Behavioral Time Domain Modeling of RF Phase-Locked Loops

A thesis submitted in partial fulfillment of the requirements of the award of the degree of

Bachelor of Technology (Honours)

in

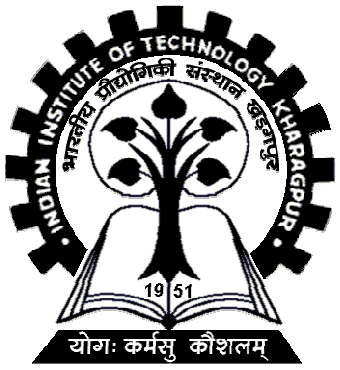
Electronics and Electrical Communication Engineering

under the guidance of

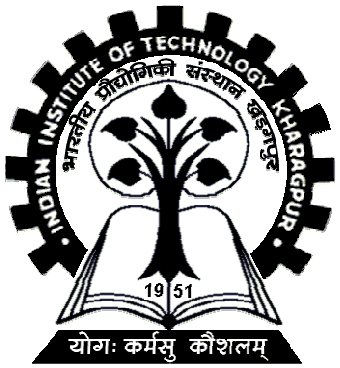
Prof. Tarun K. Bhattacharyya

by

Sushil Subramanian
(05EC1030)



Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
May 2009



Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
May 2009

Certificate

This is to certify that this thesis entitled "Behavioral Time Domain Modeling of RF Phase-Locked Loops" submitted by Sushil Subramanian to the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, in partial fulfillment of the requirements for the award of Bachelor of Technology (Honours) in 2009, is an authentic record of the work carried out by Sushil Subramanian under my guidance and supervision.

In my opinion, this work fulfills the requirement for which it has been submitted. This thesis has not been submitted to any other university or institution for any degree or diploma.

Prof. Tarun K. Bhattacharyya

Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog to Digital Conversion (Converter)
BER	Bit Error Rate
CAD	Computer Aided Design
CMOS	Complementary Metal-Oxide-Semiconductor
CP	Charge Pump
DFF	D-Type Flip Flop
DVCO	Digital Voltage Controlled Oscillator
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FS	Frequency Synthesizer
GSM	Global System for Mobile Communications
IIR	Infinite Impulse Response
ISM	Industrial, Scientific and Medical
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
PFD	Phase Frequency Detector
PLL	Phase-Locked Loop
PSD	Power Spectral Density
Q	Quality Factor
RF	Radio Frequency
SCL	Source Coupled Logic
SNR	Signal to Noise Ratio
SOC	System on Chip
VCO	Voltage Controlled Oscillator
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale Integrated Circuits

Chapter 1

Introduction

1.1 Motivation

They say that given a set of points, one can always think of a curve to fit them. However, a good mathematical model should be capable of reacting well to changes and randomness in those set of points. Nature as we know it is quite random and deterministic modeling is a very difficult task. Often, we resort to numerical methods and curve fitting techniques to make the model accurate.

Of particular interest in modeling are nonlinear circuits such as phase-locked loops (PLLs) which have wide applications in clock generation and communications. The design of PLLs and other communication electronics in general, have the requirement to catch up to current standards of Moore's Law not only in terms of the size, but also in terms of power and data rates. Consequently, criteria such as Signal to Noise Ratio (SNR), power and area efficiency, sensitivity and reaction time of communication electronics have been pushed to the very limits of analysis and design. Thus, new methods are constantly being developed to help designers analyze their circuits better and take into account non-idealities that arise with operation at lower power, harsher environments and stringent wireless communication conditions.

For the design of PLLs in particular, since the circuit is highly nonlinear, the analysis has been restricted to the frequency domain. This is because many radio frequency (RF) PLL phenomena such as transients before locking and charging of the charge pump occur within short periods of time, and simulations that provide time domain outputs have to sample and evaluate waveforms over small timesteps. This directly increases the simulation time and hence the design cycle time. However, a time domain approach in modeling will lessen the dependency of the model to the nature of the circuit elements and directly map the circuit on a system level to a deterministic set of equations or probabilities. Since the PLL is a non-linear control system, dynamical evaluation will certainly provide more insight than a $s-\text{domain}$ or $z-\text{domain}$ approach.

Therefore, recent approaches have been directed towards modeling the PLL in the time domain with novel methods such as event-driven simulation [1, 2], which reduce the time of simulation. With this approach, the designer can further focus on making the model more realistic and robust,

by including more useful and detrimental phenomena that occur while keeping up with current technology trends.

This thesis is an exploration of the time domain modeling and simulation of PLLs in general, with an initial emphasis on the system level architecture and the simulation environment. The thesis also focuses also on the non-ideality analysis of the PLL in various simulation environments, such as phase noise and timestamp errors. Using Simulink as a basis environment, the author develops mathematical models for the behavior of various circuit blocks and systems in the discrete and the event-driven simulation environment. Contrary to conventional approach, the author extends the focus on frequency synthesizers, where simulation is critical as analog and digital blocks occupy the environment. With this basic but comprehensive understanding of the simulation environment, the author develops specific mathematical models for the phase noise and divider feedback systems that help in the inclusion of the non-idealities of the PLL circuit on a system level. The models interesting show how easily the circuit and the simulation environment can complement each other, while matching actual observed data of characteristic parameters of the PLL such as settling time, phase noise performance .

Before starting with specific contributions of the thesis and the theoretical background, it is the author's duty to give brief introductions to the Phase Locked-Loop and Time Domain Modeling in the PLL Simulation scenario.

1.2 Phase-Locked Loops: A Control System

Many circuits require multiple stable clocks or perfect high frequency sinusoids for their operation. Typical examples are distributed clock System-on-Chip (SOC) designs and multiple-stage down conversion and up conversion in radios. A PLL is a circuit that generates such a stable frequency using a possibly unstable reference clock. The PLL uses a phase difference detector between the input reference and the output, low passes this error signal and feeds into a noiseless Voltage Controlled Oscillator (VCO). This VCO output is fed back into the phase difference detector and the process of correction continues.

As a phase *difference* is detected, the PLL forms a negative feedback system and thus dramatically optimizes and negates any error in the input frequency. Additionally, a feedback divider can be used (as shown in Fig. 1.1) to obtain an output a much higher output frequency. This makes the PLL useful for RF applications, which will be more of the focus of this thesis. If this divider is made *programmable* a range of frequencies can then be obtained which then makes the PLL a *frequency synthesizer* (FS), a very useful application where baseband signals on the divider can actually shift frequencies and use various portions of the spectrum.

From a historical perspective, the first PLL was invented as early as the first receiver itself, by Edwin Armstrong in 1932. However, with integrated technology in the 1960's the PLL gained a lot more applications and robustness and performance became key issues. One can note that deficiencies in the individual components of the PLL such as mismatch in the Phase detector and phase noise in the VCO and Divider cannot be removed by the corrective nature of the negative feedback as they form a part of the transfer function between the input and the output. Therefore, focus also shifted to improving the individual components and introducing corrective measures in the PLL transfer function. This led to extensive research in the phase noise of VCOs and dividers from the 60's till recently [3, 4, 5], while simultaneously many circuit level breakthroughs were made. All in all, the PLL and modeling the PLL still form an intriguing area of research as their performance directly implies good data at the baseband.

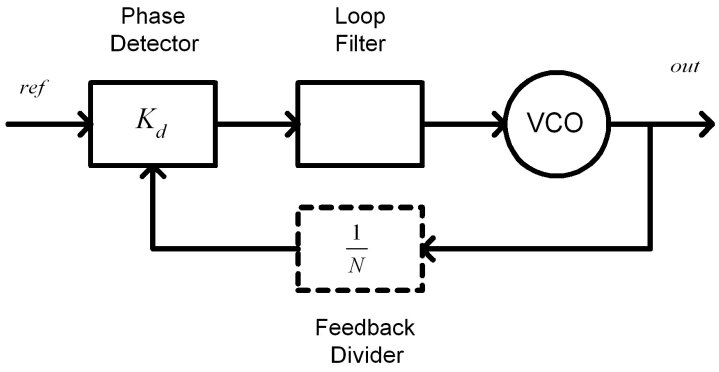


Fig. 1.1 Basic PLL Diagram

1.3 Time Domain Modeling: The Behavioral Approach

The behavioral approach to modeling is to emphasize on a black box, where the given model emulates a system by producing an *output trajectory* using *input state information* and an *input trajectory* [6]. In simpler words, a behavioral model extracts comprehensive information of the system and predicts all future outcomes based on past outcomes. This is a dynamical method of simulation, that is in contrary to frequency domain analysis where, spectral properties are indicative of a near infinite time domain sequences. It can be seen that this kind of simulation gives a clearer picture of a non-linear system. Consider for example, Gaussian Noise and Chaotic signals. Both of them are spectrally wideband and constant power. However, the behavioral evolution of the signals can classify them.

The nature of the input states and processing of the input trajectory classifies the simulation approach:

a) *Control Flow:* The simulation of each step of the model is performed without any external knowledge of the environment and the inputs are described at the time of sampling, i.e. $T_{\text{out}}(n+1) = f(T_{\text{in}}(n), t_{\text{model}})$; where T is the trajectory and t_{model} is the time step.

b) *Data Flow:* The simulation of each step of the model is performed without any external knowledge of the environment and the inputs are described beforehand, i.e. $T_{\text{out}}(n+1) = f(T_{\text{database}}, t_{\text{model}})$.

c) *State Machine:* The simulation of each step of the model is performed with an external knowledge of the environment and the inputs are described at the time of sampling, i.e. $T_{\text{out}}(n+1) = f(T_{\text{in}}(n), t_{\text{event}})$.

For the purpose of PLL simulation, cases a) and c) are most important as we are interested in inputs being available in *real time* which correspond to them. However, the essential difference between a) and c) is that a) corresponds to *discrete simulation*, whereas c) corresponds to *event-driven simulation*. Most models if simulated in control flow have a discrete clock that emulates t_{model}. This parameter directly relates to the simulation time. On the contrary, in event driven simulation, the simulation time is variable as t_{event} is variable and highly dynamic in itself.

Simulation and Modeling are integral to any design cycle, and thus the corresponding models and simulation environment working hand in hand determine the robustness of the model and thus the success of all future design. Clear tradeoffs between accuracy and time are obtained.

1.4 Contributions of this Thesis

PLLs and frequency synthesizers are such important components of any electronic circuit that an uncountable number of models exist to take in account oscillator noise, phase detector performance, device deficiencies, process errors and many other factors. Models are always first verified and then used as future simulation environments.

Most such models are restricted to individual portions of the frequency synthesizer. Thus, this thesis examines existent models in terms of the overall effect on architecture and noise in the final output of the PLL or FS and not the individual blocks such as phase detector and VCO. This enables to determine the overall effect of the individual noise models on the dynamics of the final FS output.

Further, models are sometimes simulation environment specific, such as modeling using VHDL, Simulink, Spice etc. In this thesis the author develops a mathematical framework, with which it is possible to analyze RF circuits and mixed-signal circuits in any simulation environment. The author also show the correlation between the nature of the circuit, the sampling requirements of waveforms and the interpolation of the simulation environment to be integral to the model being developed and apply the same to measuring phase noise, feedback delays in high division dividers and divider phase noise. A sampling scheme is developed for *mixed-signal circuits* in *mixed signal discrete and event-driven* simulation environments.

In essence, this thesis tries to combine four concepts:

1) Accurate circuit and system level models of the individual blocks of the PLL and FS,
2) The system level effect of one PLL block on others and the final PLL output,
3) Behavioral time domain simulation and its advantages and
4) Interfacing of the model and the simulation environment.

On the theoretical level, incorporating various divider architectures and corresponding phase noise in PLL simulations is a key contribution of this thesis.

The use of all these concepts is important in creating faster and more user friendly simulation environments for RF and mixed-signal circuits. The applications of such an approach are plenty in VLSI CAD and modeling theory of RF circuits in general.

This thesis is broken down into the following chapters. Chapter 2 discusses the various aspects of the simulation environment and the issues and the problems faced in high frequency simulation. Chapter 3 presents the complete event-driven and discrete simulation of the PLL and the problems and solutions to certain problems faced thereon. Chapter 4 discusses the extremely important topic of noise modeling and introduces various models of noise for the event-driven simulation environment. Chapter 5 concludes the thesis with some comments and future directions.

Chapter 2

High Frequency System Simulation

2.1 Introduction

In this chapter, the author will describe the simulation environment, and explain the behavior of the environment when a mixed-signal circuit model is being simulated. This forms an important part of combining the concepts of modeling the circuit and the simulation software in use, and is integral in creating faster and accurate simulation. The simulation can be performed in a *discrete* manner as decscribed in case a) in Section 1.3 or in an *event-driven* manner as described in case c) in Section 1.4. However, mixed-signal circuits impose a scenario where the simulation is a combined *discrete* and *event-driven* if chosen to be performed in that manner.

The crux of the problem is to formulate a universal theory that accounts for the reduction in the simulation time from simulating event-driven while allowing accommodation of the discrete (different from *digital* blocks, that are in the circuit sense) blocks of the mixed-signal model.

2.2 Simulation Environments and Sampling

In this section, we consider sampling in simulation environments. Most simulation environments are defined by the way they sample and interpolate data. However, the sampling may be performed in a non-uniform manner, as in event-driven simulations. Since the problem in hand is time-domain simulation, the trajectory model (from Sec. 1.3) is assumed and thus the three steps for simulation are: *sampling*, *evaluation* and *interpolation*. Consider the case study of the Simulink programming language.

2.2.1 Reduced Model - Sampling, Evaluation and Interpolation

Visual programming languages such as Simulink MATLAB [7] provide the user with the ease of using blocks and arrows to define systems and relations between systems respectively. They are distinct from programming languages such as VHDL, as the underlying timestamp and sampling

programming is done automatically. The corresponding simulator uses the ode45 Runge-Kutta Solver with automatic variable time stamps, if chosen by the user to be event-driven. The user is also given an option to specify the simulation timestep duration and the minimum and maximum step sizes. Since event-driven simulations uses variable time-stamps for optimal sampling of the present states in the model and the input [2], it is best to allow automatic selection of the above parameters and use the ode45 solver.

Simulink also adopts distinct *digital* or *continuous* blocks, which are simulated with different timestamp sequences, and the corresponding processes run separately in the computer kernel. Thus, in mixed signal modeling, any block involving digital to analog state conversion or vice versa involves numerical error simply due to varied timestamps and quantization. Models have been proposed to avoid these errors by using problem specific local samplers for each of the block, such as the case in [8], where a Fractional-N Frequency Synthesizer is broken down into individual blocks with different sampling schemes. Other cases, such as the popular simulator Spice and VHDL based mixed signal simulators involve backtracking of timestamps [9]. Since this is not possible in Simulink, numerical noise does arise and is more significant in shifting from the discrete simulation to analog simulation rather than the actual Analog to Digital Conversion (ADC). A typical warning in the MATLAB command line is shown in Fig. 2.

Henceforth, we shall consider all experiments to be performed in Simulink although the concepts may be extended to all environments as shown in [9]. In general, however, a behavioral simulation environment is described as per Fig. 2.2 [6]. The *source system* is the database based on physical observations, which is then adapted to the *model*, an approximation in the form of trajectories and input states. This model then interacts with the *simulator* to produce the output trajectory. In the case of real-time behavioral simulation the trajectory calculations reduce to

$$x_i(n+1) = f\left(x_1(n), x_2(n), \ldots\right) \tag{2.1}$$

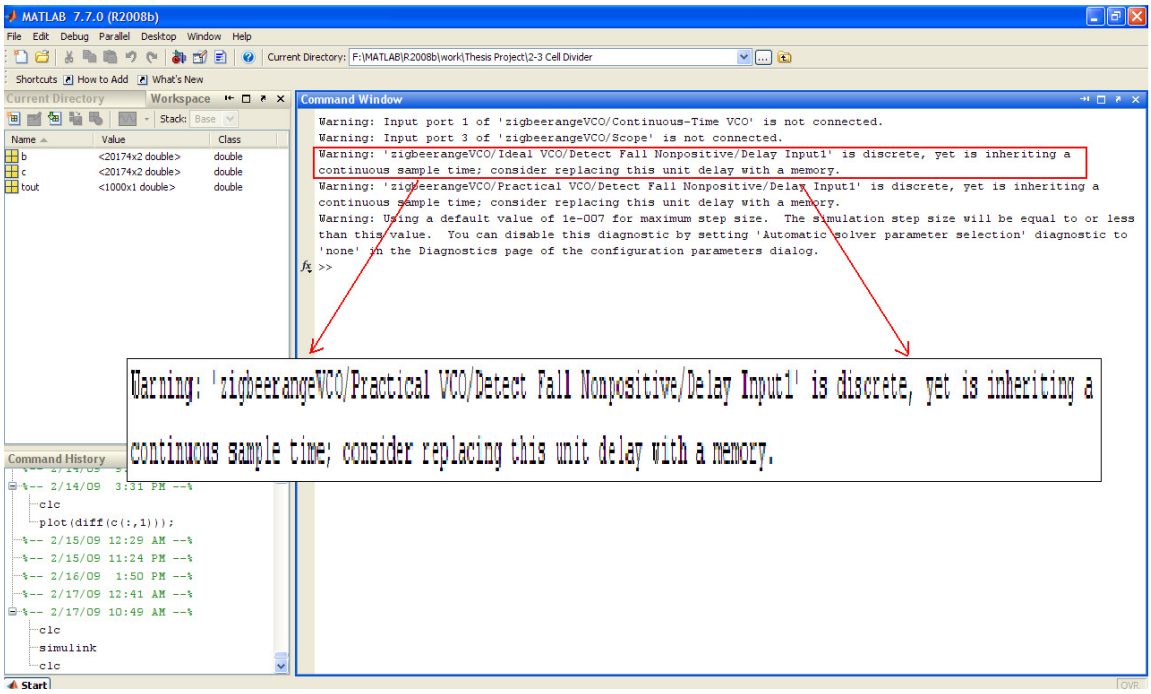


Fig. 2.1 Mixed Signal simulation warning in MATLAB

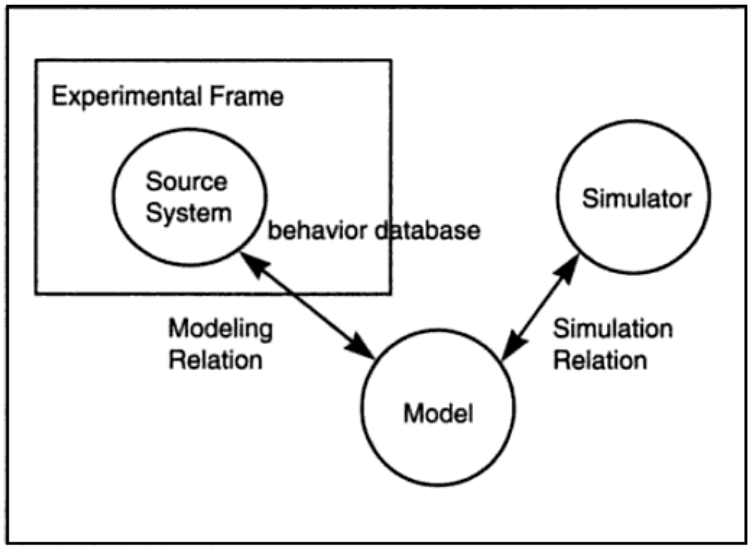


Fig. 2.2 Fundamental relation between the model and the simulation

where, the $x_i(n)$ is the state information of all the states *and* sources at a particular sample n and the function f evaluates the next point on the time axis for that state variable. In the big picture however, the sources are the input trajectories and the states are formed by specific models of the physical, biological etc. system.

The above describes the *evaluation*. The *interpolation* is a polynomial connecting $x_i(n)$ and $x_i(n+1),$ which is normally linear, as it is causal. The linear interpolation is most convenient for simulation environments as the simulation can be performed real-time and thus can be stopped at any point. If the interpolation is Shannon-Whittaker, perfect reconstruction is assured, however, real-time processing or simulation is not possible. Simulink uses linear interpolation.

The *sampling* can be uniform or non-uniform according to whether the simulation is discrete or event-driven. In the case of event-driven simulation $t_{n+1} - t_n = g\left(x_1(n), x_2(n), \ldots\right)$, whereas in the case of discrete simulation, $t_{n+1} - t_n = T$.

2.2.2 Model Type and Solver

In this subsection, the author will emphasize on the type of solvers used and the relation to the model type. All systems (especially those in circuits), can be classified into digital or analog; however observation and all possible processing can only occur at discrete intervals. However, a solver in the simulation environment may useful event-driven sampling (at non-uniform intervals) or uniform sampling (at regular intervals) to do the same processing.

Therefore, solvers may be classified into *discrete solvers* or *continuous time solvers*, as they evaluate with continuous events rather than the event as determined by the sampling clock. In Simulink, therefore, there exists a *discrete solver* and a *continuous solver*.

Let an input trajectory $T(t)$ be observed at rate $1/T_{\text{obs.}}$ and sampled at the rate $1/kT_{\text{obs.}}$. Due to linear interpolation, at most $k-1$ points are in error per cycle besides numerical integration error. Consider the case of event driven simulation where, events are occurring at random intervals, such that the expectation value of the sampling time period $t_{\text{cont.}}$ is $E(t_{\text{cont.}})$. Over a long period

of time as event accumulate in the events queue, the average frequency of sampling reduces to $1/E(t_{\text{cont.}})$, where E indicates the expected number. Therefore, at least $(E(t_{\text{cont.}})/T_{\text{obs.}}) - 1$ samples are in error due to linear interpolation besides numerical integration, per cycle.

Assume the total error in simulation in linear interpolation is e_T. The time equivalent corresponding error is:

$$e_t = \frac{e_T}{\left(\frac{dT}{dt}\right)} \tag{2.2}$$

This error will henceforth be termed as t_{ADC}. Thus, as far as interpolation is concerned, we can elucidate the properties of the two kinds of simulation as follows. If the value of k is decreased, discrete simulation will have lesser and lesser erroneous points. However, since the value of $E(t_{\text{cont.}})$ is assumed constant over $t \to \infty$ as events take place in a pseudo-random manner, this implies that an event-driven simulation cannot be rid of interpolation error unless the system itself is changed.

Note, however, that since in an event-driven simulation all events are guaranteed to be queued, the probability of error of an important event to be missed is nil. However, the error range for the discrete simulation can be up to $(k-1)T_{\text{obs.}}$. This error we shall call $t_{\text{timestamps}}$ and it essentially caused due to events not being successfully tracked by a discrete simulator. This kind of error is absent in event-driven simulation. A third kind of error occurs purely in computation. If we use the Range-Kutta ode45 method to compute the trajectories, then the error due to computation alone and the desired *oracle* response is the third kind of error called $t_{\text{numerical}}$.

To summarize, we firstly showed that a simulation is essentially the evaluation of trajectories using a model and a set of states or conditions. Since the simulation is *real time* and *behavioral*, the environment action reduces to three functions: *sampling*, *evaluation* and *interpolation*. Each of these functions correspondingly has its own error, defined by: $t_{\text{timestamps}}$, $t_{\text{numerical}}$, and t_{ADC} respectively.

Consider the example of the evaluation of jitter of a clock. Suppose the physical model evaluates that a particular point in time t_{jitter} is the net jitter in the clock. When simulated in an environment, the total jitter that shows in the output trajectory is given by:

$$t_{\text{total}} = t_{\text{jitter}} + t_{\text{ADC}} + t_{\text{numerical}} + t_{\text{timestamps}} \tag{2.3}$$

This model can be extended to any kind of measurement on any kind of behavioral real-time time domain simulations in any environment. The author's analysis and results henceforth are mainly in MATLAB, but they can be extended to any simulation language such as VHDL and C++ [8, 9].

2.2.3 Introducing Mixed-Mode Simulation

The PLL and the FS are generally mixed-signal designs. That is, both digital and analog circuit architectures work hand in hand. When such a mixed-signal design is simulated in an environment using continuous event-driven simulation, since the states of the digital circuits most often change with their internal clocks, the simulation reduces to a discrete simulation. However, events still need to be tracked for the rest of the circuit that is analog in the circuit domain, and hence event-driven simulation is also observed.

Hence what happens is a background discrete simulation with a tracker that tracks events and puts them in a queue along with the periodic discrete samples. This simulation we shall term as a

mixed-mode simulation. In such a simulation, by judiciously varying certain parameters such as the clock frequency of the fastest clock in the system, we can optimize simulation time. The corresponding theory will be presented in the subsequent sections.

2.3 RF VCO Simulation: Experiments and Results

We shall now consider the simulation of high frequency systems in mixed-mode simulation environments. A common system of high frequency is the Voltage Controlled Oscillator (VCO). The simulation environment shown in Fig. 2.2 for the mixed mode simulation can now be modified to Fig. 2.3.

