Virtuoso Functional Block Library Reference Manual

Product Version ICADVM20.1
October 2020

Contents

Preface

The Functional Block library (`functional`) is a library within the Virtuoso analog design enviroment and is used by analog designers as a high-level design tool.

This manual contains information about all the components in the Functional Block library. This manual assumes that you are familiar with the development and design of integrated circuits. This manual is for analog circuit designers who are familiar with modeling and the Virtuoso analog design and simulation environment.

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documentation
- *Component Description Format User GuideAdditional Learning Resources*
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node and methodologies (for example, ICADVM18.1) releases.

Label	Meaning
(ICADVM18.1 Only)	Features supported only in the ICADVM18.1 advanced nodes and advanced methodologies release.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

For information about licensing in the Virtuoso design environment, see *Virtuoso Software Licensing and Configuration Guide*.

Related Documentation

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide*
- *Component Description Format User Guide*
- *Cadence Application Infrastructure User Guide*

Virtuoso Tools

- *Virtuoso Schematic Editor User Guide*
- *Virtuoso Analog Design Environment L User Guide*
- *Virtuoso Analog Design Environment XL User Guide*
- *Virtuoso Analog Design Environment GXL User Guide*

Component Description Format User Guide

Additional Learning Resources

Video Library

The Video Library on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see <u>Virtuoso Videos</u>.

Rapid Adoption Kits

Cadence provides a number of <u>Rapid Adoption Kits</u> that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

To explore the full range of training courses provided by Cadence in your region, visit <u>Cadence Training</u> or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
|	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	
	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument.
	Used with brackets to indicate that you can specify zero or more arguments.
	Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

1

About the Functional Block Library

The functional block library is a library within the Cadence® analog design enviroment and is used by analog designers as a high-level design tool.

The blocks contain the essential functionality of the parts they are modelling. However, the models are not fully representative of the equivalent parts. For example, an amplifier functional block has no differential pair on its input, so input biases such as CMRR (common mode rejection ratio) are not accounted for. Also, a majority of these blocks is assumed to be ideal. The amplifier amplifies with no drift, the input impedance is infinite, and the output impedance is zero. However, you can use the functional block library with all other installed analog libraries and with basic SPICE primitives so you can add complexity and effects, such as frequency response.

Each functional block consists of at least two parts: the symbol and its associated model. The models directory must be in the model path of the Environment Options form. Also, the library directory must be present in the search path of the Set Library Search Path form.

For *spectre* simulation, the models directory path is

`your_install_dir/tools/dfII/etc/cdslib/artist/functional/allFunc.scs`

Choosing *Setup – Environment* from the simulation window opens the Environment Options form. This form contains a field where you can specify the model path.

The library directory path is

`your_install_dir/tools/dfII/etc/cdslib/artist`

Choosing *Design Manager – Search Path* from the Command Interpreter Window (CIW) opens the Set Library Search Path form.

Functional Block Library Components

The functional block library contains several components. The components in functional block library are divided into four categories. The categories are: Amplifiers, Math, Misc, and Pole.

For each component, multiple views, such as symbol view and simulator specific views are available. For some components, the schematic view is also available.

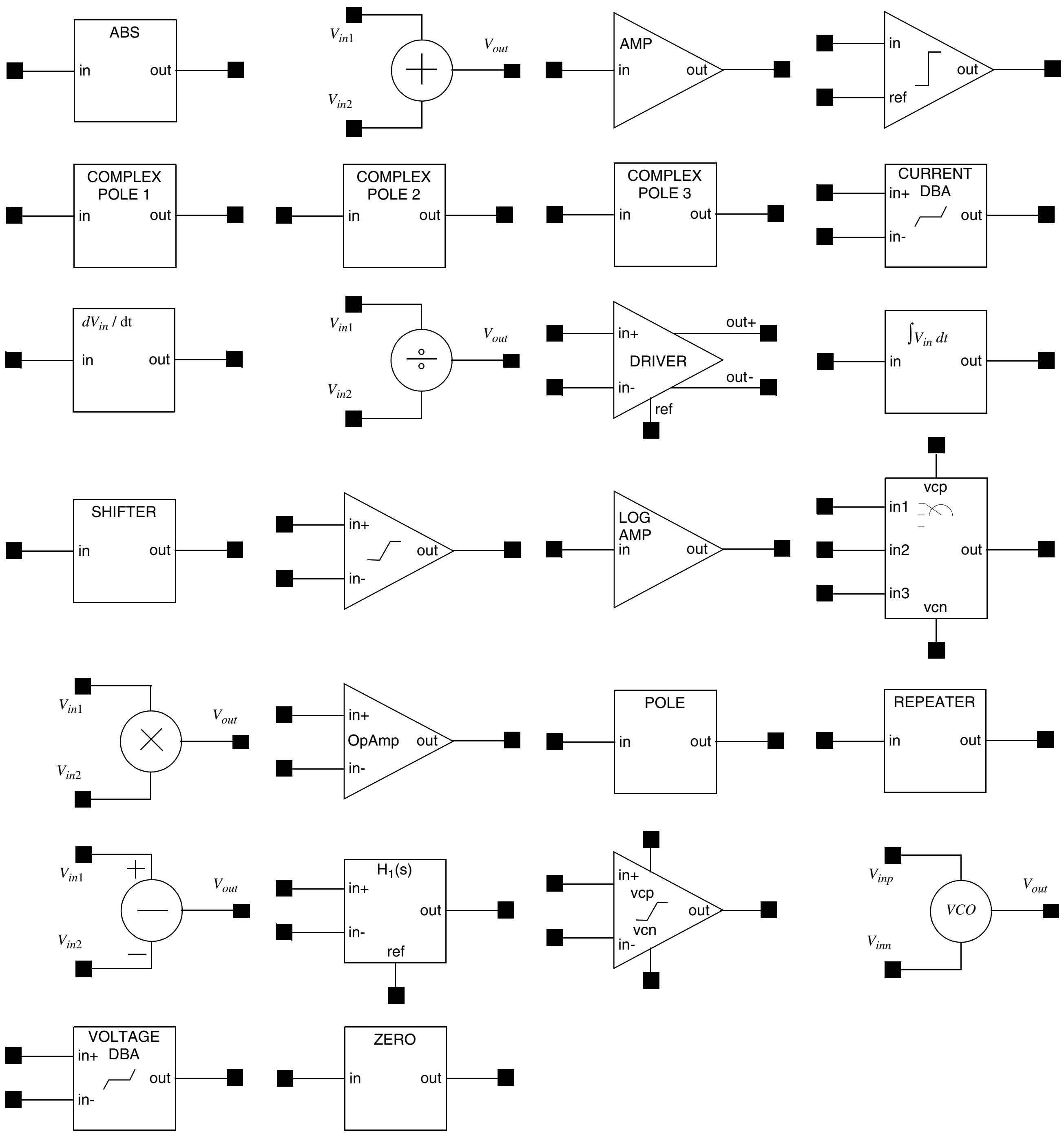
The following list shows the library names of these components. These components are described in detail in Chapter 2, "Functional Block Library Components."

- absoluteValue
- adder
- amplifier
- comparator
- comparator macromodel
- complexPole1
- complexPole2
- complexPole3
- currentDba
- differentiator
- divider
- driver
- integrator
- levelShifter
- limitingAmplifier
- logAmp
- multiplexer
- multiplier
- opAmp
- opAmp macromodel
- pole
- repeatWaveform
- subtractor

- transferFunction1
- variableGainAmplifier
- vco
- voltageDba
- zero

Note: For details about the comparator macromodel, refer to the Appendix A, "Comparator Macromodel." For details about the opAmp macromodel, refer to the Appendix B, "OpAmp Macromodel."

Functional Block Library Component Symbols

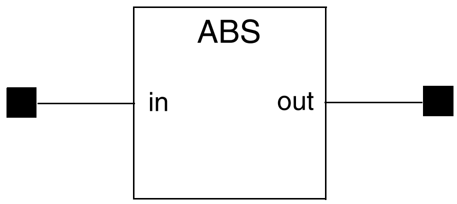


2

Functional Block Library Components

This chapter describes the functional block library components in detail.

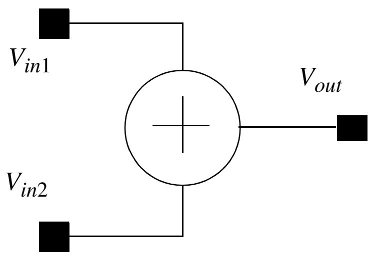
Absolute Value



Description	Output voltage is the absolute value of the input voltage
Library name	absoluteValue
Properties (Defaults)	None
Transfer function	$V_{out} = \lvert V_{in} \rvert$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

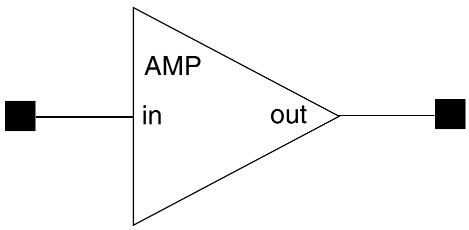
Note: The absolute value function (ABS) always returns a positive output. Its operation is that of an ideal full-wave rectifier.

Adder



Description	Output voltage is the sum of the two input voltages
Library name	adder
Properties (Defaults)	None
Transfer function	$V_{out} = V_{in1} + V_{in2}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Amplifier

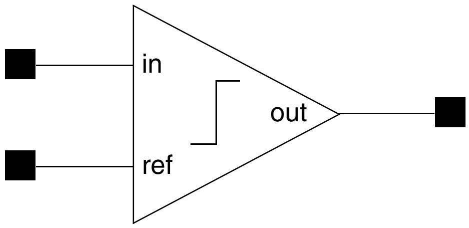


Description	Output voltage can be an amplified, attenuated, or inverted version of the input
Library name	amplifier
Properties (Defaults)	*gain* (2)
Transfer function	$V_{out} = gain \times V_{in}$ *gain* $\geq$ 1.0 gives a noninverting amplifier 0.0 < *gain* < 1.0 gives an attenuator *gain* < 0.0 results in an inverting amplifier
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

- Do not express *gain* in decibels.
- The amplifier is ideal, with no output limiting.

