
Course Project: Phase Noise Simulations

1 Introduction

Output phase noise is an important performance parameter of a PLL, especially one intended for use as a frequency synthesizer. To analyze the phase noise of our PLL, we will use two types of simulations in the Cadence Analog Design Environment: PSS (Periodic Steady-State Analysis) and Pnoise (Periodic Noise Analysis). These simulations are used with circuits that have a periodic output (as many RF circuits do), and what they do is simulate the circuit over one period of the lowest harmonic in the system, and then use this to estimate noise and a number of other things.

The intent of this document is to guide you through the phase noise simulations for your project. For a more detailed explanation of what is actually going on in the simulations and what the different parameters do, refer to the Cadence help documentation. To access this, first open a webbrowser such as netscape, then click on **Help -> Cadence Documentation**. A window should appear with documentation listings for several Cadence products. Select **Spectre RF**, then **Spectre RF User Guide**. Here you will find detailed explanations for the simulations, as well as examples of how to configure different types of RF circuits.

2 Method

The PLL itself is a periodic system, so ideally we would just run PSS and Pnoise simulations for the entire PLL. However, due to the complexity of the circuits, the simulations would take prohibitively long to run, so we will use a different approach. This will be to divide the PLL into three blocks, simulate the noise for each block, and then combine the results using a Matlab script to find the total phase noise at the output.

3 Phase Noise Simulations

This section describes the phase noise simulation for each of the blocks. The block divisions coincide with the divisions for the three projects.

3.1 Phase Detector, Charge-pump, and Loopfilter

1. This procedure will be different depending on whether or not you are using a type 1 or a type 2 PLL. If you are using a type 1 PLL, create a schematic with your Phase Detector, Charge-pump (if you are using one) and Loopfilter (LPF) connected as they would be in the PLL. Leave the output of the LPF open, and drive each PD input with a square wave (`vpulse` in `analogLib`) at the reference frequency you have

chosen. In the `vpulse`, set the rise and fall times to 5 ns, and fill in `f_in` for the Frequency name for $1/\text{period}$. Depending on the phase detector you are using, you will need to add some offset between the two inputs to the PD (eg., the equivalent of 90 degrees for the XOR phase detector) by adding a delay to one of the `vpulse` sources. If you are using a type 2 PLL, create a schematic with your Phase Detector (PD) and Charge-pump (CP) (if you are using one) connected as they would be in the PLL. Connect the output of the Charge-pump to a 10 fF capacitor, and drive both PD inputs with a square wave (`vpulse` in `analogLib`) at the reference frequency you have chosen. In the `vpulse`, set the rise and fall times to 5 ns, and fill in `f_in` for the Frequency name for $1/\text{period}$. If you are not using a PFD phase detector, you will need to add some offset between the two inputs to the PD. Further discussion on this will be provided in class.

2. Run a transient simulation, and note how long it takes for the output voltage of the loopfilter to settle to a steady state value (we will refer to this as `tstab`, for the time required to arrive at a stable operating point).
3. Disable the transient simulation, and from the **Choose Analysis** box, select PSS.
4. A fundamental tone from your `vpulse` should be listed. If anything else is listed, in this box, delete it.
5. Click on **Beat Frequency** and **Auto Calculate**.
6. Under Output harmonics, set Number of harmonics to 7.
7. Set the Accuracy Default to moderate.
8. Fill in the value you observed previously for `tstab`.
9. Click on OK to close this box.
10. From the **Choose Analysis** box, select Pnoise.
11. Set Sweeptype to absolute
12. Set Frequency Sweep Range to Start at 10K and Stop at 10M.
13. Set Sweep Type to Linear with a step size of 10K.
14. Under Sidebands, set Maximum sideband to 7.
15. Set Output to voltage, and for the Positive Output Node, select the output node of your loopfilter. Leave the Negative Output Node blank.
16. Set Input Source to none.
17. Run the simulation.

