

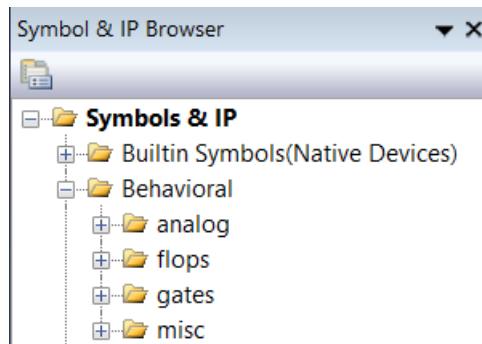
Qspice - Device Reference Guide by KSKelvin

KSKelvin Kelvin Leung

Created on 9-5-2023
Last Updated on 1-29-2026

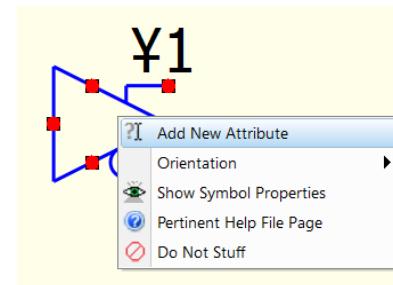
Device, Instance and Model Parameters

- Device
 - Device symbol in this guide can be found in Qspice : View > Symbol & IP Browser (shortcut F2)



- Detail description of Qspice device is in Help, Qspice > Simulator > Device Reference

- Instance and Model Parameters
 - A device normally consists of instance parameters and/or model statement
 - Instance parameters are described with text attribute. To add text attribute, right click device and select "Add New Attribute"

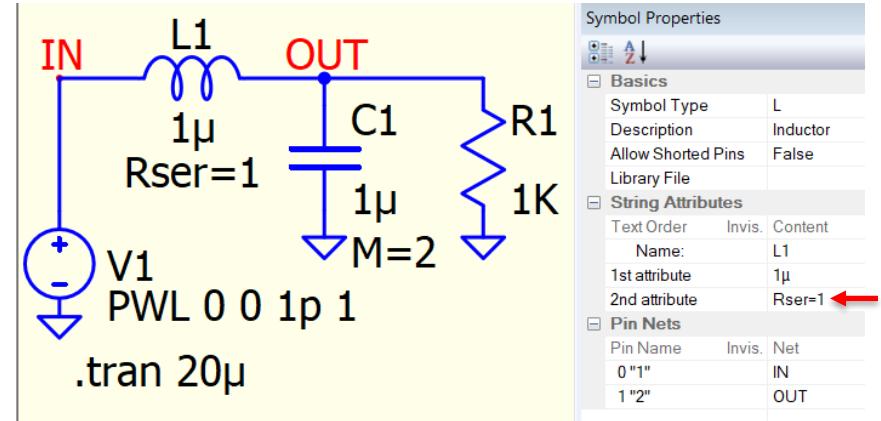


- .model statement with model parameters
 - To add .model statement, type shortcut "t" on schematic to place text
e.g. .model SW SW Ron=1 Roff=1Meg Vt=0 Vh=0

Instance Parameters

- Instance Parameters

- In this example, Rser for L1 and M for C1 are instance parameters
- Explanations of instance parameters can be found in the Qspice HELP section within each device's description
 - Many instance parameters come with default values and can be added by creating a new attribute for the device
 - A new attribute is a text string appended to the device in the circuit netlist
 - Attributes can be disabled by commenting out the attribute line (using the shortcut ;), which enables users to quickly assess the circuit's behavior with different device parameters

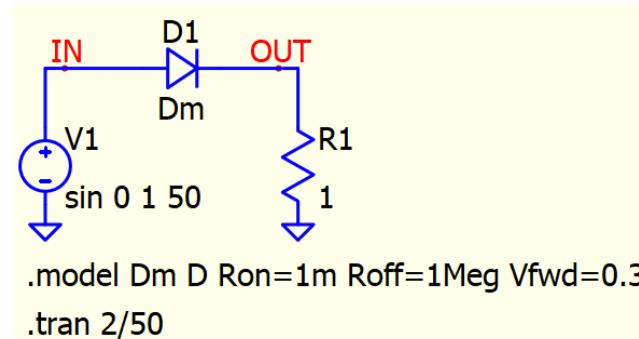


.tran 20μ

V1 IN 0 PWL 0 0 1p 1
C1 OUT 0 1μ M=2 ← Append attributes
L1 IN OUT 1μ Rser=1
R1 OUT 0 1K
.tran 20μ
.end

Model Parameters

- Model Parameters
 - In this example, $R_{on}=1m$, $R_{off}=1Meg$, and $V_{fwd}=0.3$ are model parameters specified in the .model statement
 - The .model statement configures devices that are simulated as native circuit elements
 - Explanations of model parameters can be found in the Qspice HELP section within each device's description
 - .model statement can be added directly in the netlist or linked to a library using the .lib statement
 - First attribute of a device is the model name, and if the .model statement is used, it also includes the model name



Symbol Properties		
A		
Basics		
Symbol Type	D	Silicon Diode
Description		
Allow Shorted Pins	False	
Library File		
String Attributes		
Text Order	Invis.	Content
Name:	D1	
1st attribute	Dm	
Pin Nets		
Pin Name	IN	Net
0 "A"		
1 "K"		

```
V1 IN 0 sin 0 1 50
D1 IN OUT Dm
R1 OUT 0 1
.model Dm D Ron=1m Roff=1Meg Vfwd=0.3
.tran 2/50
.end
```

B. Behavioral Source

B. Behavioral Source : Instance Parameters

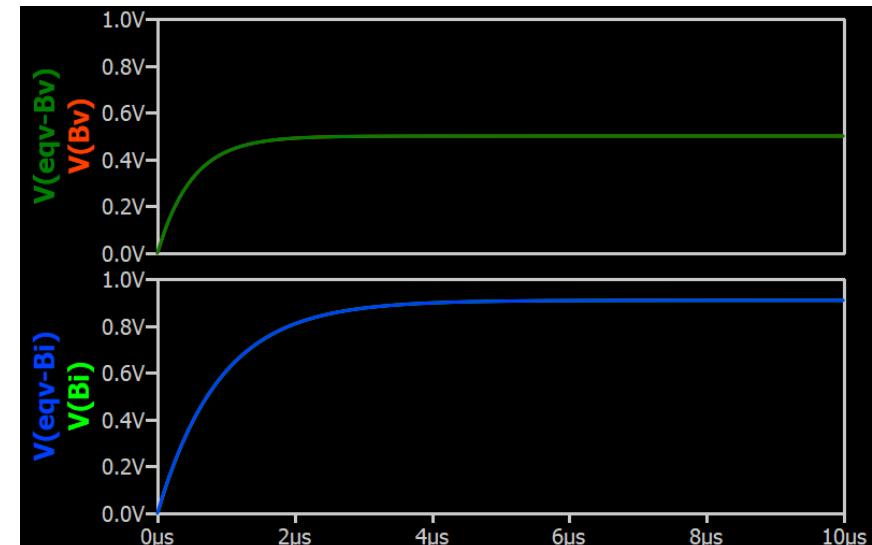
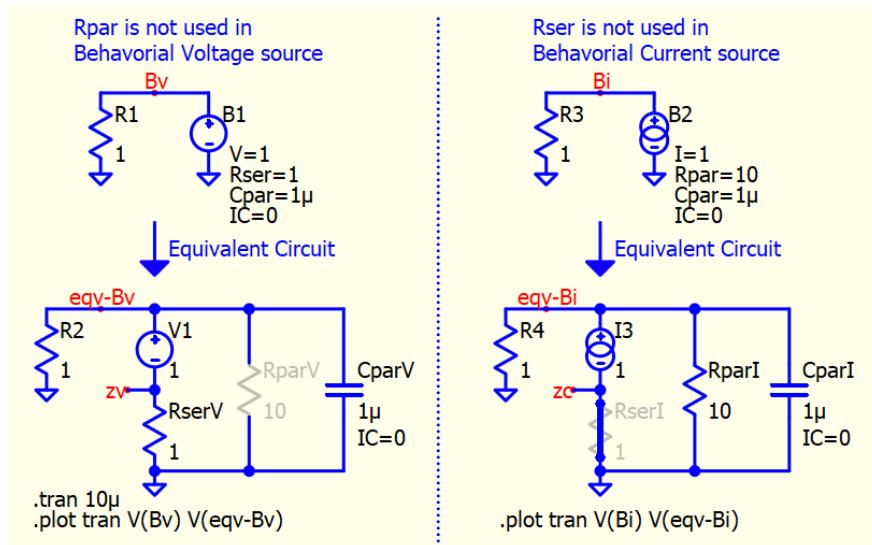
Behavioral Source Instance Parameters

Name	Description	Units	Default	
CPAR	Parallel capacitance	F	0.0	
I ¹	Expression for current	A	(not set)	
IC	Initial value	A or V	0.0	
LAPLACE	<u>Frequency dependence</u>		(none)	
NORTON	Use a Norton equivalent for a behavioral resistance		(not set)	
R ¹	Expression for resistance	Ω	(not set)	
RPAR	Ignore in V	Parallel resistance	Ω	Infinite
RSER	Ignore in I and R (Norton)	Series resistance	Ω	0.0
SYNTHESIZE	Use the synthesized Laplacian circuit even for .AC		(not set)	
THEVENIN	Use the Thévenin equivalent for a behavioral resistance		(not set)	
V ¹	Expression for voltage	V	(not set)	

B. Instance Params : Rser, Rpar, Cpar and IC

Qspice : B Source - Rpar Rser Cpar in V and I.qsch

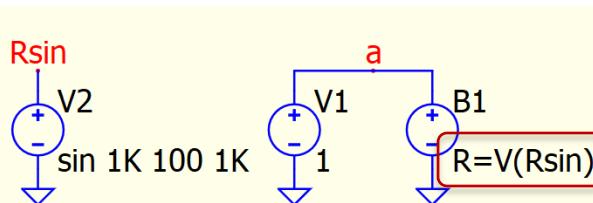
- Rser, Rpar and Cpar in Behavioral Voltage and Current Source
 - Behavioral Voltage ($V = <\text{expression}>$) : Only Rser and Cpar are available
 - Behavioral Current ($I = <\text{expression}>$) : Only Rpar and Cpar are available
 - IC initial value is used to set initial voltage to 0V for transient simulation. As behavioral source not have effect in .ac (except for Laplace), .tran is used to confirm equivalent circuit



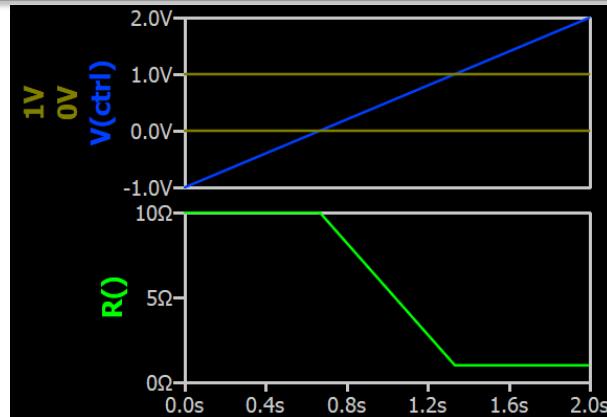
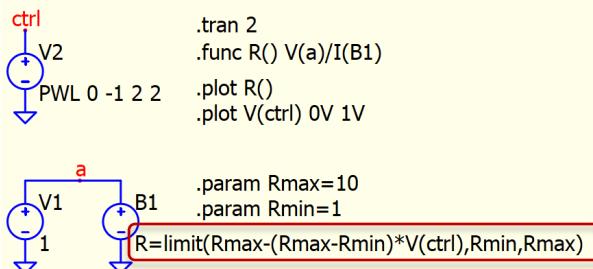
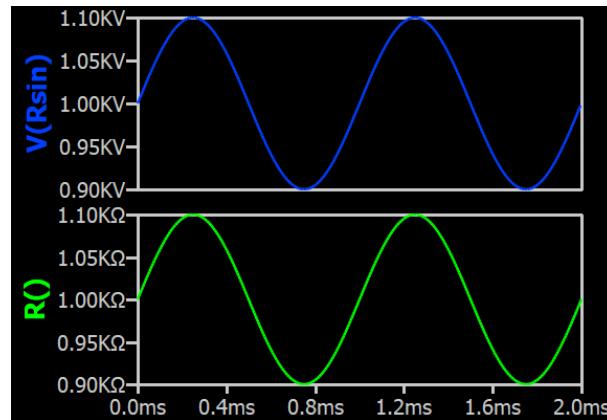
B. Instance Params : Behavioral Resistor

Qspice : B Source - Behavioral R.qsch

- Behavioral Resistor
 - Syntax : Bn nn n001 n002
 $R = <\text{expression}>$
 - Expression of resistance
 - When using a behavioral resistance, users can consider limiting the resistance to a certain value with the `limit()` function. This approach may reduce the chances of encountering convergence problems during simulation



```
.tran 2m
.func R() V(a)/I(B1)
.plot R()
.plot V(Rsin)
```



