Analog Behavioral Modeling

Applications

A collection of application notes on maximizing PSpice analog behavioral modeling capabilities.

table of contents

Preface

This book is a collection of application notes which are focused on the subject of using PSpice's Analog Behavioral Modeling (ABM) features to model specific devices or characteristics in simulations. Some of these application notes are old standbys, and some are newer; they all contain relevant and useful examples that demonstrate the flexibility and capabilities of ABM. Addenda have been appended to application notes to provide clarifications as necessary.

Analog Behavioral Modeling Using PSpice

The following is an excerpt from a previously published paper.

Introduction

Behavioral Modeling is the process of developing a model for a device or system component from the viewpoint of externally observed behavior rather than from a microscopic description.

Two important applications of Behavioral Modeling in the domain of analog simulation are: modeling new device types; and black-box modeling of complex systems.

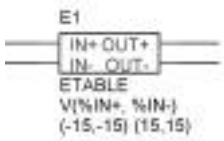
This paper discusses extensions made to PSpice to support these applications and presents detailed examples of each.

A summary of PSpice extensions, together with some simple examples, is given below.

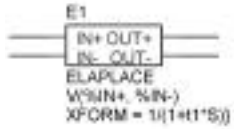
Time domain

- arbitrary expressions, can include constants, parameters, node voltages & currents, TIME, math functions including log, exp, and trig

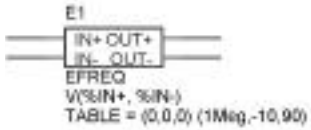
- table lookups; value of a controlling expression is linearly interpolated in a table



- Laplace expressions, including constants, parameters, and math functions in S including log, exp, and trig



- table lookups; magnitude and phase are linearly interpolated in a table



Device Modeling

Modeling Tunnel Diode

The tunnel diode has frequently been used as an example of SPICE device modeling using polynomials. The static current/voltage characteristic of the device contains a region of negative dynamic resistance. The transitions from positive to negative resistance and back again are smooth - there are no discontinuities in slope and the device does not exhibit hysteresis. The device is only operated in the vicinity of the negative resistance region; typically a span of one or two volts.

These attributes make the device eminently suitable for polynomial representation (it is no coincidence that this device has been used for illustration so often in the past).

Main characteristics of a tunnel diode current/voltage curve are peak voltage and current (Vp, Ip), valley voltage and current (Vv, Iv) and projected peak voltage (Vpp). Specific device parameters for this example:

```
Vp=50mv;Ip=4.2ma;Vv=370mv;Iv=370ua;Vpp=525mv
```

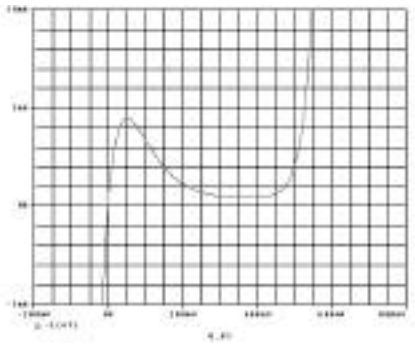


Figure1. Tunnel diode characteristics

Using Behavioral Modeling: Current flow in a tunnel diode is due to three distinct effects [1]: thermal current (analogous to a conventional diode), tunnel current (due to direct tunneling) and excess current (due to indirect tunneling). Writing these three terms in PSpice's extended syntax:

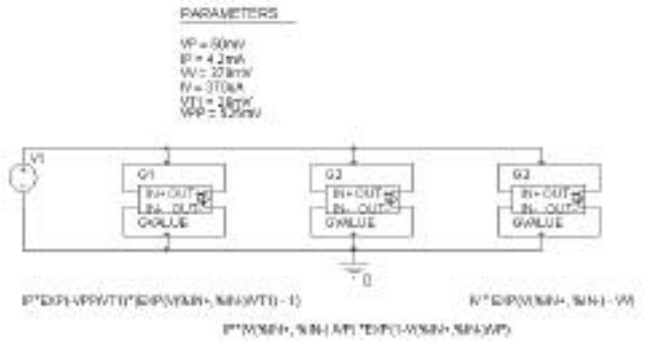


Figure 2. Behavioral model for a tunnel diode

Parameterization

Consider modeling devices with parameters different from the example set used above, for example to produce a library of devices for general use. The polynomial approach would require a set of coefficients for each distinct device. This becomes impractical for anything more than a handful of devices. It may be possible to define a "generic" tunnel diode device and map inputs and outputs appropriately, but it is not intuitively obvious what mapping to use.

The functional approach is much better suited to setting up libraries of devices owing to the presence of parameters in the equations. To model a device with a different value for Vp, for example, only that parameter's value needs to be updated. Note also that in the basic equations above, the temperature dependence is included (Vt). A sub-circuit definition can be used to package the tunnel diode model and its parameters:

```
.SUBCKTT Dak PARAMS:Vp=50mv Ip=5ma Vv=0.3v Iv=0.3ma Vpp=500mv
Gthermal a k....
Gtunnel a k....
Gexcess a k....
.ENDS
usage:
       X1 4 5 TD PARAMS:Vp=55mv ; override 50mv default
```

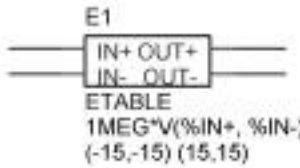
For more difficult devices, where straightforward equations may not be available, or where the relationship between the parameters in the equations and data sheet values for the device is not obvious, a lookup table approach may be used. Where possible, a normalized device characteristic can be modeled by the table, with parameterized expressions used to transform inputs and outputs.

System Modeling

Behavioral Modeling as Abstraction

In the early stages of system design, the emphasis is on high-level issues rather than on low-level details. Behavioral models allow systems to be simulated with reduced complexity and with improved computational efficiency.

A simple example is using a controlled source as a gain block rather than using a complete operational amplifier model:



PSpice extensions allow black-box simulation of many high-level circuit elements. The use of arbitrary expressions, lookup tables and Laplace formulations are powerful tools.

Modeling a Phase-Locked Loop

To contrast the high-level and low-level modeling approaches, consider a simple phase-locked loop (Figure 3). Phase locked loops contain three major components: a voltage-controlled oscillator (VCO); a Phase Detector which compares the output of the VCO with the input (target) signal to derive an error signal; and a Loop Filter. The inverted output of the Loop Filter becomes the controlling voltage for the VCO, thus forming a negative feedback control loop.

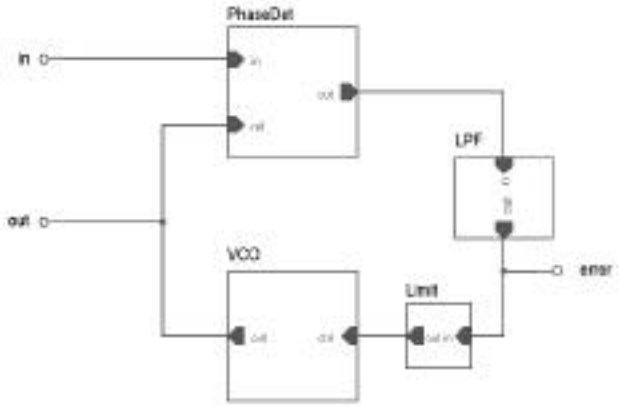


Figure 3. Phase-locked loop

Mathematical Description: The general time-domain equation for a phase-locked loop can be written as:

$$\varphi o' = K \sin[\varphi i(t) - \varphi o(t)] \Theta f(t)$$

The input signal yi and the VCO output signal yo are given by:

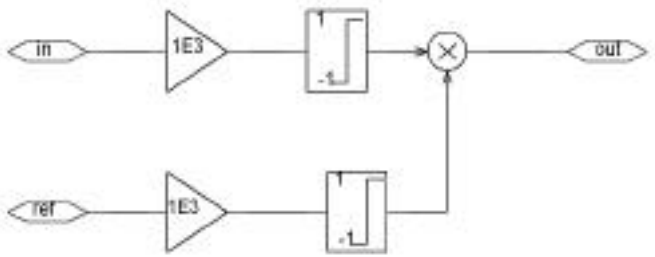
$$yi(t) = A \sin[wt + \varphi i(t)] \quad yo(t) = B \sin[wt + \varphi o(t)],$$

The symbol Θ represents convolution, and f(t) is the impulse response of the filter.