Comparator



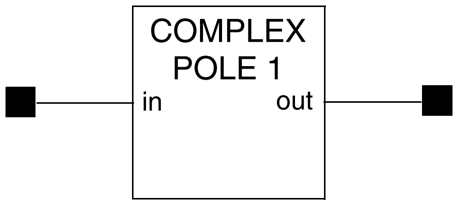
Description	Output voltage rapidly approaches one of two allowed states when the two input voltages are compared
Library name	comparator
Properties (Defaults)	outputHigh (10) outputLow (-10) *slope* (1000)
Transfer function	$V_{out} = \frac{(V_{OH} - V_{OL})}{2} \times \tanh\left(slope \times (V_{in} - V_{ref})\right) + \frac{(V_{OH} + V_{OL})}{2}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

- This comparator does not include hysteresis effects.
- *slope* determines the slope of the transfer region. The single-sided switching region for V_{out} within approximately 99.5 percent of the limits is V_{in} = 0.003.
- V_{OH} is the high level of output voltage.
- V_{OL} is the low level of output voltage.

- This comparator is not the same device created by the comparator macromodel product mentioned in Appendix A, "Comparator Macromodel." The comparator macromodel product can create a new device each time you use it. The symbol and accompanying models for the comparator macromodel are different from those used within this comparator component of the functional block library. For more details about the comparator macromodel, refer to Appendix A, "Comparator Macromodel."

Complex Pole 1

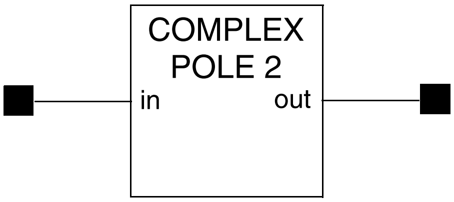


Description	Complex-pole transfer function, using σ and ω_n
Library name	complexPole
Properties (Defaults)	σ (-5) ω_n (6283.1853)
Transfer function	$$\frac{\sigma^2 + \omega_n^2}{[s - (\sigma - j\omega_n)][s - (\sigma + j\omega_n)]}$$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of σ and ω_n

Note:

- σ is the (exponential) damping coefficient.
- ω_n is the damped natural frequency, in radians.
- For left-hand half-plane poles, enter negative values for σ.

Complex Pole 2



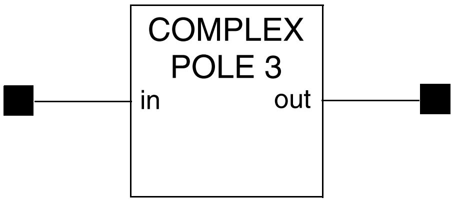
Description	Complex-pole transfer function, using ς and ω_0
Library name	complexPole2
Properties (Defaults)	ς (5E-3) ω_0 (6283.1853)
Transfer function	$\dfrac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of ς and ω_0

Note:

$$2\pi f_0 = \omega_0 \sqrt{1 - \frac{1}{2Q_0^2}}$$

- ς is the damping coefficient.
- ω_0 is the undamped natural frequency, in radians.

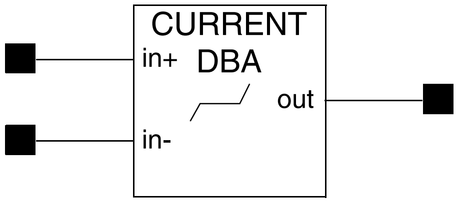
Complex Pole 3



Description	Complex-pole transfer function, using bandwidth and center frequency
Library name	complexPole3
Properties (Defaults)	Q_0 (50) f_0 (1000) peakGain (0)
Transfer function	$\dfrac{\omega_0^2}{s^2 + 2\zeta\omega_0 + \omega_0^2}$ Where you can calculate ω_0 and ς using $2\pi f_0 = \omega_0\sqrt{1 - \dfrac{1}{2Q_0^2}}$ and $Q_0 = \dfrac{1}{2\zeta}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of Q_0 and f_0

- Q_0 is the quality factor at resonance.
- f_0 is the frequency at which | $H(j_2 \pi f)$ | is maximum.
- peakGain is the gain at | $H(j_2 \pi f_0)$ |.

Current Dead-Band Amplifier (currentDba)

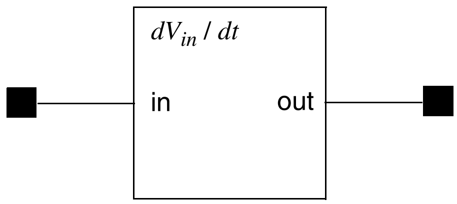


Description	Current dead-band amplifier, across specified range, with separately controllable gains in the amplifying regions
Library name	currentDba
Properties (Defaults)	I_{inLow} (-1E-3) I_{inHigh} (1E-3) I_{ol} (0) gain1 (1) gain2 (1)
Transfer function	$I_{out} = I_{ol}$ If $I_{in} \geq I_{inHigh}$ then $I_{out} = \text{gain2} \times (I_{in} - I_{inHigh}) + I_{ol}$ If $I_{in} \leq I_{inLow}$ then $I_{out} = \text{gain1} \times (I_{in} - I_{inLow}) + I_{ol}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

$I_{in} = I_{in+} - I_{in-}$

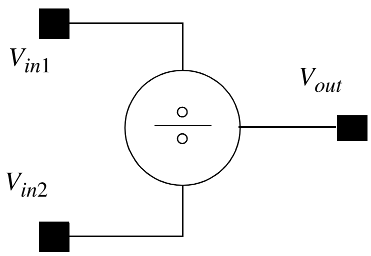
Differentiator



Description	Output voltage is the derivative of the input voltage
Library name	differentiator
Properties (Defaults)	None
Transfer function	$V_{out} = \frac{d(V_{in})}{dt}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of frequency

Note: You need to consider the expected output when using this function; the output can become large even for simple inputs.

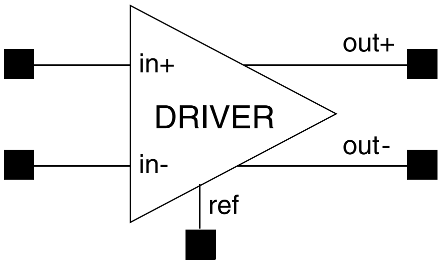
Divider



Description	Output is the quotient of the two input voltages
Library name	divider
Properties (Defaults)	None
Transfer function	$V_{out} = \frac{V_{in1}}{V_{in2}}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note: If V_{in2} is equal to zero, V_{in2} is set to 1E-12.

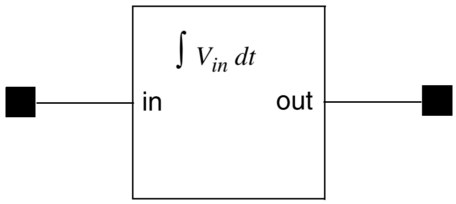
Driver



Description	The differential output voltage is $gain$ times that of the differential input
Library name	driver
Properties (Defaults)	$gain$ (1)
Transfer function	$V_{out} = \left[V_{in+} - V_{in-}\right] \times gain$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note: The differential input is fully floating, while the output is center-tapped about V_{ref}.

Integrator

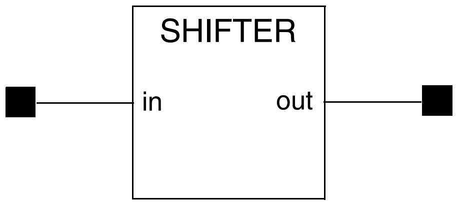


Description	Output voltage is the integral of the input voltage
Library name	integrator
Properties (Defaults)	None
Transfer function	$V_{out} = \int V_{in}\ dt$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of frequency

Caution

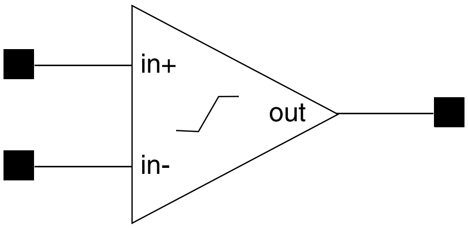
Be careful when using this function. This function is implemented as a perfect integrator; however, if you use unreasonable (electrical) values, this might cause convergence problems.

Level Shifter



Description	Output is a DC level-shifted version of the input voltage
Library name	levelShifter
Properties (Defaults)	*shift* (1)
Transfer function	$V_{out} = V_{in} + shift$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Limiting Amplifier



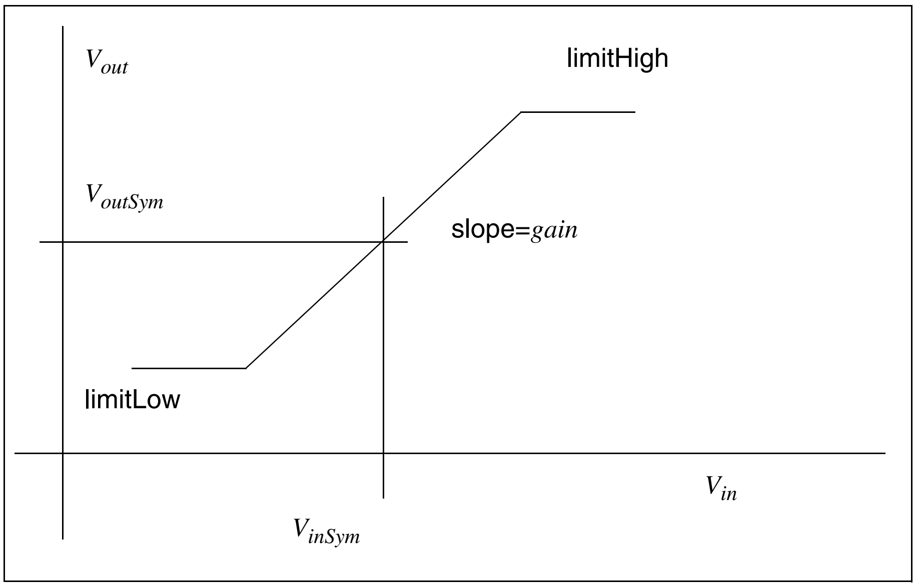
Description	Hard-limiting voltage amplifier
Library name	limitingAmplifier
Properties (Defaults)	$gain$ (1) limitHigh (10) limitLow (-10) V_{inSym} (0)
Transfer function	$V_{out} = gain \times (V_{in} - V_{inSym}) + V_{outSym}$ If V_{out} > limitHigh then V_{out} = limitHigh If V_{out} < limitLow then V_{out} = limitLow
Input impedance	
Output impedance	Zero
Frequency response	Flat

Note: Refer to the following figure.