B. Instance Params : Behavioral Resistor with Laplace

Qspice : B Source - Behavioral R with Laplace (NORTON).qsch

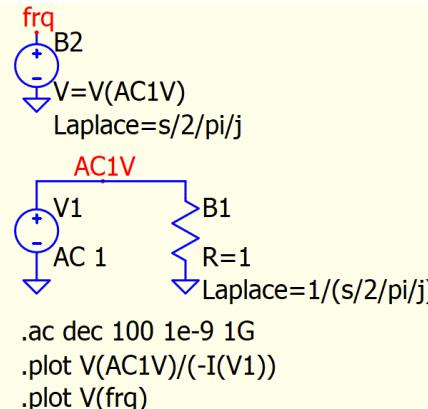
- Behavioral Resistor with Laplace

- This syntax is not specified in Qspice help, but such discussion can be traced back to LTspice

- It is observed that the impedance of a behavioral resistance is

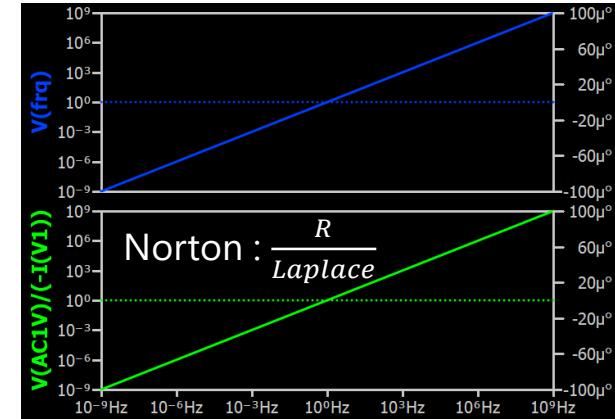
$$\frac{R}{\text{Laplace}}$$
 if in default

NORTON equivalent for a behavioral resistance



- Behavioral voltage B2

$$\begin{aligned} V(\text{frq}) &= \frac{s}{2\pi j} V(\text{AC1V}) \\ &= \frac{j2\pi f}{j} V(\text{AC1V}) \\ &= f V(\text{AC1V}) \end{aligned}$$



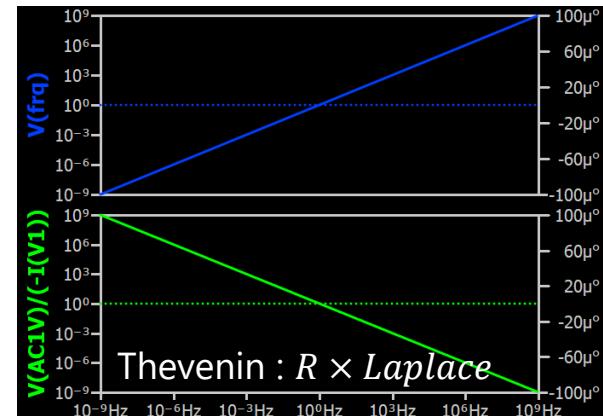
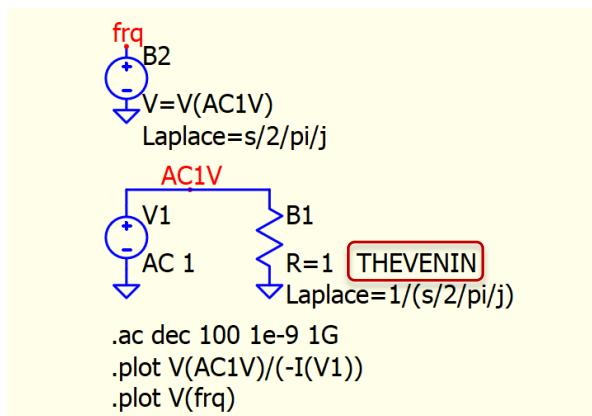
- Behavioral resistor B1

$$\frac{V(\text{AC1V})}{-I(V1)} = \frac{R}{\frac{1}{s}} = \frac{R}{\frac{1}{f}} = R_f$$

B. Instance Params : Behavioral Resistor with Laplace

Qspice : B Source - Behavioral R with Laplace (THEVENIN).qsch

- Behavioral Resistor with Laplace
 - It is observed that the impedance of a behavioral resistance is $R \times \text{Laplace}$ if **THEVENIN** equivalent is set for a behavioral resistance



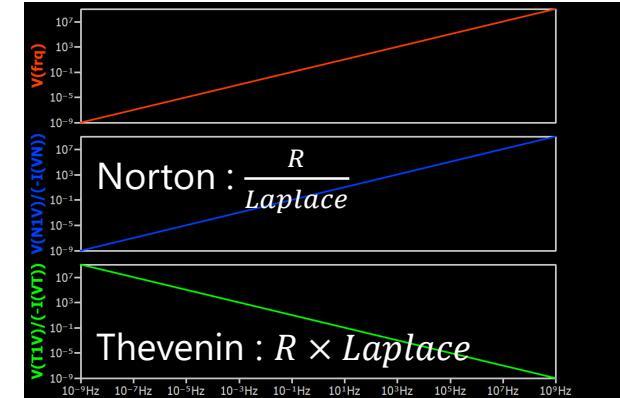
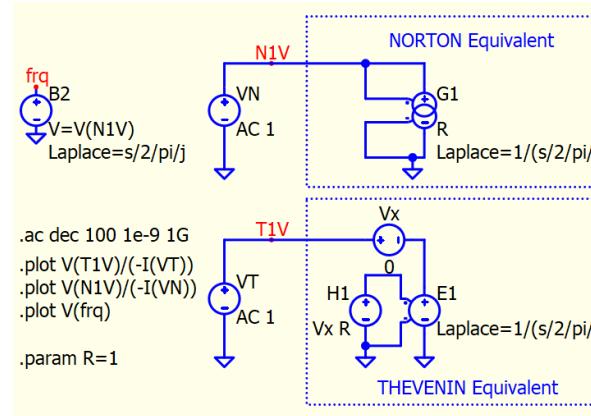
- Behavioral resistor B1

$$\frac{V(\text{AC1V})}{-I(V1)} = R \times \frac{1}{\frac{s}{2\pi j}} = \frac{R}{f}$$

B. Instance Params : Behavioral Resistor with Laplace

Qspice : B Source - Behavioral R with Laplace (Equivalent Circuit).qsch

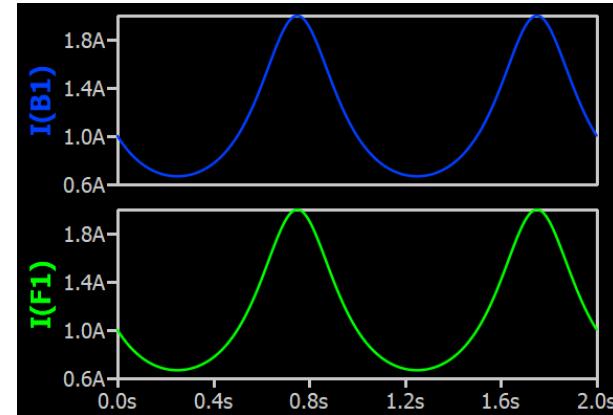
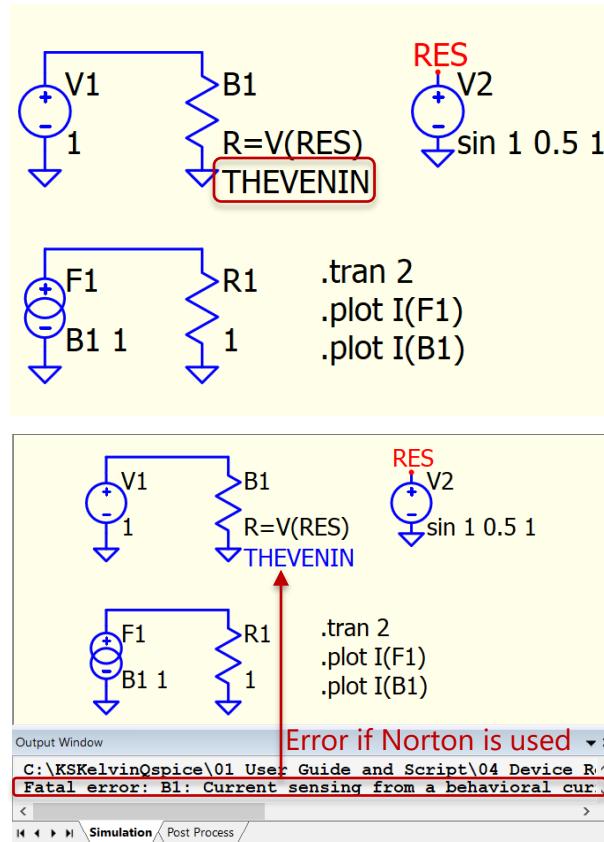
- Norton and Thevenin
 - circuit example shows how the B-source equivalent circuit is set up when Norton/Thevenin is selected
 - The **Norton** equivalent relies on a G-source (voltage-controlled current source) to sense voltage and generate current
 - The **Thevenin** equivalent relies on an H-source (current-controlled voltage source) to sense current and generate voltage.



B. Instance Params : Thevenin and Norton

Qspice : B Source - Thevenin Norton.qsch

- Thevenin and Norton
 - Thevenin and Norton determine the behavioral resistance to use Thevenin or Norton equivalent
 - Default is Norton**
 - Usage
 - For example, a F-source (current-dependent current source) must only sense the device as a Thevenin equivalent. To sense current from the behavioral resistance for the F-source, a Thevenin equivalent is required, or an error will occur



B. Instance Params : Laplace

Qspice : B Source - Laplace (.tran).qsch | B Source - Laplace (.ac).qsch

- Laplace

- Laplace equation in B-source, by adding a new attribute

Laplace=<expression>

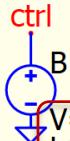
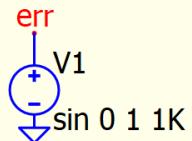
- Laplace can be specified for B-, E- and G-sources
- In this example

- $v_{ctrl} = \left(K_p + \frac{K_i}{s} + K_d s \right) v_{err}$

- Laplace supports both transient (.tran) and ac (.ac) analysis

- The Laplace function must be able to be synthesized Laplacian circuit in order to simulate it in .tran

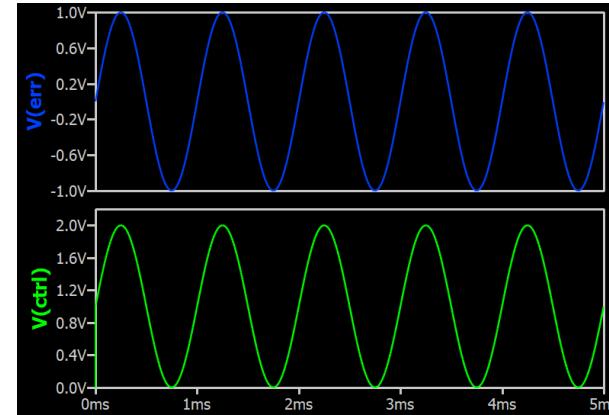
B-source with Laplace in .tran analysis



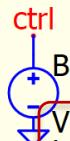
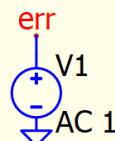
V=V(err)
Laplace=Kp+Ki/s+Kd*s

.tran 5/1K
.plot V(ctrl)
.plot V(err)

.param Kp=1
.param Ki=2*pi*1K
.param Kd=1/2/pi/1K



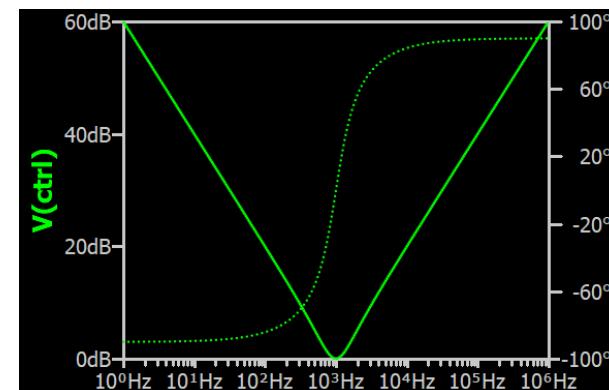
B-source with Laplace in .ac analysis



V=V(err)
Laplace=Kp+Ki/s+Kd*s

.ac dec 100 1 1Meg
.plot V(ctrl)
.plot V(err)

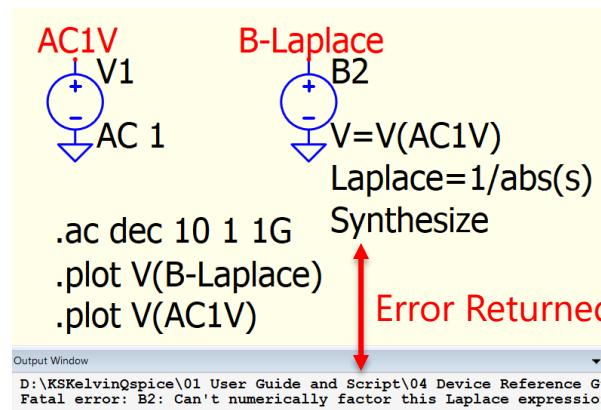
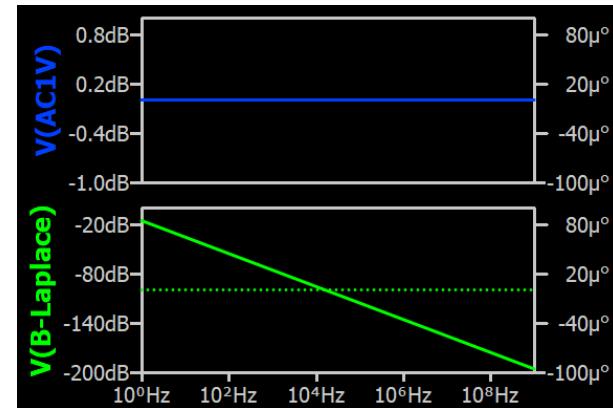
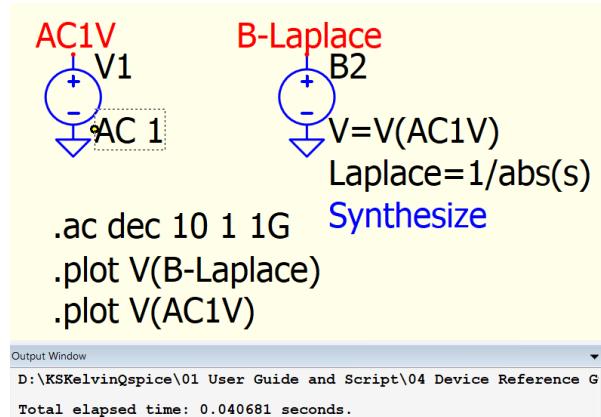
.param Kp=1
.param Ki=2*pi*1K
.param Kd=1/2/pi/1K



