SaberDesigner Design Examples

Brake System, Audio System, PLL Range Finder IC, and Power Converter

Release 5.1

Avant! Corporation

XG51-1

Registered in England No. 2372734
Registered Office: Aldwych House, Aldwych, London WC2B 4JP

Notice to Users: Read Before Using

Disclaimer

Preface

What You Need to Know to Use This Manual

This manual assumes that you are familiar with the following procedures:

- How to create a schematic in your design capture tool
- How to view the contents of a file
- How to use a text editor to create a file or edit the contents of a file

If this is your first time using SaberDesigner, you should run through the tutorial presented in the *Getting Started with SaberDesigner* document. This manual provides an excellent overview of the process described in this chapter.

What This Manual is About

In this manual you will find information about how to analyze a design using SaberDesigner from schematic capture, to executing analyses, through tuning parameter values.

SaberDesigner is an extremely powerful set of tools that allow you to analyze your design in numerous ways. Thus allowing you to save design time, lower production costs and create a more profitable product.

The following list gives an overview of the chapters in this manual:

Chapter 1:	*Designing a Electrohydraulic Brake System* describes the design and analysis of a brake system composed of electrical, mechanical, and hydraulic parts.
Chapter 2:	*Analyzing the Audio System Example* describes the design and analysis of the top level Audio Test System design. Each block in this top level design is described in the following chapters.

Chapter 3:	*Designing the Test Tone Generator* describes the design and analysis of the test tone block using Control System parts.
Chapter 4:	*Designing the Oscillator Block* describes the design and analysis of a 40kHz relaxation oscillator designed with multiple levels of abstraction (digital gate-level to MOS level)
Chapter 5:	*Designing the Analog-to-Digital Convertor* describes the design and analysis of SAR-based Analog-to-Digital convertor using a custom MAST template for the SAR part.
Chapter 6:	*Designing the Divide by 8 Block* describes the design and analysis of a simple divide by 8 circuit using 3 flip-flops.
Chapter 7:	*Designing the DSP circuit* describes the design and analysis of an echo feature and filtering blocks using Z-domain models.
Chapter 8:	*Designing the Power Amplifier* describes the design and analysis of a amplifier with emphasis on stress and distortion analysis.
Chapter 9:	*Designing the RLC circuit* describes the design and analysis of filter that illustrates the powerful analysis features of Saber on a simple circuit.
Chapter 10:	*Designing the Loud Speaker Circuit* describes the design and analysis of loud speaker by combining custom MAST templates and existing mechanical models.
Chapters 11:	*Range Finder IC Example* describes the design and analysis of a Phase Lock Loop design in the SaberSketch, Cadence, and Mentor Graphics' environments.
Chapters 12-16:	*Power Converter Design Example* describes the design and analysis of a two-switch, voltage-mode forward power converter in the SaberSketch and Mentor Graphics environments.
Appendix A:	*Running Batch Files* describes how to run the Power Converter example circuit by using Saber Command Script files.

Related Documents

Analogy documents:

- *SaberBook*, Saber's online help system, describes the features of the SaberSketch and SaberScope applications along with general SaberDesigner functionality, such as using the menus and printing. It also provides detailed information on each Saber command and SaberGuide form.
- *Getting Started with SaberDesigner* allows you to analyze two tutorial circuits and explore key processes in SaberDesigner.
- *Analyzing Designs Using SaberDesigner* describes how to use SaberDesigner to capture schematics, simulate designs, and optimize parameter values.

Conventions

This manual uses the following conventions:

keywords	Keywords are reserved words such as the MAST language word **template** that are shown in the font style at left.
`~/tutorial/circuit`	File and directory paths appear in the `Courier` font.
placeholder	An item shown in *italic* in a command line indicates that you must supply the specific text to be used in its place.
bold	Menu items, Dialog box names, and button names are shown in **bold**.
emphasis	Emphasized text such as a manual titles and new terms appears in italic.
[ ]	Italicized square brackets enclosing text indicate optional entries.
Single Click	Press and release a mouse button once quickly.
Press and Hold	Press a mouse button and do not release it.
Double Click	Press and release a mouse button twice quickly.

The following convention is used in this manual to indicate a menu selection from a sub-menu as shown in the following example:

Analyses > Operating Point > DC Transfer

In this example, the **DC Transfer** menu item is selected by first selecting the item **Analyses** from a top-level menu, then selecting **Operating Point** from a sub-menu, and finally selecting **DC Transfer** from a second sub-menu.

Revision History

Nov. 1999	Updated to reflect changes for Release 5.1
Mar. 1999	Updated to reflect changes for Release 5.0
Mar. 1998	Added the Power Converter Design Example
Dec. 1996	Updates to graphics to increase readability
Oct. 1996	First publication, accompanying Release 4.1 of SaberDesigner

Table of Contents

chapter 1

Designing an Electrohydraulic Brake System

The Brake System example illustrates the various analysis capabilities of Saber and InSpecs, as applied to multi-technology (mechatronic) design. The application is a brake-by-wire system, and includes a proportional solenoid valve and other hydraulic and mechanical elements, as well as electrical and electronic devices for sensing and control. This example is the basis for an SAE Technical Paper (No. 940184) titled “Design Analysis of an Electronically Controlled Hydraulic Braking System Using the Saber Simulator”.

The example shows the benefit of including all of the interdependent pieces of a design, in order to analyze important interactions among them that often drive design performance and overall system cost. InSpecs analyses are used extensively for that purpose.

To run this example you must have the following licenses: Saber, Component Library, Digital Simulation, Stress and Sensitivity

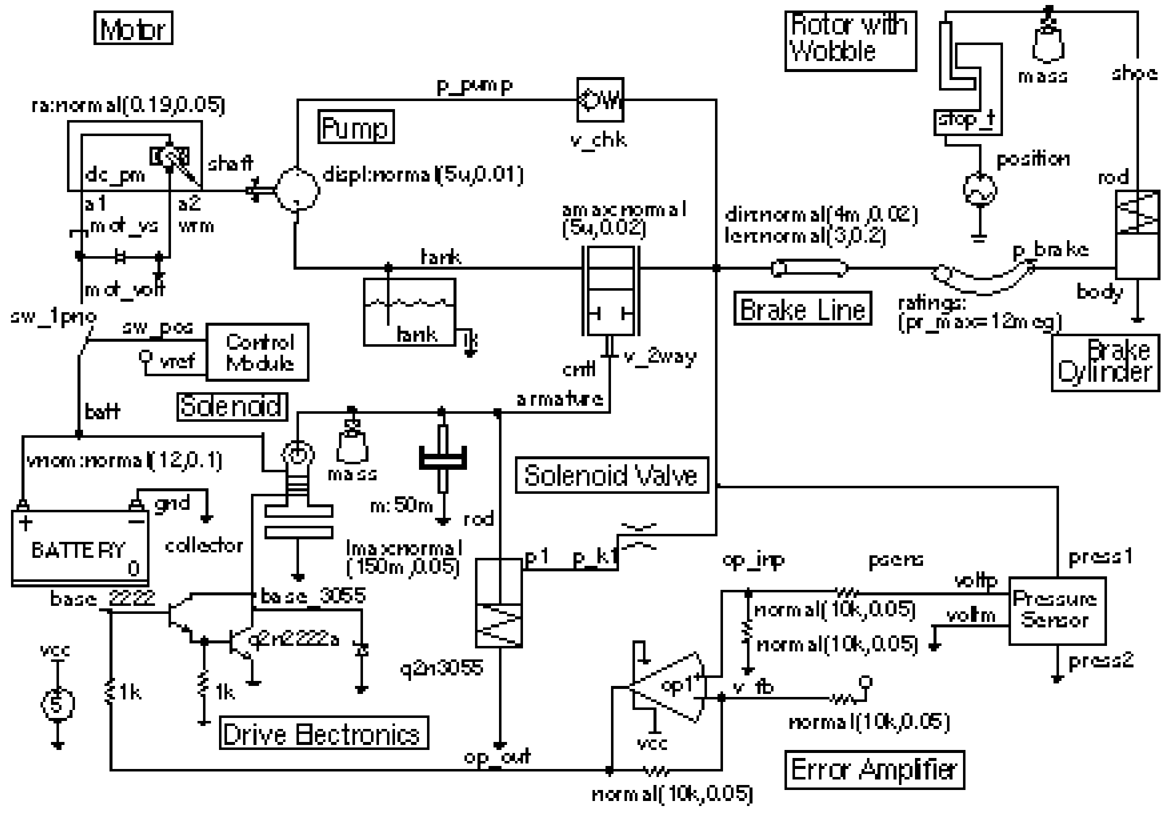
The following libraries are required: STL, OTL, CL

ViewDraw: Viewlogic Frameway license

Design Architect: Falcon Frameway license

Selecting Models for the Electrohydraulic Brake System

This schematic shows the simulation representation of an electronically controlled hydraulic braking system.



The battery powered DC motor and hydraulic pump provides system pressure. A check valve, in series with a solenoid controlled two-way valve, establishes the controlled braking pressure at the midpoint. The solenoid valve is configured with pressure feedback on the spool, through a damping orifice. This inner feedback loop provides simple pressure regulation. The actual operating pressure is then adjustable with solenoid current, because the valve spool and the solenoid armature are rigidly connected.

An outer pressure control loop uses an electronic sensor to measure the actual brake pressure. This measurement is compared with a commanded reference voltage presumably coming from a brake peddle position sensor. The difference signal drives the base of a dual-transistor amplifier, which controls the solenoid current.

The brake assembly is modeled as a single acting cylinder with spring return, plus attached load mass and a mechanical hard stop or travel limit. The hard stop models the contact point and compliance between the brake shoe and the rotor. A sinusoidally varying position source represents the instantaneous rotor transverse displacement. This allows simulation of the effects of rotor wobble on brake pressure regulation. Pressure is applied to the brake through a long rigid line, as well as a short flexible hose.

Schematics used in this example

- ex_brake - Top level Brake System schematic for SaberSketch.
- cntl_mod - Hierarchical schematic of “driver” input signal source (i.e. control module).
- pr_sens - Hierarchical schematic of pressure sensor model.

Local symbol file(s) providing the symbols for the schematic of the example circuit or system:

- `battery0` - battery symbol, calls `battery0.sin`
- `cntl_mod` - control module hierarchical symbol for `cntl_mod.sch`
- `pr_sens` - pressure sensor hierarchical symbol for pr_sens.sch

Circuit or system description file(s) for Saber input:

- `ex_brake.sin` - Top level Brake System netlist
- `battery0.sin` - Simple model of battery (ideal voltage source)

Analyzing the Design

The following list describes the process used to analyze this design:

1. "Copying the Brake System Example" on page 1-4
2. "Invoking Saber" on page 1-4
3. "Checking the Functionality of the Brake Example" on page 1-9
4. "Running Vary" on page 1-11
5. "Determining Component Sensitivity" on page 1-14
6. "Determining Component Stress Levels" on page 1-16
7. "Performing Statistical Analysis" on page 1-19

Copying the Brake System Example

Files for this example are available for SaberSketch, Design Architect, and ViewDraw. Copy the `BrakeSystem` subdirectory to a working directory of your choice as shown in the following steps. Having a local copy of the examples prevents the original example files from being overwritten.

1. Create (if necessary) and change to the directory where you would like the files to be copied.
2. Copy the `Audio` directory from the following location to your current directory:

   UNIX source - *install_home*`/example/`*your_eda*`/BrakeSystem`
   Windows NT - *install_home*`\example\`*your_eda*`\BrakeSystem`

   where *install_home* is the root directory of the Saber product installation and *your_eda* is either `SaberSketch`, `MentorGraphics`, or `Viewlogic`. You should now have a directory called `BrakeSystem`.
3. (Windows NT only) You must change the file permissions of your local copy of the designs so that they are no longer read-only as follows:
   a. Invoke Windows NT Explorer.
   b. Navigate to your local copy of the design. In the case of a design example that has more than one directory, you will need to change the permissions of all the files in each of them as described in steps c through f.
   c. In each directory, select all the files (**Edit > Select All**).
   d. Open the Properties dialog box (**File > Properties**) and select the General tab.
   e. Un-check the Read-only box.
   f. Click **OK**.

Invoking Saber

Invoke Saber by following the procedures in the appropriate topic listed as follows:

- "Opening the Brake System Design with SaberSketch" on page 1-5
- "Opening the Brake System Design with Design Architect" on page 1-5

- "Opening the Brake System Design with ViewDraw on Unix" on page 1-6
- "Opening the Brake System Design with ViewDraw on Windows NT" on page 1-6

Opening the Brake System Design with SaberSketch

To open the ex_brake design, do the following:

1. Invoke SaberSketch.using one of the methods as follows:

   (UNIX) On a command line, enter *install_home*/`bin/sketch`

   (Windows NT) **Start > Analogy >** *saberdesigner* **> SaberSketch**

   An empty schematic window appears.
2. Use the **File > Open > Design** menu choice to bring up the Open Design dialog box.
3. Browse to your local copy of the `BrakeSystem` directory.
4. Select the `ex_brake` design and open it.

Continue with the topic titled "Checking the Functionality of the Brake Example" on page 1-9.

Opening the Brake System Design with Design Architect

To open the design in Design Architect, change to the directory where the `BrakeSystem` directory is located and perform the following steps.

1. Add to your location map the softpath `SABER_EXAMPLE,` whose value is your current working directory as follows:

   ```
   setenv SABER_EXAMPLE `pwd`
   ```

2. Change your working directory to `BrakeSystem` directory and start the Design Architect schematic entry application by typing the following:

   ```
   da -sch ex_brake
   ```

3. Generate a netlist (**Saber > Netlist > Start Netlister**).
4. Start the Saber simulator (**Saber > Start SaberGuide**).

You are ready to simulate the design. Continue with the "Checking the Functionality of the Brake Example" on page 1-9 topic.

Opening the Brake System Design with ViewDraw on Unix

To open the design in ViewDraw, change to your working directory and perform the following steps:

1. Invoke Powerview with the following command:

   ```
   powerview
   ```

2. Create BrakeSystem Project as follows:
   a. In the Powerview Cockpit window, choose the **Project > Create** menu item. The Create Project dialog box appears.
   b. In the Create Project dialog box, click the **Browse** button to display the Select File dialog box.
   c. In the Select File dialog box, select (double-click) `BrakeSystem`.
   d. In the Select File dialog box, click **OK**.
   e. In the Create Project dialog box, click **OK**.
3. To open the `ex_brake.1` design from within the BrakeSystem Project:
   a. In the Powerview Cockpit window, verify that Current Project is set to *path*/`BrakeSystem`.
   b. Double-click on the **ViewDraw** icon. The File Open dialog box is displayed.
   c. In the File Open dialog box, double-click on `ex_brake.1`. The schematic appears in ViewDraw.
4. From the main Frameway session window, choose the **Saber > Netlist > Start Netlister** menu item to start the netlister.
5. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are ready to simulate the design. Continue with the "Checking the Functionality of the Brake Example" on page 1-9 topic.

Opening the Brake System Design with ViewDraw on Windows NT

1. Change directories to your `BrakeSystem` example working directory.

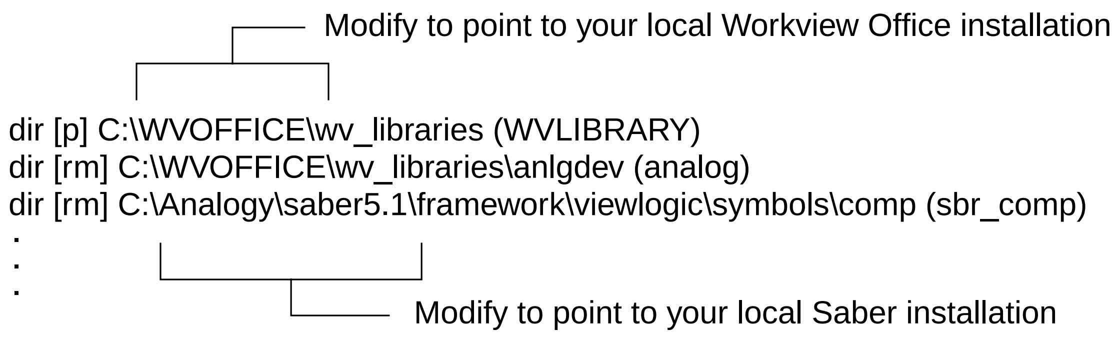
2. Copy the following file to your `System` directory:

   install_home`/framework/standard/viewdraw.ini`

   where *install_home* is the root directory of the Saber product installation. This file is template initialization file for ViewDraw that you will use as a convenient way to incorporate certain library search paths in later steps.

3. Open the `viewdraw.ini` file with a text editor. You must edit the library search paths contained in this file to match the search paths required for your local Saber installation. This a plain-text file, so be sure to save it as a text file after editing.

   Shown below is an excerpt from a `viewdraw.ini` file, showing a few of the library search-path entries.


Modify to point to your local Workview Office installation
Modify to point to your local Saber installation

```
dir [p] C:\WVOFFICE\wv_libraries (WVLIBRARY)
dir [rm] C:\WVOFFICE\wv_libraries\anlgdev (analog)
dir [rm] C:\Analogy\saber5.1\framework\viewlogic\symbols\comp (sbr_comp)
.
.
.
```

   Use the search and replace capabilities of your text editor to modify the installation-dependent portions of the search-paths so they point to the root directories for your installation.

4. Start Workview Office if it is not already active.

5. Start the Project Manager. (Click on the Project Manager icon in the Workview Office task bar).

6. Set up a project file in your `BrakeSystem` directory as follows:

   a. In the Project Manager dialog box, choose the **File > New** menu item. This activates the Project Manager wizard.

      If you have previously created a project file you may see a Project Manager Wizard message box asking whether you want to copy the library search paths that were used in that project. If this happens, click on the **Don't Copy** button.

   b. In the first dialog box of the Project Manager wizard, enter a meaningful name such as `BrakeSys` in the Project Name field.

   c. Edit the Project Directory field, if necessary, to include the correct path to your example directory (for example, `C:\brake_ex\BrakeSystem`) then click on the **Next** button to display the next dialog box.

d. Confirm that the directory location for the project file (called `BrakeSys.vpj`, for example) is the same as given in the preceding step, then click the **Next** button to display the next dialog box.

e. Again click on the **Next** button in the dialog box. (No FPGA libraries need to be added.)

f. In the ViewDraw libraries to use dialog box, add the library search paths as follows:

   a. Click on the **Import** button. The Open dialog box appears.

   b. Click on the `viewdraw.ini` file name in the Open dialog box.

   c. Click on the **Open** button. This adds the library search paths automatically from the `viewdraw.ini` file.

d. Click on the **Finish** button.

e. Confirm that the New Project Information dialog box contains the correct project information, then click on the **OK** button. This completes the `BrakeSystem` project file setup.

f. Save the project file by selecting the **File > Save** menu item.

7. Open the Brake System design:

   a. Start ViewDraw by clicking on the **ViewDraw** icon in the Workview Office task bar. The Viewdraw session window appears.

   b. Select the **File > Open** menu item. This activates the File Open dialog box.

   c. In the File Open dialog box, double-click on the schematic named `ex_brake.1`. The schematic appears in ViewDraw.

8. Pull down the ViewDraw Tools menu and check to see if a menu item containing the word **Saber** is present. This menu item will have several Saber-related entries below it, beginning with **Start SaberGuide**. If this menu item is not present, add it to the Tools menu as follows:

   - Select the **Tools > Customize** menu item.
   - Click on the **User Menu** selector button.
   - Type a name such as **Saber** in the Menu Text entry box. (The name **Saber** will be used for this menu item in all subsequent instructions.)
   - Click on the **Browse** button next to the Command entry box and navigate to the `bin` directory under your Saber installation

directory (typically `C:\`*install_home*`\bin`). Under the `bin` directory, select the file named `menu.exe`, then Click north **Open** button.

- Click on the **Add** button, then the **OK** button. The **Saber** menu item should now appear in the Tools menu.

9. From the ViewDraw session window, choose the **Tools > Saber** menu item to activate the Saber Menu window. This window contains a Saber menu, which you should use throughout the remainder of this tutorial when told to select an item from the Saber menu.
10. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.
11. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are ready to simulate the design. Continue with the topic titled "Checking the Functionality of the Brake Example" on page 1-9.

Checking the Functionality of the Brake Example

When SaberGuide appears and the design is loaded, then load the batch command file `run.scs` in the Saber transcript window: To load the batch file follow these steps.

1. In the SaberGuide Transcript window, use the **File > Saber Command File...** menu item to open the Load Command File dialog box.
2. Choose the batch file called `run.scs` and click the **Open** button.

   The `run.scs` command file executes the following Saber commands:

   a. Evaluates the DC Operating Point.

   ```
   dc
   ```

   b. Determines the Time-Domain (transient) response.

   ```
   tr (te 1, ts 1u, tn 10, ter 10u, mon 100,
   ```

   This command determines the time-domain response of the circuit during the first second in one micro second time steps, limits the number of transient Newton-Raphson iterations to 10, and specifies the truncation error for the Step Size algorithm. Analysis information is displayed every 100 time steps.

c. Limits the Signal List to the data you are specifically interested in.

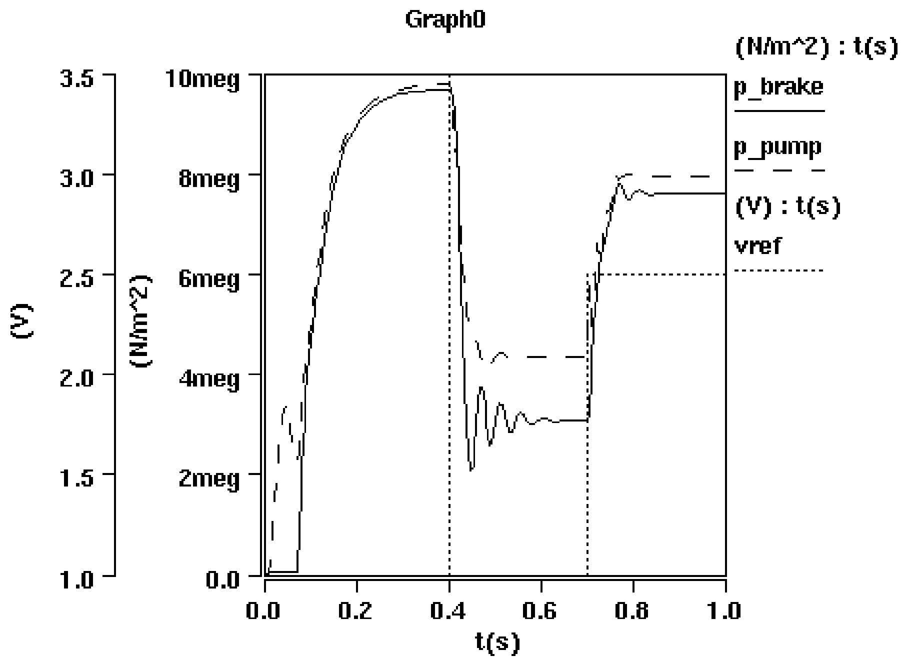
```
sigl armature base_2222 \
base_3055 batt collector\
force_mks(position.rotor)\
i(el_magnt.sole_mag) i(short.imot) mid_line\
mot_volt\
op_inp op_out p_brake p_k1 p_pump p_reg psens\
qthru_mks(line_r.rigid) rotor shaft \
shoe tank v_fb velo_mks(mass.arm_mass)\
velo_mks(mass.m_shoe) vref sw_pos
```

d. Opens the transient (`tr`) plot file.

```
pl tr
```

This executes a command script that performs a nominal transient analysis. It takes about 10 seconds to run, and brings up the plot file `ex_brake.tr`. The results show the normal operation of the brake system during the initial activation interval, with one "release/ apply" transition.

3. Plot the signals showing the pump outlet pressure (`p_pump`), the brake pressure seen at the wheel cylinder (`p_brake`), and the commanded reference voltage (`vref`):



The activation switch is turned on at time 10 milli-seconds, and the pump pressure responds very quickly. The pressure at the wheel rises also, but after a delay of approximately 75 milli-seconds. This delay is the result of the flow required to fill the excess wheel cylinder volume, moving the brake shoe up to its initial contact with the rotor. During this movement, the pressure remains low as the only force applied to the cylinder is the relatively small force of the return spring. Once in contact with the rotor, that pressure can build as the spring rate of the brake shoe on the rotor is quiet high. The system pressure is near maximum value of 10 MPa after 200 milli-seconds.

At time 400 milli-seconds the reference voltage signal, which was initially at 3.5 volts, is commanded to the lower value of 1 volt. This causes the brake pressure to fall, settling at approximately 3 MPa. There is significant brake pressure ringing following the transition, as the dynamic elements form a lightly damped resonant system.

After reduced pressure operation for 300 milli-seconds, the reference signal is raised to 3.0 volts, and the corresponding pressure recovery is observed. There is again some slight ringing, and the pressure settles at 7.6 MPa. The rate at which the pressure recovers is an important design performance indicator, and can be further analyzed by "Determining Component Sensitivity" on page 1-14.

Running Vary

Load the batch command file `run_vary.scs` in the SaberGuide transcript window: To load the batch file follow these steps.

1. Use the **File > Saber Command File...** items to open the Load Command File dialog box.
2. Choose the batch file called `run_vary.scs` and click the **Open** button.

   The `run_vary.scs` command file executes the following Saber commands.

   a. Limit the Signal List to the data you are specifically interested in.

   ```
   sigl armature p_brake p_reg
   ```

   b. Evaluate the DC Operating Point.

   ```
   dc
   ```

c. Vary the value of the orifice over three Transient analyses.

```
vary orif_se.k1/area from 30n to 300n log 3
```

d. The above command selects a logarithmic step between 30n and 300n.

```
dc (dcip dc, dcep dc
tr (te 1, ts 1u, tsmax 0.5m, tn 10, ter 100u,\
mon 100, pf tr_vary
end
```

e. Perform a Fast Fourier Transform analysis with each Transient analysis.

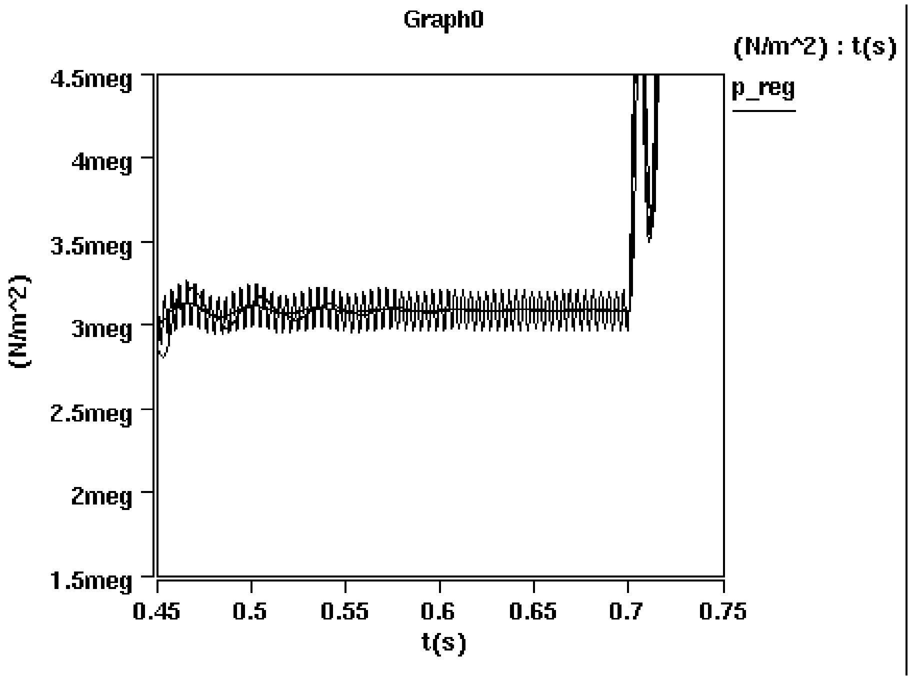
```
fft (pfin tr_vary, pfout fft_vary, xb 500m,
     xe 700m, cn p_reg, axis log
```

f. Open the plot files.

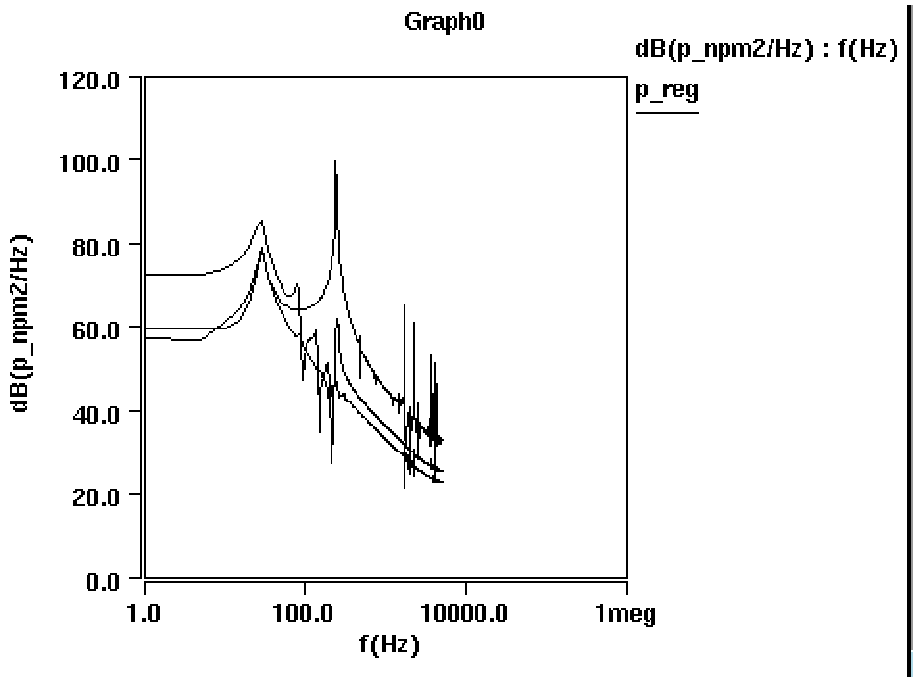
```
pl tr_vary fft_vary
```

This executes a command script that performs three transient analyses, with orifice areas 30n, 95n and 300n (m^2), and takes several minutes to run. It also performs an FFT analysis over a portion of the time domain waveform.

3. When the plot files `tr_vary` and `fft_vary` come up, plot the `p_reg` signal from the file `tr_vary`, and zoom in on the range from 0.5 to 0.7 seconds. Note the high frequency oscillation in the last of the 3 member signals (orifice area = 300n).



4. From the `fft_vary` plot file, plot `p_reg` spectrum, and note the spike at approximately 200 Hz for the last member. (Note: you must zoom in on the frequency range 10 to 10k Hz. Also, you may wish to delete the phase information).



The stabilizing effect of the solenoid valve's damping orifice is examined, by generating multiple simulations with increasing hole diameter, until continuous system "buzzing" is observed. This is a common problem for various pressure control topologies, and simulation techniques may be quite helpful in solving them.

Making trade-offs between performance requirements, such as system stability, and cost drivers, such as orifice sizing, is just one area where simulation techniques can be helpful. That is, small orifice sizing may drive up the cost of fluid filtering, so it is desirable to specify as large an area as possible, but without jeopardizing stability margins.