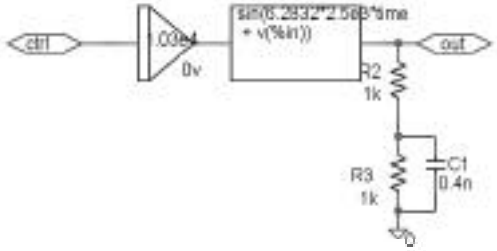
This nonlinear differential equation is not solvable in the general case. Approximate solutions may be found when the equation is linearized. The typical case where the loop filter is a simple RC network when linearized gives rise to a second-order linear differential equation.

Behavioral Model: Each of the three main components of the PLL can be expressed succinctly in PSpice's extended Behavioral Modeling syntax.

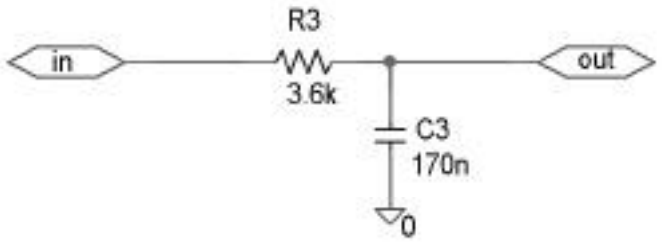
The Phase Detector is a multiplier with the output range constrained to [-1,+1]. It uses gain blocks, limiters, and a multiplier.



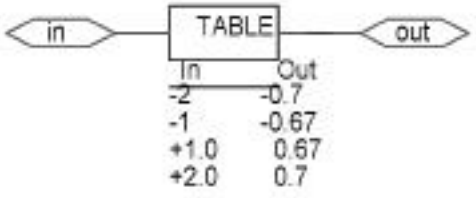
The VCO is described as a sinusoidal function of time with an additional term controlling the phase:



The Loop filter is single pole RC. The output of the loop filter is the error signal. The Loop Filter could also be conveniently described by giving its Laplace Transform, using s-domain notation.



The limiter block is a non-linear table, which shifts levels for the VCO input.



A complete phase-locked loop description consists of these four "devices," along with a test stimulus (a VCO).

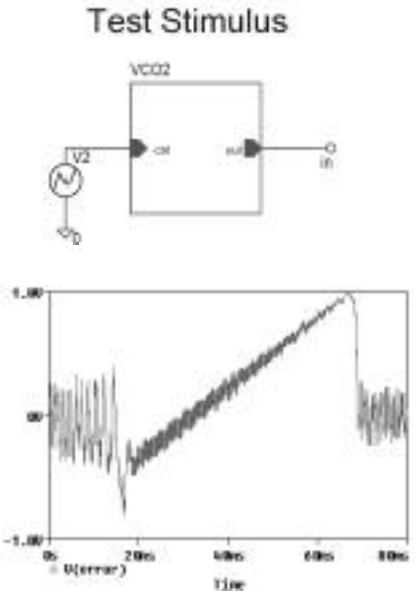


Figure 4. Phase-locked loop transient waveform

A transient analysis is run to analyze capture and lock ranges. The PLL does not respond to the input signal until the input frequency reaches the lower edge of the capture range. Up to the lower capture frequency, the VCO output frequency does not change significantly since the average voltage of the beat frequency is zero. The model then suddenly locks to the input signal, causing a negative jump in loop error voltages. Here we see the nonlinear capture transients.

Once locked the error signal tracks small frequency changes of the input signal by generating additional phase error between the VCO and the input signal. This signal is converted to the DC error voltage by the phase detector and low-pass filter.

At the upper lock range frequency, the model loses lock, and the error voltage suddenly drops. and then becomes a beat note.

Circuit Level Model: A model of the same PLL was developed using bipolar transistor circuits.

The VCO was an astabel multivibrator with the charging current proportional to the VCO control voltage. The multiplier was a double-balanced modulor using 6 BJTs. The Loop Filter consisted of two resistors and a capacitor.

A complete description consists of these circuit fragments, together with power supplies, bias resistors, bypass capacitors, etc.

Comparing the two Approaches: Table 1 contrasts the two approaches to modeling the PLL.

Compared with the Circuit model, it took about 20% of the time to develop the Behavioral model, and the transient analysis ran in about 4% of the time.

The time required to run the analysis is significant. The Behavioral model allows many more analyses to be run in a given time, permitting a higher degree of design refinement and/or test.

Table 1. Comparison of Modeling Approaches

Model	Behavior	Circuit
Devt. Time	1 day	5 days
Simulation Time	24 sec*	606 sec*
* lines	9	43

* run times measured on a Sun 4/110

Managing Convergence and Time-Step Control

Behavioral models are not restricted to well-behaved devices like the tunnel diode. Devices with abrupt behavior can be readily modeled using the TABLE forms and the logarithmic and exponential functions. Convergence control to ensure that the simulator takes small enough steps may be necessary in these models. For TABLE devices this can be implemented by setting the internal non-convergence indicator if an attempt is made to skip from a section of the TABLE device to another section that is not an immediate neighbor.

For other forms the proposed output at a given time step could be compared with the previous output and absolute and/or relative delta criteria applied. If the test failed, the time step would be reduced. The criteria could be specified per device, with global default values.

Controlling the time step may be necessary not only for convergence, but also from sampling considerations. Interpolation schemes are used for graphical display of the simulation results. The time step must be constrained so that voltage/current changes are within the scope of the interpolator.

It is not possible to deduce the frequency domain behavior of devices specified by arbitrary expressions. There is a risk of aliasing occurring if the initial and subsequent choices of timestep produce "reasonable" (but subsampled) values of a periodic function. For example, suppose there is a 1 MHz source in the circuit and the initial time step is chosen as 10 μS. If each subsequent time step is also 10 μS, the apparent value of the source will be 0.

In practice, this kind of subsampling will be readily noticed. It can be avoided by manually setting the step ceiling.

Addendum

Since the publication of this paper, University of California at Berkeley has introduced several newer versions of SPICE. The latest of these supports behavioral modeling constructs that provide some of the capabilities offered by PSpice, but is not backward compatible with the

SPICE2G6 polynomial controlled source syntax which is still used by many device vendors in their SPICE models.

PSpice has also been considerably enhanced: many more math operations are now available for use in behavioral modeling expressions, including an if-then-else function; the convergence algorithms have been improved to better handle the unusual behavior that can be specified given the flexibility of behavioral modeling; the frequency domain expressions are now more flexible and more robust when simulated in the time domain; etc.

