Delta-Sigma Dithering and Spurs

As discussed before, a second order MASH DDSM is used to drive dither the fine varactors therefore obtaining finer effective capacitance resolution. Fig. 5.2 [11] shows the second order MASH DDSM used for fractional frequency dithering of the fine DCO varactors.

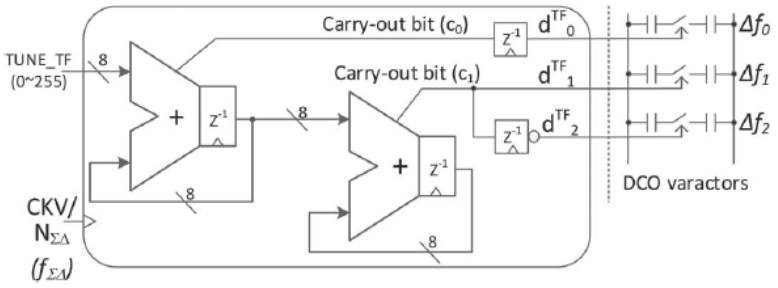


Fig. 5.2: 2^{nd} order DDSM dithering DCO varactors.

The input of the DDSM "TUNE_TF" is the fractional tuning 8-bit word obtained from the "Tuning word" depicted in Fig. 5.1. DDSM converts the 8-bit binary weighted tuning word to a high rate 3-bit unit weighted word (${d_0}^{TF}$, ${d_1}^{TF}$ and ${d_2}^{TF}$). The output of the modulator is expressed by Eq. (5.1), and can be expanded as in Eq. (5.2) [11]:

$$out^{\Delta\Sigma}[k] = {d_0}^{TF} + {d_1}^{TF} + {d_2}^{TF} \tag{5.1}$$

$$out^{\Delta\Sigma}[k] = c_0[k-1] + c_1[k] - c_1[k-1] \tag{5.2}$$

where k is the discrete-time index. The output of the DDSM drives the three fine varactors of the DCO with frequency steps ${\Delta f_0}^{T}$, ${\Delta f_1}^{T}$ and ${\Delta f_2}^{T}$ (unit of Hz). Therefore, the instantaneous value of the fractional frequency tuning f^{TF} can expressed as in Eq. (5.3) [11]:

$$f^{TF}[k] = c_0[k-1].{\Delta f_0}^{T} + c_1[k].{\Delta f_1}^{T} - c_1[k-1].{\Delta f_2}^{T} \tag{5.3}$$

Ideally, the varactors frequency steps are equal, meaning that $\Delta f_0{}^T = \Delta f_1{}^T = \Delta f_2{}^T = \Delta f^T$. Thus, the instantaneous fractional frequency tuning can be expressed as in Eq. (5.4):

$$f^{TF}[k] = out^{\Delta\Sigma}[k].\Delta f^T \tag{5.4}$$

Therefore, the time average fractional frequency command is $(TUNE\_TF.\Delta f^T)/2^8$. This means a frequency resolution of $\boldsymbol{\Delta f^T/2^8}$ is obtained. The quantization noise transfer function of a second order MASH DDSM in the z-domain is given by Eq. (5.5):

$$NTF(z) = (1 - z^{-1})^2 \tag{5.5}$$

In [3], the second order MASH DDSM is clocked at rate of $f_{\Delta\Sigma}$. The frequency domain delta-sigma phase noise resulting from the DDSM dithering at frequency offset f is given by Eq. (5.6) [12].

$$PN(f) = \frac{(\Delta f^T)^2}{12\,.\,f_{\Delta\Sigma}}\,.|N_{\Delta\Sigma}(f)|^2\,.\frac{1}{f^2} \tag{5.6}$$

where $|N_{\Delta\Sigma}(f)|^2$ is second order DDSM output quantization noise power, given the quantization noise transfer function of the second order MASH DDSM $NTF\ (z) = N_{\Delta\Sigma}(z) = (1 - z^{-1})^2$. Substituting $z = e^{j2\pi f/f_{\Delta\Sigma}}$ in $N_{\Delta\Sigma}(z)$ expression and simplifying it, thus

$$|N_{\Delta\Sigma}(f)|^2 = \left(2\sin\left(\frac{\pi f}{f_{\Delta\Sigma}}\right)\right)^4 \tag{5.7}$$

Plugging $|N_{\Delta\Sigma}(f)|^2$ in Eq. (5.6), we get the delta-sigma dithering shaped noise expression of the second order MASH DDSM [12].

$$PN(f) = \frac{(\Delta f^T)^2}{12\,.\,f_{\Delta\Sigma}}\,.\left(2\sin\left(\frac{\pi f}{f_{\Delta\Sigma}}\right)\right)^4.\frac{1}{f^2} \tag{5.8}$$

5.2.2.1 DDSM Idle Tones

With a constant input ("TUNE_TF") to the DDSM in Fig. 5.2, the output of the modulator will be periodic since the DDSM is a finite state machine. The periodic output results in periodic tones in the spectrum of the output of the DDSM, where the idle tones appear at frequency f_{idle} and its harmonic frequencies [13]. In general, with an input TUNE_TF = 2^D, the output of the MASH DDSM with N-bit accumulators repeats every 2^{N+1-D} clock cycles [3], where D is the number of consecutive zeros in the input word. For example, consider an 8-bit MASH DDSM with an input (TUNE_TF = 0000_0100), the DDSM output cycle is $2^{8+1-2} = 128$, meaning that the output of the DDSM repeats after 128 DDSM clock cycles. The DDSM is clocked at a rate of $f_{\Delta\Sigma} = f_s$. Therefore, the fundamental tone appears at f_{idle} given by Eq. (5.9):

$$f_{idle} = \frac{f_{\Delta\Sigma}}{2^{N+1-D}} \tag{5.9}$$

Consequently, with the input (0000_0100), the fundamental idle tone appears at $f_{idle} = \frac{f_s/2}{64} = 0.015\ (f_s/2)$. Fig. 5.3 shows the DCO output phase noise when the varactors' frequency step is matched ($\Delta {f_0}^T = \Delta {f_1}^T = \Delta {f_2}^T = \Delta f^T = 20\ kHz$).