The next state information is updated by the solver which also provides either the continuous sampling time steps $t_{(\text{cont.})}(l)$ (based on the event queues) or the discrete sampling time steps, $t_{(\text{disc.})}(n)$ (based on the discrete sampling). Note that the index n is different from l. Consider M blocks in the model with K of them analog and combinatorial digital (e.g. integrators, addition subtraction, gates etc.) and the rest sequential digital (e.g. clocks, flip-flops etc.). The digital block will have different state change time periods, which can be represented as $(1/f_i)$, $1 \leq i \leq (M-K)$. Let $f_{\max} = \max_i(f_i)$. Therefore, to simulate the changing states of the fastest digital block:

$$t_{(\text{disc.})}(n) = \left(\frac{1}{2}\right) \frac{1}{\max_i(f_i)} = 1/2f_{\max} = t_{(\text{disc.})} \qquad \forall\, n \tag{2.4}$$

Note that $t_{(\text{disc.})}(n)$ is constant and can be used by the solver at absolute time $n \cdot \left(t_{(\text{disc.})}(n)\right)$, $n = 1, 2, 3 \ldots$, as the state changes at these points. However, $t_{(\text{cont.})}(l)$ is a case of non-uniform sampling, which can occur at pseudorandom time intervals depending on the state of the analog blocks in the model and the solver algorithm, which is event-driven. $t_{(\text{cont.})}(l)$ and $t_{(\text{disc.})}(n)$ alternate thereby indicating changes in the kernel. Further, $t_{(\text{cont.})}(l)$ becomes increasingly insignificant as the value of $t_{(\text{disc.})}(n)$ decreases due to more frequent changes in the state of digital block with $f_{\max}$.

Plots of timestep vectors with $f_{\max} = 5 \times 10^9, 5 \times 10^{10}$ and 5×10^{11} Hz (Fig. 2.4), show the contrast in the timestep variations for a VCO. The experimental setup used for these plots is shown in Fig. 2.5. In the setup, the simulation of a continuous time VCO is considered. The continuous VCO itself in an analog block, however, the discrete time step $f_{\max}$ is modeled as a redundant clock in the model, enabling us to observe changes in the timestep with time, with increasing $f_{\max}$. As expected, the time taken to simulate increases as $f_{\max}$ increases. Experimental results in Fig. 2.4 also shows that the maximum step size is equal to $t_{(\text{disc.})}$. These plots confirm that alternate discrete and event-driven simulations occur in Simulink, a phenomenon shown to happen in other simulators such as VHDL and Spice [9].

A solution to obtain a clear tradeoff between accuracy and simulation time is to have discrete sampling only throughout; however, this defeats the purpose of event-driven simulation and consequent optimization of simulation time. Further, the nature of the results in Fig. 2.5 is qualitative and thus does not clearly show why the kernel changes occur from discrete to continuous event-driven simulation. From Fig. 2.3, we can deduce that the model architecture, i.e. the selection of the blocks and their interconnections, determines when the changes in the mode of simulation

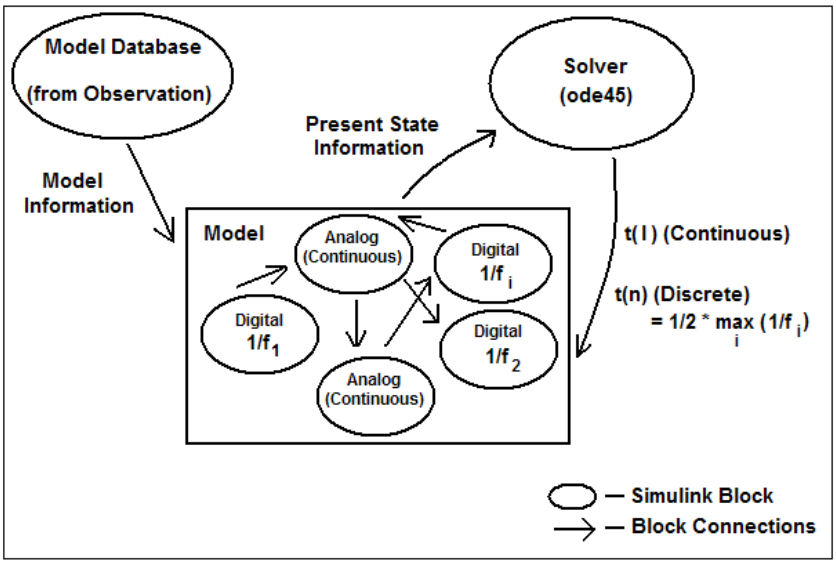


Fig. 2.3 Modified Mixed-Mode Simulation Environment

occur. This will be discussed subsequently, based on particular high-frequency truly mixed-signal Digital Voltage Controlled Oscillator (DVCO) model.

Consider the simulation of the continuous time VCO with output $v_{\text{out}} = \cos(2\pi f t)$, with a sampling scheme as described above, i.e. in the presence of digital blocks with a defined $f_{\max}$. One cycle of the simulation of this clock can be simulated discretely with $N_0 = 2\pi f t$ samples. For mixed signal simulation, T' time out of T is used for simulating discretely and the rest of the time is continuous event-driven simulation, where $T = 1/f$. The number of samples per cycle of VCO output is:

$$E(N) = 2T' f_{\max} + \frac{(T - T')}{E(t_{\text{cont.}})} \tag{2.5}$$

Plots of $E(N)$ with $f_{\max}$ are shown for the inbuilt Simulink continuous VCO in Fig. 2.6. With $f_{\max}$ increases, $T' \to T$ and hence $E(N) \to 2Tf_{\max}$, thus making the contribution of $E(t_{\text{cont.}})$ increasingly insignificant, which is verified by Fig. 2.6a. However note that, correspondingly with smaller step sizes, the simulation time also increases (Fig. 2.6b). Similarly, note that when smaller and smaller $f_{\max}$ is used, $T' \to 0$ hence making $E(N) \to T/E(t_{\text{cont.}})$ over a large time of simulation, as $E(t_{\text{cont.}})$ is pseudorandom and reaches constant value only then. Therefore, with high $f = 1/T$ (which is typical of RF circuits), operating in the completely discrete simulation domain is larger time steps. In Section 2.4, we will describe a DVCO model, and outline steps from which the essential characteristic parameters (such as phase noise, spectrum etc.) can be abstracted even with operation in the "mixed-mode simulation" range (see Fig. 4).

The models used to obtain plots in Fig. 2.6 consist of a single digital block to vary $f_{\max}$ and multiple analog blocks, like integrators, sum, memory etc. to create the mixed signal environment. Although not shown, these plots are qualitatively the same and have been verified for other types of mixed signal circuits modeled in MATLAB.

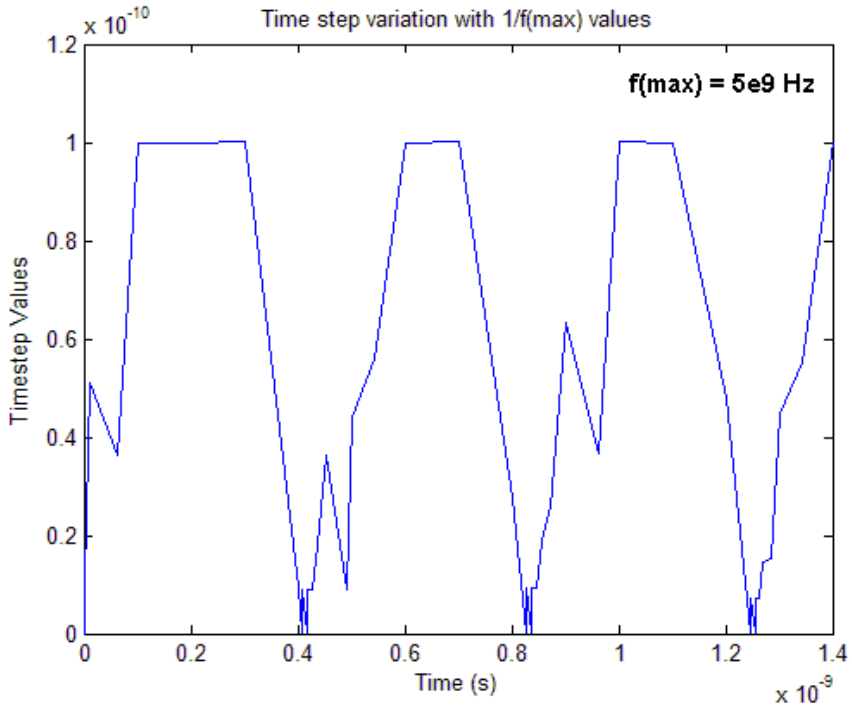


Fig. 2.4a. Simulation timestep variation $\left(f_{\max} = 5 \times 10^{9}\right)$

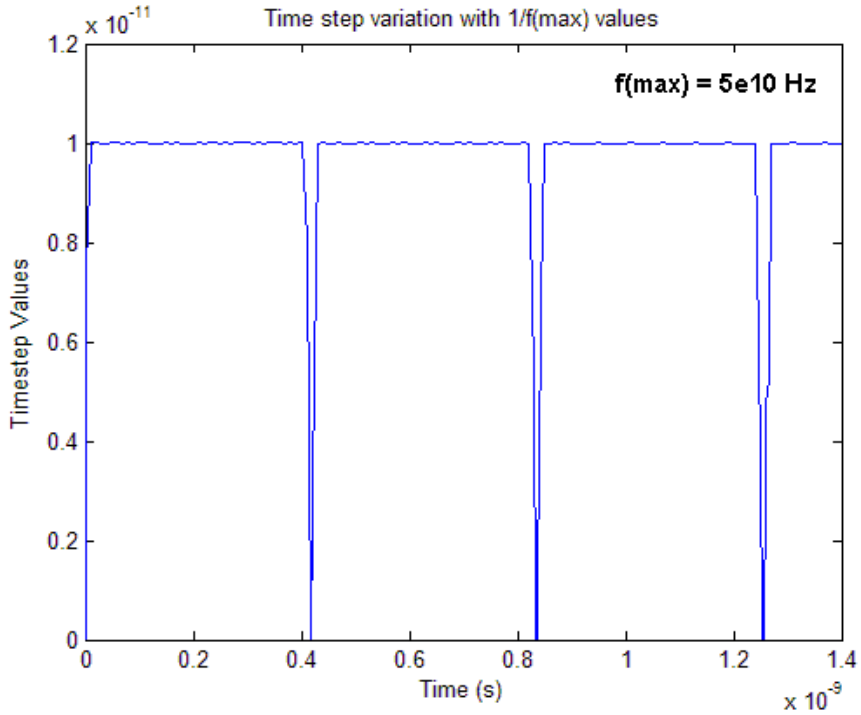


Fig. 2.4b. Simulation timestep variation $\left(f_{\max} = 5 \times 10^{10}\right)$

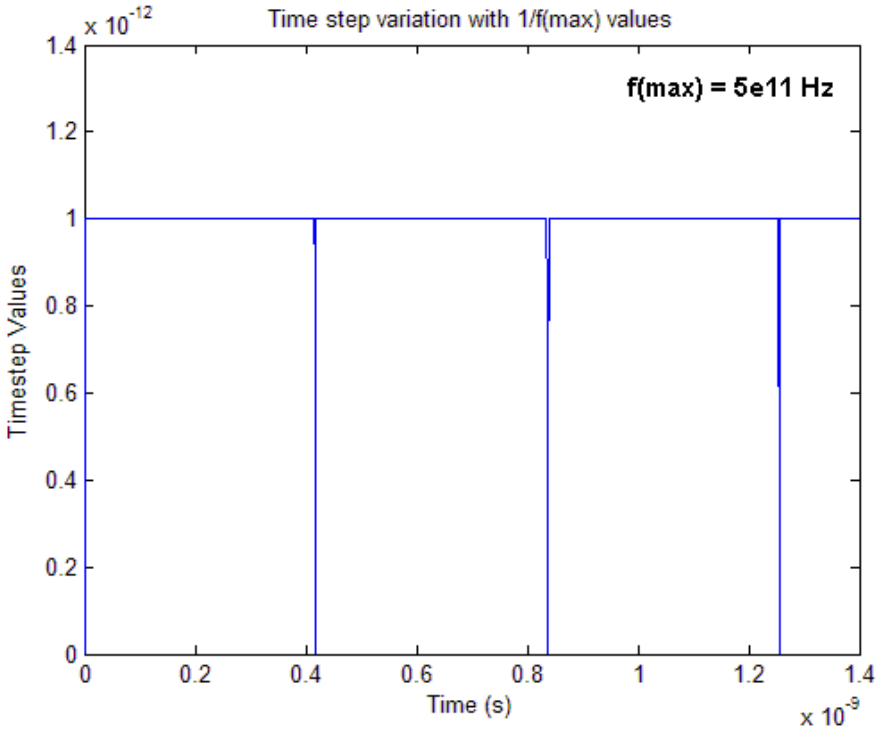


Fig. 2.4c. Simulation timestep variation $\left(f_{\max} = 5 \times 10^{11}\right)$

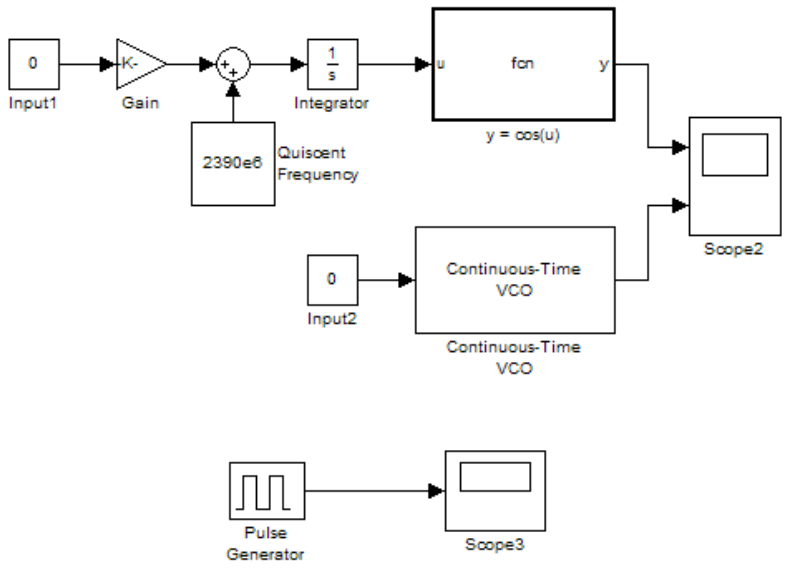


Fig. 2.5 Experimental Setup for timestamp evaluation

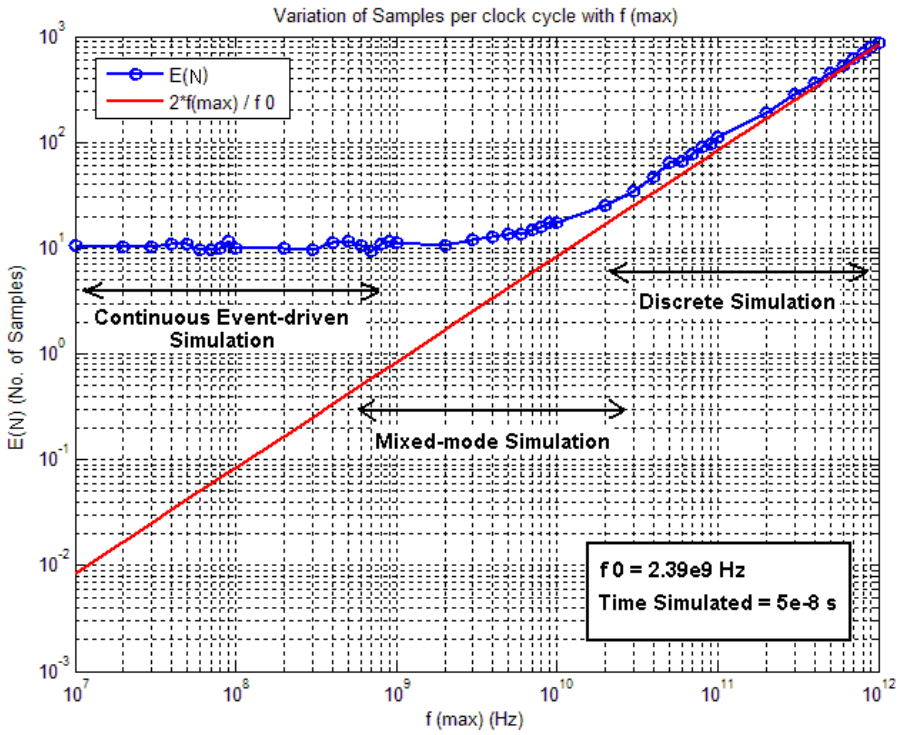


Fig. 2.6a. $E(N)$ variation with $f_{\max}$

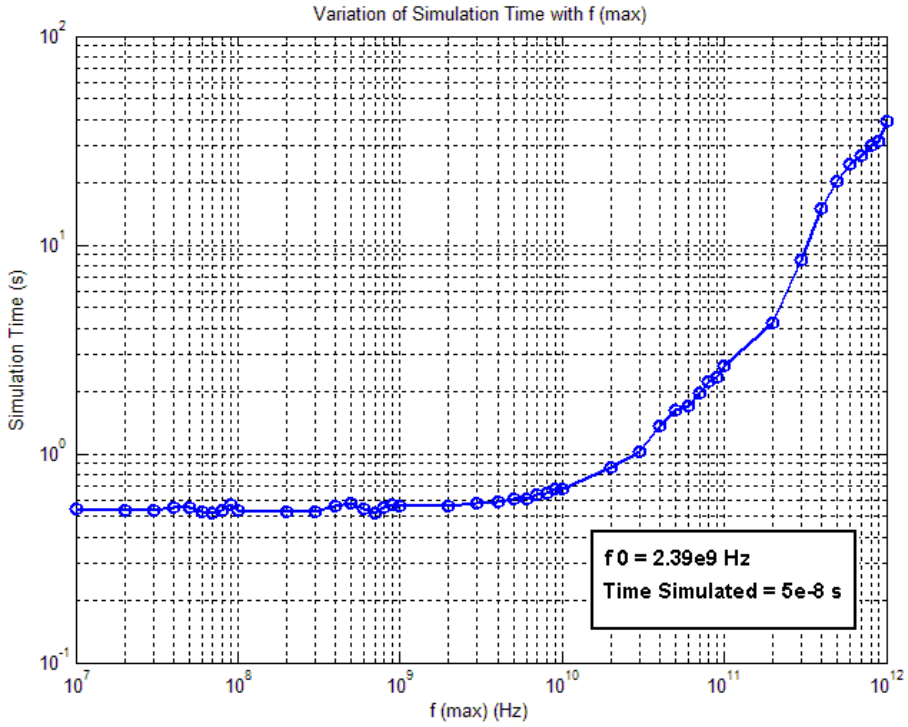


Fig. 2.6b. Simulation Time variation with $f_{\max}$

In the above simulations, the frequency of the continuous time VCO is $f = f_0 = 2.39 \times 10^9$ GHz. This frequency is chosen as it is a common frequency used in the ISM band for the Bluetooth, Zigbee, WiFi protocols. The plot of $E(N)$ approaches $2Tf_{\max}$ with increasing $f_{\max}$. Therefore, for $f_{\max} = O(f)$, the simulation approaches mixed mode nature. For $O(f_{\max}) << O(f)$, $E(N)$ is nearly constant as continuous event-driven time steps dominate and $T' \to 0$ and $E(N) \to T/E(t_{\text{cont.}})$ which is completely solver dependent.

The significance of the operation in the RF region is observed by viewing the simulation results of Fig. 2.6. If operated in IF or lower frequencies, the requirement from Nyquist-Shannon sampling theorem, the $f_{\max}$ need to be much smaller for accurate reconstruction and simulation. Therefore, the characteristic curve in Fig. 2.6a observed at much smaller $E(N)$. The overall effect can be imagined as a "slide down" of the characteristic curve of $E(N)$ over the reference straight line: $E(N) = (2/f_0)\, f_{\max}$. When this curve slides down the reference line, similarly, the simulation time curve also slides down, thus significantly reducing the simulation time at IF. Therefore, discrete simulation at IF will give the same $E(N)$ as mixed-mode simulation in RF. Hence, one can see the significance of mixed-mode simulation in RF VCOs. This is also the reason why one usually experiences long simulation times when operating in the discrete simulation mode with RF circuits. Most commercial simulators such as Cadence operate in this region.

2.4 DVCO Model Architecture

For a VCO with output $v_{\text{out}} = \cos(2\pi f t)$, perfect reconstruction is possible with the Whittaker-Shannon Interpolation Formula [10] with a sampling rate of $f_s = 2f$. However, in Simulink processing is in the time-domain with respect an absolute time scale. The interpolation is linear, and with $E(N)$ being very less in mixed-mode simulation, phase error is evidently large. Fig. 2.7 shows the simulation of the Continuous VCO with $f_{\max} = 5 \times 10^9$ Hz and $f = 2.39 \times 10^9$Hz. Notice a phase error of nearly 90° at the zero crossings only due to linear interpolation which is effectively about $1/4f$ of t_{ADC}. Instead, we a DVCO model, where changes are digital and error can be limited to $t_{\text{timestamps}}$ and $t_{\text{numerical}}$ as derived in (2.3). For subsequent modeling, we use a DVCO model based on the one proposed in [1] (Fig. 2.8).

2.4.1 Block Level DVCO Model

Changes in a DVCO can be tracked as a discrete simulation and more importantly as a square wave clock, and hence completely eliminating t_{ADC}. Also, the local oscillator feeds voltage into the feedback divider and the mixer, for a receiver. In the feedback the edge only the edge is important for activating the various flip flops. In the mixer, the oscillator acts as a switching function [11], thereby making again, only the edge important. Thereby, modeling the VCO as a DVCO is acceptable in the circuit sense.

The frequency of the VCO is given as $f_{\text{out}} = K_V V_{\text{in}} + f_q$ where K_V is the VCO gain, V_{in} is the input voltage and f_q is the quiescent frequency. The VCO transfer function pole $1/s$ is modeled as a ramp function at the input as shown in Fig. 2.8. This converts the frequency into a time dependent phase. The phase is then fed into two relational operators with 0.5 duty cycle out of phase. The relational operators threshold the incoming ramp at a duty cycle of 0.5 and thus create the first clock pulse after the AND gate. This clock pulse is then used to update a memory block in

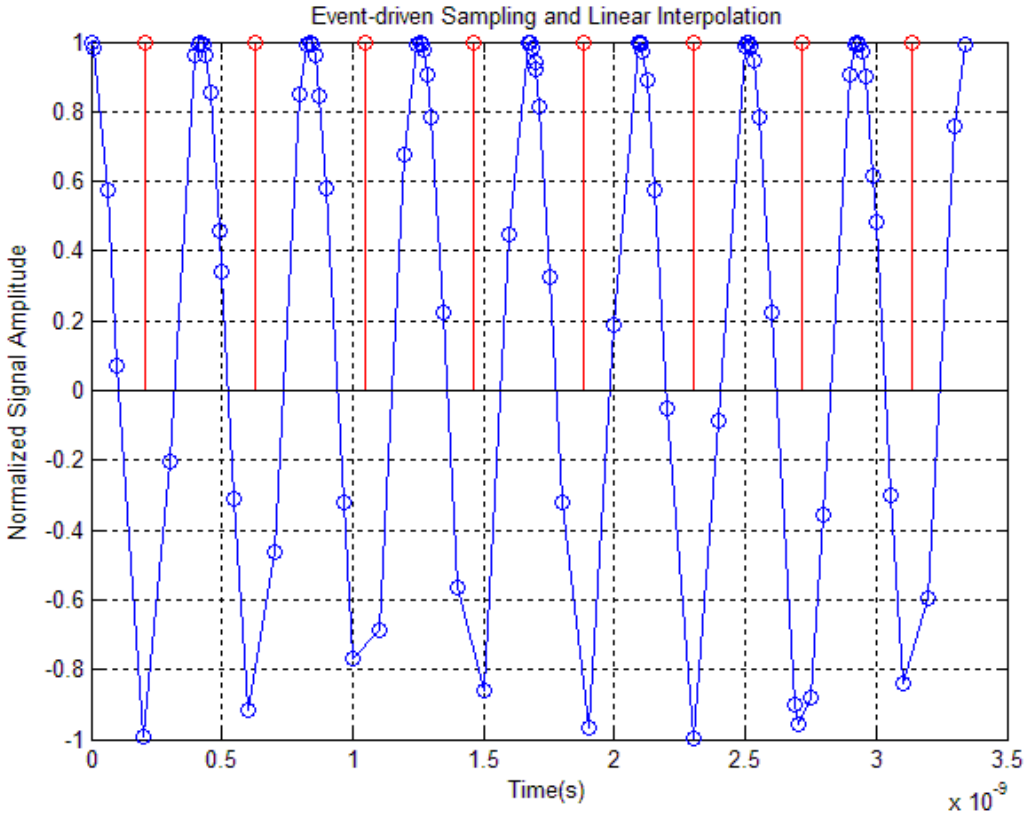


Fig. 2.7. Mixed mode Sampling and Interpolation

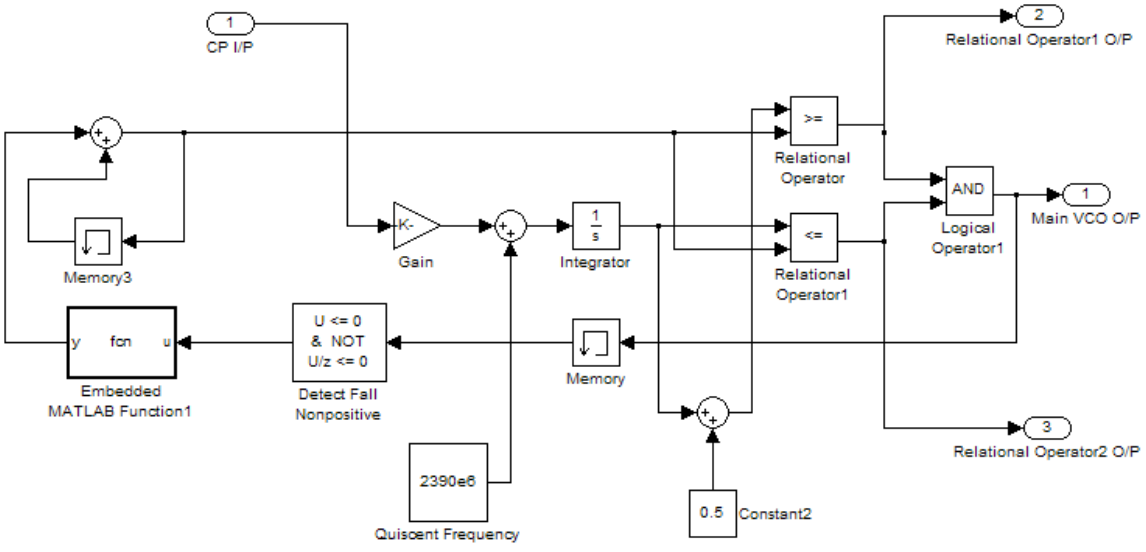


Fig. 2.8. DVCO Architecture

feedback and hence update the threshold of the relational operators. The result is a clock of duty cycle 50% as shown in Fig. 2.9, taken directly from the . To vary the duty cycle, the 0.5 constant block is varied between 0 and 1. Since all simulators are sensitive to race conditions, a memory block is added in the feedback to successfully effect the oscillation with simulation errors.