References

[1] S. M. Sze, *Physics of Semiconductor Devices*, Wiley & Sons, 1981, ch. 9, p529.

Analog Behavioral Modeling

Let's take a look at examples of how the Analog Behavioral Modeling feature of PSpice can cope when generic SPICE fails.

Note: In this document generic SPICE refers to SPICE2G6. Please see the addendum to the previous application note for further clarification.

First, let's say you need to create a signal whose voltage is the square root of another signal's voltage. Calculating square roots is simple, even for SPICE, through the use of a feedback circuit. However, this technique fails if the reference signal ever goes negative. In this case the functional form of Analog Behavioral Modeling works nicely:

```
Esqrtout_hiout_lovalue={sqrt(abs(v(input)))}
```

This takes the absolute value of the ground-referenced signal "input" before evaluating the square-root function (you could also use a floating signal-pair by replacing v(input) with v(in_hi)-v(in_lo) or v(in_hi,in_lo), for example). The absolute-value function is a nonlinear function difficult to perform in generic SPICE.

We can also introduce ideal nonlinearities using the table lookup form of Analog Behavioral Modeling. For example, the one-line, ideal opamp model:

```
Eampout0table {200K*(v(in_hi)-v(in_lo))}=
       +(-15,-15)(15,15)
```

has high gain, but its output is clamped between ±15 volts. The input to the table is the differential gain formula, but the lookup table has only two entries: so the output of the table is interpolated between these two endpoints and clamped when the input exceeds the table's range. This is a convenient use of the table lookup form, which is not available in generic SPICE.

Small systems of behavioral models are easy to design, also. For example, a true-RMS circuit can be built by decomposing the RMS function:

(i) square the signal, (ii) integrate over time, and (iii) take the square-root of the time average. These three operations can be bundled in a tiny subcircuit for use as a module:

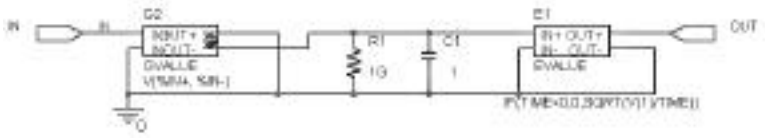


Figure 1. RMS subcircuit

```
.subckt RMS in out
G1 0 1 VALUE {V(IN)*V(IN)}
C1 1 0 1
R1 1 0 1G
E1 out 0 VALUE {IF(TIME<=0, 0, SQRT(V(1)/TIME))}
.ends
```

The current source, G1, squares the signal, which is then integrated in the capacitor. The voltage on the capacitor is time averaged, and the square-root is taken (the resistor is a dummy load that satisfies the SPICE algorithms). The voltage source E1 shows that the value of simulated "time" is available in Analog Behavioral Modeling, and may

be used as a variable in a formula. Notice that the if-than-else function is used. If time is less than or equal to zero then the output of E1 is 0. This prevents convergence problems when sqrt(v(1)/time) is evaluated at time = 0. If time is greater than zero then the output of E1 is sqrt(v(1)/time).

Parameter passing into subcircuits also works with Analog Behavioral Modeling, which makes your models more flexible. Here is a small system that is a voltage follower with hysteresis, which would be useful in simulating, say, a mechanical system with gear backlash:

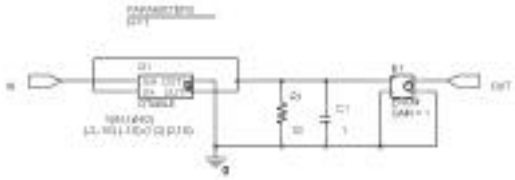


Figure 2. Hysteresis subcircuit

```
.subckt HYS in out params: H=1
G1 0 1 TABLE {V(IN,1)/(H/2)} (-2,-1G) (-1,0) (1,0) (2,1G)
C1 1 0 1
R1 1 0 1G
E1 out 0 1 0 1
.ends
```

The parameter H defines the size of the hysteresis, and is used in the formula input to the table. The combination of the formula and table defines a dead-band outside of which the output follows the input with an offset of H/2. The capacitor serves as memory for the circuit and is nearly ideal except for the DC-bias resistor, which provides a droop time constant of one billion(!) seconds. The voltage follower, E1, prevents output loading problems. E1 could also have gain representing the gear ratio of a mechanical system; then voltage would represent the total turn angle of each gear, and H the amount of angular backlash.

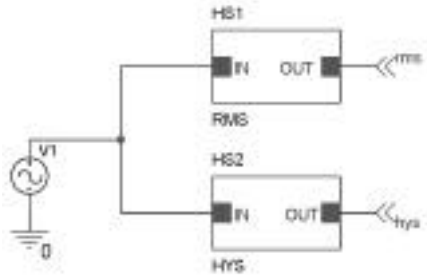


Figure 3. Circuit using RMS and hysteresis subcircuits

```
.param H=1
*
V1 in 0 SIN (0 1 1)
Xrms 1 rms RMSXhys 1 hys HYS
param: H=1
*
.tran 10m 1
.end
```

A1 Hz sine wave was used for the stimulus to the RMS and HYS circuits.

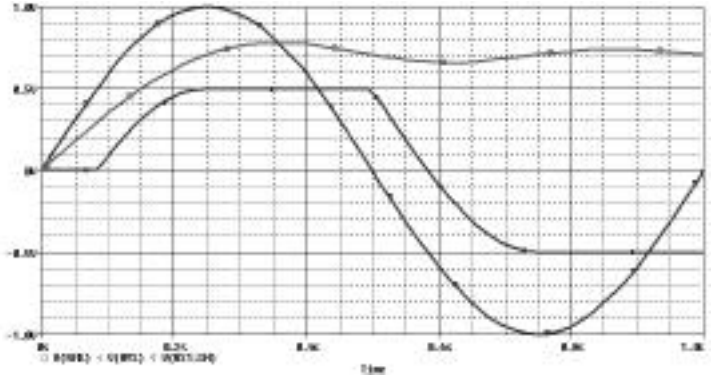


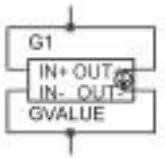
Figure 4. Output from RMS and HYS circuits

In **Figure 4** we see a Probe plot of the input, and the outputs from each circuit. Note that the RMS circuit outputs the well-known result of 0.707 volts after one input cycle, while the HYS circuit lags the input by a half volt in each direction for a total hysteresis of one volt. Perhaps these examples will give you ideas for other functions which would be "most difficult" to create with generic SPICE.

Creating Impedances with Behavioral Modeling

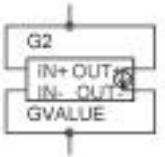
MicroSim Corporation Newsletter, October 1990

We regularly receive questions on how to create nonlinear resistors with the Analog Behavioral Modeling feature. The method for doing this can be illustrated by creating the transfer function for a linear conductance. A conductance can be thought of as a voltage-controlled current source: the current between its nodes is a constant, times the voltage across those same nodes. For example:



V(%IN+, %IN-)/1000

is a linear conductance with a value of 1 milli-mho (i.e., a 1 kilo-ohm resistor). The controlling nodes are the same as the output nodes. For a nonlinear conductance the appropriate nonlinear function is used, but the device still has the same controlling and output nodes:

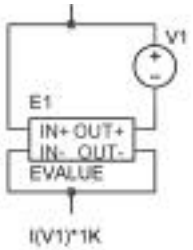


PWRS(V(%IN+, %IN-),3)*0.001

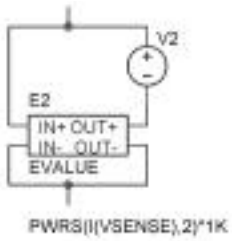
G2 has a small-signal conductance of $3 \cdot .001 \cdot V(IN+,IN-)^2$. (The small-signal conductance is the derivative of the transfer function.)