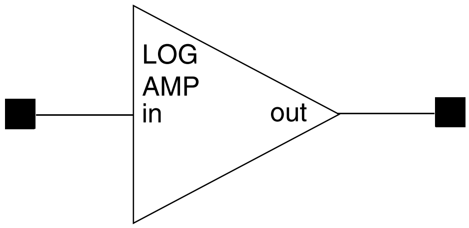
- Do not express $gain$ in decibels.
- V_{inSym} is the input voltage about which the output is symmetrical.

- V_{outSym} = (limitHigh + limitLow)/2
- $V_{in} = V_{in+} - V_{in-}$

Limiting Amplifier Operation



Logarithmic Amplifier

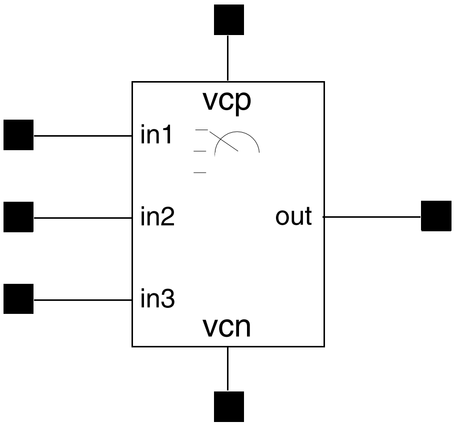


Description	Output voltage is the natural logarithm of the input voltage
Library name	logAmp
Properties (Defaults)	None
Transfer function	$V_{out} = \mathrm{logn}(\lvert V_{in} \rvert)$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

- Uses the natural logarithm.
- The use of $\lvert V_{in} \rvert$ avoids the $\mathrm{logn}(-V)$ problem.
- If V_{in} is equal to zero, V_{in} = E-9 is used to avoid $\mathrm{logn}(0)$.
- Because $\log_x A = \mathrm{logn}A/\mathrm{logn}X$, obtain $\log_x A = \mathrm{logn}A/\log_x$ by using a gain block (amplifier) set to $1/\mathrm{logn}X$.

Multiplexer

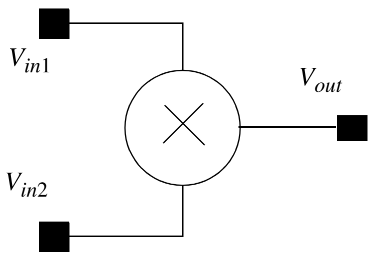


Description	3-to-1 analog selector
Library name	multiplexer
Properties (Defaults)	threshold1 (-1) threshold2 (1)
Transfer function	If $V_c <$ threshold1 then $V_{out} = V_{in1}$ If threshold1 $\leq V_c \leq$ threshold2 then $V_{out} = V_{in2}$ If $V_c >$ threshold2 then $V_{out} = V_{in3}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

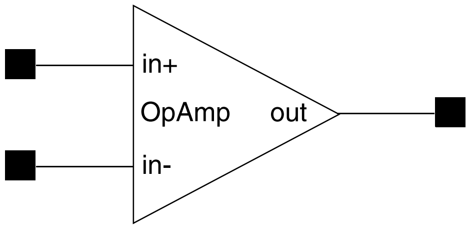
- V_c = vcp - vcn
- This multiplexor is a voltage-controlled switch, not a programmable analog selector.

Multiplier



Description	Output voltage is the product of the two input voltages
Library name	multiplier
Properties (Defaults)	None
Transfer function	$V_{out} = V_{in1} \times V_{in2}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Operational Amplifier (opAmp)

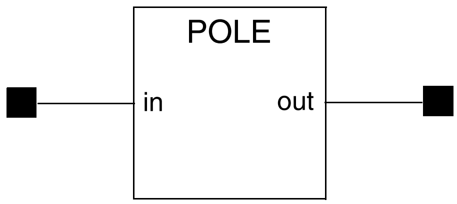


Description	OpAmp is a high gain open-loop stage, which acts like an ideal operational amplifier; for normal operation, gain control is necessary via feedback components
Library name	opAmp
Properties (Defaults)	*openLoopGain* (1E10)
Transfer function	$V_{out} = (V_{in+} - V_{in-}) \times openLoopGain$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

- This block can be used as an ideal operational amplifier to produce summing stages, active filters, gain stages, and integrators.
- OpAmp is not the same device created by the OpAmp macromodel product mentioned in Appendix B, "OpAmp Macromodel." The OpAmp macromodel product can create a new device each time you use it. The symbol and the accompanying models for the OpAmp macromodel are different from those used within this opAmp component. For more details about the OpAmp macromodel, refer to Appendix B, "OpAmp Macromodel."

Pole



Description	Simple, single-pole transfer function
Library name	pole
Properties (Defaults)	f_c (1000)
Transfer function	$\frac{V_{out}}{V_{in}} = \frac{1}{1 + j\frac{f}{f_c}}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of frequency

Note:

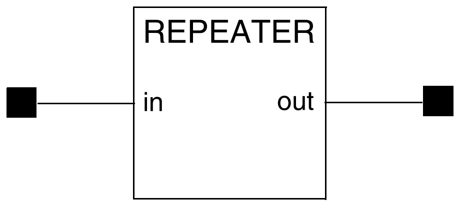
- f_c is in Hz.
- As a function of s,

$$\frac{V_{out}}{V_{in}} = \frac{a}{s + a}$$

where

$$a = 4\pi f_c$$

Repeat Waveform

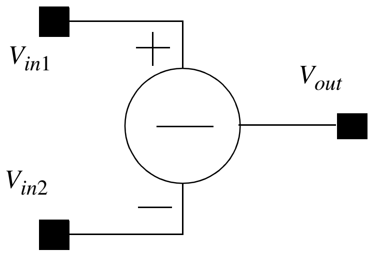


Description	Output repeats the input signal with a specified periodicity
Library name	repeatWaveform
Properties (Defaults)	*period* (1E-3)
Transfer function	$V_{out}(t) = V_{in}(t - (n \times period))$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

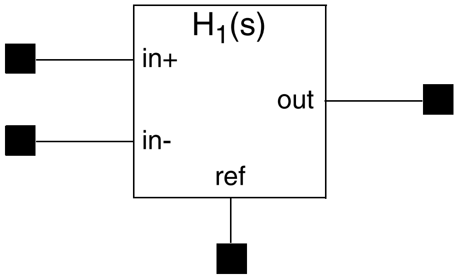
- This function takes a finite-duration aperiodic input signal and creates a periodic signal, with periodicity equal to *period*.
- *period* must be greater than the input signal duration to avoid distortion.
- This function uses the SPICE transmission line element internally.

Subtractor



Description	Output voltage is the difference between the two inputs
Library name	subtractor
Properties (Defaults)	None
Transfer function	$V_{out} = V_{in1} - V_{in2}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Transfer Function 1



Description	Complex transfer function, with up to three each of simple poles, simple zeroes, complex poles, and complex zeroes	
Library name	transferFunction1	
Properties (Defaults)	*dcGain* (1)	dc and low frequency gain
	inputR (1G ohm)	input resistance
	inputC (0)	shunt input capacitance
	outputR (0.001 ohm)	series output resistance
	outputC (0)	output capacitance, parallel with outputR
	nPoles (0)	number of real-valued poles (maximum 3)
	nZeroes (0)	number of real-valued zeroes (maximum 3)
	nConjPoles (0)	number of complex-conjugate poles (maximum 3)
	nConjZeroes (0)	number of complex-conjugate zeroes (maximum 3)
	freqPole1 (1000)	break frequency of pole 1
	freqPole2 (2000)	break frequency of pole 2
	freqPole3 (3000)	break frequency of pole 3
	freqZero1 (1000)	break frequency of zero 1

freqZero2 (2000)	break frequency of zero 2
freqZero3 (3000)	break frequency of zero 3
freqConjPole1 (1000)	damped natural frequency of complex pole 1
sigmaConjPole1 (-5)	damping (exponential) term for complex pole 1
freqConjPole2 (2000)	damped natural frequency of complex pole 2
sigmaConjPole2 (-5)	damping (exponential) term for complex pole 2
freqConjPole3 (3000)	damped natural frequency of complex pole 3
sigmaConjPole3 (-5)	damping (exponential) term for complex pole 3
freqConjZero1 (1000)	damped natural frequency of complex zero 1
sigmaConjZero1 (-5)	damping (exponential) term for complex zero 1
freqConjZero2 (2000)	damped natural frequency of complex zero 2
sigmaConjZero2 (-5)	damping (exponential) term for complex zero 2
freqConjZero3 (3000)	damped natural frequency of complex zero 3
sigmaConjZero3 (-5)	damping (exponential) term for complex zero 3

Transfer function

$$\frac{V_{out}}{V_{in}} = dcGain\,\frac{\Pi\left(1+\frac{jf}{f_{zi}}\right)\times\Pi\left[(\sigma_{czj}^2+4\pi^2f_{czj}^2-4\pi^2f^2)-j(2\sigma_{czj})\right]}{\Pi\left(1+\frac{jf}{f_{pk}}\right)\times\Pi\left[(\sigma_{cpl}^2+4\pi^2f_{cpl}^2-4\pi^2f^2)-j(2\sigma_{cpl})\right]}$$

Where

f_{zi} is the freqZero$_i$
f_{czi} is the freqConjZero$_j$
σ_{czj} is the sigmaConjZero$_j$
f_{pk} is the freqPole$_k$
f_{cpl} is the freqConjPole$_l$
σ_{cpl} is the sigmaConjPole$_l$

Input impedance: Infinite

Output impedance: Zero

Frequency response: Function of the input properties

Note:

- Higher order poles and zeroes are supported. For example, enter zero frequency and zero sigma to specify second-order poles at the origin.
- Enter zero frequency to specify single simple poles and zeroes at the origin.
- For more complex transfer functions, cascade similar blocks or cascade with the pole, zero, complex pole, differentiator, and integrator blocks.
- Always use the earlier numbered singularities first. For example, if the number of poles is set to 1, then only pole 1 has effect. The cases of zeroes and complex pairs are similar.