Determining Component Sensitivity

Load the sensitivity analysis batch command file `run_sens.scs` from the Saber transcript window:

To load the batch file follow these steps.

1. Use the **File > Saber Command File...** items to open the Load Command File dialog box.
2. Choose the batch file called `run_sens.scs` and press the **Open** button.

   The `run_sens.scs` command file executes the following Saber commands.

   a. Evaluate the DC Operating Point and save the results in `dc_nom`.

   ```
   dc (dcep dc_nom
   ```

   b. Perform Sensitivity analyses on the listed parts.

   ```
   open(pf tr_sens, df tr_sens)
   ```

   The above command appends the listed files to accommodate multi-member graphing.

```
sens (sparl \
mu \
line_r.rigid/len \
line_r.rigid/din \
damper_t.arm_d/d \
pr_sens.press_sens\lag.sen_filt/w \
pum_mot.pump/displ \
dc_pm.mot/laa \
dc_pm.mot/ra \
mass.arm_mass/m \
orif_se.k1/area \
battery0.batt/vnom \
el_magnt.sole_mag/lmax, del 0.1, rf sens
dc (dcip dc_nom
tr (pf tr_sens, sigl p_brake, te 1,tstep\
1u,tn 10,terror 0.1u, mon 100
```

c. Measure the rise time of each Transient analysis, on the signal `p_brake`, over the range 680 milli-seconds to 1 second, and using the `tr_sens` plot file as input.

```
measure rise (cn p_brake, xr 680m 1, pfin\
tr_sens
end
close
```

d. Generate a Sensitivity report.

```
sens_r
```

This executes a command script that performs 13 transient analyses, each time automatically offsetting one parameter value. The simulation requires about 5 minutes to run. It also performs a risetime measurement after each run, and then compiles a summary report on the effect of each of the 12 parameters on pressure recovery time.

3. When the simulations are finished, select **Results** from the SaberGuide menu and click on **Sensitivity Report**. When the Sensitivity Report form comes up, click **OK** to use the default report format settings. The sensitivity report appears in the Report Tool.

A sensitivity analysis on the pressure recovery rate identifies the key system parameters and components which determine the responsiveness of the brake system. An automated method sequentially changes each of a specified list of component values in the system, and analyzes the performance effects to determine which are the most important. This capability is vital if a judicious component specification is to be made by the designers.

The sensitivity value is the ratio of percentage change in performance measure per percentage change in parameter value. For example, for the simulation with all components set to their nominal value, a measure of rise time was 43.5 milli-seconds. Then the value of the motor winding resistance is increased by 10%, from 190m to 209 Ohms. Repeating the simulation and measuring the new rise time shows that value has increased to 47.025 milli-seconds, or 8.36%. The ratio, 8.36%/10% = 0.836 is the sensitivity of rise time with respect to motor resistance. If the rise time had decreased when the resistance increased, the sensitivity value would be a negative number.

Determining Component Stress Levels

Load the stress analysis batch command file `run_stress.scs` from the Saber transcript window:

To load the batch file follow these steps.

1. Use the **File > Saber Command File...** items to open the Load Command File dialog box.
2. Choose the batch file called `run_stress.scs` and press the **Open** button.

   The `run_stress.scs` command file executes the following Saber commands.

   a. Evaluate the DC Operating Point.

   ```
   dc (dcep dc_nom
   a mu = 0.4
   a position.rotor =\
   tran=(sin=(va=0.1m,f=15,vo=0,td=0))
   ```

   The above command starts the rotor wobble.

   ```
   dc (dcip dc_nom
   ```

b. Determine the Time-Domain (transient) response.

```
te 1
ts 1u
tn 10
ter 10u
mon 100
```

c. Limit the Signal List to the data you are specifically interested in.

```
sigl pum_mot.pump/torq_mks \
line_f.bhose/ft_elem.1/pres_mks \
line_f.bhose/ft_elem.2/pres_mks \
line_f.bhose/ft_elem.3/pres_mks \
collector \
i(el_magnt.sole_mag) \
i(short.imot) \
mid_line, p_brake, p_reg, psens, rotor, shoe
tr (pf tr_stress, df tr_stress
```

d. Perform Stress analyses on the listed parts.

```
stress (df tr_stress, smeasurelist
        q2n3055.drv/*
line_f.bhose/ft_elem.*/* \
pum_mot.pump/* zd.vclamp/*
stress_r (undef No, stressmin 1)
```

e. Open the plot file.

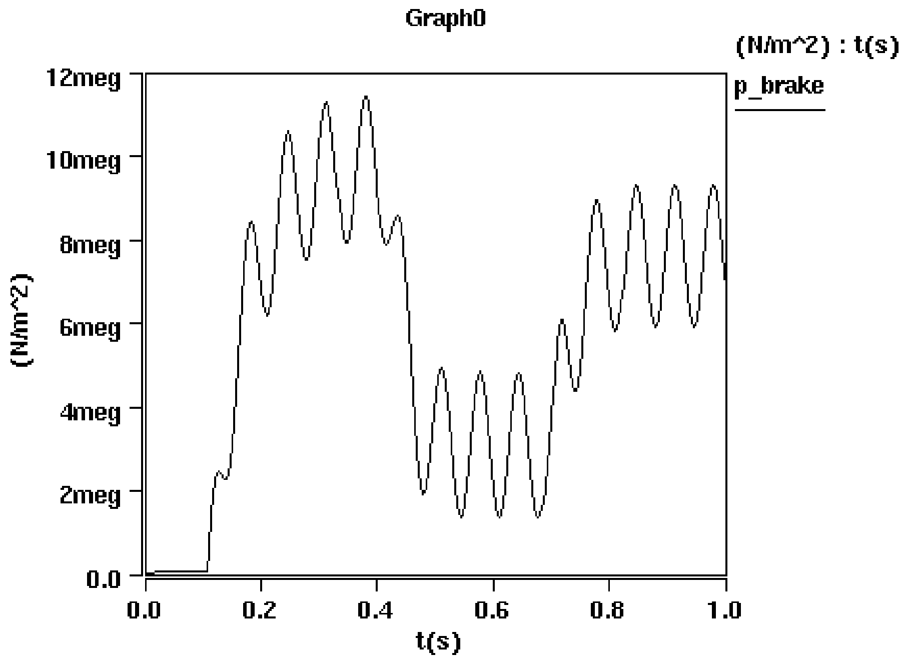
```
pl tr_stress
a mu = 14.3m
a position.rotor =
tran=(sin=(va=0.1m,f=15,vo=0,td=100))
```

The above command resets rotor wobble to its original values.

This executes a command script that performs one transient analysis, after altering the fluid viscosity (to simulate cold temperature conditions), as well as turning on the rotor wobble effect. The simulation runs in about 15 seconds. It then compiles a summary stress report.

The `tr_stress` plot file is opened.

3. When the simulation is finished, select **Results > Stress Report** from SaberGuide. When the Stress Report form comes up, click **OK** to use the default report format settings. The stress report appears in the Report Tool.
4. If there are signals displayed in the Graph window, clear the window.
5. Plot the `p_brake` signal from the `tr_stress` plot file. This signal shows the pressure ripple effects due to the rotor wobble.



A stress analysis points out that several components in the system are operating near their acceptable use limits, in particular during extreme environmental conditions. Potential reliability problems, and related warranty costs, can be avoided for parts that are overstressed during some part of their transient operating cycle. This analysis can reduce production cost, by helping designers avoid component over-rating.

For this analysis, the brake system is operated when there is a significant rotor wobble (transverse motion during wheel rotation) which causes a sinusoidal force on the brake cylinder. The wobble frequency is 15 Hz, and the amplitude is 0.1mm. The simulation conditions also assume cold temperature, as the brake fluid absolute viscosity is set at 0.4 N*sec/(m^2), where the nominal value is 0.014 N*sec/(m^2). This increased viscosity increases the flow resistance of the line, and causes the pressure ripple seen at the wheel cylinder to increase. This in turn can lead to excessive pressure on the brake hose.

The results of the simulation of these conditions shows that the pressure peaks exceed the normal operating maximum, 11.4 MPa. versus less than 10 MPa. This pressure transient peak may be a reliability problem, if there is inadequate design margin in the rated pressure of the brake hose. The stress report shows that the brake hose pressure is near its rated value of 12 MPa. (about 95%). This condition occurs when the rotor wobble combines with peak commanded pressure conditions. Note also that the internal pressure along the hose varies somewhat, as the model includes several lumped segments to approximate the distributed hose length. The maximum pressure is seen closest to the wheel cylinder, as expected.

The drive transistor `q2n3055` is the second most vulnerable part, reaching over 70% of it rated power during operation. Both of these reliability issues are highlighted by the simulator, helping the design team identify critical or "at risk" components so they can make necessary modifications. Again, sensible trade-offs across technologies can be evaluated.

Performing Statistical Analysis

Load the Monte Carlo analysis batch command file `run_mc.scs` from the Saber transcript window:

To load the batch file follow these steps.

1. Use the **File > Saber Command File...** items to open the Load Command File dialog box.
2. Choose the batch file called `run_mc.scs` and click the **Open** button.

   The `run_mc.scs` command file executes the following Saber commands.

   a. Evaluate the DC Operating Point.

   ```
   dc (dcep dc_nom
   ```

   b. Set up the test pressure reference to create a steady state for the Monte Carlo analysis.

   ```
   alter cntl_mod.control/v.vref =
   tran=(pwl=[0,3.5,400m,3.5,401m,1.5,
               700m,1.5])
   ```

c. Perform Monte Carlo analysis on the listed parts.

```
MC (runs 100 ,seed Constant,parl \
pr_sens.press_sens/lag.sen_filt/w \
pr_sens.press_sens/lag.sen_filt/k \
line_r.rigid/len \
line_r.rigid/din \
r.r2/rnom \
damper_t.arm_d/d \
v_2way.v_cntl/amax \
pum_mot.pump/displ \
dc_pm.mot/ra \
orif_se.k1/area \
r.r4/rnom \
r.r3/rnom \
r.r1/rnom \
battery0.batt/vnom \
el_magnt.sole_mag/r \
el_magnt.sole_mag/lmax, parf mcparf
dc (dcip dc_nom
tr (te 1, ts 1u, mon 100, ter 10u, tn 20,
    pf tr_mc
END
meas at end (cn p_brake, pfin tr_mc,
             pfout p_end
pfhist (pfin p_end, pfout p_hist, cn\
At_END(p_brake)
```

The above command generates histogram plot files.

```
alter cntl_mod.control/v.vref = tran =
(pwl=[0,3.5,400m,3.5,401m,1,700m,1,701m,
      2.5,1,2.5])
```

The above command resets the test pressure reference to its original values.

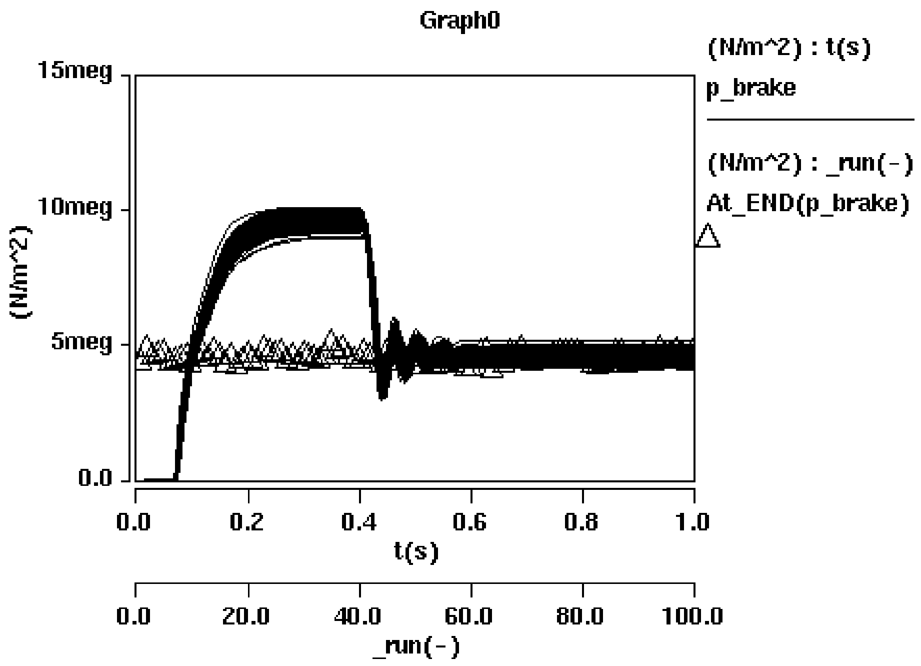
The command script performs 100 transient simulations, assigning parameter values randomly to all the parts that have statistical distributions. The simulation runs in about 15 minutes.

d. Open the plot files.

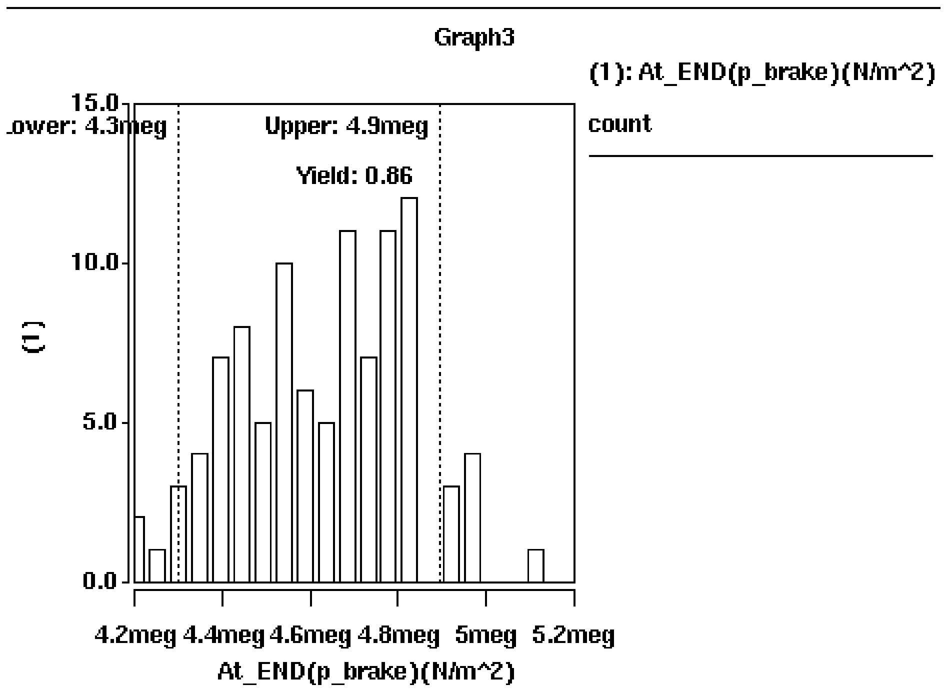
```
pl tr_mc mcparf p_end p_hist
```

The plot files `tr_mc`, `mcparf`, `p_end` and `p_hist` come up. The file `tr_mc` contains the transient waveforms for each of the 100 runs.

3. Select and plot the `p_brake` signal (`tr_mc` plot file) and observe the final pressure variation across the 100 waveforms. The file `p_end` contains the 100 final pressure values for `p_brake`, (the result of an `At_End` measure). Plot `At_END(p_brake)` to see the final pressure for each run.



4. Then on a new graph, plot the `count` signal from the file `p_hist`, to see the statistical distribution of the final pressures.



Use the Scope Measurement tool to calculate the yield. To use the Measurement tool follow these steps:

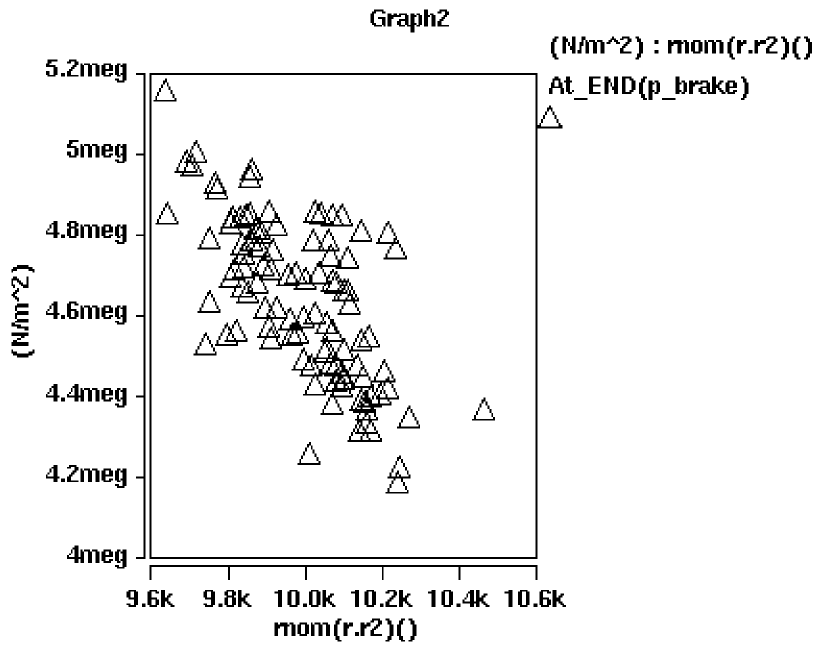
1. Single click on the Measurement tool icon.
2. In the Measurement dropdown list box choose **Statistics > Yield**.
3. In the Signal field, choose `count`.
4. Be sure that the Yield measurement is selected.
5. In the Specifications Limits fields set the Upper field to `4.9meg`, and the Lower field to `4.3meg`.
6. Click **Apply**.

The specification limits and the yield (86%) are shown on the active graph, with the histogram.

Finally, the file `mcparf` contains the individual parameter assignments for each of the 100 runs. Use this data to create scatter plots in the Scope calculator. This shows the correlation between the final pressure value and the assigned value of each parameter. To generate these plots follow these steps.

1. Single click on the Waveform Calculator icon.
2. Select `At_END` in the `p_end` plot file and place it in the calculator's entry window (Calculator window: **Edit > Paste Selected)**.

3. Select any one of the parameters from the `mcparf` plot file, (e.g. `rnom(r.r2)`) and enter it in the calculator's entry window.
4. Select **Wave > f(x)** calculator menu items, and plot the result.



A strong correlation between `r.r2` and final pressure is indicated: higher values of `r.r2` produce lower values of static pressure. Other parameters might show minimal influence on this pressure. Knowing which is which is the real value of this analysis.

The statistical methods shown here allow designers to not only estimate manufacturing yield for the particular design topology and tolerance specifications, but also to determine which parts have the most influence on the key performance characteristics. This helps identify opportunities for manufacturing cost savings. Perhaps tightening the tolerance on just one or two parts would increase the production yield to near 100%. On the other hand, perhaps some part's tolerances are being held excessively tight, and are therefore more costly to acquire. Both these findings may reduce total cost of production.

Analyzing the Audio System Example

The Audio Test System is a comprehensive example, illustrating the diverse analysis capabilities of the Saber Simulator, the richness of the supplied model Libraries, and the flexibility provide by the MAST Hardware Description Language. It illustrates the value of simulation in the design process, helping integrate diverse modules and sub-systems and improve overall system performance.

This section describes the following topics:

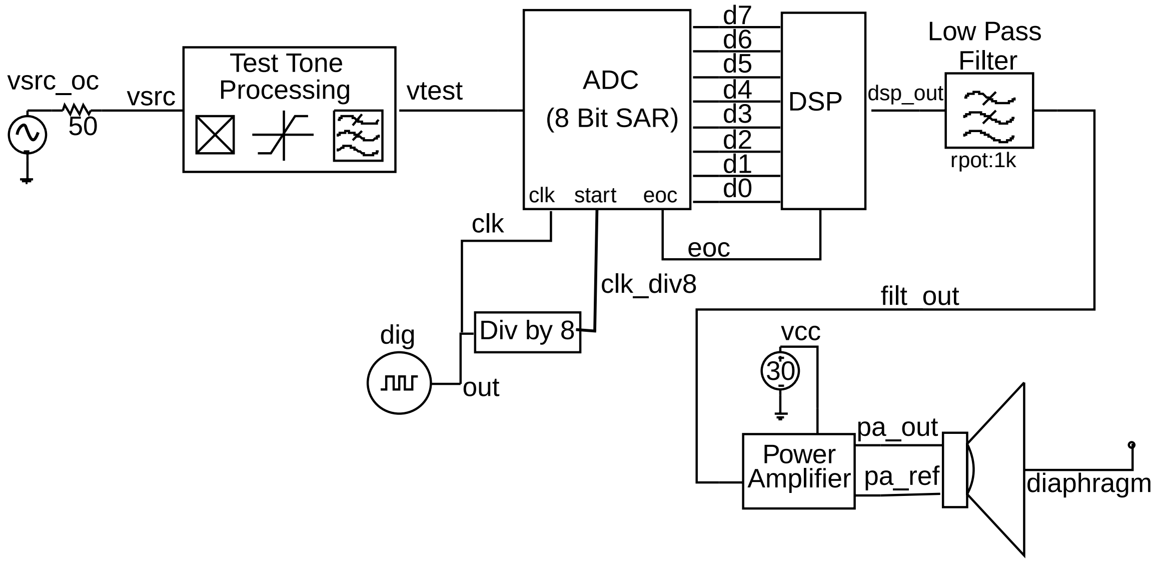
- Copying Audio Test System design files for your design capture tool
- Overview of the various blocks in the Audio System block diagram
- Analyzing the top-level Audio System design

The audio system includes mixed-signal (analog and digital) IC's for clock generation and A-to-D conversion, a DSP algorithm section for response leveling and sound effects, board level analog electronics for filtering and amplification, as well as a mixed-technology loudspeaker with mechanical non-linearities and resonance characteristics. The dynamic interactions of the modules makes design specification difficult to do in isolation. Saber, which simulates the audio system as a whole, supports total system "tuning" or performance improvement, as well accommodating cross discipline trade-off analysis for manufacturability and cost reduction.

The example also illustrates effective simulation strategies. These include selecting models with the right level of abstraction for the job, as well as bottom-up characterization of behavioral models to preserve accuracy while increasing simulation speed. It also shows the creation of analog and digital models, using the MAST Hardware Description Language. Both the successive approximation register (SAR) digital model (`sar_bhv.sin`) in the A-to-D converter, and the voice coil analog model (`voice_coil.sin`) in the Loudspeaker, were written specifically for this example. Simulation progress often depends on the user's ability to supply key models of elements specific to his design.

Selecting Models for the Audio System

The following figure shows the overall audio schematic.



The following list describes the basic functionality of each block in the audio test system. In addition to the completed Audio design, each block in the design has a separate directory containing the necessary design files (schematic and saber command batch file). For example, the "DSP" directory contains a schematic (ex_dsp), which is an example to test the DSP algorithm stand-alone. There is a batch command file (`ex_dsp.scs`) that contains a description of the test sequence and the expected results.

For details on how the block works, models used to implement the block, and the analyses run to test the block, click on the block name:

- The Test Tone Processing (CSP) block generates complex audio test-tones required to test the amplifier's performance.
- The Analog-to- Digital Converter (ADC) block digitizes analog signals so they can be processed by a Digital Signal Processor (DSP).
- The Relaxation Oscillator (OSC) block supplies a clock to the ADC and also sets the system sampling rate. In the "OSC" directory, there are several versions of the oscillator schematic, such as "ex_osc_mos" and "ex_osc_dig". These exercise oscillator models with different levels of abstraction, such as a pure analog MOSfet level model and a pure DIGital model, respectively.
- The Divide by 8 Block (N8DIV) divides the clock signal generated by the relaxation oscillator by a factor of eight, setting the sampling rate of the analog-to-digital converter.

- The Digital Signal Processor (DSP) block compensates for speaker resonance, such as mechanical resonance, and allows for the introduction of special sound effects.
- The RLC Filter block primarily filters sampling noise introduced by the ADC/DSP prior to reaching the power amplifier.
- The Power Amplifier block amplifies the filtered DSP output signals and applies them to the speaker through an impedance-matching transformer.
- The Speaker block converts the power amplifier signal's electrical energy into mechanical energy, moving the diaphragm to create sound.

Copying the Audio Test System Example

The Audio Test system example is available in the Sketch, Cadence, Mentor Graphics, and Viewlogic design environments. The following four sections describe how to copy the Audio Test system design files (schematics, netlists, and Saber batch files) for your specific design capture tool.

- "Copying Audio System Files for SaberSketch" on page 2-3
- "Copying Audio System files for Cadence (Artist)" on page 2-4
- "Copying Audio System files for Mentor Graphics (Design Architect)" on page 2-5
- "Copying Audio System files for ViewLogic (ViewDraw) on UNIX" on page 2-5
- "Copying Audio System files for ViewLogic (ViewDraw) on Windows NT" on page 2-6

Copying Audio System Files for SaberSketch

To copy the Audio System files from the *install_home* software tree to your local directory, follow these steps:

1. Make sure that all the necessary environment variables are set for use with SaberDesigner. This is typically set up by a System Administrator.
2. Create (if necessary) and change to the directory where you would like the files to be copied.

3. Copy the `Audio` directory from the following location to your current directory:

   UNIX source - *install_home*`/example/SaberSketch/Audio`
   Windows NT - *install_home*`\example\SaberSketch\Audio`

   You should now have a directory called `Audio`.

4. (Windows NT only) You must change the file permissions of your local copy of the designs so that they are no longer read-only as follows:

   a. Invoke Windows NT Explorer.

   b. Navigate to your local copy of the design. In the case of a design example that has more than one directory, you will need to change the permissions of all the files in each of them as described in steps c through f.

   c. In each directory, select all the files (**Edit > Select All**).

   d. Open the Properties dialog box (**File > Properties**) and select the General tab.

   e. Un-check the Read-only box.

   f. Click **OK**.

You can view the designs in the `Audio` directory with SaberSketch by following the steps in the topic titled "Invoking SaberSketch on the Audio System" on page 2-7.

Copying Audio System files for Cadence (Artist)

To copy the Audio System files from the *install_home* software tree to your local directory, follow these steps:

1. Make sure that all the necessary environment variables are set for use with SaberDesigner. This is typically set up by a System Administrator.

2. Create (if necessary) and change to the directory where you would like the files to be copied.

3. Copy the `Audio` directory from the following location to your current directory:

   install_home`/example/Cadence/Audio`

   You should now have a directory called `Audio`.

4. You can view the schematic of one of the blocks in the Audio Test System design using the steps described in the topic titled "Invoking Cadence (Artist) on the Audio System" on page 2-8.

Copying Audio System files for Mentor Graphics (Design Architect)

To copy the Audio System files from the *install_home* software tree to your local directory, follow these steps:

1. Make sure that all the necessary environment variables are set for use with SaberDesigner. This is typically set up by a System Administrator.
2. Create (if necessary) and change to the directory where you would like the files to be copied.
3. Using Design Manager (`dmgr`), copy the directory:

   install_home`/example/MentorGraphics/Audio`

   Note that this directory, in order for dve to properly recognize it, must be named `Audio`.

You can view the schematic of one of the blocks in the Audio Test System design using the steps described in the topic titled "Invoking Design Architect on the Audio System" on page 2-9.

Copying Audio System files for ViewLogic (ViewDraw) on UNIX

To copy the Audio System files from the *install_home* software tree to your local directory, follow these steps:

1. Make sure that all the necessary environment variables are set for use with SaberDesigner. This is typically set up by a System Administrator.
2. Create (if necessary) and navigate to the directory where you would like the files to be copied.
3. Copy the `Audio` directory from the following location to your current directory:

   install_home`/example/ViewLogic/Audio`

   You should now have a directory called `Audio`.
4. You can view the schematic of one of the blocks in the Audio Test System design using the steps described in the topic titled "Invoking Powerview on the Audio System (UNIX)" on page 2-9.

Copying Audio System files for ViewLogic (ViewDraw) on Windows NT

To copy the Audio System files from the Saber installation tree to your local directory, follow these steps:

1. Create (if necessary) and navigate to the directory where you want the Audio example files to be located.
2. Copy the following directory to your current directory:

   install_home`\example\Viewlogic\Audio`

   where *install_home* is the path to the Saber installation. You should now have a local directory called `Audio`.
3. Change your directory to `Audio/System`.
4. Copy the following file to your `System` directory:

   install_home`/framework/standard/viewdraw.ini`

   This file is template initialization file for ViewDraw that you will use as a convenient way to incorporate certain library search paths in later steps.
5. (Windows NT only) You must change the file permissions of your local copy of the designs so that they are no longer read-only as follows:
   a. Invoke Windows NT Explorer.
   b. Navigate to your local copy of the design. In the case of a design example that has more than one directory, you will need to change the permissions of all the files in each of them as described in steps c through f.
   c. In each directory, select all the files (**Edit > Select All**).
   d. Open the Properties dialog box (**File > Properties**) and select the General tab.
   e. Un-check the Read-only box.
   f. Click **OK**.

You can view the schematic of one of the blocks in the Audio Test System design using the steps described in the topic titled "Invoking Powerview on the Audio System (Windows NT)" on page 2-10.

Invoking Your Schematic Capture Tool

After you have made a local copy of the Audio system, you can view the schematic by using your appropriate schematic capture tool as described in the appropriate topic listed below for your particular environment:

- "Invoking SaberSketch on the Audio System" on page 2-7
- "Invoking Cadence (Artist) on the Audio System" on page 2-8
- "Invoking Design Architect on the Audio System" on page 2-9
- "Invoking Powerview on the Audio System (UNIX)" on page 2-9
- "Invoking Powerview on the Audio System (Windows NT)" on page 2-10

Prior to running any of the design examples, you should run the Getting Started tutorials to familiarize yourself with the SaberDesigner tool set.

Invoking SaberSketch on the Audio System

You can view the designs in your local copy of the `Audio` directory by following these steps:

1. Invoke SaberSketch using one of the methods as follows:

   (UNIX) On a command line, enter *install_home*`/bin/sketch`

   (Windows NT) **Start > Analogy >** *saberdesigner* **> SaberSketch**

   An empty schematic window appears.

2. Use the **File > Open > Design** menu choice to bring up the Open Design dialog box.

   In the Open Design dialog box, navigate to the design in the Audio Test System design (in the `Audio` directory) that you want to view. The top level design is in the `System` directory. Select the schematic file name (`ex_audio.ai_sch`) and click on the **OK** button to open the schematic.

3. Start the netlister by selecting the **Design > Netlist ex_audio** menu item.

You are now ready to analyze the design as described in the topic titled "Analyzing the Design (Audio System)" on page 2-14.