The simulation to obtain Fig. 2.9 was performed for $f_0 = 2.39 \times 10^9$ GHz with a redundant clock modeling $f_{\max} = 5 \times 10^9$ GHz just as in the case for the continuous VCO simulation, thus ensuring mixed-mode simulation and hence optimized tradeoff between simulation time and accuracy (Note: Accuracy here is measured as directly related to the number of samples taken, i.e. $E(N)$ as derived in 2.5).

Notice however, that the rising edge is created at a multiple of $1/2f_{\max}$ in the absolute time scale in Fig. 2.9. However, the falling edge is perfectly produced at $1/2f$.

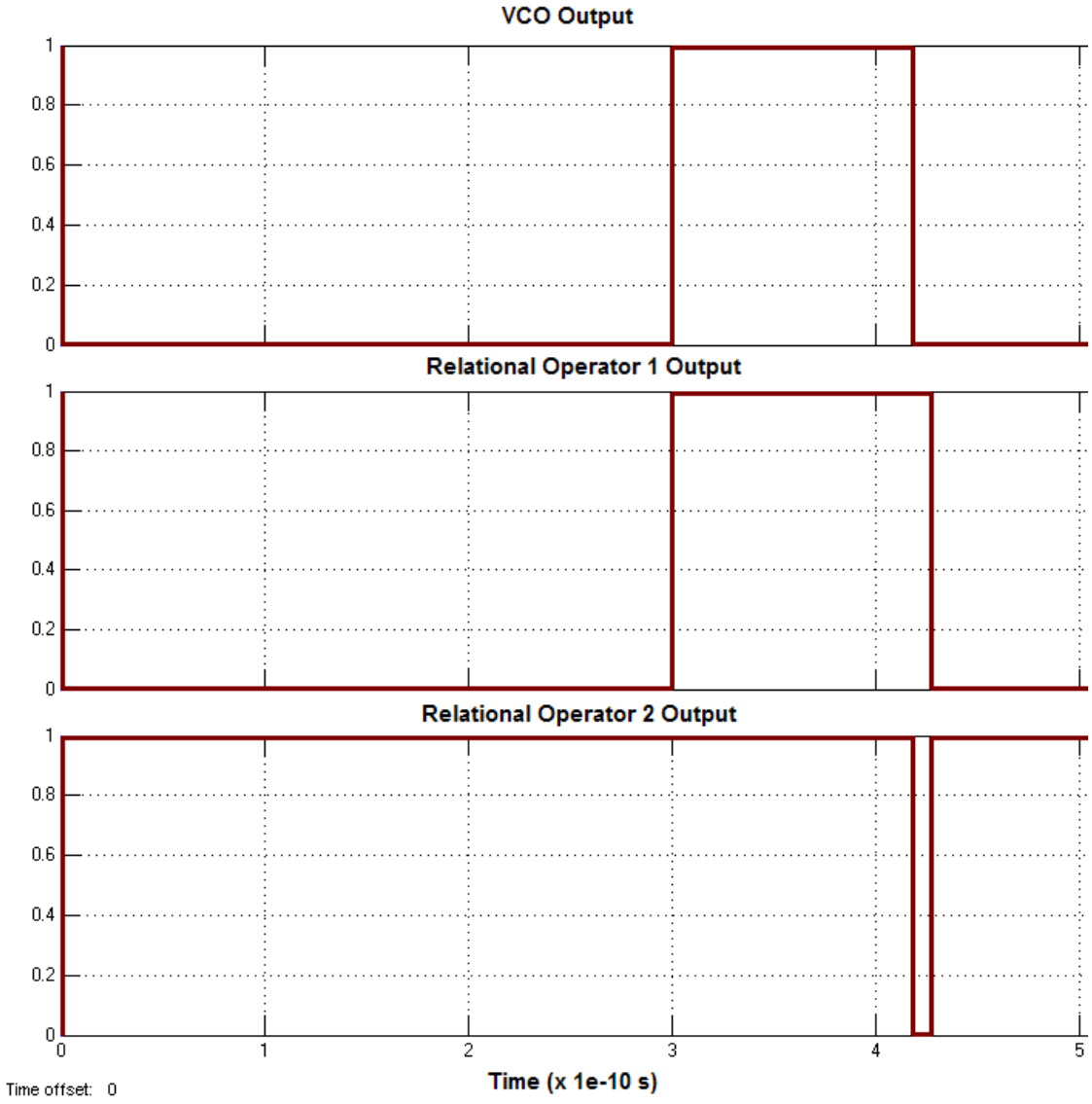


Fig. 2.9. DVCO Scope Output

2.4.2 Rising and Falling Edge Measurements

The relation operators take an analog input and give a digital output while simultaneously; their thresholds are updated by an analog memory circuit. Therefore, for the digital output of the VCO to react at the frequency $f = 2.39 \times 10^9$ Hz, a redundant "Pulse Generator" block of pulse rate 5×10^9 Hz is placed within the model making the $f_{\max} = O\left(10^9\right)$ Hz, in accordance to the theory presented in Section 2.3.

Note that from Fig. 2.9 the state change for the rising edge should occur at $t = (1 + 2n)/(2 \times 2.39 \times 10^9)\ s$ where $n = 1, 2, 3 \ldots$ and so on. However, since the rising edge is actuated by the relational operators who's output is digital, the corresponding time stamp occurs at the integral multiple of $1/2f_{\max}$ immediately succeeding $t = (1 + 2n)/(2 \times 2.39 \times 10^9)$. The falling edge is activated by the update in the analog memory unit, which occurs as the continuous event-driven simulated part of the model, resulting in a near perfect edge corresponding to the actual edge observed in the clock. Therefore, we may define rising and falling timestamps vectors T_{RE} and T_{FE} respectively, as:

$$T_{RE} = \frac{m}{2f_{\max}};\ \ S.T.\ \ \min_m \left(\frac{m}{2f_{\max}} - \frac{1+2n}{2f}\right)\ \text{ and }\ \frac{m}{2f_{\max}} > \frac{1+2n}{2f};\ \ n \in \mathbb{N} \tag{2.6}$$

$$T_{FE} = \frac{n}{f};\ \ n \in \mathbb{N} \tag{2.7}$$

This shows that the set of T_{RE} can give erroneous readings which corresponds to $t_{\text{timestamps}}$ from (2.3). For the optimization problem, with variable $f_{\max}$:

$$\min_{f_{\max}} \left(\frac{m}{2f_{\max}} - \frac{1+2n}{2f}\right)\ \text{ and }\ \frac{m}{2f_{\max}} > \frac{1+2n}{2f};\ \ n \in \mathbb{N} \tag{2.8}$$

the solution is $f_{\max} = \infty$ for large m. This means that the discrete sampling can only give perfect edges at the rising edge at very large sampling rates which is impractical in terms of simulation time. Therefore, it is only practical to take all measurements regarding timing information and clock frequency at the falling edge.

2.5 Conclusions for Chapter 2

In this chapter the author developed the basic concepts of a simulation environment and linked it with the simulation of a high frequency RF VCO. For this particular case, we defined the simulation environment as performing mixed-mode simulation and that the optimal accuracy and simulation time trade off can be obtained by varying the fastest digital block frequency. Various components of the timing errors (t_{ADC} etc.) were defined and quantified. For the specific case of the DVCO architecture we find that part of the model is discretely simulated which gives error in edge timing measurements of the rising edge whereas gives perfect falling edges.

This specific analysis is developed for the VCO as it is critical in the measurement of the phase noise and other parameters at the edge of the clock. Such analysis can be extended to other blocks that have both discrete simulation and continuous event-driven simulation sub-blocks.

Consider the original PLL as described in Section 1. The PLL contains now a DVCO (in order to eliminate the t_{ADC}). However the rest of the PLL contain only purely discrete or analog blocks. The feedback divider and the Phase Detector are purely digital; and the charge pump and the loop filter are purely analog.

Therefore, the only block that directly creates or is responsible for the mixed-mode simulation is the VCO. Therefore the timestamp analysis (which is what is essentially performed in Chapter 2) is extended directly to the entire PLL without analyzing the effects on the timestamps from the rest of the PLL model.

With this background and extensive analysis of the simulation environment, the author will proceed now to the physical modeling of the PLL and FS.

Chapter 3

Behavioral Modeling of the PLL

3.1 Introduction

The behavioral modeling of the PLL is essentially defining equivalent blocks for the circuit of the PLL. However, there exist some interesting results that arise during the modeling with these blocks. For example, the circuit level architecture of the divider may interact with the rest of the circuit to induce some offsets in the final FS output.

In the analysis above, we have assumed that the solver is capable of detecting events and queue timestamps during the simulation. This was expressed as a pseudo-random value in the timesteps and used in the analysis of timing errors. However, the unpredictable queuing of random events occurring in one sub-block can affect the simulation of other sub-blocks. In a cascaded linear system this will not be observed, as the simulations and events are triggered one after other. However, when sub-blocks are used in feedback (such as in the PLL), the events may not be properly queued. The general modeling of the PLL and these effects in the mixed-mode simulation environment will be the focus of this chapter.

3.2 Target Frequency Synthesizer Designs

We now analyze the frequency synthesizer as described by Fig. 1.1 for two target protocols. The transfer function of the closed loop FS is given as the following [12].

$$H(s) = \frac{\Theta_{\mathrm{o}}(s)}{\Theta_{\mathrm{REF}}(s)} = N \frac{\frac{K_d K_v}{N} \frac{Z(s)}{s}}{1 + \frac{K_d K_v}{N} \frac{Z(s)}{s}} \tag{3.1}$$

In (3.1), K_d is the gain of the charge pump, K_v is the VCO gain in $\mathrm{Hz/V}$, N is the feedback division ratio and $Z(s)$ is the transfer function of the loop filter. Fig. 3.1 gives a detailed picture of

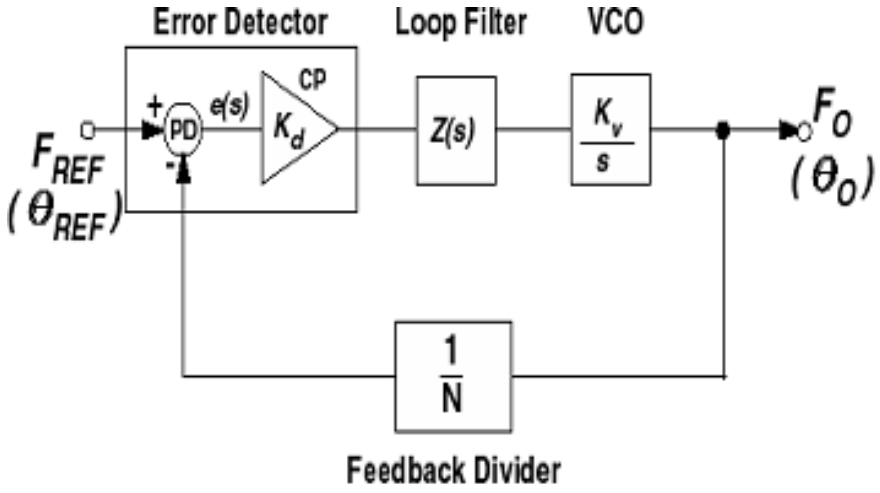


Fig. 3.1. Expanded Frequency Synthesizer Diagram

the individual components of the frequency synthesizer. The transfer function is a representation of the transfer of the phase between the reference and the output and hence a justification of the term phase-locked loop. The phase detector detects a phase which is first converted to a current response and by the charge pump and then filtered and then passed through the VCO to obtain the final response. The $1/s$ term in the transfer function arises due to the conversion of the frequency input to a phase output and hence a waveform, which is typical of the VCO transfer function.

The above transfer function is used with the DVCO as described in Section 2. Therefore, th divider input is a perfectly square clock and so is the input to the Phase Frequency Detector (PFD). To design specific frequency synthesizers, the above transfer function has to analyzed for stability. We will perform the stability and subsequent modeling and design for two frequency synthesizers. The next few subsections will focus on the design philosophy.

3.2.1 GSM Frequency Synthesizer Design and Results

The GSM protocol operates in many bands but we will consider one that operates in the band 924-927 MHz range [13]. The spectrum is divided into 11 channels giving a step size of 300 KHz. This implies a reference frequency of 300 KHz. Assume the VCO works between a voltage 0.3 V and 3.3 V. Table 3.1 describes the entire specifications. From these specifications, we can derive the various parameters as:

$$K_v = \Delta f / \Delta V_{\mathrm{VCO}} = 1 \text{ MHz/V} \tag{3.2}$$

$$N = 924/0.3 \text{ - } 927/0.3 = 3080 \text{ - } 3090 \tag{3.3}$$

$$f_q = 924 - (0.3 \times 1 \text{ MHz/V}) = 923.7 \text{ MHz} \tag{3.4}$$

The terms are as defined from before. Assuming the loop filter is of order 2 (as time domain and frequency domain stability is assured only for *loop filter order* > 2 [12]), we can assume a loop filter circuit as shown in Fig. 3.2. The design of this loop filter will complete the system design of the entire PLL.

Table 3.1. GSM Frequency Synthesizer Specifications

Parameter	*Specification*
Frequency Range	924 MHz – 927 MHz
Step Size	300 KHz
Control Voltage	0.3 V – 3.3 V

The stability can be analyzed using time constants. As a thumb rule, the bandwidth of the PLL should be less than 10 times the reference frequency for stability. Therefore, for $f_{\mathrm{REF}} = 300$ KHz the bandwidth should be less than 30 KHz. For a loop filter of second order shown (Fig. 3.2), the transfer function of the loop is:

$$H(s) = N\frac{Ks + K\omega_2}{s^3/\omega_3 + s^2 + Ks + K\omega_2} \tag{3.5}$$

where:

$K = K_d K_v Z_h / N$

$Z_h = \frac{R(C_1+C_2)}{C_1}$

$\omega_2 = \frac{1}{R(C_1+C_2)}$

$\omega_3 = \frac{1}{RC_2}$

The above transfer function can be verified for by substituting the transfer function for the loop filter in Fig. 3.2 in (3.1). For stability, the 3_{dB} bandwidth is set as 20 KHz. Therefore, from open loop characteristics:

$$\omega_{3_{\mathrm{dB}}} = K_d K_v Z_h \tag{3.6}$$

For small settling time: $\omega_{3_{\mathrm{dB}}}/4 = \omega_2$ and $4\omega_{3_{\mathrm{dB}}} = \omega_3$. The value of the charge pump gain is K_d is set as $10\ \mu$A. This gives three equations in R, C_1 and C_2 which when solved gives:

$R = 5.775\ \mathrm{M\Omega}$

$C_1 = 32.4765\ \mathrm{pF}$

$C_2 = 2.1654\ \mathrm{pF}$

The corresponding transfer function for the closed loop, for $N = 3080$ is:

$$H(s) = \frac{\left(4.775 \times 10^6\right) s + \left(2.386 \times 10^{10}\right)}{\left(1.2505 \times 10^{-5}\right) s^3 + s^2 + \left(4.775 \times 10^6\right) s + \left(2.386 \times 10^{10}\right)} \tag{3.7}$$

The step response of $H(s)/N$ shown in Fig. 3.3 indicates a stable closed loop (no scaling with N done there). Also, since the transient is observed to settle at about $1.4 \times 10^{-4} s$, the simulation time and settling time of the FS is the same. As a result, the control voltage characteristics should also follow a typical curve, settling at a certain value after $1.4 \times 10^{-4} s$. This can be observed with time domain simulations only, as there is no direct output for time domain voltage in $H(s)$.

The results of the time domain simulation of this frequency synthesizer will be discussed in subsequent sections.

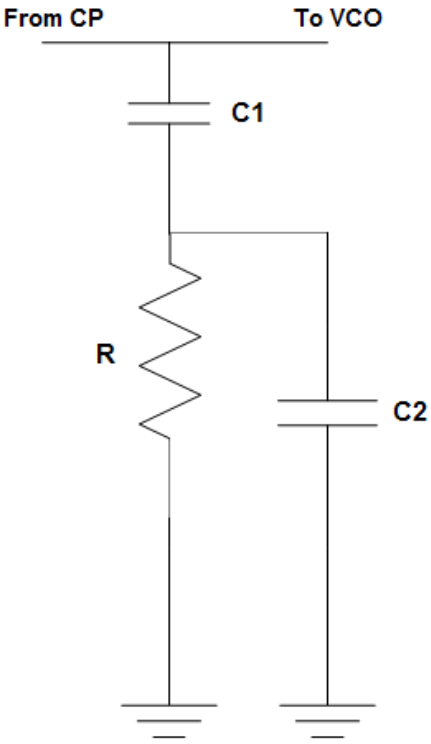


Fig. 3.2. Second Order Loop Filter for GSM FS

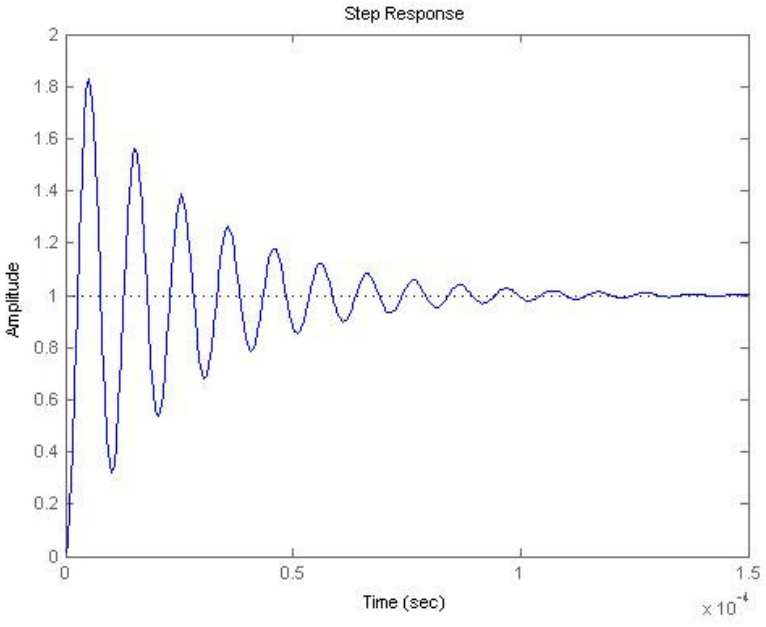


Fig. 3.3. Step Response for the GSM FS Transfer Function

3.2.2 Zigbee Frequency Synthesizer Design and Results

A change in design philosophy is followed in the design of the Zigbee FS. The Zigbee protocol works between the frequencies of 2.4 and 2.48 GHz frequencies and contains 16 channels [14]. This implies a step size and hence a reference frequency of 5 MHz. Assuming the VCO works between 0 V and 1.6 V, we have the complete specifications defined by Table 3.2. From these specifications, we can derive the various parameters as:

$$K_v = \Delta f / \Delta V_{\text{VCO}} = 60 \text{ MHz/V} \tag{3.2}$$

$$N = 2.40/0.005 \text{ - } 2.48/0.005 = 480 \text{ - } 496 \tag{3.3}$$

$$f_q = 2.48 - (1.5 \times 0.06 \text{ MHz/V}) = 2.39 \text{ GHz} \tag{3.4}$$

The above assumes that a noise margin on the voltage high of 0.1 V. Using a second order filter, (shown in Fig. 3.4), we can analyze the stability of the closed loop system. The GSM frequency synthesizer was designed keeping in mind the stability and the settling time requirements. However, a cleaner procedure is to observe the phase margin at the cutoff frequency and maximize at the point to obtain confirmed stability.

The transfer function of the loop filter is:

$$Z(s) = \left(\frac{1}{C_1 + C_2}\right) \cdot \frac{1 + sC_2R_2}{s\left(1 + s\frac{R_2C_2C_1}{C_1+C_2}\right)} \tag{3.5}$$

Define $T_z = C_2R_2$ and $T_p = R_2C_2C_1/(C_1 + C_2)$. This gives an open loop gain (the open loop is defined as the path from Θ_{REF} to $\Theta o/N$) as:

$$G(j\omega) = \frac{K_dK_v}{j\omega(C_1 + C_2)} \cdot \left(\frac{1 + j\omega T_z}{j\omega(1 + j\omega T_p)}\right) \cdot \frac{1}{N} \tag{3.6}$$

Differentiating the phase margin to obtain a maxima, we get $\omega = \omega_m = 1/\sqrt{T_zT_p}$. Since due to stability requirements, $\omega_m = \omega_c = \omega|_{G(j\omega)=1}$, the following final equations are obtained

$$C_1 + C_2 = \frac{K_dK_v}{N\omega_c^2} \cdot \sqrt{\frac{1 + \omega_c^2T_z^2}{1 + \omega_c^2T_p^2}} \tag{3.7}$$

$$T_p = \frac{\sec\phi_0 - \tan\phi_0}{\omega_c} \tag{3.8}$$

Table 3.2 Zigbee Frequency Synthesizer Specifications

Parameter	*Specification*
Frequency Range	2.4 GHz – 2.48 MHz
Step Size	5 MHz
Control Voltage	0.2 V – 1.6 V

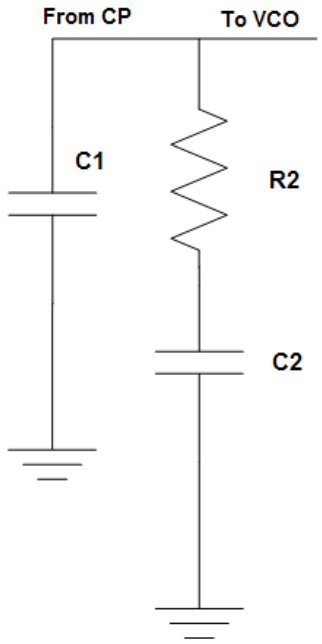


Fig. 3.4. Second Order Loop Filter for Zigbee FS

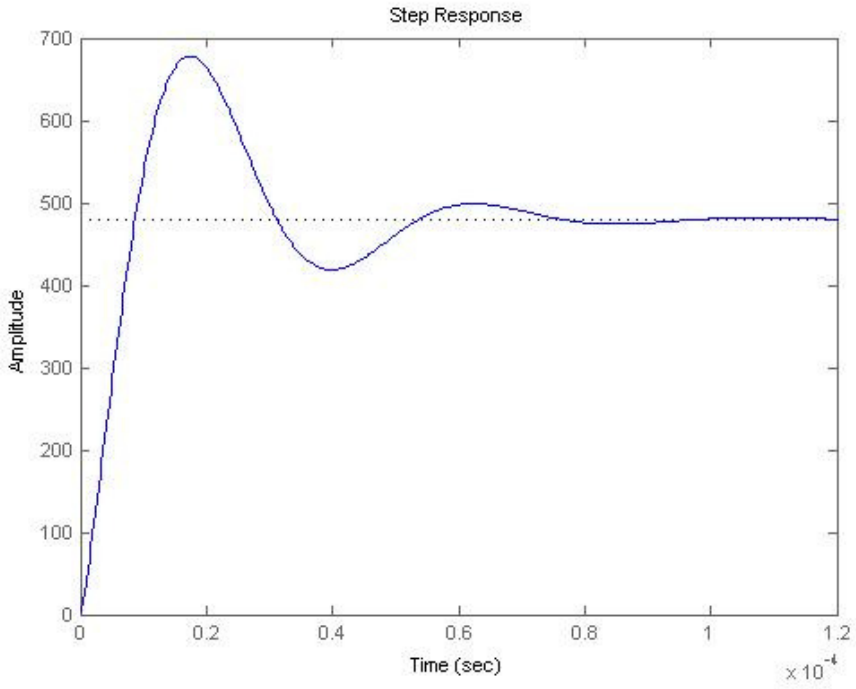


Fig. 3.5. Step Response for the GSM FS Transfer Function

Using the above equations, if we set $K_v = 60$ MHz,$K_d = I_{cp}/2\pi = 24/2\pi$ μA $= 3.8197$ μA, $N = 480$, we obtain with a target bandwidth = $\omega_c = 500$ KHz and required phase margin = $\phi_0 = 50°$:

$R_2 = 235$ kΩ

$C_1 = 1$ pF

$C_2 = 21$ pF

These parameters are used in the loop filter to obtain a stable response from the FS. The final transfer function of the FS is given in (3.9) and the corresponding step response in Fig. 3.4.

$$H(s) = \frac{\left(5.141 \times 10^7\right) s + \left(1.042 \times 10^{11}\right)}{\left(2.243 \times 10^{-7}\right) s^3 + s^2 + \left(1.071 \times 10^5\right) s + \left(2.170 \times 10^{10}\right)} \tag{3.9}$$

Although the design has been made for $N = 480$ it will be stable for up to $N = 496$ as there will be only marginal changes in the values of ϕ_0 and ω_c.