Example

To specify a transfer function with

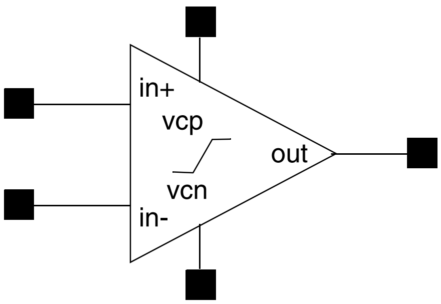
- A single zero at f = 1000 Hz
- A second-order pole at the origin
- A complex pole at f = 3000 Hz, with sigma = -1.0

- A dc gain of 10

then

```
dcGain = 10
nZeroes = 1
nConjPoles = 2
freqZero1 = 1000
freqConjPole1 = 0.0
sigmaConjPole1 = 0.0
freqConjPole2 = 3000
sigmaConjPole2 = -1.0
```

Voltage-Controlled Variable-Gain Amplifier

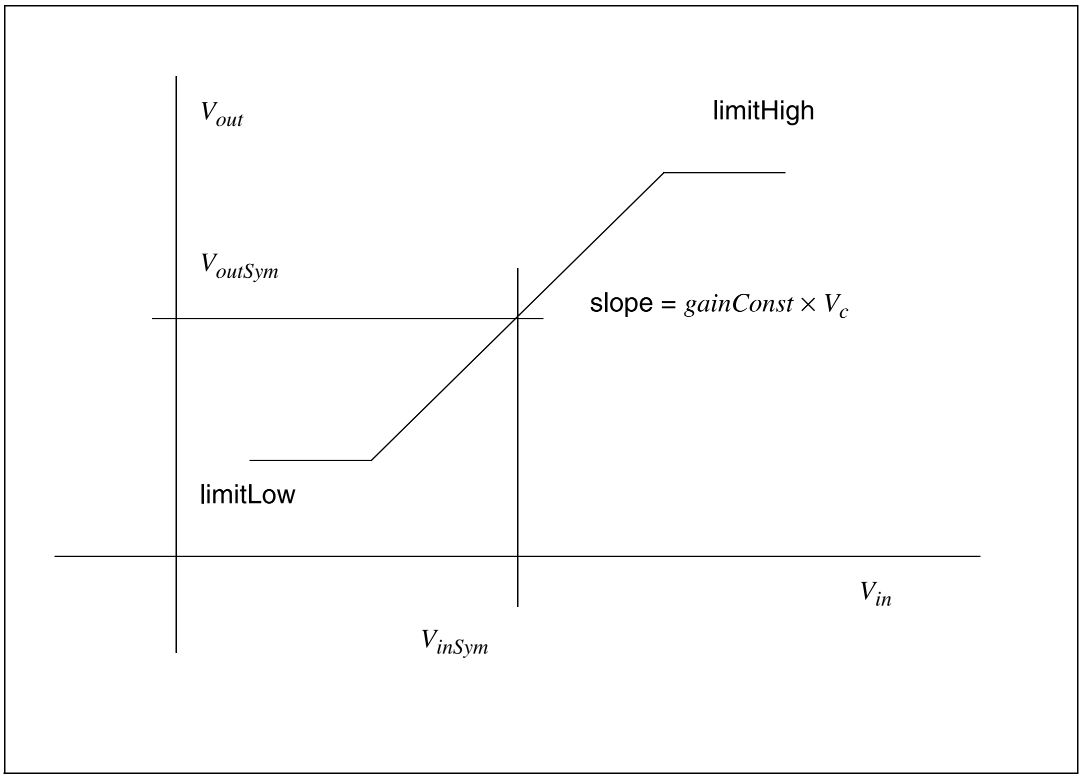


Description	The gain of this limiting voltage amplifier is a function of the voltage (*vcp* - *vcn*)
Library name	variableGainAmplifier
Properties (Defaults)	$gainConst$ (1) limitHigh (10) limitLow (-10) V_{inSym} (0)
Transfer function	$V_{out} = gainConst \times V_c(V_{in} - V_{inSym}) + V_{outSym}$ If V_{out} > limitHigh then V_{out} = limitHigh If V_{out} < limitLow then V_{out} = limitLow
Input impedance	
Output impedance	Zero
Frequency response	Flat

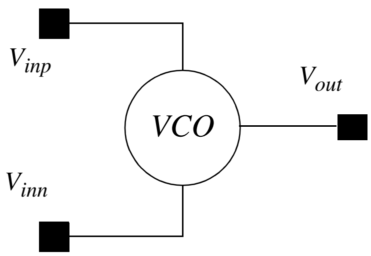
Note: Refer to the following figure.

- V_{inSym} is the input voltage about which the output is symmetrical.
- V_{outSym} = (limitHigh + limitLow)/2
- V_c is the control voltage, (vcp - vcn)
- $V_{in} = V_{in+} - V_{in-}$

Variable Gain Amplifier Operation



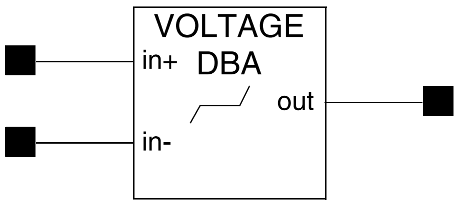
Voltage-Controlled Oscillator (VCO)



Description	Voltage-controlled oscillator
Library name	vco
Properties (Defaults)	*amplitude* (1) *centerFreq* (1000) *VCOgain* (1000)
Transfer function	$V_{out} = amplitude \times \sin 2\pi(centerFreq)\left[VCOgain \times (V_{inp} - V_{inn}) \times t\right]$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Function of applied properties

Note: *VCOgain* is the Hz/Volt proportionality variable, often called VCO sensitivity.

Voltage Dead-Band Amplifier (VoltageDba)

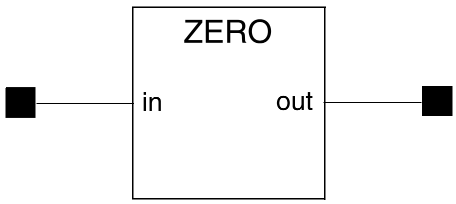


Description	Voltage dead-band amplifier, across a specified range, with separately controllable gains in the amplifying regions
Library name	voltageDba
Properties (Defaults)	V_{inLow} (-10) V_{inHigh} (10) V_{OL} (0) $\mathrm{gain1}$ (1) $\mathrm{gain2}$ (1)
Transfer function	$V_{out} = V_{OL}$ If $V_{in} \geq V_{inHigh}$ then $V_{out} = \mathrm{gain2} \times (V_{in} - V_{inHigh}) + V_{OL}$ If $V_{in} \leq V_{inLow}$) then $V_{out} = \mathrm{gain1} \times (V_{in} - V_{inLow}) + V_{OL}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	Flat

Note:

$V_{in} = V_{in+} - V_{in-}$

Zero



Description	Single-zero transfer function
Library name	zero
Properties (Defaults)	f_c (1000)
Transfer function	$\frac{V_{out}}{V_{in}} = 1 + \frac{jf}{f_c}$
Input impedance	Infinite
Output impedance	Zero
Frequency response	F ($j\omega$)

Note:

- f_c is in Hz.
- As a function of s,

  $$\frac{V_{out}}{V_{in}} = b\left(s + \frac{1}{b}\right)$$

  where

  $$b = \frac{1}{2\pi f_c}$$

3

Functional Block Library Examples

As described in Chapter 1, “About the Functional Block Library,” functional blocks contain the essential functionality of the parts they represent. One important application of these blocks is the creation and simulation of high-level systems. The components in the library are used as primitives to create more complex blocks. For example, you can arrange the system shown in example 1 to appear as a single symbol for use in a communication system design.

Examples 1 and 2 illustrate the use of the functional blocks and also demonstrate that you do not have to use the blocks in isolation. Regular circuit components and components from any other installed library can be placed on a schematic and simulated.

DSB-AM Example

The example system shown in Figure 3-1 on page 54 shows how the process of double-sideband amplitude modulation (DSB-AM) can be performed using functional blocks. The piecewise linear voltage source generates one cycle of a triangular wave, as shown on waveform *c* in Figure 3-2 on page 55.

The repeater block takes as its input the one cycle triangular wave and repeats it, with a period of 10 ms. The output from the repeater is the periodic triangular waveform shown in waveform *b* in Figure 3-2 on page 55.

This signal is the baseband signal, $V_s(t)$, which modulates the carrier. The depth of modulation required is 100 percent, so it is necessary for the carrier amplitude and the effective baseband amplitude to be equal.

Here, the carrier is set to 1 volt at 1000 Hz. As the modulation process is performed by the multiplier block (product or balanced modulation), the amplitude of the baseband signal is raised from 0.5 volts at the output of the repeater, to 1 volt. To achieve DSB-AM, the baseband signal must also be level-shifted by an amount equal to the carrier amplitude.

Mathematically,

$$V_{mod}(t) = \left(V_c + V_s(t)\right) \times \sin(2\pi f_c t)$$

where

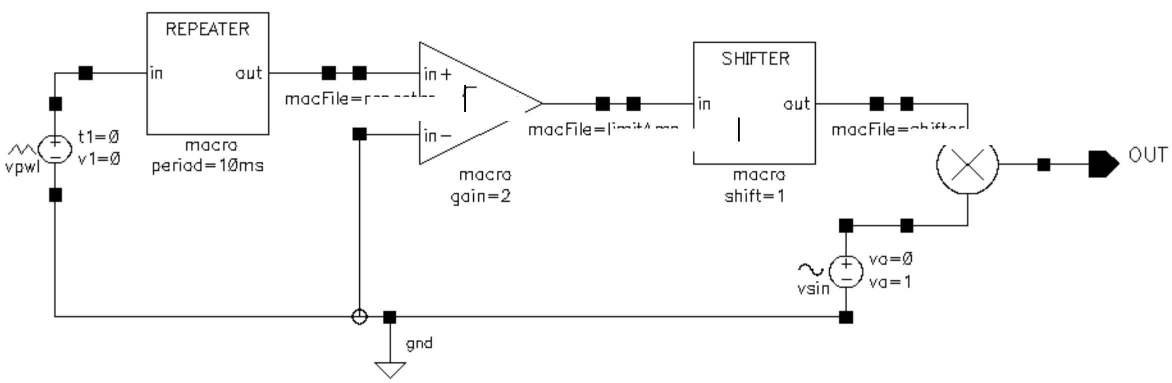
V_c = carrier amplitude

f_c = carrier frequency

$V_s(t)$ = modulating symbol

To achieve this, set the gain of the amplifier block to 2 and the amount of shift to 1 volt. The modulated result is shown in waveform *a* in Figure 3-2 on page 55.