Any nonlinear resistance can be expressed as a nonlinear conductance by inverting the transfer function. Sometimes, however, it is convenient to implement it directly. This can be done by noting that a resistor is a current-controlled voltage source.

For example:



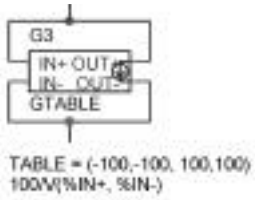
E1 is a linear resistor with a value of 1 kilo-ohm. V1 is needed to measure the current through E1. A quadratic resistor is then:



The PWRS (signed power) function is used instead of $I(V2)^2$ because we want the sign of the voltage across E2 to become negative when the current through V2 is negative.

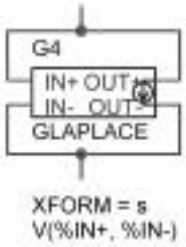
There are a couple of things to watch for when creating nonlinear devices this way. First, all physical impedances have zero current at zero voltage.

Second, one needs to be careful of the asymptotic behavior of the device. It is very easy to create devices which generate power at high voltages. Even though the real circuit may not operate at such voltages, there is nothing to prevent PSpice from finding an unrealistic solution at a high voltage. In general, it is good practice to use the TABLE form to limit the output of devices. For example, here is a constant-power load:

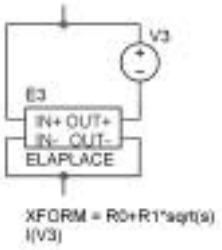


G3 tries to dissipate 100 watts of power regardless of the voltage across it. For very small voltages the formula 100/V(IN+,IN-) can lead to unreasonable values of current. The TABLE limits the current to be between -100 and +100 amps.

This approach can also be used to create frequency-dependent impedances. The main difference is that the LAPLACE or FREQ type is used. For example, a capacitor can be written as:



The current through G4 is the integral of V(IN+,IN-). However, the LAPLACE device uses much more computer time and memory than does the built-in capacitor (C) device. We recommend the LAPLACE form only for cases where its flexibility is needed. Note that, in general, frequency-dependent impedances have varying phases as well as varying magnitudes of impedance. For example, the formula for a wire with skin effect is:



The wire's impedance is constant (and real) at low frequencies. At high frequencies, its impedance behaves as the square root of the frequency and becomes half real and half imaginary (i.e., it becomes half inductive).

Filter Models Implemented with ABM

by Bashir Al-Hashimi, PhD, School of Engineering, Staffordshire University, Stafford, England

Analog behavior modeling (ABM) allows the simulation of analog circuits using mathematical equations. This article shows how filter behavioral models are developed and implemented using the Laplace function of PSpice. Given the filter bandwidth and order, the models simulate lowpass, highpass, bandpass, and band-reject filters. For ease of use, the models are developed as parameterized subcircuits. Simulation examples are included to demonstrate the use of these models.

Introduction

Filters are often described in terms of a number of parameters including type, order, and response. There are four filter types:

- Lowpass
- Highpass
- Bandpass
- Band-reject

The order of the filter usually determines the amount of attenuation the filter provides—the higher the order, the more the attenuation. There are a number of filtering responses available. The most commonly used are Butterworth, Chebyshev, and Bessel. Each response has its advantages and disadvantages.

The Butterworth response, for example, has a maximally flat magnitude passband, while the Chebyshev has steeper attenuation characteristics than the Butterworth. The Bessel has a linear phase response and therefore an excellent pulse response. More information on filters is available in the *Electronic Filter Design Handbook* by A.B. Williams. See reference [2].

Lowpass Filter Behavioral Models

A block diagram of a general lowpass filter is shown in Figure 1. The diagram consists of one first order and a number of second order sections, allowing different filter orders to be simulated. For example, connecting one first and two second-order sections yields a fifth-order filter. The overall voltage transfer function of the circuit shown in **Figure 1** is obtained by multiplying the transfer functions (TF) of the individual sections:

$$(V_{out}/V_{in})=(1st\text{-}orderTF)*(2nd\text{-}orderTF)^N$$

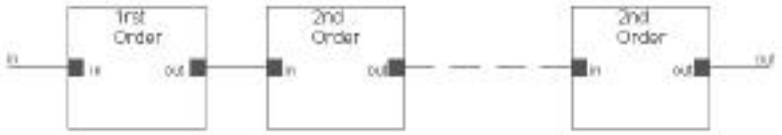


Figure 1. Block-Diagram of General Lowpass Filter

The first- and second-order section transfer functions, H(s), are

$$H(s)=x/(s+x);\ x=2\pi Fc\alpha$$
$$H(s)=x^2/(s^2+(x/Q)s+x^2);\ x=2\pi Fc\varpi_0$$

where s is the Laplace variable, and Fc is the filter cutoff frequency or bandwidth. The parameters α, ϖ_0, and Q define the pole positions of the various filtering responses (Butterworth, Chebyshev, and Bessel). To simulate the three different responses, it is necessary to define three models, one for each response.

Figure 2 shows the behavioral Butterworth lowpass filter model.

```
* Butterworth lowpass filter model
.subckt Butt_LP 1 2 params: Fc=1 ord=1
.func lp_1(x) {x/(s+x)}                    ; 1st-order lowpass
filter TF
.func lp_2(x,Q) {(x*x)/(s*s+(x/Q)*s+x*x)} ; 2nd-order filter TF
*
* a1-a4 and b1-b4 determine which filter sections are selected
* given the filter order
*
.param a1={table(ord,1,1,2,0,3,1,4,0,5,1,6,0,7,1,8,0,9,1)}
.param a2={stp(ord-1.5)} a3={stp(ord-3.5)} a4={stp(ord-5.5)}
+ a5={stp(ord-7.5)}
.param b1={1-a1} b2={1-a2} b3={1-a3} b4={1-a4} b5={1-a5}
*
```

```
* alpha, omega and Q values
*
.param alpha_b={table(ord,1,1,2,0,3,1,4,0,5,1,6,0,7,1,8,0,9,1)}
.param omega1_b={table(ord,1,0,2,1,3,1,4,1,5,1,6,1,7,1,8,1,9,1)}
.param Q1_b={table(ord,1,0,2,0.707,3,1,4,1.307,5,1.618,6,1.932
+ ,7,2.247,8,2.564,9,0.532)}
.param omega2_b={table(ord,3,0,4,1,5,1,6,1,7,1,8,1,9,1)}
.param
Q2_b={table(ord,3,0,4,0.541,5,0.618,6,0.707,7,0.802,8,0.90,9,0.653)}
.param omega3_b={table(ord,5,0,6,1,7,1,
+ 8,1,9,1)}
.param Q3_b={table(ord,5,0,6,0.518,7,0.555,8,0.601,9,1)}
.param omega4_b={table(ord,7,0,8,1,9,1)}
.param Q4_b={table(ord,7,0,8,0.509,9,2.879)}
**
E1 2 0 laplace {V(1)}= {
+ (b1+a1*lp_1(2*pi*alpha_b*Fc))*
+ (b2+a2*lp_2(2*pi*omega1_b*Fc,Q1_b))*
+ (b3+a3*lp_2(2*pi*omega2_b*Fc,Q2_b))*
+ (b4+a4*lp_2(2*pi*omega3_b*Fc,Q3_b))*
+ (b5+a5*lp_2(2*pi*omega4_b*Fc,Q4_b))}
.ends Butt_LP
```

Figure 2. Behavioral Butterworth Lowpass Filter Model

The model implements the overall voltage transfer function of the filter using a controlled voltage source (E component) that has the Laplace description. Although the concept is general in that it allows an nth-order lowpass filter to be simulated, Figure 2 shows that the model is limited to simulating a maximum of a ninth-order filter, which is made up of one first-order section and four second-order sections. It is often considered that a ninth-order filter is adequate for most applications. The filter model can easily be increased to simulate filters greater than 9th order by adding second-order sections.