Invoking Cadence (Artist) on the Audio System

You can view the designs in your local copy of the `Audio` directory by following these steps:

1. Navigate into the `Audio` directory.
2. Invoke icms. (`icms`)
3. Add a new library definition:
   a. From the icms-Log window, choose the **Tools > Library Manager** menu item. A Library Manager window appears.
   b. In the Library Manager window, choose the **Edit > Library Path** menu item.
   c. Following the instructions at the bottom of the CdsLibEditor window, set the Library Path to the directory that contains the schematic you want to invoke. For example, to look at the `ex_audio` schematic in the `System` directory:

Library	Path
`ex_audio`	*path*`/analogy_tutorial/Audio/System/ex_audio`

   d. Choose the **File > Save** menu item.
   e. Close the CdsLibEditor window (**File > Exit**).
   f. Close the Library Manager window (**File > Exit**).
4. Open the design as follows:
   a. From the icms banner, open the schematic by choosing the **File > Open** menu item. The Open File dialog box appears.
   b. In order to see the `ex_audio` schematic, select `ex_audio` as the Library Name, `ex_audio` as the Cell Name, and `schematic` as the View Name. Click on the **OK** button.
5. Select the **Saber > Set Working Directory** menu item. In the Project Information dialog box, insert your working directory path into the Project Directory field, and click on **OK**. (For example, to invoke the `ex_audio` top level schematic, insert *path*/`Audio`/`System`.)
6. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.

You are now ready to analyze the design as described in the topic titled "Analyzing the Design (Audio System)" on page 2-14.

Invoking Design Architect on the Audio System

You can view the designs in your local copy of the `Audio` directory by following these steps:

1. Create a `SABER_EXAMPLE` environment variable, whose value is your current working directory, the directory that contains `Audio`.

   ```
   setenv SABER_EXAMPLE your_data_path
   ```

2. Change your working directory to `Audio`.

   ```
   cd $SABER_EXAMPLE/Audio
   ```

3. Create a `SABER_DATA_PATH` environment variable.

   ```
   setenv SABER_DATA_PATH
   $SABER_EXAMPLE/Audio/templates
   ```

4. In order to see the top level schematic, located in the `Audio/System` directory, change to the `System` directory.
5. Start the Design Viewpoint Editor application by typing:

   ```
   dve ex_audio
   ```

6. Setup Saber by selecting the **Setup > Saber** menu item.
7. Select the **File > Save Design Viewpoint > With Same Name > Cleanup Un-used References** menu item.
8. Click on the **OPEN SHEET** icon to open the schematic.
9. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.

You are now ready to analyze the design as described in the topic titled "Analyzing the Design (Audio System)" on page 2-14.

Invoking Powerview on the Audio System (UNIX)

You can view the designs in your local copy of the `Audio` directory by following these steps:

1. Change your directory to `Audio`.

   In order to simulate the top level circuit, you will perform the following steps:

2. Invoke Powerview with the following command:

   ```
   powerview
   ```

3. Create Audio Project as follows:

   a. In the Powerview Cockpit window, choose the **Project > Create** menu item. The Create Project dialog box appears.

   b. In the Create Project dialog box, click the **Browse** button to display the Select File dialog box.

   c. In the Select File dialog box, select (double-click) `Audio` and then the directory (`System`) containing the block in the Audio Test System that you want to view.

   d. In the Select File dialog box, click **OK**.

   e. In the Create Project dialog box, click **OK**.

4. To open the `ex_audio` design from within the Audio Project:

   a. In the Powerview Cockpit window, verify that Current Project is set to *path*`/Audio/System`.

   b. Double-click on the **Viewdraw** icon. The File Open dialog box is displayed.

   c. In the File Open dialog box, double-click on the schematic name (`ex_audio.1`). The schematic appears in ViewDraw.

5. From the main Frameway session window, choose the **Saber > Netlist > Start Netlister** menu item to start the netlister.

You are now ready to analyze the design as described in the topic titled "Analyzing the Design (Audio System)" on page 2-14.

Invoking Powerview on the Audio System (Windows NT)

You can view the designs in your local copy of the `Audio` directory by following these steps:

1. Confirm that the Viewlogic and Saber installations have been completed and are accessible on your workstation. Note the directory locations of the Saber and Viewlogic installations.

2. Open the local copy of the `viewdraw.ini` file with a text editor. You must edit the library search paths contained in this file to match the search paths required for your local Saber installation. This a plain-text file, so be sure to save it as a text file after editing.

Show below is an excerpt from a `viewdraw.ini` file, showing a few of the library search-path entries.

Modify to point to your local Workview Office installation

```
dir [p] C:\WVOFFICE\wv_libraries (WVLIBRARY)
dir [rm] C:\WVOFFICE\wv_libraries\anlgdev (analog)
dir [rm] C:\Analogy\saber5.1\framework\viewlogic\symbols\comp (sbr_comp)
.
.
.
```

Modify to point to your local Saber installation

Use the search and replace capabilities of your text editor to modify the installation-dependent portions of the search-paths so they are correct for your installations.

3. From the **Start** menu (or WorkView Office shortcut if present) invoke the Viewlogic Workview Office (if it is not already active).
4. Start the Project Manager by clicking on the Project Manager icon in the Workview Office task bar (or use the **Start** menu or a shortcut, as appropriate).
5. Set up a project file in your `System` directory as follows:
   a. In the Project Manager dialog box, choose the **File > New** menu item. This activates the Project Manager wizard.

      NOTE
      If you have previously created a project file you may see a Project Manager Wizard message box asking whether you want to copy the library search paths that were used in that project. If this happens, click on the **Don't Copy** *button.*

   b. In the first dialog box of the Project Manager wizard, enter `System` in the Project Name field.
   c. Edit the Project Directory field, if necessary, to include the correct path to your `System` directory (for example, `C:\vwlogic_ex\Audio\System`) then click on the **Next** button to display the next dialog box.
   d. Confirm that the directory location for the project file (`System.vpj`) is the same as given in the preceding step, then click the **Next** button to display the next dialog box.

e. Again click on the **Next** button in the dialog box. (No FPGA libraries need to be added.)

f. In the "ViewDraw libraries to use" dialog box, add the library search paths as follows:

    a. Click on the **Import** button. The Open dialog box appears.

    b. Click on the `viewdraw.ini` file name in the Open dialog box.

    c. Click on the **Open** button. This adds the library search paths automatically from the `viewdraw.ini` file.

d. Click on the **Finish** button.

e. Confirm that the New Project Information dialog box contains the correct project information, then click on the **OK** button. This completes the `System` project file setup.

f. Save the project file by selecting the **File > Save** menu item.

6. Navigate to each of the following directories under the `Audio` directory and set up a project file: ADC, CSP, DSP, `Lspkr`, `N8div`, `Osc`, `Pwramp`, and RLC. This will allow you to open these design modules in ViewDraw during later analyses.

    a. In the Project Manager dialog box, activate the New Project wizard by selecting the **File > New** menu item.

    b. A Project Manager Wizard message box asks whether you want use the library search paths from the previous project. Click on the **Copy** button to accept those paths.

    c. Edit the Project Directory field to include the correct path to the desired project directory (for example, `C:\vwlogic_ex\Audio\ADC`).

    d. Enter an appropriate name (such as ADC) in the Project Name entry box, then click on the **Next** button.

    e. Confirm that the directory location for the project file (*project_name*`.vpj`) is the same as given in Step c, then click on the **Next** button.

    f. Click on the **Next** button in the next dialog box. (No additional FPGA libraries need to be added.)

    g. Click on the **Finish** button in the next dialog box. (No ViewDraw libraries need to be added.)

    h. Confirm that the New Project Information dialog box contains the correct project information, then click on the **OK** button. This completes the project file setup for the current directory.

i. Save the project file by selecting the **File > Save** menu item.

j. Repeat Step a through Step i for each of the listed project directories.

7. Open the top-level schematic of the Audio design:

   a. In the Project Manager, navigate to the `System` directory and open the project file (`System.vpj`).

   b. Click on the **ViewDraw** icon in the Workview Office task bar (or use the Start menu or a shortcut). The Viewdraw session window appears.

   c. Select the **File > Open** menu item. This activates the File Open dialog box.

   d. In the File Open dialog box, double-click on the schematic named `ex_audio`. The schematic appears in ViewDraw.

8. Pull down the ViewDraw Tools menu and check to see if a menu item containing the word **Saber** is present. This menu item will have several Saber-related entries below it, beginning with **Start SaberGuide**. If this menu item is not present, add it to the Tools menu as follows:

   - Select the **Tools > Customize** menu item.
   - Click on the **User Menu** selector button.
   - Type a name such as **Saber** in the Menu Text entry box. (The name **Saber** will be used for this menu item in all subsequent instructions.)
   - Click on the **Browse** button next to the Command entry box and navigate to the `bin` directory under your Saber installation directory (typically `C:\Analogy\saberdesigner5.1\bin`). Under the `bin` directory, select the file named `menu.exe`, then Click onthe **Open** button.
   - Click on the **Add** button, then the **OK** button. The **Saber** menu item should now appear in the Tools menu.

9. From the ViewDraw session window, choose the **Tools > Saber** menu item to activate the Saber Menu window. This window contains a Saber menu, which you should use throughout the remainder of this tutorial when told to select an item from the Saber menu.

10. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.

You are now ready to analyze the design as described in the topic titled "Analyzing the Design (Audio System)" on page 2-14.

Analyzing the Design (Audio System)

The test configuration presented in this section makes it easy for the designer to excite the system with various test tones, pulses and waveforms, as well as to experimentally adjust any of the audio system modules. The effect of design changes can be readily assessed, for any of the following:

DSP algorithm	A-to-D sample rates or quantization
Analog output filter	power amplifier electronics
Electro-mechanical parameters of the loudspeaker	

Because the modules interact, as in an actual audio system, the effects of loading (static and dynamic) are included in any performance results. In addition, this interaction is important for verifying proper signal levels, to avoid clipping and the resultant distortion.

The integrated, hierarchical audio test system is contained in the `System` directory (ex_audio). All audio system modules and a test tone generator model are assembled to analyze end-to-end system performance.

After you make a local copy of the design and netlist it, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the `Audio` directory:

Step 1. Invoke SaberDesigner if it is not already active.

From the Frameway schematic-capture applications, select the **Saber > Start SaberGuide** menu item.

From SaberSketch, click the icon.

From the UNIX command line, type `saber`.

From Windows NT, select **Start Programs > SaberDesigner > Saber**.

Step 2. Open the Saber netlist

SaberSketch users skip to the next step.

To open the Saber netlist follow these steps:

1. Display the Open Design dialog box (**File > Open > Design...**)
2. Browse to the directory containing the Audio example files (*path*`/Audio/System`)

3. Select the `ex_audio.sin` file and click the **Open** button.

   This action should add `ex_audio.sin` to the Design Name field in the Open Design dialog box.

4. Load the design into Saber by clicking the **OK** button in the Open Design dialog box.

The remaining steps exercise the entire Audio Test System (ex_audio). Two time domain analyses compare the pulse tone response of the system with and without the response equalizing filter in the DSP.

Step 3. Evaluate the DC Operating Point

1. Display the DC Operating Point form (**Analyses > Operating Point > DC Operating Point...**)
2. Execute the DC analysis by clicking the **OK** button.

This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report...**)

Step 4. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**)
2. Edit the following fields in the Transient Analysis form

   End Time: 200m

   Time Step: 1u

   Monitor Progress: 1000

   Plot File (Input/Output tab): tr_filt

   Data File (Input/Output tab): _

   Max Truncation Error (Calibration tab): 100u

3. Perform the analysis by clicking the **OK** button

   This command determines the time domain response of the system with the DSP equalizing filter and delay element active during the first 200 milliseconds. The Saber simulator saves resulting waveforms for each signal on the root of the design in a Plot File called tr_filt. It also displays the information to the SaberGuide Transcript Window on every 1000th calculated data point of the simulation. Because there are numerous elements in this design, a data file is not created to save disk space.

The truncation error was decreased in order to increase simulation accuracy.

Step 5. Disable the DSP filter

1. Display the List/Alter Design form (**Edit > List/Alter ...**)
2. Double-click on the `dsp.dsp1` instance in the Hierarchical Instance List
3. Select the `zlti.zlti1` instance under the `dsp.dsp1` part.
4. Display the Alter Components form by clicking the **Edit...** button in the List/Alter Design form.
5. Change the values to those shown below to disable the DSP filter:

   ```
   a=0.039
   den=[1,0,0]
   num=[1,0,0]
   ```

6. Make the changes to the in-memory design by clicking the **OK** button. This step does not affect the schematic or the `ex_audio.sin` file.
7. Close the List/Alter Design form.

In the remaining steps, you will run a second transient analysis and save new results to the `tr_nofilt` Plot File.

Step 6. Find the Operating Point with the DSP filter disabled

1. Display the DC Operating Point form (**Analyses > Operating Point > DC Operating Point...**)
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report**)

Step 7. Determine the time-domain (transient) response with the DSP filter disabled.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**)
2. Edit the following fields in the Transient Analysis form. This step uses the same settings as the previous transient analysis except the

Plot File name is changed so that you can compare the effects of the DSP filter.

End Time: 200m

Time Step: 1u

Monitor Progress: 1000

Plot File (Input/Output tab): **tr_nofilt**

Data File (Input/Output tab): _

Max Truncation Error (Calibration tab): 100u

3. Perform the analysis by clicking the **OK** button

   This command runs the same transient analysis as in Step 4 except the results are saved to a different plot file (`tr_nofilt`).

Step 8. Plot the transient responses

- Frameways:
  1. Display the Signal Manager (**Tools > Signal Manager**)
  2. In the Signal Manager, click the **Open Plotfiles** button.
  3. In the Open Plot Files dialog box, select plotfile name `ex_audio.tr_filt.ai_pl` and click the **Open** button.
  4. Repeat steps 2 and 3 for the `ex_audio.tr_nofilt.ai_pl` file.
- SaberSketch:
  1. Display the View Plot Files dialog box (**Results > View Plotfiles in Scope...**). The View Plot Files dialog box appears.
  2. In the View Plot Files dialog box, click the **Browse...** button.
  3. In the Select List, select both `ex_audio.tr_filt` and `ex_audio.tr_nofilt` and click **OK**.
  4. In the View Plot Files dialog box, click the **OK** button. The Scope Waveform Analyzer starts with both plot files opened in the Signal Manager.

Step 9. Compare the signals

You can now view and compare the signals in the audio design in the Scope Waveform Analyzer. In both plot files, the "echo" sound effect is active (also part of the DSP). A second tone burst (lower amplitude, 100 msec delayed from the first) is observed in the output. Note that the output signal is named `diaphragm`. It represents the instantaneous position of the loudspeaker diaphragm or cone, and is measured in meters (typically several milli-meters peak-to-peak displacement during operation).

Compare the `diaphragm` signals from the two files, and note that significantly higher level of undesired signal at the loudspeaker mechanical resonance frequency. The test tone (stimulus) of the system is `vtest`.

chapter 3

Designing the Test Tone Generator

This circuit generates a test tone for the audio system design. Because this circuit will only be used for testing the audio system in simulation, you want to use models that can manipulate signals at a high level of abstraction. Designing this circuit with discrete parts could also be very time consuming and inflexible. In order to quickly design a test tone that can be easily modified, Continuous (Signal Flow) Block diagram models from the Control Systems Library are the perfect solution.

This circuit demonstrates the following capabilities:

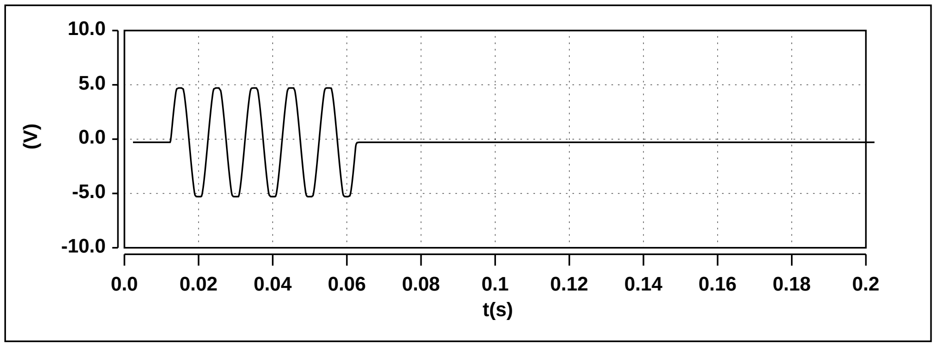
- Using Saber models to generate complex stimulus
- Using Control and Electrical system models in the same circuit

This example uses the following process to develop and test the schematic block:

1. "Specifying Design Parameters" on page 3-2
2. "Selecting Models for the CSP Circuit" on page 3-2
3. "Analyzing the CSP circuit" on page 3-3

Specifying Design Parameters

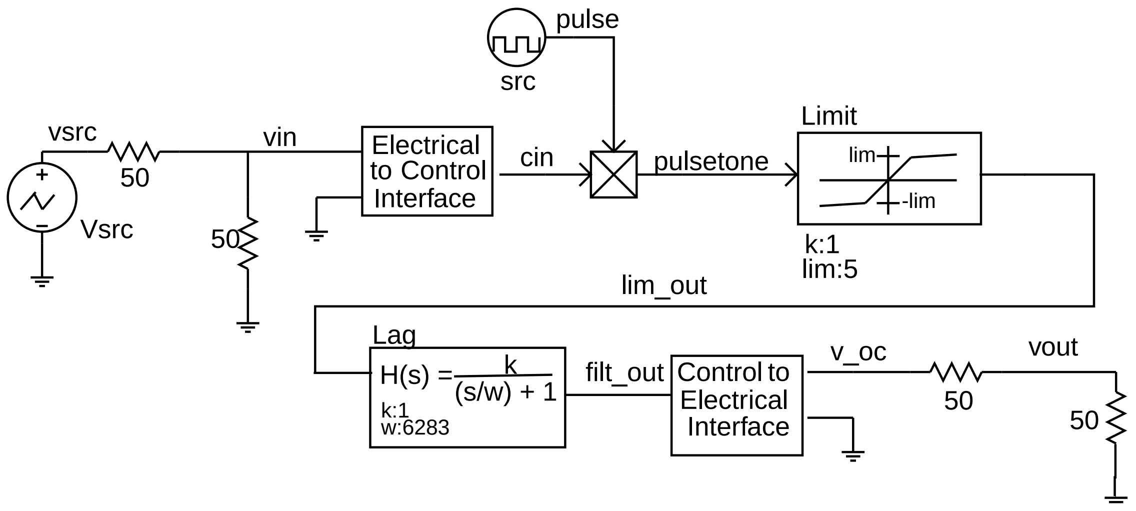
In order to test the audio system, you want to design a circuit that produces a 50mS duration of a 100Hz 12Vpeak-to-peak sine wave as illustrated in the following graph.



You also want to design the test generation circuit with maximum flexibility which would enable you to easily introduce various test patterns into the system.

Selecting Models for the CSP Circuit

The following circuit was designed to meet the parameters specified in the previous section. After you examine the models used in this design, you can simulate it, as described in the next topic.



- All resistors in this design were specified at 50 ohms by changing the value of the `rnom` property to 50.
- The `vsrc` instance uses the **v** template to produce a 6 volt 100Hz sine wave. You specify this waveform by modifying the `saber_model`

property to `trans=(sin=(va=6, f=100, vo=0))`. Because you will only analyze this circuit in the time-domain, you only need to define the transient waveform.

- The `e2c1` instance uses the **elec2var** template to convert the electrical signal (voltage) to a control signal (unitless). This conversion is necessary to exchange signals between analog and control system parts in the design. The `v2e1` instance is used for a similar purpose.
- The `src1` instance uses the **src** template from the Control system library to produce a 50 millisecond pulse with a 200millisecond period and an amplitude of 1 units by editing the `Saber_model` property to `tran=(pulse=(v1=0, v2=1,tr=1u,tf=1u,td=10m,pw=10m,per=200m))`.
- The `mult1` instance uses the **mult** template from the Control System library to multiply the pulse waveform (generated by `src1`) and the sine waveform (generated by `vsrc`). This part produces a 50 millisecond window of a 100 Hz sinewave every 200 milliseconds. Because the `mult1` instance can only multiple 2 control signals, the `e2c1` symbol was added to convert the sinewave from an electrical signal in volts to an unitless control signal.
- The `limit1` instance uses the **limit** template to limit the output of the multiplier to a 5 unit maximum (because pulsetone is a control signal, it has no units).
- `lag` filters out some of the harmonics associated with the pulse signal produced by vpulse. Break frequency is at 1KHz. This filter also slightly smooths the output signal.

Many other parameter adjustments can be made to achieve other signal conditioning effects. Alternate signal conditioning blocks from the Control System Library can be substituted as well.

Analyzing the CSP circuit

After you make a local copy of the design and set up your environment, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command:

install_home`/bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design dialog box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*/`Audio/CSP`).
3. Select the `ex_csp` file and open it.

The remaining steps exercise the Continuous Signal Processing (ex_csp) block of the Audio Test system. This block generates the test tone for the Audio circuit. It runs two time domain analyses to compare the system response to moving the break frequency of the filter.

Step 3. Evaluate the DC Operating Point

1. Display the Operating Point Analysis form (**Analyses > Operating Point > DC Operating Point...**).
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report**)

Step 4. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form.

   End Time: 200m

   Time Step: 1u

   Monitor Progress: 100
3. Perform the analysis by clicking the **OK** button.

   This command determines the time domain response of the circuit during the first 200 milliseconds and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`. It also displays the information to the SaberGuide transcript window on every 100th data point of the simulation.

The remaining steps determine the affects of reducing the filter bandwidth.

Step 5. Reduce the break frequency of the filter.

To reduce the break frequency of the filter from 1KHz to 100 Hz, you can change the value of the `w` parameter by following these steps:

1. Display the List/Alter Design form (**Edit > List/Alter**).
2. Select the Netlist tab.
3. Display the Alter Components form by selecting the **lag.lag1** part from the instance list and clicking the **OK** button.
4. Edit the `w` parameter value by changing the value of the `w` parameter to 628.3 and click the **OK** button.

This action changes the break frequency of the filter from 1kHz to 100 Hz.

Step 6. Perform a Transient analysis using the new break frequency.

1. Display the Transient Analysis form (**Analyses > Time-domain > Transient**).
2. Edit the following fields in the Transient Analysis form:

   End Time: 200m

   Time Step: 1u

   Monitor Progress: 100

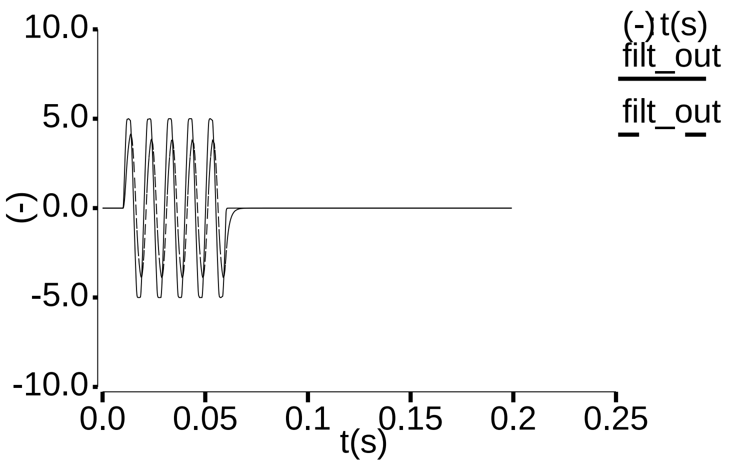
   Plot File: (Input/Output tab): tr_filt
3. Perform the analysis by clicking the **OK** button.

   This command performs a transient analysis using the same arguments, except the results are saved to the `tr_filt` Plot File. This method allows you to compare the circuit behavior with filter break frequencies.

Step 7. Plot the waveforms.

You can compare the `filt_out` signal in both plot files by opening the plot files in the Signal Manager, selecting the `filt_out` signal from both plot file windows and plotting the signals in the graph window.

The resulting waveforms should look similar to the following graph:



Notice that the X-axis values are unitless because `filt_out` is a control signal type.

Step 8. Return the filter bandwidth to original value.

You can accomplish this task by repeating the procedure in Step 5 and set the `w` parameter to 6283.

After you finish simulating this block, you can either continue to analyze this block using other analyses, parameters, and parts or you can examine other blocks in the Audio Test System example.

chapter 4

Designing the Oscillator Block

This circuit generates a 40kHz clock for the Analog-to-Digital Converter block in the audio system design. Using the basic ring oscillator approach, this circuit was implemented at different levels of abstraction.

This circuit demonstrates the following capabilities:

- Using different levels of abstraction to implement a design (digital, mixed analog/digital, and pure analog)
- Performing a parametric sweep using the Vary command
- Using Measurements

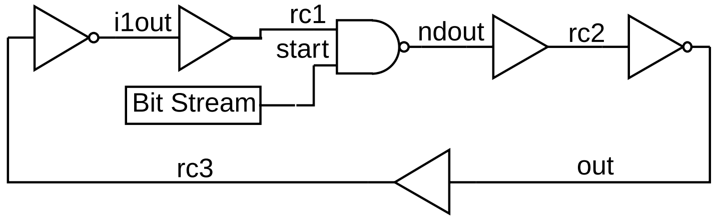
This example uses the following process to develop and test the schematic block:

1. "Selecting Models for the Oscillator Circuit" on page 4-1
2. "Simulating the Mixed-Signal Circuit" on page 4-3

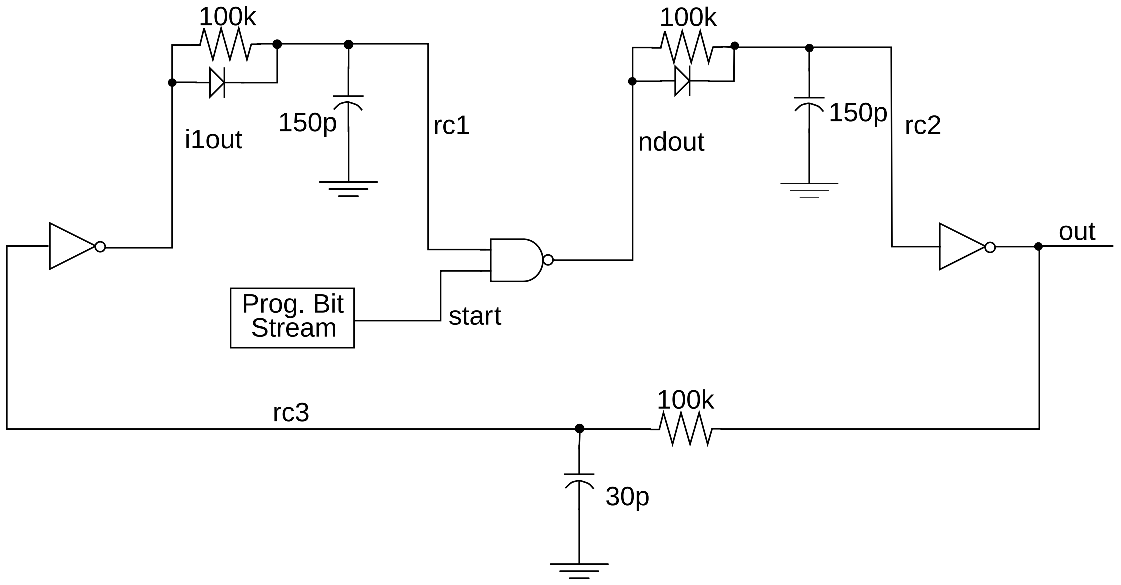
Selecting Models for the Oscillator Circuit

Four versions of this design were implemented to demonstrate the various levels of abstraction supported by the Saber simulator. The circuits are:

- `ex_osc_dig`: A pure digital "ring" oscillator. It includes buffers with long delay times, that have been calibrated from the "relaxation" times of the analog RC[D] circuits. This digital version of the oscillator is very fast, and is used in the audio system for improved simulation speed.



- `ex_osc_mos`: A pure analog hierarchical circuit that includes MOS level for the inverters and NAND gate shown in the following schematic. The proper loop delay characteristics are achieved using resistor-capacitor-diode (RCD) and RC networks on the top-level schematic (analog equivalent of the buffers in the ex_osc_dig schematic).



- `ex_osc_mm`: A hierarchical “mixed-signal” or analog/digital version of the oscillator. The MOS inverters and nand gate are replaced by their digital equivalents, but the analog RC[D] circuits remain. Hypermodels, which have been characterized for threshold level based on the MOS devices, are inserted at the interfaces. This model runs much faster than the pure analog (MOS) circuit as shown in the previous figure.
- `ex_osc_mmflat`: A non-hierarchical (i.e. “flat”) version of the mixed-signal oscillator (ex_osc_mmflat)1.

Be sure to use the `ex_osc.shm` file if you want to netlist this circuit. This file contains the proper Hypermodel logic thresholds, which have an effect on the oscillation frequency. These thresholds have been set to correspond to the MOS (all analog) version of the oscillator, contained in the “ex_osc_mos” model file.

Simulating the Mixed-Signal Circuit

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner.

Before running your simulation you will need to add Hypermodels to the ex_osc_mm design.

Step 1. Open the schematic.

From your schematic capture tool, open the `ex_osc_mm` schematic.

Step 1. Invoke the Saber/Netlister Settings... form.

Choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.

Step 2. Click on the Netlister tab.

Then, click on the Hypermodels tab. The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `ex_osc.shm`.

Step 3. Add the custom Hypermodel.

1. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
2. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
3. Navigate to the `ex_osc.shm` file.
4. Select the `ex_osc.shm` file and click the **Open** button in the File Selection dialog box.
5. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
6. Add the Hypermodel to the Selected listbox by selecting the newly-added `ex_osc` item in the Available listbox and then clicking the **<<>>** button between the listboxes.

Step 4. If necessary, set the Search Path.

Depending upon which Frameway you are using, you may have to set the Search Path to the Hypermodel file. If so, select the Simulation tab followed by the Search Path tab and add the path to `ex_osc.shm`. Click on the **Apply**, **Save**, and **Close** buttons.

Step 5. Generate the netlist and open the Saber simulator.

Choose one of the following steps for your environment:

- From SaberSketch, select the **Design > Simulate ex_osc_mm** menu item to generate a netlist of the schematic and load it into Saber.
- From the Frameway schematic capture tool, select **Saber > Netlist > Start Netlister** to generate a netlist. Once the netlist has been generated, load it into Saber by selecting **Saber > Start SaberGuide**.

You are now ready to perform the simulation.

The remaining steps exercise the Relaxation Oscillator (ex_osc) block of the Audio Test system. This block provides the clock for the Analog-to-Digital converter block of the Audio Test system.

Step 1. Evaluate the DC Operating Point.

1. Display the Operating Point Analysis form (**Analyses > Operating Point > DC Operating Point...**).
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report**).

Step 2. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient**).
2. Edit the following fields in the Transient Analysis form:

   End Time: 100u

   Time Step: 10n

   Monitor Progress: 100

   Plot After Analysis: Yes - Open Only

   Signal List (Input/Output tab): osc_mm.dut

   Plot File (Input/Output tab): tr_normal

3. Perform the analysis by clicking the **OK** button.

This command determines the time domain response of the circuit during the first 100 microseconds and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr_normal`. It also displays the information to the SaberGuide transcript window on every 100th data point of the simulation. After the analysis completes, Scope Waveform Analyzer opens the `tr_normal` Plot File.

Step 3. Measure the frequency on the `out` signal.