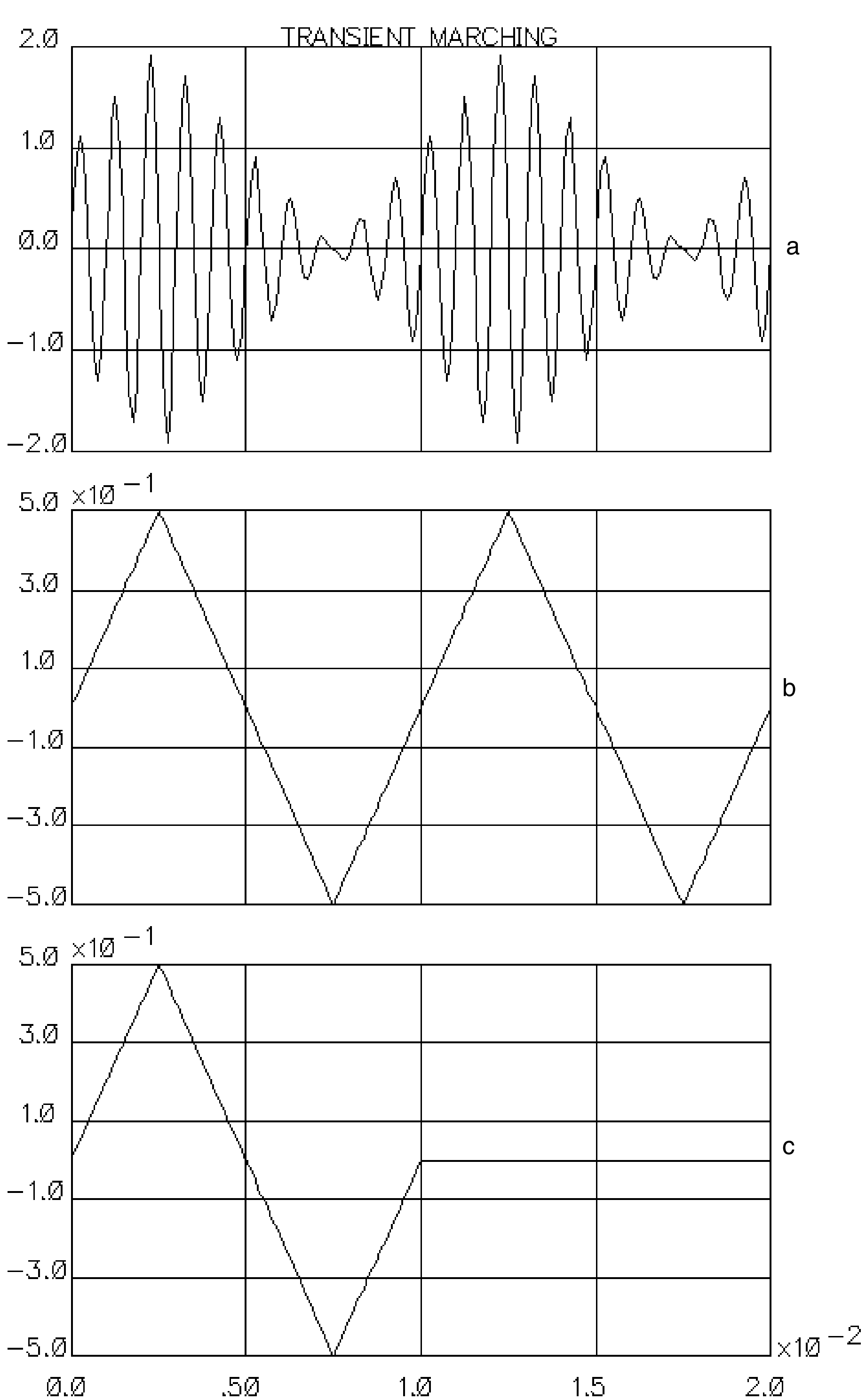
Figure 3-1 DSB-AM Schematic



The waveforms for this example are shown in Figure 3-2 on page 55, where

- Waveform *a* is the modulated result of the DSB-AM modulation
- Waveform *b* is the periodic triangular waveform that the repeater block generates as output
- Waveform *c* is one cycle of a triangular wave that the linear voltage source generates

Figure 3-2 Waveforms for DSB-AM Schematic



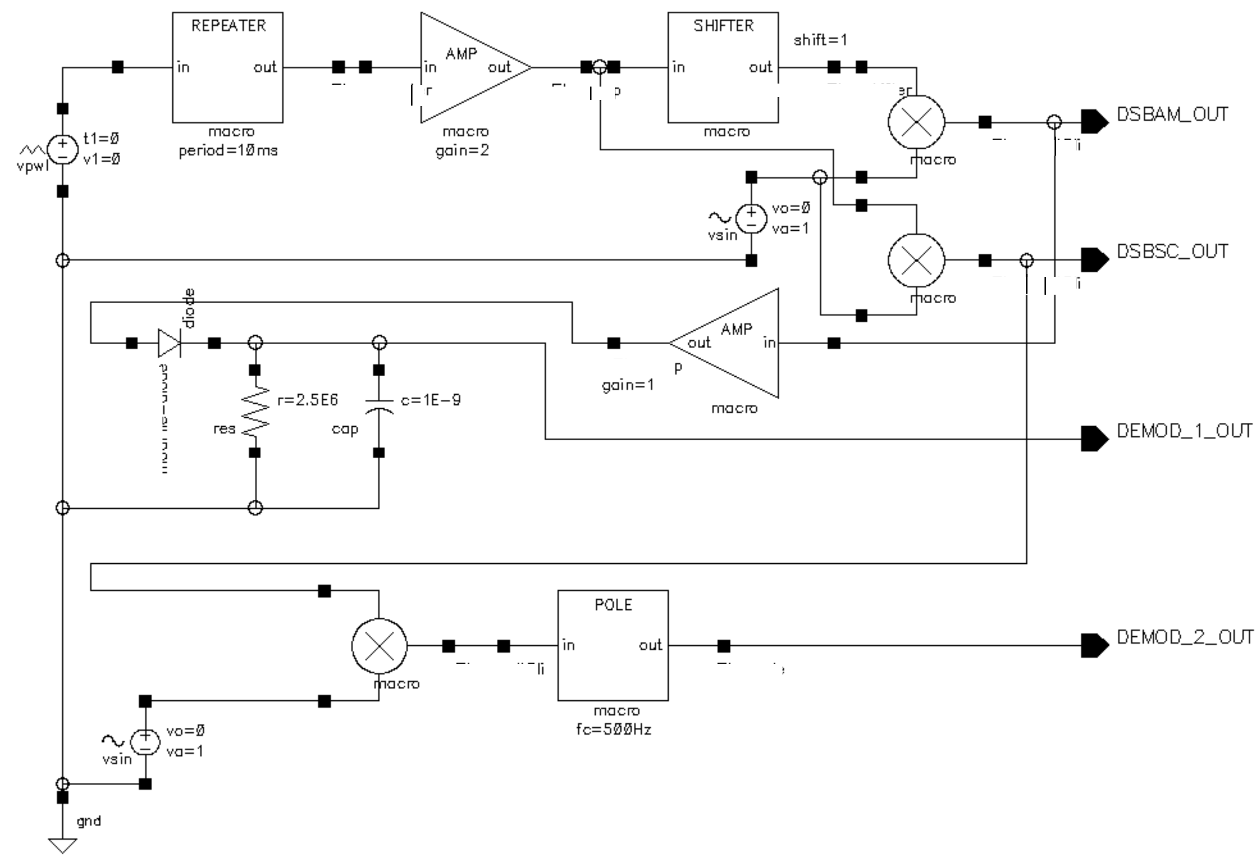
DSB-SC Example

Example 2 uses the circuitry already described in Example 1. The circuitry in Example 2 has the following additional features that enable DSB-SC modulation:

- An additional multiplier stage
- Two demodulator sections
    - Demodulator 1 is a simple noncoherent diode detector (envelope detector)
    - Demodulator 2 is a fully coherent decoder (remodulator), with low-pass filtering on its output

The frequency of the carrier reinsertion oscillator is set to 2000 Hz, which is the frequency of the carrier signal used in this example. Further, the cutoff (3 dB) frequency of the single pole (low-pass) filter at the output of the remodulator was set to 500 Hz. This allows up to the fifth harmonic of the detected triangular wave to be passed to the output.