Referring to Fig. 5.3, where discrete-time frequency of π rad/sample corresponds to $f_s/2$, the first idle tone in the DDSM output spectrum appears at the fundamental idle frequency $f_{idle} = 0.015\ \left(\frac{fs}{2}\right) Hz$.

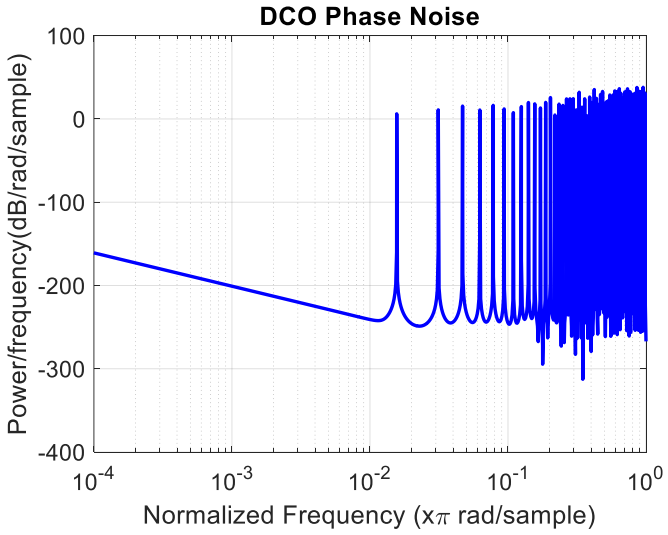


Fig. 5.3: DCO output phase noise. N = 8, TUNE_TF = (0000_0100). Matched varactors' frequency step ($\Delta {f_0}^T = \Delta {f_1}^T = \Delta {f_2}^T = \Delta f^T = 20\ kHz$).

Having high power tones in the spectrum of the DDSM output has significant impact on the DCO output phase noise, as these high tones are not hidden under the intrinsic phase noise of the DCO shown in Fig 5.8. This results in high spurs at the output of the frequency synthesizer.

5.2.2.2 Spurs Due to Varactors Frequency Step Mismatch

Referring to Fig. 5.2, The DDSM drives the three fine fractional varactors of the DCO. The varactors' fine frequency step is: ${\Delta f_0}^T$, ${\Delta f_1}^T$, ${\Delta f_2}^T$. The ideal case is when all the varactors' frequency steps are equal, meaning that ${\Delta f_0}^T = {\Delta f_1}^T = {\Delta f_2}^T = \Delta f^T$. In this section, we study the effect of varactors' frequency step mismatch on the output phase noise of the DCO. Consider the case when ${\Delta f_1}^T = {\Delta f_2}^T = \Delta f^T$, and ${\Delta f_0}^T = \Delta f^T + \boldsymbol{\delta f^T}$. Consequently, the fractional tuning frequency is given by Eq. (5.10) [11]:

$$\widehat{f^{TF}}[k] = (c_0[k-1] + c_1[k] - c_1[k-1])\Delta f^T + c_0[k-1]\delta f^T \qquad (5.10)$$

The fractional tuning frequency $\widehat{f^{TF}}$ (in the case of frequency step mismatch) is equivalent to the fractional tuning frequency f^{TF} (in the ideal case) plus a first order DDSM driving a varactor having a frequency step $\delta f^T = {\Delta f_0}^T - \Delta f^T$. The additional term can be a source of possible spurs. Fig. 5.4 shows the DCO output phase noise when the fine DCO varactors' frequency step is 10% mismatched (${\Delta f_0}^T = 22\ kHz,\ {\Delta f_1}^T = 18\ kHz,\ {\Delta f_2}^T = 20\ kHz$) compared to the case of the matched varactors, where the nominal tuning frequency step is $\Delta f^T = 20\ kHz$. The DCO fine varactors are driven by a second order 8-bit MASH DDSM with input TUNE_TF = (0000_0100).

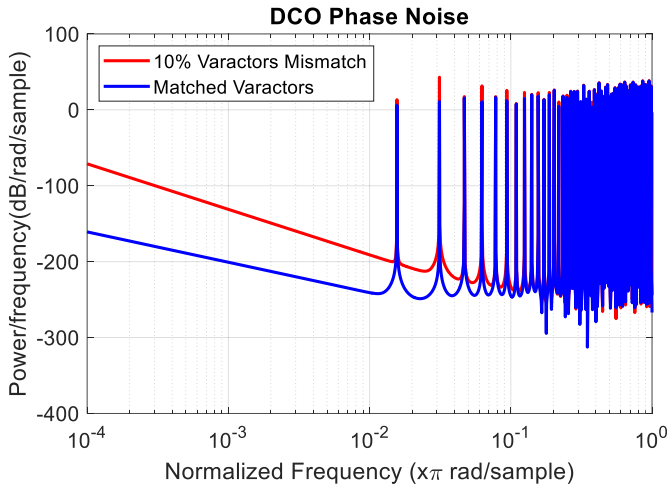


Fig. 5.4: DCO output phase noise. N = 8, TUNE_TF = (0000_0100). 10% varactors' frequency step mismatch (${\Delta f_0}^T = 22\ kHz,\ {\Delta f_1}^T = 18\ kHz$ and ${\Delta f_2}^T = 20\ kHz$) compared to the case when varactors are matched.

The fractional tones in the case of 10% varactors frequency step mismatch appear at the same frequency when the varactors are matched. However, the tones appear to have higher power in the case of mismatched varactors and the noise floor is higher than in the case of matched varactors.