1. Plot the `out` signal by selecting the `- analog -osc_mm.dut -out` signal from the `ex_osc_mm.tr_normal` Plot File window and clicking on the **Plot** button.
2. Display the Measurement Tool (**Tools > Measurement**).
3. Measure the Frequency by editing the following fields in the Measurement tool:
   - Measurement: Frequency (Select **Time Domain > Frequency** from the menu in this field
   - Signal: `osc_mm.dut/out` (selected from the menu in this field)
4. Perform the Measurement by clicking the **Apply** button.

   The measured frequency is about 40 kHz.

Step 4. Determine the frequency vs. capacitor “c1” relationship.

You can use the vary command to accomplish this task as described in the following steps:

1. Display the Looping Commands form (**Analyses > Parametric > Vary...**).
2. Define the “vary” parameter by clicking on **vary** button.

   This action display the Parameter Sweep form.
3. Edit the following fields in the Parameter Sweep form:
   - Parameter Name: `osc_mm.dut/buf_rcd.b1/c.c1/c`
   - Variation Type: `Step By`
   - from `150p` to `250p` by `25p`
4. Add the parameter sweep definition by clicking on the **Accept** button.
5. Add a DC analysis in the loop (**AddAnalysis > Within Loop(s)> DC Operating Point**).

6. Add a Transient analysis in the loop (**AddAnalysis > Within Loop(s) > Transient**).
7. Edit the following Transient analysis fields by clicking on the newly added **tranalysis** button:

   End Time: 100u

   Time Step: 10n

   Monitor Progress: 100

   Plot After Analysis: No

   Plot File (Input/Output tab): tr_vary

   Data File (Input/Output tab): tr_vary
8. Add the new transient arguments by clicking on the **Accept** button.
9. Add a Plot command after the loop as follows:
   - Select **AddAnalysis > After Loop(s) > View Plotfiles in Scope**.
   - Click on the **plot** button**.**
   - Select **Open Only** in the Plot Action field.
   - Click on the **Accept** button.
10. Perform the vary sweep by clicking the **OK** button.

Step 5. Measure the frequency on the `out` signal.

1. Plot the `out` signal by selecting the `-analog - osc_mm.dut -out` signal from the `ex_osc_mm.tr_vary` Plot File window and clicking on the **Plot** button.
2. Display the Measurement Tool (**Tools > Measurement**).
3. Measure the frequency by editing the following fields in the Measurement tool:
   - Measurement: `Frequency` (Select **Time-Domain > Frequency** from the menu in this field
   - Signal: `osc_mm.dut/out` (selected from the menu in this field)
   - Create New Waveform on Active Graph: Frequency vs. osc_mm.dut/buf_rcd.b1/c.c1/c
4. Perform the Measurement by clicking the **Apply** button.

After you finish simulating this block, you can either continue to analyze this block using other analyses, parameters, and parts or you can examine other blocks in the Audio Test System example.

chapter 5

Designing the Analog-to-Digital Converter

This circuit converts the analog signal, produced by the CSP test tone processor, into a 8 bit digital signal for the Digital Signal Processing block in the Audio Test System example.

This circuit demonstrates the following capabilities:

- Using custom MAST models to implement functionality
- Using transient analysis to verify functionality of a design

This example uses the following process to develop and test the schematic block:

1. "Specifying Design Parameters" on page 5-1
2. "Selecting Models for the Circuit" on page 5-2
3. "Analyzing the Design" on page 5-4

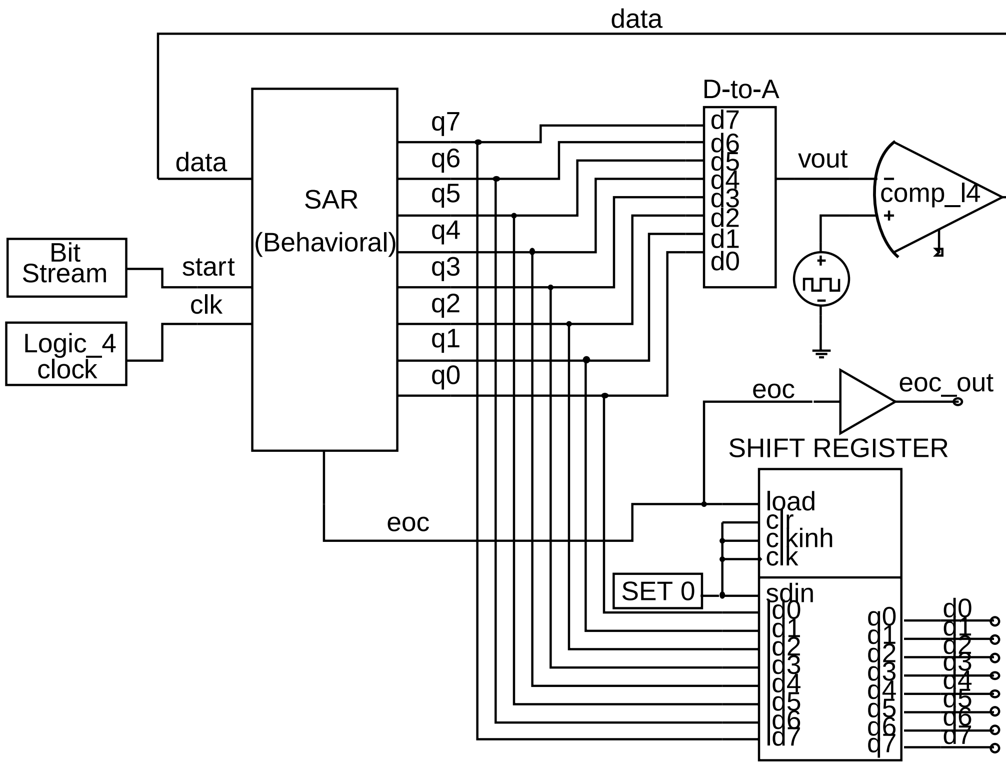
Specifying Design Parameters

The Analog-to-Digital converter must meet the following design parameters:

- Convert an analog signal into a 8-bit digital bus.
- Convert the analog signal every 400 milliseconds
- Produce an output pulse at the end of each conversion.

Selecting Models for the Circuit

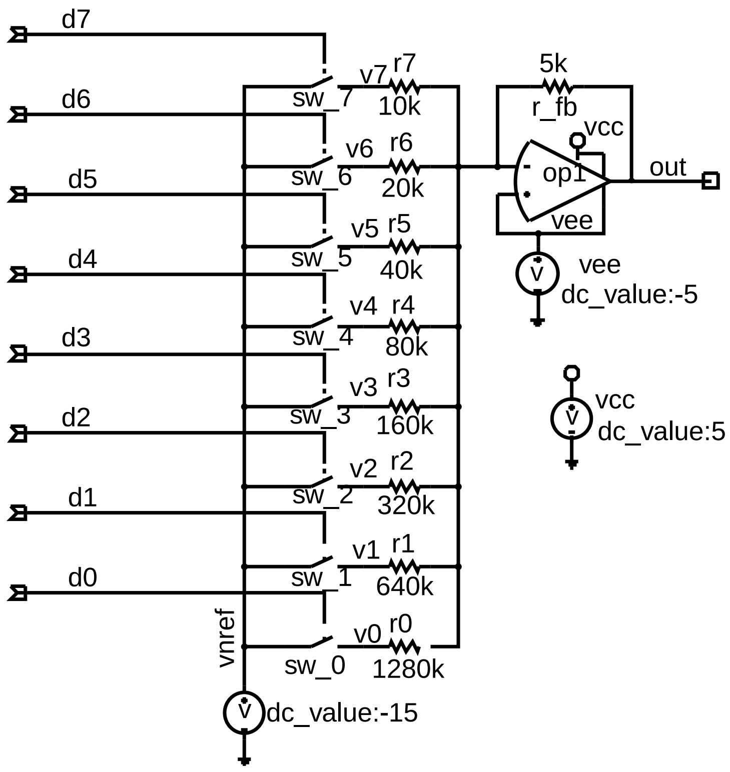
The following circuit was designed to meet the parameters specified in the previous section. After you examine the models used in this design, you can simulate it, as described in the next topic:



- The `set0` instance uses the **set_l4_0** template to provide a constant logic ‘0’ to some of the control pins on the shift register.
- The `sh1` instance uses the **shft8_l4** template to latch the output of the SAR when the end-of-conversion (eoc) signal goes high. Because the other control pins on this shift register are held at a logic ‘0’, this shift register acts as a 8-bit digital latch.
- The `pr1` instance uses the **prbit_l4** template to inform the SAR to begin a new conversion every 200 microseconds.
- The `clk1` instance uses the **clock_l4** to provide a 40 KHz digital clock signal for the SAR. In the Audio design, the Oscillator block generates this clock signal. In order to simulate this block independently from the larger system, the **clk1** instance was added.
- The `vin` instance uses the **v** template to provide a 12 volt peak-to-peak sawtooth waveform to test the full range of the analog-to-digital convertor. This waveform was generated by defining a pulse waveform with a 20 millisecond period and 10 millisecond rise and fall times.

This part was added so that the ADC could be tested independently from the complete Audio system. When used in the system level design, this waveform is produced by the test signal processor (csp) block.

- The `cmp1` uses the **comp_l4** template to compare the analog input signals from the d-to-a converter and the sawtooth waveform generator (vin). This part produces a digital signal.
- SAR model is a behavioral digital model written in MAST. It is simple compared to a full digital (gate level) representation, but produces similar results. This capability allows you to work with mixed levels of abstraction, using complete design details (gate or analog primitives) of one section concurrently with an abstract (high level behavioral) model of another section. Hence, simulation work can continue even with disparate levels of progress on individual sub-systems.
- The `d2a` instance implements a digital-to-analog convertor using a R2R ladder to convert the digital signal, produced by the SAR, into an analog signal for the comparator. The digital switches use the **sw_l4** template. The following schematic shows how this block was implemented:



Analyzing the Design

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command.

saber_home`/bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design dialog box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*`/Audio/ADC`).
3. Select the `ex_adc` file and open it.

The remaining steps exercise the analog-to-digital convertor (ex_adc) circuit from the Audio test system design, using a test ramp signal as input to the 8-bit SAR type A to D converter. The “d[0-7]” outputs are latched at the end of each conversion cycle, whereas the “q[0-7]” states are interesting to observe during each conversion, as they show the successive approximation process.

Step 3. Evaluate the DC Operating Point

1. Display the DC analysis form (**Analyses > Operating Point > DC Operating Point...**).

   This menu item displays the Operating Point Analysis form.

2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Point Report (**Results > Operating Point Report...**).

Step 4. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain >**

Transient...).

2. Edit the following fields in the Transient Analysis form:
   - End Time: 10m
   - Time Step: 10n
   - Monitor Progress: 100
3. Perform the analysis by clicking the **OK** button.

   This command determines the time domain response of the circuit during the first 10 milliseconds and saves the resulting waveforms for each signal on the root of the design in a Plot File with a `.tr` extension. It also displays the information to the SaberGuide transcript window on every 100th data point of the simulation.

Step 5. Plot the waveforms in Scope

You can observe the waveforms to verify that the ADC block is converting the analog waveforms properly.

After you finish simulating this block, you can either continue to analyze this block using other analyses, parameters, and parts, or you can examine other blocks in the Audio Test System example.

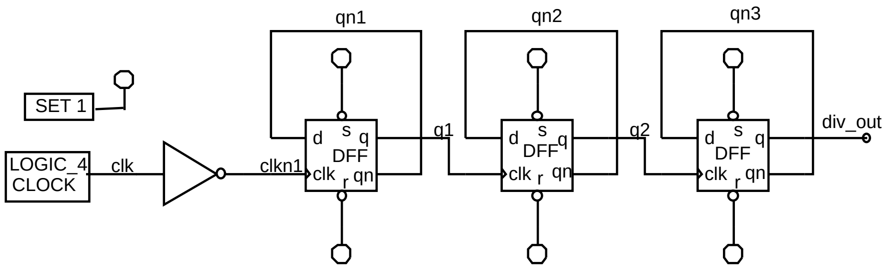
chapter 6

Designing the Divide by 8 Block

This circuit divides the 40 KHz clock signal, generated by the relaxation oscillator (osc) by 8 to produce a 5 KHz clock signal. The output of this block generates the correct sampling rate (at the right phase) for the analog-to-digital converter.

Implementing the Design

This simple circuit was implemented using parts from the Saber digital library as shown in the following figure:



The following list shows the models used in this design:

- `logic clock` provides the 40KHz clock signal for the design. In the Audio Test System example, this signal is provided by the Oscillator block. This model is only used to test the block separate from the Audio Test System.
- `set` provides a logic ‘1’ to the global net called ‘`vcc`’. This global net is attached to all set and reset pins on the D-type flip-flops in the circuit.
- `inv` inverts the clock signal so the signal produced by the “Divide by 8” block and the “Oscillator” block will be in-phase.
- `dff` provide the digital divide by eight functionality.

Simulating the Design

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command.

saber_home/`bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist

To open the Saber netlist follow these steps:

1. Display the Open Design Dialog Box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path/*`Audio/N8div`).
3. Select the `ex_n8div.sin` file and open it.

The remaining steps exercise the Divide-by-8 (`ex_n8div`) block of the Audio Test system. This block divides the clock signal from the Relaxation Oscillator by 8 to inform the system when the 8-bit ADC has finished converting the test tone signal to a digital waveform.

Step 3. Evaluate the DC Operating Point.

1. Display the DC analysis form (**Analyses > DC Operating Point > DC Operating Point...**).

   This menu item displays the Operating Point Analysis form.
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report...**).

Step 4. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**).

2. Edit the following fields in the Transient Analysis form:

   End Time: 1.1m

   Time Step: 5n

3. Perform the analysis by clicking the **OK** button.

   This command determines the time domain response of the circuit during the first 1.1 milliseconds and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`.

Step 5. View the transient response in Scope Waveform Analyzer

You can verify the “divide-by-8” functionality by plotting the `clk` and `div_out` signals. You can view these signals by opening the `tr` plot file in the Signal Manager, selecting the `clk` and `div_out` signal from the `ex_n8div.tr` plot file window and plotting the signals in the graph window.

After you finish simulating this block, you can either continue to analyze this block using other analyses, parameters, and parts or you can examine other blocks in the Audio Test System example.

chapter 7

Designing the DSP circuit

This circuit provides an "echo effect" of the signal originally produced by the Test Tone (csp) block. This circuit does some filtering using Z-domain models.

The DSP algorithm design performed here can later be transferred to the entire audio system simulation model, which includes the power amplifier and the electro-mechanical speaker model. This type of analysis can be of value to the designer, who must integrate the entire system and achieve an overall performance objective.

This circuit demonstrates the following capabilities:

- Using Fourier analysis to check the frequency spectrum of a transient signal
- Using Sampled Data System (SDS) models and Electrical system models in the same circuit

This example uses the following process to develop and test the schematic block:

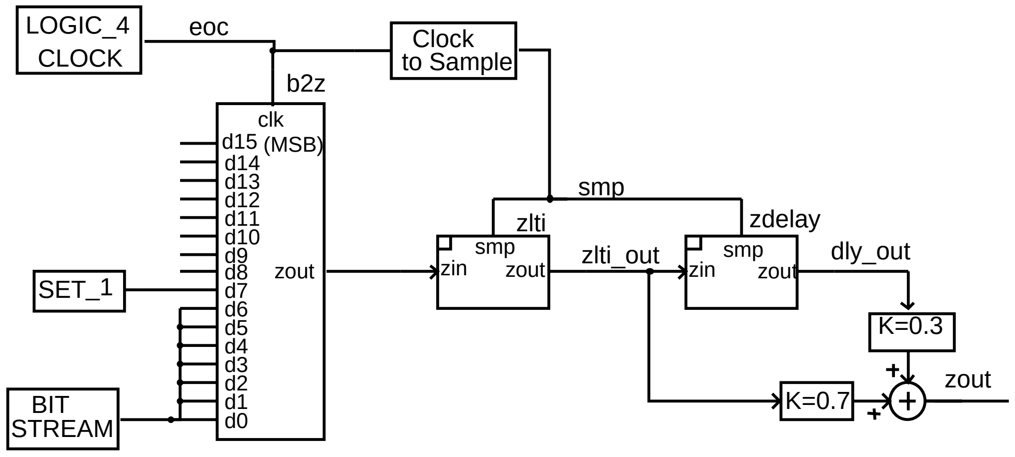
1. "Specifying Design Parameters" on page 7-1
2. "Selecting Models for the DSP Circuit" on page 7-2
3. "Analyzing the DSP Circuit" on page 7-3

Specifying Design Parameters

The simulation also shows the "echo" behavior of the delay function, which is set to a 500 sample delay (= 100 msec.) and with a 0.7 to 0.3 split ratio to the direct (non-delayed) signal. Also, compare the internal 16-bit output of the filter (zlti_out) with the more coarse 8-bit analog voltage output from the chip (dsp_out).

Selecting Models for the DSP Circuit

The following circuit was designed to meet the parameters specified in the previous section. After you examine the models used in this design, you can simulate it, as described in the next section.



- The `fix_d7` instance uses the **set_l4** template to keep the `d7` pin at a constant logic level '`1`' by editing the `level` property on the fix_d7 to "`_1`".
- The `clk1` instance uses the **clock_l4** template to provide a 5 KHz clock to the circuit by editing the `freq` property on the clk1 instance to `5k.` The output of this instance is a 4-state digital logic signal that oscillates between a logic 1 and a logic 0.
- The `pr_d3` instance uses the **prbit_l4** template to generate a programmable stream of logic_4 bits. In this case, the bit stream is connected to the 6 least significant bits of the zlti1 instance. The bit stream initializes to logic 0, goes to logic 1 at 100microseconds, and returns to logic 0 at 500 microseconds as defined by the `bits` parameter bits=[(0,_0),(100u,_1),(500u,_0)].
- The `zlti1` instance uses the **zlti** template to provide some additional filtering using a Z-domain transfer function. The **a** parameter defines the gain. The `den` and `num` parameters define the denominator and numerator of the Z-domain transfer function. The `min` and `max` parameters define the minimum and maximum output states of the instance.
- The `c2s1` instance uses the custom **clk2smp** template converts a convention, digital clock signal into a Z-domain type sampling signal. On the rising edge of the dig. clock, the z output changes state. The output looks like the frequency/2, but its sampling effect is at the

original clock frequency. You can view this simple MAST template by examining the ASCII file at *path*/`Audio/DSP/clk2samp.sin`.

- The `z2a1` instance uses the **z2a** template to convert the event-driven analog (Z-domain) signal into a continuous analog signal. There is a 100nanosecond delay during the conversation, due to editing the `tt` property to `100n`.
- The `add1` instance uses the Z-domain **zlcmb** template to sum the output direct and delayed signals. This instance multiples the direct signal by a factor of 0.7 and the delayed signal by a factor of 0.3 prior to summing the two Z-domain inputs (as defined using the `a` and `b` parameters). This instance also limits the output to +/- 5 units.
- The `dly1` instance uses the Z-domain **zdelay** template to delay the input signal by 500 clock cycles at the output by defining the `k` parameter to 500.
- The `b2z1` instance uses the **b2z** template to convert the 8 bit, binary, logic_4 signal to an event-driven analog signal.

Analyzing the DSP Circuit

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command:

install_home/`bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design Dialog Box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*/`Audio/DSP`).
3. Select the `ex_dsp.sin` file and open it.

The remaining steps exercise the DSP (ex_dsp) block of the Audio Test system. This block provides the "echo" feature of the Audio System and does some filtering.

Step 3. Evaluate the DC Operating Point.

1. Display the DC analysis form (**Analyses > Operating Point > DC Operating Point...**).

   This menu item displays the Operating Point Analysis form.

2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report...**).

Step 4. Determine the time-domain (transient) response.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form:

   End Time: 200m

   Time Step: 50u

   Monitor Progress: 300

   Signal List (Input/Output tab): `dsp_out dly_out`
   `smp zin zlti_out zout d0_6 d7 eoc`

3. Perform the analysis by clicking the **OK** button.

   This command determines the time domain response of the circuit during the first 200 milliseconds and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`. It also displays the information to the SaberGuide transcript window on every 300th data point of the simulation.

Step 5. Plot the results in Scope Waveform Analyzer.

Step 6. Continue the Transient analysis to 1 second.

1. Display the Transient form (**Analyses > Continue > Transient**).
2. Change the value of End Time to 1.
3. Continue the Transient analysis by clicking the **OK** button.

Step 7. Determine the frequency components of the zlti_out signal.

1. Display the FFT Transform form (**Analyses > Fourier > FFT**).
2. Edit the following fields in the Transient Analysis form:

   Signals to Transform: (Input/Output tab) zlti_out

   Input Plot File: (Input/Output tab) tr

   Output Plot File: (Input/Output tab) fft

   X-axis Scale (Control tab): Log

3. Perform the transform by clicking on the **OK** button.

   This command transforms the `zlti_out` curve into the frequency spectrum.

Step 8. Plot the frequency spectrum of the zlti_out signal in Scope.

Note that the “notch” characteristic may help suppress the loudspeaker resonance at 56 Hz, in the actual audio system.

After you finish simulating this block, you can either continue to analyze this block using other analyses, parameters, and parts or you can examine other blocks in the Audio Test System example.

chapter 8

Designing the Power Amplifier

This circuit amplifies the signal, with a gain of 150, after it is filtered by the RLC filter block. The output of this block drives the speaker block of the Audio Test System.

This circuit demonstrates the following capabilities:

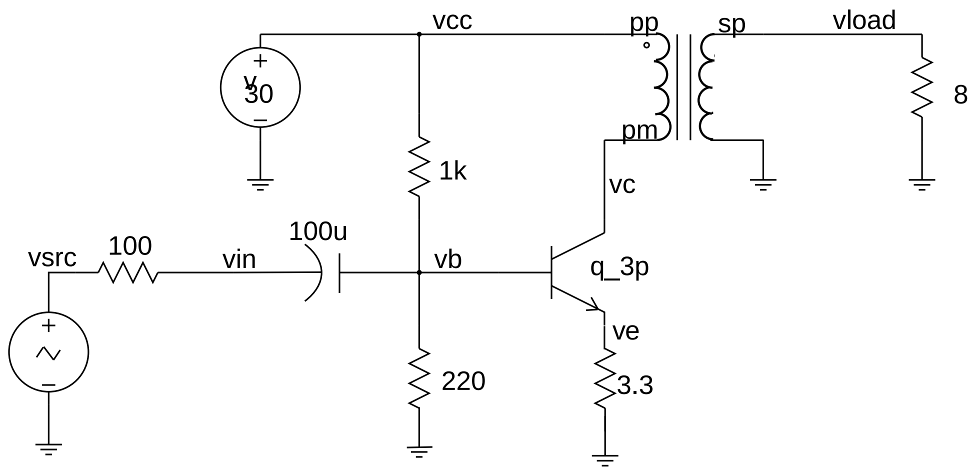
- Using Stress analysis to identify individual component stress levels during operation
- Using Fourier Transform to analyze harmonics
- Using Noise analysis to analyze the noise contribution of the various parts in the design
- Using the ssp command to extract the small signal parameters

This example uses the following process to develop and test the schematic block:

1. "Selecting Models for the Circuit" on page 8-2
2. "Analyzing the Power Amplifier circuit" on page 8-3

Selecting Models for the Circuit

The following figure shows the schematic used to implement the power amplifier:



The following models were used to implement the power amplifier design shown in the previous figure:

- The **SaberInclude** symbol altered the default value `c_vrmax=5`, `r_pdmax=1`, and `c_vmax=50`.
- Resistors `r1`, `r2`, and `re` provide biasing for the transistor amplifier.
- `rload` provides a 8 ohm load to mimic the speaker in the next stage. In order to determine the stress on this part, the ratings property was used to specify a maximum power dissipation (pdmax_ja) of 20 and a maximum voltage drop (vmax) over the resistor of 30 volts.
- `x1` modifies the generic **xfr** transformer template to meet the design specification. the ratings property was edited provide spec ratings for Stress analysis.
- `vcc` provides a 30 volt DC voltage source for the circuit.
- `vin` provides a 100 Hz, 5Volt peak-to-peak sinewave voltage source by using `tran=(sin=(va=5,f=100,vo=0)`. An 1 volt AC waveform was also defined in the voltage source.
- `q1` customizes the generic **q_3p** transistor template to meet the design specifications. This symbol modified the ratings property to provide stress ratings and the model property to specify transistor parameters such as beta, saturation current, and internal resistances.

Analyzing the Power Amplifier circuit

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command:

install_home`/bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design dialog box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*`/Audio/Pwramp`).
3. Select the `ex_pwramp` file and click the **Open** button.

The following steps exercises the power amplifier (ex_pwramp) block of the Audio Test system. This block amplifies the signal for the loudspeaker. These steps check for a smooth output sine waveform, determine stress levels on critical parts, determine harmonic content of the output signal using Fourier analysis, examine whether any components contribute significant amount of noise to the design and determine the small signal gain of the amplifier.

Step 3. Perform a nominal transient analysis.

This analysis run will provide a brief excitation interval (50 msec.) during which the 100 Hz, 5v peak input signal will be amplified by ~150.

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient analysis form:
   - End Time: `50m`
   - Time Step: `1u`
   - Run DC analysis first: `Yes`

- Plot after analysis: `Yes - Open Only`
- Max Truncation Error (Calibration tab): `10u`

3. Perform the analysis by clicking the **OK** button.

This command examines the transient response over the first 50milliseconds of operation and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`. After the analysis completes, Saber automatically adds the plot file to the Signal Manager.

When you view the output signal (`vload`) of the amplifier, you should see a sinewave output.

Step 4. Run a Stress analysis.

1. Display the Stress Analysis form (**Analyses > Stress...**).
2. Edit the following fields in the Stress Analysis form:
   - Use Input from: `Transient Analysis`
   - Input Data File: `tr`
   - Report after analysis: `Yes`
   - Derating File Name:(Transformations): `derating.file`
   - XWindow (Transformations tab): `10m`
     (to roughly model the effect of heat capacity)
3. Perform the analysis by clicking the **OK** button.

   This command uses the information in the transient data file from the previous step to determine the stress levels on each component in the design. After the analysis completes, Saber displays the Stress Report in the SaberGuide Transcript window.

   As shown in the stress report the power transistor's voltage rating (`vcemax` = 50 volts) was exceeded by 14%.

   Note that although the supply voltage (vcc) is only 30 volts, vce is overstressed. Note also that the emitter resistor's (`r.re`) derated power rating (`pdmax`) is borderline stressed.

Step 5. Continue the transient analysis.

To allow any start-up transients to settle out, continue the transient analysis until 100 milliseconds.

1. Display the Continue Transient Analysis form (**Analyses > Continue > Transient...**).

2. Specify the new End Time to be `100m` and click **OK**.

Step 6. Determine the harmonic content of the amplifier output.

To accomplish this task, you will run a Fourier analysis on the amplifier output waveform, `vload`. The Fourier analysis will be based on the 100 Hz fundamental frequency (from the input source), and will transform the last observed cycle in the transient analysis run. It will include the first 14 harmonics, out to 1.3 kHz.

1. Display the Fourier Transform form (**Analyses > Fourier > Fourier...**).
2. Edit the following fields in the Fourier Transform form:
   - Number of Harmonics `14`
   - Fundamental Frequency: `100`
   - Period End: `end`
   - Plot after analysis: `Yes - Open Only`
   - Input Data File: (Input/Output tab) `tr`
   - Output Plot File: (Input/Output tab) `fou`
3. Perform the analysis by clicking the **OK** button.

   This command transforms the time-domain signals into the frequency spectrum. This transform saves the resulting waveform in a Plot File called `fou`.

   After the transform completes, add the `fou` Plot File to the Signal Manager in Scope Waveform Analyzer.
4. In Scope, observe the output waveform (`vload` in plot file `fou`) on a dB scale. There is about 37 dB separation between the fundamental (at 100 Hz) and the 2nd harmonic (at 200 Hz), and over 50 dB separation with the 3rd harmonic (at 300 Hz).

Step 7. Increase the peak input amplitude so that the output signal is clipped.

In this step, you will increase the input voltage amplitude to 7 volts.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.
3. Select the `v_sin.vin` instance from the Hierarchical Instance listbox. For Cadence, use `v_sin.vin1`
4. Click the **Edit** button in the List/Alter Design form.

This action displays the Alter Components form.

5. Change the parameter value on the `v_sin.vin` instance to: `ac_phase=0,offset=0,frequency=100,amplitude=7,ac_mag=1`
6. Click on the **OK** button to change the value in the in-memory Saber netlist.

Step 8. View the effects of the clipped output waveform

In this step, you will re-run the transient and Fourier analyses, and observe the clipping effects both in the time-domain and in the frequency spectrum

1. Display the Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form.
   - End Time: `100m`
   - Time Step: `1u`
   - Plot after analysis: `Yes - Open Only`
   - Plot File (Input/Output tab): `tr_7in`
   - Data File (Input/Output tab): `tr_7in`
   - Max Truncation Error (Calibration tab): `10u`
3. Perform the analysis by clicking the **OK** button.

   When you view the output signal (`vload`) from the `tr_7in` plot file, you should notice the clipping of the load voltage in the transient waveform. You can now use the Fourier analysis to determine how the clipped waveform affects the frequency spectrum (harmonic content).
4. Display the Fourier Transform form (**Analyses > Fourier > Fourier...**).
5. Edit the following fields in the Fourier Transform form:
   - Number of Harmonics: `14`
   - Fundamental Frequency: `100`
   - Period End: end
   - Plot after analysis: `Yes - Open Only`
   - Input Data File: (Input/Output tab) `tr_7in`
   - Output Plot File: (Input/Output tab) `fou_7in`

6. Perform the analysis by clicking the **OK** button.

   When you view the frequency spectrum of the `vload` signal from the `fou_7in` plot file, you should notice that the harmonic content has grown substantially in the Fourier spectrum, with 26 dB separation from the 2nd harmonic and only 18 dB from the 3rd harmonic. Clearly, you do not want to overdrive this power amplifier.

Step 9. Perform a noise analysis.

This analysis shows which of the passive and active components are contributing most to the output noise spectrum.

1. Display the Noise Analysis form (**Analyses > Frequency > Noise...**).
2. Edit the following fields in the Small-Signal Noise Analysis form.
   - Starting Frequency: `10`
   - End Frequency: `100k`
   - Output Signal List: `vload`
   - Number of Points: `1000`
   - Plot after analysis: `Yes - Open Only`
3. Perform the analysis by clicking the **OK** button.

   When you view the noise spectrum on the `q_3p` signal, you should notice that the spectrum peaks at 49 kHz, with a spectral density of approximately 500nV/rt(Hz). The power transistor is shown to be the greatest contributor, with much smaller noise being generated by the circuit's resistors.

Step 10. Determine the small-signal gain and input impedance.

Perform a "two-port" analysis, to find the power amplifier's gain and input impedance as a function of frequency. First, the external source impedance is removed, so that only the power amplifier's internal characteristics are measured. The proper loading (8 Ohms) is retained.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.
3. Select the `r.rsrc` instance from the Hierarchical Instance listbox.
4. Click the **Edit** button in the List/Alter Design form.
5. This action displays the Alter Component form.
6. Change the parameter value on the `r.rsrc` instance to `1m`.