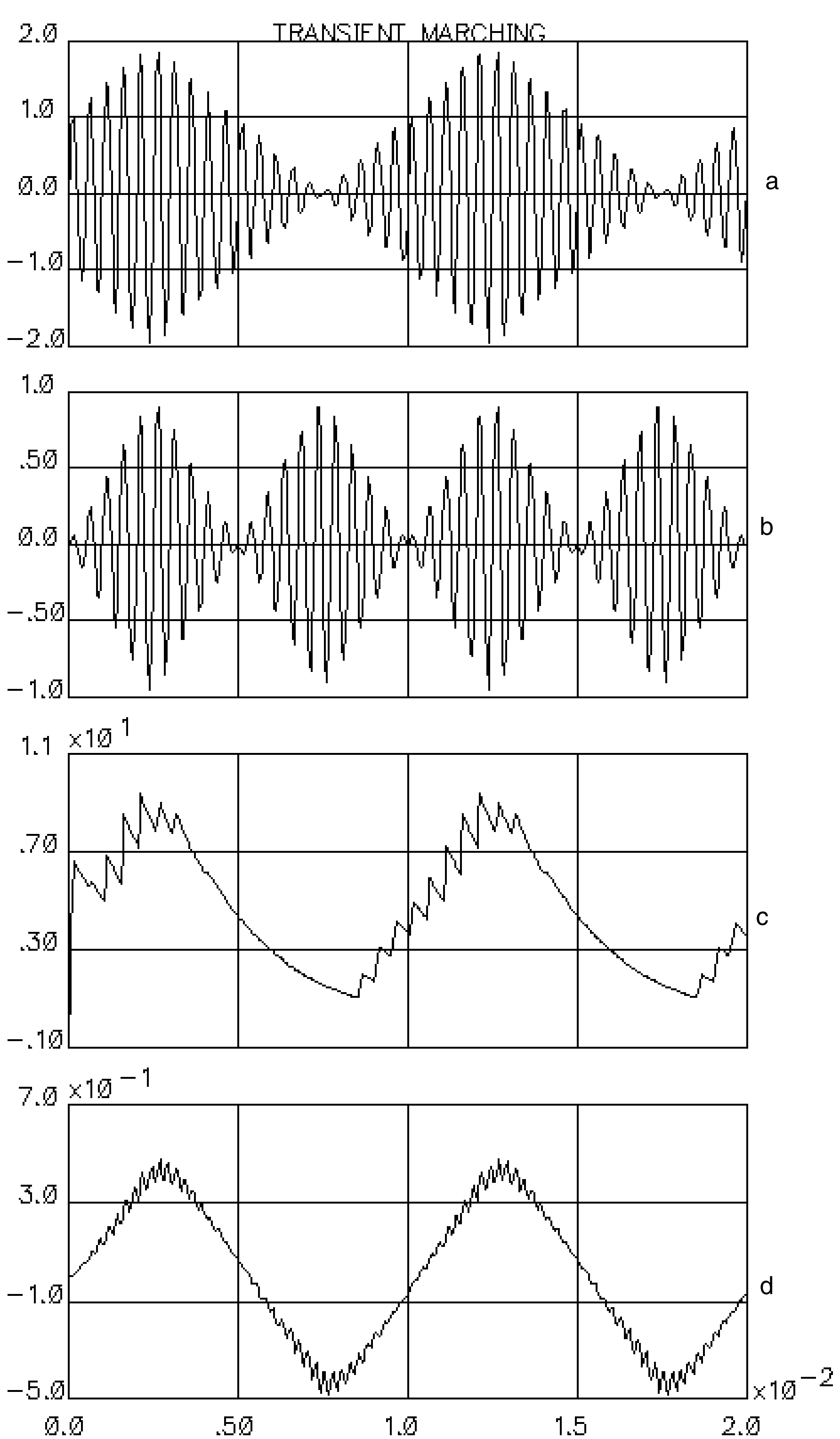
Figure 3-3 DSB-AM and DSB-SC Schematic



The waveforms for this example are shown in Figure 3-4 on page 58, where

- Waveform *a* is the DSB-AM output
- Waveform *b* is the DSB-SC output
- Waveform *c* is the output of the incoherent demodulator
- Waveform *d* is the output of the coherent detector

Figure 3-4 DSB-AM and DSB-SC Schematic



A

Comparator Macromodel

The comparator macromodel is a parameterized macromodel in the analog circuit design environment. This macromodel uses a simplified equivalent circuit to emulate the DC, AC, and transient nature of a real comparator's electrical performance.

A macromodel is advantageous because it reduces computation time and space, but it still retains accuracy. The comparator model shown in this appendix is a modified model of the Harris level_0 comparator macromodel, which has a simple RC circuit with the clamping stage. This macromodel represents both the IC comparator's performance and an ideal comparator as accurately as possible.

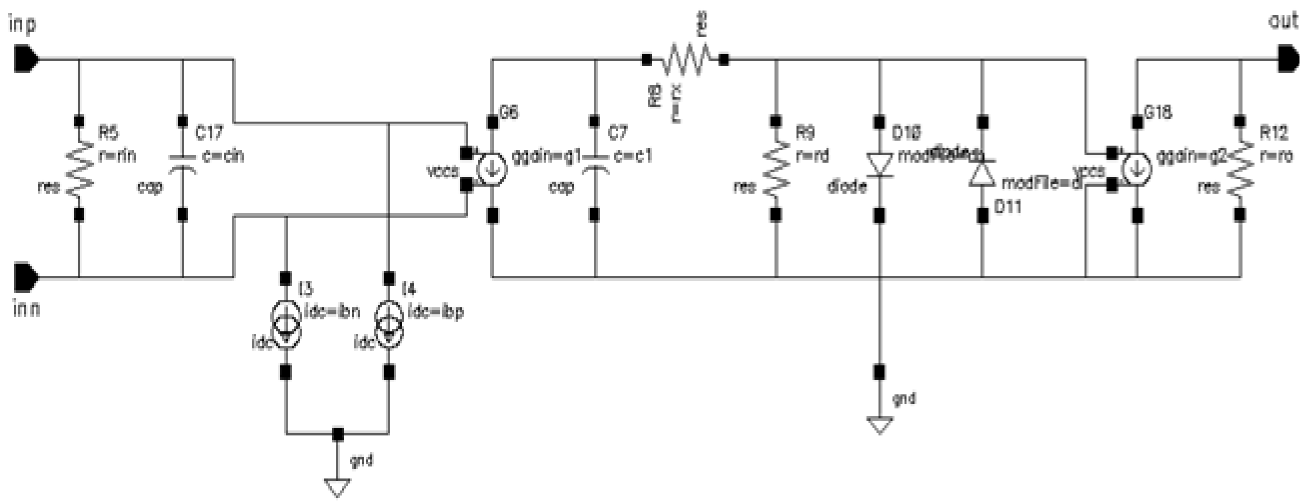
Note: The use of the comparator macromodel does not create the same device as that denoted earlier in Chapter 2, "Functional Block Library Components." For details about the comparator component, refer to "Comparator" on page 21.

The comparator macromodel simulates the following effects:

- Input bias current
- Input offset current
- Input offset voltage
- Input resistance
- Input capacitance
- Open loop gain
- Small-signal delay time
- Small-signal rising and falling time
- Overdrive voltage for small-signal transient response test
- Output voltage swing
- Output resistance

Circuit Diagram

The following figure is a schematic of the comparator macromodel circuit.



Functional Description

There are three stages in the comparator macromodel:

- Input stage
- Gain and delay stage
- Output stage

Input Stage

The input stage is built with four components:

- R_{in} = input resistance
- C_{in} = input capacitance
- I_{bp} = bias current
- I_{bn} = bias current

The input stage of the comparator macromodel uses a rough linear approximation. The four components model the input current versus voltage variation around the active region. However, the input stage does not model the saturation and cutoff effects of the input currents and terminal resistances.

Gain and Delay Stage

The gain and delay stage is constructed with a voltage-controlled current source and a single RC time constant circuit.

G_1, C_1, R_x, and R_d create a single pole roll-off amplifier circuit. R_x, R_d, and G_1 model small signal gain.

The clamping voltages of the two diodes D_h and D_l are set to the value of the positive and negative output voltage swing. These diodes are built to be referenced directly to ground. If the negative output voltage swing V_{OL} is defined as zero like the default value, an independent voltage source of value $V_{OH}/2$ is inserted between the diodes and ground. This allows the output voltages to have symmetric limiting characteristics.

Output Stage

G_2 and R_o build the output stage, which forms a gain of one buffer. G_2 is put there to isolate the gain stage and the outside load to reduce the unwanted influence of the load to the modeling of delay time.

Macromodel Netlist

The following shows a netlist of the comparator macromodel.

```
# comparator 0.s
#
# Description:  Small signal model of input stage, no
#               Io limits or power supply effects, simple
#               resistive input stage, small large signal
#               delay, constant Rout and no current limit
#
# Format:
#
# USE  comparator name ibias vos ios rin cin voh vol vis trs tds gol ro
#     &1 &2 &3 &4 &5 &6 &7 &8 &9 &10 &11 &12 &13
#
# Default values of the parameters:
#
# &1   model name                               name = &1
# &2   Input bias current                       ibias = 200n
# &3   Input offset voltage                     vos = 0
# &4   Input offset current                     ios = 10n
# &5   Input resistance                         rin = 1meg
# &6   Input capacitance                        cin = 1p
# &7   Positive output voltage swing            voh = 5
# &8   Negative output voltage swing            vol = 0
# &9   Overdrive voltage for delay time test    vis = 2m
# &10  Small signal rising and falling time test  trs = 20n
# &11  Small signal delay time                  tds = 50n
# &12  Open loop game                           gol = 100k
# &13  Output resistance                        ro = 200
```

```
#
.subckt &1 10 20 30
#
# Input stage
#
RIN   10  20   (&5)
CIN   10  20   (&6)
IBP   10  0    (&2) +(&4)/2
IBN   20  0    (&2) -(&4)/2
#
# Gain Stage
#
G1  0   2   poly    1  10  20 =-(&3)*(&12)  1E5=(&12)/1E5
C1  2   0   ((&11-&10/2)/log(2))/(1E5/&
             (&12*(&9)*(&10)/(&7-(&8))/((&11-&10/2)/log(2))-1))
RX  2   3   1E5/(&12*(&9)*&10/(&7-(&8))/((&11-&10/2)/log(2))-1)
#
If &7*(&8)<>0 then
  RD   3 0   1E5
  DH   3 0   DH
  DL   0 3   DL
 .MODEL DH D  IS = 0.3*(&12)/1E33  N = &7/VT/LOG(1E28)
 .MODEL DL D  IS = 0.3*(&12)/1E33  N = -(&8)/VT/LOG(1E28)
else
  RD   3 8    1E5
  DH   3 8    DM
  DL   8 3    DM
  VM   8 0    (&7+(&8))/2
 .MODEL DM D  IS=0.3*(&12)/1E33  N=(&7+&(&8))/2/VT/LOG(1E28)
endif
#
#   Output Stage
#
RO  0 30   (&13)
G2  0 30   poly 1 3 0  0 =1/(&13+0.1)
.ends &1
```

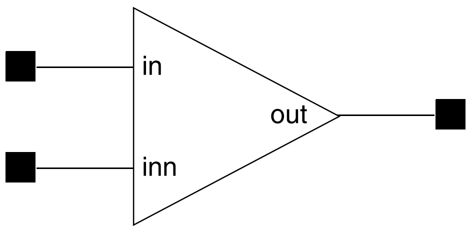
Using the Comparator Macromodel

You can use the comparator macromodel by

- Defining the common specification sheet values, available in most semiconductor product catalogs
- Placing the comparator macromodel symbol in your schematic
- Defining the CDF parameter values and by using default values for parameters that you do not set

Entering the Macromodel into Your Design

The following is a comparator macromodel symbol:



For proper operation of the macromodel, you need to connect the following three I/O pins.

