AN ABSTRACT OF THE THESIS OF

Shanthi S. Bhagavatheeswaran for the degree of Master of Science in

Title: Design Methodology for Low-Jitter Phase-Locked Loops

Abstract approved: Redacted for Privacy

Terri S. Fiez

This thesis presents a systematic top-down methodology for simulating a
phase-locked loop using a macro model in Verilog-A. The macromodel has been
used to evaluate the jitter due to supply noise, thermal noise, and ground bounce.
The noise simulation with the behavioral model is roughly 310 times faster (best
case) and 125 times faster (worst case). The accuracy of the model depends on
the accurate evaluation of the non-linear transfer function from the various noisy
nodes to the output. By modeling the noise transfer function of the circuit as closely
as possible, 100% accuracy for the behavioral noise simulations compared with the
HSPICE noise simulations is obtained.

The macro model is written for a charge-pump phase-locked loop, but can
be easily extended to other architectures. The simulations are completed using
SpectreS in Cadence. The designer can use the model to estimate the jitter at the
output of the PLL in a top-down design methodology or cross verify the performance
of an existing chip in a bottom-up approach.
©Copyright by Shanthi S. Bhagavatheeswaran

February 23, 2001

All rights reserved
Design Methodology for Low-Jitter
Phase-Locked Loops

by

Shanthi S. Bhagavatheevaran

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed February 23, 2001
Commencement June 2001
Master of Science thesis of Shanthi S. Bhagavatheeswaran presented on February 23, 2001

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Shanthi S. Bhagavatheeswaran, Author
ACKNOWLEDGMENT

This work is an encomium for all the teachers and professors who have contributed to my academic growth. I would like to express my sincere gratitude and appreciation to my advisor, Prof. Terri Fiez, whom I regard as my "Anne Sullivan", guiding me into the world of Analog Design. She was with me throughout the course of my Master's program, extending tremendous support at every crucial step, providing bountiful opportunities, and helping me complete my studies. I am most grateful to Prof. Kartikeya Mayaram for his timely helps and constructive feedbacks. My sincere thanks and appreciation are due to Dr. Monte F. Mar of Cypress Microsystems whose design expertise and suggestions came in time for my research work as also for agreeing to serve in my committee. Special thanks goes to Prof. Dean Jensen for his time to serve in my defense committee.

Thanks goes to the engineers of Cypress Semiconductors for their extra time, especially, James Wilson and Grant Erwin, who helped us with setting up simulations. My thanks are due to Robert Batten and Uma Chilakapati for their helping me with almost everything throughout this course, especially during my initial stay in Pullman. The alacrity shown by Jose’ Silva in answering my CADENCE questions deserves mention. Thanks also to Ruoxin Jiang for helpful discussions.

I had the honor of working with a group of talented engineers while being an intern at Motorola Inc. I am most thankful to my mentor and supervisor Jeff Ganger, for his precious time to teach me engineering skills. My sincere thanks goes to Dr. Rajesh Zele for his creating a conducive atmosphere for my learning. Thanks also goes to my colleague engineers Kiyong, Srinath, Praveen, Ramana Aisola, Gary, Jim, Wade, Dave, Brian, Ling, Bob Garbs, Bob Rice, Steve Hamm, Kiran Gullappalli, and David Fitts.
I would like to thank Nathen Barton and Aline Sadate for all their help and friendship. Thanks to Tina Batten, Vinay Chandrasekhar, Taras Dudar, Christina Fetc, Radu Fetc, Kalyan Ghatak, Yutao Hu, Zhiming Li, Ravi Naiknaware, Yuxian Ou, Dicle Ozis, Chennam Madhusudhan Rao, Anil Samavedam, Nilakantan Seshan and Haiming Tang for their friendly association. Thanks goes to my friends Indu Mahadevan, Nilesh Bhide, Satish Damodaran, and Jian Wang for making my life in Pullman a pleasant one and to Nagesh, Srinivas, and Srikar of Austin.

Special thanks to our E.C.E graduate secretary, Ferne Simendinger for all the help during my admission to Oregon State University and subsequent schooling there. Thanks goes to Ruby Young - the graduate secretary at Washington State University (W.S.U) - who helped me simplify my admission to that school, and also to Cliff Fallon, my T.A. supervisor at W.S.U. for making my teaching a good learning experience.

Thanks goes to all my family members who supported me through this program. Last but not the least, it is the support showered on me by my parents and my elder sister Gayathri Bhagavatheeswaran, that helped me reach this level in my life. Truely, I appreciate their patient endurance.

I am thankful to God for blessing me with a wonderful circle of teachers, family and friends.
TABLE OF CONTENTS

1 INTRODUCTION ........................................................................ 1
  1.1 Background ........................................................................ 1
  1.2 Literature Review ................................................................ 7
  1.3 Contribution of this work .................................................. 10

2 PROPOSED METHODOLOGY .................................................. 12

3 MACRO MODEL ...................................................................... 15
  3.1 Functionality ...................................................................... 15
  3.2 Simulating a Tracking Circuit ............................................ 23

4 APPLICATIONS ...................................................................... 27
  4.1 Model Used to Evaluate Supply Noise and Ground Bounce .... 27
  4.2 Setup .................................................................................. 28
  4.3 Simulation .......................................................................... 32
  4.4 Measurement of period jitter .............................................. 34
  4.5 Results ............................................................................... 35
  4.6 Inferences .......................................................................... 36