7. Click on the **OK** button to change the value in the in-memory Saber netlist.
8. Display the Two-Port Analysis form (**Analyses > Frequency > Two-Port...**).
9. Edit the following fields in the Two-Port Analysis form.
   - Starting Frequency: `10`
   - Ending Frequency: `1k`
   - Input Source: `v(v_sin.vin)` [For Cadence, use `v(v_sin.vin1)`]
   - Output Ports: `vload`
   - Number of Points: `1000`
   - Plot after analysis: `Yes - Open Only`
10. Perform the analysis by clicking the **OK** button.

    The input impedance `zin(v(v_sin.vin))` is high at very low frequencies (due to the AC coupling capacitor), but maintains the desired level (approximately 100 Ohms) over the operating range of 30 Hz to 1kHz. The amplifier gain is approximately 4.5 (i.e. 13 dB), and is reasonably flat across the operating band. This gain is determined by the product of the transformer turns ratio (2:1) and the r.rload/r.re ratio.

    Because of the feedback effect of the emitter resistor, the amplifier gain is not strongly dependent on the individual transistor parameters. If the circuit design were modified to operate without this feedback, then the transistor parameters would be significant. The “ssp” (Small Signal Parameters) analysis can be used to identify these parameters.

Step 11. Eliminate the external emitter resistance.

First the circuit is modified to eliminate the external emitter resistance, (with corresponding bias circuit changes), and the “ssp” analysis and another two-port analysis are performed.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.
3. Select the `r.re` and `r.r1` instances from the Hierarchical Instance listbox.
4. Click the **Edit** button in the List/Alter Design form.

This action displays the Alter Component form.

5. Change the following parameters:
    - `r.re`: Change `rnom` to `1m`
    - `r.r1`: Change `rnom` to `2.5k`
6. Click on the **OK** button to change the value in the in-memory Saber netlist.

Step 12. Determine the Small-Signal Parameters.

In this step, you will re-run the DC analysis (because the resistor value changes affected the operating point in the design), and determine the small-signal parameters using the ssp command and the two-port analysis.

1. Re-run the DC analysis (**Analyses > Operating Point > DC Operating Point...**) using the default analysis values.
2. Run the Small-Signal Report (**Results > Small-Signal Report**) using the default values.

    The results of the Small Signal Parameters Report (**`ssparlist`**) are shown in the output transcript. Note that the significant parameters are the transconductance (`gmf=29.4`), bp-ep resistance (`rpi=3.12`), bias resistance (`rx = 2`) and the internal emitter resistance (`re=0.05`). Over the desired frequency band, these parameters, in conjunction with the external circuit parameters, determine the transfer gain of the amplifier.

    The Small Signal Parameters Report is useful for identifying a linearized device model at an operating point. For example, if the bias circuit is modified, you can easily see how these key performance parameters are affected.
3. Display the Two-Port Analysis form (**Analyses > Frequency > Two-Port...**).
4. Edit the following fields in the Two-Port Analysis form:
    - Starting Frequency: `10`
    - Ending Frequency: `1k`
    - Input Source: `v(v_sin.vin)`
    - Output Ports: `vload`
    - Number of Points: `1000`
    - Plot after analysis: `Yes - Open Only`

- Plot File (Input/Output tab): `tp_xre`

5. Perform the analysis by clicking the **OK** button.

   In the two-port analysis results note that the gain rises to approximately 150, near the high frequency end of the band (1kHz). At lower frequencies, the AC coupling capacitor affects the gain, but at higher frequencies, gain is set by the load resistance and the transistor parameters:

   ```
   Gain = (2*rload)*(gm)*(1/[(1+gm*re) +
                              (rx/rpi)]) = 150
   ```

chapter 9

Designing the RLC circuit

This circuit provides additional filtering for the Audio Test System example. The main focus of this circuit block is to demonstrate the following Saber capabilities using a simple circuit:

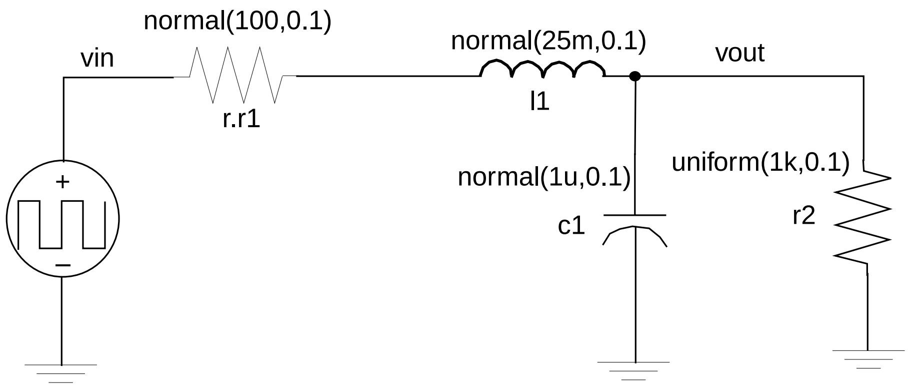
- Using DC, Transient, and AC analyses to verify the functionality of a design
- Using Vary, Sensitivity, and Monte Carlo analyses to tune the parameters in the design

This example uses the following process to develop and test the schematic block:

1. "Selecting Models for the RLC Filter Circuit" on page 9-2
2. "Verifying the Functionality of the RLC Filter Circuit" on page 9-2
3. "Sweeping Design Parameters" on page 9-5
4. "Determining Parameter Sensitivity" on page 9-7
5. "Analyzing the Statistical Effects of Part Variation Using Monte Carlo Analysis" on page 9-9

Selecting Models for the RLC Filter Circuit

The following schematic shows the circuit used to meet the design.



The following list describes the components used in the previous schematic:

- `r1` is a 100 ohm resistor. The `rnom` property value of normal(100,0.1) informs Saber that the 100 ohm value is used for all analyses except Monte Carlo. During MC analysis, Saber varies the resistor value within 10% of the nominal 100 ohm value using a normal distribution curve. The available distribution curves for Monte Carlo analysis are described in the SaberBook topic titled "Statistical Modeling with MAST" on page -1.
- `r2` is a 1000 ohm resistor with a 10% tolerance level. This resistor used the uniform distribution during Monte Carlo analysis.
- `v1` provides a waveform used in transient and small-signal analyses. The transient waveform specification `tran=(pulse=(v1=0,v2=1,tr=1u,tf=1u,td=1m,pw=5m,per=10m)` provides a 1 volt peak-to-peak square wave with 1 microsecond rise and fall times 10 millisecond period and a 50% duty cycle.
- `l1` uses the generic **l** inductor template to implement a 25 millihenry inductor.
- `c1` uses the generic **c** capacitor template to implement a 25microfarad capacitor.

Verifying the Functionality of the RLC Filter Circuit

After you make a local copy of the design, you can simulate the various block within the Audio system in SaberDesigner. The following procedure invokes

SaberDesigner on the pre-generated netlist in the RLC directory of the Audio Test System:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command:

install_home`/bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design Dialog Box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*`/Audio/RLC`).
3. Select the `ex_rlc.sin` file and open it.

The remaining steps analyze the transient and frequency domain response of the ex_rlc circuit using transient and AC analyses and measurement capability to examine key performance results.

Step 3. Evaluate the DC Operating Point.

1. Display the Operating Point Analysis form (**Analyses > Operating Point > DC Operating Point...**).
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report...**).

Step 4. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form:
   - End Time: `100m` (10 cycles X 10 millisecond period)
   - Time Step: `10n` (1/100th of the periodic input signal)
   - Monitor Progress: `100`

- Plot after analysis : `Yes - Open Only`
- Max Truncation Error (Calibration tab): `100u`

3. Perform the analysis by clicking the **OK** button.

   This command examines the effects of the first 10 cycles of the 1KHz input sine wave and saves the resulting waveforms for each signal on the root of the design in a Plot File with a `.tr` extension. It also displays the information to the SaberGuide transcript window on every 100th data point of the simulation.

   After the analysis completes, Saber automatically adds the plot file to the Signal Manager.

Step 5. Plot the output voltage (`vout`) waveform.

Step 6. Measure the key time-domain performance characteristics of the output voltage

1. Using the Measurement Tool (**Tools > Measurement**), measure the overshoot by editing the following fields in the Measurement Tool and clicking the **Apply** button:

   Measurement: `Overshoot` (**Time-domain > Overshoot**)

   Signal: `vout`

2. To measure the risetime, edit the following fields in the Measurement Tool and click on the **Apply** button:

   Measurement: `risetime` (**Time-domain > Risetime**)

   Signal: `vout`

You can examine the rise times on the other rising edges by placing the mouse cursor over either risetime measurement marker, holding down the left mouse button, and sliding the mouse cursor to the next rising edge.

Step 7. Analyzing the Frequency response.

1. Display the Small-Signal Frequency Analysis form (**Analyses > Frequency > Small-Signal AC...**).

   Edit the following fields in the Small-Signal Frequency Analysis form:

   Starting Frequency: `10`

   Ending Frequency: `100k`

   Number of Points: `1000`

Plot after analysis: `Yes - Open Only`

2. Perform the analysis by clicking the **OK** button.
3. Select and plot the `vout` signal.

   This command examines the frequency response between 10 and 100KHz. The analysis uses 1000 logarithmically-spaced data points and saves the resulting waveforms for each signal on the root of the design in a Plot File called ac. After the analysis completes, Saber automatically adds the plot file to the Scope Signal Manager.

Step 8. Measure the key frequency-domain performance characteristics of the filter using the Measurement Tool.

1. To measure the break frequency, edit the following fields in the Measurement Tool and click on the **Apply** button:

   Measurement: `Lowpass` (**Frequency Domain> Lowpass (3dB Point)**)

   Signal: `vout` (The `dB(V):f(Hz)` waveform)

   Offset: `-3`

2. Edit the following fields in the Measurement Tool and click on the **Apply** button:

   Measurement: Threshold (**General > Threshold (At Y)**)

   Signal: `vout`

   Y value: `-90`

   Trigger: falling edge icon

3. To measure the slope of the frequency roll-off, edit the following fields in the Measurement Tool and click on the **Apply** button:

   Measurement: `Slope` (**Frequency Domain> Slope**)

   Signal: `vout`

   X value: `10k`

   Option: `Per Decade`

Sweeping Design Parameters

To perform a parametric analysis of the RLC filter, you will sweep (vary) the value of resistor r2 across the expected range of the potentiometer, perform a time domain simulation at each value, then use the measurement capability to generate a performance vs. design parameter curve.

Step 1. Vary the value of the r.r2 resistor.

1. Display the Looping Commands form (**Analyses > Parametric > Vary...**).
2. Define the parameter sweep by clicking on the **vary** loop button.
3. In the resulting Parameter Sweep form, edit the following fields:

   Parameter Name: `rnom(r.r2)`

   from `300` to `1k` by `100`
4. Click the **Accept** button to add the vary definition to the Looping Command form.
5. Add a DC analysis to the loop (**AddAnalysis > Within Loop(s) > DC Operating Point**).
6. Add a Transient analysis to the loop (**AddAnalysis > Within Loop(s) > Transient**).
7. Click on the **tranalysis** button within the loop and edit the following values:

   End Time: `50m`

   Time Step: `10n` (1/100th of the periodic input signal)

   Monitor Progress: `100`

   Plot after analysis : `Yes - Open Only`

   Max Truncation Error (Calibration tab): `100u`

   Plot File (Input/Output tab): `tr_vary`
8. Click the **Accept** button to add the **tranalysis** definition to the Looping Command form.

   To summarize, the Looping Commands form should have the following entries:

   vary: `vary rnom(r.r2) from 300 to 1k by 100`

   dcanalysis: dc

   ```
   tranalysis: tr (monitor 100,pfile tr_vary,
           tend 50m,terror 100u, tstep 10n
   ```
9. You can execute the Vary loop by clicking the **OK** button in the Looping Commands form.

Step 2. Plot the multi-member `vout` **waveform.**

1. In the Signal Manager, click the **Open Plotfiles** button.
2. Select the ex_rlc.tr_vary Plot File and click **Open**.
3. Select `vout` in the ex_rlc.tr_vary Plot File window and click on the **Plot** button.

   A multi-member waveform appears. Each member of the waveform is associated with a resistance value from the Vary loop.

Step 3. Measure the overshoot voltage vs. resistance of the r2 instance.

This step requires a Batch Measure license.

1. Display the Measurement Tool (**Tool > Measurement**).
2. Edit the following fields:

   Measurement: `Overshoot` (**Time-domain > Overshoot**)

   Signal: `vout`

   Create New Waveform on Active Graph: `Overshoot vs. rnom(r.r2)`
3. Create the `overshoot vs. rnom(r.r2)` curve by clicking on the **Apply** button in the Measurement tool.

Determining Parameter Sensitivity

This topic requires an InSpecs Parametric Analysis license.

This sensitivity analysis runs multiple time domain simulations, changing specified parameters (one at a time) by a small amount, and measuring the resulting change in performance (overshoot). The relative influence of these parameters is then reported. To execute the sensitivity analysis, follow these steps:

1. Display the Sensitivity Analysis form (**Analyses > Parametric > Sensitivity...**).
2. Edit the following fields in the top section of the Sensitivity Analysis form:

   Parameter List: `r.r1/rnom r.r2/rnom c.c1/c`
   `l.l1/l`

   Perturbation: `0.01`

Report after analysis: `Yes`

3. Add a DC analysis to the analysis list at the bottom of the form (**AddAnalysis > Basic > DC Operation Point**).
4. Add a Transient analysis to the analysis list at the bottom of the form (**AddAnalysis > Basic > Transient**).
5. Edit the following fields in the Transient Analysis form by clicking the **tranalysis** button in the analysis list at the bottom of the form.

   End Time: `50m`

   Time Step: `10n` (1/100th of the periodic input signal)

   Monitor Progress: `100`

   Plot after analysis : `No`

   Max Truncation Error (Calibration tab): `100u`

   Plot File (Input/Output tab): `tr_sens`

   When you finished making these changes, click the **Accept** button in the Transient Analysis form.
6. Add a overshoot performance measure to the analysis list at the bottom of the form (**AddAnalysis > Batch Measure**).
7. Click on the **measure** button in the analysis list and edit the following fields in the Batch Measurement form.

   Measure: `Overshoot` (Select **Time-Domain > Overshoot** from the pulldown menu in the Measure field)

   Input Plot File: `tr_sens`

   Curve Name: `vout`

   When you finished making these changes, click the **Accept** button in the Batch Measurement form.

   In summary, the completed Sensitivity Analysis form should contain the following entries:

   Parameter List: `r.r1/rnom r.r2/rnom c.c1/c l.l1/l`

   Perturbation: `0.01`

   Report after analysis: `Yes`

   dcanalysis: dc

   tranalysis:
   ```
   tr (monitor 100,pfile tr_sens,
       tend 50m,terror 100u,testep 10n
   ```

measure: `meas overshoot (cnames vout,pfin`
`tr_sens`

Execute the Sensitivity analysis by clicking the **OK** button the Sensitivity Analysis form.

This analysis runs produces a Sensitivity Report indicating that the `r1` resistor affects the overshoot measurement more than any other parameter.

Analyzing the Statistical Effects of Part Variation Using Monte Carlo Analysis

This Monte Carlo analysis randomly assigns (within their tolerance range) values to parameters and records these assigned values. These varied parameters may later be checked for correlation with performance variations using the calculator. A time domain simulation is run for each set of assigned values, and the measurement capability is used to generate statistical design information.

This procedure requires an InSpecs Statistical Analysis license.

The following list describes how to complete the Monte Carlo form:

1. Display the Looping Commands form (**Analyses > Statistical > Monte Carlo**).
2. Edit the following fields in the Monte Carlo form by clicking the **mc** button in the Looping Commands form.

   Runs: `50`

   Seed: `Constant`

   Parameter List: `rnom(r.r1) rnom(r.r2) c(c.c1) l(l.l1)`

   Parameter File: `pars_mc`

   When you finished making these changes, click the **Accept** button in the Monte Carlo Analysis form.
3. Add a DC analysis to the loop body section of the Looping Commands form (**AddAnalysis > Within Loop(s) > DC Operation Point**).
4. Add a Transient analysis to the loop body section of the Looping Commands form (**AddAnalysis > Within Loop(s) > Transient**).
5. Edit the following fields in the Transient Analysis form by clicking the **tranalysis** button to the loop body section of the Looping Commands form:

End Time: `50m`

Time Step: `10n` (1/100th of the periodic input signal)

Max Truncation Error (Calibration tab): `100u`

Plot File (Input/Output tab): `tr_mc`

When you finished making these changes, click the **Accept** button in the Transient Analysis form.

6. Measure the overshoot of each generated curve (**AddAnalysis > After Loop(s) > Batch Measure**).
7. Click on the **measure** button and edit the following fields in the Batch Measurement form.

   Measure: `Overshoot` (Select **Time-Domain > Overshoot** from the pulldown menu in the Measure field)

   Input Plot File: `tr_mc`

   Curve Name: `vout`

   Output Plot File (Transform tab): `over_mc`

   When you finished making these changes, click the **Accept** button in the Batch Measure form.

8. Generate a Histogram of the Overshoot Measurement (**AddAnalysis > After Loop(s) > Histogram**).
9. Click on the **pfhistogram** button and edit the following fields in the Plot File Histogram form:

   Curve Name: `over(vout)`

   Input Plot File: `over_mc`

   Output Plot File: `over_hist`

   When you finished making these changes, click the **Accept** button in the Plot File Histogram form.

10. Examine the completed Looping Commands form.

    The finished form should contain the following entries:

```
mc: mc (parfile pars_mc,parlist rnom(r.r1)
      rnom(r.r2) c(c.c1) l(l.l1),runs 50 seed
      constant

  dcanalysis: dc

  tranalysis: tr (monitor 100,pfile tr_mc,
                 tend 50m,terror 100u,tstep 10n
```

end:

```
measure: meas overshoot (cnames vout,pfin
        tr_mc,pfout over_mc

pfhistogram: pfhist (cnames over(vout),pfin
                     over_mc,pfout over_hist
```

11. Execute the Looping Commands form containing the Monte Carlo Analysis by clicking the **OK** button.
12. View the resulting Plot Files in Scope Waveform Analyzer.

    You can view the histogram by selecting the `count` signal in the `ex_rlc.over_hist` Plot File window.
13. View the correlation trend, showing that as r1 increases, the value of the overshoot tends to decrease.
    a. Display the Calculator (**Tools > Calculator**) in Scope.
    b. Enter the overshoot measurement into the calculator by displaying the `over_mc` plot file, selecting the `Over(vout)` signal, moving the mouse cursor to the X-register in the calculator, and pressing the middle mouse button.
    c. Enter the `rnom(r.r1)` waveform from the `pars_mc` plot file into the X-register of the calculator.
    d. The calculator should now display the `Over(vout)` and `rnom(r.r1)` waveforms in the stack listing.
    e. Select the **Wave > f(x)** menu item from within the calculator.
    f. Plot the correlation by pressing the **Graph X** button in the icon bar of the calculator.

chapter 10

Designing the Loud Speaker Circuit

This circuit models the load speaker at the end of the Audio Test System example. This circuit demonstrates the following capabilities:

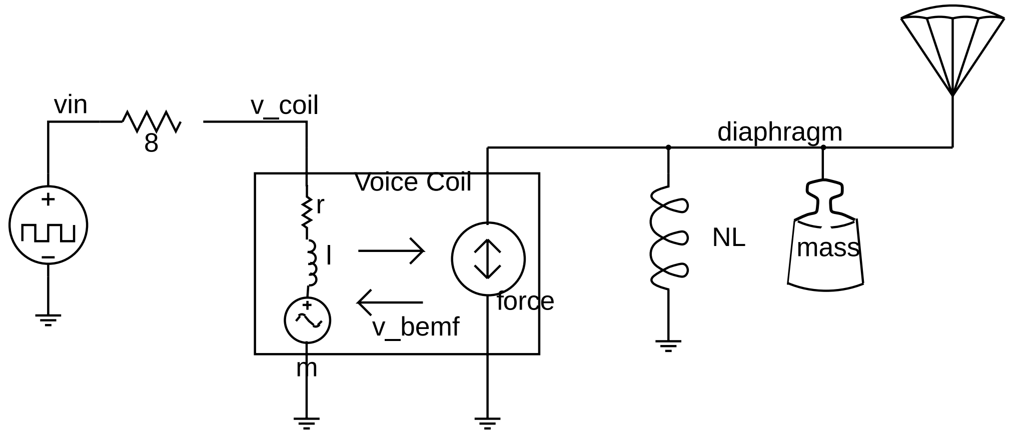
- Writing Custom MAST templates
- Using DC Transfer to determine the static response
- Using Distortion analysis
- Using Fourier and FFT Transforms

These analyses provide a better understanding of the complex behavior of this non-linear system. This example uses the following process to develop and test the schematic block:

1. "Selecting Models for the Loudspeaker" on page 10-2
2. "Determining the Static Response of the Loudspeaker Circuit" on page 10-2
3. "Analyzing the Non-Linear Response" on page 10-4
4. "Analyzing the Linear Response" on page 10-6
5. "Analyzing the Distortion Effects" on page 10-9

Selecting Models for the Loudspeaker

The following schematic implements the Loud Speaker design:



- The `vin` instance uses the **v** template to implement the expected input signal from the previous stage (power amplifier).
- The `susp` instance uses the **spring_nl** template to implement the force that pulls the voice coil back to equilibrium.
- The **voice_coil** template is a custom MAST template to translate the source electrical signal into speaker cone movement.
- The `rsrc` instance uses the **r** template to represent the 8 ohm load of the speaker.
- The `air` instance uses the **winddrag** template to model the resistance of the speaker cone through air.
- The `mdia` instance uses the **mass** template to represent the weight of the speaker coil.

The loudspeaker design characteristics are based on performance specifications given in "*Loudspeaker and Headphone Handbook*", Edited by John Borwick, 2nd Edition, Focal Press.

Determining the Static Response of the Loudspeaker Circuit

After you make a local copy of the design, you can simulate the Audio system in SaberDesigner. The following procedure invokes SaberDesigner on the pre-generated netlist in the Audio directory:

Step 1. Invoke SaberDesigner

(UNIX) Enter the following command:

install_home`/bin/saber`

(Windows NT) Choose the following menu item:

Start > Analogy > *saberdesigner* **> SaberGuide**

Step 2. Open the Saber netlist.

To open the Saber netlist follow these steps:

1. Display the Open Design dialog box (**File > Open > Design...**).
2. Browse to the directory containing the Audio example files (*path*`/Audio/Lspkr`).
3. Select the `ex_lspkr` file and open it.

The following steps exercise the Loud Speaker (`ex_lspkr`) block of the Audio Test system. These steps determine the static response, analyze both the non-linear and linear responses, and analyze the effects of non-linear distortion on the design.

First, perform a static (dt) analysis to determine the diaphragm displacement as a function of applied dc voltage. The suspension stiffness (spring_nl) is non-linear with displacement.

Step 3. Determine the DC Operating Point.

1. Display the Operating Point Analysis form (**Analyses > Operating Point > DC Operating Point...**).
2. Execute the DC analysis by clicking the **OK** button.

   This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report**).

Step 4. Sweep the input voltage and determine the Operating Point at each value.

1. Display the DC Transfer Analysis form (**Analyses > Operating Point > DC Transfer...**).
2. Edit the following fields in the DC Transfer Analysis form:

   Independent Source: `v(v.vin)`
   [In Cadence, use `v(v.vin1)`]

Sweep Range: `Step By`

from: `-30` to: `30` by: `0.1`

Plot after analysis: `Yes - Open Only`

Signal List (Input/Output tab): `diaphragm`

3. Perform the analysis by clicking the **OK** button.

This command determines the operating points when the input voltage source is swept from −30 volts to 30 volts and saves `diaphragm` waveform in a Plot File called `dt`. After the analysis completes, Saber automatically adds the plot file to the Signal Manager. By plotting the diaphragm waveform in Scope Waveform Analyzer, you can determine the maximum movement of the diaphragm.

Analyzing the Non-Linear Response

Perform a small signal “ac” analysis and compare to the fft results. Note there is considerable difference in shape near the resonance peak. This is largely due to the non-linear damping effect of the air-drag on the moving diaphragm, which is not accounted for in the “ac” result.

This section assumes that you already invoked Saber on the lspkr design and found the DC operating point.

Step 1. Determine the time-domain (transient) analysis.

Now perform a transient impulse response test. A 4kv pulse with unity area is applied, and the diaphragm response is observed.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form:

End Time: `1`

Time Step: `100n`

Monitor Progress: `300`

Plot after analysis: `Yes - Open Only`

Max Truncation Error (Calibration tab): `10u`

Max Time Step (Integration Control tab): `0.2m`

Plot File: (Input/Output tab): `tr`

3. Perform the analysis by clicking the **OK** button.

   This command examines the transient response over the first second of operation and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`. It also displays the information to the SaberGuide Transcript Window on every 300'th data point of the simulation. After the analysis completes, Saber automatically adds the plot file to the Signal Manager and opens it. Plot the diaphragm response by selecting the `diagphragm` signal in the Plot File window and clicking on the **Plot** button.

Step 2. Determine the frequency components at the `diaphragm` and `vin` nodes.

1. Display the FFT Transform form (**Analyses > Fourier > FFT...**).
2. Edit the following fields in the FFT form:

   Number of Points: 4096

   Plot after analysis: `Yes - Open Only`

   Signals to Transform (Input/Output tab): `diaphragm`
   `vin`

   Input Plot File: (Input/Output tab): `tr`

   Output Plot File: (Input/Output tab): `fft`

3. Perform the analysis by clicking the **OK** button.

   This command determines the frequency components of the `diaphragm` and `vin` signals and saves the resulting waveform in a Plot File called `fft`**.** After the analysis completes, Saber automatically adds the plot file to the Signal Manager and opens it. You can plot the results of the analysis by selecting `diaphragm` and `vin` in the Plot File window and then clicking on the **Plot** button.

Step 3. Double the AC magnitude of the v.vin source to mimic the fft frequency foldover.

The fft displays spectral amplitude at non-zero frequencies as the sum of the positive and negative frequency components. As these are equal for physical systems, the values are twice the expected single-sided levels. This is observed by looking at the spectrum of the input impulse, which is flat at 6 dB (rather than 0 dB) beyond 1kHz.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.

3. Select the `v.vin` [in Cadence, `v.vin1`] instance from the Hierarchical Instance listbox.
4. Click the **Edit** button in the List/Alter Design form.

   This action displays the Alter Components form.
5. Change the `mag` parameter to 2.
6. Click on the **OK** button to change the value in the in-memory Saber netlist.

Step 4. Determine the linear frequency response.

1. Display the Small-Signal Analysis form (**Analyses > Frequency > Small-Signal AC...**).
2. Edit the following fields in the AC Analysis form:

   Starting Frequency: `1`

   Ending Frequency: `1k`

   Number of Points: `1024`

   Plot after analysis: `Yes - Open Only`
3. Perform the analysis by clicking the **OK** button.

   This command examines the frequency response between 1 and 1KHz. The analysis uses 1024 logarithmically-spaced data points and saves the resulting waveforms for each signal on the root of the design in a Plot File called ac. Later in this example, you will transform the diaphragm waveform produced by the AC analysis into the time-domain using the iFFT transform. Because the iFFT requires the number of input data point to be a power of 2, the number of data point in the AC analysis was set to the default value used by the iFFT transform (1024).

   After the analysis completes, Saber automatically adds the plot file to the Scope Signal Manager and opens the plot file. Plot the diaphragm response by selecting the `diagphragm` signal in the Plot File window and clicking on the Plot button.

Analyzing the Linear Response

In this section, you will linearize the system by zeroing the non-linear parameters of both the spring and the air-damping effect. Repeat the impulse response and the fft analysis, and compare this new spectrum with the small signal ac results. Also, perform an inverse fft analysis on both the fft generated spectrum and the ac generated spectrum, and compare these with the (linear) transient impulse response.

The following steps assume that you already have Saber invoked on the ex_lspkr design.

Step 1. Zero out the non-linear parameters.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.
3. Select the `spring_nl.susp` and `winddrag.air` instances from the Hierarchical Instance listbox.
4. Click the **Edit** button in the List/Alter Design form.

   This action displays the Alter Components form.
5. Change the following parameters:

   spring_nl.susp: Change `k3` to `0`

   winddrag.air: Change `w` to `0`
6. Click on the **OK** button to change the value in the in-memory Saber netlist.

Step 2. Determine the time-domain (transient) response

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form.

   End Time: `1`

   Time Step: `100n`

   Monitor Progress: `300`

   Plot after analysis: `Yes - Open Only`

   Plot File: (Input/Output tab): `tr_lin`

   Max Truncation Error (Calibration tab): `10u`

   Max Time Step (Integration Control tab): `0.2m`
3. Perform the analysis by clicking the **OK** button.

   This command examines the transient response over the first second of operation without the non-linear parameters and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr_lin`. It also displays the information to the SaberGuide Transcript Window on every 300'th data point of the simulation.

After the analysis completes, Saber automatically adds the plot file to the Signal Manager and opens it.

Step 3. Determine the frequency components at the `diaphragm` node

1. Display the FFT Transform form (**Analyses > Fourier > FFT...**).
2. Edit the following fields in the FFT form.

   Plot after analysis: `Yes - Open Only`

   Signals to Transform (Input/Output tab): `diaphragm`

   Input Plot File: `tr_lin`

   Output Plot File: `fft_lin`
3. Perform the analysis by clicking the **OK** button.

   This command determines the frequency components of the diaphragm signal without the non-linear system parameters set and saves the resulting waveform in a Plot File called `fft_lin`. After the analysis completes, Saber automatically adds the plot file to the Signal Manager.

In the next two steps, you will transform the FFT results from the linear transient response and the previous AC analysis run. Because both plot files sources are from linear frequency responses, the transform time-domain results should be similar.

Step 4. Transform the linear frequency response of the diaphragm signal into the time-domain.

1. Display the IFFT Transform form (**Analyses > Fourier > IFFT...**).
2. Edit the following fields in the IFFT form:

   Plot after analysis: `Yes - Open Only`

   Signals to Transform (Input/Output tab): `diaphragm`

   Input Plot File: `fft_lin`

   Output Plot File: `ifft_fft`
3. Perform the analysis by clicking the **OK** button.

   This command determines the time-domain response of the `diaphragm` signal using the previous FFT analysis as input. This command saves the resulting waveform in a Plot File called `ifft_fft`. The plot file is added to the Signal Manager and opened.

Step 5. Transform the linear frequency response of the diaphragm signal into the time-domain.

1. Display the IFFT Transform form (**Analyses > Fourier > IFFT...**).
2. Edit the following fields in the IFFT form:

   Plot after analysis: `Yes - Open Only`

   Signals to Transform (Input/Output tab): `diaphragm`

   Input Plot File: `ac`

   Output Plot File: `ifft_ac`

3. Perform the analysis by clicking the **OK** button.

   This command determines the time-domain response of the `diaphragm` signal using the previous AC analysis as input. This command also saves the resulting waveform in a Plot File called `ifft_ac`.

   The plot file is added to the Signal Manager and opened. You should now be able to compare the `diaphragm` curve from the `ifft_ac` and `ifft_fft` plot files.