`IN`	Positive input terminal
`INN`	Negative input terminal
`OUT`	Output terminal

Setting Macromodel Parameters

There are 12 parameters you can use to describe the behavior of the comparator macromodel.

Property	Typical Value	Description
ibias	200 n	Input bias current
vos	0	Input offset voltage
ios	10 n	Input offset current
rin	1 meg	Input resistance
cin	1 p	Input capacitance
voh	5	Positive output voltage swing
vol	0	Negative output voltage swing
vis	2 m	Overdrive voltage for small-signal transient response test
trs	20 n	Small-signal rising and falling time
tds	50 n	Small-signal delay time

Property	Typical Value	Description
gol	100 k	Open loop gain
ro	200	Output resistance

The cell CDF description defines these parameters. You can make these properties arbitrary functions of temperature (`TEMPDC`), random number generators, or other user-defined variables. For example, you can specify the offset voltage “vos” linear function for temperature by entering its value in the CDF description as follows:

```
3m + 0.01m * (TEMPDC - 25)
```

Accuracy and Limitations

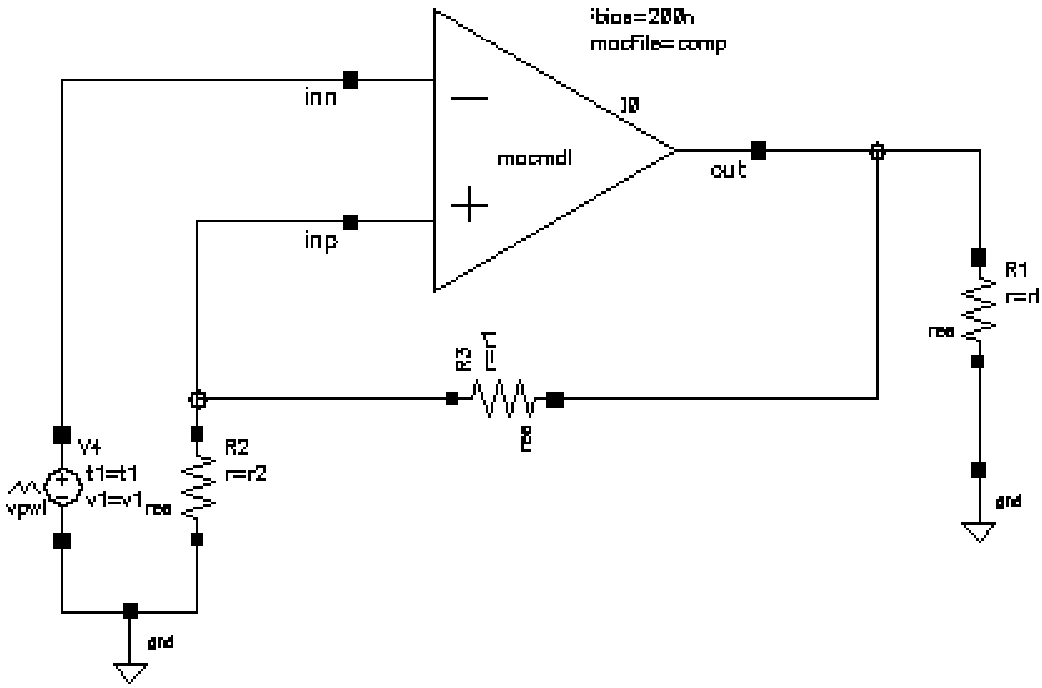
Electrical effects are modeled accurately by this macromodel. However, such a simple circuit might not work as a comprehensive model. The comparator macromodel

- Cannot model specifications that are not defined as macromodel parameters
- Cannot model high frequency response

  The behavior of the comparator at a frequency higher than $1/(4 \times t\ ds)$ is not expected to match the circuit measurements. The capacitor of the single time constant circuit is not fully charged. The delay time might not be realistic.
- Cannot vary delay time according to the input overdrive voltage
- Cannot model hysteresis effects

Example

Below is a schematic of a Schmitt Trigger that uses the comparator macromodel.



B

OpAmp Macromodel

The OpAmp macromodel uses a simplified equivalent circuit to emulate the DC, AC, and transient nature of real amplifier electrical performance.

The electrical performance is defined by common data sheet performance factors or parameters (such as offset voltage and slew rate) as properties on the macromodel symbol. You use these parameters as arguments in complex algebraic expressions defining the elemental values of the components making up the macromodel.

Note: The use of the OpAmp macromodel does not create the same device as that denoted earlier in "Operational Amplifier (opAmp)" on page 39.

The OpAmp macromodel simulates the following effects:

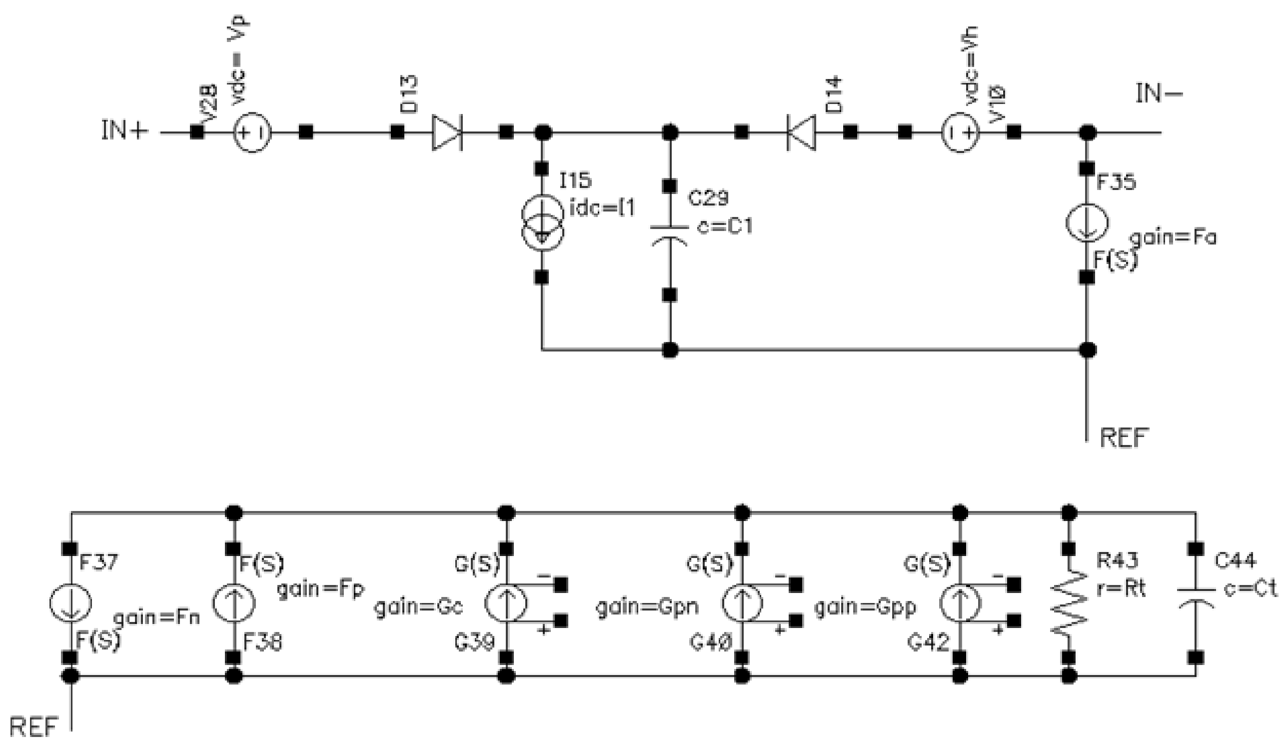
- Input bias current
- Input offset current
- Input offset voltage
- Input capacitance
- Positive and negative slew rate
- Positive feedthrough
- Open loop gain
- Common mode rejection ratio
- Gain bandwidth product
- Second pole frequency
- Saturation delay time
- High frequency output impedance
- AC output resistance
- DC output resistance

- Positive and negative output swing
- Positive and negative output current limit
- Positive and negative power supply rejection ratio
- Positive and negative power supply current

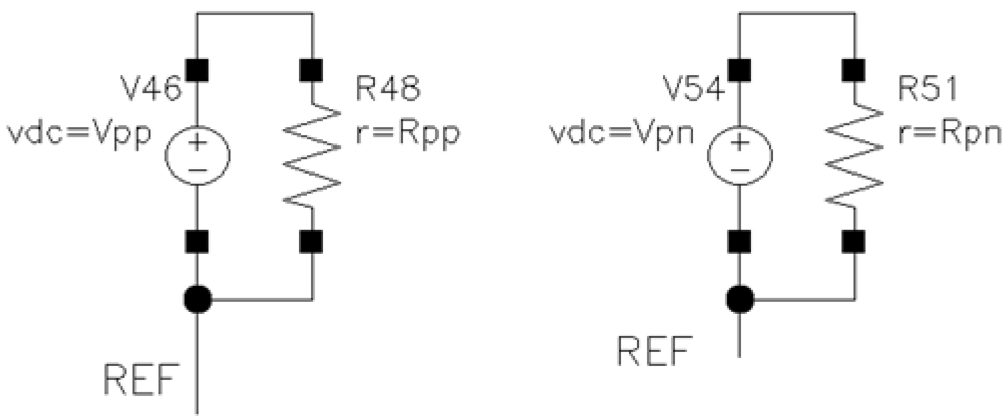
Equivalent Circuit

The following figure is a schematic of the OpAmp macromodel circuit.