5 CONCLUSION .......................................................................... 43

6 SCOPE FOR FURTHER RESEARCH ........................................ 44

BIBLIOGRAPHY ......................................................................... 45

APPENDICES .............................................................................. 47
## TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>APPENDIX</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Verilog-A code for the phase frequency detector shown in Fig. 3.2</td>
<td>48</td>
</tr>
<tr>
<td>B</td>
<td>Verilog-A code for the charge pump shown in Fig. 3.2</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>Verilog-A code for the Input VCO and VCO shown in Fig. 3.2</td>
<td>51</td>
</tr>
<tr>
<td>D</td>
<td>Verilog-A code for the phase frequency detector/charge pump shown in Fig. 3.8</td>
<td>52</td>
</tr>
<tr>
<td>E</td>
<td>Verilog-A code for the VCO in set up in Fig. 3.8</td>
<td>54</td>
</tr>
<tr>
<td>F</td>
<td>HSPICE netlist for the Inverter1 shown in Fig. 4.2 used to generate Noisy ground</td>
<td>55</td>
</tr>
<tr>
<td>G</td>
<td>HSPICE netlist for the Inverter1 shown in Fig. 4.2 used to generate Noisy supply</td>
<td>56</td>
</tr>
<tr>
<td>H</td>
<td>Verilog-A code for the phase frequency detector shown in Fig. 4.3</td>
<td>57</td>
</tr>
<tr>
<td>I</td>
<td>Verilog-A code for the VCO shown in Fig. 4.2, without the factor</td>
<td>59</td>
</tr>
<tr>
<td>J</td>
<td>Verilog-A code for the VCO shown in Fig. 4.2, with the factor</td>
<td>60</td>
</tr>
<tr>
<td>K</td>
<td>Verilog-A code for the phase frequency detector shown in Fig. 4.4</td>
<td>61</td>
</tr>
<tr>
<td>L</td>
<td>Verilog-A code for the VCO shown in Fig. 4.4, without the factor</td>
<td>63</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Basic block diagram of a phase-locked loop.</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Typical transfer characteristic of a voltage controlled oscillator.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Various architectures for the blocks.</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Transfer characteristics of different kinds of phase detectors.</td>
<td>6</td>
</tr>
<tr>
<td>1.5</td>
<td>Table of comparison showing the emphasis of each approach.</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Top-down simulation methodology for phase-locked loop.</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Bottom-up verification of the phase-locked loop design.</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Block diagram of a charge pump phase-locked loop.</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Setup for simulating a charge pump phase-locked loop.</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Pseudo code for the phase frequency detector.</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Pseudo code for the charge pump.</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Pseudo code for the voltage controlled oscillator.</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>(a) Reference voltage input and (b) VCO voltage locked to the reference.</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>(a) Control voltage to the VCO and (b) Zoomed in Control voltage.</td>
<td>22</td>
</tr>
<tr>
<td>3.8</td>
<td>Block diagram of the phase-locked loop tracking circuit.</td>
<td>23</td>
</tr>
<tr>
<td>3.9</td>
<td>Input signal for the tracking circuit.</td>
<td>24</td>
</tr>
<tr>
<td>3.10</td>
<td>Output with C1=100fF and with C1=1pF.</td>
<td>24</td>
</tr>
<tr>
<td>3.11</td>
<td>Phase error due to an input frequency step, for varying phase detector gains.</td>
<td>25</td>
</tr>
<tr>
<td>3.12</td>
<td>Phase error due to an input phase step, for varying phase detector gains.</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>Illustration of different kinds of jitter.</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Set up for simulating the power supply noise and ground bounce.</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Set up of the Test PLL for simulating the supply noise.</td>
<td>30</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Set up of the Test PLL for simulating the ground bounce.</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>Noisy supply waveform.</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparison of speed between behavioral simulations.</td>
<td>36</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of results for model and HSPICE for $L_{di/dt}$ supply noise.</td>
<td>37</td>
</tr>
<tr>
<td>4.8</td>
<td>Comparison of results for model for $L_{di/dt}$ ground noise.</td>
<td>38</td>
</tr>
<tr>
<td>4.9</td>
<td>Output jitter for sinusoidal noise injection with reference at 50MHz.</td>
<td>39</td>
</tr>
<tr>
<td>4.10</td>
<td>Output jitter for sinusoidal noise injection with reference at 169MHz.</td>
<td>40</td>
</tr>
<tr>
<td>4.11</td>
<td>Output jitter for square noise injection with reference at 50MHz.</td>
<td>40</td>
</tr>
<tr>
<td>4.12</td>
<td>Output jitter for square noise injection with reference at 169MHz.</td>
<td>41</td>
</tr>
<tr>
<td>4.13</td>
<td>Variation of output period with respect to time from behavioral simulation.</td>
<td>41</td>
</tr>
<tr>
<td>4.14</td>
<td>Variation of output period with respect to time from HSPICE simulation.</td>
<td>42</td>
</tr>
</tbody>
</table>
I don't know what I may seem to the world, but, as to myself, I seem to have been only like a boy playing on the sea shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.

SIR ISAAC NEWTON
1. INTRODUCTION

1.1. Background

A phase-locked loop synchronizes an output signal generated by a source (like a crystal oscillator) to an input signal in frequency as well as in phase [1]. A basic block diagram of a phase-locked loop is as shown in Fig. 1.1.

FIGURE 1.1. Basic block diagram of a phase-locked loop.
It offers two types of corrections - frequency correction for large differences between the reference and feedback inputs and phase corrections for small differences. The process by which a phase-locked loop locks onto a reference signal is highly nonlinear and is known as acquisition or capture of the signal. The phase-locked loop first tries to acquire the frequency, the process being called frequency acquisition, and then tries to lock onto the phase, called phase acquisition. The range of frequencies for which a phase-locked loop (PLL) acquires the frequency is called the pull in range and the range of frequencies for which the PLL does phase acquisition is called the lock in range of the PLL. The PLL finds application in a variety of wireless and digital systems for jitter reduction, clock data recovery, and frequency synthesis [1]-[5]. One of the key performance parameters for a phase-locked loop is the timing jitter [6], or the phase-noise in frequency domain [7]. For example, the clock jitter can degrade the performance of a microprocessor especially at high frequencies [8] or it can worsen the sensitivity of the receiver in a communication system.

The input signal and the feedback signal are compared by the phase detector. The phase detector produces an output that is proportional to the phase difference between the two signals. The loop filter low-pass filters the output of the phase detector and produces a voltage whose polarity is such as to change the frequency of oscillation of the voltage controlled oscillator. In this way, the voltage controlled oscillator (VCO) tracks the input in frequency and phase. A typical VCO transfer characteristic is shown in Fig. 1.2. The DC voltage at the input of the VCO generates a output whose frequency is determined by the transfer characteristic shown in Fig. 1.2. The output signal is fed back to the input of the phase detector where it is compared with the input signal frequency. If the input signal frequency is less than the VCO frequency, the phase error output of the phase detector is such that the
low-pass filter produces a negative voltage. This error control voltage when applied to the VCO reduces the frequency of oscillation and brings the VCO frequency close to the input signal frequency. If the input signal frequency is more than the VCO frequency, the phase error output of the phase detector is such that the low-pass filter produces a positive voltage. This error voltage increases the oscillation frequency of the VCO until it becomes equal to the input signal frequency. This is how the PLL offers frequency correction.

\[ f_o = f_c + VCOGAIN \cdot Vc \]

FIGURE 1.2. Typical transfer characteristic of a voltage controlled oscillator.

There are various architectures for each block of a phase-locked loop. They are summarized in Fig. 1.3. The parameters of the phase-locked systems, including pull in range, transient response depend on the properties of the phase and frequency detectors employed in such systems [5]. The basic features of different kinds of phase detectors is summarized below.

Analog multipliers and sequential phase detectors are the two broad classes of devices used as phase detectors in a PLL. A multiplier acts as a phase detector [3] according to the identity

\[ \sin(A)\cos(B) = 0.5\sin(A - B) + 0.5\sin(A + B) \]  

(1.1)
If the inputs to the multiplier are two sinusoids given by

$$v_i = V_i \sin(\omega_i t)$$
$$v_o = V_o \cos(\omega_i t)$$

(1.2)

then the output of the multiplier is

$$v_d = K_m v_i v_o$$

(1.3)

Topologies such as Gilbert cell multipliers can be used to implement an analog multiplier. Analog multipliers have a limited linear range for phase tracking (only $$-\pi/2 < \theta_e < \pi/2$$) and its gain varies as the sine of the phase error.

Sequential phase detectors do not have these limitations. Among the sequential phase detectors are the exclusive-OR phase detector, two-state phase detectors (like the RS flip-flops, JK flip-flops), and the most widely used three-state phase frequency detector. An EX-OR [3] circuit is similar to an over-driven multiplier circuit. The disadvantages of the circuit are that it malfunctions if the duty cycle of the waveforms is not 50%. This results in reduction in the gain. The output signal is a triangular function of the phase error [1]. Though the linearity of phase
tracking is more than that of an analog multiplier, it is still confined to the range of \(-\pi/2 < \theta_e < \pi/2\).

In a two-state phase detector, the waveform symmetry is not important. The linearity of tracking is maintained over \(-\pi < \theta_e < \pi\). Let the two signals be referred to as IN1 and IN2. The phase detector is sensitive only to either edge of IN1 and IN2. When IN1 goes high, say the detector is in state1; then when IN2 goes high after this, the state changes to state2. Thus it keeps toggling between the two states. Though this phase detector can detect both phase and frequency for very small differences in frequency, the pull-in time can be very large. This is explained in conjunction with the phase/frequency detector. The transfer characteristic of different kinds of phase detectors is shown in Fig. 1.4.