5.2.3 DCO Output Spurs Mitigation

5.2.3.1 Idle Tones Stochastic Mitigation Technique (Dithering)

The tones in the output of the DCO are inherently coming from the periodicity of the output of the DDSM when the input is constant. The tones get more significant when the DDSM output cycle (period) is short, meaning that the noise power is distributed over smaller number of tones, giving higher power tones. Therefore, the tones' power can be reduced if the cycle length is maximized. A stochastic technique we consider is called "LSB dithering", which is one of the techniques to mitigate the spurs due to DDSM idle tones. The method is to randomly dither the LSB input of the DDSM input to break the periodicity of the DDSM output, thus achieving a longer output cycle. This method is commonly used in practice to improve the modulator's spectral performance [10]. Ideally, dithering improves the statistical properties of the quantization noise such that the white noise approximation applies [2]. When the quantizer noise is white, the output quantization noise is free of spurious tones. In this work, we consider 1-bit dithering added to the input word as shown in Fig. 5.5, where a 1-bit pseudo random signal d is added to the input of the DDSM.

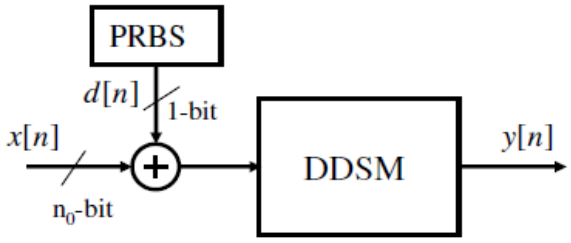


Fig. 5.5: 1-bit dithering of the DDSM.

Therefore, the input word of the DDSM "TUNE_TF" that corresponds to x in Fig. 5.5 is still 8-bit. However, an additional bit d coming from a random source PRBS is added to the input

word and fed to the first accumulator of the MASH DDSM. Fig. 5.6 illustrates the block diagram of the modified second order MASH DDSM used to drive the DCO varactors [11]:

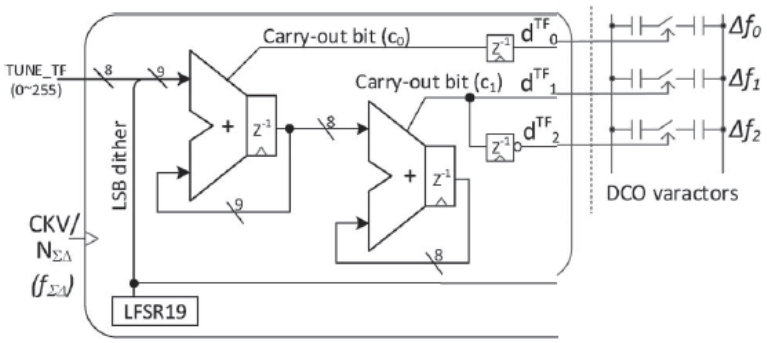


Fig. 5.6: LSB dithering of the DDSM input (TUNE_TF).

The Linear Feedback Shift Register (LFSR19) is modelled in MATLAB as a random binary source, which is considered as a dither to the constant 8-bit input to the DDSM. Dithering the least significant bit of the DDSM input eliminates the high-power tones by distributing the noise power over more frequency components. However, LSB dithering introduces low frequency noise at the output of the DCO, which rolls of at a rate off -20dB/dec. LSB dithering low frequency noise is typically not a concern as the dithering noise can be kept below the intrinsic phase noise of the DCO as shown in Fig. 5.8 [11]. The dithering contribution to the output phase noise can be expressed as in Eq. (5.11) [11].

$$PN_{LSBdither}(f) = \frac{(\Delta f^T)^2}{2^{2N+2} \cdot f^2 \cdot f_{\text{dither}}} \tag{5.11}$$

where f_{dither} is the rate at which the LSB is dithered ($f_{\text{dither}} = f_{\Delta\Sigma} = f_s = 500\ MHz$). The total dithering noise therefore is the sum of Eq. (5.11) and Eq. (5.8). Fig. 5.7 shows the DCO output phase noise when the input is dithered with same input and initial conditions of the DDSM.

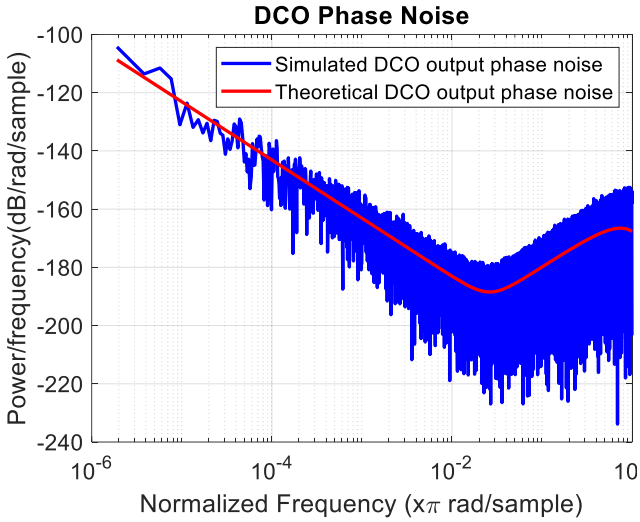


Fig. 5.7: DCO output phase noise with LSB dithering.

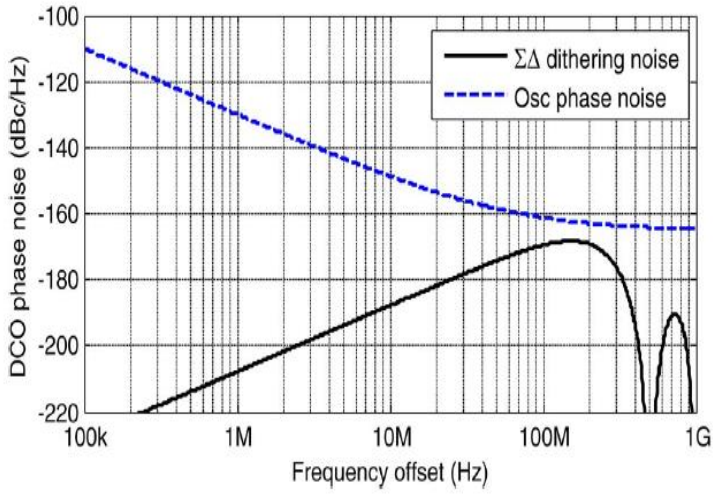


Fig. 5.8: Oscillator phase noise and dithering noise.