Therefore, as far as design is concerned, the values derived above for the capacitances, resistances, charge pump current, VCO gain and division ratios will be used for subsequent modeling. Fundamental differences in modeling the above two FS will now be considered.

3.3 Modeling of FS – Everything but the Divider

The modeling of the FS is not a very complex issue once the modeling of the VCO has been done. However, in event driven simulation, the simulation is highly sensitive to the model architecture. Thus, it is first important to consider the all components without accounting the simulation environment. The models for the individual components of the FS (all, but the divider) is explained in this section.

3.3.1 Modeling the Phase Frequency Detector

On the circuit level, a more useful method to implement the phase detector is to use a phase frequency detector (PFD), which will prevent the PLL from locking onto harmonics and the no detection stage. The PFD model (Fig. 3.6) uses two D-type Flip Flops and a NAND gate for feedback to clear the states [12].

In the circuit level, there is a delay usually created due to the presence of the NAND gate. If this delay is high enough, it can prevent any other delay mismatch between the D-type flip flops. However, the mismatch is well noticed when the delay is not high enough. Since a clear tradeoff between mismatch and delay is required, in the design a 0.5% delay is introduced. This delay is 0.5% of $O\left(1/f_{\mathrm{REF}}\right) = 0.005/\left(300 \times 10^3\right) = 1.6667 \times 10^{-8} s$. This delay will sufficiently avoid any mismatch and the requirement for modeling the mismatch.

At the output of the UP and DOWN voltages, the phase difference is indicated at the UP voltage if Reference signal is ahead and is indicated in the DOWN voltage if the Divider Signal is ahead. The inbuilt MATLAB functions are used to convert the logical data-type to double data-type required for future processing.

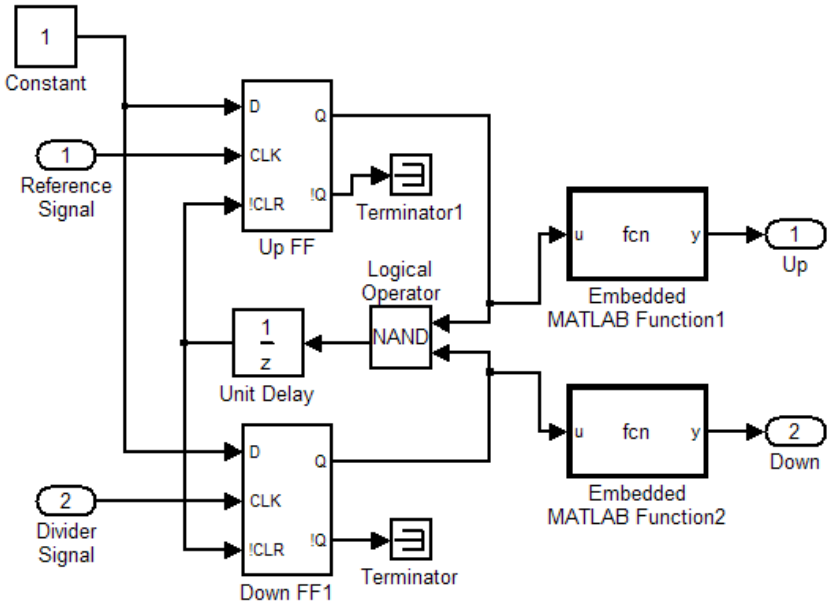


Fig. 3.6. PFD Model

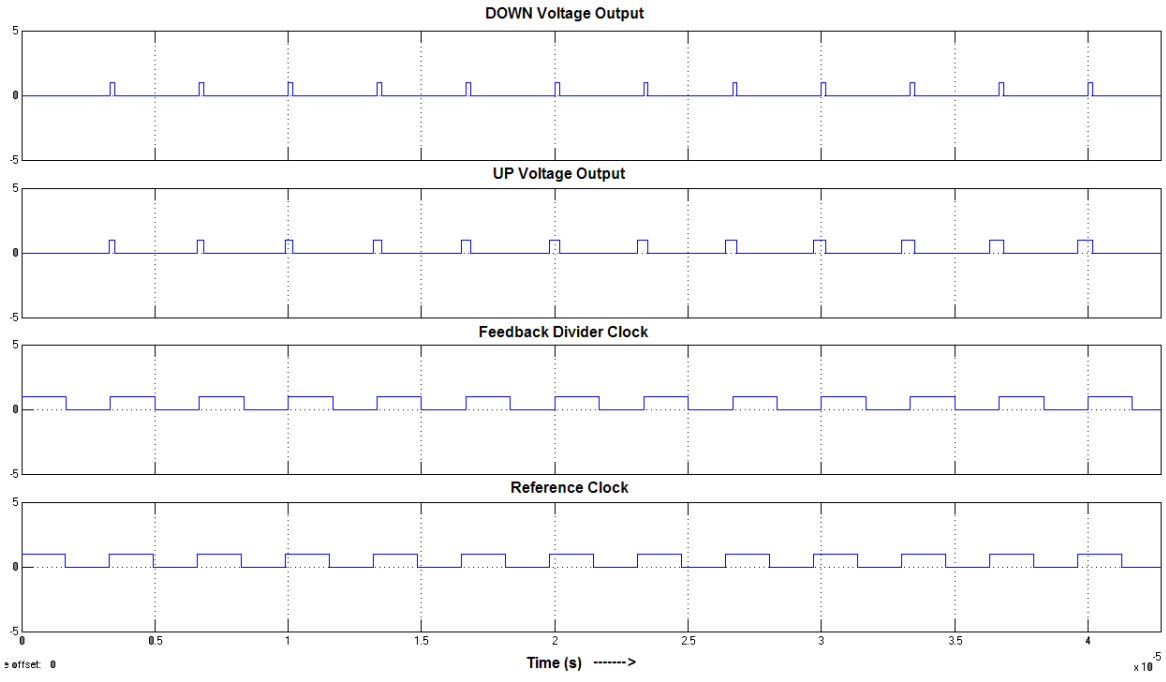


Fig. 3.7. PFD Scope Output in Discrete Simulation

The simulation of the PFD alone is completely discrete in nature as all the blocks are digitally activated. Therefore the simulation time step (from the theory in Section 1) is equal to the fastest reacting block which is the delay unit. Therefore, $t_{\text{step}} = 1/f_{\max} = 1.6667 \times 10^{-8} s$. Interestingly, the delay in the feedback determines the simulation time. However, note that:

$$O(t_{\text{step,PFD}}) << O(1/f_{\text{o}}) \tag{3.10}$$

Therefore, in the case of mixed-mode simulation (from Fig. 2.6), the PFD will be simulated near perfectly, as it operates in the low frequency region and is automatically sampled enough by the simulation environment.

The simulated output of the PFD in discrete simulation is shown in Fig. 3.6. The frequency of the feedback clock is slightly more than the reference frequency clock. This gives ever-increasing phase detection in the UP Voltage Output and a constant pulse in the DOWN Voltage Output created due to the delay in the feedback. The final output to the charge pump is $V_{\text{UP}} - V_{\text{DOWN}}$. This gives the final output as a signed pulse stream proportional in duty cycle to the phase difference and $-\text{ve}$ if ahead in phase with respect to f_{REF} and $+\text{ve}$ if behind in phase with respect to f_{REF}. The performance of the PFD in the completed model will be analyzed subsequently.

3.3.2 Modeling the Charge Pump and Loop Filter

Once the discrete PFD block provides the output, the charge pump (CP) and the loop filter are pure continuous time blocks. Also, they involve no feedback and are simple feedforward blocks. Further, the circuit level design of the charge pump and the loop filter involve simple amplifiers and passive elements (for the loop filters).

The models of the charge pump and loop filter are simple gain blocks (with the designed I_{CP}) and the corresponding transfer function (predefined s-block available in MATLAB). The reader should note that the even though modeling is in the time domain, there are absolutely no delays introduced in the charge pump and loop filter; thus an $s-\text{domain}$ representation is equivalent to a time-domain model for the loop filter. Similarly, gain in the charge pump is the same in the $s-\text{domain}$ and the time domain. Fig. 3.8. shows the CP and loop filter model in cascade.

The models shown below assume no non-idealities and are shown for the GSM protocol, with $I_{\text{CP}} = 10\ \mu\text{A}$ and Z(s) as derived in Sec. 3.2.1.

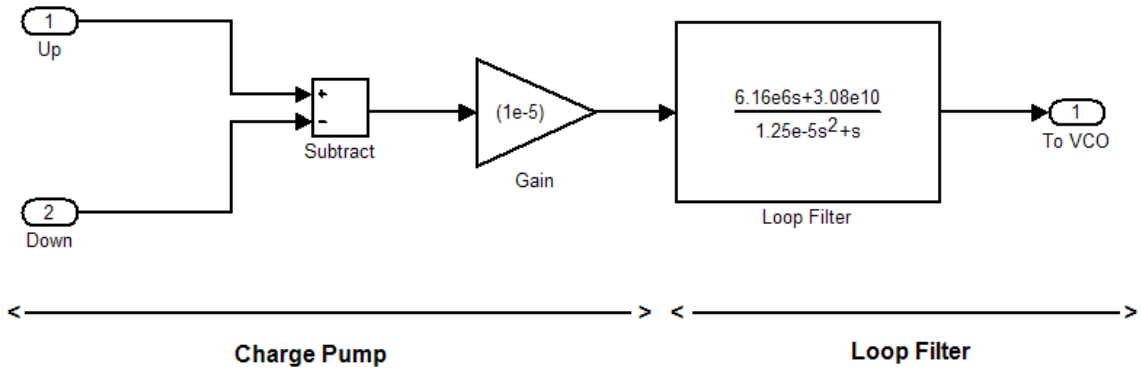


Fig. 3.8. Charge Pump and Loop Filter Model

3.3.3 Modeling the VCO

The DVCO architecture described in Section 2.4 is used in the case here too. In Fig. 2.8 the value of the gain K_v is changed according to the design and the value of the quiescent frequency f_q is changed according to the design.

The non-idealities in the VCO are of prime importance in the performance of any PLL, due to many reasons, two of them mainly being: (i) The non-idealities in all the other blocks if modeled as noise can be filtered by the loop filter which is low pass (ii) The non-idealities are generally highly non-linear and require time-variant modeling and are directly reflected in the output of the VCO. Therefore, with careful measurement, we can eliminate t_{ADC}, $t_{\mathrm{timestamps}}$ and $t_{\mathrm{numerical}}$ to focus only on jitter measurements.

Note in the modeling of the VCO, a redundant clock of the order $O(f_{\mathrm{o}})$ is very important as only then the sampling will be assured in the mixed mode domain and the output of the VCO be observed as a waveform with near uniform duty cycle. In simulations we have used for example a redundant clock that gives $f_{\max} = 5 \times 10^{9}$. cycle. We will consider simulation results separately in subsequent sections. Now that the RF portion of the PLL has been modeled, it is important to focus on the baseband operated divider.

3.4 Divider Architecture

The divider is critical to operating the FS in low power as it is generally the largest and the most power hungry block of the entire PLL. Besides, the digital circuit of the divider should be capable of switching at RF frequencies and also should not introduce delays. We will analyze two divider architectures for the two protocols and underline the difference arising in the simulation environment due to them.

3.4.1 Large Prescaler based Divider

We model the GSM protocol FS divider using this architecture. The large prescaler architecture is equivalent to modeling the using a toggling prescaler with a main and swallow counters in cascade [12] (see Fig. 3.9). The prescaler divides by $P+1$ till the swallow counter finishes down-counting and then divides by P till M finishes down-counting. The down-counting in itself is division, therefore, the total division ratio is:

$$(M-A)(P) + A(P+1) = MP + A \tag{3.11}$$

Using Flip Flops and simple gates the Prescaler based divider can be modeled as shown in Fig. 3.11. Since we are using this divider for the GSM protocol, we obtain $M = 48$, $P = 64$ and A varying from 8 to 18, in order to obtain $MP + A$ as between 3080 and 3090. This can be implemented as a simple baseband loading vector as shown in the model in Fig. 3.11. The design of the swallow counter, divide-by-64 and divide-by-65 counters, and the main counters is not explained as they are very simple and require minimal digital design techniques.

Note however, that the multiplexer is triggered by the swallow counter and so is the input RF waveform toggled using AND gates and the output of the swallow counter. This is done to ensure that due to varying division by the divide-by-64 and divide-by-65, the edges don't mismatch during the simulation.

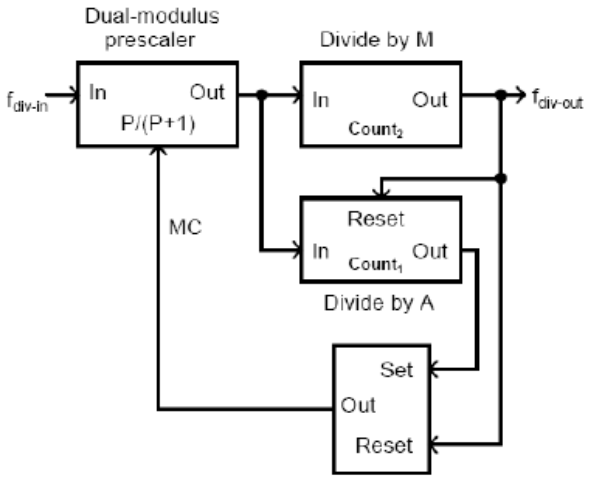


Fig. 3.9. Prescaler based Divider

There are two unit delays also introduced in the model. The unit delay between the multiplexer and the main counter is to prevent a race-condition loop, and the unit delay between the main counter and the swallow counter is to prevent the trigger to reaching before the clock pulse from the prescaler. The entire divider is a discrete simulation block and hence the time step is completely determined by the fastest changing discrete block. Since the VCO operates at mixed-mode simulation, the fastest the changing block in the divider is the RF input and hence, if the divider is simulated with the VCO, the simulation step size will be affected by the order of the $f_{\max}$. Hence, divider simulation outputs have to be noted when the PLL is simulated. However, a qualitative output of the divider in *discrete simulation mode* is shown in Fig. 3.10.

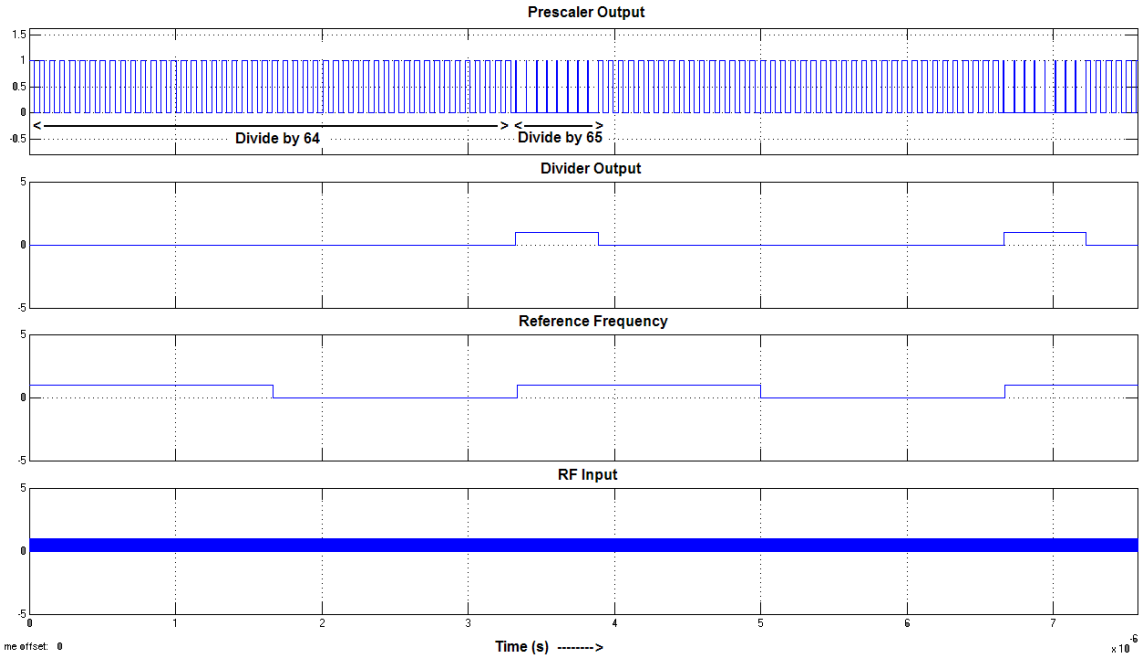


Fig. 3.10. Prescaler based Divider Scope Output

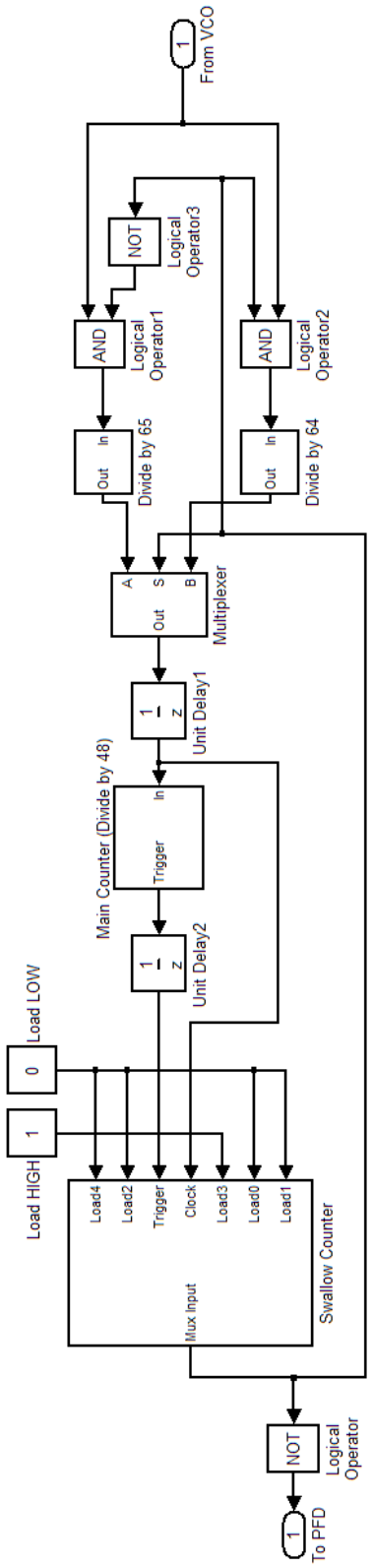


Fig. 3.11. Prescaler based Divider Model (please rotate document to view)

The input to the simulation result in Fig. 3.10 is a 924 MHz clock and the swallow counter has been set to a binary vector of 8 to obtain a division ratio of 3080. Although not shown, when zoomed in, the divider output edge does not exactly coincide with the reference frequency output. This anomaly will be addressed in subsequent sections. Also note that the output of the trigger of the swallow counter can itself be used to detect as a clock, as the edge is sufficient to activate the PFD.

3.4.2 2/3 Cell based Multimodulus Divider Architecture

This divider architecture, more recently proposed in [15], is a scalable efficient architecture without requiring the redesign of the entire divider module and using feedback. The basis of the architecture is the use of a 2/3 cell (shown in Fig. 3.11). The 2/3 cell is capable of dividing by either 2 or 3 depending on the input p. The implementation essentially consists of two MOS gates attached to Flip Flops and two separate Flip Flops.

Keeping this structure in mind, we connect many 2/3 cells in cascade to obtain a divider architecture as shown in Fig. 3.12. The division ratio is controlled by the word $(p_0, p_1 \cdots, p_{n-1})$.

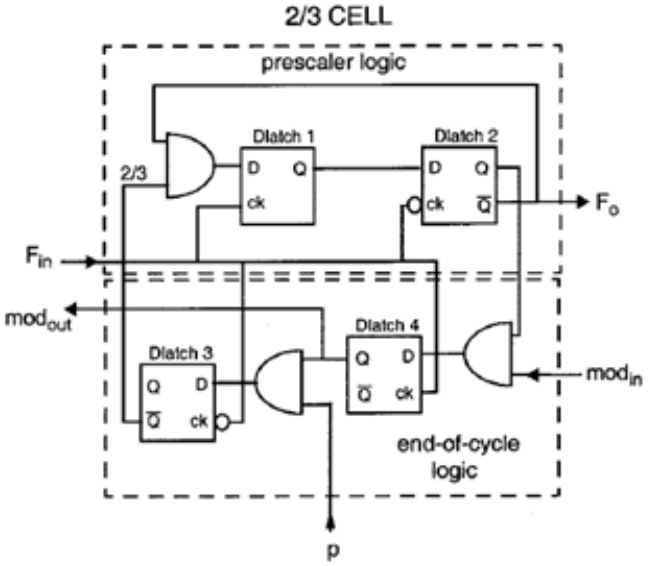


Fig. 3.12. 2/3 Divider Cell

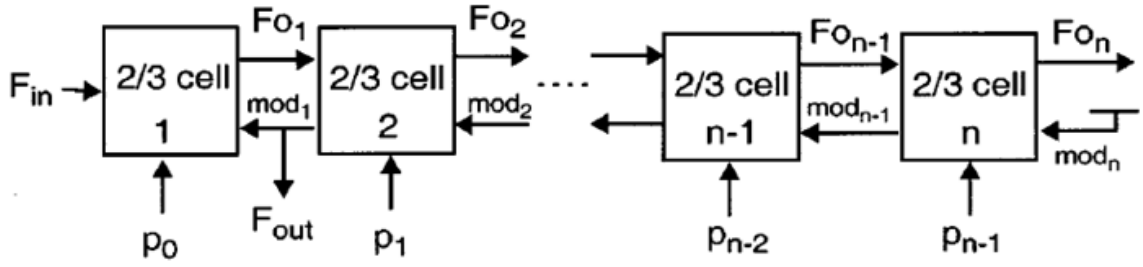


Fig. 3.13. 2/3 Cell Divider Architecture

By default all the clocks divide by 2 atleast thereby n cells provide atleast a 2^n division ratio. This ratio is now enhanced by allowing the propagation of the *mod* signal Each time the mod signal arrives at a particular cell the clock is allowed to add one more cycle thus activating the divide by 3, if the value of p at that stage is set at 1. This in turn propagates throughout the cell to finally reach F_{out}.

Therefore, picture the whole system as a clock where the divide by 2 goes on indefinitely in one direction, while simultaneously the *mod* creates the extra division. The final division ratio is given as:

$$\begin{aligned} T_{\text{out}} &= 2^n + 2^{n-1} \cdot T_{\text{in}} \cdot p_{n-1} + 2^{n-2} \cdot T_{\text{in}} \cdot p_{n-2} + \ldots + 2 \cdot T_{\text{in}} \cdot p_1 + T_{\text{in}} \cdot p_0 \\ &= 2^n + \left(2^{n-1} \cdot p_{n-1} + 2^{n-2} \cdot p_{n-2} + \ldots + 2 \cdot p_1 + p_0\right) \cdot T_{\text{in}} \end{aligned} \tag{3.12}$$

The advantage of the above divider is its scalability and the relatively lesser feedback. The feedback only exists within the 2/3 cell and the rest of the system only has two feedforward propagation lines. As mentioned before, the events for the feedforward lines get queued up, however if there exists a large amount of feedback, the event queuing is quite complex in an event-driven simulation environment.

The Zigbee FS has been modeled with the 2/3 cell architecture. Each 2/3 cell is created using simple gates and without any requirement for delays (as there are no combinational loops). The simplified experimental setup for the *discrete simulation* is shown in Fig. 3.13. The corresponding output with the word $(p_0, p_1 \ldots, p_{n-1}) = 11100000$ to obtain a division ratio of 256 + 224 = 480 (refer to subsection 3.2.2 for details on how 480) is shown in Fig. 3.15. The input is a frequency of 2.40×10^9 GHz and the output is a frequency of 5×10^6 MHz. Since this architecture is superior to the large prescaler one (subsection 3.4.1), we will later analyze many other phenomena such as noise modeling with respect to this architecture.