The transfer functions are specified through .FUNC statements. The parameters a1 through a4 and b1 through b4 determine which filter sections of the model are selected by using a simple selection algorithm, based on the order of the filter specified by the user. The selection algorithm is implemented using .PARAM statements as shown in the listing. The filter order is defined by the subcircuit parameter, ORD. Note that the PSpice function "stp," which is used in the model, describes a step function where stp(x) is 1 if x>0 and is 0 otherwise. The α, ϖ_0, and Q values of the Butterworth response are defined as lookup tables using

.PARAM statements. These values are looked up automatically given the order of the filter.

The model is described as a subcircuit called Butt_LP with the input at node "in" and the output at node "out" as shown in **Figure 3**.

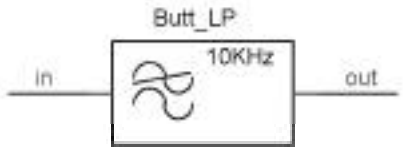


Figure 3. Behavioral Butterworth Lowpass Filter Symbol

The subcircuit has two parameters: the filter cutoff frequency (FC) and its order (ORD). These subcircuit parameters are given default values, which are arbitrarily set to 1. They will be changed to the required cut-off frequency and order when the subcircuit is called.

Similar Chebyshev and Bessel lowpass filter models can be easily developed. The Chebyshev filter model is called *Ch2p5_LP*, while the Bessel model is called *Bessel_LP*. Chebyshev filtering response exists for a range of passband ripple [1]. The ripple has been fixed at 0.25dB in the case of the *Ch2p5_LP* model. To develop Chebyshev models with different values of ripple, the values α, ϖ_0, and Q are needed for the required ripple, which are readily available [1]. Note that both of these models are capable of simulating up to ninth-order filters.

Example 1

To illustrate the use of the models, consider the following example. Here, the Butterworth lowpass filter model is used to obtain a family of curves for second- through ninth-order responses. Assume the filter has a cutoff frequency of 10 kHz. Using Orcad Capture, the circuit of Figure 4 is drawn.

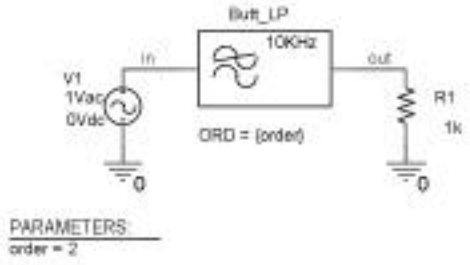


Figure 4. Butterworth Lowpass Filter Circuit

The filter Butt_LP has an AC source on its input and a 1K load resistor on its output. On the Butt_LP symbol, the attribute FC has a value of 10K, and ORD is set to the global variable defined in the global parameter block (PARAM). Finally, a VDB voltage marker is placed on the output node to view the results.

In Orcad Capture, be sure to configure the part library "filters.olb" (obtained from the ftp site) in the schematic Editor, and the model library "filters.lib" in the Libraries tab of the Simulation Settings.

The analysis consists of a 100 point per decade AC sweep and a parametric analysis. The parametric analysis steps the global parameter, ORD, from 2 to 9. The simulated frequency response of the filter for various orders is shown in **Figure 5**.

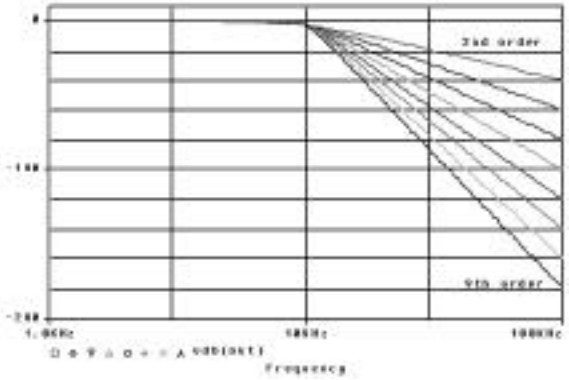


Figure 5. Butterworth Lowpass Filter Frequency Response for Various Orders

Highpass Filter Behavioral Models

Three highpass filter models are developed similarly. The three models are:

- Butt_HP (Butterworth filter)
- Ch2p5_HP (0.25dB ripple Chebyshev filter)
- Bessel_HP (Bessel filter)

Each model is capable of simulating up to a ninth-order filter and is used in a similar manner to that of the lowpass as described in **Example 1**.

Bandpass Filter Behavioral Models

One approach to obtain bandpass filters is to cascade lowpass and highpass circuits [2]. The cutoff frequency of the lowpass circuit defines the lower –3dB point of the bandpass filter bandwidth, while the cutoff frequency of the highpass filter defines the upper –3dB point of the filter bandwidth. A behavioral bandpass filter model representation is shown in Figure 6, based on this approach and by using the lowpass and highpass filter models discussed earlier.

Figure 6. Behavioral Butterworth Bandpass Filter Model

The three bandpass filter models are:

- Butt_BP (Butterworth filter)
- Ch2p5_BP (0.25dB ripple Chebyshev filter)
- Bessel_BP (Bessel filter)

Example 2

To demonstrate the use of the bandpass filter models, consider simulating a bandpass circuit with the following specifications:

lower –3dB point=1kHz
upper –3dB point=5kHz
30dB minimum at 0.3kHz and 20kHz

Assume a Butterworth response is required. To meet the specifications, third-order lowpass and highpass filters are required [2].

The circuit is shown in **Figure 7.**

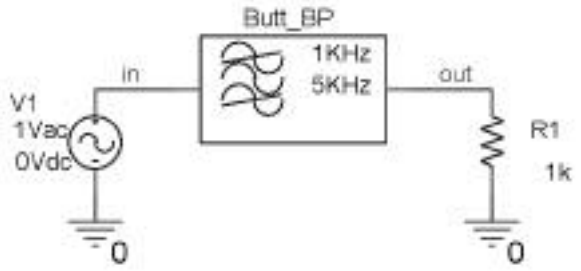


Figure 7. Butterworth Bandpass Filter Circuit

The *Butt_BP* filter symbol has attributes FCL=1kHz and FCH=5kHz for low and high cutoff frequencies, respectively. A single AC source is frequency swept over 0.25kHz to 50kHz (1000 points). The simulated frequency response of the filter is shown in **Figure 8.**

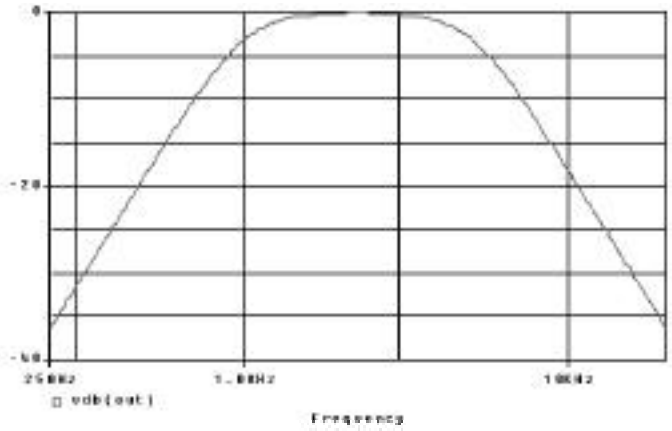


Figure 8. Butterworth Bandpass Filter Frequency Response

Band-Reject Filter Behavioral Models

Figure 9 is a block diagram representation of how a band-reject filter can be realized [2].