The advantage of the phase frequency detector can be described by referring to Fig. 1.4. In the unlocked state, the phase error of the phase frequency detector remains either positive if the reference input signal leads the VCO, or negative if the reference input signal lags behind the VCO. This provides an indication of the direction of the frequency error and is in acquiring lock when the two frequencies are initially different. Thus in the unlocked condition, one terminal (UP or DOWN, depending on the sign of frequency difference) will remain active as long as the loop is out of lock and the other terminal switches back and forth between levels in a manner determined by the frequency and phase differences [2]. But in the case of an ex-OR or a two-state phase detector, the average output does not remain monotonic since it is a periodic function of the phase error, Fig. 1.4. Since the three-state phase detector can provide both frequency and phase acquisition, it is called a phase frequency detector whereas the ex-OR and the two state are just phase detectors.
FIGURE 1.4. Transfer characteristics of different kinds of phase detectors.
The detection of frequency by a phase frequency detector is made possible by the addition of a third state in the state machine. Instead of toggling to state2 for the sequence IN1, IN2, the phase frequency detector toggles to an intermediate state, state0. Hence the error voltage is monotonically positive or negative for the unlocked state unlike the two-state case, and this offers for frequency detection.

The pull-in range for a phase frequency detector phase-locked loop is only limited by the overload limit of the voltage controlled oscillator. This helps acquisition of signals even under the worst operating conditions. The linearity of tracking range is \(-2\pi < \theta_e < 2\pi\). According to [3], three-state phase frequency detectors function well only if the input signal is periodic. The concept can be extended to an n-state phase frequency detector.

One of the difficulties in the design of the a phase-locked loop is the long simulation time required to drive the loop to steady state. This is done in order to see if the loop is acquiring lock and henceforth to extract some critical performance parameters like timing jitter.

This thesis discusses the top-down design methodology which can be used for designing a phase-locked loop with increased speed and accuracy. This reduces the design cycle time as well. Before describing the approach taken in this research, an overview of the existing simulation technique is provided.

1.2. Literature Review

The circuit level simulation of the phase-locked loop and hence the evaluation of output jitter is difficult for the following reasons [9]. Large spread of time constants of the order of milliseconds for the loop filter and nanoseconds for the oscillator require long run times.
Jitter specifications of a few hundred pico seconds necessitate small time steps. Typically, it may need several thousand million points for simulation before the loop acquires lock. This has led to development of behavioral models for fast and accurate simulation of the loop.

Several mixed-mode behavioral models for simulation of the phase-locked loop have been presented before in [9]-[17]. In [9], a generic and top level behavioral model (which does not include jitter) has been implemented in ABCDL Analog Behavior Circuit Description Language (ABCDL) and simulated in the AT&T Bell Laboratories ADAMS simulator. An example of how the model can be used for simulating the step response and to track an input signal has been given. A high-speed digital phase-locked loop is simulated at the behavioral level and at a mixed-level using device level models for the phase-detector to measure the lock-in time and detect false locking. The CPU time for the multi-level models versus an all behavioral model is reported to be 5min 17.00sec versus 3min 15.21sec for simulating 10 microseconds on a Sun SparcStation1+

In [10], Demir has an approach of simulating a phase-locked loop at a high level. Behavioral simulation of phase/delay locked systems is explained. A powerful new simulation algorithm, based on nonlinear stochastic differential equation is given. The approach provides accurate and efficient prediction of phase-locked loop jitter behavior once the noise behavior of the blocks have been characterized. The behavioral simulation is reported to be around 103 times faster than the SPICE simulation, both for the simulation of acquisition in a DEC 5000/260.

In [11], a phase-locked loop model using an event driven algorithm to speed up the simulation is given. It incorporates jitter and noise analysis while still maintaining a tight connection to the physical circuit parameters. Examples of how the design can be related to direct design decisions is also given. But the algorithm
is embedded in a simulator that is not readily available. It is reported to take 40 seconds of actual time on a Sun Ultra 60 station to generate 1ms of data (400,000 samples) for a PLL running at 400MHz.

In [12], three different models are presented. A discrete linear model, a discrete non-linear model and an event driven model have been compared with a transistor level simulations. It has been shown that the event-driven model is the most accurate. A comparison of the lock transient between various models has been done and it has been concluded that the computational effort of the behavioral models for a given accuracy is significantly less than a full SPICE simulation.

In [13], phase-locked loop simulation technique using VERIMIX, Simulink and Simulink/C-MEX has been studied. Using Simulink/C-MEX allows part of a model written in C language and then integrate it with rest of the simulink model. The behavioral level analog with VERIMIX is experimented first, where Verilog is used to simulate the logic part of the design and MCSPICE for the analog part. It has given accurate results, however has taken more time than Simulink simulations. In addition to this, in order to tighten the tolerance for better accuracy, speed had to be traded off. In the Simulink simulations, a model for the VCO has been developed; however, it has been cited that it is hard to do a mixed-level simulation in Simulink. It has been reported in this that Simulink/C-MEX, which is the fastest (among the three approaches identified), is 75 times faster than VERIMIX with behavioral level model.

In [14], a nonlinear modeling of phase-locked and delay-locked systems is given. Behavioral simulators for these systems are written in C language with non-linear behavioral models for VCO and VCDL (used in delay-locked systems). The performance of the simulators is evaluated for noise performance by adding a delay into the delay cell.
In [15], nonlinear phase-locked loop modeling of is presented in SpectreHDL with emphasis on phase noise evaluation.

In [16], after giving a brief introduction to the top-down design and bottom-up verification methodology, necessary models for the systematic design of a frequency synthesizer used in telecommunication applications is given.

In [17], phase-locked loop modeling has been implemented in ELDO simulator which takes into account the nonlinearities of the blocks, the parasitic and thermal noises from various sources, but without actual verification from circuit level simulations.

In [18], is given the modeling of jitter in phase-locked loop frequency synthesizers using Verilog-A, considering the random noise sources. The phase-locked loop has been modeled taking the random noise sources into account for each of the blocks. The ideas were exemplified with loop parameters taken from a paper-based design. For a reference frequency of 25MHz and a divide ratio (in the feedback path) of 10,000, the simulation of period jitter (due to a given amount of phase noise) is reported to have taken 7.5 minutes for 450k time-points on a HP 9000/735.

A table which summarizes the unique feature of each one of the above with the present work is given Fig. 1.5.

1.3. Contribution of this work

In this thesis, a macro model using Verilog-A [19] for modeling phase-locked loop jitter due to noises from supply and ground is presented. Simulations are completed in SpectreS. The model can be easily integrated into a commonly used design tool like Cadence. The noise simulation with behavioral model is roughly 310 times faster (best case) and 125 times faster (worst case) than the HSPICE noise
simulations. The accuracy of the model depends on the accurate evaluation of the nonlinear transfer function from the various noisy nodes to the output. By modeling the noise transfer function of the circuit as closely as possible, 100% accuracy for the behavioral noise simulations compared with the HSPICE noise simulations is obtained.

The methodology is given in Chapter 2, the macro model in Chapter 3, applications are described in Chapter 4, the conclusion is provided in Chapter 5 and the scope for further research in Chapter 6.

FIGURE 1.5. Table of comparison showing the emphasis of each approach.
2. PROPOSED METHODOLOGY

As explained in Section 1.2, phase-locked loops are a difficult class of systems to be simulated because of the extensive run times required. To speed up the simulation time, high level behavioral modeling is used. The level of abstraction sets the trade off between the simulation speed obtained and the accuracy of simulation. The approach can be a top-down or a bottom-up one [9]. The top-down methodology can be used for examining the design trade-offs between the various loop parameters for a given application and a bottom-up approach to verify an existing designs.