B. Instance Params : Laplace – Synthesize

Qspice : B Source - Synthesize.qsch

- Synthesize
 - Use the synthesized Laplacian circuit even for .AC
 - In this example, laplace function $\frac{1}{|s|}$ is used. This function cannot be synthesized into a laplacian circuit and cannot run .tran
 - If "synthesize" is set, .ac will force to use synthesized circuit to run simulation and in this case, an error returned



B. Instance Params : Laplace

Qspice : B Source - Laplace - Table Gain ONLY.qsch

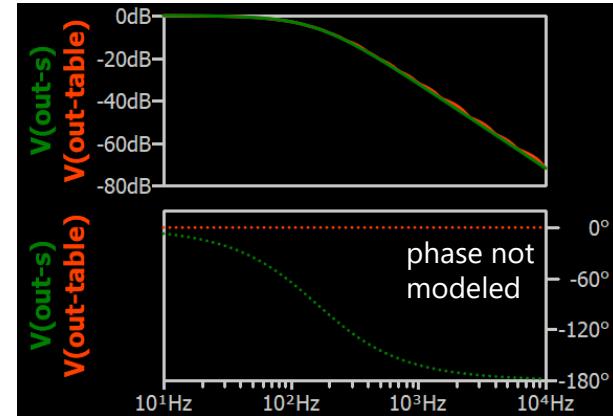
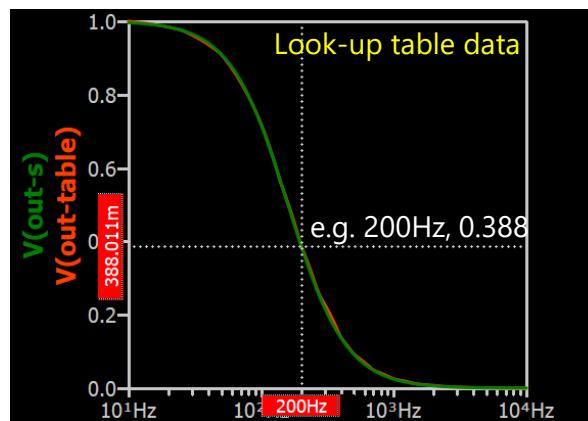
- Real-Valued Table
 - Laplace supports real-valued lookup tables in Laplace expressions
 - $\text{Table}(x, x_1, y_1, x_2, y_2, \dots)$ interpolate x from look-up table given as a set of pairs x_n, y_n
 - To model gain vs frequency into laplace, the look-up table have to setup to look for gain at different frequency
 - By setting $x = s/j/2/\pi$, this is equal frequency, and (x_n, y_n) is (freq,gain) pair data
 - Next slide explain how to model both gain and phase

06/18/2024 Added support for real-valued lookup tables in Laplace expressions

in V1 out-s B3 .param zeta=1
AC 1 V=V(in) .param wn=1K
Laplace=wn^2/(s^2+2*zeta*wn*s+wn^2)

out-table B2 In s-domain, $s=j*omega=j*2*pi*freq$
s/2/pi/j equals freq, therefore,
this table interpolate freq and real-value for Laplace
V=V(in)
Laplace=table(s/j/2/pi, 10, 25, 0.976, 50, 0.91,
+100, 0.72, 140, 0.56, 200, 0.388, 270, 0.258,
+400, 0.137, 500, 0.092, 700, 49m, 1K, 25m, 1.5K, 11.2m,
+2.5K, 0.004, 4K, 1.59m, 6K, 705μ, 10K, 253μ)

.ac oct 10 10 10K
.plot V(out-table) V(out-s)
.plot V(out-table) V(out-s)

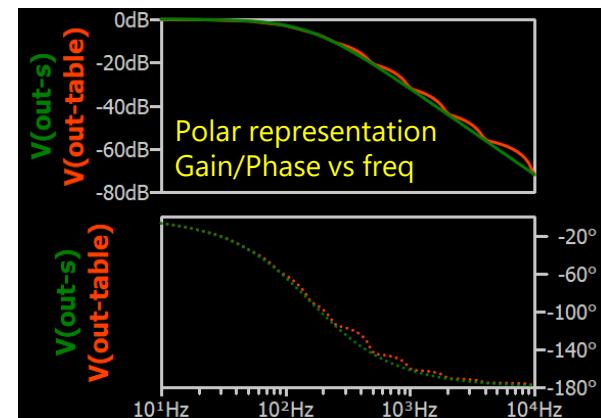
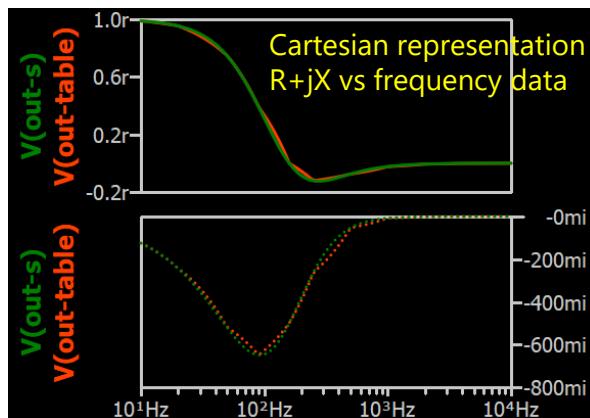
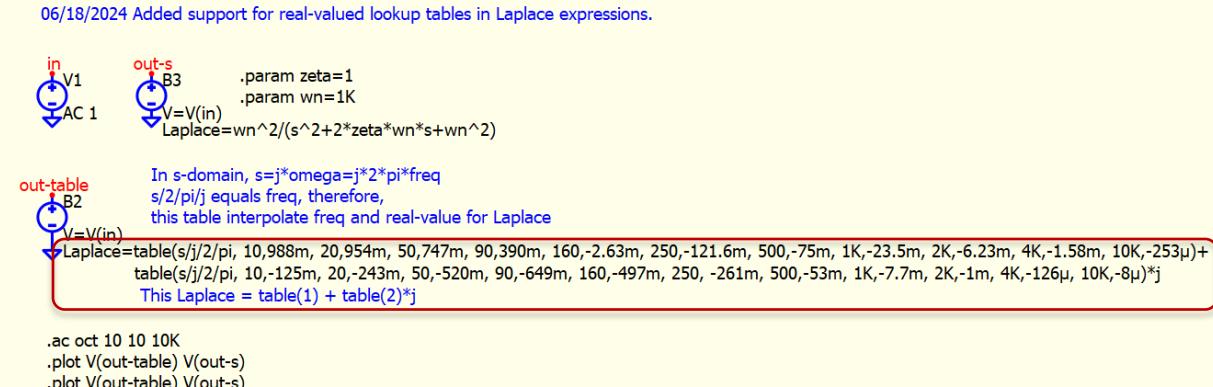


B. Instance Params : Laplace

Qspice : B Source - Laplace - Table Gain and Phase.qsch

- Real-Value Table

- Lookup table can model gain/phase vs frequency
- Laplace can accept R+jX format, ideal is to model R+jX at different frequencies
- $\text{Laplace} = \text{table(freq,freq1, re1,freq2,re2,...) + table(freq,freq1,i1,freq2,im2,...)*j}$
- If measurement result is gain/phase vs freq data, user have to convert it to real/imag vs freq data with this approach

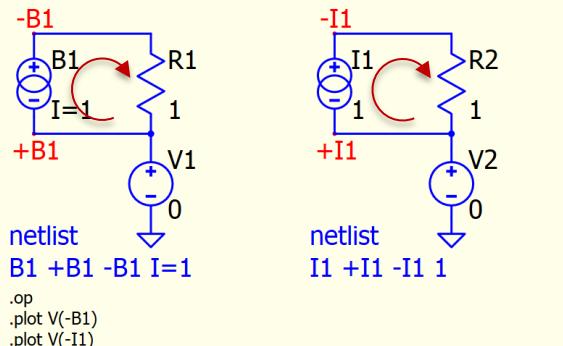


B. Current Direction and Netlist format

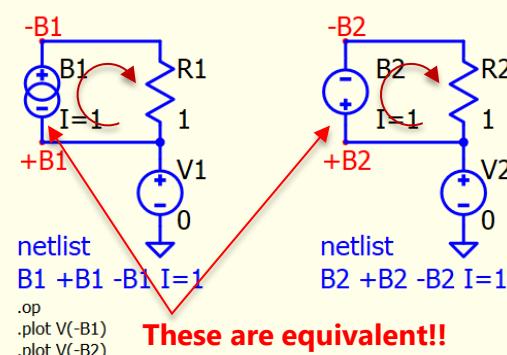
Qspice : B Source - I - direction.qsch | B Source - I - direction2.qsch

- Current Direction

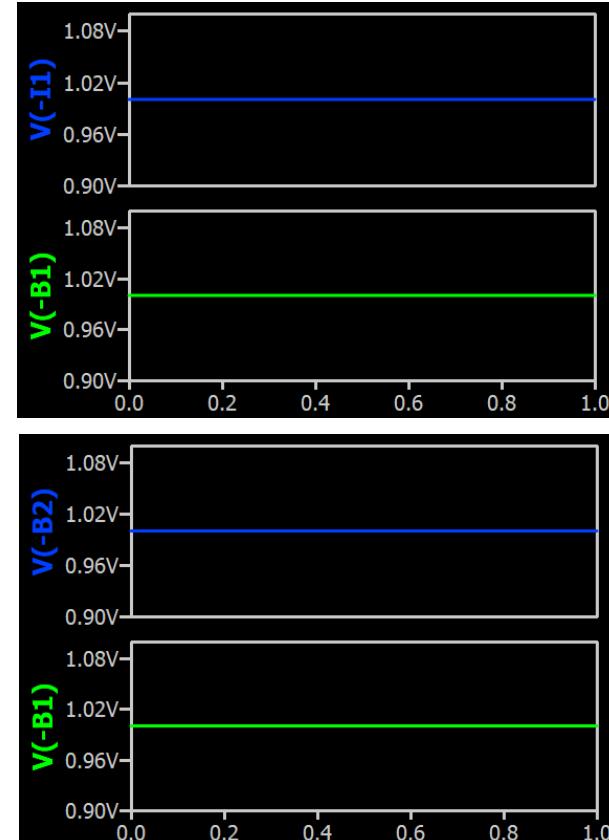
- In Spice, positive current always flows from first node (+) to second node (-) in a device
- However, for B-source and I-source, their symbol labels are visually connected in a way that the "+" label is linked to the second node (-ve) in the source netlist, and the "-" label is connected to the first node (+ve) in the source in netlist
- The reason this is important is that if you accidentally use the B-source voltage symbol to describe a current ($I=...$), as its symbol label is not reversed to its netlist, the current direction will follow the Spice definition. As a result, an equivalent symbol to the B-source current will actually appear reversed in terms of label polarity!