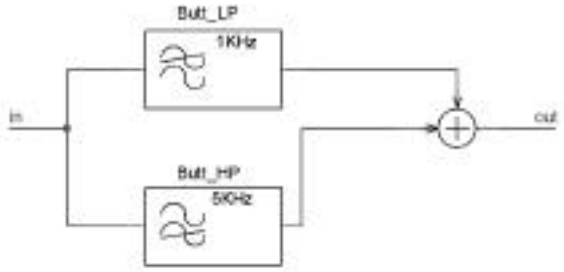


Figure 9. Butterworth Band-Reject Filter Behavioral Model

This model is based on summing outputs of the lowpass and highpass filter models. The cutoff frequency of the lowpass circuit defines the lower –3dB point of the bandpass filter bandwidth, while the cutoff frequency of the highpass circuit defines the upper –3dB point of the filter bandwidth.

Three band-reject filter models are contained in "filters.lib." Each sub-circuit has three parameters, FCL, FCH, and ORD. The three models are:

- Butt_BR (Butterworth filter)
- Ch2p5_BR (0.25dB ripple Chebyshev filter)
- Bessel_BR (Bessel filter)

Example 3

Figure 10 contains a fifth-order, .25dB ripple Chebyshv band-reject filter with a lower -3dB point at 1kHz and th eupper -3dB point at 5kHz.

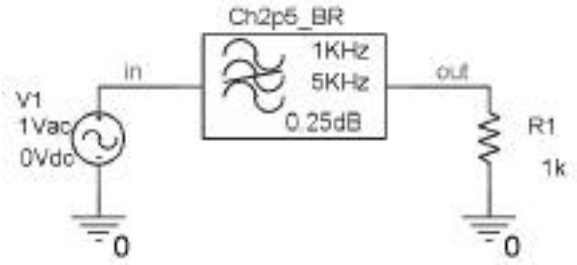


Figure 10. Chebyshev Band-Reject Filter Circuit

The simulated frequency response is shown in **Figure 11**.

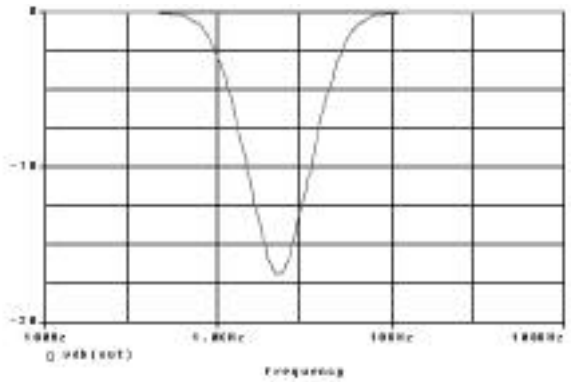


Figure 11. Chebyshev Band-Reject Filter Frequency Response

Library Availability

The symbol and model libraries used in this article are contained in a self-extracting zip file which can be downloaded from ftp://ftp.orcad.com/dwn_file/Pspice/Tech_support/filters.exe

References

[1] Van Valkenburg, M.E., *Analog Filter Design*, Holt, Rinehart & Winston, New York, 1982.

[2] Williams, A.B., *Electronic Filter Design Handbook*, McGraw Hill Book Company, USA, 1981.

[3] Al-Hashimi, B.M. *The Art of Simulation Using PSpice: Analog & Digital*, CRC Press, USA, 1995.

Voltage-Controlled Oscillators

In this discussion, let's take a look at modeling Voltage Controlled Oscillators (VCOs), and see how several different VCOs can be modeled using PSpice. Most of the examples use PSpice's Analog Behavioral Modeling (ABM) capabilities.

Sine Function VCO

A simple form of a VCO is obtained by starting with the time domain function for a sinusoidal source.

```
sin((twopi*fc*time)+phi)
```

In this ABM expression, twopi, fc and phi are all (constant) global parameters defined with a parameter block (PARAM part).

In order to obtain good resultution from a simulation using the above exression it will be necessary to set the maximum time step to be taken by the simulator. A simple approach is to set the time step ceiling for the Transient Analysis. For example, to view 10 cycles of a 1 MHz source, with 20 samples per cycle, a good choice for the step ceiling would be 50ns.

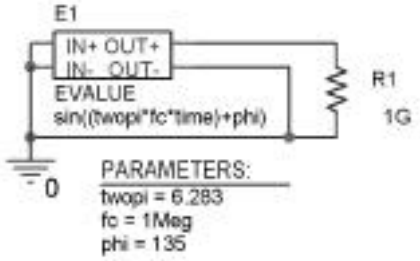


Figure 1. Simple VCO circuit

The single frequency source can be turned into a VCO by making phi a function of a controlling voltage instead of a constant:

$$y(t) = \sin(2\pi f_c t + \varphi(t))$$

The instantaneous frequency is given by the time derivative of total phase:

$$2\pi f_{inst} = 2\pi f_c + \varphi'(t)$$

The relationship between the frequency deviation, $f_d = f_{inst} - f_c$ and φ is given by:

$$\varphi(t) = \int 2\pi f_d(t)\, dt$$

For a linear VCO we want to be proportional to the controlling voltage, so:

$$\varphi(t) = 2\pi k_1 \int v_{ctrl}(t)\, dt$$

where k_1 is in Hertz/volt.

Using PSpice, the integrator can be modeled as a controlled current source plus a capacitor. The varying phase term is added into the controlled voltage (sine) source. To complete the example, a controlling voltage is required. Here is a piece-wise linear stimulus that starts at 0 v, remains at this level for 5 µs, then steps to 1 v and stays there:

```
pwl(0,0v5us,0v5.01us,1v)
```

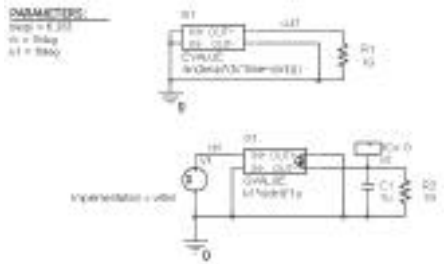


Figure 2. VCO with variable phase

Used with the VCO above and setting fc to 1 MHz and k1 to 1 MHz/volt gives an output signal that is 1 MHz for the first 5 µs and 2 MHz for the next 5 µs.