The various steps in a top-down approach can be enumerated as follows.

1. Develop an ideal macro model with only the functionalities of the blocks for a given set of specifications. This allows the designer to make a first-order estimate of the system parameters required.

2. Design the circuit level component blocks. Extract the non-idealities and plug it back into the model to get a non-ideal model with first order non-idealities. Also find out the component sensitivities to the various sources of noise like device parasitic noise, thermal noise, supply noise and ground bounce noise. This helps in identifying the design trade-offs to be made.

3. Implement the phase-locked loop and validate the simulated results from the model with the actual measured results from the implementation.

The above steps are summarized with the flowchart in Fig. 2.1.
FIGURE 2.1. Top-down simulation methodology for phase-locked loop.
Once there is a macro model to start with, we can use a bottom-up approach to verify an existing design.

1. Measure the performance parameters of the design from silicon.

2. Validate the simulated results from the model with the measured results from the implementation.

The corresponding flowchart is shown in Fig. 2.2.
3. MACRO MODEL

3.1. Functionality

A description of the macro model is given in this section. The macro model is developed using Verilog-A, choosing a charge pump phase-locked loops as a typical architecture. It can be extended to any other architecture and application. A charge pump phase-locked loop with a second-order loop filter has the following advantages. The charge pump switches current off for zero phase error between the reference and feedback signals. This eliminates the need for a large DC gain or the use of active filters. In a second-order loop filter (refer Fig. 3.1), the small capacitor C2 prevents VCO overloading and sets the stability limit for the choice of loop parameters. But the ratio, \( C1/C2 \) has to be large enough to avoid instability at moderate loop gains (of the order of 2 to 10) [20].

The block diagram of a charge pump phase-locked loop is shown in Fig. 3.1.

![Block diagram of a charge pump phase-locked loop](image.png)

**FIGURE 3.1.** Block diagram of a charge pump phase-locked loop.
The following are the reasons for choosing Verilog-A to implement the macro model. A mixed-mode (along with Verilog-XL), mixed level simulation (with some blocks as behavioral models and some as circuits) is possible. It is integrated into a commonly used design environment. Bottom-up verification is easier. Fast simulation times for the loop compared with the SPICE level simulation, while doing a locking transient to evaluate the jitter.

Referring to Fig. 3.2, a brief description of the models is given below. The dc voltage source at the input is used instead of an pulse source in order to be able to get a continuous variation of frequency.

**FIGURE 3.2.** Setup for simulating a charge pump phase-locked loop.

1. Phase Frequency Detector

   The phase frequency detector is implemented as a state machine, sensitive only to the positive edges of the reference signal \( \text{ref} \) or the fed back signal \( vco_{out} \) from the voltage controlled oscillator. Its output goes into the charge pump block.
2. Charge Pump

The charge pump has two switched current sources, controlled by the output from the phase frequency detector. This pulsed current goes into the loop filter to generate the control voltage for the voltage controlled oscillator. The most important advantage of the charge pump phase-locked loops arises because the charge pump opens up when the input phase difference is zero. This eliminates the need for a large DC gain or the use of active filters.

3. Voltage Controlled Oscillator

The VCO receives input from the loop filter. In Fig. 3.2, ref is the reference input to the phase frequency detector and \( vco_{out} \) is the corrected output of the voltage controlled oscillator. If ref leads \( vco_{out} \), the charge pump sources current into the loop filter such that \( lpf_{out} \) is positive and increases the frequency of the voltage controlled oscillator. When the ref lags \( vco_{out} \), charge pump sinks current, and \( lpf_{out} \) is negative, thereby decreasing the VCO frequency. Inside the VCO block, a frequency is first generated according to the magnitude of the \( lpf_{out} \), and then it is integrated to get the corresponding phase (because the phase frequency detector compares the phases of two signals).

The models for the phase frequency detector, charge pump and the voltage controlled oscillator are written in Verilog-A. As shown in Fig. 3.2, the loop filter is implemented using electrical components.

The behavioral simulations are done using the Cadence interface with SpectreS. The functionality of the phase frequency detector, charge pump and voltage controlled oscillator are verified. The logic is given by the pseudo code in Fig. 3.3 - Fig. 3.5. For the actual code, refer to Appendix A.
'include HEADER FILES
module pfd(REF, VCO, UP, DOWN);
  port declarations;
  parameter threshold voltage crossing;
  parameter delay time, rise time, fall time;
  variable declarations;
  analog begin
    // rising edge sensitive
    // positive logic

      if (REF leads VCO)
        UP=HIGH;
      else if (VCO leads REF)
        DOWN=HIGH;
      else if (REF and VCO are ON at the same time)
        UP=LOW;
        DOWN=LOW;
    end
  endmodule

FIGURE 3.3. Pseudo code for the phase frequency detector.
`include HEADER FILES
module cp(UP, DOWN, OUT);
  port declarations;
  parameter threshold voltage crossing;
  parameter delay time, rise time, fall time;
  variable declarations;

  analog begin
    // rising edge sensitive
    // positive logic
    if (UP)
      OUT = source current into loop filter;
    else if (DOWN)
      OUT = sink current to ground;
    else if (UP && DOWN)
      OUT = 0;
  end
endmodule

FIGURE 3.4. Pseudo code for the charge pump.
FIGURE 3.5. Pseudo code for the voltage controlled oscillator.

In the set up, the reference input is initially 16MHz while the output of the voltage controlled oscillator is 15.625MHz. The following can be observed from the Fig. 3.7 During the acquisition process, the loop is nonlinear and the loop filter output shows a damped response. The high frequency component is due to the charging and discharging of the loop filter capacitor by the pulses from the phase frequency detector. The small signal settling time (envelope of the waveform) can be changed by changing the loop bandwidth. To decrease the settling time, we increase the loop bandwidth, for a given value of the ratio, $C1/C2$. The loop filter output provides a positive voltage for the voltage controlled oscillator to lock onto the reference, because the reference input is greater than the initial voltage controlled oscillator frequency. The steady state error will be driven to zero after some time for
a charge pump phase-locked loop. In lock, there is no cycle slip between the reference frequency and the VCO frequency. This means that the two cycles (frequencies) are equal in lock.

The SpectreS simulator took 152.95 seconds for 300k time points in a Sun Sparc Ultra10 machine.

All the simulations were done in Sun Sparc Ultra10.

![Graph of reference voltage input](attachment:image1)

![Graph of VCO voltage](attachment:image2)

FIGURE 3.6. (a) Reference voltage input and (b) VCO voltage locked to the reference.
FIGURE 3.7. (a) Control voltage to the VCO and (b) Zoomed in Control voltage.
3.2. Simulating a Tracking Circuit

The macro model can be used to simulate a tracking circuit. The set up is shown in Fig. 3.8. The code used for phase frequency detector/charge pump arrangement is given in Appendix D and the code for the voltage controlled oscillator is in Appendix E. The design is based on [20].

![Block Diagram of the Phase-Locked Loop Tracking Circuit](image)

FIGURE 3.8. Block diagram of the phase-locked loop tracking circuit.

The simulation of the above system took 2.94 seconds on a Sun Sparc Ultra10 workstation. From Fig. 3.10, we can see that the output of the loop filter of a phase-locked loop can be used to demodulate an input signal. This finds application especially in the field of communications. The details of how to carry out the simulation is also shown in Fig. 3.8. One application of the set up is to evaluate the output jitter. An example of how the model can be used to predict the effect of various loop parameters, in this example, the phase detector gain, on the output jitter is shown in Fig. 3.11. The step in frequency input is 10MHz and the step in phase input is 180 degrees.
FIGURE 3.9. Input signal for the tracking circuit.