** in spice positive current always flow from + node to - node in netlist
+/- in I-source or B-source symbol is for visual purpose only



These are equivalent!!
** in spice positive current always flow from + node to - node in netlist
+/- in I-source or B-source symbol is for visual purpose only

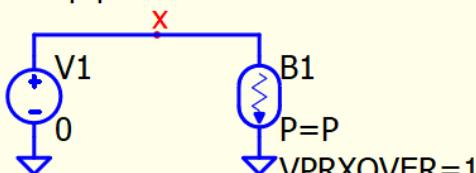


B. Instance Params : Behavioral Power Source

Qspice : B Source - P.qsch

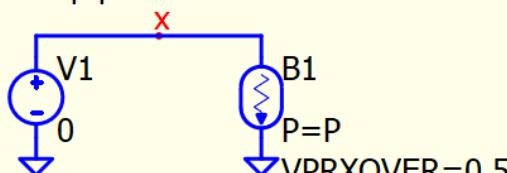
- P – Power Source
 - $P = <\text{expr}>$
 - Expression of Power
 - Sink power equals to $P = <\text{expr}>$
 - Only supported in Norton
 - VPRXOVER
 - Minimum voltage to regulate the power when P is specified
 - Default VPRXOVER=1
 - Voltage at VPRXOVER, Power source drive power equals $P = <\text{expr}>$
 - Or, voltage at $\frac{VPRXOVER}{\sqrt{2}}$, power also equals $\frac{P}{2}$ (W)

.step param P list 1 5 10

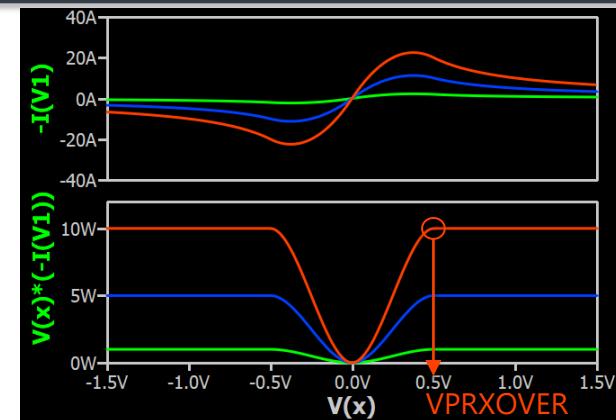
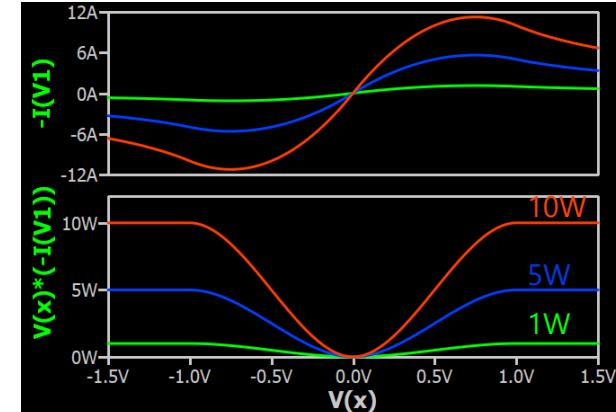


```
.dc V1 -1.5 1.5 0.01  
.plot V(x)*(-I(V1))  
.plot -I(V1)  
.plot V(x)
```

.step param P list 1 5 10



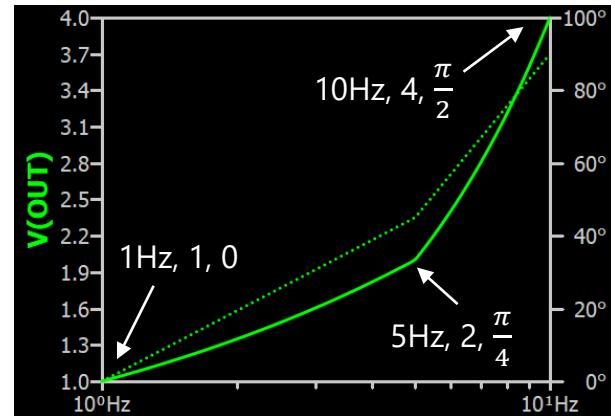
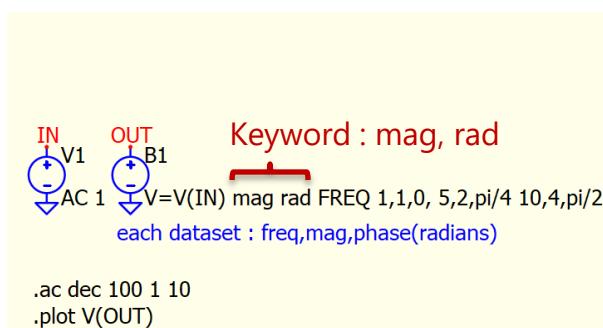
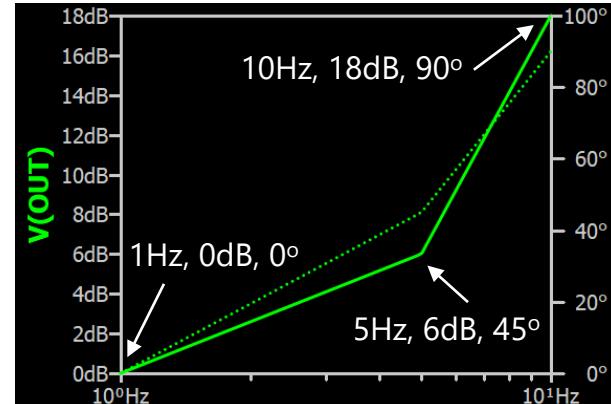
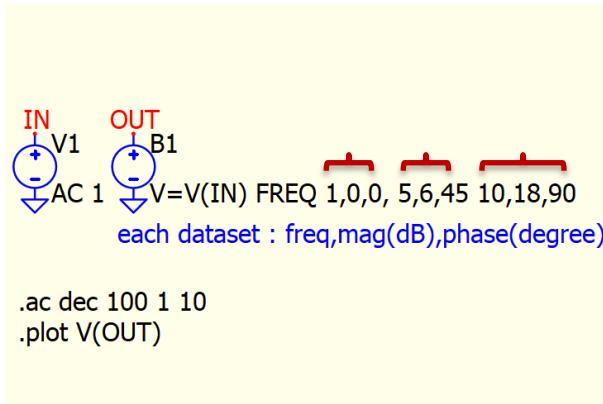
```
.dc V1 -1.5 1.5 0.01  
.plot V(x)*(-I(V1))  
.plot -I(V1)  
.plot V(x)
```



B. Instance Params : Freq (.ac only)

Qspice : B Source - Freq (.ac).qsch | B Source - Freq - mag rad.qsch

- Freq (.ac only)
 - Frequency domain lookup table
 - Freq circuit element specified by an ordered list of points of freq (Hz), mag (dB) and phase (degree) as $\langle (f_1, m_1, p_1) [(f_2, m_2, p_2)] \dots \rangle$
 - Not support time domain simulation
 - Keyword
 - Mag : magnitude (non-dB)
 - Rad : phase in radian
 - R_I : Real and Imaginary parts (f_1, r_1, x_1)



B. Behavioral Source

Functions Fcn

B. Function and Operators

HELP > Simulator > Device Reference > B. Behavioral Sources

Functions

Name	Description
abortsim(x)	stop simulation if $x > .5$, otherwise return x
abs(x)	absolute value of x
acos(x)	arc cosine of x
arccos(x)	synonym for acos()
acosh(x)	arc hyperbolic cosine of x
asin(x)	arc sine of x
arcsin(x)	synonym for asin()
asinh(x)	arc hyperbolic sine
atan(x)	arc tangent of x
arctan(x)	synonym for atan()
atan2(y,x)	four quadrant arc tangent of y/x
atanh(x)	arc hyperbolic tangent
buf(x)	1 if $x > .5$, else 0
ceil(x)	integer equal or greater than x
cos(x)	cosine of x
cosh(x)	hyperbolic cosine of x
dtt(x)	time derivative x
delay(x,y)	x delayed by y
delay(x,y,z)	x delayed by y, but store no more than z history
dlim(x,y,z)	x bounded by y which it asymptotically starts to approach at y+z as a first inverse order Laurent series
exp(x)	e to the x
floor(x)	integer equal to or less than x
hypot(x,y)	$\sqrt{x^2 + y^2}$
idt(x,y,z)	time integral of x with initial condition of y reset when z > .5
	$\int x \, dt$

if(x,y,z)	if $x > .5$, then y else z
int(x)	convert x to integer
inv(x)	0. if $x > .5$, else 1.
limit(x,y,z)	intermediate value of x, y, and z
In(x)	natural logarithm of x
log(x)	alternate syntax for ln()
log10(x)	base 10 logarithm
max(x,y)	the greater of x or y
min(x,y)	the smaller of x or y
pow(x,y)	x^y
pwr(x,y)	$ x ^y$
pwrs(x,y)	abs(x) ^y
	sgn(x)*abs(x) ^y
random(x)	random number from 0. to 1. depending on the integer value of x. Interpolation between random numbers is linear for non-integer x
resetwave(x)	clear waveform data if $x > .5$
sin(x)	sine of x
sinh(x)	hyperbolic sine of x
sqr(x)	\sqrt{x}
state(n,x)	value of x n time steps ago. n rounded to nearest integer and limited to the range 0 to 3
table(x,a,b,c,d,...)	interpolate x from the look-up table given as a set of pairs of constant values.
tan(x)	tangent of x
tanh(x)	hyperbolic tangent of x
ulim(x,y,z)	x bounded by y which it asymptotically starts to approach at y-z as a first inverse order Laurent series

Available Function in B source not listed

- Trunc(x) ; floor(x) ; int(x) : rounded down integer
- Rint(x) ; round(x) : rounded to nearest integer
- Ceil(x) : rounded up integer
- Ustep(x) : $x > 0 ? 1 : 0$
- Uramp(x) : $x > 0 ? x : 0$
- Minmag(x,y), Maxmag(x,y)

Operators grouped in reverse order of precedence of evaluation

Operand	Description
&	Boolean AND & or &&
	Boolean OR or
^^	is Boolean XOR
>	True if expression on the left is greater than the expression on the right.
<	True if expression on the left is less than the expression on the right.
>=	True if expression on the left is greater than or equal the expression on the right.
<=	True if expression on the left is less than or equal the expression on the right.
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	** / ^ Raise left hand side to power of right hand side. Same as '^'.
!	Boolean not the following expression.

Available Variable or Constant not listed in HELP

- Time : Simulation Time
- Temp : Temperature
- PI : 3.14159265358979323846
- E : 2.7182818284590452354
- K : 1.380649e-23 J/K
- Q : 1.602176487e-19 Coulomb

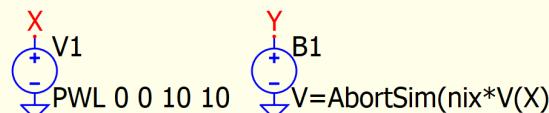
B. Function and Operators – abortSim(x), variable/constant

Qspice : B Source - AbortSim.qsch | B Source - Buildin Variable Constant.qsch

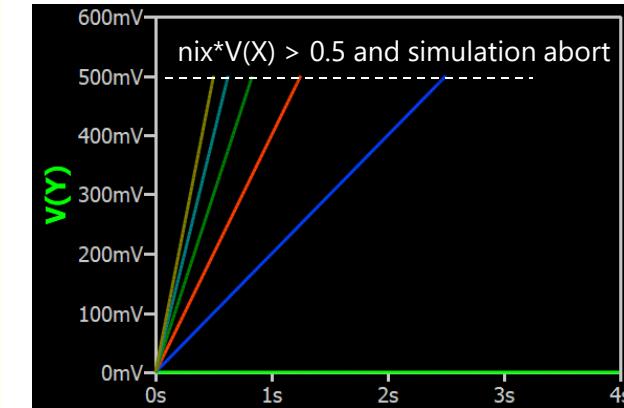
AboutSim(x)

- To stop the current simulation from a behavioral source
- If $x > 0.5$: *abort simulation*
- else : *return x*
- The function works for .dc and .tran
- Refer to Ø-Device, MaxExtStepSize() for same feature but in .dll

syntax : AbortSim(double arg)
if arg > 0.5 : stops simulation and goes to next step (if any)
else : returns the argument

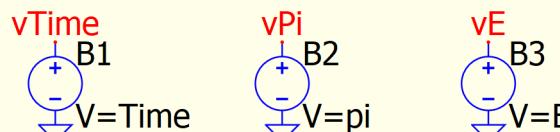


.tran 4
.step param nix 0 1 0.2



Build-in Variable / Constant

- Time : Simulation Time
- Temp : Temperature
- PI : 3.14159265358979323846
- E : 2.7182818284590452354
- K : 1.380649e-23 J/K
- Q : 1.602176487e-19 Coulomb

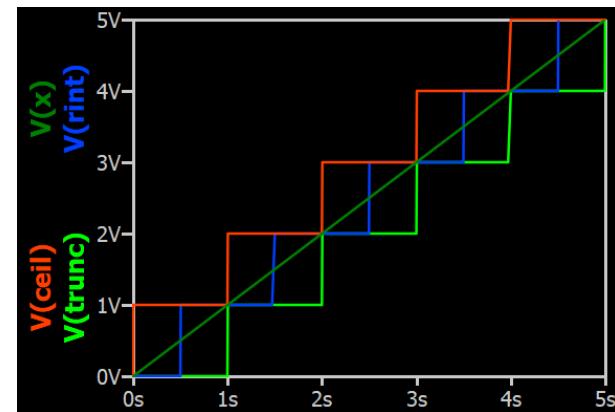
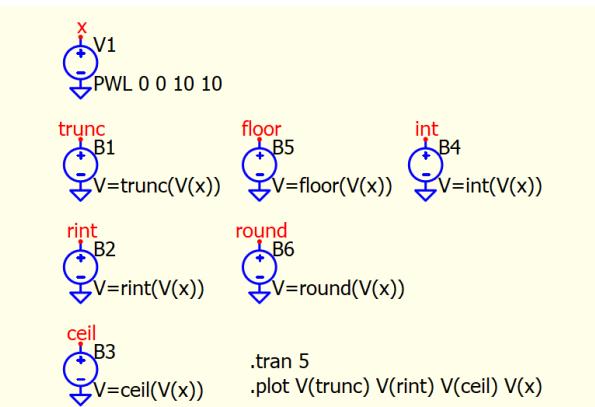


.tran 1

B. Function and Operators – trunc(), floor(), int(), rint(), round(), ceil()

Qspice : B-source - Trunc Rint Ceil.qsch

- Return Integer
 - $\text{trunc}(x)$, $\text{floor}(x)$, $\text{int}(x)$: rounded down integer
 - $\text{rint}(x)$, $\text{round}(x)$: rounded to nearest integer
 - $\text{ceil}(x)$: rounded up integer