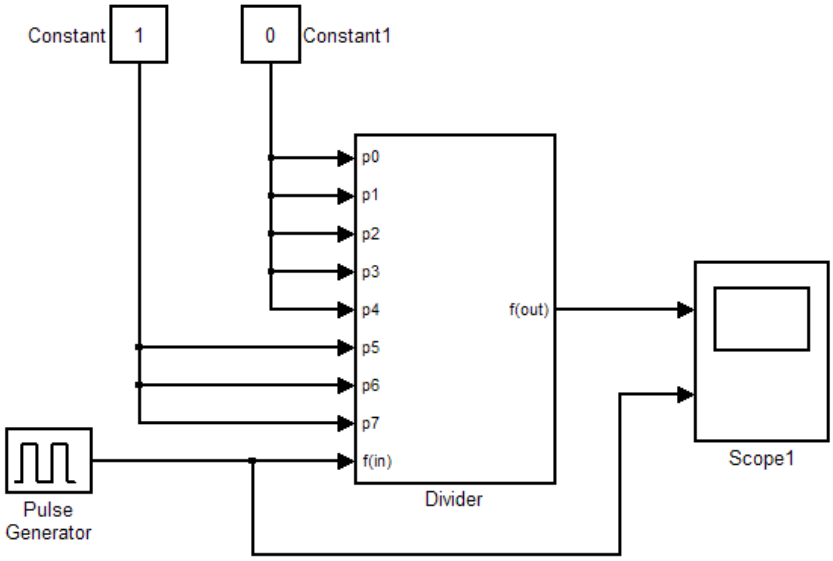


Fig. 3.14. 2/3 Cell Divider Model

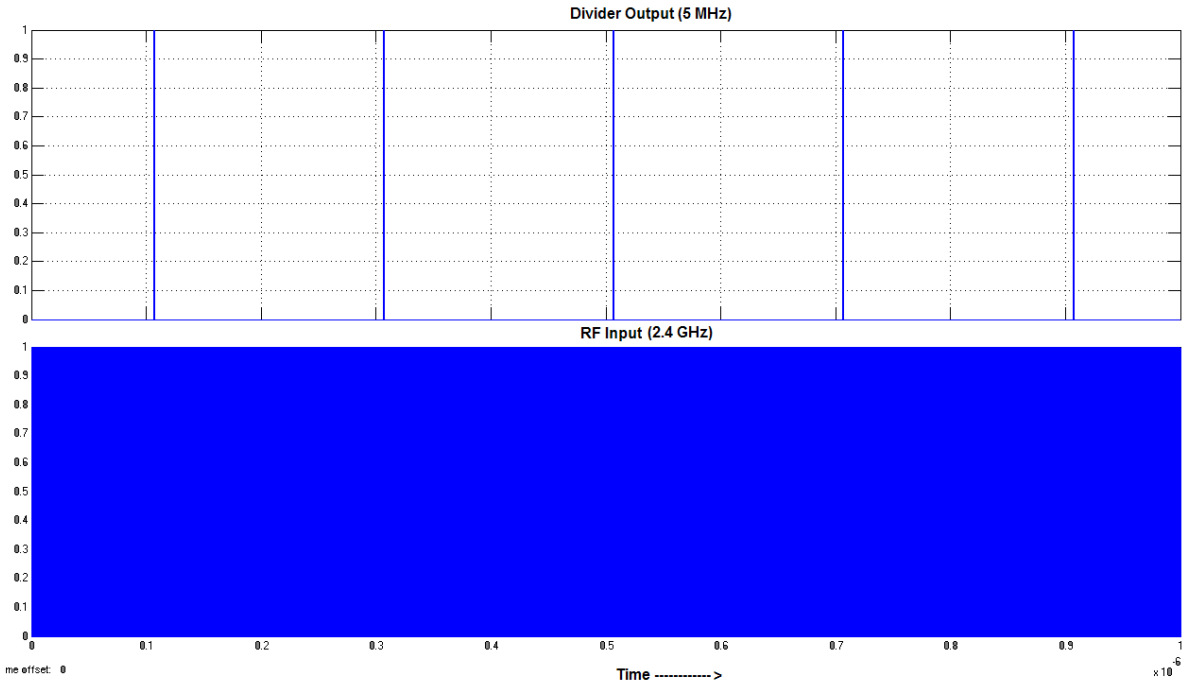


Fig. 3.15. 2/3 Cell Divider Discrete Simulation Results

3.5 Simulation and Results

The simulation of both the frequency synthesizers is performed after cascading the various block models and including the redundant digital clock to set $f_{\max}$. Interestingly, one should note that the digital blocks of the divider work on the rising edge as the simulation is performed discretely at these edges. However, the VCO gives perfect edges only at the negative edge. Therefore working in at least the mixed-mode simulation region will ensure sampling at the right positions in order to create credible simulation results. This and other phenomenon will be discussed in this section.

3.5.1 Event Queue Handling – Practical Considerations

Consider the simulation of a system such as the one shown in Fig. 2.3. However, if we consider two clusters of systems instead of one, this implies that one solver will handle event queues from two models in the same environment. Therefore the timing of the index l, t_l will be a function of two input trajectories. This implies that the value of $t_{n+1} - t_n$ from subsection 2.2.1 will change to give:

$$t_{n+1} - t_n = g\left(x_{11}, x_{12}, \ldots, x_{21}, x_{22}, \ldots\right) = t_{\text{step}} \tag{3.13}$$

where g is a function of the based on the rate of events and x_{1i} are the events occurring in subsystem 1 and x_{2i} are the events occurring in subsystem 2. This poses a problem for analysis as if either of the subsystems dominates then the simulation of the other will be affected. Since for mixed-mode simulation, t_{step} is directly related to $f_{\max}$, one model will get sampled more than the

other thus resulting in unnecessary increase in simulation time. As a corollary, for the VCO, ideally the simulation of the system in the setup in Fig. 2.5 should not produce any oscillations if the value of the redundant clock frequency is lesser than $O(f_o)$. With the introduction of another continuous event-drive simulation model in the same schematic, the solver tries to accommodate the two models together and thus produces erroneous results.

Therefore, as a practical consideration, the experimentalist should always simulate two schematics of different orders of f_{max} separately.

Another important effect that produces erroneous results is the simulation of feedback systems. A typical example is the prescaler based divider architecture discussed in Sec. 3.4.1. In feedback systems, the event driven simulation queues a lot events in a short period of time, therefore, the rate at which computation takes place may reduce in comparison to the rate at which events of queued. This will be discussed soon in a subsequent subsection. However, it is best to avoid models with a lot of feedback; an example is our change in section 3.4.2 to the 2/3 cell divider architecture in simulating the Zigbee FS.

3.5.2 Simulation of the GSM FS

The GSM FS is simulated using the design developed in Sec. 3.2.1 and the prescaler based divider architecture. The entire model is shown in Fig. 3.17. The simulation settings are Range-Kutta ode45 solver and automatic timesteps evaluation. A locking condition is usually confirmed by observing the control voltage at the VCO input. If this voltage is stable after sometime then the lock condition has been approached.

Fig. 3.16 shows the locking condition in the frequency synthesizer. The results of the simulation are discussed below.

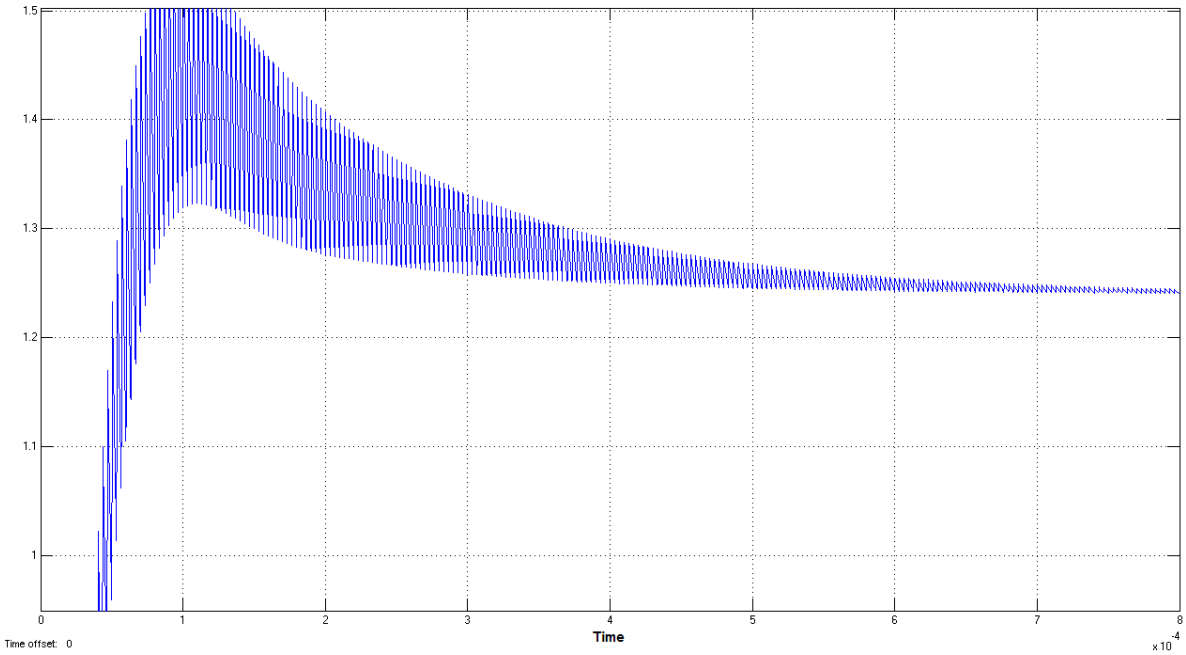


Fig.3.16. Locked Condition in GSM FS

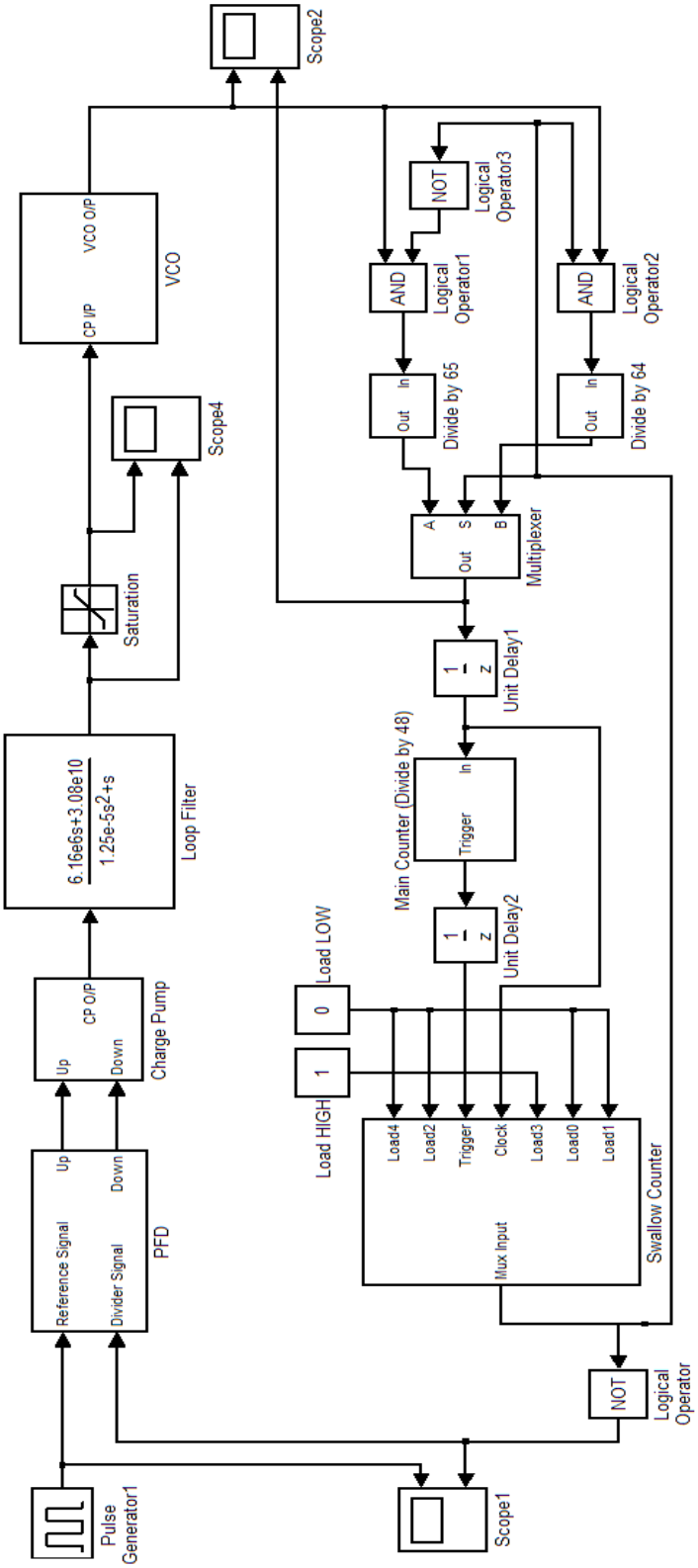


Fig. 3.17. Complete GSM FS Model (please rotate document to view)

With a changed K_v to 10 MHz, the quiescent frequency f_q becomes 914 MHz. This is used as the new model, as in this case the voltage across the VCO is restricted to below 2 V which is more relevant to present day CMOS technology. Since for stability, the new value of $\omega_{3_{dB}}$ will only be more with the already derived value of Z_h (from Sec. 3.2.1).Consequently, the value during the locked was as follows:

1. Locked C/P Output = 1.24 V; corresponding to f_o = 926.4 MHz
2. Expected Locked C/P Output = 1.3 V; corresponding to f_o = 927 MHz
3. Settling Time = $9 \times 10^{-4} s$
4. Locking Range = 0.3 V; corresponding to f_o = 923.4-926.4 MHz
5. Frequency Offset at output = 600 KHz

There is no explanation from the circuit level as to why this offset should occur. However, the offset can be explained with the use of event-driven simulation defect analysis as will be dealt with in a subsequent section.

3.5.3 Simulation of the Zigbee FS

The Zigbee FS is simulated with the design values derived in Sec. 3.2.2. The 2/3 cell divider architecture was used to simulate the Zigbee FS. The entire model is shown in Fig. 3.19. The simulation settings are Range-Kutta ode45 solver and automatic timesteps evaluation. This choice of simulation environment will ensure that mixed-mode simulation occurs. Lock condition is again taken as the criterion as stability in the output VCO frequency. The locked condition is shown in Fig. 3.18.

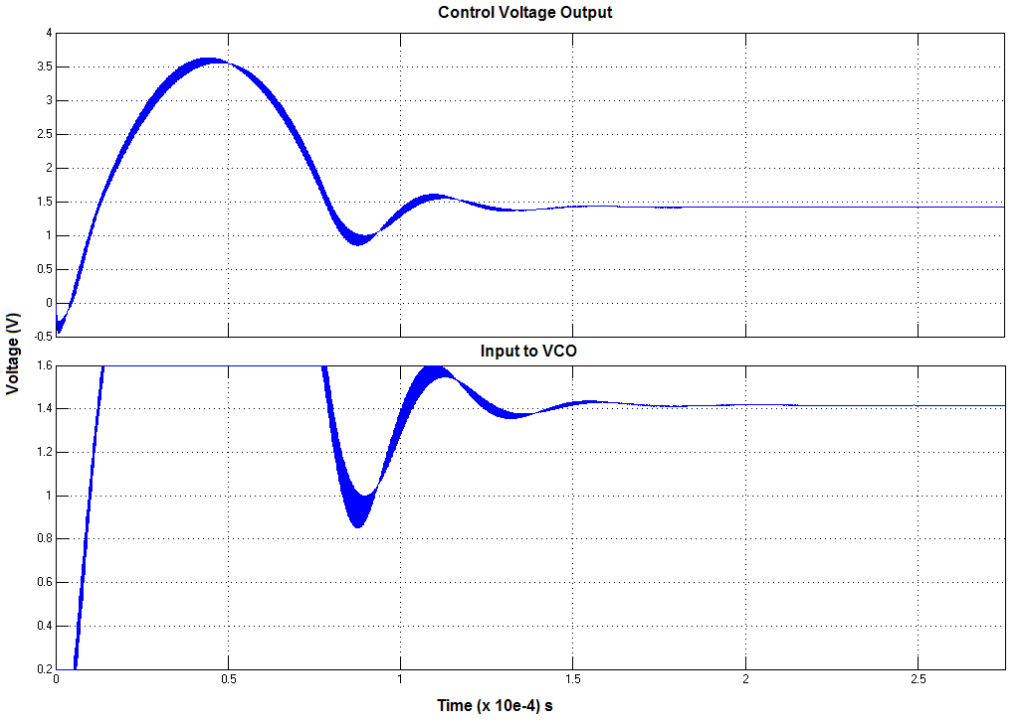


Fig. 3.18. Locked Condition in Zigbee FS

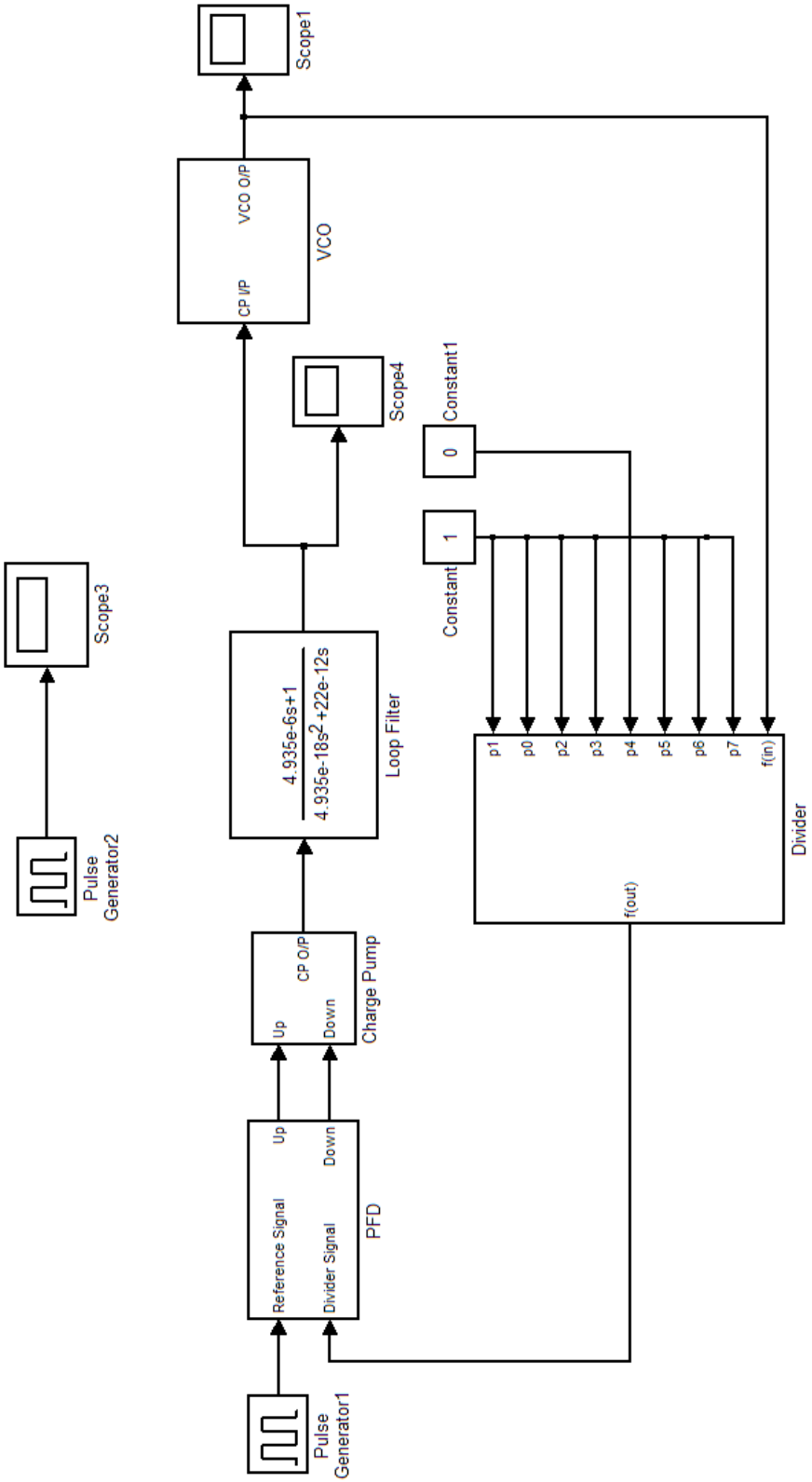


Fig. 3.19. Complete Zigbee FS Model (please rotate document to view)

The Zigbee Range does not show any offsets after simulation. This can be attributed to the lesser amount of feedback in the divider architecture and therefore lesser defects due to the queuing of events. Also, note that the simulation results show the CP output voltage being negative for a short period of time in the transient. This is because there is no first output for the divider, due to which the f_{REF} is always ahead of f_{fb} for a while, giving a negative voltage. The values of the parameters during the locked condition are as follows:

1. Locked C/P Output = 1.4167 V; corresponding to f_{o} = 2.475 GHz
2. Expected Locked C/P Output = 1.4167 V; corresponding to f_{o} = 2.475 MHz
3. Settling Time = $2 \times 10^{-4} s$
4. Locking Range = 1.5 V; corresponding to f_{o} = 2.40-2.48 GHz
5. Frequency Offset at output = 0 MHz

Since due to the inherent accuracy of this model and the corresponding divider architecture (2/3 cell one), we shall use the Zigbee FS for further modeling. Also, if the modeling is successfully built in the event-driven simulation environment, it can be compared with simulation results in other circuit models such as Cadence, using results from a chip developed in the Advanced VLSI Design Laboratory, IIT Kharagpur, India [14].

3.6 Event-Driven Simulation Effects

As observed in Sec. 3.5, the GSM FS shows a lot of offset when simulated in the event-driven mixed mode simulation environment. However, the Zigbee FS does not. In this section, we shall attempt to describe the deficiency in terms of the event-driven queuing of events and propose a solution to avoid such situations in the future.

3.6.1 Description of the Problem

As mentioned before, when the rate of queuing of events exceeds the rate of computation of events, there will be certain defects. Consider the case of the GSM divider shown in Fig. 3.11. The divider has to contain two delays for correct operation. However, an interesting point is noted while evaluating the divider alone (without the VCO and other sub-blocks). When we see the waveform for the divider output (f_{fb}) with respect to reference frequency (f_{REF}), it is always ahead of f_{REF}. This provides a more negative voltage at the CP output thus giving some offset at the VCO output. The waveforms are shown in Fig. 3.20.

It is noted that the divider output is ahead as the divide-by-65 start computation slightly ahead of when it is actually supposed to. That is, the event for the divide-by-65 is queued before the end of the computation of the various events in the last divide-by-64 cycle. An explanation for this phenomenon can be seen by observing the timing diagram for the queuing of events (Fig. 3.21). The explanation is as: the swallow counter sends a trigger for the prescaler to toggle earlier than required. This implies that at the 48$^{\mathrm{th}}$ count the main counter finishes division earlier than expected. This implies that the computation of divide-by-64 is not complete before the trigger arrives for the system to toggle the division ratio, due to which there is an advance in the occurrence of the divider output rising edge.