FIGURE 3.10. Output with $C_1=100\text{fF}$ and with $C_1=1\text{pF}$. 
FIGURE 3.11. Phase error due to an input frequency step, for varying phase detector gains.

FIGURE 3.12. Phase error due to an input phase step, for varying phase detector gains.
As seen from Fig. 3.11, the loop filter is of second order, the additional pole from the VCO makes the loop a third order one. From Control Systems theory, the steady state error of a third order system to a step input and a ramp input is equal to zero. Here, the loop variable is phase error. So, the steady state phase error due to a frequency step and phase step should be zero. From Fig. 3.11 and Fig. 3.12, the static phase error is equal to zero, the loop corrects the error quickly for higher phase detector gains. The loop shows a damped response typical of a third order system, since the phase-locked loop basically operates on phase errors. The ringing is more for higher phase detector gains, but the ringing in phase error settles faster for the case of higher phase detector gains than the ones with lesser phase detector gains.
4. APPLICATIONS

4.1. Model Used to Evaluate Supply Noise and Ground Bounce

The uncertainty in the timing of an event is called *Timing jitter*. Jitter measurements are mainly of three types [21]: cycle-to-cycle, period, and long-term. The measurements are made at a specified voltage. An illustration of these three types of jitter is shown in Fig. 4.1. The change in a clock's output transition from its corresponding position in the previous cycle is called *cycle-to-cycle* jitter. Period jitter measures the maximum change in a clock's output transition from its ideal position. Long term jitter measures the maximum change in a clock's output transition from its ideal position over many cycles.

![Illustration of different kinds of jitter.](image)

The cycle-to-cycle jitter contains information about the short term dynamics of the system [6] and is meaningful for oscillators, whereas long term jitter and
period jitter are meaningful for phase-locked loops, because the loop dynamics tends to even out the short term variations in frequency and phase. Usually, the jitter measurements are made over multiple cycles and the maximum over these is given as the jitter.

Jitter is one of the key performance parameters of a phase-locked loop used in communication applications, analog-mixed signal blocks or for phase noise evaluation. In this section it is shown how the model can be used to investigate the effect of supply noise and ground bounce on the output jitter. The simulations from the model are cross verified with the simulations from a commercial design.

4.2. Setup

The setup used to simulate and measure the output jitter are shown in the figures Fig. 4.2-Fig. 4.4.

The setup in Fig. 4.2 consists of three distinct blocks. The test PLL is the one whose jitter performance is being evaluated. The supply noise and ground noise which could arise in an actual chip are emulated here using the inverter and package model blocks. In the actual chip, pad drivers are used at the output of the phase-locked loop for driving the load. This is emulated by Inverter1 in Fig. 4.2. Inverter2 represents a second phase-locked loop on the chip running possibly at a different frequency and creating noise. The inductors used in Fig. 4.2 represent the bond-wire inductors, the capacitors across them are parasitic capacitors. The inductances KS1 and KG1 represent the mutual inductance between the parasitic inductances. Here, the worst case inductance is used for the noise simulation.

The Inverter1 and package model blocks in Fig. 4.2 are used to generate the Noisysupply in Fig. 4.2. This signal emulates the switching noise in the supply due
FIGURE 4.2. Set up for simulating the power supply noise and ground bounce.
FIGURE 4.3. Set up of the Test PLL for simulating the supply noise.
to the presence of parasitic inductors and capacitors. The inputs to the inverters are ideal pulses, IN1 and IN2 where IN1 emulates the output of the test PLL output at lock. The Noisy supply in Fig. 4.2 is taken from the drain of the PMOS transistor of inverter1. This goes as data input to supply.dat in Fig. 4.3. The supply.dat is a piecewise linear source whose positive node is connected to a power supply filter and the negative node is connected to the ideal ground. The function of the power supply filter is to filter out the high frequency components in the supply noise. The output of the power supply filter is the net named noisysupply in Fig. 4.3. This net is in turn connected to the ports noisy supply of the phase/frequency detector and the VCO. Addition of noise to the charge pump block slows down the simulation roughly by 10 times. But the noise from the charge pump block can be ignored because the noise gets filtered by the loop filter. So the charge pump is assumed to be an ideal block in this setup. The output noise from the phase/frequency detector also gets
filtered by the loop filter, but the noise is added to the phase/frequency detector block for the sake of completion. The outputs of the phase/frequency detector and the VCO are referenced with respect to an ideal ground.

The Inverter1 and package model blocks in Fig. 4.2 are also used to generate the Noisy ground in Fig. 4.2. This signal emulates the ground bounce due to the presence of parasitic inductors and capacitors. The inputs to the inverters are ideal pulses, IN1 and IN2 where IN1 emulates the output of the test PLL output at lock. The Noisy ground in Fig. 4.2 is taken from the source of the NMOS transistor of inverter1. This goes as data input to supply.dat in Fig. 4.4 The ground.dat is a piecewise linear source whose positive node is connected to the net named noisyground in Fig. 4.4. This net is in turn connected to the ports noisyground of the phase/frequency detector and the VCO. The outputs of the phase/frequency detector and the VCO are referenced with respect to the noisyground.

The Inverter1 and package model blocks in Fig. 4.2 are not used for the case of simulating a sinusoidal or square wave noise in the power supply. To simulate these noises, the setup in Fig. 4.3 is used. An ideal sine wave or a square wave is used as data input to supply.dat.

4.3. Simulation

In order that the introduction of the transistor-level circuits do not impair the speed of simulation, the simulation of the inverters is carried out separately using HSPICE. The inputs to the inverters are ideal pulses, IN1 and IN2 where IN1 emulates the output of the Test PLL output at lock. The noisy supply and noisy ground outputs are taken from the Inverter1 and applied to the Test PLL which is simulated using SpectreS, since HSPICE cannot handle Verilog-A. The HSPICE
netlist for the Inverter1 to generate Noisy ground is given in Appendix F, the one to generate Noisy supply is given in Appendix G. The code for the phase frequency detector for the supply noise setup is given in Appendix H. The code for VCO for the supply noise setup is given in Appendix I. The code in Appendix J is for the VCO with the factor used to match the behavioral simulations with the HSPICE simulations. The code for the phase frequency detector for the ground bounce setup is given in Appendix K. The code for VCO for the ground bounce setup is given in Appendix L. For evaluating the output jitter due to supply and ground noise, the following cases are identified.

1. Supply Ldi/dt noise

   This is the case when the supply noise alone switches due to the surge current through the parasitic elements. The setup of test PLL is shown in Fig. 4.3.

2. Ground Ldi/dt noise

   This case simulates the ground bounce due to the surge current through the parasitic elements. The setup for test PLL is shown in Fig. 4.4. The simulation for the above two cases is done for a supply of 3.3V, a load capacitor of 15pF, and at a temperature of -40C with the PLL reference frequency at 100MHz for both the model and HSPICE. The supply, load capacitor and the temperature chosen are arbitrary and can be changed for a different setup. The simulation is done for various inverter sizes of \((W/L)_{max} = 1000, 2000, 3000, 4000\) with \((W/L)_{pmos}/(W/L)_{nmos} = 4\) in each of these cases. The results are summarized in Fig. 4.6-Fig. 4.11. The observation window for the period jitter is 1000 cycles.