The DCO output phase noise is the sum of the delta-sigma shaped noise and the LSB dithering noise that rolls off at 20 dB/dec. The DDSM dithering noise is more critical at high frequencies due to the stringent phase noise requirements such that the dithering noise is kept hidden under the DCO intrinsic phase noise [11].

To examine the effect of varactors mismatch on dithered DDSM performance, the DCO output phase noise when varactors frequency steps are 10% mismatched is shown in Fig. 5.9. The phase noise floor is higher in the case of mismatched DCO varactors. Additionally, significant tones appear at the DCO phase noise, which are problematic as they violate the stringent phase noise requirement at high frequencies.

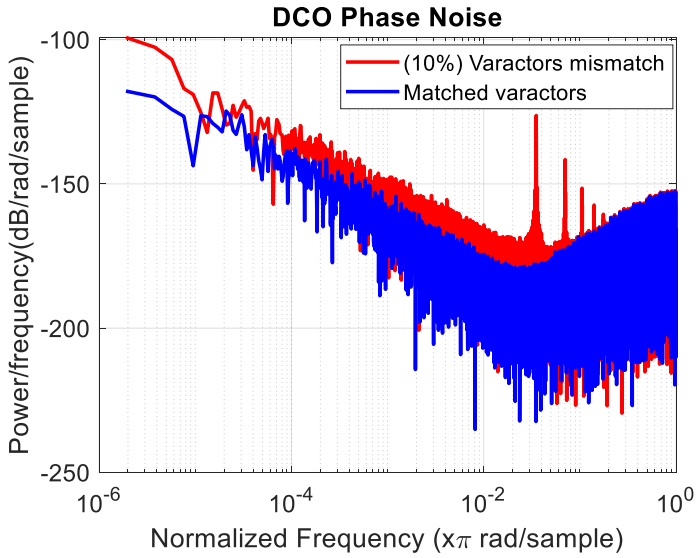


Fig. 5.9: DCO output phase noise (with LSB dithering): 10% varactors' frequency step mismatch ($\Delta {f_0}^T = 22\,kHz$, $\Delta {f_1}^T = 18\,kHz$ and $\Delta {f_2}^T = 20\,kHz$) compared to output phase noise in the case of matched varactors.

5.2.3.2 Mitigation of Spurs Due to Varactors Mismatch

The spurs caused by the frequency step mismatch of the DCO varactors can be mitigated by adding a random bit rotator between the output of the DDSM and the controlled varactors [11] as shown in Fig. 5.10. Previously, each bit of the 3-bit output of the DDSM was mapped to a designated DCO varactor. In the proposed bit rotator scheme, the 3 bits output of the DDSM are dynamically mapped to the DCO varactors, meaning that the bits are rotated over all the varactors.

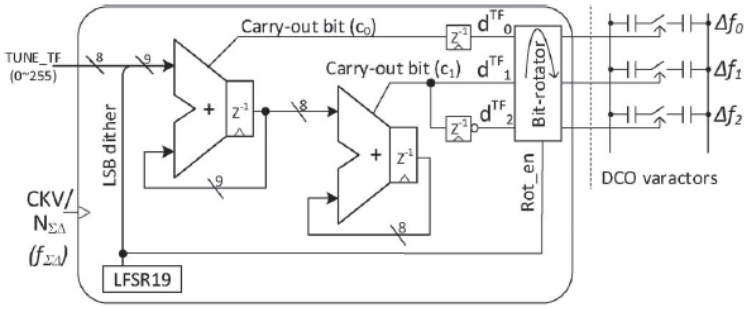


Fig. 5.10: Modified block diagram (bit rotator).

For instance, the sequence [2, 1, 0] → [0, 2, 1] → [1, 0, 2] → [2, 1, 0] . . ., where the bit rotator is controlled by "Rot_en" bit. The bit rotator randomizes the energy caused by frequency step mismatch resulting in white noise at the input of the DCO [11]. The unit mismatch randomization noise is given by Eq. (4.12) [11].

$$PN_{MM\ dither}(f) = \frac{(\sigma f^T)^2}{2 \cdot f^2 \cdot f_{\Delta\Sigma}} \tag{5.12}$$

where σf^T is the standard deviation of the tunning step sizes. Fig. 5.11 shows the simulation results when a bit rotator was implemented to mitigate the tones that appear due to varactors' frequency step mismatch.

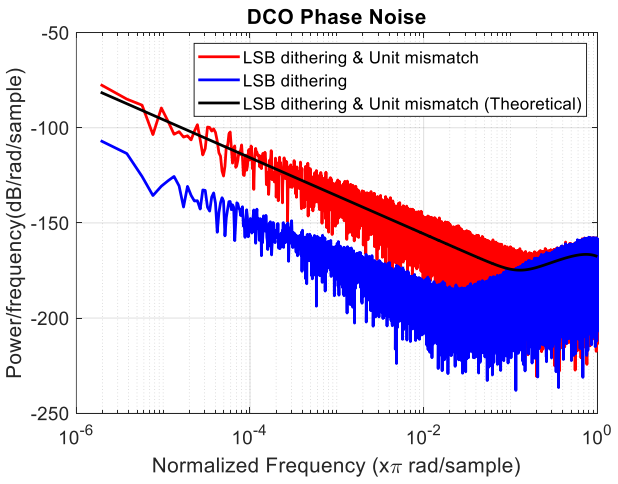


Fig. 5.11: DCO output phase noise.

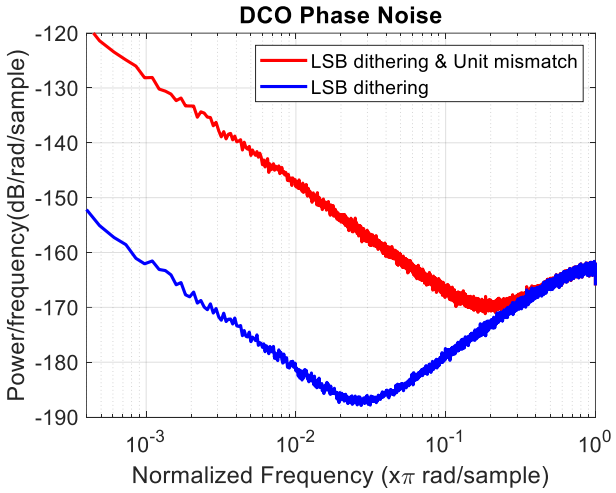


Fig. 5.12: DCO output phase noise (welch).