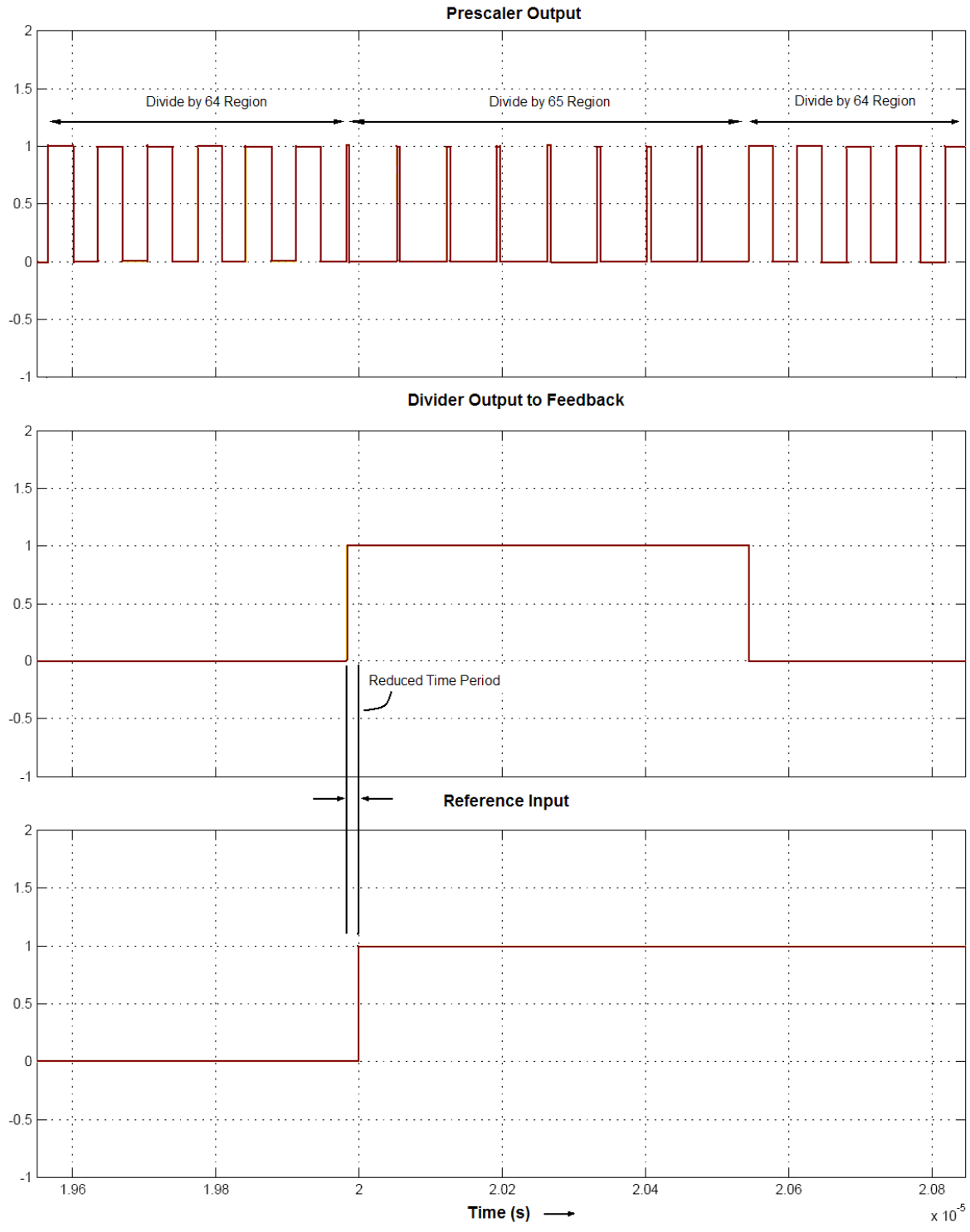


Fig. 3.20. Scope plot of Defective Divider Output

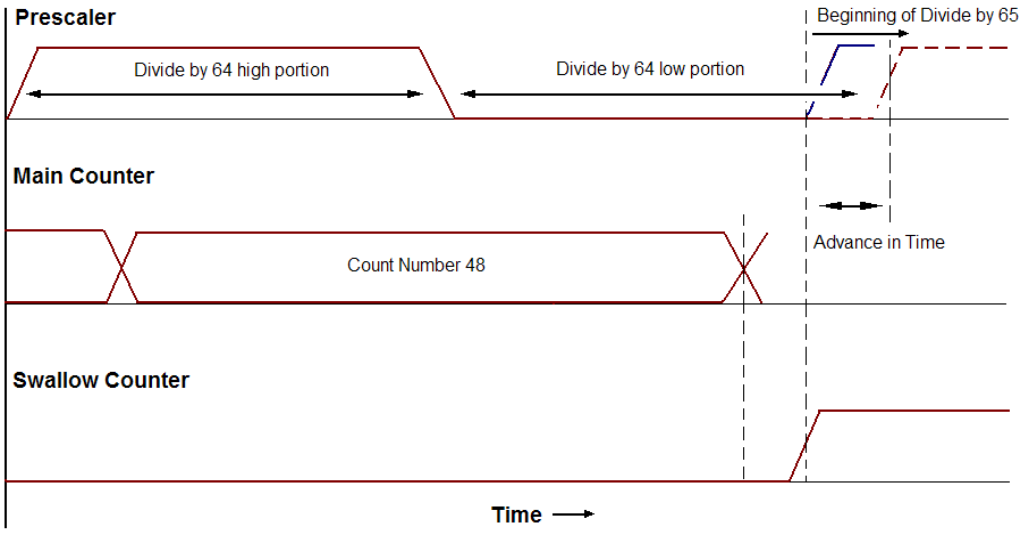


Fig. 3.21. Timing Diagram of the Prescaler based Divider

This phenomenon is found to be independent of the digital word, and hence shows it is a function of the divider only and not the other components in the PLL. That is, the offset is always 600 KHz for all A = 8 to 18. We will now present a solution to this phenomenon and discuss its effects on the final FS output.

3.6.2 A Solution

One solution to the problem is to prevent the onset of the trigger event as much as possible. This can be done by stopping the main counter to finish the last count early and give an indication to the swallow counter. This can be further implemented as a delay in the trigger to the main counter from the divide-by-64 stage as shown in Fig. 3.22.

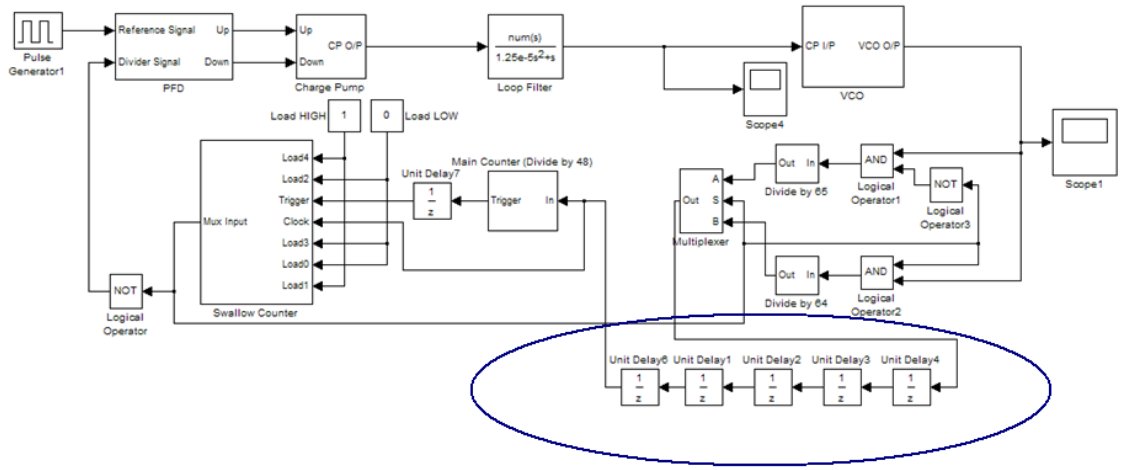


Fig. 3.22. Introduction of Delays in the Trigger Stage

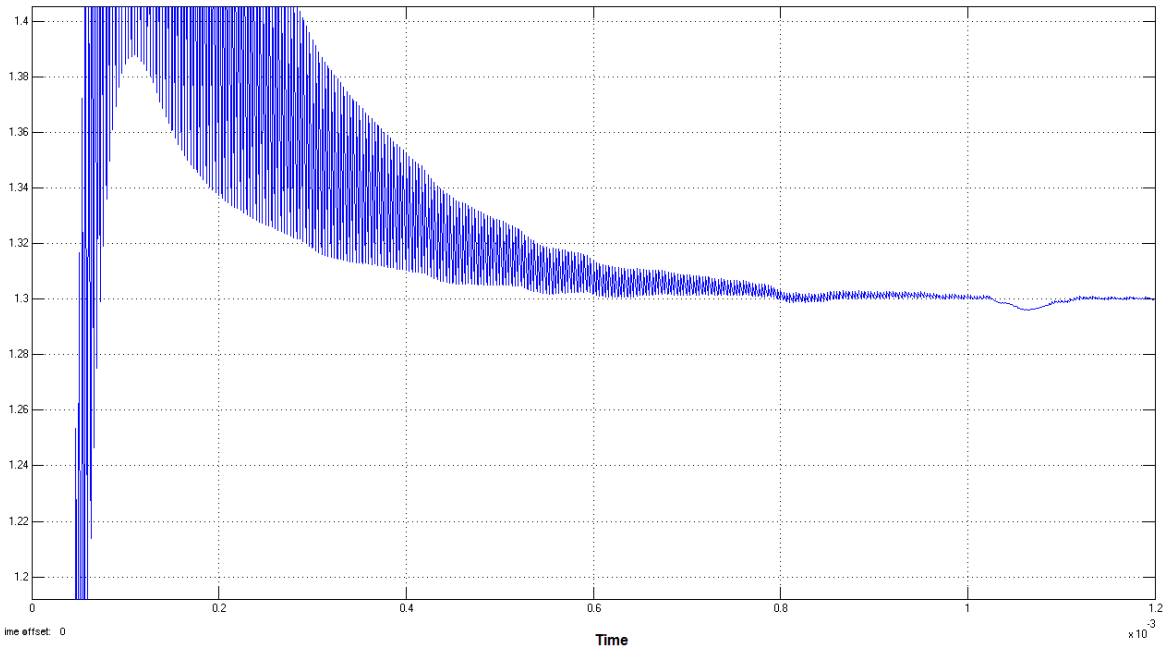


Fig. 3.23. Final output of the CP in the modified model

The above model was used in place of the model in Fig. 3.17. Five delays were added between the divide-by-64 stage and the main counter. It is to be noted that adding delays to the other path (i.e. the one between swallow counter and main counter) will not affect a change in the delay of the clock between the divide-by-64/divide-by-65 stage and the main counter/swallow counter.

We now observe the output of the charge pump to the VCO. It is seen that the delay tries to push the VCO output to the actual value of 1.3 V whenever there is a possibility of negative charge accumulating in the charge pump. As seen above, the voltage is around 1.3 V for a frequency output of 927 MHz. However queuing still produces a lot of defects in the final CP output, and the frequency hovers around the final value. For example, notice the sudden deep null at around $1.07 \times 10^{-3} s$ in Fig. 3.23.

As a future direction, some more analysis may be performed on this phenomenon of queuing and frequency offset to get a better picture of the final dynamics of the VCO.

3.7 Conclusions for Chapter 3

This chapter has focused on the modeling of a PLL and in general a frequency synthesizer in an event-driven simulation environment. Many concepts of Chapter 2 were used to further improve the design and in general analyze the effects of such a modeling.

We generally focused on the modeling of two particular designs, one on the GSM protocol and the other on the Zigbee protocol. Most of the modifications in these protocols existed in the construction of the divider architecture and the design philosophy. Further, the final designs of the

GSM protocol based FS was improved using the concepts of event queuing and time of computation taken into account. Generally FS is modeled using the frequency domain nature of the various components, i.e. the loop filter is low pass, the VCO is high pass etc. However, in this paradigm, the approach has been to equate the concepts of band passing to the concepts of the mixed mode simulation specifically in terms of $f_{\max}$.

We now move further on to consider the case of noise. Noise is a very important component of FS (detrimental but important!) in terms of modeling it and considering the same for design. Sudden noise changes affect receivers the most and the modeling of noise is of prime importance to the design of good receivers and to keep the Bit Error Rate (BER) at acceptable levels.

Chapter 4

Noise Modeling

4.1 Introduction

The primary problems faced in the design of PLLs is the high amount of noise that has to considered generally in order to make the receivers work at low power, high efficiency and small BER. Further, noise phenomena are very difficult to model and with the introduction of circuits, noise effects are highly non-linear and involve very complicated analysis. Further, current noise models are mainly in the frequency domain, or if modeled in the time domain are not available in the closed form.

This chapter examines some noise models, and also proposes methods to model this noise into the PLL design and the event-driven simulation environment proposed and discussed in the previous two chapters.

4.2 Noise Modeling for CP and Loop Filter

The noise model for the charge pump is simply the effect of dead-zone on the final output [12]. The dead zone occurs during the locking condition, when there is nearly no current pumped into the loop filter. At this time, the MOSFETs that pump the current and in the PFD work at near zero voltages and hence they have non-linear current around the zero-current region (see Fig. 4.1 for a general picture of the dead zone).

However, the effects of the dead zone are not very visible in the final output, and dead zone can easily be covered up by locking the VCO at a voltage reference greater than 0. Therefore, in our PLL model we do not consider the modeling of the dead zone. Further, there also exists a current noise in the PLL that inputs thermal noise into the charge. However, we realize that all high frequency phenomena are subdued by the loop filter which is essentially low pass. Hence we also avoid the modeling of the noise source parallel to the current sources. Fig. 4.1 shows a slight non-linear kink at the output of the PFD which is directly reflected as an extra charge in the output of the charge pump.

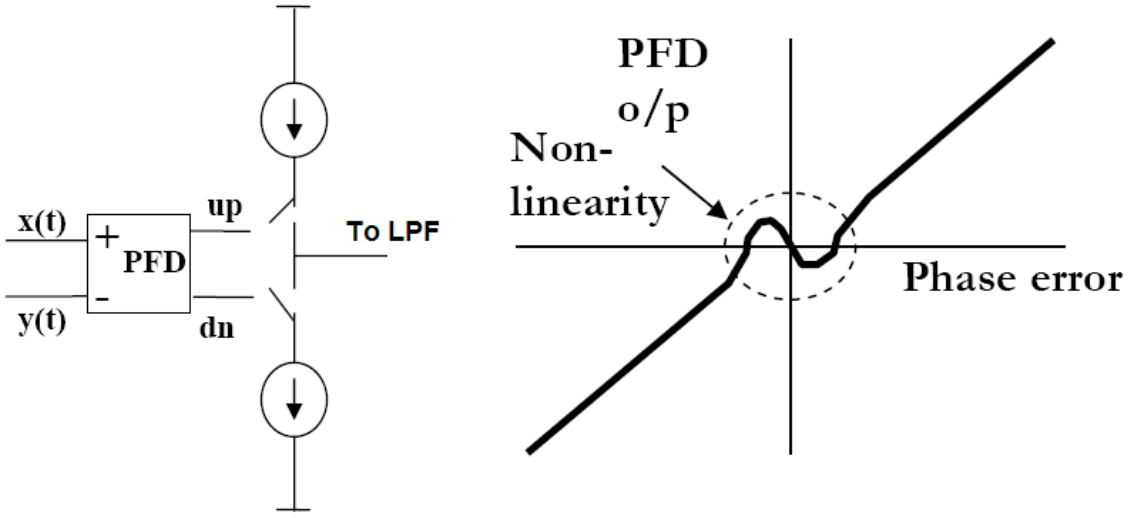


Fig. 4.1. Dead Zone in the PFD and non-linear effects

Since the loop filter is made up of resistances (generally only one resistance for the 2nd order), the resistance also has a noise voltage with an average voltage $\sqrt{4kT\Delta fR}$. This voltage is also avoided in the modeling as it is also low pass by the effect of the filter. Further, there is no qualitative improvement in the noise plot of power spectral density (PSD) and spectrum, and only a raise in the thermal levels is seen [16].

Hence, we avoid the modeling of the noise and non-idealities of the PFD, the charge pump and the loop filter in the Simulink model. We will now focus on the modeling of the noise in dividers, the VCO and the source voltage, V_{dd}.

4.3 Noise Modeling in the VCO

Of all components of the PLL, the noise in the VCO is of prime importance, as it is not filtered further and due to its highly non-linear nature can completely affect the dynamical trajectory of the VCO. In this section we will consider the VCO noise modeling in detail and its adaptation to the discrete and event-driven simulation scenarios.

4.3.1 Adding Non-idealities in the VCO

To add non-idealities to the simulation environment in the VCO the best possible method is to add them in the VCO charge pump input. This can be done before or after the ramp input in the VCO as shown in Fig. 4.2. Fig. 4.2 shows two noise generators, one for the V_{dd} and the other for the oscillator noise.

The addition of the phase noise at this stage is justified as, the phase ϕ is generated at this stage after the ramp function $1/s$. Therefore adding $\Delta\phi$ or Δf at this stage is perfectly compatible with the matching of the phase error values obtained by comparing edges.

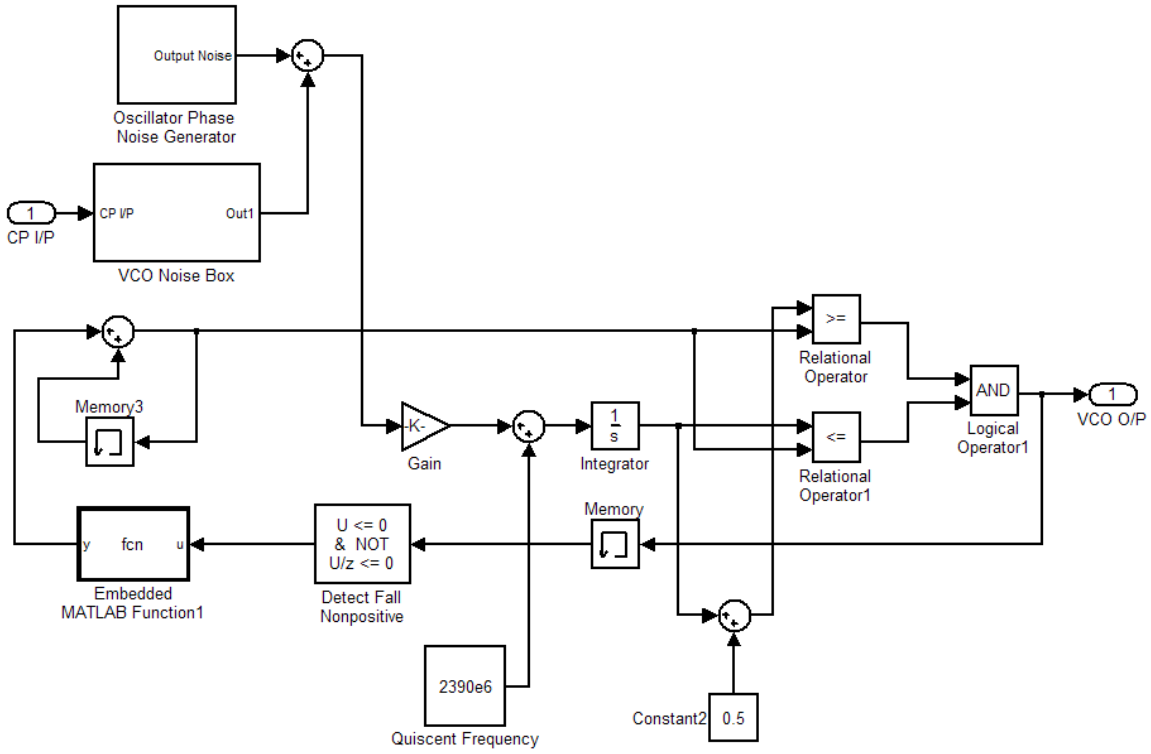


Fig. 4.2. Adding noise generators in VCO CP Input

4.3.2 Popular VCO Noise Models – Leeson's

The most popular model for VCO noise is the linear time-invariant model for noise proposed by Leeson [4]. This model is an empirical model and gives an expression for the single sideband power spectral density $\mathcal{L}$. As the phase noise is of prime importance around the chief angular frequency of the VCO, say ω_0, $\mathcal{L}$ is measured as an offset from the fundamental frequency $\mathcal{L}(\Delta\omega)$.

Leeson provided an expression for $\mathcal{L}(\Delta\omega)$ as:

$$\mathcal{L}(\Delta\omega) = 10 \cdot \log\left\{\frac{2FkT}{P_s} \cdot \left[1 + \left(\frac{\omega_0}{2Q_L\Delta\omega}\right)^2\right] \cdot \left(1 + \frac{\Delta\omega_{1/f^3}}{|\Delta\omega|}\right)\right\} \tag{4.1}$$

where, F is the noise factor of the device, P_s is the power dissipate in the resistive part of the tank, Q_L is the quality factor of the tank, k is the Boltzmann Constant, T is the temperature, $\Delta\omega$ is the frequency offset and $\Delta\omega_{1/f^3}$ is the offset frequency before which the device flicker noise dominates in the noise spectrum [11].

The above model is a simple empirical representation of the observations of the sideband spectrum of the VCO output. The properties of F are highly uncertain and are usually used to fit the model after the observations. The model is used in most practical purposes for analysis of the phase noise as it is of closed form.

A more accurate model is the time variant Hajimiri model which we shall briefly discuss soon. Fig. 4.3 shows in a nutshell the characteristic of the curve formed by the equation (4.1). The

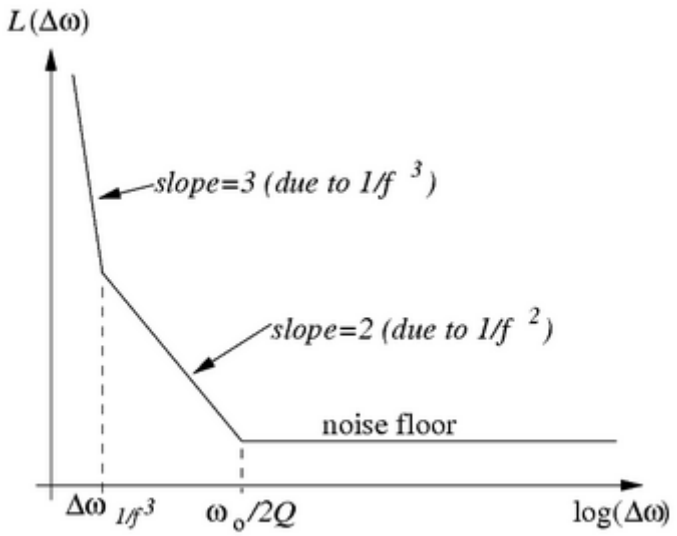


Fig. 4.3. Leeson's Phase Noise Spectrum

measurement is made beyond the offset frequency and the phase noise is observed to have three distinct regions, a flat thermal noise floor part, a section with $1/f^2$ noise characteristics (which is essentially upconverted thermal noise), and a section with $1/f^3$ noise characteristics (which is essentially upconverted flicker noise).

4.3.3 Popular VCO Noise Models – Hajimiri's

A more recent phase noise model was developed in [5] by Hajimiri and Lee. This model assumes no posteriori fitting with the parameter F in the phase noise spectrum and is a linear time-variant model. This means, that the phase noise is capable varying with time and is not purely described by the spectrum. The model is more accurate but less useful practically as due to its time variant nature it is not easy to manipulate in the frequency domain and a closed form expression is not available.

The Hajimiri model explains the phase addition to an oscillation in terms of an impulse of noise. The system adds a constant charge with a periodic response with the magnitude of the charge depending on where the impulse is applied. The corresponding phase error response is given by:

$$h_\phi(t,\tau) = \frac{\Gamma(\omega_0\tau)}{q_{\max}} u(t-\tau) \tag{4.2}$$

Where the function Γ is the periodically varying response specific to an oscillator, and $q_{\max}$ is the maximum noise charge injected. The corresponding phase error can be calculated as a function of the current impulse input:

$$\phi(t) = (1/q_{\max}) \int_{-\infty}^{t} \Gamma(\omega_0\tau) i(\tau)\, d\tau \tag{4.3}$$

This can be extended to random noise sources, such as in white noise. Although this model is very popular for design, we shall use the Leeson Model as definite regions of response are defined.

4.3.4 Measuring Phase and Eliminating Numerical Noise

Till now, we have discussed the removal of t_{ADC} and $t_{\mathrm{timestamps}}$ by considering a DVCO architecture and considering the measurement of the falling edge (due to the event-driven simulation happening at that edge) in the simulation environment. However, we have not discussed the removal of $t_{\mathrm{numerical}}$.

Consider the simulation of the ideal VCO shown in Fig. 2.8. We simulate the above structure in Simulink and store the corresponding samples in the MATLAB workspace. This workspace has the timestamp vector and the corresponding logic level. Since from Fig. 2.9, we understand that measuring time jitter at the falling edge will eliminate $t_{\mathrm{timestamps}}$. Hence, we store the points in the timestamp vector where the 1-logic level becomes 0-logic level and compare it with a reference.

The choice of the reference is critical in eliminating $t_{\mathrm{numerical}}$. If we choose the reference to be n/f_0 an oracle reference given we as the experimentalists can know f_0, we can isolate the numerical noise. With this kind of measurement, Fig. 4.4 shows the numerical phase noise arising (by taking simple Fast Fourier Transforms (FFTs) and squaring. The nature of the noise is $1/f^2$, the reason for which will be discussed shortly. The phase noise is considerable and may affect the characteristic nature of the measurement of t_{jitter} (from (2.3)). Therefore, the numerical noise should be avoided.

A simple method to avoid the numerical is to consider measurement using a difference of the phase vectors of the Ideal and the Noisy VCO, as shown in a corresponding model in Fig. 4.5. Both the scope outputs are stored in the MATLAB and the falling edge timestamps are compared.

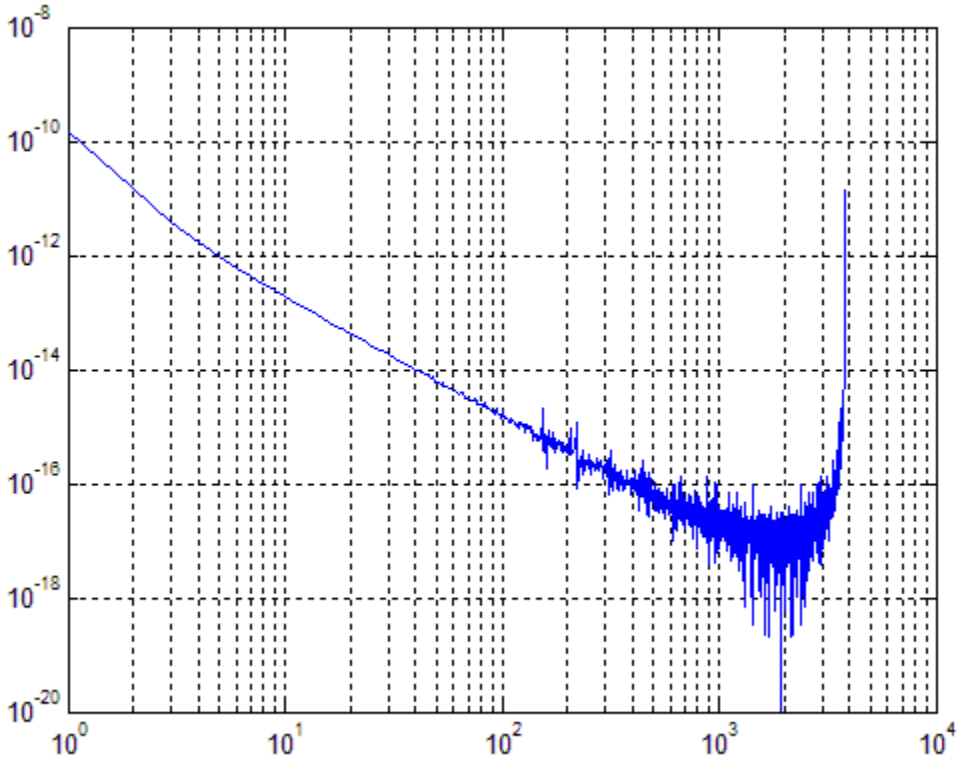


Fig. 4.4 Numerical Noise – $t_{\mathrm{numerical}}$ phase noise plot

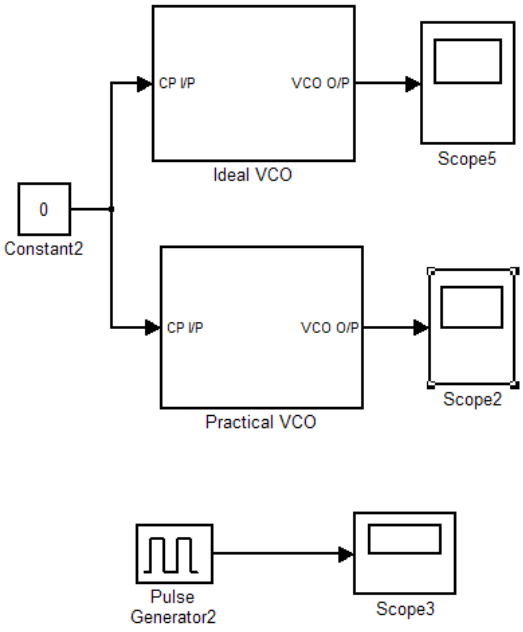


Fig. 4.5. Phase Measurement Experimental Setup

4.4 Model for Source and Oscillator Noise

In this section we will consider the models for source noise and the oscillator noise. These are the noise models that can be essentially added to the charge pump output, going into the integrator of the VCO. We shall present some specific models for these noise sources and generators and also see the difference in simulating them in the discrete environment and the continuous mixed-mode environment.