3. Sine noise at supply

   The setup of the test PLL used for this case is shown in Fig. 4.3 with supply.dat
replaced with a sinusoidal source. The simulation is done for supply of 3.3 V, a load capacitor of 15pF, and at a temperature of 50°C. The supply, load capacitor and the temperature chosen are arbitrary and can be changed for a different setup. The results are given in Fig. 4.9- Fig. 4.10. The observation window for the period jitter is 1000 cycles.

4. Square noise at supply

The set up of the Test PLL is shown in Fig. 4.3 with supply.dat replaced with a pulse source. The simulation is done for supply of 3.3 V, a load capacitor of 15pF, and at a temperature of 50°C. The supply, load capacitor and the temperature chosen are arbitrary and can be changed for a different setup. The results are given in Fig. 4.11- Fig. 4.12. The observation window for the period jitter is 1000 cycles.

A typical noisy-supply waveform is shown in the figure Fig. 4.5. As the amplitude and frequency of ringing in the waveform increases, the output jitter also increases.

4.4. Measurement of period jitter

Jitter is the random variation of the zero-crossings of a timing signal. For measuring period jitter, first the zero-crossings of the timing waveform is collected and the period of each cycle is evaluated from this data. The difference between the maximum and the minimum period over the given observation window is calculated as the period jitter.
4.5. Results

A table of comparison for the speed of simulation for the model and HSPICE is given in Fig. 4.6. The simulations are done in Sun Sparc Ultra10 machine.

From Fig. 4.6, the noise simulation with behavioral model is roughly 310 times faster (best case) and 125 times faster (worst case) than the HSPICE noise simulations. The accuracy of the model depends on the accurate evaluation of the non-linear transfer function from the various noisy nodes to the output. By modeling the noise transfer function of the circuit as closely as possible, 100% accuracy for the behavioral noise simulations compared with the HSPICE noise simulations is obtained.
 FIGURE 4.6. Comparison of speed between behavioral simulations.

4.6. Inferences

Figure 4.7 shows the variation of period jitter with the inverter sizes for a given reference frequency (of 100MHz) of the VCO output for the case of supply $L_{di/dt}$ noise. The ringing in the noise waveforms generated from the inverters in fig. 4.2 decides the noise injection level of $supply.dat$ in Fig. 4.3. As the transistor sizing of the MOS devices in the inverters increase, the amplitude of ringing of the noisy waveform increases. This causes period jitter to increase at the VCO output for a given reference frequency.

The VCO is the most dominant source of noise to the output jitter. The the power supply rejection ratio of the VCO block in the circuit was not taken into account in the model while modeling the VCO. So there was a discrepancy between the HSPICE simulations and the simulations from the model.

In order to match the simulations from the model to those from HSPICE, a fit was made for each data point from the model simulation. This is accounted for by the term factor in the VCO code shown in Appendix J. This factor varied
Figure 4.7. Comparison of results for model and HSPICE for Ldi/dt supply noise.

with noise frequency in a non-linear manner showing that the noise transfer function from VCO input to output is also non-linear.

The remaining figures from Fig. 4.8-Fig. 4.12 show the results from the model only, because it was very hard to have the model simulation match with the HSPICE simulation.

Figure 4.8 shows the variation of period jitter with the inverter sizes for a given reference frequency (of 100MHz) of the VCO output for the case of ground Ldi/dt noise. The trend shown is that as the transistor sizing of the MOS devices in the inverters increase, the amplitude of ringing of the noisy waveform increases. This causes period jitter to increase for increasing inverter sizes, for a given reference frequency.

The magnitude of jitter is of the order of nanoseconds (high) for a reference frequency 100MHz. This can be explained with the aid of Fig. 4.13 and Fig. 4.14. Fig. 4.13 is the plot of period of the output versus time from the behavioral simu-
FIGURE 4.8. Comparison of results for model for $L_{di/dt}$ ground noise.

lation and Fig. 4.14 is for the corresponding HSPICE simulation. The simulation is done for $(W/L)_{nmos} = 1000$ and reference frequency of $100\text{MHz}$. The variation of output period with respect to time shows that the PLL has not reached the steady state. So the jitter measured at the output is the one introduced due to the settling transients. So the period jitter is of the order of nanoseconds.

From figures Fig. 4.9-Fig. 4.12, we can see that the output jitter decreases as the noise frequency increases for a given amplitude and PLL reference frequency. This is because output jitter is shaped by the loop transfer function, which has a low-pass characteristics. Thus the output jitter due to noise of $100\text{MHz}$ is lesser than the jitter due to $10\text{MHz}$. Jitter of the order of nanoseconds arises even in the steady state if the noise switches at a fast rate.

The jitter characteristic for $L_{di/dt}$ noise depends on the ringing of the waveform at that frequency. Since ringing is higher for higher inverter sizes, the output jitter is also higher. The simulated jitter from the model is matched from the
simulated jitter from HSPICE using a factor in the voltage controlled oscillator. Physically, this factor represents the value of the transfer function for the noise from supply or ground (as the case may be) to the VCO output. From simulations, it was found that the factor varies nonlinearly with frequency and noise voltage injected. The maximum (peak-to-peak) period jitter depends on the observation window also.
FIGURE 4.10. Output jitter for sinusoidal noise injection with reference at 169MHz.

FIGURE 4.11. Output jitter for square noise injection with reference at 50MHz.
FIGURE 4.12. Output jitter for square noise injection with reference at 169MHz.

FIGURE 4.13. Variation of output period with respect to time from behavioral simulation.
FIGURE 4.14. Variation of output period with respect to time from HSPICE simulation.
5. CONCLUSION

A method of simulating power supply noise for designing a low-jitter phase-locked loop has been developed, in Verilog-A. The set up can be integrated in a commonly used design environment like CADENCE. The method is fast. The accuracy of the model depends on the accurate evaluation of the non-linear transfer function from the various noisy nodes to the output. By modeling the noise transfer function of the circuit as closely as possible, 100% accuracy for the behavioral noise simulations compared with the HSPICE noise simulations is obtained. The period jitter of the order of nanoseconds in the case of \( \frac{Ldi}{dt} \) supply and ground noise can be accounted from the fact that the settling transient gives rise to a higher jitter. For the sine wave or square wave injection case, jitter of the order of nanoseconds arises even in steady state, when the noise switches at a fast rate.

The model does not match with the HSPICE for some cases of noise simulations because the noise transfer function from various nodes (like the phase/frequency detector node, VCO node) has been ignored while modeling. This is because this noise transfer function of the circuit is not characterized at this point of time.

For no-noise case, the model is 310 times faster than circuit level simulation and, after adding noise, it is about 125 times faster than the circuit level simulation. The simulations were done using a Sun Sparc Ultra10 workstation.

In addition to doing a noise analysis for measuring the output jitter, the model can be used to simulate phase-locked loops for various applications. This enables the designer for quick identification of the design space in a top-down approach and to complete a bottom-up verification for an existing design.
6. SCOPE FOR FURTHER RESEARCH

The following ideas can be identified as scope for further research.

In the phase frequency detector, can try to include the dead band zone effect (this is the insensitivity of the phase detector to small phase differences between the signals, arising due to the finite delays in the feedback path of the phase/frequency detector. The phase frequency detector cannot respond to small phase errors). But since the silicon was free from this non-ideality, it was not included here.

In the charge pump, the non-idealities due to charge injection errors and the loading effect due to the voltage controlled oscillator can be included.

In each of the blocks, the output could be modified to include jitter due to thermal noise and other parasitic noises. More importantly, it would be desirable to quantify those according to the actual measured values.