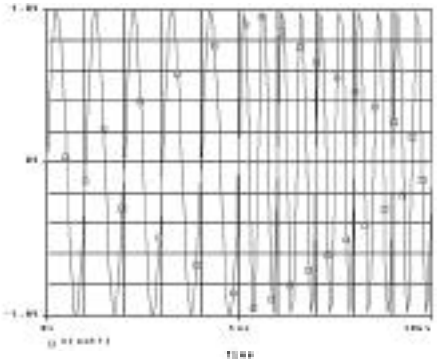


Figure 3. Output of variable phase VCO

Dual Integrator VCO

An alternate approach to a behavioral model VCO is to start from a 2-integrator loop. Changing the time constant of one or both integrators allows the frequency of oscillation to be controlled. Some form of limiting is required in order to produce output of bounded amplitude.

A particularly elegant example can be found in Reference[1]. This sinusoidal VCO has independent control of both frequency and amplitude. Its black-box representation reduces to:

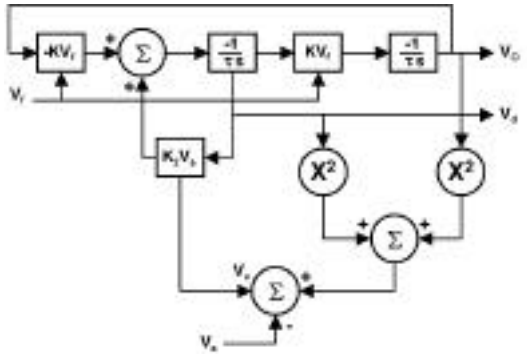


Figure 4. Dual integrator VCO

Outputs vd and vo are in quadrature. The amplitude error term is obtained by squaring and summing the quadrature outputs, and subtracting the amplitude-setting voltage Va.

With some sleight-of-hand and use of the .FUNC command,

```
.func ve(x,y,z)  {k*(pwr(x,2)+pwr(y,2)-pwr(z,2))}
```

the entire VCO can be reduced to two integrators with controlled current sources whose expressions incorporate the multiplications, additions and subtractions needed.

Parameter values of 0.1 for k, and 10 u for cv, give a VCO with frequency of 1.5 kHz per volt at vf and amplitude of 3.2 v peak at va = 1 v.

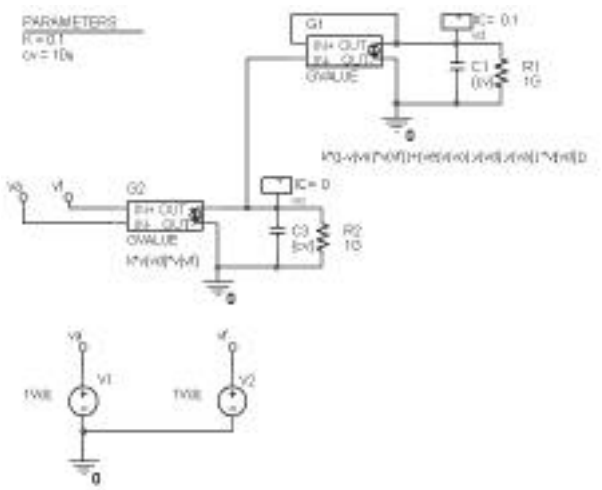


Figure 5. Filanovsky VCO

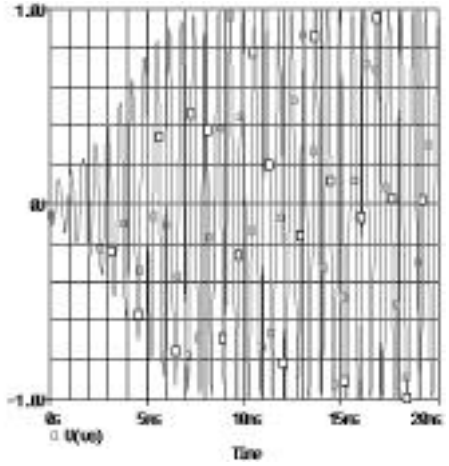


Figure 6. Output of Filanovsky VCO

Controlled Reactance VCO

A more circuit oriented approach to building a VCO is to take a simple tank oscillator and use a voltage-dependent reactance for one of the tank components. The example below is a Colpitts oscillator. The zx device (from ANL_MISC.LIB) is used to implement a voltage-controlled lossy inductor:

This simple VCO suffers from a number of drawbacks: variable output amplitude, square-root variation of frequency with control voltage, and slow start-up.

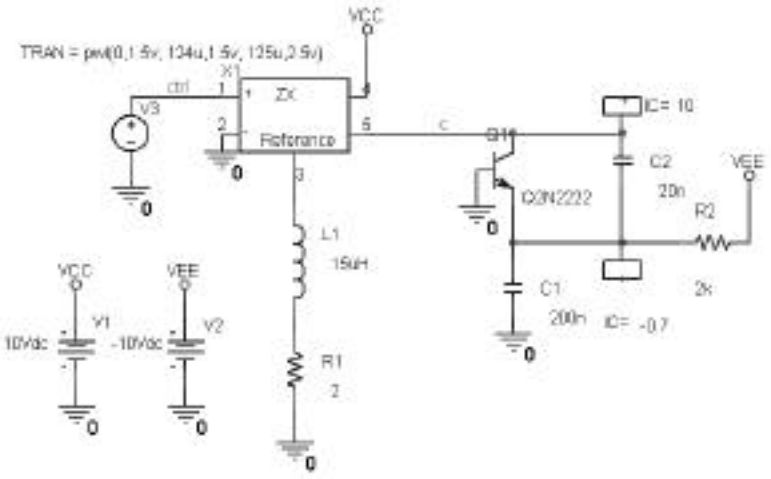


Figure 7. Controlled Reactance VCO

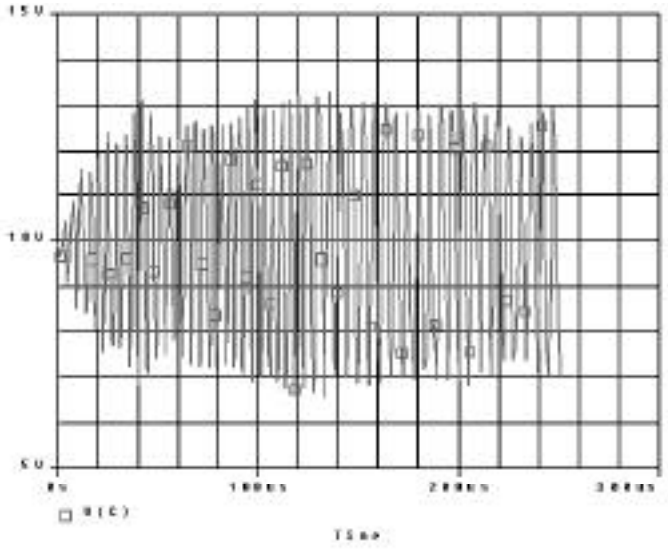


Figure 8. Output of Controlled Reactance VCO

Square Wave/Triangle Wave VCO

The previous examples have all been sine wave sources. If a square wave output is required, either a sine wave oscillator can be used and the output passed through a limiter (easily implemented in one line with a TABLE device), or a switching type oscillator can be used. The following example shows a current-steering VCO using ideal switches and current sources. Switches s1-s4 route the current flow around timing capacitor ct. A Schmitt trigger looks at the voltage across ct and reconfigures the switches for the next cycle.

There are many more ways of modeling VCOs using PSpice. Hopefully, the methods outlined above will be a useful starting point.