18. To plot the results, choose Direct Plot - Main Form.
19. Under PSS, you can look at the voltage of the LPF output in both the time and frequency domain. The frequency domain levels will give you some indication of the spur levels you will have at the output when you do the DFT simulations.
20. Under Pnoise, you can plot the Output Noise. To export the data for use in the Matlab script, type:

```
ocnPrint(db(getData("out" ?result "pnoise")) ?output "~/PCL-pn.txt" ?numberNotation  
'none')
```

3.2 Voltage Controlled Oscillator

1. Create a schematic with your Voltage Controlled Oscillator. Connect the outputs to the a copy of the first stage of your Prescaler to include loading effects (you only need to add the input transistors, not the entire Prescaler as this would slow down the simulation). Use a voltage source to bias the control voltage input at 1 GHz.
2. Run a transient simulation, and note how long it takes for any transient effects from start-up to disappear (we will refer to this as `tstab`, for the time required to arrive at a stable operating point). Also obtain an estimate of the oscillation frequency.
3. Disable the transient simulation, and from the **Choose Analysis** box, select PSS.
4. Click on **Beat Frequency** and enter your frequency estimate in the box. Be sure that Auto Calculate is not highlighted.
5. Under Output harmonics, set Number of harmonics to 7.
6. Set the Accuracy Default to moderate.
7. Fill in the value you observed previously for `tstab`.
8. Click on Oscillator, and select the output as your Oscillator Node. If your design is differential, select the other output node as the Reference Node, otherwise leave it blank.
9. Click on OK to close this box.
10. From the **Choose Analysis** box, select Pnoise.
11. Set Sweeptype to relative, and set the Relative Harmonic to 1.
12. Set Frequency Sweep Range to Start at 10K and Stop at 10M.
13. Set Sweep Type to Linear with a step size of 10K.
14. Under Sidebands, set Maximum sideband to 7.

15. Set Output to voltage, and for the Positive Output Node, select the VCO output node. Leave the Negative Output Node blank unless your design has differential outputs, in which case select the other output node.
16. Set Input Source to none.
17. Run the simulation.
18. To plot the results, choose Direct Plot - Main Form.
19. Under PSS, you can look at the waveform of the VCO output in both the time and frequency domain.
20. Under Pnoise, you can plot the Output Noise. To export the data for use in the Matlab script, type:

```
ocnPrint(phaseNoise(1 "pss_fd" ?result "pnoise") ?output "~/VCO_pn.txt"
?numberNotation 'none)
```

3.3 Divider

1. Create a schematic with your Prescaler. Connect the outputs to the a copy of the first stage of your Prescaler to include loading effects (you only need to add the input transistors, not the entire Prescaler as this would slow down the simulation). Use a vpulse voltage source to drive the inputs at 1 GHz, and make sure that you enter a name in the property "Frequency name for 1/period".
2. Run a transient simulation, and note how long it takes for any transient effects from start-up to disappear (we will refer to this as `tstab`, for the time required to arrive at a stable operating point).
3. Disable the transient simulation, and from the **Choose Analysis** box, select PSS.
4. Click on **Beat Frequency** and enter the output frequency in the box (e.g., 0.5 GHz if your Prescaler divides by 2). Be sure that Auto Calculate is not highlighted.
5. Under Output harmonics, set Number of harmonics to 7.
6. Set the Accuracy Default to moderate.
7. Fill in the value you observed previously for `tstab`.
8. Click on OK to close this box.
9. From the **Choose Analysis** box, select Pnoise.
10. Set Sweeptype to relative, and set the Relative Harmonic to 1.
11. Set Frequency Sweep Range to Start at 10K and Stop at 10M.

12. Set Sweep Type to Linear with a step size of 10K.
13. Under Sidebands, set Maximum sideband to 7.
14. Set Output to voltage, and for the Positive Output Node, select the divider output node. Leave the Negative Output Node blank (it will default to gnd).
15. Set Input Source to none.
16. Run the simulation.
17. To plot the results, choose Direct Plot - Main Form.
18. Under PSS, you can look at the waveform of the divider output in both the time and frequency domain.
19. Under Pnoise, you can plot the Output Noise. To export the data for use in the Matlab script, type:

```
ocnPrint(phaseNoise(1 "pss_fd" ?result "pnoise") ?output "~/DIV_pn.txt"  
?numberNotation 'none')
```

4 Using the Matlab Script

1. Download the Matlab file `Phase_Noise.m` from the course website.
2. Open the file in the matlab editor, and enter in the parameters for your PLL. You will likely have to enter in a different loopfilter transfer function, unless you are using the one that was supplied with the VerilogA framework.
3. Start the Matlab command window and import the phase noise data files that were generated from Cadence in the previous sections. You will have to tell Matlab to ignore the header lines of the data files. After you have imported the data, format the variables created to eliminate the frequency information, so that you are left with a vector of noise powers.
4. Run the `Phase_Noise.m` function, supplying it with the variable names created in the previous step.
5. Read off the phase noise totals at the pertinent offset frequencies. Include these totals and the plot that was generated in your final report.