Looking at Fig. 5.11, the tones that appeared when the varactors are mismatched are now eliminated. However, noise floor is higher due to the mismatch randomization noise contribution expressed in Eq. (5.12). Fig. 5.12 shows smooth plots of the DCO output phase noise using (pwelch). Simulation starting from discrete-time frequency $\omega = 4x10^{-4}\ rad/sample$ which corresponds to continuous frequency 100 kHz since sampling frequency $f_s = 500\ MHz$. The total dithering noise is now the sum of Eq. (5.12), Eq. (5.11) and Eq. (5.8).

The theoretical plots obtained using MATLAB for each noise component at the output of the DCO (the ΔΣ dithering shaped noise, the LSB dithering noise and the unit mismatch randomization noise) and the total dithering noise which is the sum of all components are shown in Fig. 5.13.

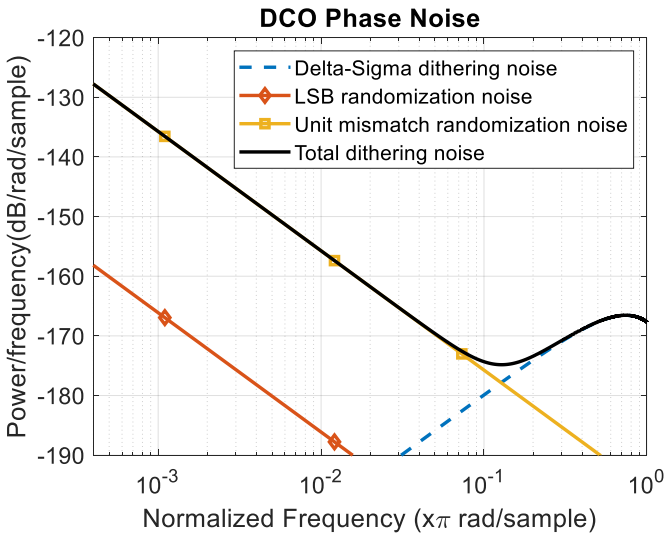


Fig. 5.13: Theoretical noise profiles.

It can be noticed from Fig. 5.13 that the total dithering noise profile is dictated by the unit mismatch randomization noise at low frequencies, whereas at high frequencies is dictated by the ΔΣ dithering noise.

Chapter 6. INIS DDSM

In this chapter, we briefly describe the INIS DDSM. In the traditional MASH DDSM discussed in chapter 4 and chapter 5, the DDSM produces spurs (idle tones) when the DDSM output interacts with the nonlinearities in the DCO. These tones appear in the in the DCO output phase noise. The nonlinearity in the DCO considered was varactors' frequency step mismatch, that was the nonlinearity the DDSM output interacted with that produced spurs in the DCO output phase noise. To mitigate the spurs due to mismatch, a bit rotator was implemented to randomize the unit mismatch of the varactors' frequency step which randomizes the energy caused by frequency step mismatch resulting in white noise at the input of the DCO.

The INIS DDSM is inherently immune against spurs, meaning that the output of the INIS DDSM does not produce spurs when it interacts with the nonlinearity in the DCO. Therefore, unit mismatch randomization is not needed to produce a white noise at the output. The output of the INIS DDSM is defined as in Eq. (6.1).

$$Y(z) = STF(z)\,X(z) + NTF(z)\,Eq(z) \tag{6.1}$$

where $STF(z)$ is the signal transfer function and $NTF(z)$ is the quantization noise transfer function. The noise transfer function $NTF(z)$ can expressed as the multiplication of two separate noise transfer functions $NTF_1(z)$ and $NTF_2(z)$ as in Eq. (6.2) [14].

$$NTF(z) = NTF_1(z) \,.\, NTF_2(z) \tag{6.2}$$

The first term of the noise transfer function $NTF_1(z)$ is given by Eq. (6.3) and the second term $NTF_2(z)$ is given in Eq. (6.4) [14].

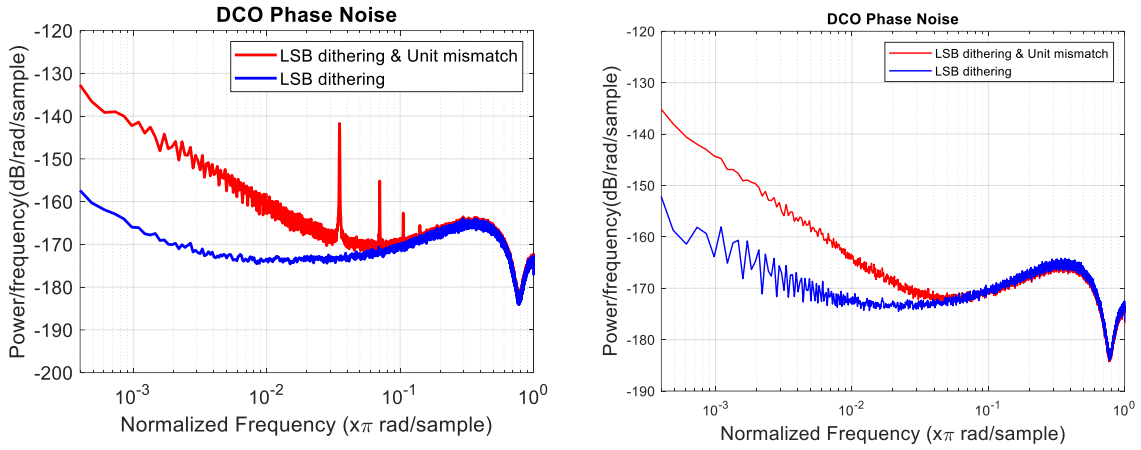
$$NTF_1(z) = (1 - z^{-1}) \tag{6.3}$$

$$NTF_2(z) = (1 - z^{-2} - z^{-3}) \tag{6.4}$$

Therefore $NTF(z)$ is given by Eq. (6.5)

$$NTF(z) = (1 - z^{-1} - z^{-2} + z^{-4}) \tag{6.5}$$

The INIS DDSM was implemented by modifying the input of second stage of the traditional second order MASH DDSM in MATLAB code in appendix 1. However, the original MATALB code that the previous results were obtained from produced spurs when it was modified to an INIS DDSM. Fig. 6.1 (a) shows the DCO output phase noise obtained when the INIS DDSM was implemented by applying the modification to the second stage of the traditional second order MASH DDSM.