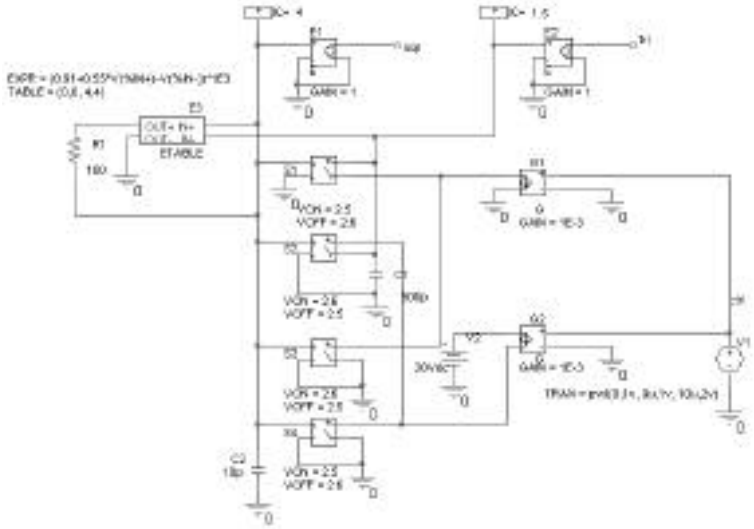


Figure 9. Current Steering VCO circuit

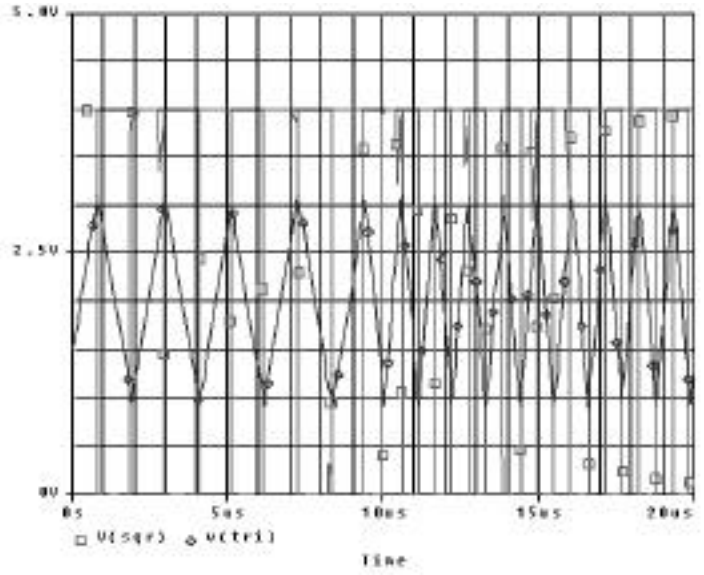


Figure 10. Output of Current Steering VCO

Voltage-Controlled Wien Bridge Oscillator

As a another example, we will use a voltage-controlled capacitor to adjust the frequency of oscillation of a Wien bridge oscillator.

First, a simplified operational amplifier is created using a voltage-controlled voltage source (an E device). A voltage divider network provides negative feedback to the amplifier. The closed-loop gain of the opamp must be at least 3 for oscillations to occur since the Wien bridge attenuates the output by 1/3 at the frequency of oscillation. The back-to-back Zener diodes limit the gain of the opamp as the oscillations build so that saturation does not occur.

As shown in **Figure 11**, the Wien bridge consists of two resistors and two voltage-controlled capacitors.

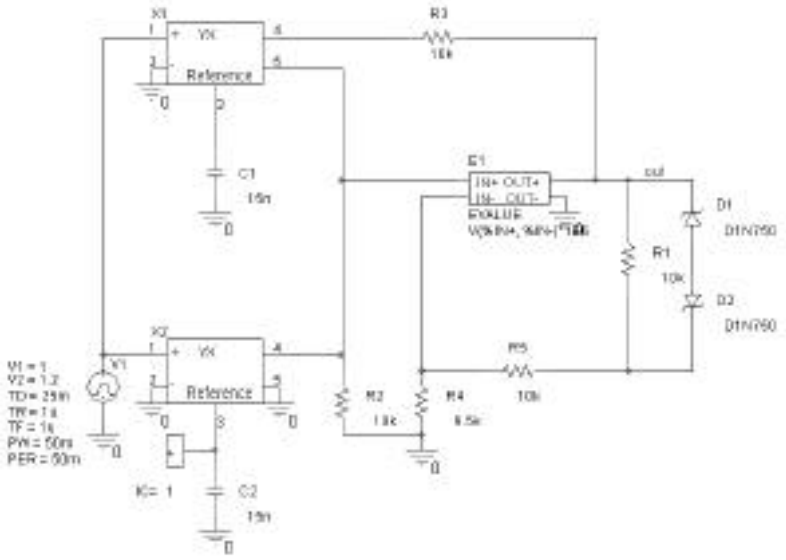


Figure 11. Frequency controllable Wien Bridge oscillator

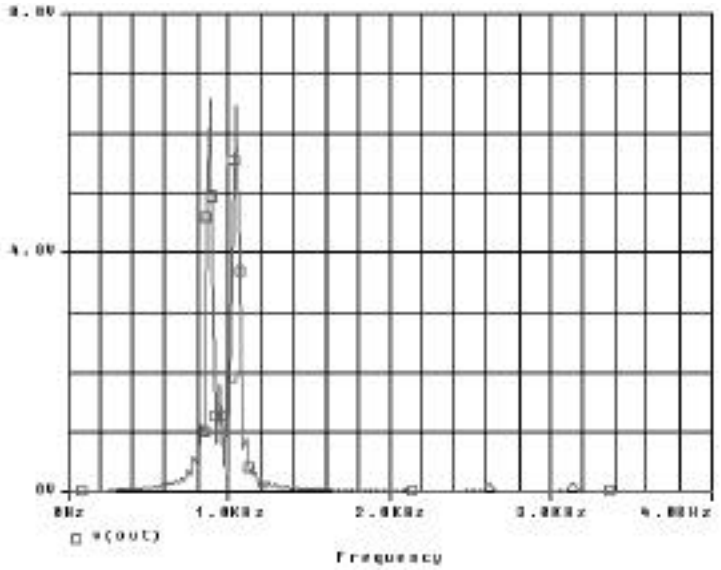


Figure 12. Frequency controllable Wien Bridge oscillator output

Each of these capacitors uses the YX subcircuit from the file ANL_MISC.LIB and its own reference capacitor. In this example 15nF capacitors are used.

The voltage control of the oscillations is given by V1 which is a pulse that moves from 1.0 volts to 1.2 volts, 25 ms into the run. This changes the admittance from that of a 15 nF capacitor to a 18 nF capacitor. The frequency of oscillation will then change. The initial condition on C2 (IC) causes PSpice to begin simulation with an IC of 1 volt on the capacitor so oscillation can begin.

Figure 12 shows the Fourier transform of voltage V(OUT), the output of the oscillator. Using this capability, we can easily see the transition from the first frequency to the second. The resonant frequency is given as $1/(2\pi*R*C*V1)$. The first frequency is 1/(6.28 * 10k * 15n * 1.0V) = 1kHz. The second frequency is 1/ (6.28 * 10k * 15n * 1.2V) = 0.886kHz. We can see the two peaks on the plot indicating the two resonant frequencies. It can also be noted that the period of the oscillations is proportional to the control voltage V1.

Reference

[1] I. M. Filanovsky, *Sinusoidal VCO with Control of Frequency and Amplitude*, Proceedings of the 32nd Symposium on Circuits and Systems, IEEE, Vol I, 446-449 (1989).