4.4.1 Noise in the Source Voltage

The voltage in the source is modeled as a sinusoidal voltage with a normalized amplitude of given by $\Delta V_{\mathrm{DD}}/V_{\mathrm{DD}}$ [1] and a proportionality factor for the modeling of the frequency of the sinusoidal signal Δf_0 given by $\alpha = (\Delta f_0/f_0)\,/\,(\Delta V_{\mathrm{DD}}/V_{\mathrm{DD}}) = 0.1$. The value of Δf_0 can be up to 60 GHz for general sinusoidal noise in sources [17].

The final model for the Source Noise is an additive sinusoidal noise added to the Charge Pump output as shown in Fig. 4.6. The Hajimiri model presented in subsection 4.3.3 predicts that there will be an impulse on the spectrum at offsets of $\Delta\omega$ will appear in the phase spectrum due to a sinusoidal noise source. The Single sideband PSD of the phase shown in Fig. 4.7 confirms the corresponding theory. The plot shows a spike in the PSD at around the 10^2 of the FFT PSD plot.

The source noise can also be modeled to obtain an expression of the PSD as given in [1]. However, this is to verify the model with theoretical expressions. Since we have used nearly the exact model, we assume that the theoretical expressions in [1].

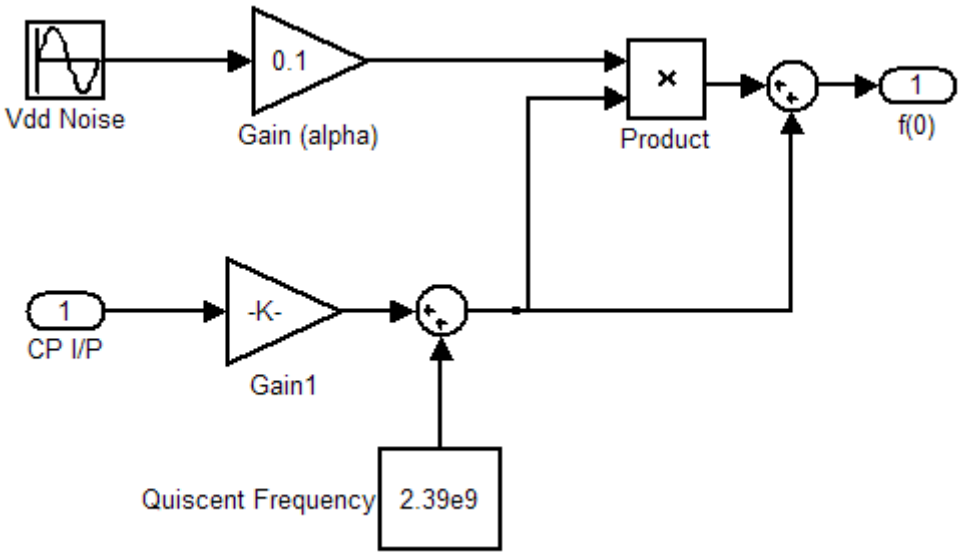


Fig. 4.6. Source Noise Generator Model

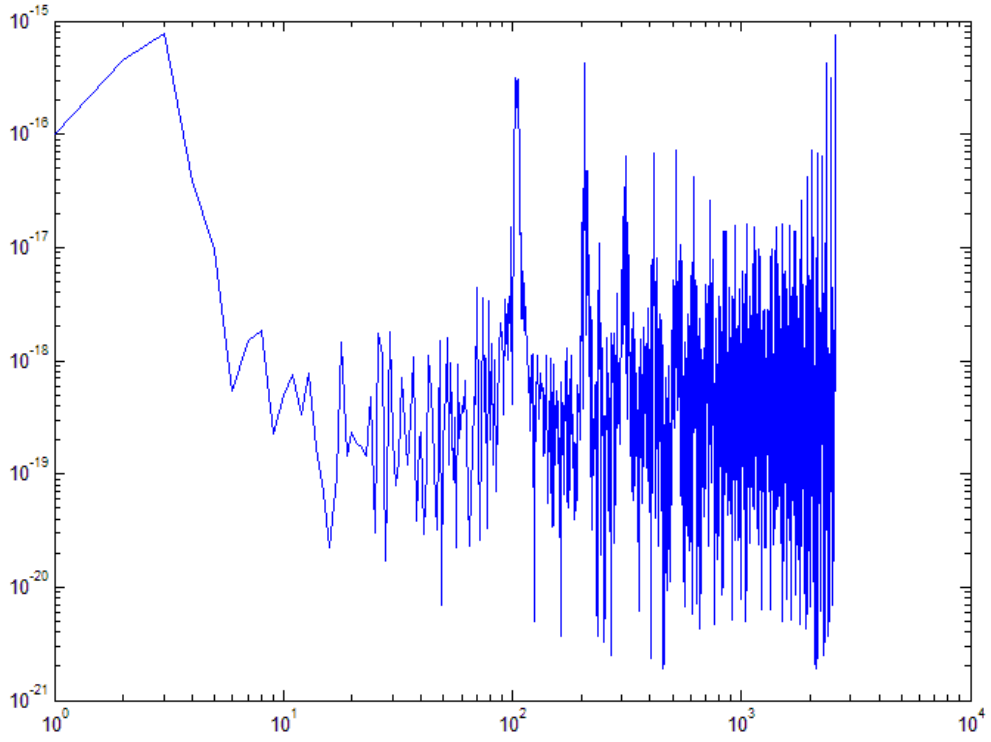


Fig. 4.7. Source Phase Noise Discrete PSD Plot

4.4.2 Discrete Simulation of Thermal Noise Components

We now embark on to model the phase noise of the VCO using the Leeson Noise Model as a reference. It is useful to consider some time domain discrete models for the various portions of the PSD plot shown in Fig. 4.3. According to the theory presented in [4], the thermal noise floor and the upconverted $1/f^2$ noise is a result of random thermal noise and random accumulative thermal noise.

Suppose the value of the time jitter is $t(i)$ for the ith clock pulse. This implies that the corresponding thermal noise model for any clock pulse is:

$$t_{jitter} = t(i) \quad \text{for thermal noise floor} \tag{4.4}$$

$$t_{jitter} = t(i) + t(i-1) \quad \text{for} \quad 1/f^2 \text{ noise} \tag{4.5}$$

It has been shown in [18], that the accumulative jitter described by (4.5) gives a $1/f^2$ plot in the phase PSD. Therefore a simple way to model the above would be to use random number generator and use a FIR filter to generate (4.5), as shown in the model in Fig. 4.8. The corresponding discrete phase PSD is shown in Fig. 4.9 showing a $1/f^2$ characteristic.

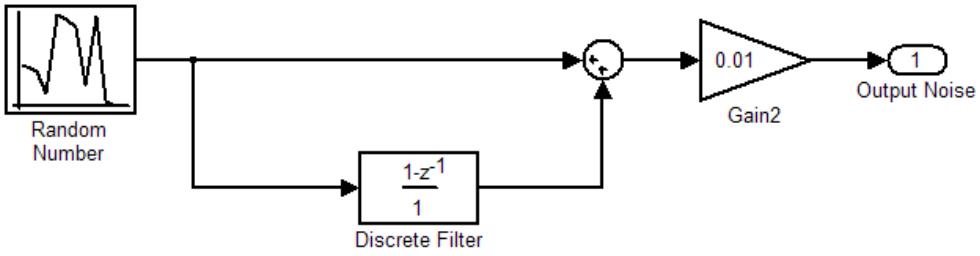


Fig. 4.8. Model of Thermal Noise Generator

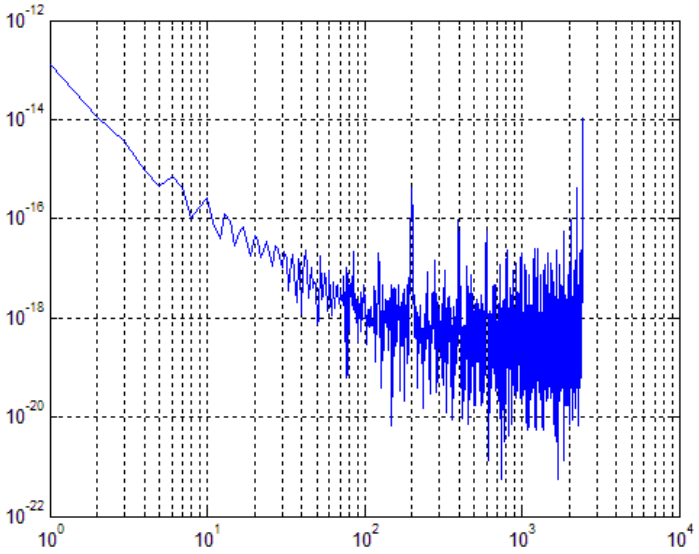


Fig. 4.9. Thermal and Upconverted Thermal Noise PSD

The accumulative jitter is also responsible for the numerical error plot earlier described using Fig. 4.7. The numerical error generally accumulates over periods of evaluation and assuming that the error is at least pseudorandom, we obtain the characteristic curve in Fig. 4.7.

4.4.3 Discrete Simulation of Flicker Noise Components

The discrete modeling of the flicker noise components is not an easy task, as creating naturally the $1/f^3$ is not possible with any digital FIR or IIR filter with finite number of coefficients. Therefore, we resort to a technique proposed in [19] that uses a bank of IIR filters with multiple cutoff frequency spread over a range of frequencies to emulate the $1/f^3$ characteristic. For an image see Fig. 4.10 [19].

The plot shows the emulation of the *device flicker* noise which has a $1/f$ characteristic plot in its PSD. This plot when upconverted will give the $1/f^3$ characteristic in the phase noise spectrum. To produce the bank of IIR filters, shown as a model in Fig. 4.11, we will first create $k = 6$ filters between 100 Hz and 1 MHz. A recursive formula for the IIR filters would be the following [19]:

$$y_k[i] = (1 - a_k)\, y_k[i-1] + a_k A^{-(k-1)} x[i] \tag{4.6}$$

which gives a $z-\text{domain}$ representation as:

$$\frac{Y(z)}{X(z)} = \frac{a_k A^{-(k-1)}}{1 - (1 - a_k) z^{-1}} \tag{4.7}$$

The values of a_k are given by $a_k = 2\pi f_{c,k}/f_s$ where $f_{c,k}$ is the frequency of cutoff chosen by the designer and f_s is the sampling frequency. A is such that slope = 10 dB/decade = A_{dB}/r where $r = f_{c,k+1}/f_{c,k}$. With $k = 6$ and the frequencies to range from 100 Hz to 1 MHz, we get $r = 10$.

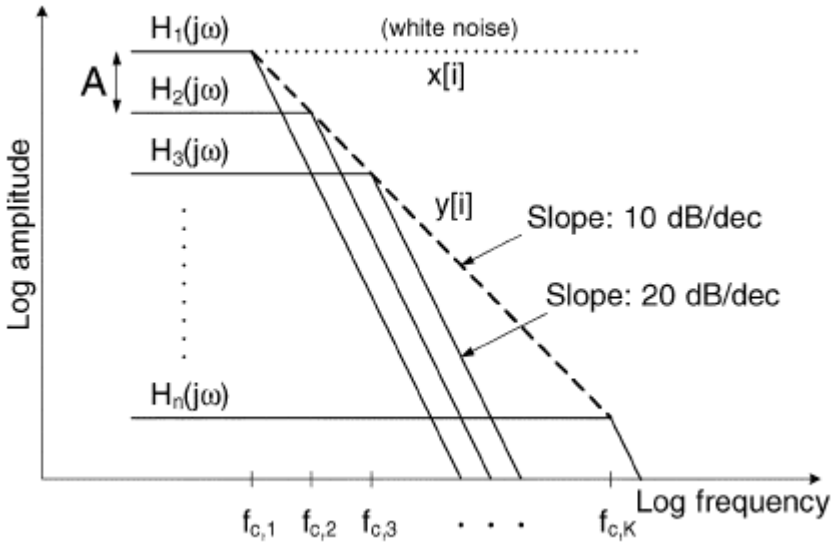


Fig. 4.10. Bank of IIR Filters for $1/f^3$ noise generation

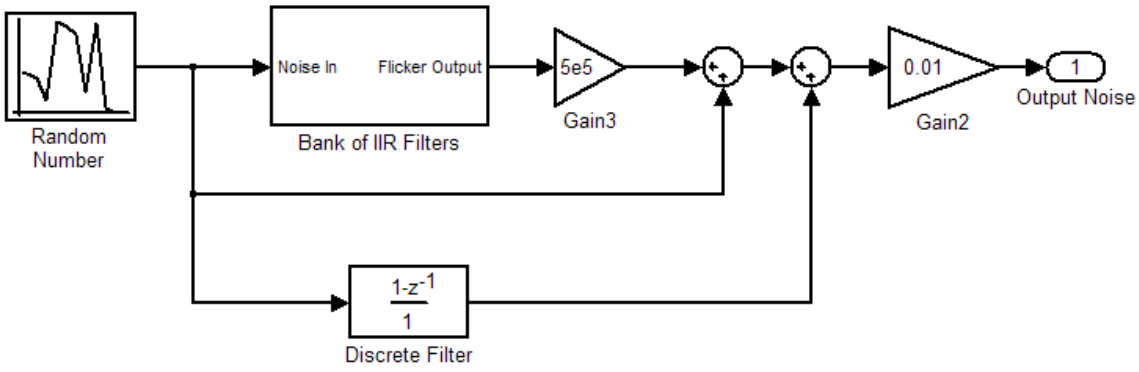


Fig. 4.11. Bank of IIR Filters Model

The responses of the filter are all added to get a bank of filters, whose response is then added to the discrete filter and the thermal noise. For example, with the above parameter values, we obtain for the filter with cutoff between 100 KHz and 1 MHz:

$$y_6[i] = (0.994)y_6[i-1] + \left(1.9868 \times 10^{-5}\right) x[i] \tag{4.8}$$

The random value $x[i]$ is the same random number generator. With some gain adjustments on the individual phase noise components, we obtain plot for the phase noise with the $1/f^3$ region included as shown as a digital phase noise PSD in Fig. 4.12.

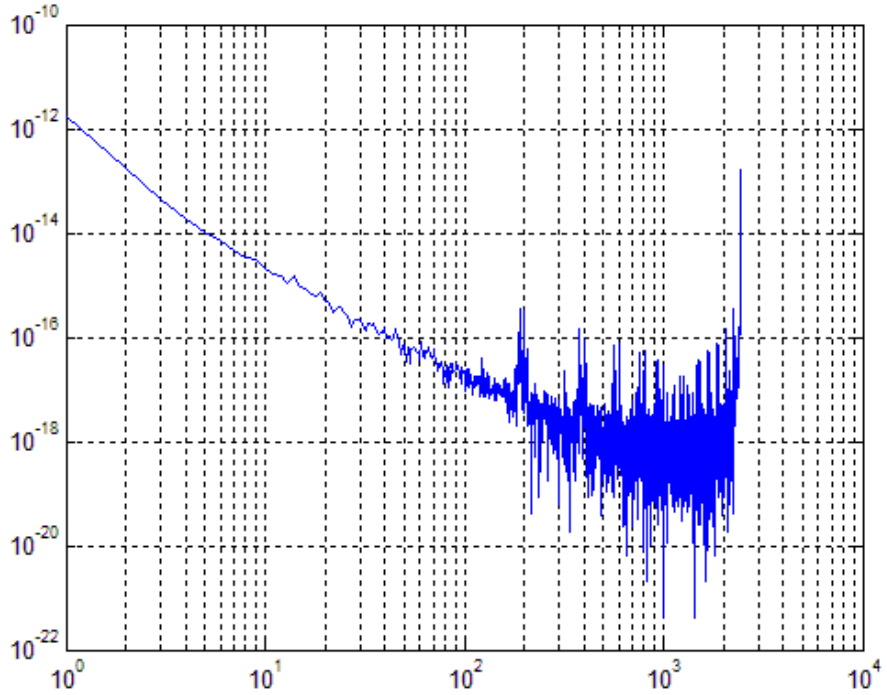


Fig. 4.12. Upconverted Flicker Noise PSD

The plot in Fig. 4.12 shows a $1/f^3$ region in the initial part of the graph. Since the plot is highly qualitative, and we are not very concerned about discrete simulation, we use the ideas developed about for developing continuous domain models for event-driven simulation of phase noise characteristics.

4.4.4 Event-Driven Simulation of Thermal Noise Components

We have asserted before in Section 2 that all measurements of phase has to be done in the continuous event-driven simulation mode and hence the falling edges time stamps have to be taken into account. However, the discrete models above work only for discrete time steps. Therefore, corresponding analog models have to be developed for the phase noise characteristics. We now present our, the first known, analog phase noise models for incorporation into event-driven simulation environment.

There is a need to demonstrate the measurement of phase noise in the rising and falling edges before the modeling of the same. $t_{\text{timestamps}}$, the timestamp vector is now completely described by (2.6) and (2.7) for the DVCO, and t_{ADC} has been eliminated by the use of the digital VCO. Phase noise is measured as a difference in the timestamps of an ideal VCO and a practical VCO. This will also eliminate any quantization errors due to rounding off. Also, phase noise *cannot* be measured using the positive edge, as discrete timestamps will completely overshadow the changes in the edge due to phase error. Thus phase error vector is given as

$$\Delta\phi = \left(T_{FE/\text{noisy}} - T_{FE/\text{ideal}}\right) \cdot f \tag{4.9}$$

where f is the output frequency of the VCO. f can be calculated from the vector of T_{FE} using the linear regression method as shown in [1] or the known required frequency of operation of the synthesizer. The formula using the linear regression is (with $f = 1/T$):

$$T = \frac{\sum_{i=0}^{n-1} i t_i - \frac{n-1}{2}\sum_{i=0}^{n-1} t_i}{\frac{(n^2-1)n}{12}} \tag{4.10}$$

Thermal phase noise is generated using a random Gaussian number generator (an analog simulation block) of amplitude V_G and variance $\sigma_{V_G} = 1$. Without using digital filters (as they inherit a discrete simulation time) like in the subsections above, we can construct the phase noise expression for the sum of the noise floor $(\Delta\phi_0)$ and the upconverted thermal noise $(\Delta\phi_1)$ as

$$\Delta\phi = \Delta\phi_0 + \Delta\phi_1; \text{ where } \Delta\phi_0 = (K_0 V_G) \text{ and } \Delta\phi_1 = \int_0^t (K_1 V_G)\ dt \tag{4.11}$$

A detailed derivation of the nature of K_0 and K_1 is given below. The above can be incorporated as one random number generator, with zero mean and unity variance, and two gain blocks.

For standard receivers, the noise floor is zero mean Gaussian random process with variance $\sigma_0 = 1/(2\pi)\sqrt{\mathcal{L}f}$ where $\mathcal{L}$ is the sensitivity of the receiver, and $T = 1/f$ is the frequency [19]. The total output phase of the VCO is given by:

$$\phi_0 = \int_0^t (f_0 + 2\pi V_{CP} K_v)\ dt + \sigma_0 V_G = \int_0^t (f_0 + 2\pi V_{CP} K_v) + \frac{\mathrm{d}\left(\sigma_0 V_G\right)}{\mathrm{d}t}\ dt \tag{4.12}$$

where f_0 is the quiescent frequency, V_{CP} is the charge pump output, K_v is the VCO gain and V_G is a zero-mean Gaussian random number with unity variance. In the $s-\text{domain}$, $\Phi_0(s) = ((f_0 + 2\pi V_{CP}(s)K_v)/s) + \sigma_0 V_G(s)$, giving $\Delta\Phi_0(s) = \sigma_0 V_G(s)$. Therefore, $K_0 = \sigma_0$, if it is assumed that the multiplication by the frequency f as per (4.9) is performed at a later analysis stage.

In the case of upconverted thermal noise, we model it as accumulative noise [19]. The accumulative jitter vector is given by

$$t_{\text{upconverted}}(n) = \sigma_1 V_G(n) + t_{\text{upconverted}}(n-1)\ ;\ \ n \in \mathbb{N} \tag{4.13}$$

With $\sigma_1 = \Delta\omega\sqrt{\mathcal{L}\{\Delta\omega\}/2\pi\omega}$, where, $\mathcal{L}\{\Delta\omega\}$ is the wander sensitivity of the receiver and $\Delta\omega$ is the wander frequency. For continuous timestep simulation at the falling edge, it is convenient to write t_{flicker} as

$$t_{\text{upconverted}} = \sigma_1 \int_0^t \frac{V_G}{T}\, dt \ \ \Rightarrow\ \ \Delta\Phi_1(s) = \frac{(\sigma_1 f)\cdot V_G(s)}{s} \tag{4.14}$$

Hence $K_1 = \sigma_1 f$ again assuming that the multiplication by the frequency f as per (6) is performed at a later analysis stage. It can be shown that the spectral density of $t_{\text{upconverted}}$ above follows the inverse square law with frequency offset [18], thus successfully modeling the upconverted thermal noise.

Using the two gain blocks as before a model can be created as shown in Fig. 4.13. This model uses the values $\sigma_0 = 33\ fs$ and $\sigma_1 = 5\ fs$, standard values for receivers in the ISM range. The phase noise plot is calculated the simple sum of discrete-time fourier transforms [10] (we cannot use the FFT, as the sampling is now event-driven). The formula for calculating the PSD is as follows:

$$\text{PSD}(f) = 10\log_{10}\sum_{l=1}^{n}\left|V_l\left(e^{j2\pi f}\right)\right|^2 \tag{4.15}$$

This formula can be used with a standard DSP window. However, it performs accurately well to analyze the model in hand. In the above model, V_l is the vector value in the phase error at index l. Using the above formula we may plot the value of the PSD as shown in Fig. 4.14 with frequency offset (or frequency as aliasing takes place anyway). The values of gains involving K_0 and K_1 derived were:

$G_0 = K_0 \cdot f = \sigma_0 \cdot f = (33\ fs)\cdot\left(2.39\times10^9\right) = 7.92\times10^{-5}$

$G_1 = K_1 \cdot f = \sigma_1 \cdot f^2 = (5\ fs)\cdot\left(2.39\times10^9\right)^2 = 28800$

4.4.5 Event-Driven Simulation of Flicker Noise Components

The event-driven continuous domain model for $1/f^3$ is not easy to derive. The author has not completely created a model for the same; however he has made some progress in the literature survey, analysis and the possible solution.