B. Function and Operators – ddt(x) and limit(x,y,z)

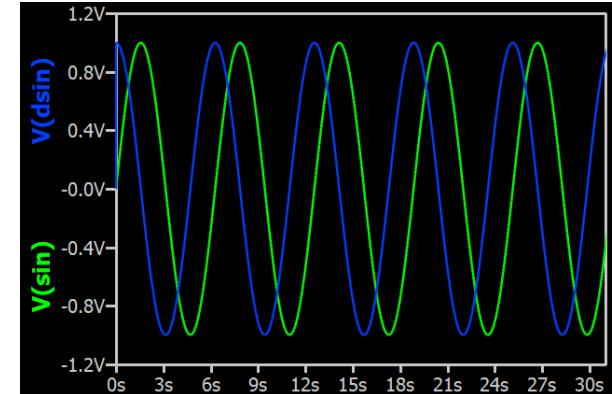
Qspice : B Source - ddt.qsch | B Source - limit.qsch

- $\text{ddt}(x)$
 - Time derivative
 - $= \frac{dx}{dt}$

Syntax : $\text{ddt}(x)$
Time derivative x

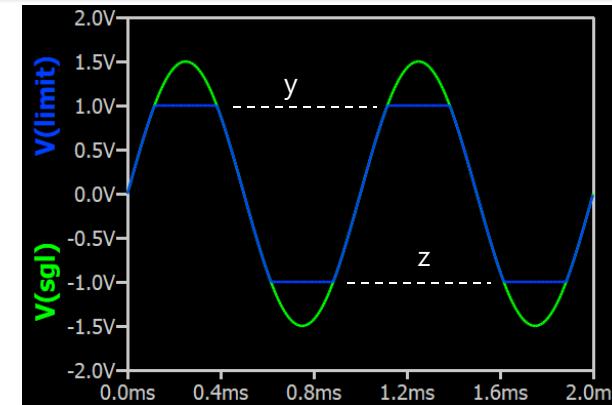
Formula
 $d(\sin(\omega t))/dt = \omega \cos(\omega t)$

```
.param f=1/2/pi
.param omega=2*pi*f
.tran 5/f
.plot V(sin) V(dsin)
```



- $\text{limit}(x,y,z)$
 - Intermediate value of x, y, and z

```
.tran 2m
.plot V(sgl) V(limit)
```



B. Function and Operators – dlim(x,y,z) and ulim(x,y,z)

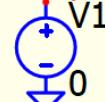
Qspice : B Source - dlim ulim.qsch

- dlim(x,y,z) and ulim(x,y,z)
 - dlim(x,y,z) : x bounded by y which it asymptotically starts to approach at $y+z$ as a first inverse order Laurent series
 - ulim(x,y,z) : x bounded by y which it asymptotically starts to approach at $y-z$ as a first inverse order Laurent series
 - These functions are used for soft limit

Syntax : dlim(x,y,z) or ulim(x,y,z)

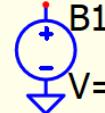
x bounded by y which it asymptotically starts to approach at $y+z$ as a first inverse order Laurent series

ramp



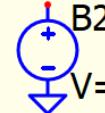
```
.dc V1 -4 4 0.1  
.step param z list 0 1 2  
.plot V(dlim)  
.plot V(ulim)
```

dlim

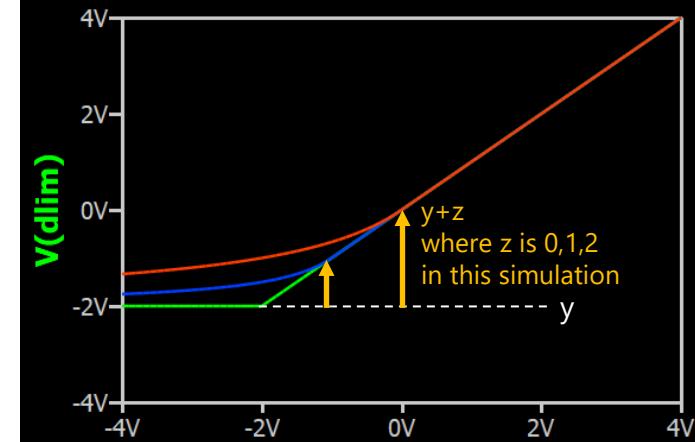
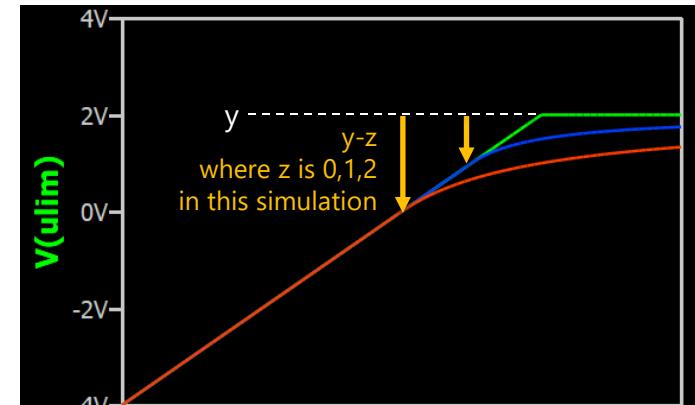


V=dlim(V(ramp),-2,z)

ulim



V=ulim(V(ramp),2,z)

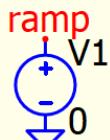


B. Compare Qspice dlim() / ulim() and LTspice dnlim(), uplim()

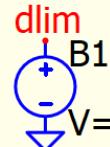
Qspice : B Source - dlim - compare LTspice.qsch | B Source - dlim - compare LTspice.cir

- Comparison

- Qspice : dlim() / ulim()
 - limit is only asymptotically approached
- LTspice : dnlim() / uplim()
 - two straight lines connected by a quadratic
 - Issue : Its slope vanishes after hits the limit and can causes convergence problems

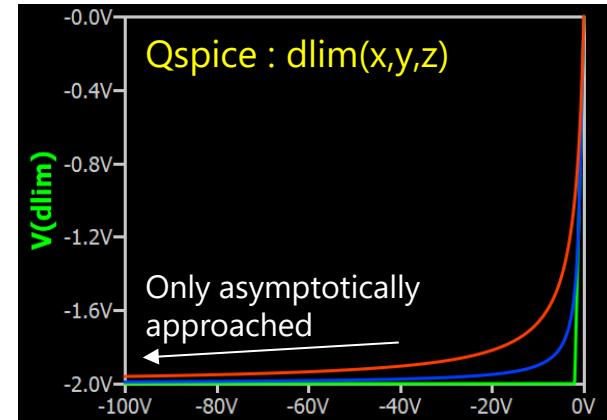


```
.dc V1 -100 0 0.01  
.step param z list 0 1 2  
.plot V(dlim)
```



```
V=dlim(V(ramp),-2,z)
```

```
* D:\KSKelvinQspice\01 User Guide and Script  
B1 dnlim 0 V=dlim(V(ramp),-2,z)  
V1 ramp 0 0  
.plot V(dlim)  
.step param z list 0 1 2  
.dc V1 -100 0 0.01  
.end
```



B. Function and Operators – delay(x,y) or delay(x,y,z)

Qspice : B Source - delay fix.qsch

- `delay(x,y)` or `delay(x,y,z)`
 - `Delay(x,y,z)` can reduce the amount of waveform memory that must be stored

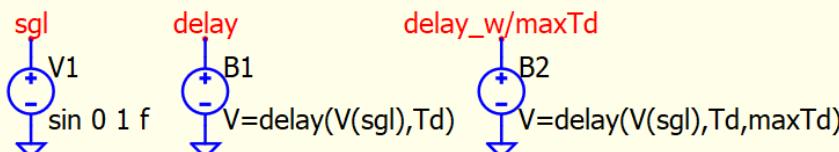
$\text{delay}(x,y)$	x delayed by y
$\text{delay}(x,y,z)^1$	x delayed by y , but store no more than z history

- y can be parameter and z must be fixed constant
 - y must less than z or error will return

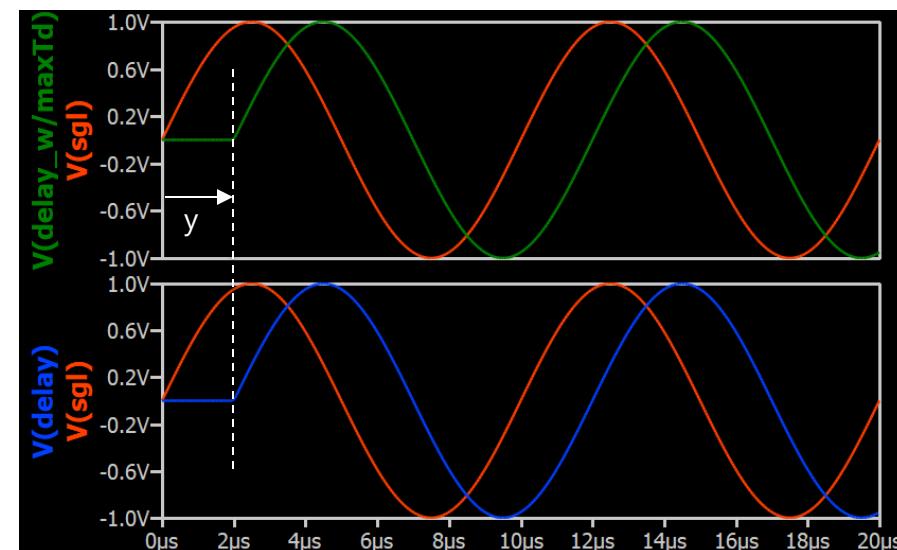
Syntax : delay(x,y) - x delayed by y

Syntax : delay(x,y,z) - x delayed by y, but store no more than z history

```
.param f = 100K  
.param Td = 2μ  
.param maxTd = 5μ
```



```
.tran 2/f  
.plot V(sgl) V(delay)  
.plot V(sgl) V(delay)
```

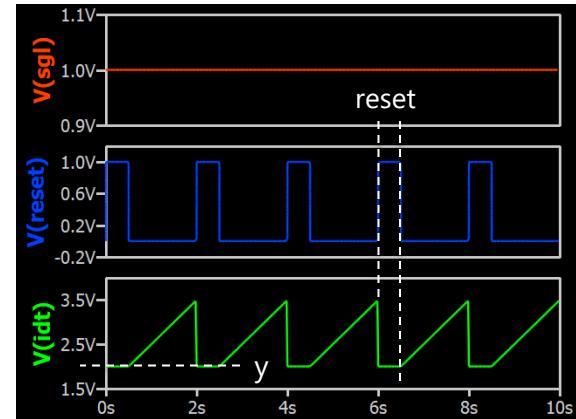
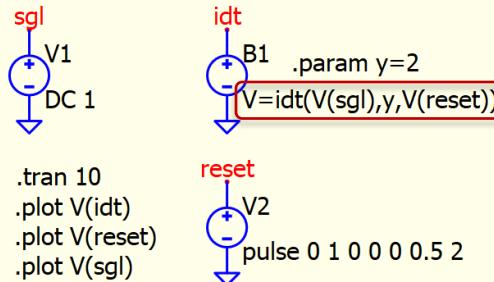


B. Function and Operators – idt(x,y,z)

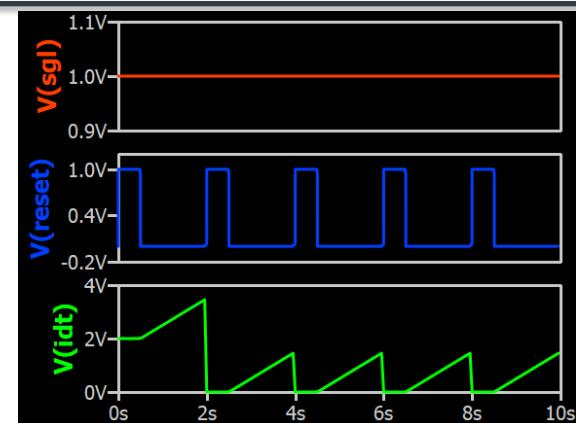
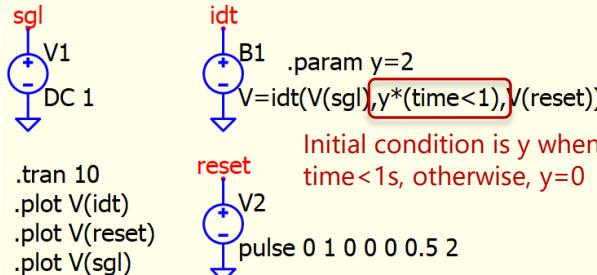
Qspice : B Source - idt.qsch | B Source - idt (y-conditional).qsch

- $\text{idt}(x,y,z)$
 - Time Integral of x
 - $= \int x dt + y$
 - y is initial condition, which is the DC value during integral function is reset
 - z is to reset integral
 - Special technique for initial condition
 - If initial condition needs to be changed over time, conditional function can be used
 - User can also use voltage source to control initial voltage y over time

Syntax : $\text{idt}(x,y,z)$
Time integral of x with initial condition of y
reset when $z > .5$