Analyzing the Distortion Effects

In this topic, you will use distortion analysis and the Fourier transform to determine distortion effects in the loud speaker design.

Step 1. Add the non-linear parameters.

Restore the original non-linearities that were removed during the non-linear analysis of the design. Also, apply a DC bias, set the input voltage to a sinusoid, and setup the ac source to be the same amplitude as the transient source.

1. Display the List/Alter Design form (**Edit > List/Alter...**).
2. Select the Netlist tab.
3. Select the `spring_nl.susp`, `winddrag.air`, and `v.vin` [in Cadence, `v.vin1`] instances from the Hierarchical Instance listbox.

4. Click the **Edit** button in the List/Alter Design form.

   This action displays the Alter Component form.

5. Change the following parameters:

   spring_nl.susp: Change `k3` to `95meg`

   winddrag.air: Change `w` to `0.1`

   v.vin [in Cadence, `v.v1`]: Change to `ac=(5,0),` `tran=(sin=(va=5,f=33,vo=15))`

6. Click on the **OK** button to change the value in the in-memory Saber netlist.

Step 2. Determine the DC Operating Point.

1. Display the Operating Point Analysis form (**Analyses > Operating Point > DC Operating Point**).
2. Execute the DC analysis by clicking the **OK** button.

This action performs a DC analysis on the circuit. You can view the resulting DC values in the Operating Report (**Results > Operating Point Report...**).

Step 3. Determine the distortion products at the diaphragm signal.

1. Display the Distortion Analysis form (**Analyses > Frequency > Distortion...**)
2. Edit the following fields in the Small-Signal Distortion Analysis form:

   Starting Frequency: `1`

   End Frequency: `1k`

   Output Signal List: `diaphragm`

   Number of Points: `1000`

   Plot after analysis: `Yes - Open Only`

   Compute Desensitization (Control tab): `Yes`

3. Perform the analysis by clicking the **OK** button.

   This command determines the distortion products of the `diaphragm` signal and saves the resulting waveforms in a Plot File called `ds`. The plot file is added to the Signal Manager and opened. Select and plot the signals.

4. Select the **Graph > Members...** menu choice. The Member Attributes dialog box appears. The distortion types are listed by their signal names, in this case HD2, HD3, CMP2, and CMP3.

Step 4. Determine the time-domain (transient) response of the system.

In this step, you will determine the time-domain response of the loud speaker design. You will then determine the frequency spectrum of the `diaphragm` waveform using the Fourier transform and compare the harmonics produced by the fourier and distortion analyses.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the Transient Analysis form:

   End Time: `300m`

   Time Step: `100n`

   Monitor Progress: `300`

   Plot after analysis: `Yes - Open Only`

   Max Truncation Error (Calibration tab): `10u`

   Max Time Step (Integration Control tab): `1m`

   Plot File: (Input/Output tab): `tr_bias`
3. Perform the analysis by clicking the **OK** button.

   This command determines the transient response and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr_bias`. It also displays the information to the SaberGuide Transcript Window on every 300'th data point of the simulation.

   After the analysis completes, Saber automatically adds the plot file to the Signal Manager and opens it.

Step 5. Determine the frequency components at the diaphragm node.

1. Display the Fourier Transform form (**Analyses > Fourier > Fourier...**).
2. Edit the following fields in the Fourier Analysis form:

   Number of Harmonics: `5`

   Fundamental Frequency: `33`

   Period End: `end`

Plot after analysis: `Yes - Open Only`

Input Data File (Input/Output tab): `tr`

Output Plot File: `fou`

3. Perform the analysis by clicking the **OK** button.

   This command transforms the time-domain signals into the frequency spectrum. This transform determines the 6 values (the fundamental frequency plus the first five harmonics) and saves the resulting waveform in a Plot File called `fou`. After the analysis completes, Saber automatically adds the plot file to the Signal Manager and opens it.

Step 6. Determine the average voltage at the diaphragm node.

With the `diaphragm` waveform from the `tr_bias` plot file displayed in Scope, you can measure the average voltage at the `diaphragm` node by following these steps:

1. Display the Measurement tool (**Tools > Measurement**).
2. Edit the following fields in the Measurement Tool and click on the **Apply** button:

   Measurement: `Levels (average)`(**Levels > Average**)

   Signal: `diaphragm`

Step 7. Analyze the results in Scope Waveform Analyzer.

The static (dc) position of the diaphragm is 1.624 mm with the 15 volts bias. The measured average position of the diaphragm for the 33 Hz transient (dynamic) analysis is 1.593 mm. The difference is due to “compression”, as indicated by the distortion analysis value of CMP2 at 33 Hz (-26.1 dB). This value, de-normalized by the fundamental amplitude at 33 Hz (0.586 mm), yields the dc compression of 0.03 mm. Also, compare the 2nd harmonic predicted by the small signal distortion analysis vs. the actual (large signal) results of the Fourier analysis. Both show the 2nd harmonic (at 66Hz) approximately −10 dB down from the fundamental.

chapter 11

Range Finder IC Example

This example demonstrates top-down and bottom-up design methodologies of a mixed- signal system. The circuit can be simulated with the Saber/Verilog mixed-simulator package, or it can be simulated with Saber's native mixed-signal simulator. In native mode, the Monte Carlo and Measurement capabilities of the InSpecs package are shown.

To run this example you must have the following library licenses: Standard Template Library (STL), Optional Template Library (OTL), and the Component Library (CL).

This example does not go into the detailed operation of the circuit. Rather, it compares three methods of simulating a mixed-signal design so that you can compare the speed and results of each method:

- a Saber analog simulation with the digital circuitry represented as MOS gates
- a Saber native mixed-signal simulation with the digital circuitry represented as digital models
- a Saber/Verilog mixed-signal simulation with the digital circuitry represented as digital models

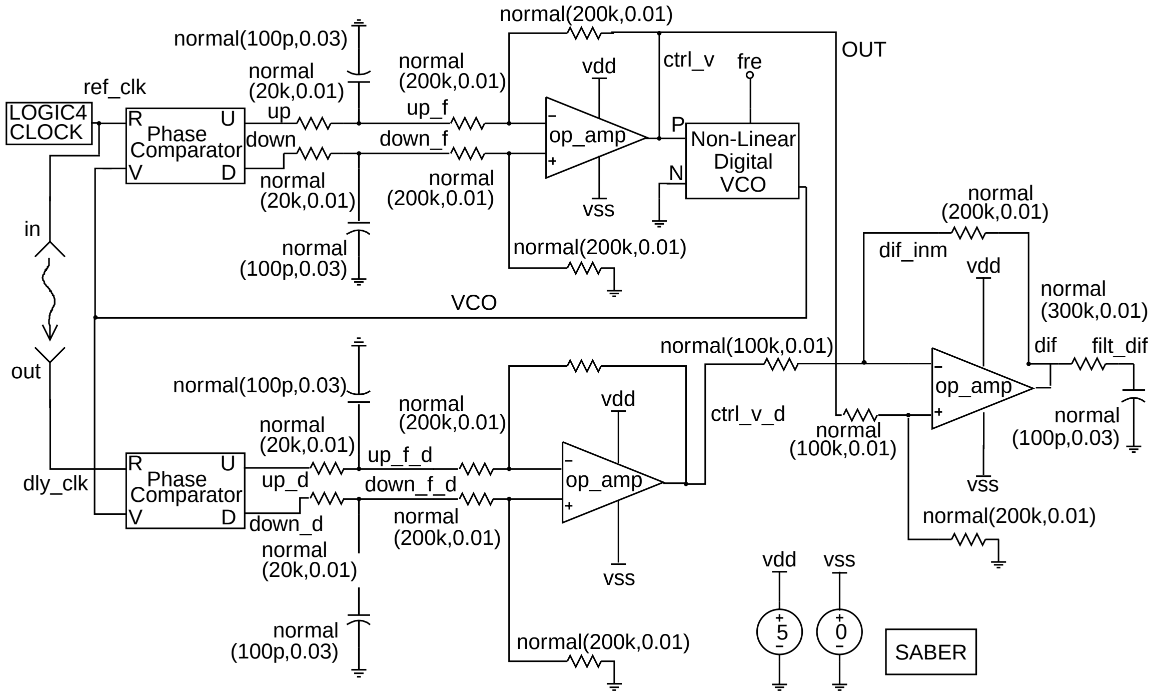
This example is supported in the SaberSketch, DVE (Mentor Graphics), and Artist (Cadence) environments. Prior to simulating this design, you must make a local copy.

You can simulate this example at the following levels of abstraction:

- "Testing the MOS-Level Range Finder Design Example" on page 11-3—Saber analog simulation
- "Testing the Range Finder OR Gate in Saber" on page 11-12—characterization of a two-input OR gate
- "Testing the Gate-Level Range Finder Design Example in Saber" on page 11-19—Saber native mixed-signal simulation
- "Testing the Gate-Level Range Finder Design Example in Saber-Verilog" on page 11-27—Saber/Verilog mixed-signal simulation

Selecting Models for the Design

This circuit illustrates the mixed-signal capabilities of Saber using a range finding system, whose output voltage is a function of the round trip time delay of a transmitter-receiver pair. As the time delay increases, the phase difference between two input channels of the phase lock loop (PLL) increases, and this difference is amplified by the difference amp. The following schematic shows the simulation representation of a Phase Lock Loop:



Schematic files of the example circuit:

- `range` - Top level circuit of range
- `opamp1` - MOS schematic of opamp
- `phsd` - Gate level schematic of digital phase comparator
- `and2` - MOS schematic of and2
- `buf` - MOS schematic of buf
- `or2` - MOS schematic of or2
- `ilch` - MOS schematic of ilch
- `tst_or` - Top level characterization circuit for OR gate
- `xpath` - transmitter-receiver pair symbol, calls `buf_l4.sin`

This example also shows how to create and use custom Hypermodels at the boundary of the analog and digital domains.

Testing the MOS-Level Range Finder Design Example

This section will give you instructions on copying the example, viewing the MOS-level version in your schematic capture system, and performing a Saber analog simulation on the MOS-level version.

Prior to testing the range finder example, you must make a local copy of the design files based on the environment that you are using (either SaberSketch, Artist, or DVE). This procedure will prevent the original example files from being overwritten.

The following topics describe how to set up the design in your environment:

SaberSketch

- "Viewing the Range Finder Design in SaberSketch" on page 11-3
- "Preparing Range Finder Design for Simulation - SaberSketch" on page 11-4

Artist

- "Viewing the Range Finder Design in Artist" on page 11-5
- "Preparing Range Finder Design for Simulation - Artist" on page 11-6

DVE

- "DVE Range Finder Design Set Up" on page 11-7
- "Viewing the Range Finder Design in DVE" on page 11-7
- "Preparing Range Finder Design For Simulation - DVE" on page 11-8

Viewing the Range Finder Design in SaberSketch

To make a local copy of the range finder design and view the design, perform the following steps:

Copy the Design

1. Copy the example in the following directory to a local location:

UNIX - *install_home*`/example/SaberSketch/RangeFinderIC`
Windows NT - *install_home*`\example\SaberSketch\RangeFinderIC`

2. Navigate to your local copy of the `RangeFinderIC` directory.
3. (Windows NT only) You must change the file permissions of your local copy of the designs so that they are no longer read-only as follows:
   a. Invoke Windows NT Explorer.
   b. Navigate to your local copy of the design. In the case of a design example that has more than one directory, you will need to change the permissions of all the files in each of them as described in steps c through f.
   c. In each directory, select all the files (**Edit > Select All**).
   d. Open the Properties dialog box (**File > Properties**) and select the General tab.
   e. Un-check the Read-only box.
   f. Click **OK**.

Open the Schematic

4. To open the schematic `range` start the SaberSketch design editor by typing:

   install_home/`bin/sketch`

5. Click on the **File > Open > Design** items.
6. Navigate to the `RangeFinderIC` directory, click on the `range.ai_sch` file in the Open Design dialog box, and click on the **Open** button.
7. Use the schematic for the design by selecting the **Design > Use > range** menu item.
8. Specify the netlister options as described in the next topic.

Preparing Range Finder Design for Simulation - SaberSketch

To prepare for simulation, you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. Choose the **Edit > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd`. In the Ground Net Name field, type `vss`.
3. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.

4. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
    a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
    b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.
    c. Navigate to the `range.shm` file.
    d. Select the `range.shm` file and click the **Open** button in the Select dialog box.
    e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
    f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
6. Close the Saber/Netlister Settings form by clicking on the **Close** button.
7. Start the netlister by selecting the **Design > Netlist range** menu item.
8. Invoke Saber by selecting the **Design > Simulate range** menu item.
9. If the SaberGuide Transcript window is not visible, click the **>cmd** icon to display it.

You are now ready to simulate the MOS-level design in Saber. Go to page 11-9.

Viewing the Range Finder Design in Artist

To make a local copy of the range finder design and view the design, perform the following steps:

1. Copy the example in the following directory to a local location:

    install_home`/example/Cadence/RangeFinderIC`

2. Navigate to your local `RangeFinderIC` directory.
3. Invoke `icms`.
4. Create a new `pll_range` library with the **Tools > Library Path Editor** pulldown menu item.

5. In the icms window, start the Artist schematic entry application by choosing the **File > Open** pulldown menu items.
6. In the Library Name field type `pll_range`.
7. In the Cell Name type `range`.
8. In the View Name select `config`. Click on the **OK** button.
9. Select the **Saber > Set Working Directory** menu item. In the Project Information dialog box, insert your working directory (*path*/`RangeFinderIC)` path into the Project Directory field.

You are now ready to setup the netlister options, as described in the next topic.

Preparing Range Finder Design for Simulation - Artist

To prepare for simulation, you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd!`. In the Ground Net Name field, type `vss!`.
3. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
4. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:

   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

   b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.

   c. Navigate to the `range.shm` file.

   d. Select the `range.shm` file and click the **Open** button in the Select dialog box.

   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

5. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
6. Close the Saber/Netlister Settings form by clicking on the **Close** button.
7. Start the netlister by selecting the **Saber > Netlist > Start netlister** menu item.
8. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are now ready to simulate the MOS-level design in Saber. Go to page 11-9.

DVE Range Finder Design Set Up

You will first make a local copy of the design. In order for the Design Viewpoint Editor to read the local copy of your example, you must define the `SABER_EXAMPLE` environment variable and set up the `SABER_DATA_PATH` as described in the following procedure:

1. Using `dmgr`, copy the example in the following directory to a local location:

   install_home`/example/MentorGraphics/RangeFinderIC`

2. Add to your location map the softpath `SABER_EXAMPLE`, whose value is your current working directory, the directory that contains `RangeFinderIC`.

   `setenv SABER_EXAMPLE` *your_data_path*

3. Change your working directory to `RangeFinderIC`.

   ```
   cd $SABER_EXAMPLE/RangeFinderIC
   ```

4. Create a `SABER_DATA_PATH` environment variable.

   ```
   setenv SABER_DATA_PATH
   $SABER_EXAMPLE/RangeFinderIC/templates
   ```

Viewing the Range Finder Design in DVE

In your `RangeFinderIC` directory perform the following steps.

1. Start the Design Viewpoint Editor application by typing:

   ```
   dve range
   ```

2. Setup Saber by selecting the **Setup > Saber** menu item.
3. Select the **File > Save Design Viewpoint > With Same Name > Cleanup Un-used References** menu item.
4. Click on the **OPEN SHEET** icon to open the schematic.

Preparing Range Finder Design For Simulation - DVE

To prepare for simulation, you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.` In the Ground Net Name field, type `vss.`
3. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range.map`.
4. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.map` file (*path*`/RangeFinderIC/templates/range.map`).
   d. Select the `range.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.

6. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
    a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
    b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.
    c. Navigate to the `range.shm` file (*path*`/RangeFinderIC/templates/range.shm`).
    d. Select the `range.shm` file and click the **Open** button in the Select dialog box.
    e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
    f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.
7. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
8. Close the Saber/Netlister Settings form by clicking on the **Close** button.
9. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.
10. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are now ready to simulate the MOS-level design in Saber. Go to page 11-9.

Simulating the MOS-level Range Finder in Saber

You will now perform a Saber analog simulation on the MOS-level circuit.

The circuit will first be simulated at the MOS transistor level. There are roughly 150 MOS transistors, contained within the digital gates of the phase comparators and the opamp. However, the long simulation times makes it difficult to make design changes. To get faster simulation times, the transistor level circuits are characterized as behavioral models, and these behavioral models are substituted for the transistor level models. Because they are characterized, they are accurate, and because they are behavioral, they are fast. The circuit can now be quickly changed and rerun, or multiple runs, such as temperature range, can be performed.

When SaberGuide appears and the design is loaded, then load the batch command file `range.scs` in the Saber Transcript window. To load the batch file follow these steps:

1. In the SaberGuide Transcript window, select **File > Saber Command File...** to open the Load Command File dialog box and to navigate to the `range.scs` file.
2. Choose the batch file called `range.scs` and click the **Open** button.

   The `range.scs` command file executes the following Saber commands.

   a. Evaluate the DC Operating Point.

   ```
   dc (algst dyn_ramp
   ```

   The `algst dyn_ramp` options set the analysis to use dynamic supply ramping.

   b. Determine the Time-Domain (transient) response. (**NOTE:** Sketch and Artist create a plot file called `range.tr`, and DVE creates a plot file called `range_mos.tr`.)

   ```
   tr (te 50u, ts 10p, mon 100)
   ```

   This command determines the time domain response of the circuit during the first 50 micro seconds with an initial time step at 10 pico seconds. Analysis information is displayed every 100 time steps.

   The `range.scs` command script performs a transient analysis that can take up to 10 minutes to run.

When the simulation finishes, view the graphs of the output voltage, `filt_dif`, and the output of the diffamp, `dif`.

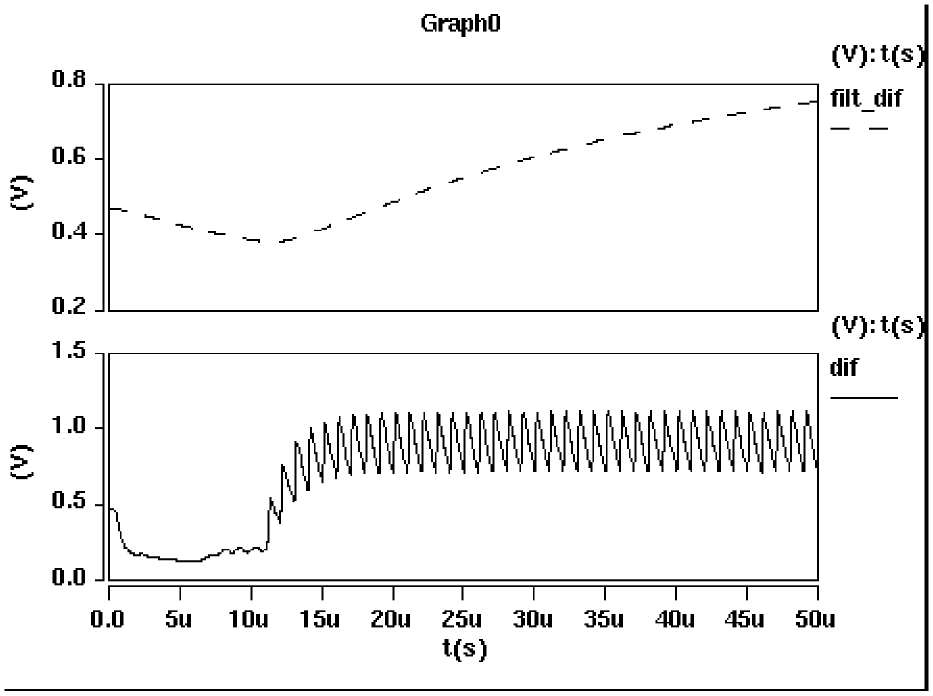
Graph the Output Voltages

To graph the output voltage, `filt_dif`, and the output of the diffamp, `dif`, follow these steps.

1. In the SaberGuide Transcript window, select **Results > View Plotfiles in Scope...** to open Scope Waveform Analyzer.
2. Click **OK** in the View Plotfiles dialog box to load the last generated plot file.

   Note that Sketch and Artist create a plot file called `range.tr`, and DVE creates a plot file called `range.tr_mos`.
3. Select the signals `dif` and `filt_dif` by holding the <Control> key and clicking on each signal.

4. Graph the signals in the graph window by pressing the **Plot** button.



The output voltage, `filt_dif` has not reached its steady state value, though the output of the diffamp, `dif`, has.

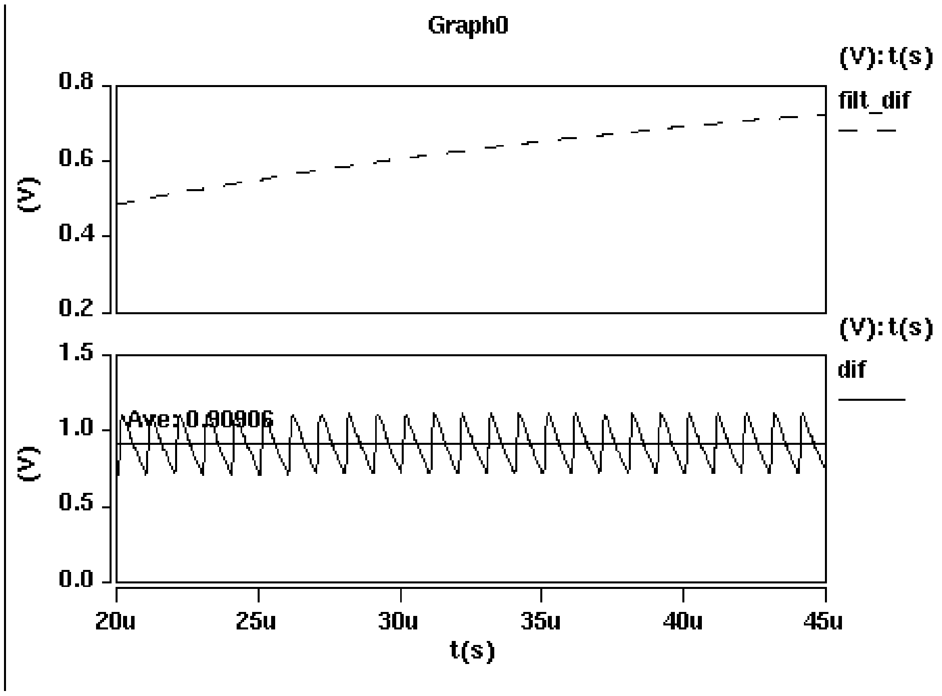
Next, use the Measurement tool to find the average value of the `dif` signal.

Find the Average Value of the Diffamp Output

In Scope Waveform Analyzer, use the Measurement tool to find the average value of the `dif` signal. This is faster than running the simulation until the output has settled. To find the average value follow these steps.

1. Place the mouse cursor at approximately 20u in the time axis, press, hold and drag the cursor to the right and stop at around 45u. This will display the flat part of the signal.
2. Click on the `dif` signal in the graph window.
3. Click on the Measurement tool icon in the Scope Tool bar. This will open the Measurement dialog box.
4. Click on the downward pointing arrow in the Measurement field. Choose the **Levels > Average** items.
5. Click on the **Visible X and Y range only** check box, and press the **Apply** button. The average voltage will be displayed in the graph.

6. Click on the **Close** button to close the dialog box.



7. To continue, see the next topic titled "Testing the Range Finder OR Gate in Saber" on page 11-12.

Testing the Range Finder OR Gate in Saber

This example shows how to characterize the 2-input OR gate.

The boolean logic of the gate is straight forward; it is the timing information that needs to be determined. Distributions are placed on the geometry of transistors. 100 Monte Carlo simulations were run, the delay time was automatically measured, and a distribution was created. This distribution is used as an argument in the behavioral model. This method is not shown for the other devices. This characterized digital model is used in our behavioral level simulation.

The following topics show how to view and prepare the design for simulation in your environment:

SaberSketch

- "Viewing the Range Finder OR Gate in SaberSketch" on page 11-13
- "Preparing to Simulate the Range Finder OR Gate in SaberSketch" on page 11-13

Artist

- "Viewing the Range Finder OR Gate in Artist" on page 11-14
- "Preparing to Simulate the Range Finder OR Gate in Artist" on page 11-15

DVE

- "Viewing the Range Finder OR Gate in DVE" on page 11-16
- "Preparing to Simulate the Range Finder OR Gate in DVE" on page 11-16

Viewing the Range Finder OR Gate in SaberSketch

1. To open the schematic `tst_or` start the SaberSketch design editor by typing:

   (UNIX) On a command line, enter *install_home*`/bin/sketch`

   (Windows NT) **Start > Analogy >** *saberdesigner* **> SaberSketch**

   An empty schematic window appears.
2. Click on the **File > Open > Design** items. Click on the **Open** button.
3. Click on the `tst_or.ai_sch` file in the Open Design dialog box, and click on the **Open** button.
4. Use the schematic for the design by selecting the **Design > Use > tst_or** menu item.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Range Finder OR Gate in SaberSketch

The following steps describe how to change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. Choose the **Edit > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.`In the Ground Net Name field, type `vss.`
3. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.

4. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
    a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
    b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.
    c. Navigate to the `range.shm` file.
    d. Select the `range.shm` file and click the **Open** button in the Select dialog box.
    e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
    f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
6. Close the Saber/Netlister Settings form by clicking on the **Close** button.
7. Start the netlister by selecting the **Design > Netlist tst_or** menu item.
8. Invoke Saber by selecting the **Design > Simulate tst_or** menu item.
9. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the MOS-level OR gate in Saber. Go to page 11-18.

Viewing the Range Finder OR Gate in Artist

In your `RangeFinderIC` directory perform the following steps.

1. Invoke `icms`.
2. Start the Artist schematic entry application by choosing the **Open > Design** pulldown menu items.
3. In the Library Name field type `pll_range`.
4. In the Cell Name field type `tst_or`.
5. In the View Name field type `schematic`. Click on the **OK** button.

6. Select the **Saber > Set Working Directory** menu item. In the Project Information dialog box, insert your working directory path into the Project Directory field.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Range Finder OR Gate in Artist

The following steps describe how to change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd!`. In the Ground Net Name field, type `vss!`.
3. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
4. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.shm` file.
   d. Select the `range.shm` file and click the **Open** button in the Select dialog box.
   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
6. Close the Saber/Netlister Settings form by clicking on the **Close** button.

7. Start the netlister by selecting the **Design > Netlist > Start Netlister** menu item.
8. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are now ready to simulate the MOS-level OR gate in Saber. Go to page 11-18.

Viewing the Range Finder OR Gate in DVE

In your `RangeFinderIC` directory perform the following steps.

1. Start the Design Viewpoint Editor application by typing:

   ```
   dve tst_or
   ```

2. Setup Saber by selecting the **Setup > Saber** menu item.
3. Select the **File > Save Design Viewpoint > With Same Name > Cleanup Un-used References** menu item.
4. Click on the **OPEN SHEET** icon to open the schematic.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Range Finder OR Gate in DVE

The following steps describe how to change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.` In the Ground Net Name field, type `vss.`
3. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range.map`.

4. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:

   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

b. Click on the **Browse...** button in the Add Entry dialog box. The Select dialog box appears.

c. Navigate to the `range.map` file (*path*/`RangeFinderIC/templates/range.map`).

d. Select the `range.map` file and click the **Open** button in the Select dialog box.

e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.

5. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.

6. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:

   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

   b. Click on the **Browse...** button in the Add Entry dialog box. The File Selection dialog box appears.

   c. Navigate to the `range.shm` file (*path*/`RangeFinderIC/templates/range.shm`).

   d. Select the `range.shm` file and click the **OK** button in the File Selection dialog box.

   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

   f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

7. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.

8. Close the Saber/Netlister Settings form by clicking on the **Close** button.

9. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.

10. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.

You are now ready to simulate the MOS-level OR gate in Saber. Go to page 11-18.

Simulating the MOS-Level Range Finder OR Gate in Saber

You will now simulate the two-input OR gate.

When SaberGuide appears and the design is loaded, then load the batch command file `tst_or.scs` in the SaberGuide Transcript window. To load the batch file follow these steps:

1. In the SaberGuide Transcript window, select **File > Saber Command File...** to open the Load Command File dialog box and to navigate to the `tst_or.scs` file.
2. Choose the batch file called `tst_or` and press the **Open** button.

   The `tst_or.scs` command file executes the following Saber commands.

   a. Limit the Signal List to the data you are specifically interested in.

   ```
   sigl in_in_out out
   ```

   b. Use `in_in_out` as the reference waveform for calculating delay.

   ```
   refcn in_in_out
   cn delay(out,in_in_out)
   ```

   c. Perform Monte Carlo analysis on the listed parts.

   ```
   mc (runs 100, seed constant, parl
       or2.*/m.*/w or2.*/m.*/l, parf mcparf)
   dc
   tr (te 40n, ter 1m, ts 10p, tn 3, mon 0,
   pf tr_mc)
   end
   ```

   d. Measure the delay on the listed signals.

   ```
   meas delay (cn out, count last rising same,
     pfout dly_rise, pfin tr_mc)
   meas delay (cn out, count last falling same,
     pfout dly_fall, pfin tr_mc)
   ```

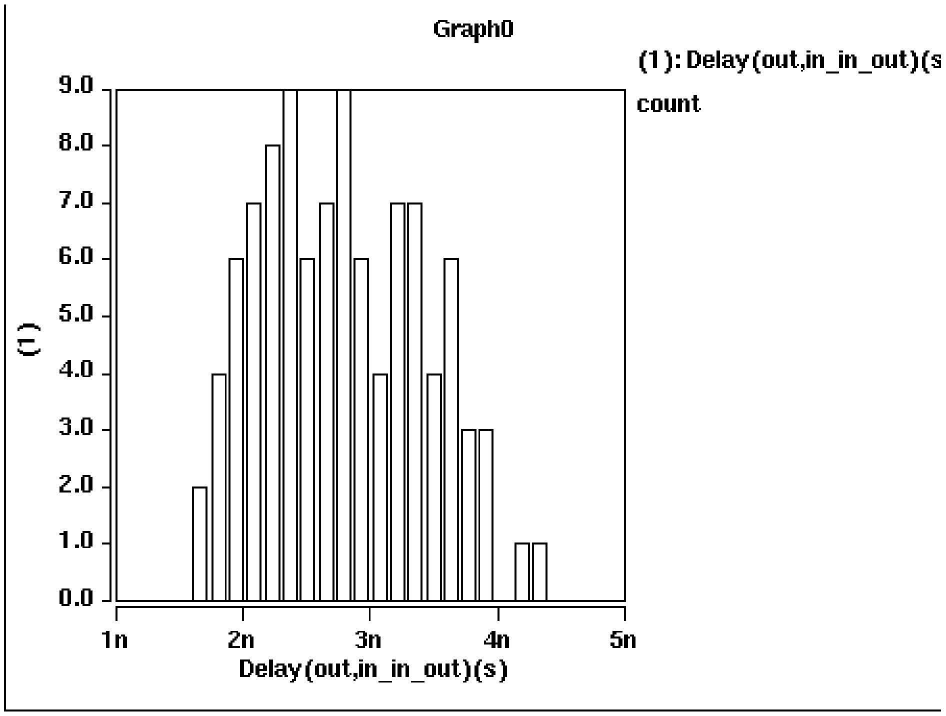
   e. Display as histograms.