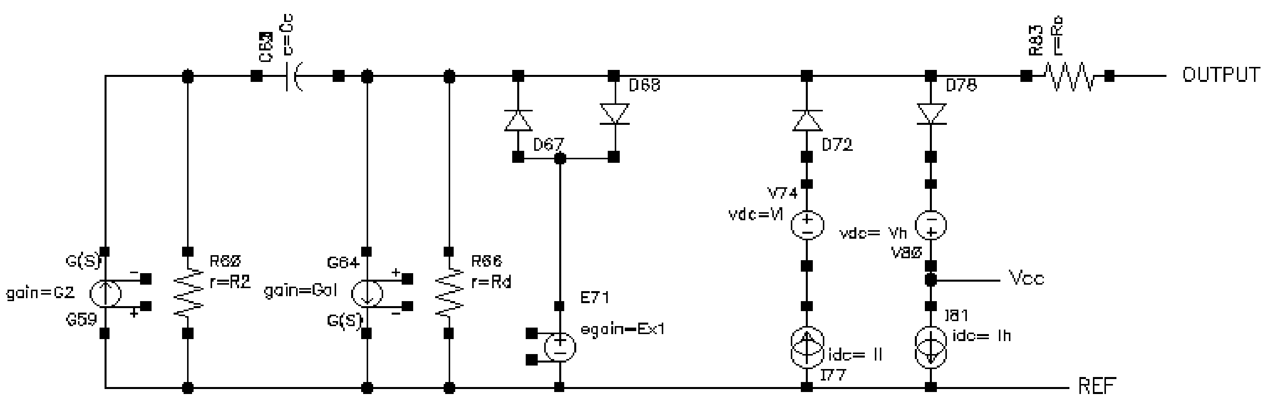
Input Stage



Supply Voltages



Output Stage



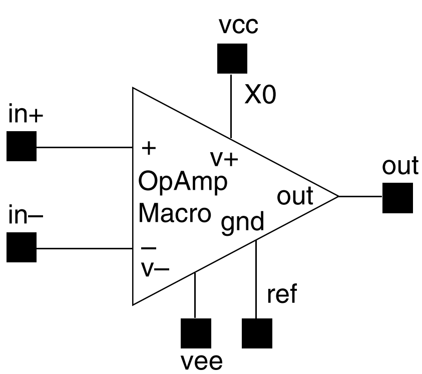
Using the OpAmp Macromodel

You can use the the OpAmp macromodel by

- Defining the common specification sheet values, available in most semiconductor product catalogs
- Placing the OpAmp symbol in your schematic
- Defining the CDF parameter values and by using default values for parameters that you do not set

Entering the Macromodel into Your Design

The following is an OpAmp macromodel symbol.



For proper operation of the macromodel, you need to connect the following six I/O pins.

`IN+`	Positive differential input pin
`IN-`	Negative differential input pin
`OUT`	OpAmp output pin
`VCC`	Positive power supply pin
`VEE`	Negative power supply pin
`REF`	Reference voltage (typically ground)

Setting Macromodel Parameters

There are 23 parameters you can use to describe the DC, AC, and transient behavior of the OpAmp in the macromodel.

Property	Typical Value	Description
ccc	30 p farads	Internal compensation capacitor
ibs	100 n amps	Input bias current
ios	10 n amps	Input offset current
vos	1 m+.01 m* (tempdc-25) volts	Input offset voltage
srp	10 me volts/second	Positive slew rate
srn	10 me volts/second	Negative slew rate
ftp	0.1	Positive feedthrough
gol	100 k volts/volt	Open loop gain
cmr	90 db	Common mode rejection ratio
gbw	10 MHz	Gain-bandwidth product
ff2	20 MHz	Second pole frequency
tsd	500 n seconds	Overdrive storage delay
roa	20 ohms	AC output resistance
rod	30 ohms	DC output resistance
vsn	-14 volts	Negative output swing limit
vsp	14 volts	Positive output swing limit
ion	50 m amps	Negative output current limit
iop	50 m amps	Positive output current limit
prn	90 db	Negative power supply rejection ratio
prp	90 db	Positive power supply rejection ratio
iee	5 m amps	Negative power supply current
icc	5 m amps	Positive power supply current
vnm	15 volts	Nominal power supply

You define these parameters in the cell CDF description.

The property value can be a

- Number
- Variable
- Valid expression (with variables)

For example, to define the offset voltage as a simple linear function of temperature, enter the following expression as the value of the offset voltage property:

```
vos = 1m + 0.01m * (TEMPDC - 25)
```

Obtaining Macromodel Parameters

You can obtain model parameters in several ways, depending on your application.

- IC vendor catalog

  Obtain most of the parameters from the specific amplifier catalog data sheet

- Schematics

  Simulate and measure the performance parameters you want using the OpAmp transistor schematic and associated SPICE device models

- Component

  Make the measurements in the lab using the OpAmp component

You can use the default values or set some reasonable values for the remaining parameters. For example, you might set parameters to some typical values (or use the defaults) when exploring your system design space before selecting a particular amplifier, then modify the values later.

Applying Macromodels

You can use operational amplifier macromodels for a variety of applications. For example, these macromodels can be used for

- “What-if” exploratory system simulation
- Fast system simulation
- Library characterization and protection

These three applications are described in detail below.

“What-if” Exploratory System Simulation

You can perform the “what-if” exploratory system simulation with the following steps.

- Implement a circuit with generic OpAmp macromodels early in an analog system design cycle. This way, you can quickly enter the OpAmps into the schematic, simulate, and independently set and change the OpAmp parameters (such as slew rate, bandwidth, and offset voltage) to understand their effects on system performance.
- Center the design, locate an OpAmp as close as possible to the required specifications in a catalog, then enter its data sheet values to verify that your system functions with the real parameters.
- Enter or design the OpAmp transistor level circuits to the defined specifications.

Note: Because most macromodel parameters are independent, be careful not to specify difficult or unrealistic combinations of specifications, such as low power and high speed.

Fast System Simulation

It might be impractical to simulate the entire system interactively at the transistor level when the system includes many OpAmps.

- Change and simulate the system quickly using macromodels.
- Run final verification simulations on the entire system in background or noninteractive mode.

Library Characterization and Protection

The OpAmp macromodel also finds application in library characterization and protection (as described here).

- Many CAD, semiconductor, and ASIC vendors provide qualified macromodel libraries of commonly used catalog components or cells.
- Systems designers get faster simulations using these macromodels.
- Vendors protect their circuit designs and process parameters.

OpAmp System Size

A primary reason for using macromodels in large systems is for faster simulations. These simulations are faster when compared to transistor-level simulations. These simulations are about 10 to 50 times faster for complex OpAmps containing between 30 and 100 devices. Storage requirements are also reduced using macromodels.

The fast simulation time and reduced storage requirement of smaller amplifiers (fewer than 10 devices) might not compensate for the loss of accuracy from using macromodels. However, macromodels are still useful for exploratory tuning and for protecting libraries.

Accuracy and Limitations

The OpAmp macromodel might not work as a comprehensive model. Its limitations are described here.

- Macromodel accuracy

  The OpAmp macromodel is less accurate and comprehensive than the transistor level representation, and even less accurate than the circuit measurements.

- Model scope

  Specifications not defined as macromodel parameters are not modeled, including common-mode input range, noise, transient power dissipation, and output impedance.

- Temperature

  Temperature effects are not implicitly included, but can be explicitly defined as an algebraic or table function. For example:

  offset voltage = 1m + 0.01 m * (tempdc - 25)

  or

  offset voltage = TABLE(tempdc,-75,0,25,1m,125,2m)

- Parameter independence

  To a first order of accuracy, you can set the macromodel parameters independently, ignoring interactions.

- Small input bias currents

  When using small input bias currents (~1E-12), decrease the simulation control variable GMIN (minimum conductance) to ~1E-15 (default is 1.0E-12). This helps to prevent relatively large leakage currents and corresponding accuracy loss.

- Unrealistic conditions

  The macromodel is particularly good for modeling the OpAmp components. Avoid the following unrealistic parameter values or extreme conditions:

  - Offset voltages less than 10 µvolts vary due to numerical errors
  - The quantities below affect the accuracy of the slew rate of the macromodel in small but measurable ways
    - Gain-bandwidth product (GBW)
    - Large-signal saturation delay (TSD)
    - Second pole frequency (FF2)

    This effect can be significant for the following examples of extreme cases:

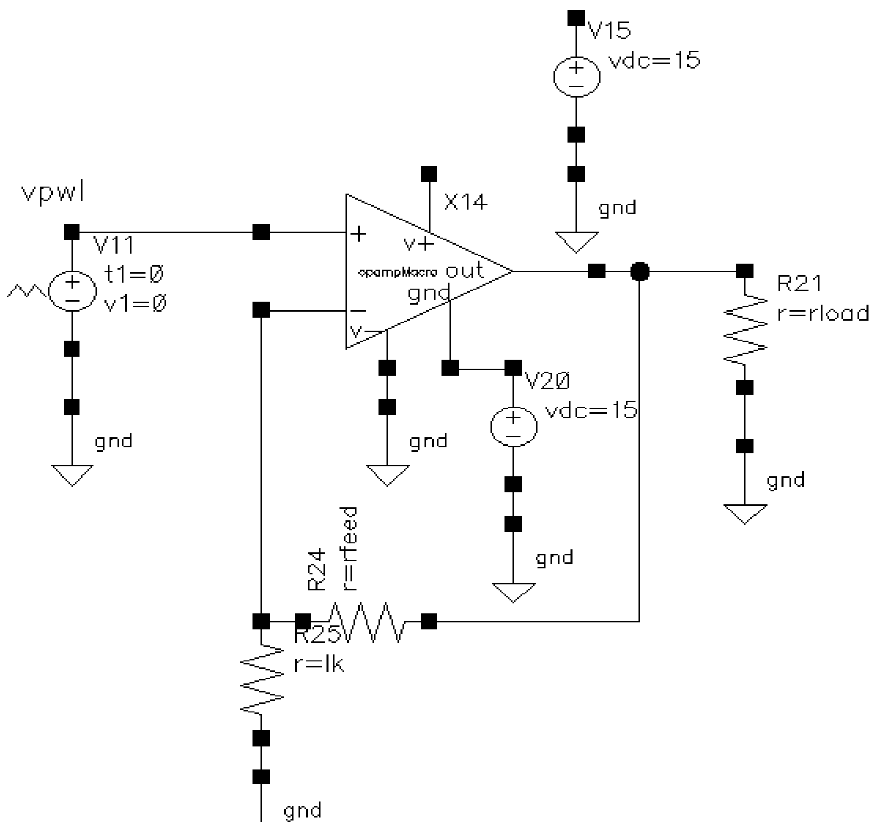
    - TSD<<(1/GBW)
    - FF2<<GBW

    In these situations, measure the simulated macromodel response and adjust the parameters accordingly.

- Additional parameter requirements
  - The slew rates must conform to the inequality $0.1 \leq srn/srp \leq 10$
  - The output DC and AC resistances must conform to the inequality rod > roa > 0

Example

Below is a schematic of a simple noninverting amplifier with closed-loop DC gain = rfeed/(rfeed + 1K).



The macromodel property list values are the typical values listed in <u>"Setting Macromodel Parameters"</u> on page 73.