(a) (b)

Figure 6.1: DCO output phase noise: (a) original code modified to implement the INIS DDSM (without a bit rotator). (b) DCO output phase noise: Valerio's code implementing the INIS DDSM (without a bit rotator).

In Fig. 6.1 (a), the envelope of the DCO output phase noise came out as expected. However, in the case of unit mismatch between DCO varactors' frequency step, it can be noticed that high power tones appear in the DCO output phase noise even though tones are not expected to appear in the output since the INIS DDSM is inherently immune to nonlinearities in the DCO. Therefore, another MATLAB code from (Valerio Mazzaro - appendix 2) that implements the INIS DDSM was used to obtain the correct result for the DCO output phase noise shown in Fig. 6.2 (b). The DCO output phase noise with varactors' frequency step mismatch is spur free, meaning that no tones in the output of the DCO spectrum that might violate the intrinsic phase noise profile of the DCO. This confirms that the INIS DDSM is immune to mismatch nonlinearity in the DCO.

Chapter 7. Conclusion

In brief, it can be concluded that DCOs are necessarily nonlinear and driven by controllers to mitigate the effects of the nonlinearity. Digital Delta-Sigma Modulator (DDSM) was studied and used to control the DCO. However, the conventional DDSM produces tones in the output spectrum of the DCO when it interacts with nonlinearities in the DCO. These tones may violate the intrinsic phase noise of the DCO, meaning that the tones can not be hidden under the intrinsic phase noise profile of the DCO. In this work, a modification to the conventional DDSM that implements the INIS DDSM was described. The INIS DDSM does not produce tones when it interacts with nonlinearities in the DCO. Therefore, better spectral purity is obtained in the output spectrum of the DCO.

References

[1] B. Razavi, "Challenges in the design of frequency synthesizers for wireless applications," in *Proc. of IEEE CICC 1997, IEEE Custom Integrated Circuits Conference*, 1997.

[2] M. P. Kennedy and K. Hosseini, Minimizing Spurious Tones in Digital Delta-Sigma Modulators, Springer , 2011.

[3] S. Pellerano, S. Levantino, C. Samori and A. Lacaita, "A dual-band frequency synthesizer for 802.11 a/b/g with fractional-spur averaging technique," in *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, 2005, pp. 104–106..

[4] B. Razavi, Phase-Locking in High-Performance Systems, IEEE Press, 2003.

[5] Z. Ye, "Modelling, simulation and architecture modification of delta-sigma fractional-N frequency synthesizers," in *Ph.D. dissertation*, 2008.

[6] J. Du, "Reference Oversampling for Digital Frequency Synthesis," in *Ph.D. Dissertation*, 2021.

[7] D. Leipold, K. Mohammad and D. Leipold, "All-digital TX frequency synthesizer and discrete-time receiver for bluetooth radio in 130-nm CMOS," in *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2278–2291, Dec. 2004..

[8] R. B. Staszewski, D. Leipold, C. M. Hung and P. Balsara, "A first digitally-controlled oscillator in a deep-submicron CMOS process for multi-GHz wireless applications," in *IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, 2003.

[9] P. M. Kennedy, "Recent advances in the analysis, design and optimization of Digital Delta-Sigma Modulators," 2012.

[10] S. R. Norsworthy, R. Schreier and G. C. Temes, Delta-Sigma Data Converters: Theory, Design and Simulaion, New York: IEEE Press, 1997.

[11] J. Zhuang, K. Waheed and R. B. Staszewski, "Design of Spur-Free ΣΔ Frequency Tuning Interface for Digitally Controlled Oscillators," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 1, pp. 46-50, Jan. 2015.

[12] R. B. Staszewski, C.-M. Hung, N. Barton, M.-C. Lee and D. Leipold, "A digitally controlled oscillator in a 90 nm digital CMOS process for mobile phones," in *IEEE Journal of Solid-State Circuits*, vol. 40, no. 11, pp. 2203-2211, Nov. 2005.

[13] A. V. Peterchev and S. R. Sanders, "Quantization resolution and limit cycling in digitally controlled PWM converters," in *IEEE 32nd Annual Power Electronics Specialists Conference (IEEE Cat. No.01CH37230)*, 2001.

[14] V. Mazzaro and M. P. Kennedy., A Family of ΔΣ Modulators with Inherent Spur Immunity when interacting with a Nonlinearity. (preprint).