   ```
   pfhist (pfin dly_rise, pfout hist_rise)
   pfhist (pfin dly_fall, pfout hist_fall)
   ```

f. Open the plot files.

```
    pl tr_mc dly_rise dly_fall
      hist_rise hist_fall
```

This executes a command script performing a Monte Carlo analysis that takes up to 10 minutes to run. The script measures the delay from the input to output, and generates histograms (labeled `count`), for the delay of the risetime and falltime. (The histogram for the `count` signal in the `tst_or.hist_rise` plot file is shown in the following figure.) This matches the parameters `tplh` and `tphl` on the OR gate in the phase comparator.



Testing the Gate-Level Range Finder Design Example in Saber

The following topics show how to test the design using the behavioral model instead of mapping to MOS Hypermodels, meaning that the digital circuitry will be represented by digital models rather than by MOS gates, as in the last simulation. This difference will result in a much quicker simulation time without a loss in accuracy. Instead of a Saber analog simulation, you will be performing a Saber native mixed-signal simulation.

SaberSketch

- "Viewing the Gate-Level Range Design in SaberSketch" on page 11-20
- "Preparing to Simulate the Gate-Level Range Design in SaberSketch" on page 11-20

Artist

- "Viewing the Gate-Level Range Design in Artist" on page 11-22
- "Preparing to Simulate the Gate-Level Range Design in Artist" on page 11-22

DVE

- "Viewing the Gate-Level Range Design in DVE" on page 11-24
- "Preparing to Simulate the Gate-Level Range Design in DVE" on page 11-25

Viewing the Gate-Level Range Design in SaberSketch

1. To open the schematic `range` start the SaberSketch design editor by typing:

   (UNIX) On a command line, enter *install_home*`/bin/sketch`

   (Windows NT) **Start > Analogy >** *saberdesigner* **> SaberSketch**

   An empty schematic window appears.
2. Click on the **File > Open > Design** items. Click on the **Open** button.
3. Click on the `range.ai_sch` file in the Open Design dialog box, and click on the **Open** button.
4. Use the schematic for the design by selecting the **Design > Use > range** menu item.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in SaberSketch

In the following steps you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. Choose the **Edit > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.

2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.` In the Ground Net Name field, type `vss.`
3. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range.map`.
4. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.map` file.
   d. Select the `range.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
6. To add (or use the one you added in a previous step) the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The File Selection dialog box appears.
   c. Navigate to the `range.shm` file.
   d. Select the `range.shm` file and click the **Open** button in the Select dialog box.
   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

   f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

7. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
8. Close the Saber/Netlister Settings form by clicking on the **Close** button.
9. Start the netlister by selecting the **Design > Netlist range** menu item.
10. Invoke Saber by selecting the **Design > Simulate range** menu item.
11. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate-level design in Saber as described in the topic titled "Simulating the Gate-Level Range Design in Saber" on page 11-26.

Viewing the Gate-Level Range Design in Artist

In your `RangeFinderIC` directory perform the following steps.

1. Invoke `icms`.
2. Start the Artist schematic entry application by choosing the **Open > Design** pulldown menu items.
3. In the Library Name field type `pll_range`.
4. In the Cell Name field type `range`.
5. In the View Name field type `config`. Click on the **OK** button.
6. The Open Configuration or Top CellView dialog box opens.
   a. Click no on the Configuration “pll_range range config” line.
   b. Click yes on the Top Cell View “pll_range range schematic” line.
   c. Click on the **OK** button.
7. Select the **Saber > Set Working Directory** menu item. In the Project Information dialog box, insert your working directory path into the Project Directory field.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in Artist

In the following steps you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd!`. In the Ground Net Name field, type `vss!`.
3. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range.map`.
4. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.map` file.
   d. Select the `range.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
6. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.shm` file.

d. Select the `range.shm` file and click the **Open** button in the Select dialog box.

e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

7. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
8. Close the Saber/Netlister Settings form by clicking on the **Close** button.
9. Start the netlister by selecting the **Saber > Start netlister** menu item.
10. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.
11. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate-level design in Saber as described in the topic titled "Simulating the Gate-Level Range Design in Saber" on page 11-26.

Viewing the Gate-Level Range Design in DVE

In your `RangeFinderIC` directory perform the following steps.

1. Start the Design Viewpoint Editor application by typing:

```
dve range
```

2. Setup Saber by selecting the **Setup > Saber** menu item.
3. Select the **File > Save Design Viewpoint > With Same Name > Cleanup Un-used References** menu item.
4. Click on the **OPEN SHEET** icon to open the schematic.
5. Click on the **ADD PRIM** icon.
6. In the dialog box, add `COMP` to the Name field and click **OK**.
7. Save the viewpoint name as `digital`.
8. Use the pulldown menu **File > Save Design Viewpoint > Save As,** add `digital` to the New Name field.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in DVE

In the following steps you will change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.`In the Ground Net Name field, type `vss.`
3. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range.map`.
4. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.map` file (*path*`/RangeFinderIC/templates/range.map`).
   d. Select the `range.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
5. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
6. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.

   c. Navigate to the `range.shm` file (*path*`/RangeFinderIC/templates/range.shm`).

   d. Select the `range.shm` file and click the **Open** button in the Select dialog box.

   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

   f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

7. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
8. Close the Saber/Netlister Settings form by clicking on the **Close** button.
9. Start the netlister by selecting the **Saber > Netlist > Start Netlister** menu item.
10. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.
11. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate-level design in Saber as described in the topic titled "Simulating the Gate-Level Range Design in Saber" on page 11-26.

Simulating the Gate-Level Range Design in Saber

You will now perform a Saber native mixed-signal simulation on the design.

When SaberGuide appears and the design is loaded, then load the batch command file `range.scs` in the Saber Simulation Transcript window. To load the batch file follow these steps.

1. Select **File > Saber Command File...** to open the Load Command File dialog box and to navigate to the `range.scs` file.
2. Choose the batch file called `range` and press the **Open** button.

   The `range.scs` command file executes the following Saber commands.

   a. Evaluate the DC Operating Point.

   ```
   dc (algst dyn_ramp
   ```

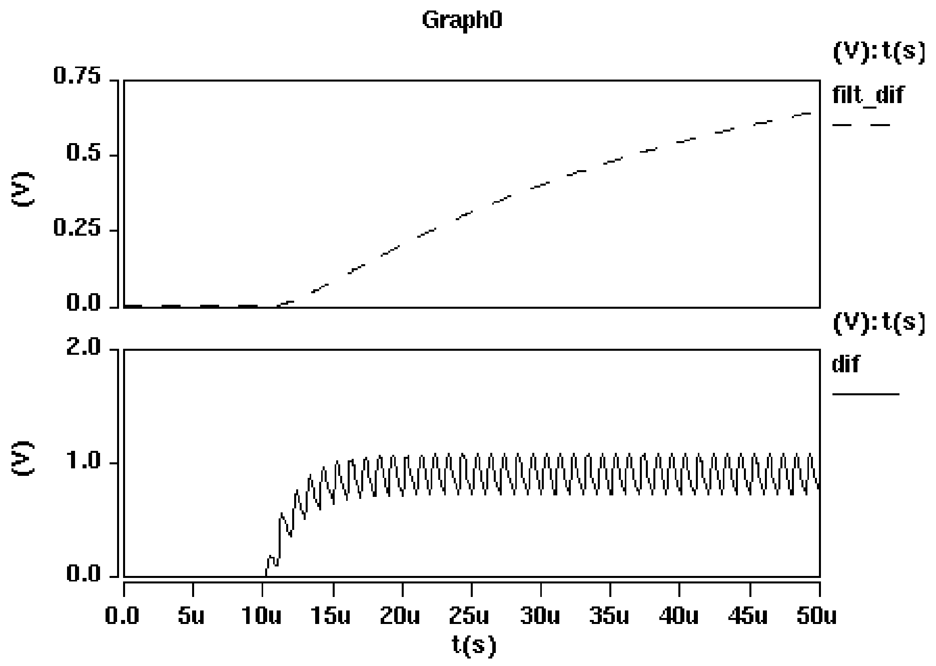
   The `algst dyn_ramp` options set the analysis to use dynamic supply ramping.

b. Determine the Time-Domain (transient) response.

```
tr (te 50u, ts 10p, mon 100, pf tr_mod, df _)
```

This command determines the time domain response of the circuit during the first 50 micro seconds with an initial time step at 10 pico seconds. Analysis information is displayed every 100 time steps.

This executes a command script that performs a transient analysis that takes about 10 seconds to run, 60X faster than the MOS level simulation.



Testing the Gate-Level Range Finder Design Example in Saber-Verilog

The following topics show that digital gates can be simulated in the Verilog-XL simulator from Cadence. A mapping file places digital parts in the schematic into a Verilog netlist. Instead of a Saber native mixed-signal simulation, as in the previous test, you will be performing a Saber/Verilog mixed-signal simulation.

SaberSketch

- "Viewing the Gate-Level Range Design in SaberSketch/Saber-Verilog" on page 11-28
- "Preparing to Simulate the Gate-Level Range Design in SaberSketch/Saber-Verilog" on page 11-28

Artist

- "Viewing the Gate-Level Range Design in Artist/Saber-Verilog" on page 11-30
- "Preparing to Simulate the Gate-Level Range Design in Artist/Saber-Verilog" on page 11-30

DVE

- "Viewing the Gate-Level Range Design in DVE/Saber-Verilog" on page 11-32
- "Preparing to Simulate the Gate-Level Range Design in DVE/Saber-Verilog" on page 11-32

Viewing the Gate-Level Range Design in SaberSketch/Saber-Verilog

1. To open the schematic `range` start the SaberSketch design editor as follows:

   (UNIX) On a command line, enter *install_home*`/bin/sketch`

   (Windows NT) **Start > Analogy >** *saberdesigner* **> SaberSketch**

   An empty schematic window appears.
2. Click on the **File > Open > Design** items. Click on the **Open** button.
3. Click on the `range.ai_sch` file in the Open Design dialog box, and click on the **Open** button.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in SaberSketch/Saber-Verilog

To prepare for simulation, you will invoke Saber Verilog, change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. Choose the **Edit > Saber/Netlister Settings...** menu item to bring up the

Saber/Netlister Settings form.

2. Click on the Co-simulation tab. In the Co-Simulator field, select the **Verilog** button.
3. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.` In the Ground Net Name field, type `vss.`
4. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range_v.map`.
5. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range_v.map` file.
   d. Select the `range_v.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
6. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
7. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.shm` file.
   d. Select the `range.shm` file and click the **Open** button in the Select dialog box.

e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

8. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
9. Close the Saber/Netlister Settings form by clicking on the **Close** button.
10. Start the netlister by selecting the **Design > Netlist range** menu item.
11. Invoke Saber by selecting the **Design > Simulate range** menu item.
12. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate level design in SaberVerilog as described in the topic titled "Simulating the Gate-Level Range Design in Saber-Verilog" on page 11-34.

Viewing the Gate-Level Range Design in Artist/Saber-Verilog

In your `RangeFinderIC` directory perform the following steps.

1. Invoke `icms`.
2. Start the Artist schematic entry application by choosing the **Open > Design** pulldown menu items.
3. In the Library Name field type `pll_range`.
4. In the Cell Name field type `range`.
5. In the View Name field type `config`. Click on the **OK** button.
6. Select the **Saber > Set Working Directory** menu item. In the Project Information dialog box, insert your working directory path into the Project Directory field.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in Artist/Saber-Verilog

To prepare for simulation, you will invoke Saber Verilog, change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister Settings form.
2. Click on the Co-simulation tab. In the Co-Simulator field, select the **Verilog** button.
3. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd!`.In the Ground Net Name field, type `vss!.`
4. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range_v.map`.
5. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:

   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.

   c. Navigate to the `range_v.map` file.

   d. Select the `range_v.map` file and click the **Open** button in the Select dialog box.

   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
6. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
7. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:

   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.

   b. Click on the **Browse** button in the Add Entry dialog box. The File Selection dialog box appears.

   c. Navigate to the `range.shm` file.

   d. Select the `range.shm` file and click the **OK** button in the File Selection dialog box.

   e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

   f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

8. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
9. Close the Saber/Netlister Settings form by clicking on the **Close** button.
10. Start the netlister by selecting the **Saber > Start netlister** menu item.
11. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.
12. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate level design in SaberVerilog as described in the topic titled "Simulating the Gate-Level Range Design in Saber-Verilog" on page 11-34.

Viewing the Gate-Level Range Design in DVE/Saber-Verilog

In your `RangeFinderIC` directory perform the following steps.

1. Start the Design Viewpoint Editor application by typing:

```
dve range/digital
```

2. Setup Saber by selecting the **Setup > Saber** menu item.
3. Select the **File > Save Design Viewpoint > With Same Name > Cleanup Un-used References** menu item.
4. Click on the **OPEN SHEET** icon to open the schematic.

You are now ready to setup the netlister options, as described in the next topic.

Preparing to Simulate the Gate-Level Range Design in DVE/Saber-Verilog

To prepare for simulation, you will invoke Saber Verilog, change the Hypermodel power nodes, load the Hypermodel netlisting file to characterize each digital gate as a MOS device, and open SaberGuide.

1. From the main Frameway session window, choose the **Saber > Saber/Netlister Settings...** menu item to bring up the Saber/Netlister

Settings form.

2. Click on the Co-simulation tab. In the Co-Simulator field, select the **Verilog** button.
3. Click on the Netlister tab followed by the Basic tab. In the Power Net Name field, type `vdd.` In the Ground Net Name field, type `vss.`
4. Click on the Map Files tab.

   The Available listbox displays the pre-defined Mapping Files you can use during simulation. In the next step, you will add a custom Mapping File called `range_v.map`.
5. To add the custom Mapping Files for the Range Finder circuit to the list of Available Mapping Files, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range_v.map` file (*path*`/RangeFinderIC/templates/range_v.map`).
   d. Select the `range_v.map` file and click the **Open** button in the Select dialog box.
   e. Add the Mapping File to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.
   f. Add the Mapping File to the Selected listbox by clicking the **<<>>** button between the listboxes.
6. Click on the Hypermodels tab.

   The Available listbox displays the pre-defined Hypermodels you can use during simulation. In the next step, you will add a custom Hypermodel called `range.shm`.
7. To add the custom Hypermodel for the Range Finder circuit to the list of Available Hypermodels, do the following:
   a. Click on the **Add** button under the Available listbox. The Add Entry dialog box appears.
   b. Click on the **Browse** button in the Add Entry dialog box. The Select dialog box appears.
   c. Navigate to the `range.shm` file (*path*`/RangeFinderIC/templates/range.shm`).

d. Select the `range.shm` file and click the **Open** button in the Select dialog box.

e. Add the Hypermodel to the Available listbox by clicking the **Insert** button in the Add Entry dialog box.

f. Add the Hypermodel to the Selected listbox by clicking the **<<>>** button between the listboxes.

8. In the Saber/Netlister Settings form, click the **Apply** button, then the **Save** button.
9. Close the Saber/Netlister Settings form by clicking on the **Close** button.
10. Start the netlister by selecting the **Saber >Netlist > Start Netlister** menu item.
11. Invoke Saber by selecting the **Saber > Start SaberGuide** menu item.
12. Select the **>cmd** icon to display the SaberGuide Transcript window.

You are now ready to simulate the gate level design in SaberVerilog as described in the topic titled "Simulating the Gate-Level Range Design in Saber-Verilog" on page 11-34.

Simulating the Gate-Level Range Design in Saber-Verilog

You will now perform a Saber/Verilog mixed-signal simulation on the design.

When SaberGuide appears and the design is loaded, then load the batch command file `range_v.scs` in the Saber Simulation Transcript window. To load the batch file follow these steps.

1. Select **File > Saber Command File...** to open the Load Command File dialog box and to navigate to the `range_v.scs` file.
2. Choose the batch file called `range_v` and press the **Open** button.

   The `range_v.scs` command file executes the following Saber commands.

   a. Specify Verilog command line arguments.

   ```
   parg ilch_bhv.v range.v
   ```

b. Evaluate the DC Operating Point, determine the Time-Domain (transient) response, specify the plot file for the results of the analysis, and do not create a data file.

```
dctr (te 50u, ts 10p, mon 400, pf tr_v, df _)
```

This executes a command script that performs a transient analysis that takes about 30 seconds to run. The output voltage, `filt_diff` has not reached its steady state value thought the output of the diffamp, `dif`, has.

In Scope, use the measure command to find the average value of the output. This is faster than running the simulation until the output has settled.

chapter 12

Introduction: Power Converter Design Example

This example shows the design of a two-switch, voltage-mode forward converter using simulation to provide data as an aid in component selection and to verify the accuracy of results generated in earlier steps.

The design process has been broken into four steps, each associated with a circuit that models the power supply at a particular stage of the design:

1. The Power Stage is used to design the major elements of power conversion: the duty cycle, the transformer turns ratio, the output filter, and the switching frequency.
2. The Average Circuit is used to design the converter feedback compensation.
3. The Closed Loop Circuit is used to design the modulation circuitry and to validate the power supply design prior to the last step of the design process.
4. The Final Component Level Design contains all remaining circuit elements—such as snubbers, transistor model switches, and drive circuitry—allowing the complete design to be tested by simulation.

This is one approach to the problems encountered in the design of a power converter. Any given design will vary based on specification requirements and individual practices.

Specification

Output specifications for the power converter:

- Vout: 15VDC
- Vout (ripple): ≤ 0.025V p-p
- Iout: 0.05A to 2A
- Iout (ripple): ≤ 0.1A p-p

- Pout (max) = (15V)(2A) = 30W

Input specifications for the power converter:

- Line input: 150VDC, ± 6V
- $$P_{in}(max) = \frac{P_{out}(max)}{Efficiency} = \frac{30}{0.85} = 35W$$

Other specifications:

- Efficiency ≥ 85%
- Switching Frequency: 200kHz

Copying the Power Converter Design Example

Prior to simulating this design, you must make a local copy of the Power Converter Design Example.

For Windows NT:

1. Invoke Windows NT Explorer and create, if necessary, the directory where you want the files to be copied.
2. Navigate to *saber_home*`\example\SaberSketch\PowerConverter\` and copy all files with the extensions `.ai_sch`, `.ai_dsn`, and `.scs` to your working directory.
3. Verify that your working directory contains the following files:

`f_ol.ai_sch`	`f_ol.ai_dsn`	`f_ol.scs`
`f_avg.ai_sch`	`f_avg.ai_dsn`	`f_avg.scs`
`f_cl.ai_sch`	`f_cl.ai_dsn`	`f_cl.scs`
`f_final.ai_sch`	`f_final.ai_dsn`	`f_final.scs`
`comp10.ai_sch`		
`mod_cm.ai_sch`		

   `.ai_sch` is an extension for SaberSketch schematic files, `.ai_dsn` is an extension for SaberSketch design files, and `.scs` is an extension for Saber command batch files.

4. You must change the file permissions of your local copy of the files using Windows NT Explorer so that they are no longer read-only as follows:

a. Select all the files (**Edit > Select All**).

b. Open the Properties dialog box (**File > Properties**) and select the General tab.

c. Un-check the Read-only box.

d. Click **OK**.

<u>For SaberSketch (in a UNIX environment):</u>

1. Create, if necessary, the directory where you want the files to be copied.
2. Go to *saber_home*/example/SaberSketch/PowerConverter, and copy all files with the extensions `.ai_sch`, `.ai_dsn`, and `.scs` to your working directory.
3. Verify that your working directory contains the following files:

`f_ol.ai_sch`	`f_ol.ai_dsn`	`f_ol.scs`
`f_avg.ai_sch`	`f_avg.ai_dsn`	`f_avg.scs`
`f_cl.ai_sch`	`f_cl.ai_dsn`	`f_cl.scs`
`f_final.ai_sch`	`f_final.ai_dsn`	`f_final.scs`
`comp10.ai_sch`		
`mod_com.ai_sch`		

`.ai_sch` is an extension for SaberSketch schematic files, `.ai_dsn` is an extension for SaberSketch design files, and `.scs` is an extension for Saber command batch files.

<u>For Mentor Graphics:</u>

1. Create, if necessary, the directory where you want the files to be copied.
2. Copy the PowerConverter directory to your current location using Design Manager (`dmgr`).
3. The remainder of this design example refers to SaberSketch as the schematic capture tool. Mentor Graphics users should use Design Architect to open the schematics, netlist the schematics, and invoke SaberGuide.

Each of the Power Converter example circuits comes with a batch (`.scs`) file that runs simulations similar to the ones performed in this example. These batch files can be used as guides to setting up other simulations. For information on running Saber command batch files, see "Running Batch Files" on page A-1.

chapter 13

Power Stage Circuit

Once the topology of a power supply is selected—in this case, a two-switch forward converter—the duty cycle, the transformer turns ratio, and the output filter can be designed. A transient simulation is then performed to verify that the output voltage is correct for a given duty cycle, and that the output ripple voltage and ripple current meet the specification requirements.

The Power Stage Circuit is open loop because feedback is not controlling the switches to correct for variations in the input voltage.

The Power Stage Circuit is described in the following topics:

- "Designing the Duty Cycle and Transformer Turns Ratio" on page 13-1
- "Designing the Output Filter" on page 13-3
- "Verifying the Power Stage Circuit" on page 13-6

Designing the Duty Cycle and Transformer Turns Ratio

1. Define the duty cycle and turns ratio of the transformer.

   In a forward converter, the basic relationship of the output voltage to the input voltage (Vin), duty cycle (D), and turns ratio (n) is

$$V_{out} \approx V_{in} \times \left(\frac{1}{n}\right) \times D$$

   where

   Vout = DC output voltage

   n = turns ratio = np / ns

   D = duty cycle

   In this example, Vout = 15VDC and Vin = 150VDC.

The duty cycle of this forward converter will be designed to be less than 0.5. Choose a value that is between 0 and 0.5. For this example, set D = 0.3.

$$15 \approx 150 \times \left(\frac{1}{n}\right) \times 0.3$$

Solve for n.

$$n \approx 0.3 \times \left(\frac{150}{15}\right)$$

The turns ratio (n) must equal 3.

2. Calculate the maximum, minimum, and nominal duty cycle.

   The duty cycle can be found by the following equation:

$$D = \frac{V_{out} + V_{d2}}{\left(V_{in} \times \frac{1}{n}\right) + \left(V_{d2} - V_{d1}\right)}$$

where V_{d1} is the diode drop during the switch on time and V_{d2} is the diode drop during the switch off time. Assuming they are equal, the equation reduces to

$$D = \frac{V_{out} + V_d}{V_{in} \times \frac{1}{n}}$$

The maximum duty cycle is defined as

$$D_{max} = \frac{V_{out} + V_d}{V_{in(min)} \times \frac{1}{n}}$$

where

n (turns ratio) = 3

$V_{in(min)}$ = 150VDC - 6VDC = 144VDC

V_{out} = 15V

$V_d \approx$ 0.85V

Using these values, the maximum duty cycle is calculated as

$$D_{max} = \frac{15 + 0.85}{144 \times \frac{1}{3}} = 0.3302$$

A value of 0.3302 for Dmax is acceptable because it is less than the maximum duty cycle (0.5) allowed in a forward converter.

The minimum duty cycle is defined as

$$D_{min} = \frac{V_{out} + V_d}{V_{in(max)} \times \frac{1}{n}}$$

where

$V_{in(max)}$ = 156 VDC

Using this value, the minimum duty cycle is calculated as

$$D_{min} = \frac{15.85}{156 \times \frac{1}{3}} = 0.3048$$

The nominal duty cycle is defined as

$$D_{nom} = \frac{V_{out} + V_d}{V_{in(nom)} \times \frac{1}{n}}$$

where

$V_{in(nom)}$ = 150 VDC

Using this value, the nominal duty cycle is calculated as

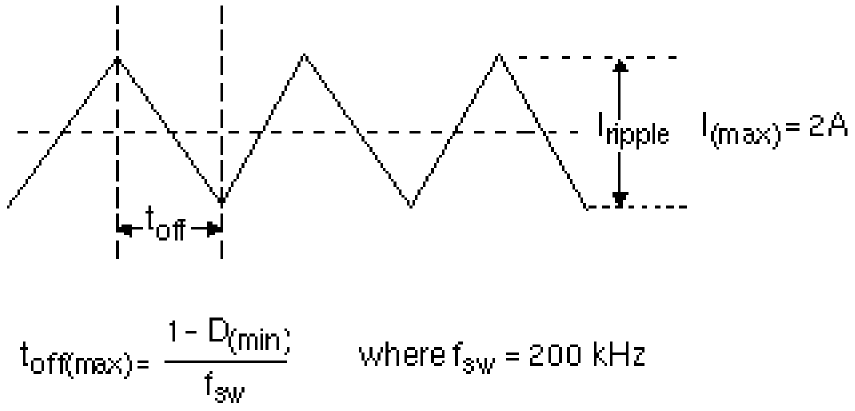
$$D_{nom} = \frac{15.85}{150 \times \frac{1}{3}} = 0.317$$

This value of Dnom is greater than what was selected earlier for D (0.3) because it takes the output diode losses into account.

Designing the Output Filter

Key elements of the output filter include the output inductor, the output capacitor, and the equivalent series resistance (ESR) of the capacitor.

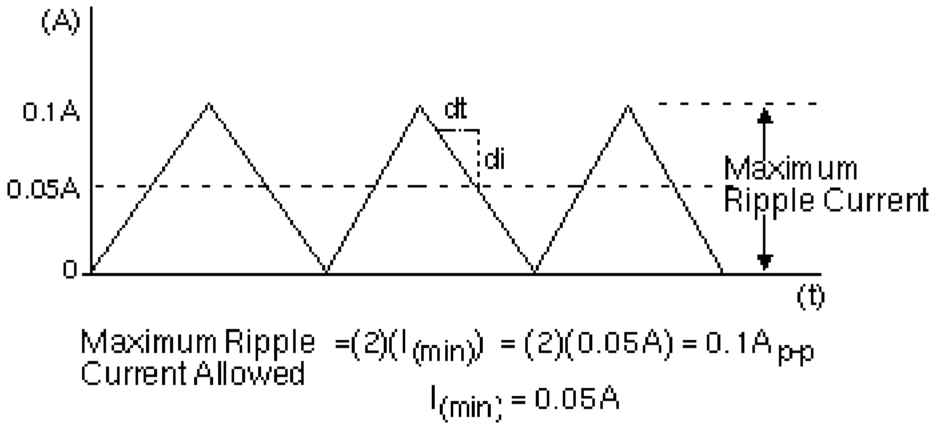
The following figure shows the current waveform through the filter inductor.



Current through the filter inductor

The maximum peak-to-peak current in the inductor is determined by the minimum load current ($I_{(min)}$ = 0.05A). If the load current falls below 0.05A, part of the inductor current goes to zero, putting the power converter into discontinuous mode.

The following figure shows the maximum ripple current.



Maximum ripple current

The inductor's current decreases during the OFF time of the switch. In order to prevent discontinuous operation, the inductor current must not go to zero during this OFF time (at minimum load of 0.05A).

Therefore, the inductor will be sized to limit the peak-to-peak current to 0.1A p-p

1. Design the inductor.

Use the equation:

$$V_L = L_p \times \frac{di}{dt}$$

where

V_L = output voltage = 15V

di = 0.1A

dt = (1 - D_{min})/ f_{sw} = (1 - 0.3048)/200kHz ≈ 3.5 μs

$$15 = L \times \frac{0.1}{3.5\mu}$$

Therefore,

$$L = 15 \times \frac{3.5\mu}{0.1} \approx 0.53\,mH$$

L = 0.53mH

2. Design the capacitor.

The $V_{out(ripple)}$ specification, along with the calculated I_{ripple} coming through the inductor, determines the size of the output capacitor.

The following is used to calculate the capacitor value:

$$C = \frac{\frac{1}{8}(I_{ripple})}{f \times V_{ripple}}$$

where

I_{ripple} = 0.1A

f = 200kHz

V_{ripple} = 0.025V

$$C = \frac{\frac{1}{8}(0.1)}{200K \times 0.025} = 2.5\mu F$$

Calculate the ESR of the capacitor:

ESR_{max} = ΔV / ΔI = 0.025 / 0.1 = 0.25Ω

The ESR of the capacitor must not exceed 0.25Ω, or the ripple voltage will increase beyond specifications.

Verifying the Power Stage Circuit

Step 1. Open the schematic.

1. Invoke SaberSketch.
2. Open the design (**File > Open > Design**).

   Navigate to the design in the Open Design dialog box. Select `f_ol`.

Step 2. Invoke the SaberGuide icon bar.

Click on the **Show/Hide SaberGuide** icon.

Step 3. Invoke the SaberGuide Transcript window.

Click on the **>cmd** button on the SaberGuide icon bar.

Step 4. Load the f_ol netlist file into SaberGuide.

Choose the **Design>Simulate f_ol** menu item to generate a netlist file and load it into SaberGuide.

The remaining steps analyze the transient response of the Power Stage Circuit to examine key performance results. The simulations in this section can also be run using batch files (see the Appendix, Running Batch Files).

Step 5. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient**).
2. Edit the following fields in the transient analysis form:

   Basic tab

   End Time: `500u`

   Time Step: `1.1u`

   Monitor Progress: `100`

   Run DC Analysis First: `No`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: `tr`

   Data File:_

Initial Point File: `zero`

End Point File: `tr`

Calibration tab

Max Truncation Error: `1u`

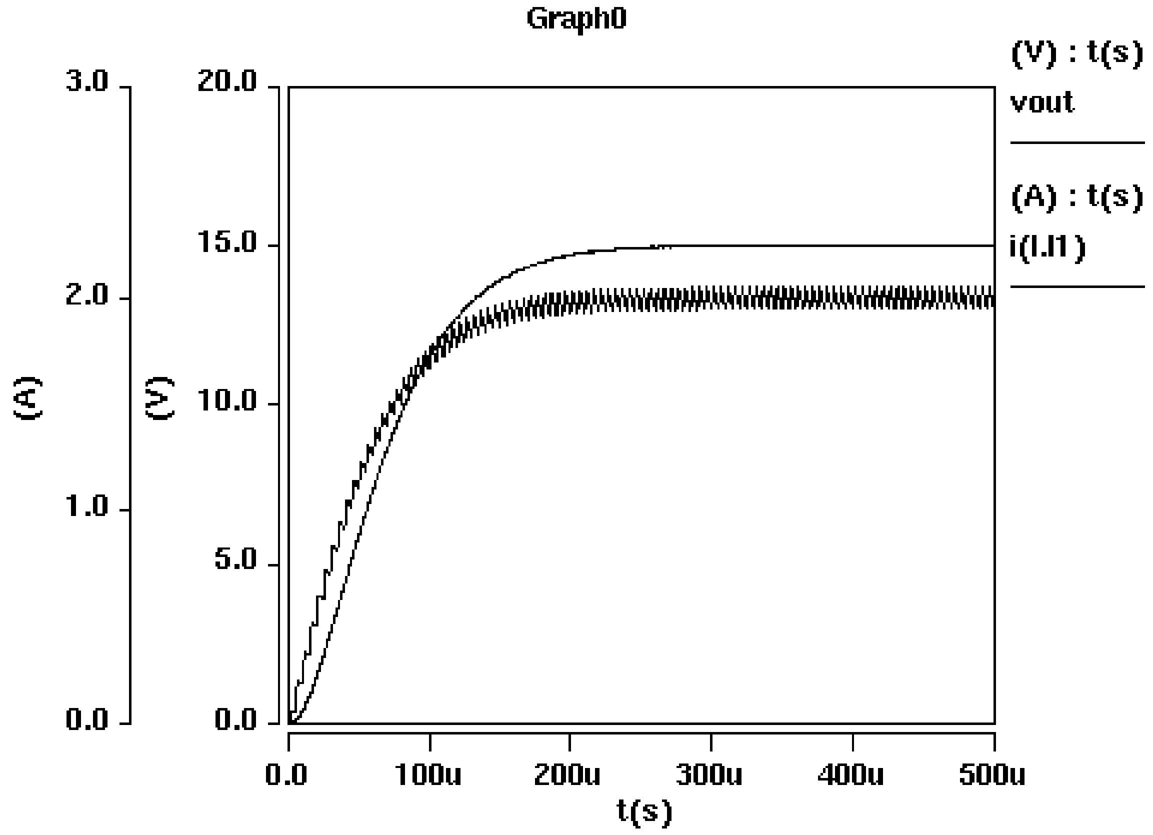
Sample Point Density: `1k`

3. Perform the analysis by clicking the **OK** button.

   This command simulates the transient response over the first 500us of operation and saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr`. Every 100th data point will be displayed in the SaberGuide transcript window.