One possible solution to obtaining a noise plot of characteristic $1/f^3$ PSD is to find equivalent analog domain recursive filter banks similar to the bank of IIR filters derived in subsection 4.4.3. However, if we notice the Fig. 4.13, we see that we are now adding the phase noise directly to the *phase* output, after integration, and not before. Therefore, this gives us an opportunity to explore the

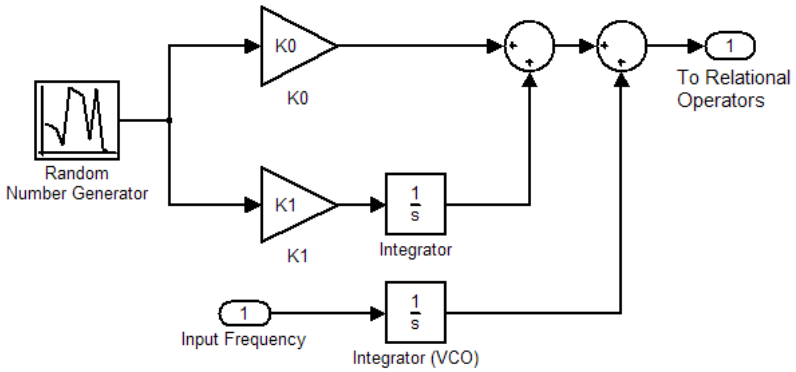


Fig. 4.13. Thermal Noise Generator Model for Event-Driven Simulation

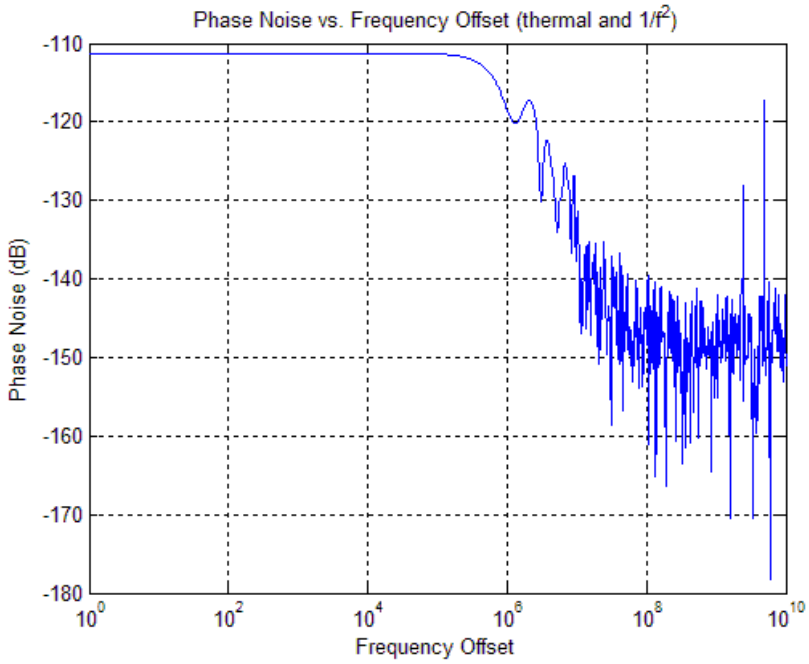


Fig. 4.14. Thermal noise PSD for Event-driven Mode

possibility of analyzing the jitter directly and hence create alternate models based on random numbers and probability distribution functions.

It is proposed in [20], that a random number with variance given by the following formula can produce a PSD with a $1/f$ curve:

$$\sigma(t) = 2\left(\frac{2\gamma_c t - 1 + e^{-\gamma_c t} - \gamma_c t e^{-\gamma_c t} + (\gamma_c t)^2 E_1(\gamma_c t)}{\gamma_c^2}\right) \tag{4.16}$$

$$E_1(t) = \int_1^{\infty} \frac{\exp(-tz)}{z}\, dz \tag{4.17}$$

where the value of γ_c is the cutoff point in the frequency where the $1/f^3$ characteristic is no more observed and the in-band noise dominates. When we use this time dependent $\sigma(t)$, to create the phase jitter variable:

$$\phi_{\text{flicker}}(t) = x(t) \cdot \sigma(t) \tag{4.18}$$

where $x(t)$ is the random Gaussian variable of unity variance. This variable when integrated will have spectral characteristics of $1/f^3$ nature. This can be shown by considering the following:

$$\text{PSD} \propto \frac{X(s) * \sigma(s)}{s} = \frac{K}{s^3} \tag{4.19}$$

Therefore, this method is an easy way to simulate the PSD. A direct implementation would involve a functional block in Simulink that implements $\sigma(t)$ and multiplies it with a random number and then goes on to integrate the same using a ramp function. The author will present the same in a paper to be submitted soon for review (see end of document for list of publications).

4.4.6 Comparisons of Performance with Chip

The phase noise performance of the models is compared with an actual Zigbee FS fabricated chip [14]. The data of performance of the chip is used here with permission of the authors of [14]. One must note that since the model presented about uses the Leeson's Formula (given by (4.1)), it is important to characterize the *posteriori* parameter F.

We start out with comparing the performance of the settling time. The settling time of the chip in [14] shows about $25\ \mu\text{s}$ for a change of about 75 MHz. Since we have evaluated the total settling time, for a change in frequency of about 60 MHz. The settling time is about $100\ \mu\text{s}$ as shown in Fig. 3.18. From analysis before we know that the 2/3 divider does not produce the first clock cycle due to which there is a slight extension in the time domain settling time. Overall, the orders of the settling time matches perfectly.

Given the matching performance in settling time, we now focus on the phase noise performance. The PSD of the phase $(S_\phi(\omega))$ from the model is shown in Fig. 4.14. The PSD of the amplitude $(S_V(\Delta\omega))$ is shown in Fig. 4.15 from measurements in the chip. The Fig. 4.14 is in dB and the Fig. 4.15 is plotted in dBc.

Since the event-driven model is available only for the region of $1/f^2$ noise (as the $1/f^3$ noise is still future scope), we compare with the chip only the phase noise for the region between 1 MHz and 10 MHz (see Fig. 4.15). This shows us that the frequency span for the two $1/f^2$ regions is exactly matching after which the thermal noise floor comes in. The ratio of the phase to voltage phase modulation (as shown in [5]) is about $30\ \text{dB}$ and this is created due to amplitudes of current and the tank Q values. Therefore as an approximation we may estimate:

$$\frac{2FkT}{P_s} \cdot \frac{1}{4Q_L^2} \approx 30\ \text{dB} \tag{4.20}$$

With this scaling, we see that the phase noise performance exactly matches with observed chip values and hence are verified. The verification of the $1/f^3$ will also be done in the near future.

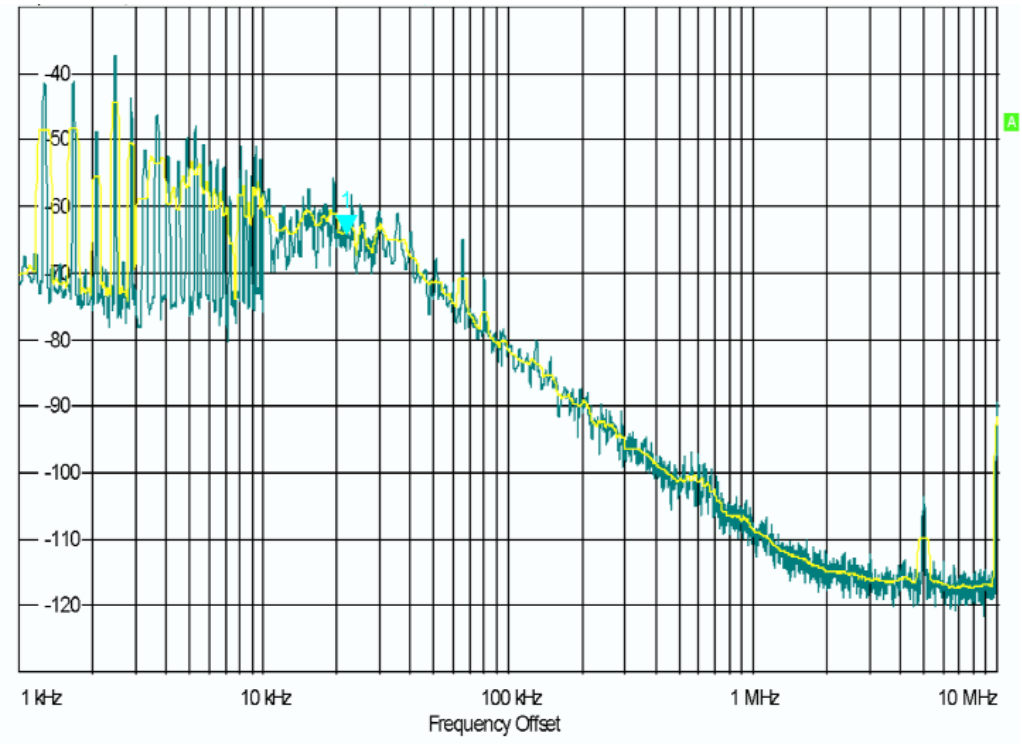


Fig. 4.15. Phase Noise Measured From Chip

4.5 Divider Phase Noise

The divider is a very power hungry system with multiple cascaded MOSFETs and the possibility of each of them producing noise. The divider performs as a RC system in contrary to the oscillator which is essentially a non-damped LC system. Therefore, the noise models for the divider are certainly different from those of the oscillator.

The divider noise model has been taken into account in [21] very carefully for the case of Source Coupled Logic (SCL) Technology. Coincidentally, the same technology is used in the chip developed in AVLSI Design Laboratory in IIT Kharagpur, India [14].

[21] examines the case of the divider to be a damped RC system, essentially for the SCL divide-by-2 flip flop. Again, coincidentally, this can be extended and shown to be the same for the cascaded AND and D-type Flip Flop based 2/3 divider architecture described in Section 3.4.2. Therefore, as far as the technology is concerned, it is highly relevant to the development and modeling of the Zigbee FS which we have been considering till now.

This section will briefly explain the author's approach on modeling the divider phase noise using the reference [21]. However, the model has not been created completely yet and again, will be ready before submission of a paper for review (see end of document for list of publications).

For the SCL D-type Flip Flop shown in Fig. 4.15 (with the reduced RC diagram shown on the right), the phase noise was derived to have the following expression [21]:

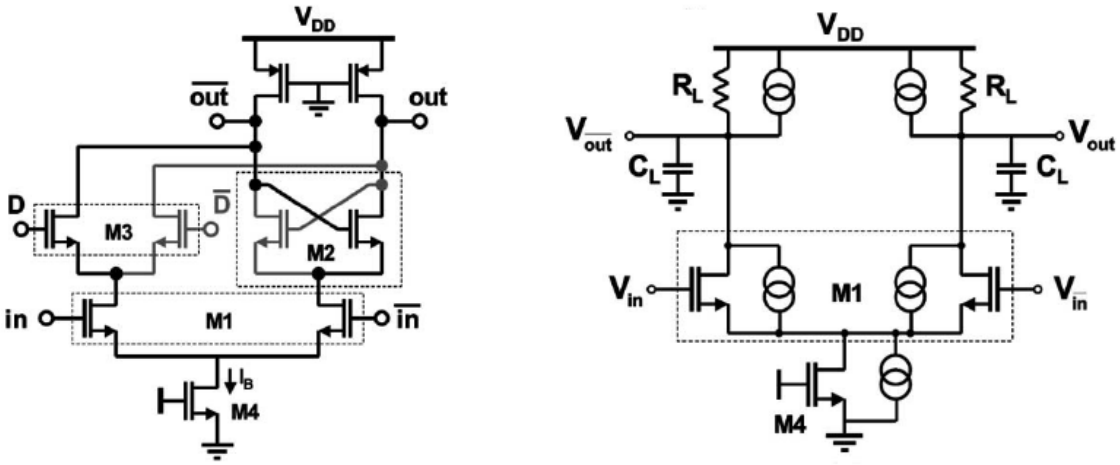


Fig. 4.16. SCL Divider Phase Noise Derivation

$$\mathcal{L}_W = 8\pi^2 \left(1 + \frac{\gamma}{\alpha} + \frac{\gamma_T g_{m_T} R_L}{2\alpha_T}\right) \cdot \frac{kTC_L}{I_B^2} \cdot f_{\text{out}} \tag{4.21}$$

where I_B is the bias current and all other parameters are transistor parameters, or common physical constants. R_L and C_L obtained from the equivalent model in Fig. 4.15 and f_{out} is the output frequency of the divider.

The flicker noise expression is given by (SL is the slope of the voltage with time):

$$\mathcal{L}(f_m) = \frac{2\pi^2 f_{\text{out}}^2}{(SL)^2} \cdot S_V\left(f_m\right) \tag{4.22}$$

These two when added will give the total phase noise. For the case of the cascaded D-type Flip Flop (FF) and an AND gate, the circuit looks as shown in Fig. 4.16. [15]. Note that the AND gate does not add any noise as such as one of the inputs perfectly switched off when the DFF works, and the other input forms a part of the DFF itself. Therefore, we assumes the (4.21) and (4.22) hold even in the case of the cascaded AND and DFF case.

Assuming this, we find that from Fig. 3.12, three of the FFs are used when p is active and 2 are used when p is not active. Therefore, the phase noise of the final stage is given by taking the output of each stage (whether it divides by 2 or 3) and multiplying with the corresponding ratio. With this taken into account we get the total phase noise for a cascade of 2/3 cells as:

$$\mathcal{L}_{W,\text{total}} = \sum_{k=0}^{n-1} 8\pi^2 \left(1 + \frac{\gamma}{\alpha} + \frac{\gamma_T g_{m_T} R_L}{2\alpha_T}\right) \cdot \frac{kTC_L}{I_B^2} \cdot \frac{f_{\text{o}}}{\left(p_k + 2\right)^2} \tag{4.23}$$

$$\mathcal{L}(f_m) = \sum_{k=0}^{n-1} \frac{2\pi^2 f_{\text{o}}^2}{(SL)^2 \left(p_k + 2\right)^{(k-1)}} \cdot S_V\left(f_m\right) \tag{4.24}$$

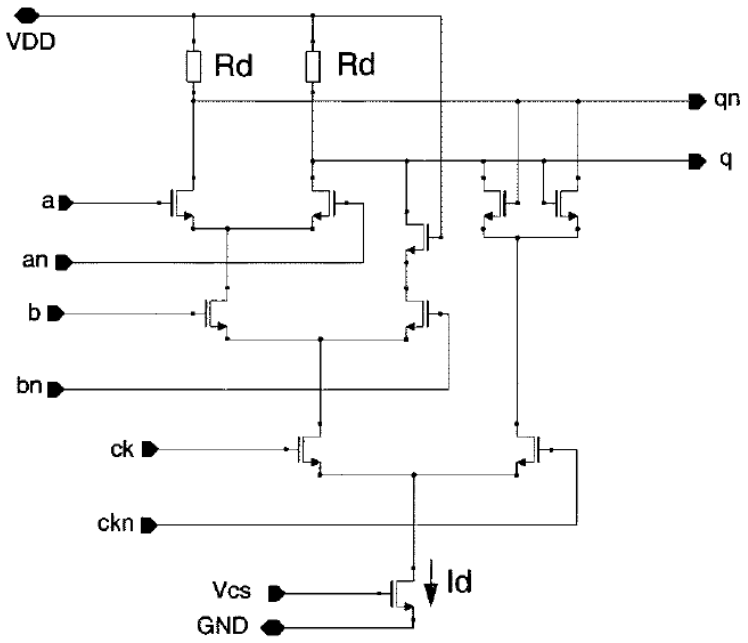


Fig. 4.17. SCL DFF with AND Gate cascaded

For a detailed understanding of the above two formulae please refer to Fig. 3.12 and Fig. 3.13 and refer the corresponding sections. The above derivation is complete if the word is available, however certain other effects such as loading of one divider has to be taken into account while modeling. These will be considered during the modeling phase.

Therefore, we see that the SCL DFF can be used for the case of extracting the phase noise expression for the 2/3 cell architecture divider for any length and word with the above formulation. The actual model will be made taking into account the $S_V(f_m)$ which is equivalent to the actual flicker noise of the dividers. Therefore, the phase noise of the dividers has an effective $1/f^2$ kind of characteristic with a noise floor.

This completes our analysis of the phase noise of the divider. Note that we have analyzed the phase noise of the prescaler based divider architecture, which is a more challenging task as DFFs lie in lengthy feedback and phase gets quickly accumulated. This is left as a future direction, although this divider architecture is usually not preferred.

4.6 Conclusions for Chapter 4

This chapter has comprehensively explored the modeling of noise in PLLs. We have taken into consideration the non-idealities of every sub-block of the PLL and have also accounted for the simulation taking place in the continuous event-driven mode.

Using the concepts derived in Chapter 2 and the nature of the FS and PLL discussed in Chapter 3 we were essentially able to eliminate the modeling of some blocks of the PLL while concentrating on the others. Further, we were able to derive equations for the phase noise of the VCO and divider

after taking help of the fact that models exist in the discrete domain for the phase noise of the VCO and the other blocks. The derived phase noise equations were used to compare the noise in the $1/f^2$ region and the thermal noise floor using the observations in [14].

This chapter in conclusion is the final important chapter of this thesis and concludes the behavioral modeling and understanding of a PLL simulated in Simulink. We have accounted for three important features: the *simulation environment*, the *PLL and FS* and the *non-idealities in the PLL and FS*. We shall now move onto a chapter concluding this thesis and visioning the future possibilities of time domain modeling and its advantages in understanding RF circuits better.

Chapter 5

Conclusive Remarks and Future Work

5.1 Some Comments on the Work

There are many areas or certain small points in the work carried out thus far which the author feels requires further analysis and will help improve the work. These are not for future scope, but are essential for an even more complete understanding of the problem.

Firstly, it is important to remove ourselves from the comfortable shell that is Simulink MATLAB. Therefore, it is important for us to completely at least once code directly the simulation environment shown in Fig. 2.3. This will give us a good understanding of the Range-Kutta ode45 solver method and further also gives us the actual algorithm followed to determine the time steps and the timestamp index.

Also, it is important to note the theory behind the phase noise derivations in the MATLAB simulation environment. Most of the theoretical phase noise derivations are highly approximate as shown in Chapter 4 and require further analysis to push the match with theory and observed values. It will benefit if the readings of phase noise are matched with those of an actual chip.

The offset derivations for feedback systems are of prime importance. We found that since the $f_{\max}$ for the PFD is different from that of the divider and VCO, the feedback effects of the PFD do not affect the output of the VCO. However, the mega feedback in the GSM divider clearly affected the output in the case of the GSM FS. We could start out with relating the various stages of the PLL, the low pass stage and the band pass stage as directly related to the value of $f_{\max}$. This derivation may be used for the future analysis of the PLL to determine more quantitatively the effect of feedback systems.

The incomplete portions of the project are as follows:

1. The derivation and modeling of the $1/f^3$ noise in the continuous event-driven time domain simulation environment.
2. The derivation and modeling of the divider phase noise

These will be performed soon as the theory has been mostly already developed in Chapter 4.

5.2 Future Directions

Some future directions to the work are discussed in this section. The PLL is a very intriguing circuit and there have been attempts to model it comprehensively in the circuit level, in terms of phase noise [22]. However, still, a comprehensive model for the PLL and its noise performance is difficult to derive, as it works in both the frequency and the time domain.

Further, the effect of one block on the other is very important to note in PLLs and the noise performance may affect each other. For example, [23] attempts to find the noise performance effects of one block in the receiver on the other. This concept may be extended to the sub-blocks of the PLL to determine the overall noise performance.

Lastly, the noise models from the physical point of view are highly dependent on the kind of technology used and therefore are not easily extracted to be modeled in the behavioral domain. Further, in the behavioral domain, the models are highly affected by the sampling frequency and the event-driven simulation environment. A more comprehensive understanding and design of the physical and behavioral models will certainly benefit the approached envisioned in the above work.

A visionary future direction is to finally develop the ideas above into software that can be distributed for use in academia and industry.

5.3 The Conclusion

This thesis is a take on the possibility of reducing the simulation time and design cycle of the design of PLLs and Frequency Synthesizers. The key problems faced are the fact that the simulation environment sampling at non-uniform intervals will result in much faster simulation however, some tradeoff in accuracy are obtained. The conditions for these were discussed in Chapter 2.

Using the various concepts derived in Chapter 2 we completely model the PLL and the FS in Chapter 3 and also consider the measurement and modeling of noise in Chapter 4 with some comparisons with actually observed data. This thus completes the modeling of the frequency synthesizer using an event-driven simulation environment. Some portions that are possibly incomplete and require more time to work out are presented in Sec. 5.1 and 5.2.

We sign off from this project after understanding that indeed PLLs can be simulated much faster using event-driven simulation environments. If all the properties of the devices, the process, the voltages and the non-linear dynamics are taken into account in this kind of a simulation environment, we can finally say we have a comprehensive model. Further, non-idealities need extra attention while modeling the same and have been taken into account.

The above project has been an extensive journey into every nook and corner of this very interesting little circuit! The worldwide research in PLL still continues to charm and fascinate many and will certainly do so for many years to come considering the wide range of applications and deeper and deeper probe into higher frequencies of operation. Behavioral modeling is also a key research area of many.

The author only wishes that this document will further such a research philosophy and the work done here will improve one's understanding of the PLL and its behavioral modeling.

Bibliography

[1] Bizjak, L., Da Dalt, N., Thurner, P., Nonis, R., Palestri, P. and Selmi, L. (2008). "Comprehensive Behavioral Modeling of Conventional and Dual-tuning PLLs". *IEEE Transactions on Circuits and Systems I: Regular Papers*, July 2008, Vol. 55, No. 6, Page No. 1628-1638.

[2] Zhuang, J., Du, Q. and Kwasniewski, T. (2006). "Event Driven Modeling and Simulation of a Digital PLL". *2006 IEEE International Behavioral Modeling and Simulation Workshop*, September 2006, Page No. 67-72.

[3] Egan, W.F. (1990). "Modeling Phase Noise in Frequency Dividers". *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, July 1990, Vol. 37, No. 4, Page No. 307-315.

[4] Leeson, D.B. (1966). "A Simple of Feedback Oscillator Noise Spectrum". *Proceedings of the IEEE*, February 1966, Vol. 54, No. 2.

[5] Hajimiri, A. and Lee, T.H. (1998). "A General Theory of Phase Noise in Electrical Oscillators". *IEEE Journal of Solid-State Circuits*, February 1998, Volume 33, No. 2, Page No. 179-194.

[6] Zeigler, B.P., Praehofer, H. and Kim, T.G. (2000). "Theory of Modeling and Simulation". *Academic Press*, 2000, Edition 2.

[7] The Mathworks, (2008). "MATLAB Documentation", http://www.mathworks.com/, October 2008, for MATLAB R2008b edition.

[8] Perrott M.H. (2002). "Fast and accurate behavioral simulation of fractional-N frequency synthesizers and other PLL/DLL circuits". *39th Annual ACM Design Automation Conference*, 2002, Page No. 498-503.

[9] Schubert, M. (1999). "Mixed-Signal Event-Driven Simulation of a Phase Locked Loop". *1999 IEEE International Behavioral Modeling and Simulation Conference*, October 1999.

[10] Oppenheim, A.V. and Schafer, R.W. (1989). "Discrete-Time Signal Processing". *Prentice-Hall International*, 1989.

[11] Lee, T.H. (2004). "The Design of CMOS Radio-Frequency Integrated Circuits". *Cambridge University Press*, 2004, Edition 2.

[12] Banchi, G. (2005). "Phase-Locked Loop Synthesizer Simulation". *McGraw-Hill Professional*, March 9th 2005, Edition 1.

[13] Reehal, G. (1998). "A Digital Frequency Synthesizer Using PLL Technique". *Thesis – Information Electronics Research Group*, *Ohio State University,* 1998.

[14] Mandal, D. and Bhattacharyya, T.K. (2007). "7.95mW 2.4GHz Fully-Integrated CMOS Integer N Frequency Synthesizer". *20th International Conference on VLSI Design*, January 2007, Page No. 156-164.

[15] Vaucher, C.S., Ferencic, I., Locher, M., Sedvallson, S. Voegeli, U. and Wang, Z (2000). "A Family of Low-Power Truly Modular programmable Dividers in Standard 0.35um CMOS Technology". *IEEE Journal of Solid-State Circuits*, July 2000, Vol. 35, No. 7, Page No. 1039-1046.

[16] Brigati, S. Francesconi, F. Malvasi, A. Pesucci, A. and Poletti, M. (2001). "Modeling of Fractional-N Division Frequency Synthesizers with Simulink and MATLAB". *8th IEEE Conference on Electronics, Circuits and Systems*, September 2001, Vol. 2, Page No. 1081-1084.

[17] Yuen, R. (2003). "Effect of VDD Noise on Phase Jitter". *Technical Report –University of Waterloo,* 2003.

[18] Moon, U., Mayaram, K. and Stonick, J.T. (2002), "Spectral Analysis of Time Domain Jitter Measurements". *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, May 2002, Vol. 49, No. 5, Page No. 321-327.

[19] Staszewski, R.B., Fernando, C. and Balsara, P.T. (2005). "Event Driven Simulation and Modeling of Phase Noise of an RF Oscillator". *IEEE Transactions on Circuits and Systems*, April 2005, Vol. 52, No. 4, Page No. 723-733.

[20] Demir, A. (2002). "Phase Noise and Timing Jitter in Oscillators with Colored Noise Sources". *IEEE Transaction on Circuits and Systems*, December 2002, Vol. 49, No. 12, Page No. 1782-1791.

[21] Levantino, S., Romano, L., Pellerano, S., Samori, C. and Lacaita, A.L. (2004). "Phase Noise in Digital Frequency Dividers". *IEEE Journal of Solid-State Circuits*, May 2004, Vol. 39, No. 5, Page No. 775-784.

[22] Hajimiri, A. (2001). "Noise in Phase-Locked Loops". *2001 IEEE Mid-West Symposium on Mixed-Signal Design*, Invited Paper, 2001.

[23] Sheng, W., Ahmed E. and S-Sinecio, E (2006). "CMOS RF Receiver System Design: A Systematic Approach". *IEEE Transactions on Circuits and Systems – Part 1; Regular Papers*, May 2006, Vol. 53, No. 5, Page No. 1023 – 1034.

Project Presentations and Publications

[1] "Behavioral Event-Driven Time Domain Simulation of Frequency Synthesizers", *to be submitted soon to IEEE Transactions on Circuits and Systems - I.*

[2] The project was presented at various symposia and was also a part of the official RFIC group, Advanced VLSI Design Laboratory report for 2008-2009 on the design and analysis of frequency synthesizers.