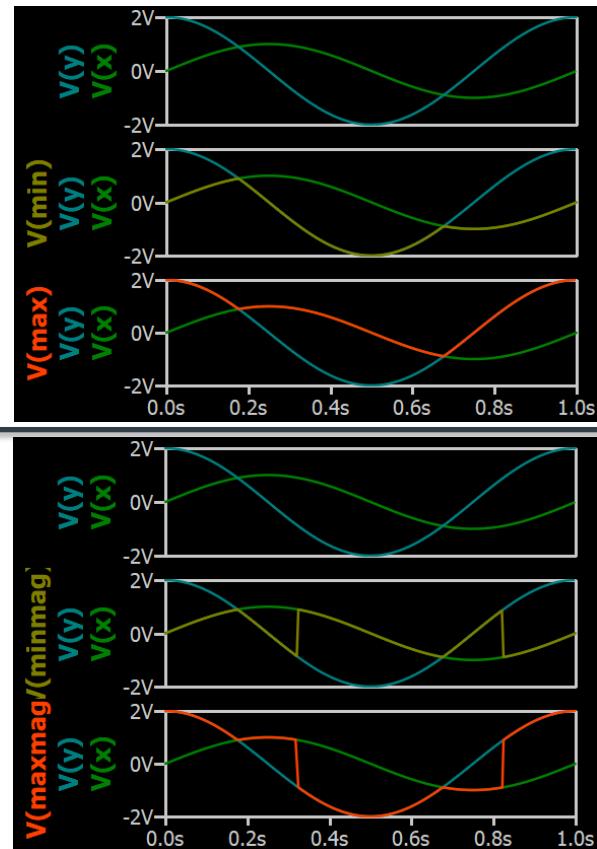
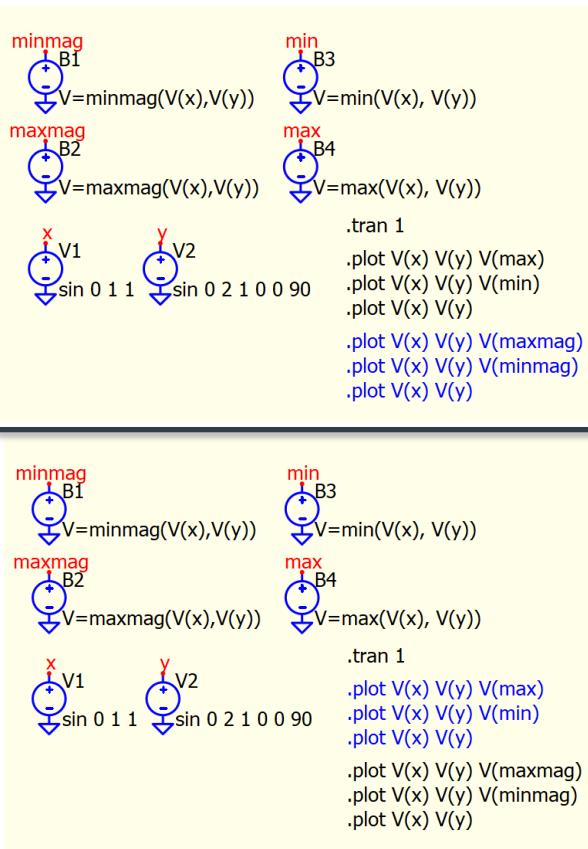
Syntax : $\text{idt}(x,y,z)$
Time integral of x with initial condition of y
reset when $z > .5$



B. Function and Operators – min(), max(), minmag() and maxmag()

Qspice : B-source - min max minmag maxmag.qsch

- $\min(x,y)$ and $\max(x,y)$
 - $\min()$: Minimum of x and y
 - $\max()$: Maximum of x and y
 - These function returns min or max value
- $\minmag(x,y)$
 $\maxmag(x,y)$
 - $\minmag(x,y)$: minimum magnitude of x and y
 - $\maxmag(x,y)$: maximum magnitude of x and y
 - These function returns min or max value based on magnitude and reserve same sign



B. Function and Operators – random(x) [Basic Concept]

Qspice : B Source - random (seed).qsch

• Random(x)

- Random(x) : Random number from 0 to 1 depending on the integer value of x
- This is pseudo random function which takes a seed and output a number "appears random"

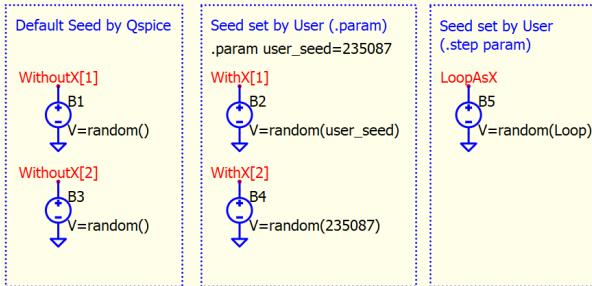
• Random()

- Seed is determined by Qspice
- If .option seed=<value> is used, user can assign an initial seed
- If .option seedclock is used, seed is from 10MHz clock and process ID, which appear random in every run

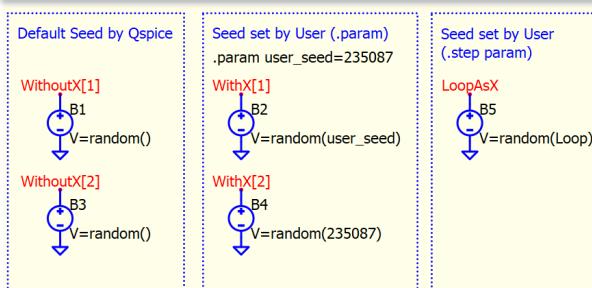
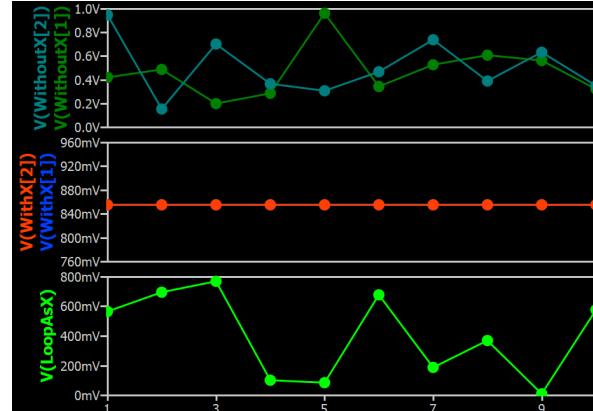
• Random(x)

- Seed is set by user
- Assign a fixed value (e.g. .param), or varying value (from .step param)

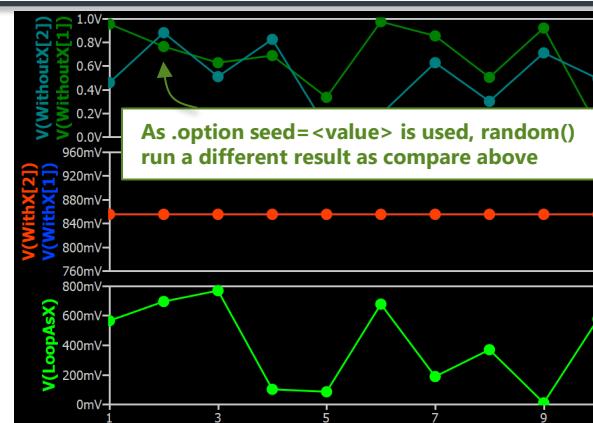
- These two examples run same results every time in Qspice as initial seed not be changed in each run



No .option



.option seed

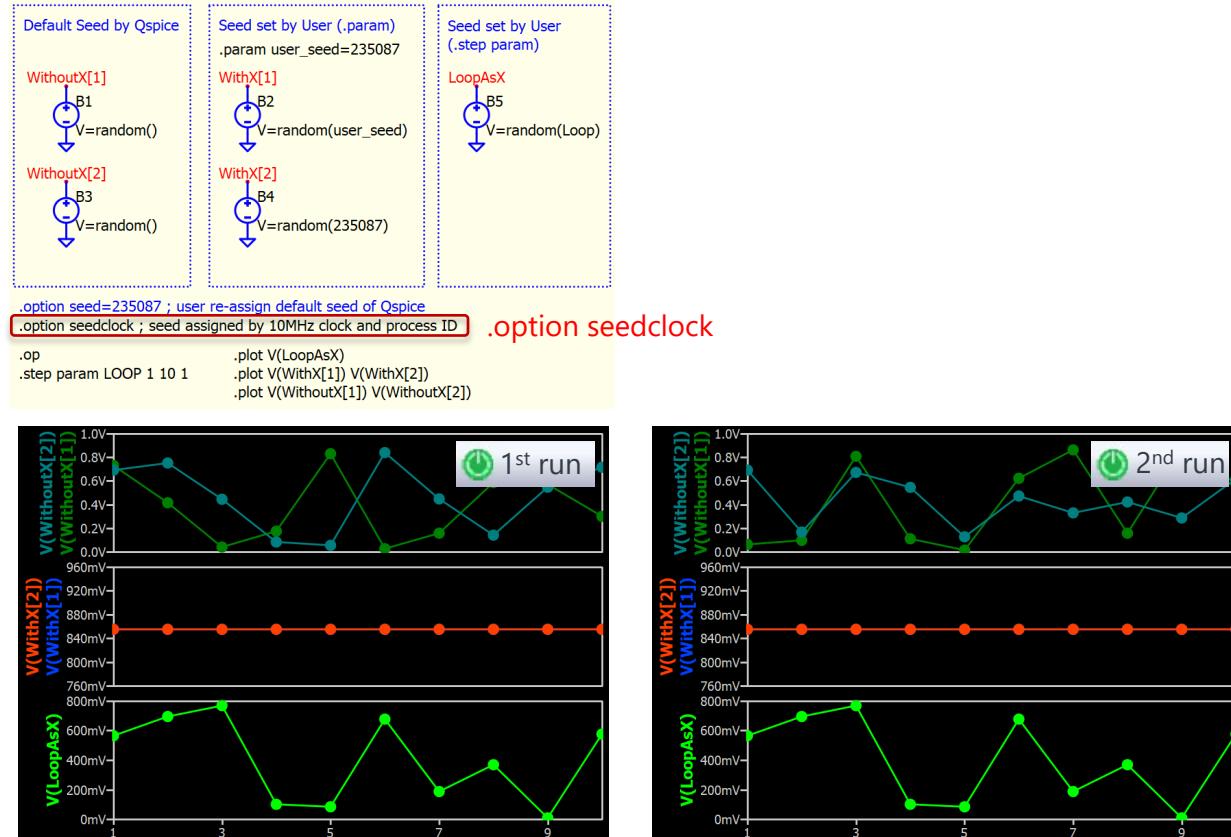


As .option seed=<value> is used, random() run a different result as compare above

B. Function and Operators – random(x) [.option seedclock]

Qspice : B Source - random (seed).qsch

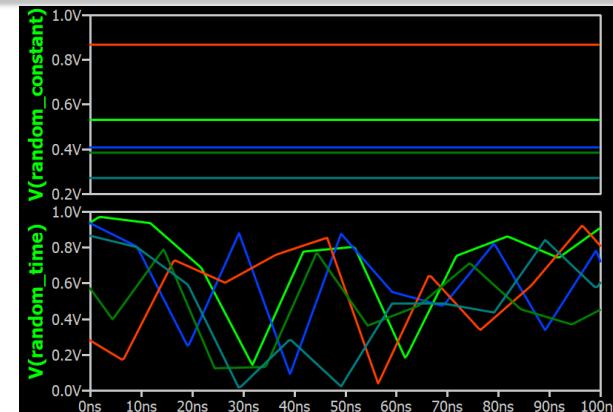
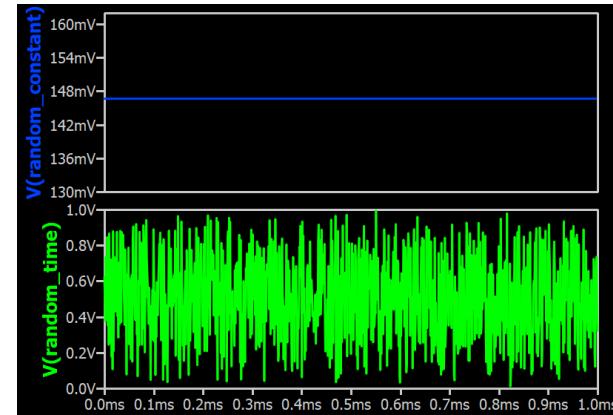
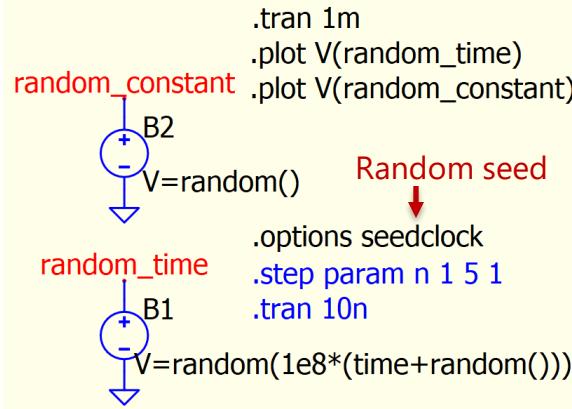
- Random(x)
 - .option seedclock
 - Initialize the random number generator with a 10Mhz clock and the process ID number
 - If seedclock is used, initial seed are different in every run
 - random() give different random pattern in every run
 - But random(x) with user defined seed and not be impacted



B. Function and Operators – random(x) [Random in .tran]

Qspice : B Source – random (.tran).qsch

- Random(x)
 - Random number from 0 to 1 depending on the integer value of x
 - The one without arguments must be processed in the preprocessor (from Mike Engelhardt)
 - To create a random sequence in transient, x must change in each time step : i.e. $\text{random}(1\text{e}8 * \text{time})$, multiple time with $1\text{e}8$ is for x to be a large integer for random generation
 - For random run in each step, function can be
 - $\text{random}(1\text{e}8 * (\text{time} + \text{random}()))$