4. In SaberScope, plot the output inductor current, `i(l.l1)`, and the output voltage, `vout`. The output voltage should ramp to 15 volts and have a ripple voltage of approximately 25mVp-p, and the inductor current should ramp to 2 amps and have a ripple current of approximately 100mAp-p.

These results should match the following figure:



Output Voltage and Inductor Current at Startup

The next step in the design process is to analyze the Average Circuit

chapter 14

Average Circuit

The second step in the process is to design the converter feedback compensation. The ideal switches used in the Power Stage Circuit are replaced, in the Average Circuit, with an average model of the two-switch forward converter. By eliminating the non-linearities associated with the switches, the average model allows small signal frequency analysis to be performed on the circuit. This is the equivalent of performing state-space averaging on the circuit. When used in transient analysis, the average model reduces simulation times to a few seconds.

The first task in this step is to design the control voltage. With this information, a transient analysis can be run to verify the correct operation of the Average Circuit in an open loop configuration.

In order to design the feedback compensation, the control to output transfer function must be determined by running a frequency analysis on the Average Circuit, still in an open loop configuration.

After the feedback compensation is designed, a frequency analysis is run to verify that the converter has been properly compensated, and a transient analysis is run to verify that closing the loop on the Average Circuit yields the expected system performance.

This chapter is divided into the following topics:

Calculating the Control Voltage

To determine the control voltage, use the following control to output relationship for the forward converter:

$$V_{out} = \left(V_{in} \times \frac{1}{n} \times \frac{V_c}{V_{ramp}}\right) - V_d$$

where

Vout = 15V

Vin = 150V

n = 3

Vramp = 2.5V

Vd = 0.85V

Rearrange the equation to determine the control voltage:

$$V_c = V_{ramp} \times n \times \frac{V_{out} + V_d}{V_{in}} = 2.5 \times 3 \times \frac{15 + 0.85}{150} = 0.7925$$

Verifying the Average Circuit

Step 1. Open the schematic.

1. Invoke SaberSketch.
2. Open the design (**File > Open > Design**).

   Navigate to the design in the Open Design dialog box. Select `f_avg`.

Step 2. Invoke the SaberGuide icon bar.

Click on the **Show/Hide SaberGuide** icon.

Step 3. Invoke the SaberGuide Transcript window.

Click on the **>cmd** button on the SaberGuide icon bar.

Step 4. Load the f_avg netlist file into SaberGuide.

Choose the **Design > Simulate f_avg** menu item to generate a netlist file and load it into SaberGuide. If necessary, select **Design > Use > f_avg** first.

Step 5. Change the input source to the average model.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit > List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **Edit** button. This displays the Alter Components form.
4. Enter `input=use1` in the Value field, and click the **OK** button to change the netlist. This connects the Control Voltage to the average model through the breakpoint switch.

The remaining steps analyze the transient and frequency domain response of the Average Circuit by using transient and AC analyses to examine key performance results. The simulations in this section can also be run using batch files (see the Appendix, Running Batch Files).

The first open loop transient simulation is run to validate that the average model is providing the correct output voltage for a given control and input voltage. This transient simulation can also be used to set up the operating point for the small signal ac simulation generating the control to output transfer function.

Step 6. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the transient analysis form:

   Basic tab

   End Time: `500u`

   Time Step: `1u`

   Monitor Progress: `10`

   Run DC Analysis First: `No`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: `tr1`

Data File: _

Initial Point File: `zero`

End Point File: `tr1`

Calibration tab

Max Truncation Error: `1u`

Sample Point Density: `1k`

3. Perform the analysis by clicking the **OK** button.

   This command simulates the transient response over the first 500us of operation, saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr1`, and displays every 10th data point in the SaberGuide transcript window.

4. In SaberScope, plot the output inductor current, `i(l.l1)`, and the output voltage, `vout`. Note that, because the average model eliminates switching effects, the switching component of the waveforms is gone.

5. Keep the above plots and add signals `i(l.l1)` and `vout` from the `f_ol.tr` plot file. The results should overlap, demonstrating that the average model is producing the same results as the switching model used by the Power Stage Circuit.

Determining the Control to Output Transfer Function

The next simulation, a small signal ac analysis, is performed to evaluate the control to output transfer function. The results are used to design the error amplifier compensation circuit.

Step 1. Change the input source to the average model.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit>List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **Edit** button to display the Alter Components form.
4. Enter `input=use2` in the Value field, and click the **OK** button to change the netlist. This connects the AC Voltage source to the average model through the breakpoint switch.

Step 2. Analyze the frequency response.

1. Display the Small-Signal Frequency Analysis form (**Analyses > Frequency > Small-Signal AC...**).
2. Edit the following fields in the AC analysis form:

   Basic tab

   Start Frequency: `0.1`

   End Frequency: `100meg`

   Monitor Progress: `10`

   Sample Point Density: `128`

   Number of Points: `1000`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: ac

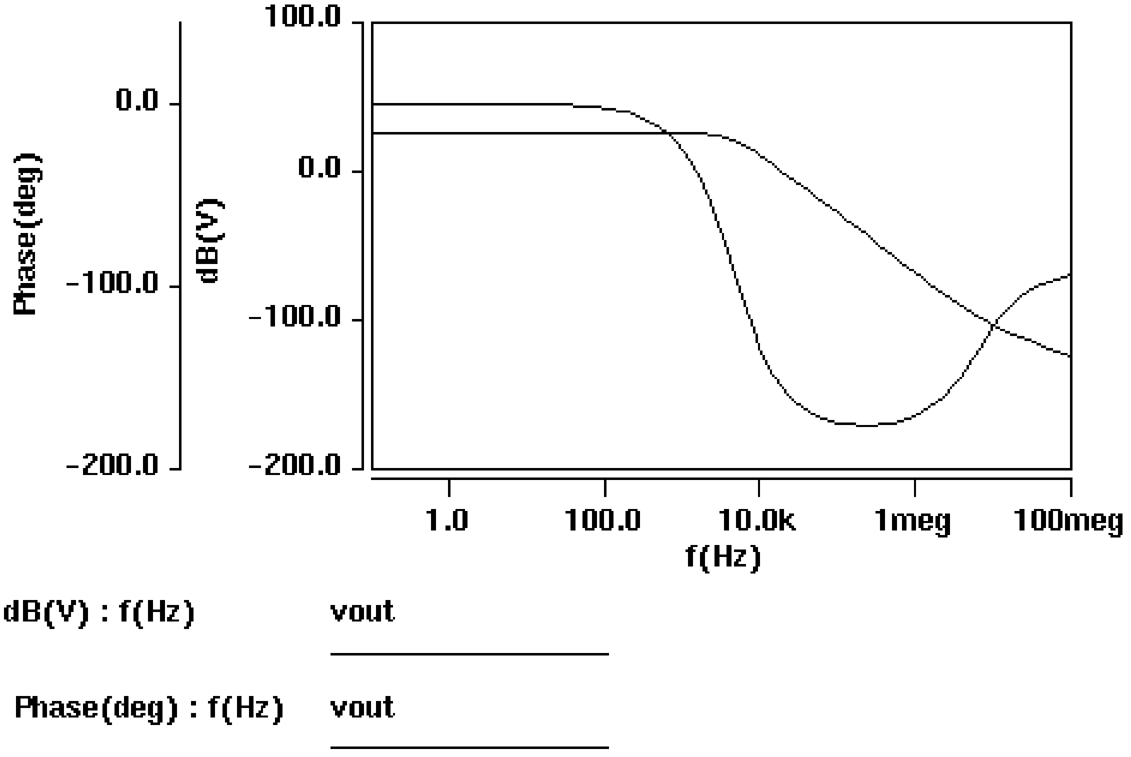
   Data File:_

   Initial Point File: `tr1`

   Entering the `tr1` file into the Initial Point File field sets the end point from the previous transient analysis as the operating point for this ac analysis.
3. Perform the analysis by clicking the **OK** button.

   The result is the frequency response from 0.1 to 100MHz. The analysis uses 1000 logarithmically-spaced data points and saves the resulting waveforms for each signal on the root of the design in a Plot File called `f_avg.ac`.

4. In SaberScope, plot the gain and phase of the output voltage `vout` from the `f_avg.ac` plot file to give the control to output transfer function. These results should match the following figure.



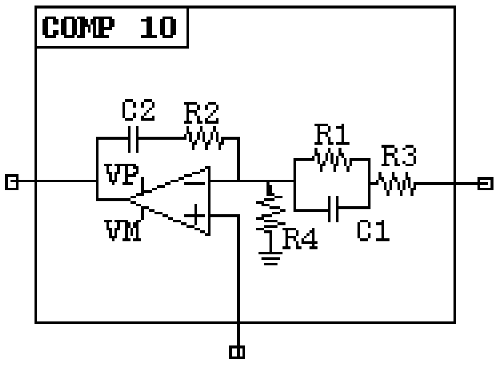
Control to Output Transfer Function

This information is used to design the feedback compensation.

Designing the Feedback Compensation

The compensator will need two zeros to cancel out the effects of the two poles. The frequency of these zeros will be one-half the resonant frequency of the filter. The compensation network itself is an integrator, so it adds another pole at the origin. Another pole will be at one-quarter the switching frequency, which cancels the effects of the ESR of the capacitor (zero). The overall response yields a single-pole roll-off at the crossover frequency.

The following figure shows the compensation network used in this design, the hierarchical COMP 10 compensator.



Type 10 compensation network

C1 and R1 determine a zero (f_{z1}), C2 and R2 determine a zero (f_{z2}), and C1 and R3 determine a pole (f_{p2}). Note that f_{p1} is approximately 0 Hz because this compensator is an integrator.

Calculate values for the compensation components.

1. Using the resonant frequency of the output filter, f_{res}, find the desired zero frequencies (f_{z1}, f_{z2}) for the compensator.

$$f_{res} = \frac{0.159}{\sqrt{LC}} = \frac{0.159}{\sqrt{0.53m \times 2.5\mu}} = 4.3kHz$$

$$f_{z1} = f_{z2} = (0.5)\, f_{res}$$

Therefore, f_{z1} and f_{z2} = (0.5)(4.3kHz) = 2.13 kHz.

2. Using the switching frequency (f_{sw}) of the supply, which has been specified as 200 kHz, find the desired pole frequency (f_{p2}) for the compensator.

$$f_{p2} = (0.25) f_{sw} = (0.25)(200kHz) = 50kHz$$

3. Calculate the values for R2 and R3 such that the high-frequency gain at the desired crossover (50 kHz) is 0 dB.

The phase and gain plot generated by the ac analysis performed in "Determining the Control to Output Transfer Function" on page 14-4 shows that a crossover (0 dB) at 50 kHz requires 16.37 dB of additional gain from the error amplifier. Another 3 dB of gain is required because of the pole, fp2, at 50 kHz.

R2 / R3 = (16.37 + 3) dB = 19.37 dB

19.37 dB = $\log^{-1}$ (19.37 / 20) = 9.3

Setting R2 = 50k gives

$$R3 = \frac{50k}{9.3} = 5.38k$$

The gain required at fz1 and fz2 is

$$Av_{(2.15kHz)} = \frac{2.15K}{50K} \times Av_{(50kHz)}$$

Av(50kHz) = 9.3, therefore,

$$Av_{(2.15kHz)} = \frac{2.15K}{50K} \times 9.3 = 0.4$$

The gain at 2.15 kHz is determined by R2 / (R + R1).

$$\frac{50k}{5.38k + R1} = 0.4$$

Solving for R1,

$$R1 = \frac{50k}{0.4} - 5.38k = 119.62k$$

4. Calculate the capacitor values.

$$f_{z1} = f_{z2} = \frac{1}{2\pi R_1 C_1} = \frac{1}{2\pi R_2 C_2} = 2.15kHz$$

Therefore,

$$C_1 = \frac{0.159}{(R_1)(f_{z1})} = \frac{0.159}{(119.62k)(2.15k)} = 618pF$$

$$C_2 = \frac{0.159}{(R_2)(f_{z2})} = \frac{0.159}{(50k)(2.15k)} = 1479pF$$

5. Calculate the value of R4, which provides a voltage divider for the 15V output. The reference voltage used is 5V, which means the value of R4 must be specified so that the 15V output is divided down to 5V:

   $$15\left(\frac{R4}{R4 + (R3 + R1)}\right) = 5$$

   Solving for R4 produces R4 = 62.5k.

   Here are the final values for the compensator:

   R1 = 119.62kΩ
   R2 = 50kΩ
   R3 = 5.38kΩ
   R4 = 62.5kΩ
   C1 = 618pF
   C2 = 1479pF

Although this example uses an average model that operates only in the continuous conduction mode (CCM), average models that operate in both CCM and discontinuous conduction mode (DCM) are also available. Of these, the average model that corresponds to the model used in this example is **frwdavg**.

Verifying the Feedback Compensation Frequency Response

Before running a small signal ac simulation to verify the loop response of the converter, a transient analysis is run. This accomplishes two tasks. It creates an initial point file for the ac simulation in this section, and it provides time-domain data for the next section, "Verifying the System Parameters" on page 14-12.

Step 1. Change the input source to the average model.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit>List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **Edit** button to display the Alter Components form.
4. Enter `input=use3` in the Value field, and click the **OK** button to change the netlist. This closes the loop around the circuit through the breakpoint switch.

Step 2. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the transient analysis form:

   Basic tab

   End Time: `500u`

   Time Step: `1u`

   Monitor Progress: `10`

   Run DC Analysis First: `Yes`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: `tr2`

   Data File: _

   Initial Point File: `dc`

   End Point File: `tr2`

   Calibration tab

   Max Truncation Error: `10u`

   Sample Point Density: `128`
3. Perform the analysis by clicking the **OK** button.

   The resulting waveforms for each signal are saved on the root of the design in a Plot File called `tr2`. This file is used as the Initial Point File in the following ac analysis.

Step 3. Change the input source to the average model.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit > List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **Edit** button. The Alter Components form appears.
4. Enter `input=use2` in the Value field, and click the **OK** button to change the netlist. This connects the AC Voltage source to the average model through the breakpoint switch.

Step 4. Analyze the frequency response.

1. Display the Small-Signal Frequency Analysis form (**Analyses > Frequency > Small-Signal AC...**).
2. Edit the following fields in the AC analysis form:

   Basic tab

   Start Frequency: `0.1`

   End Frequency: `100meg`

   Monitor Progress: `10`

   Sample Point Density: `128`

   Number of Points: `1000`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

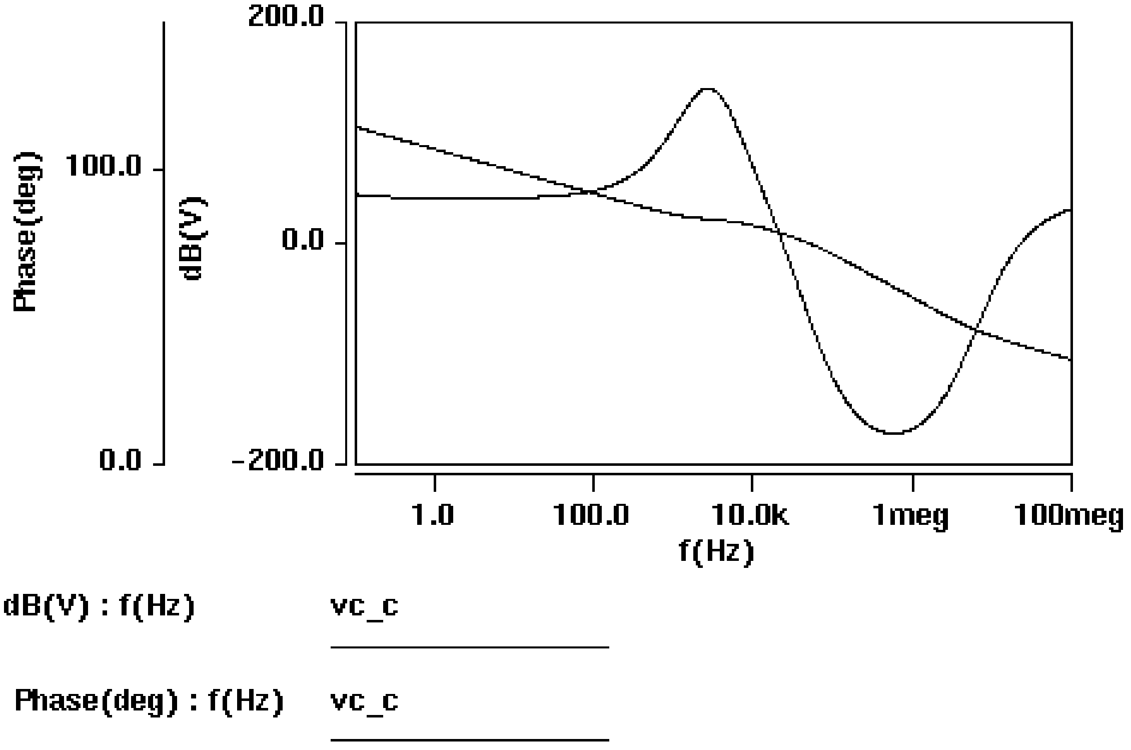
   Plot File: `ac2`

   Data File:_

   Initial Point File: `tr2`
3. Perform the analysis by clicking the **OK** button.

   The result is the frequency response from 0.1 to 100MHz. The analysis uses 1000 logarithmically-spaced data points and saves the resulting waveforms for each signal on the root of the design in a Plot File called `f_avg.ac2`.
4. Using SaberScope to plot the gain and phase of the output voltage `vc_c` from plot file `f_avg.ac2` gives the response, which is very

close to a single pole roll-off with a phase margin of approximately 50 degrees. The following figure shows Loop Response.



Loop Response

Verifying the System Parameters

1. Use the data generated during the transient analysis performed in "Verifying the Feedback Compensation Frequency Response" on page 14-9 to verify that the closed loop circuit yields the expected output voltage, control voltage, and duty cycle.
2. In SaberScope, plot the control voltage, `vc_c`, from `f_avg.tr2`, and confirm that it is approximately 0.7925.
3. On another graph, plot the output voltage, `vout`, from `f_avg.tr2`, and confirm that it is approximately 15 volts.
4. On a third graph, plot the duty cycle, `d(pwm_frwd_cvm.avg1)`, from `f_avg.tr2`, and confirm that it is approximately 0.317.

chapter 15

Closed Loop Circuit

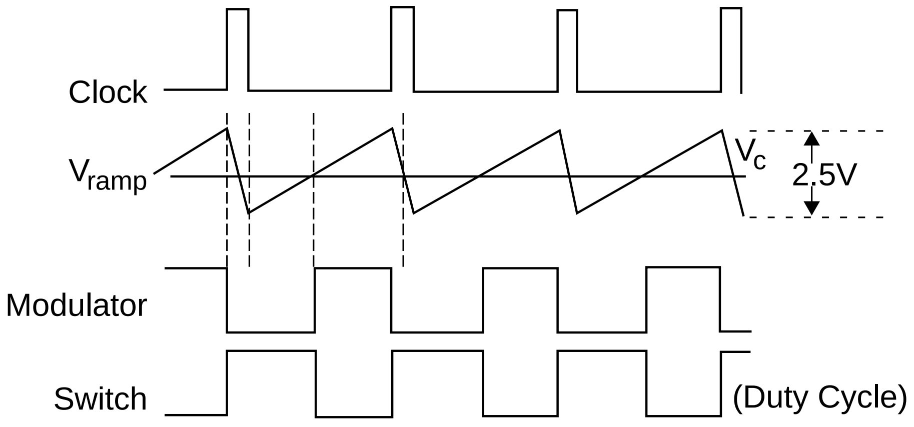
The third step in the process is to design the modulation circuitry. The modulation circuitry takes the control voltage from the output of the error amplifier and converts it into a switch signal to turn the MOSFET switches on and off.

The Closed Loop circuit is used for two simulations: an open-loop simulation to validate the modulation circuitry and a complete closed-loop simulation to validate the design to this point as shown in the following topics:

- "Designing the Modulation Circuitry" on page 15-2
- "Verifying the Modulation Circuitry" on page 15-3
- "Verifying the Closed Feedback Loop Transient Response" on page 15-5

Designing the Modulation Circuitry

As seen in the following figure, when the clock pulse goes high, the switch turns on, and when Vramp crosses the control voltage (Vc), the switch turns off.



Duty Cycle, V_{ramp} and V_c Relationship

The duty cycle (D) is related to the control voltage (Vc) and the ramp (Vramp) by the following equation:

$$D = \frac{V_c}{V_{ramp}}$$

D = 0.317, calculated in "Designing the Duty Cycle and Transformer Turns Ratio" on page 13-1

Set Vramp = 2.5V

$$V_c = 2.5V \times 0.317 = 0.7925V$$

Verifying the Modulation Circuitry

To validate the modulation circuitry, perform an open-loop simulation by using the control voltage as the input to the modulator. Note that the breakpoint model used in the schematic is the same model used in the Average Circuit, allowing several types of simulations to be run from a single schematic. For the validation of the modulation circuitry, the breakpoint model opens the feedback loop and uses the control voltage as the input to the modulation circuit. The results of this simulation show the relationship of the output voltage with respect to the control voltage for a specific modulation circuit design. Note that the control voltage (0.7925) yields the correct output voltage and duty cycle.

Step 1. Open the schematic.

1. Invoke SaberSketch.
2. Open the design (**File > Open > Design**)

   Navigate to the design in the Open Design dialog box and select `f_cl`.

Step 2. Invoke the SaberGuide icon bar.

Click on the **Show/Hide SaberGuide** icon.

Step 3. Invoke the SaberGuide Transcript window.

Click on the **>cmd** button on the SaberGuide icon bar.

Step 4. Load the f_cl netlist file into SaberGuide.

Choose the **Design>Simulate f_cl** menu item to generate a netlist file and load it into SaberGuide. If necessary, select **Design>Use>f_cl** first.

Step 5. Set the modulator input voltage to 0.7925 volts.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit>List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **Edit** button. This displays the Alter Components form.
4. Enter `input=use2` in the Value field, and click the **OK** button to change the netlist. This connects the control voltage to the modulation circuitry, as well as opening the feedback loop.

The remaining steps analyze the transient response of the circuit, open loop, to examine key performance results. The simulations in this section can also be run using batch files (see the Appendix, Running Batch Files).

Note that the Signal List field in the following transient analysis should be left at its default setting of All Toplevel Signals, signified by the slash symbol (/), to reduce the size of the files generated by the simulation.

Step 6. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the transient analysis form:

   Basic tab

   End Time: `3m`

   Time Step: `0.3u`

   Monitor Progress: `100`

   Run DC Analysis First: `No`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: `trvc`

   Data File: `_`

   Initial Point File: `zero`

   End Point File: `trvc`

   Calibration tab

   Max Truncation Error: `1u`

   Sample Point Density: `1k`
3. Perform the analysis by clicking the **OK** button.

   This command simulates the transient response over the first 3 ms of operation and saves the resulting waveforms for each signal on the root of the design in a Plot File called `trvc`. Every 100th data point will be displayed in the SaberGuide transcript window.

   This simulation, an open loop configuration that uses the same control voltage used in the Average Circuit, confirms that, given the control voltage input into the modulation circuit, the switches will operate at the correct duty cycle to produce a 15 volt output.

4. Using SaberScope, plot the `vout` signal from `f_cl.trvc`. Note that in this simulation, the control voltage (.7925) is being fed into the modulation circuit to verify that the correct duty cycle and output voltage are obtained.
5. Plot the ramp voltage, `n#54`, the control voltage, `vc_c2`, the clock pulse `v_clk`, and the switch gate drive, `switch`. These should look like the waveforms in the figure, Duty Cycle, Vramp, and Vc Relationship in "Designing the Modulation Circuitry" on page 15-2.

Verifying the Closed Feedback Loop Transient Response

Step 1. Close the feedback loop.

1. In the SaberGuide Transcript window, display the List/Alter Design dialog box (**Edit>List/Alter...**).
2. On the Netlist tab, select `switch_vin_breakpt1` from the Hierarchical Instance List.
3. Click the **OK** button. This displays the Alter Components form.
4. Enter `input=use3` in the Value field, and click the **OK** button to change the netlist.

   Note that the Signal List field in the following transient analysis should be left at its default setting of All Toplevel Signals, signified by the slash symbol (/), to reduce the size of the files generated by the simulation.

Step 2. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).
2. Edit the following fields in the transient analysis form:

   Basic tab

      End Time: `3m`

      Time Step: `0.3u`

      Monitor Progress: `100`

      Run DC Analysis First: `No`

      Plot after analysis: `Yes - Open Only`

   Input/Output tab

Plot File: `trcl`

Data File: _

Initial Point File: `zero`

End Point File: `trcl`

Calibration tab

Max Truncation Error: 1u

Sample Point Density: 128

3. Perform the analysis by clicking the **OK** button.

   The resulting waveforms for each signal are saved on the root of the design in a Plot File called `trcl`.

   This simulation validates the closed loop response of the switching circuit. The breakpoint model is used to close the feedback loop and take the control voltage source out of the circuit.

4. In SaberScope, plot the output inductor current, `i(l.l1)`, and the output voltage, `vout`, from `f_cl.trcl`. Verify that `vout` is 15 volts and that `i(l.l1)`is 2A. Note that the output voltage has a small amount of overshoot, which is the result of the closed loop response. Because the simulation in the previous section was performed on an open loop circuit, no overshoot appears on the output waveforms from the `f_cl.trvc` plot file.

   Zoom in on the waveforms and check that the voltage ripple is 25mV and the current ripple is 100mA.

   Note that the duty cycle settles out at approximately 0.317, which is the value calculated earlier.

chapter 16

Final Component Level Design

The final design configuration differs from the closed loop circuit by the addition of the following:

- the 1825 PWM model, connected in the voltage mode, that also replaces the modulation circuitry and the error amplifier used in the Closed Loop Circuit
- snubber networks across the switching devices
- the IRF250 (200 volt) MOSFET model to replace the ideal switches
- drive circuitry to turn the power MOSFET devices on and off
- a current transformer to switch the power MOSFETs' drive circuitry and to provide primary-to-secondary DC isolation

A transient analysis on this design verifies that the converter performs as expected as shown in the topic "Verifying the Final Component Level Design" on page 16-1.

Verifying the Final Component Level Design

Step 1. Open the schematic.

1. Invoke SaberSketch.
2. Open the design (**File > Open > Design**).

   Navigate to the design in the Open Design dialog box and select `f_final`.

Step 2. Invoke the SaberGuide icon bar.

Click on the **Show/Hide SaberGuide** icon.

Step 3. Invoke the SaberGuide Transcript window.

Click on the **>cmd** button on the SaberGuide icon bar.

Step 4. Load the f_final netlist file into SaberGuide.

Choose the **Design>Simulate f_final** menu item to generate a netlist file and load it into SaberGuide. If necessary, select **Design>Use>f_final** first.

The simulations in this section can also be run using batch files (see the Appendix, Running Batch Files).

Step 5. Determine the time-domain (transient) response.

1. Display the Time-Domain Transient Analysis form (**Analyses > Time-Domain > Transient...**).

   Note that the Signal List field in the Time-Domain Transient Analysis form should be left at its default setting of All Toplevel Signals, signified by the slash symbol (/), to reduce the size of the files generated by the simulation.

2. Edit the following fields in the transient analysis form:

   Basic tab

   End Time: `2.003m`

   Time Step: `1u`

   Monitor Progress: `100`

   Run DC Analysis First: `No`

   Plot after analysis: `Yes - Open Only`

   Input/Output tab

   Plot File: `tr1`

   Data File: `_`

   Initial Point File: `dc`

   End Point File: `tr1`

   Calibration tab

   Max Truncation Error: `10u`

   Sample Point Density: `1k`

3. Perform the analysis by clicking the **OK** button.

   This command simulates the transient response over the first

2.003ms of operation, saves the resulting waveforms for each signal on the root of the design in a Plot File called `tr1`, and displays every 100th data point of the simulation to the SaberGuide transcript window.

Step 6. In SaberScope, plot the output inductor current, `i(l.l1)`**, and the output voltage,** `vout`**.**

The output voltage ramps up to 15 volts with a ripple voltage of approximately 25mVp-p, and the inductor current ramps up to 2 amps with a ripple current of approximately 100mAp-p per the design specifications. The switching frequency is 200kHz.

Note that in the final design we see a similar overshoot to the one in the previous simulation. The differences are due to the added transformer in the feedback, the use of the uc1825, and the MOSFET switches/drivers.

appendix A

Running Batch Files

Batch files (files with an `.scs` extension) are also known as command files and Saber Command Script files. These files contain Saber commands that can perform tasks normally requiring a user's active input, such as setting up and running simulations. Batch files can save the user from having to perform repetitive tasks like changing simulation parameters before the start of multiple simulations. They are also a good way of keeping a record of the types of simulations performed on a circuit.

Many of the design example circuits come with a batch file that runs simulations similar to those performed in the *Design Example* manual. These batch files can be used as guides to setting up other simulations.

Running a Batch File:

Once the netlist file has been opened, load the batch file:

1. From the SaberGuide Transcript window, invoke the Load Command File dialog box (**File > Saber Command File**).
2. Go to the directory containing the Saber Command Script (`.scs`), or batch, file you want to load, select the file, and click the **Open** button. This runs the batch file.
3. Invoke SaberScope to view the simulation results.

Creating a Batch File:

1. In a text file, type the Saber command-line commands you want to run on your circuit, one to a line.
2. Save this text file, giving it an extension of `.scs`.