In order to investigate the effects of noise due to a divider on the output jitter, a divider can be inserted in the feedback path.

Another interesting feature would be to identify the noise sensitivity of each block and figure out the transfer function from each node to output. The problem is complicated for noise sources like the power supply noise which do not have a white noise characteristic.

Modeling the substrate noise coupling through the loop filter and the VCO to the reference voltage could be another field of research.

Due to the decoupling between the clock noise generator and the PLL, the bond wire inductors can typically produce a standing wave whose magnitude is difficult to be predicted by the model.
BIBLIOGRAPHY


APPENDICES
APPENDIX A. Verilog-A code for the phase frequency detector shown in Fig. 3.2

'include "constants.h"
'include "discipline.h"

module pfd(VCO, REF, UP, DOWN);

input VCO, REF;
output UP, DOWN;

parameter real vthresh=1.65;
parameter real Vhi=3.3;
parameter real Vlo=0;
parameter real iamp=5e-6;
parameter real tdel=0;
parameter real trise=1e-12;
parameter real tfall=1e-12;

electrical VCO, REF, UP, DOWN;

integer up, down, flag1, flag2;

analog begin

@(cross(V(REF)-vthresh, +1))
begin
if(flag2==0)
flag1=1;
else
begin
flag1==0;
flag2==0;
end
end

@(cross(V(VCO)-vthresh, +1))
begin
if(flag1==0)
flag2=1;
else
begin
flag2==0;
flag1==0;
end
end

end
end
@(cross(V(REF)-vthresh,-1))
flag1=0;
@(cross(V(VCO)-vthresh,-1))
flag2=0;

case(1)
  (flag1==1) : up=Vhi;
  (flag2==1) : down=Vhi;
endcase
V(UP)<+transition(up,0,1e-12,1e-12);
V(DOWN) <+transition(down,0,1e-12,1e-12);
end
endmodule
APPENDIX B. Verilog-A code for the charge pump shown in Fig. 3.2

`include "constants.h"
`include "discipline.h"

module cp(OUT, UP, DOWN);
  output OUT;
  input UP;
  input DOWN;

  parameter real vthresh=1.65;
  parameter real iamp=5e-6;
  parameter real tdel=0;
  parameter real trise=1e-12;
  parameter real tfall=1e-12;
  electrical OUT, UP, DOWN;

  real iout;
  analog begin

    if (V(UP))
      iout=-iamp;
    else if (V(DOWN))
      iout=iamp;
    else if ((V(UP) == 0) && (V(DOWN) == 0))
      iout=0;

    I(OUT) <+ transition(iout, tdel, tfall, trise);

  end

endmodule
APPENDIX C. Verilog-A code for the Input VCO and VCO shown in Fig. 3.2

```
#include "constants.h"
#include "discipline.h"
module vco(out,in);
    input in; output out;
    electrical out,in;
    parameter real dir=1;
    parameter real Vmin=0;
    parameter real Vmax=1 from(Vmin:inf);
    parameter real Fmin=100e6 from (0:inf);
    parameter real Fmax=2*Fmin from(Fmin:inf);
    parameter real tt=0.01/Fmax from(0:inf);
    parameter real Vlo=0,Vhi=3.3;
    parameter real ttol=1u/Fmax from (0:1/Fmax);
    real freq,phase,flag;
    integer n;
    analog begin
        freq=(V(in)-Vmin)*(Fmax-Fmin)/(Vmax-Vmin) + Fmin;
        phase=2*3.14*idtmod(freq,0.0,1.0,-0.5);
        @((cross(phase+3.14/2,dir,ttol) or
            cross(phase-3.14/2,dir,ttol))
        begin
            n=(phase>=-3.14/2) && (phase<3.14/2);
        end
        V(out)<+transition(n?Vhi:Vlo,0,tt);
    end
endmodule
```
APPENDIX D. Verilog-A code for the phase frequency detector/charge pump shown in Fig. 3.8

`include "constants.h"
`include "discipline.h"

module pfd(REF, VCO, OUT);
inout REF, VCO, OUT;
electrical REF, VCO, OUT;

    parameter real vthresh=0.5;
    parameter real Ipump = 100e-6;

    parameter real tdel=0;
    parameter real trise=1e-12;
    parameter real tfall=1e-12;

    integer flag1,flag2;
    integer state;

    analog begin
        @(cross (V(REF)-vthresh,+1))
        begin
            flag1=1;
        end
        @(cross (V(REF)-vthresh,-1))
        begin
            flag1=0;
        end
        @(cross (V(VCO)-vthresh,+1))
        begin
            flag2=1;
        end
        @(cross (V(VCO)-vthresh,-1))
        begin
            flag2=0;
        end

        if(flag1==1 && flag2==0 )
            state =1;
        else if(flag1==0 && flag2==1)
state = -1;
else if(flag1 == 0 && flag2==0 || flag1==1 && flag2==1)
begin
    state=0;
end
I(OUT)<+Ipump*transition(state,tdel,tfall,trise);
end
endmodule
APPENDIX E. Verilog-A code for the VCO in set up in Fig. 3.8

`include "constants.h"
`include "discipline.h"

module vco(IN, OUT);
  inout IN, OUT;
  electrical IN, OUT;

  parameter real gain=62e3, fc=7000000, amp=1;
  real inst_freq;
  real wc; // center freq in rad
  real phase_lin; // wc*time
  real phase_nonlin; // integ(k*f(t)) of phase
  real sig_val;
  integer num_cycles;

  analog begin
    wc=2*3.14*fc;

    phase_lin=wc*$realtime;
    num_cycles = phase_lin/(2*3.14);
    phase_lin=phase_lin-num_cycles*2*3.14;

    phase_nonlin=2*3.14*gain * idt(V(IN),0,0,1e-14);

    sig_val= amp*cos(phase_lin+phase_nonlin);

    V(OUT) <+ sig_val;
  end
endmodule
APPENDIX F. HSPICE netlist for the Inverter1 shown in Fig. 4.2 used to generate Noisy ground

* The PLL model parameter file
.include models.level13
MP1 OUT1 IN1 supply supply pshort w=6400u l=1.6u
MN1 OUT1 IN1 noisegnd1 0 nshort w=1600u l=1.6u
LG1 noisegnd1 0 16.36n
CG1 noisegnd1 0 0.2522p

MP2 OUT2 IN2 supply supply pshort w=6400u l=1.6u
MN2 OUT2 IN2 noisegnd2 0 nshort w=1600u l=1.6u
LG2 noisegnd2 0 16.36n
CG2 noisegnd2 0 0.2522p

K1 LG1 LG2 0.78n
CL1 OUT1 0 15p
CL2 OUT2 0 15p

VDD supply 0 3.3
* Ideal input pulses
VIN1 IN1 0 pulse(0 3.3 0 0.2n 0.2n 5n 10n)
VIN2 IN2 0 pulse(0 3.3 0 0.2n 0.2n 5n 10n)
.option h9007 gmin=1n method=gear lvltim=2
.temp -40
.tran 100p 10u START=9.9u
.print V(noisegnd1)
.end
APPENDIX G. HSPICE netlist for the Inverter shown in Fig. 4.2 used to generate Noisy supply

* The hspice netlist of the circuit is found in the file "netlist"
.include models.level13

LS1 supply noisesupply1 16.36n
MS1 OUT1 IN1 noisesupply1 supply pshort w=6400u l=1.6u
MN1 OUT1 IN1 0 0 nshort w=1600u l=1.6u
CS1 noisesupply1 supply 0.2522p