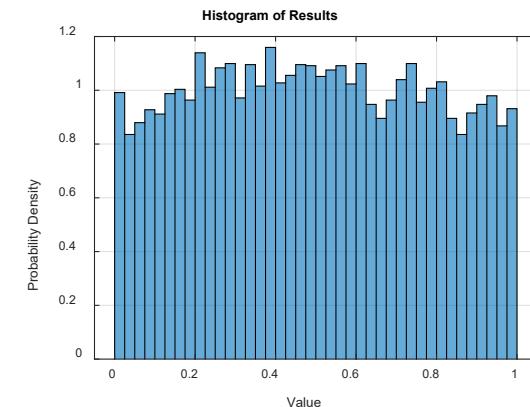
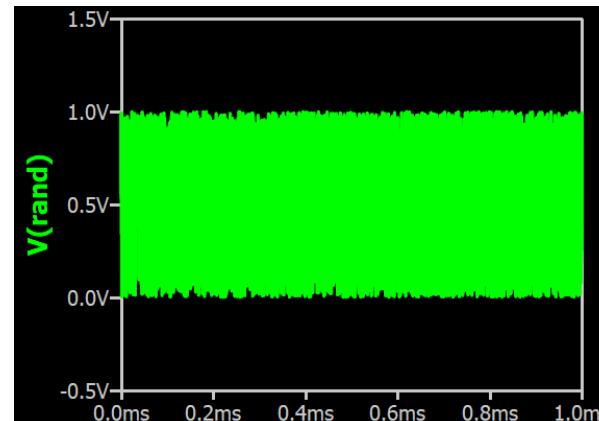
With **random(1e8*(time+random()))**, it allows random voltage also at 0s

B. Function and Operators – random(x) [Random in .tran]

Qspice : B Source - random uniform (.tran).qsch | histogram_matlab.m

- Random in .tran to ensure uniform distribution
 - As time change through the simulation, it is used as seed for random() function at each time step
 - However, the seed that can generate a uniform distribution requires randomly between 0 and 1e15
 - To ensure each time step can generate a seed between 0 and 1e15, it requires dynamic scaling for time with $1e15 \times \frac{time}{10^{\text{floor}(\log_{10} time)}}$, where divider $10^{\text{floor}(\log_{10} time)}$ is to scale time into 0 and 1
 - In the formula, **Rscale*random()** is an offset seed, if .option seedclock is used, this allow each simulation run to generate a random set of number

```
rand  
B1 .param Rscale=1e15  
V=random(Rscale*time/V(divider)+Rscale*random())  
  
divider  
B7 V=pow(10,floor(log10(time)))  
  
.param Tstop=1m  
.tran 0 Tstop 0 Tstop/1e4  
.plot V(rand)  
.options seedclock  
.system %QUX% -Export "%RAWFILE%" V(rand) all CSV
```



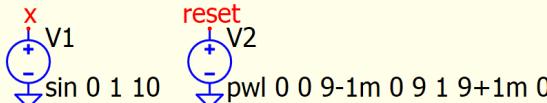
B. Function and Operators – resetwave(x)

Qspice : B-source - resetwave.qsch

- ResetWave(x)
 - clears the waveform data (.qraw) if $x > 0.5$
 - Be aware that whenever $x > 0.5$, the clear action operates continuously, which can increase the simulation time.

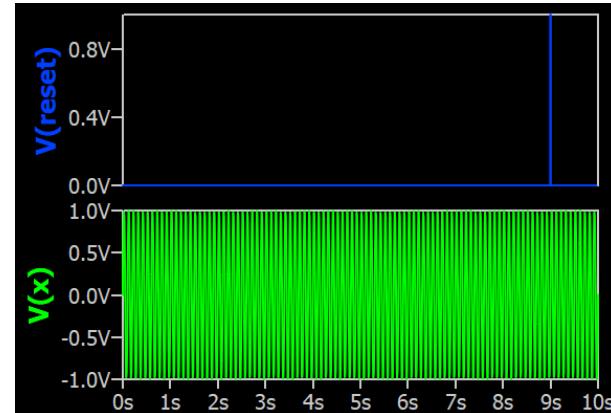
Therefore, in this example, a single pulse is used for the reset wave

resetwave(x): clears the waveform data if $x > 0.5$

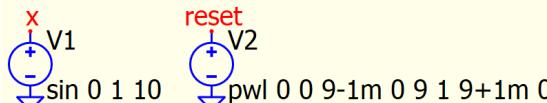


.tran 10
.plot V(x)
.plot V(reset)
 $V=\text{resetwave}(V(\text{reset}))$

Disable resetwave(x)

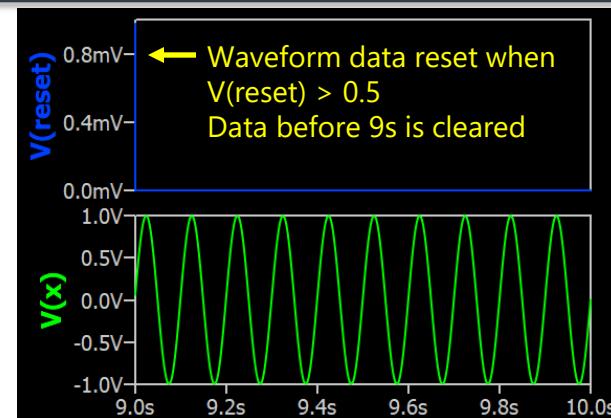


resetwave(x): clears the waveform data if $x > 0.5$



.tran 10
.plot V(x)
.plot V(reset)
 $V=\text{resetwave}(V(\text{reset}))$

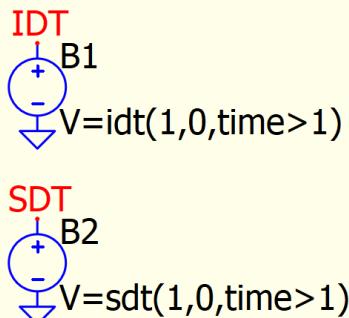
Enable resetwave(x)



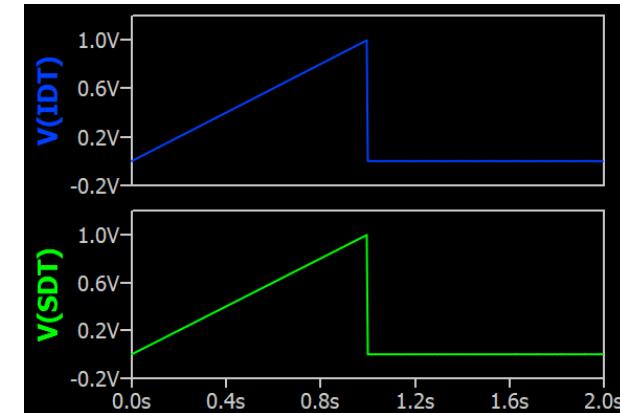
B. Function and Operators – sdt(x,y,z)

Qspice : B Source - sdt.qsch

- $sdt(x,y,z)$
 - $SDT()$ is synonym for $IDT()$
 - Time integral of x



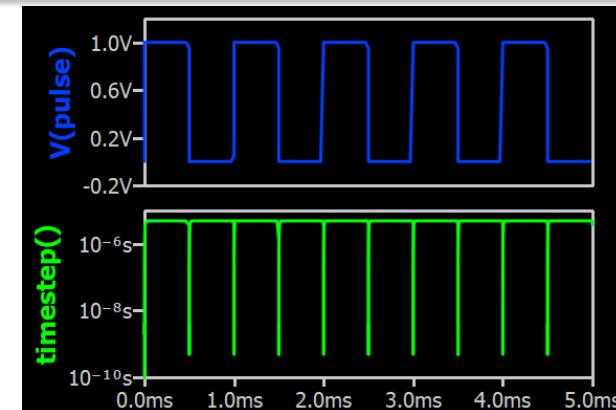
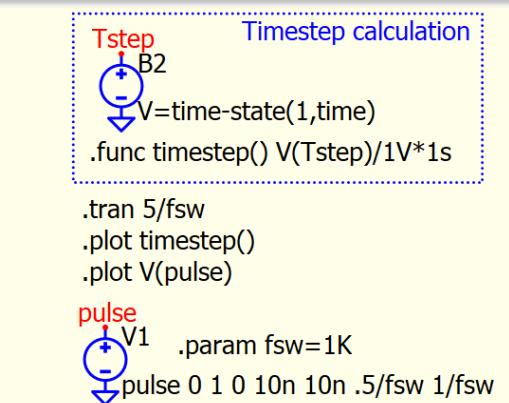
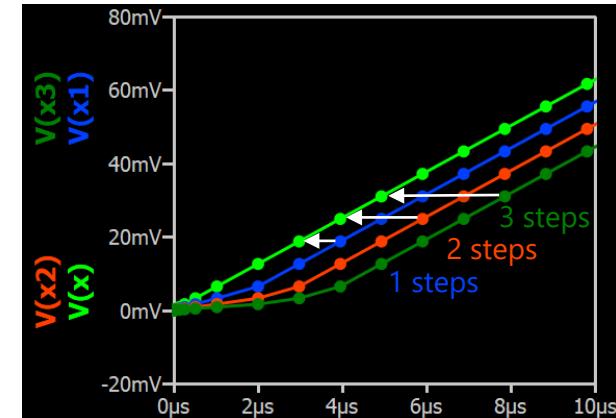
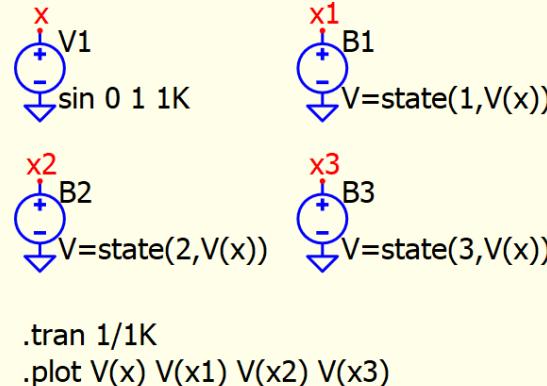
.tran 2
.plot V(SDT)
.plot V(IDT)



B. Function and Operators – state(n,x)

Qspice : B source - state.qsch | B source - state for timestep.qsch

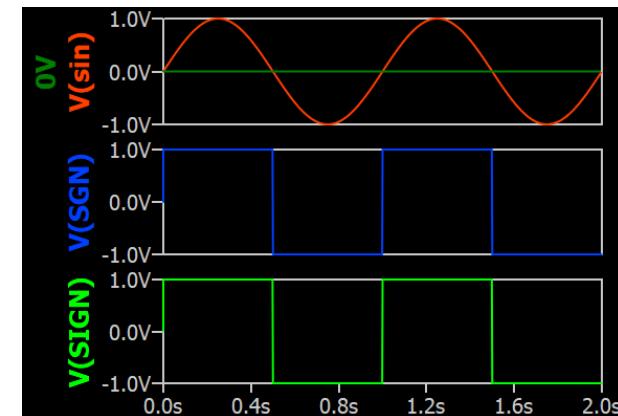
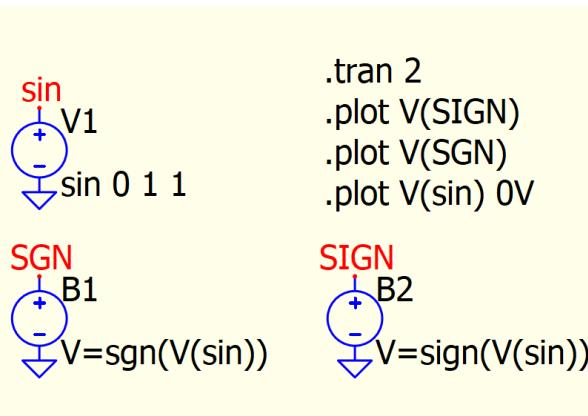
- State(n,x)
 - Value of x, n times steps ago. N rounded to nearest integer and limited to the range 0 to 3
 - Step is refer to each simulation timestep
 - A case of application
 - This function can help to get the profile of simulation timestep



B. Function and Operators – $\text{sgn}(x)$, $\text{sign}(x)$

Qspice : B Source - sgn sign.qsch

- Sgn(x), Sign(x)
 - Sign of x
 - If $x > 0$, output 1
 - If $x = 0$, output 0
 - If $x < 0$, output -1

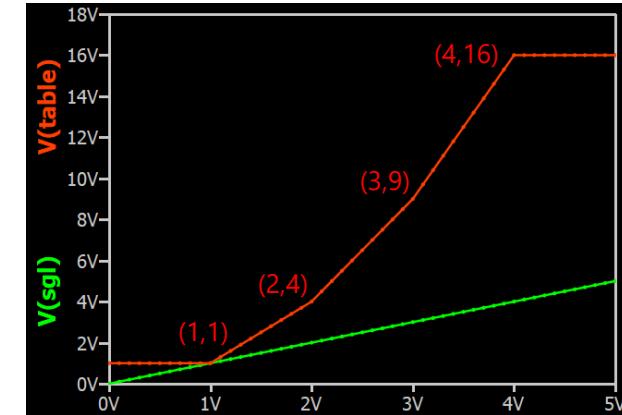
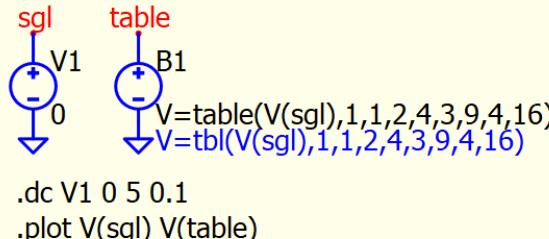