APPENDIX

1) Mohammad's Code (MASH DDSM, INIS DDSM)

```
clear;
n = 8; % DDSM number of bits
M=2^n; % Quantizer modulus
% Define the number of simulation points
samples= 2^20;
% Define the input
order = 2;
input=4;
% Calculate the variance for linear prediction
delta_fT = 20e3; % Nominal frequency tuning step size
f_sigma_delta = 500e6;  % Sampling frequency (DDSM clock)
variance= delta_fT^2/(12*f_sigma_delta);  % Quantization
noise variance

d_sw=1;  % Dither switch
if (d_sw == 1)
dither =[1:samples;round(rand(1,samples))]'; % Random
dither signal generation
else
dither = [1:samples;zeros(1,samples)]';  % Dither signal
zero
end


x = zeros(1,samples); % Inputs vector
x(:) = input; % Input vector set to a constant value

%
%{
```

```
============= pre-define matrices for speed
v1 = zeros(Nsamples, 1);
e1 = zeros(Nsamples, 1);
s1 = zeros(Nsamples, 1);
y1 = zeros(Nsamples, 1);
%}

e1 = zeros(1,samples);
e2 = zeros(1,samples);
e3 = zeros(1,samples);

to_normalized_phase = 1/delta_fT;
% Varactors noiminal freuqncy tuning steps
delta_f1_nom = delta_fT *to_normalized_phase;
delta_f2_nom = delta_fT *to_normalized_phase;
delta_f3_nom = delta_fT *to_normalized_phase;
% Mismatched varactors' tuning steps
delta_f1 = 0.9 *delta_fT*to_normalized_phase;
delta_f2 = 1  *delta_fT*to_normalized_phase;
delta_f3 = 1.1 *delta_fT*to_normalized_phase;

mismatched_frequency_steps  = [delta_f1  delta_f2
delta_f2];
sigma_fT = std(mismatched_frequency_steps); % Mismatched
varactors' tunning step standard deviation.


INIS_DDSM = 1; % switching between INIS DDSM and
conventional MASH DDSM

%====== MASH 1-1-1 MODEL
for n = order+2:samples

   %====================== First EFM1 stage
   s1(n) = e1(n-1); %Second input
   v1(n) = x(n) + s1(n)+ dither(n,2); % Dither is added to
the input signal

   if (v1(n) >= M) % Quantizer
     y1(n)= 1;
   else
    y1(n)= 0;
   end
   e1(n) = v1(n)- M*y1(n); % Error signal
```

```
    %======================== Second EFM1 stage
    if (INIS_DDSM)
        s2(n) =  e2(n-2) + e2(n-3);
    else
        s2(n) = e2(n-1);
    end

    v2(n) =  e1(n) + s2(n);

    if (v2(n) >= 2*M)
      y2(n)= 2;
    elseif (v2(n) >= M)
     y2(n)= 1;
    else
     y2(n)= 0;
    end
    e2(n) = v2(n)- M*y2(n);
    %=========================
    % BIT ROTATOR:
    rotator_enabled = 0;
    if (rotator_enabled)
      if (dither(n,2) == 1)
        temp = delta_f1;
        delta_f1 = delta_f2;
        delta_f2 = delta_f3;
        delta_f3 = temp;

      end
    end
    %=========================

  y(n) = delta_f1_nom * y1(n) + delta_f2_nom * y2(n) -
delta_f3_nom * y2(n-1); % DCO output frequncy
  % y1 (C0), y2 (C1), y (ftf)
  y_distorted(n) = delta_f1 * y1(n) + delta_f2 * y2(n) -
delta_f3 * y2(n-1); % DCO distorted output frequency
  % y1 (C0), y2 (C1), y (ftf)

end


%=======================
y_mean = mean(y); % mean vlaue of the output
```

```
y_zero_mean = y - y_mean; % mean normalization of the
output values
y_acc = cumsum(y_zero_mean); % cumulative sum of the output
frequnecies to obtain the phase
y_acc = y_acc.*2*pi./(f_sigma_delta); % converting to
discrete
%=========================
y_distorted_mean = mean(y_distorted);
y_distorted_zero_mean = y_distorted - y_distorted_mean;
y_NL  = cumsum(y_distorted_zero_mean);
y_NL = y_NL.*2*pi./(f_sigma_delta);
%=========================
a3 = 0.5;
a2 = 0.1;
%y_NL  =  y_acc + a2*(y_acc).^2 + a3*(y_acc).^3;
%-==========================
% Define a Hanning window
window=hann(length(y));
% Calculate the PSD using periodogram or pwelch
%[Pyy, w] = periodogram(y,window,samples);
%[Py_acc, w] = periodogram(y_acc,window,samples);
%Py_NL = periodogram(y_NL,window,samples);
[Py_acc, w] = pwelch(y_acc, 10000, 1);
Py_NL = pwelch(y_NL, 10000, 1);
%=============================================
% Consider sigma delta dithering and calculate the PSD
using the linear prediction
% Sigma Delta Dithering Noise
%PSD_predicted=
dither_contribution+variance*(2*sin(w/2)).^2;
PSD_predicted= variance * 4*pi^2/f_sigma_delta^2
*(2*sin(w/(2))).^4./(w).^2;
%=============================================================
=
% Mismatch Randomization noise
mismatch_noise_randomization =
sigma_fT^2/(2*f_sigma_delta);
mismatch_noise_randomization =  4*pi^2/f_sigma_delta *
mismatch_noise_randomization./w.^2;
%=============================================================
=
% LSB Dithering Noise
dithering_noise = (delta_fT)^2 / (2^18 * f_sigma_delta);
dithering_noise =  dithering_noise./w.^2;
```

```
dithering_noise = dithering_noise *4*pi^2/f_sigma_delta^2;
%=========================================================
==
total_noise = dithering_noise + PSD_predicted +
mismatch_noise_randomization;
%=========================================================
=
% Plot output PSD and the PSD using the linear prediction
tick = -200: 10: -120;
%
figure;
semilogx(w/pi,10*log10(PSD_predicted),'--');
hold on
semilogx(w/pi,10*log10(dithering_noise),'-d',
'MarkerIndices',[10  100  1000 10000 20000]);
hold on
semilogx(w/pi,10*log10(mismatch_noise_randomization),'-s',
'MarkerIndices',[10  100  600  5000 20000]);
hold on
semilogx(w/pi,10*log10(total_noise),'black',
'MarkerIndices',[50  300  5000 100000]);
hold off
xlim ([4e-4, 1]);
ylim ([-190, -120]);
grid on
yticks (tick);
legend ('Delta-Sigma dithering noise','LSB randomization
noise','Unit mismatch randomization noise','Total dithering
noise');
%-============================================
%}
%
figure;
semilogx(w/pi,10*log10(Py_NL),'r');
hold on
semilogx(w/pi,10*log10(Py_acc),'b');
hold on

yticks (tick);
xlim ([4e-4, 1]);
ylim ([-200, -120]);
legend ('LSB dithering & Unit mismatch','LSB dithering');
%}
%=========================================
```

```
grid on
xlabel('Normalized Frequency (x\pi
rad/sample)','fontsize',14);
ylabel('Power/frequency(dB/rad/sample)','fontsize',14);
title ('DCO Phase Noise');
set(gca,'fontsize',14);
set(findobj(gca,'Type','line'),'LineWidth',2);
```