LS2 supply noisesupply2 16.36n
CS2 noisesupply2 supply 0.2522p
MP2 OUT2 IN2 noisesupply2 supply pshort w=6400u l=1.6u
MN2 OUT2 IN2 0 0 nshort w=1600u l=1.6u

K1 LS1 LS2 0.78n

CL1 OUT1 0 15p
CL2 OUT2 0 15p

VDD supply 0 3.3
VIN1 IN1 0 PULSE(0 3.3 0 0.2n 0.2n 5n 10n)
VIN2 IN2 0 PULSE(0 3.3 0 0.2n 0.2n 5n 10n)

.option h9007 gmin=1n method=gear lvltim=2
.temp -40
.tran 100p 10u START=9.9u
.print V(noisesupply1)
.end
APPENDIX H. Verilog-A code for the phase frequency detector shown in Fig. 4.3

'include "constants.h"
'include "discipline.h"

module pfd_supplynoise(VCO, REF, UP, DOWN, noise);
  input VCO, REF, noise;
  output UP, DOWN;

  parameter real vthresh = 1.65;
  parameter real Vhi = 3.3;
  parameter real Vlo = 0;
  parameter real iamp = 5e-6;
  parameter real tdel = 0;
  parameter real trise = 1e-12;
  parameter real tfall = 1e-12;
  electrical VCO, REF, UP, DOWN, noise;

  integer up, down, flag1, flag2;
  real noisy;

  analog begin
    noisy = V(noise);
    @(cross(V(REF) - vthresh, +1))
      begin
        if(flag2 == 0)
          flag1 = 1;
        else
          begin
            flag1 = 0;
            flag2 = 0;
          end
      end
    @(cross(V(VCO) - vthresh, +1))
      begin
        if(flag1 == 0)
          flag2 = 1;
        else
          begin
            flag2 = 0;
          end
  end
flag1=O;
end
end @(cross(V(REF)-vthresh,-1))
flag1=O;
@(cross(V(VCO)-vthresh,-1))
flag2=O;
case(1)
  (flag1==1) : up=noisy;
  (flag2==1) : down=noisy;
  default begin up=Vlo;down=Vlo; end
endcase

V(UP)<transition(up,0,1e-12,1e-12);
V(DOWN)<transition(down,0,1e-12,1e-12);
end
endmodule
APPENDIX I. Verilog-A code for the VCO shown in Fig. 4.2, without the factor

`include "constants.h"
`include "discipline.h"
module vco_supplynoise(out,in,noise);
  input in,noise; output out;
  electrical out,in,noise;
parameter real dir=1;
parameter real Vmin=0;
parameter real Vmax=1 from(Vmin:inf);
parameter real Fmin=100e6 from (0:inf);
parameter real Fmax=2*Fmin from(Fmin:inf);
parameter real tt=0.01/Fmax from(0:inf);
parameter real Vlo=0,Vhi=3.3;
parameter real ttol=1u/Fmax from (0:1/Fmax);
real freq,phase,flag,noisy;
integer n;

analog begin
  noisy=V(noise);
  freq=(V(in)-Vmin)*(Fmax-Fmin)/(Vmax-Vmin) + Fmin;
  phase=2*3.14*idtmod(freq,0.0,1.0,-0.5);
  @(cross(phase+3.14/2,dir,ttol) or
cross(phase-3.14/2,dir,ttol))
  begin
    n=(phase>-3.14/2) && (phase<3.14/2);
  end
  V(out)<+transition(n?noisy:Vlo,0,tt);
end
endmodule
APPENDIX J. Verilog-A code for the VCO shown in Fig. 4.2, with the factor

```
#include "constants.h"
#include "discipline.h"
module vco_noissweep(out,in,noise);
    input in,noise; output out;
    electrical out,in,noise;
    parameter real dir=1;
    parameter real Vmin=0;
    parameter real Vmax=1 from(Vmin:inf);
    parameter real Fmin=100e6 from (0:inf);
    parameter real Fmax=2*Fmin from(Fmin:inf);
    parameter real tt=0.01/Fmax from(0:inf);
    parameter real Vlo=0,Vhi=3.3;
    parameter real ttol=lu/Fmax from (0:1/Fmax);
    parameter real factor=0.7e-8;
    real freq,phase,flag,noisy,jitter,dt;
    integer n;
    analog begin
        jitter=factor*V(noise);
        freq=(V(in)-Vmin)*(Fmax-Fmin)/(Vmax-Vmin) + Fmin;
        freq=freq/(1+dt*freq);
        phase=2*3.14*idtmod(freq,0.0,1.0,-0.5);
        @(cross(phase+3.14/2,dir,ttol) or
            cross(phase-3.14/2,dir,ttol))
            begin
                n=(phase>=-3.14/2) && (phase<3.14/2);
                dt=jitter;
            end
        V(out)<+transition(n?Vhi:Vlo,0,tt);
    end
endmodule
```
APPENDIX K. Verilog-A code for the phase frequency detector shown in Fig. 4.4

```
'include "constants.h"
'include "discipline.h"

module pfd_noise(VCO, REF, UP, DOWN, noise);
    input VCO, REF, noise;
    output UP, DOWN;

    parameter real vthresh=1.65;
    parameter real Vhi=3.3;
    parameter real Vlo=0;
    parameter real iamp=5e-6;
    parameter real tdel=0;
    parameter real trise=1e-12;
    parameter real tfall=1e-12;
    electrical VCO, REF, UP, DOWN, noise;

    integer up, down, flag1, flag2;
    real noisy;

    analog begin
        noisy=V(noise);
        @(cross(V(REF)-vthresh,+1))
            begin
                if(flag2==0)
                    flag1=1;
                else
                    begin
                        flag1==0;
                        flag2=0;
                    end
            end
        @(cross(V(VCO)-vthresh,+1))
            begin
                if(flag1==0)
                    flag2=1;
                else
                    begin
                        flag2=0;
                        flag1=0;
                    end
```
end
end
@(cross(V(REF)-vthresh,-1))
flag1=0;
@(cross(V(VCO)-vthresh,-1))
flag2=0;

case (1)
  (flag1==1) : up=Vhi;
  (flag2==1) : down=Vhi;
  default begin up=Vlo+noisy;down=Vlo+noisy; end
endcase

V(UP)<+transition(up,0,1e-12,1e-12);
V(DOWN)<+transition(down,0,1e-12,1e-12);
end
endmodule
APPENDIX L. Verilog-A code for the VCO shown in Fig. 4.4, without the factor

'include "constants.h"
'include "discipline.h"

module vco_noise(out,in,noise);
  input in,noise; output out;
  electrical out,in,noise;

  parameter real dir=1;
  parameter real Vmin=0;
  parameter real Vmax=1 from(Vmin:inf);
  parameter real Fmin=100e6 from (0:inf);
  parameter real Fmax=2*Fmin from Fmin:inf);
  parameter real tt=0.01/Fmax from(0:inf);
  parameter real Vlo=0,Vhi=3.3;
  parameter real ttol=1u/Fmax from (0:1/Fmax);

  real freq,phase,flag,noisy;
  integer n;

  analog begin
    noisy=V(noise);

    freq=(V(in)-Vmin)*(Fmax-Fmin)/(Vmax-Vmin) + Fmin;

    phase=2*3.14*idtmod(freq,0,0.1.0,-0.5);

    @(cross(phase+3.14/2,dir,ttol) or
cross(phase-3.14/2,dir,ttol))
begin
  n=(phase>=-3.14/2) && (phase<3.14/2);
end

V(out)<+transition(n?Vhi:Vlo+noisy,0,tt);
end
endmodule