2) Valerio's Code (INIS DDSM)

```
% INIS_MASH_test_rev2.m
%
% 201013: Octave original by Yann Donnelly
% 201014: Translated to Matlab by MPK
% 220822: Adapted from mashwithnonlinearity.m by MPK
% 220823: Modified for capcitor mismatch by MPK
%
clear all;
Nsamples = 2^20; % number of simulation points

B = 8;           % number of bits in DDSM
M = 2^B;         % DDSM modulus
X = 1;           % DDSM input

dith = unidrnd(2,[Nsamples 1])-1; % dither signal
d0_enable = 1;          % zeroth order dither enable
d1_enable = 0;          % first order dither enable

epsilon = 0.1;            % fractional mismatch error

% coefficients of transfer function *INIS MASH*
delay = 6;
%
a1 = -1;
a2 = -1;
a3 = 0;
a4 = 1;
a5 = 0;
a6 = 0;
%}

%{
 coefficients of transfer function *MASH 1-1*
a1 = -2;
a2 = 1;
a3 = 0;
a4 = 0;
a5 = 0;
a6 = 0;
```

```
%}
% pre-define matrices for speed
v1 = zeros(Nsamples, 1);
e1 = zeros(Nsamples, 1);
s1 = zeros(Nsamples, 1);
y1 = zeros(Nsamples, 1);

ic1 = 1;
ic2 = 0;
ic3 = 0;

deltaf1 = 1;
deltaf2 = 1;
deltaf3 = 1;

deltaf1_nl = 1-epsilon;
deltaf2_nl = 1;
deltaf3_nl = 1+epsilon;

s1(1+delay) = ic1;
for n = 1+delay : 1 : Nsamples
    v1(n) = X+dith(n)*d0_enable+s1(n);
    y1(n) = floor(v1(n)/M);
    e1(n) = v1(n)-M*y1(n);
    s1(n+1) = -a1*e1(n)-a2*e1(n-1)-a3*e1(n-2)-a4*e1(n-3)-
a5*e1(n-4)-a6*e1(n-5);
    y(n) = y1(n);
    switch y(n)
        case -1
            ddsm(n) = 0;
            ddsm_nl(n) = 0;
        case 0
            ddsm(n) = deltaf1;
            ddsm_nl(n) = deltaf1_nl;
        case 1
            ddsm(n) = deltaf1 + deltaf2;
            ddsm_nl(n) = deltaf1_nl + deltaf2_nl;
        case 2
            ddsm(n) = deltaf1 + deltaf2 + deltaf3;
            ddsm_nl(n) = deltaf1_nl + deltaf2_nl +
deltaf3_nl;
    end
end
```

```
% to approximate error accumulation, subtract mean from
MASH output
% and accumulate to obtain y_acc
f_sigma_delta = 500e6;
y_acc = cumsum(ddsm - (mean(ddsm)));
y_acc = y_acc.*2*pi./(f_sigma_delta);
% to approximate effects of capacitor mismatch, apply
nonlinearity function
% to spur-free y -> produces spurs
y_NL = cumsum(ddsm_nl - (mean(ddsm_nl)));
y_NL = y_NL.*2*pi./(f_sigma_delta);

window=hann(length(y_acc));
%[Py_acc, w] = periodogram(y_acc,window,Nsamples);
%Py_NL = periodogram(y_NL,window,Nsamples);
[Py_acc, w] = pwelch(y_acc, 10000, 100);
Py_NL = pwelch(y_NL, 10000, 100);
tick = -190:10:-120;
figure;
semilogx(w/pi,10*log10(Py_NL),'r', 'LineWidth',1);
hold on
semilogx(w/pi,10*log10(Py_acc),'b', 'LineWidth', 1);
hold off
grid on;
yticks (tick);
xlim ([4e-4, 1]);
ylim ([-190, -120]);
legend ('LSB dithering & Unit mismatch', 'LSB dithering' );
xlabel('Normalized Frequency (x\pi
rad/sample)','fontsize',14);
ylabel('Power/frequency(dB/rad/sample)','fontsize',14);
title ('DCO Phase Noise');
%}

%{
 plot y_acc (no spurs), then the nonlinearity, then y_nl
(strong spur)
figure(1)
P1=abs(fft(yacc)); % could also use Welch methods
semilogx((1:length(P1))/length(P1),20*log10(P1)-
10*log10(length(P1)));
title('y_{acc}')
xlabel('Normalized frequency (Hz)')
```

```
% figure(2)
% plotx=linspace(min(yacc),max(yacc),1001);
% plot(plotx,(plotx+0.01*plotx.^2),'r')
% title('Chosen nonlinearity: N(x)=1+0.01*x^2')
% xlabel('x')
% ylabel('N(x)')
% xlim([min(plotx),max(plotx)])

figure(3)
P2=abs(fft(ynl)); % could also use Welch methods
semilogx((1:length(P2))/length(P2),20*log10(P2)-
10*log10(length(P2)));
title('y_{nl}')
xlabel('Normalized frequency (Hz)')

% fix axis scales, optional
Pymin = min([min(20*log10(P1)),min(20*log10(P2))])-
10*log10(length(P1));
Pymax = max([max(20*log10(P1)),max(20*log10(P2))])-
10*log10(length(P1));
figure(1)
xlim([10^(-1*floor(log10(Nsamples))),0.5]) % reduce x axis
scale
ylim([Pymin-10,Pymax+10]) % equalize y axis scales
figure(3)
xlim([10^(-1*floor(log10(Nsamples))),0.5]) % reduce x axis
scale
ylim([Pymin-10,Pymax+10]) % equalize y axis scales
%}
```