B. Function and Operators – table(x,a,b,c,d,...)

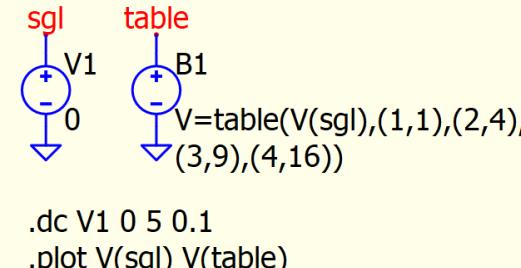
Qspice : B Source - table.qsch | B Source - table(.func).qsch | B Source - table(attribute).qsch

- Table(x,a,b,c,d,...)
 - Interpolate x from the look-up table given as a set of pairs of constant values
 - **tbl(x,a,b,c,d,...)** is an alternative name for Table, and it is also supported in Qspice
- Setup a table
 - As table may contain a lot of data points, for visual purpose in schematic, user can consider
 - Each pair of (x,y) data uses a bracket
 - Break table with 2nd or more attribute
 - Use .func to define a table

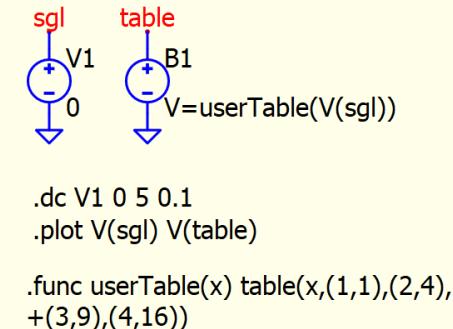
Basic table setup in B-source



Define a table with 2nd or more attribute



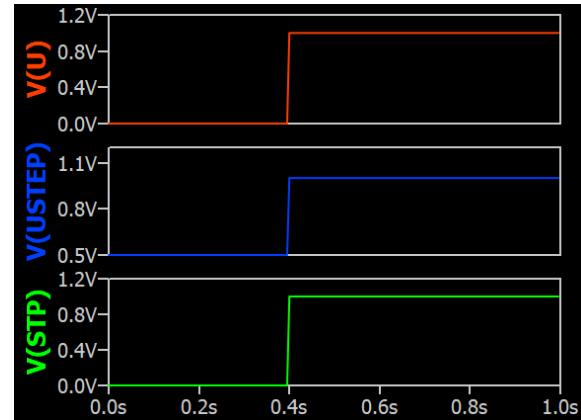
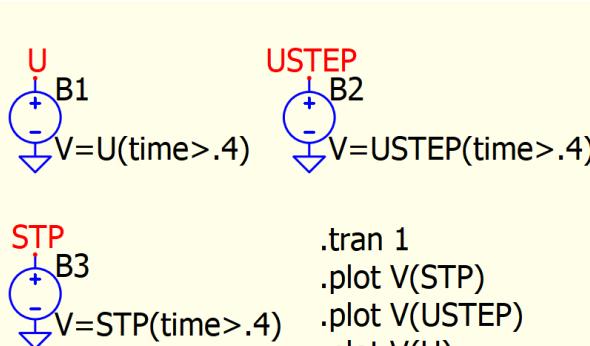
Define a table with .func and with + line break



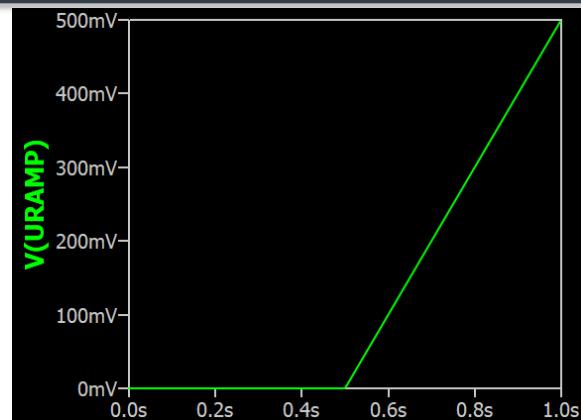
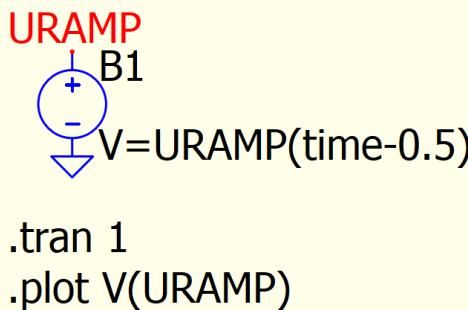
B. Function and Operators – $u(x)$, $ustep(x)$, $stp(x)$ and $uramp(x)$

Qspice : B Source - u $ustep$ $stp.qsch$ | B Source - $uramp.qsch$

- $U(x)$, $USTEP(x)$, $STP(x)$
 - $x > 0 ? 1 : 0$
 - If $x > 0$, output 1
 - If $x \leq 0$, output 0



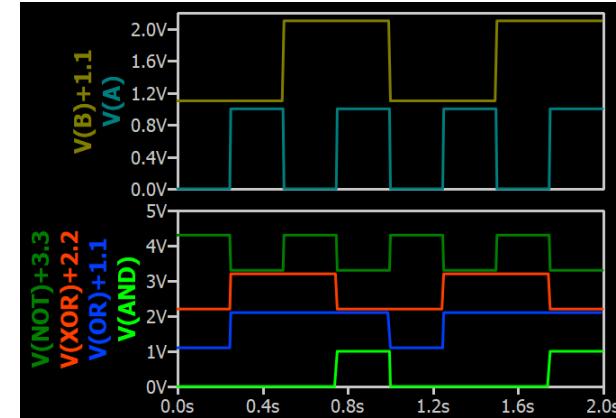
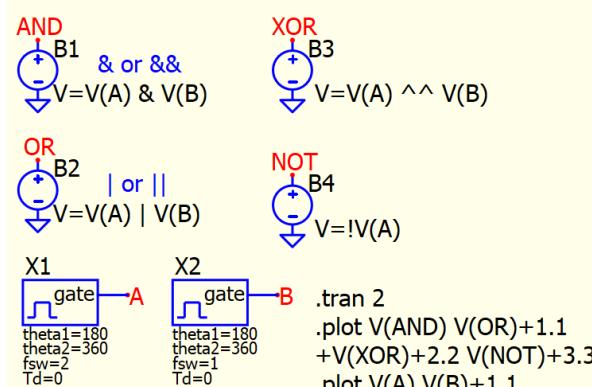
- Uramp(x)
 - $x > 0 ? x : 0$
 - If $x > 0$, output x
 - If $x \leq 0$, output 0



B. Function and Operators – Boolean : AND, OR, XOR, NOT

Qspice : B Source - Boolean.qsch

- Boolean
 - Standard Boolean operators are
AND : & or &&
OR : | or ||
XOR : ^ ^
NOT : !



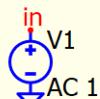
B. Behavioral Source

Laplace – Filter
Functions

B. Laplace – Filter Functions (Gaussian Filter)

Qspice : B Source - Laplace - Gaussian Filter.qsch

```
.param FREQ=1K  
.param BW=200  
.ac lin 1000 1 5K  
.step param N list 1 2 4 8
```



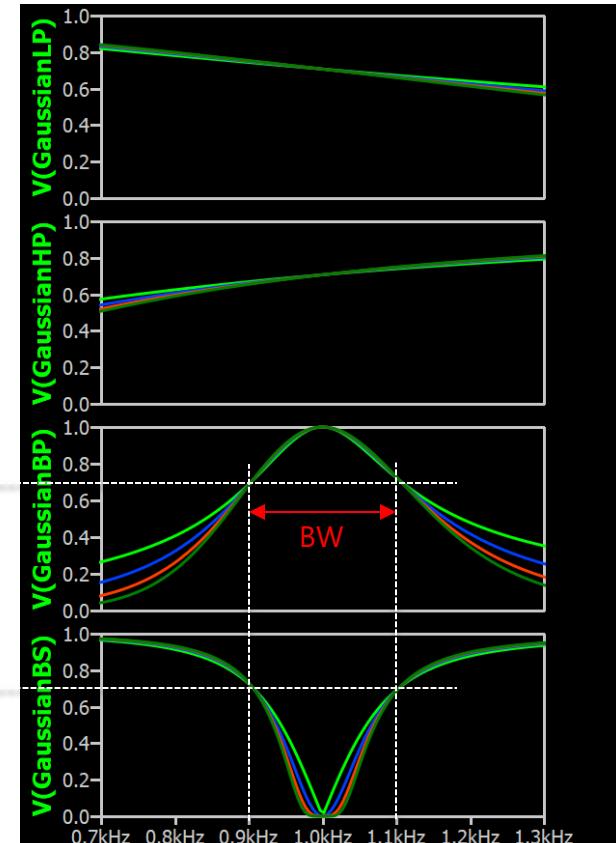
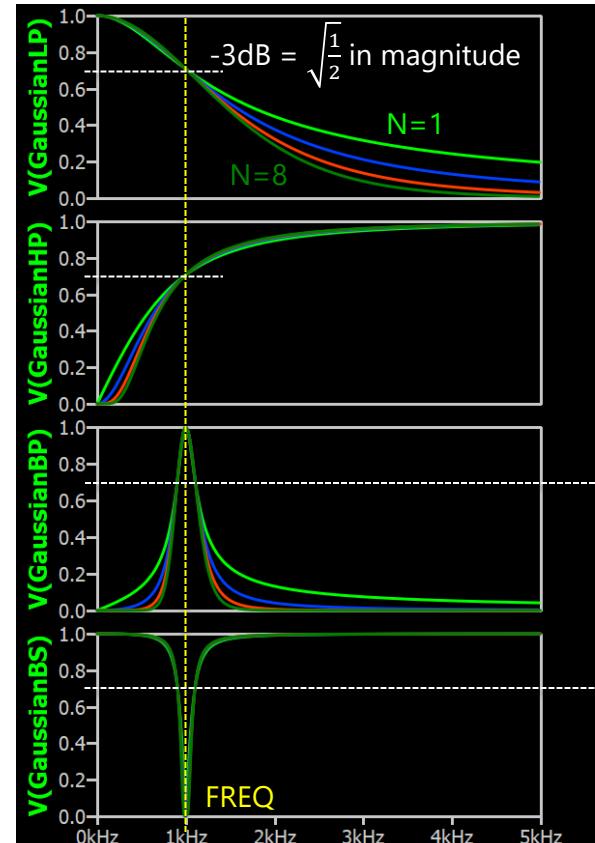
GaussianLP
B1
V=V(in)
Laplace=GaussianLP(N,Freq)

GaussianHP
B3
V=V(in)
Laplace=GaussianHP(N,Freq)

GaussianBP
B2
V=V(in)
Laplace=GaussianBP(N,Freq,BW)

GaussianBS
B4
V=V(in)
Laplace=GaussianBS(N,Freq,BW)

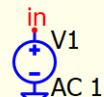
```
.plot V(GaussianBS)  
.plot V(GaussianBP)  
.plot V(GaussianHP)  
.plot V(GaussianLP)
```



B. Laplace – Filter Functions (Bessel Filter)

Qspice : B Source - Laplace - Bessel Filter.qsch

```
.param FREQ=1K  
.param BW=200  
.ac lin 1000 1 5K  
.step param N list 1 2 4 8
```



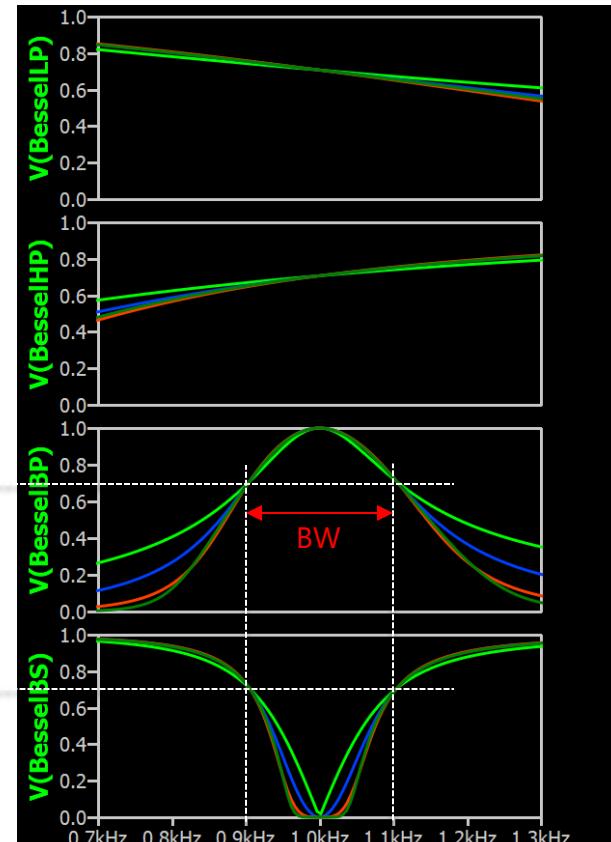
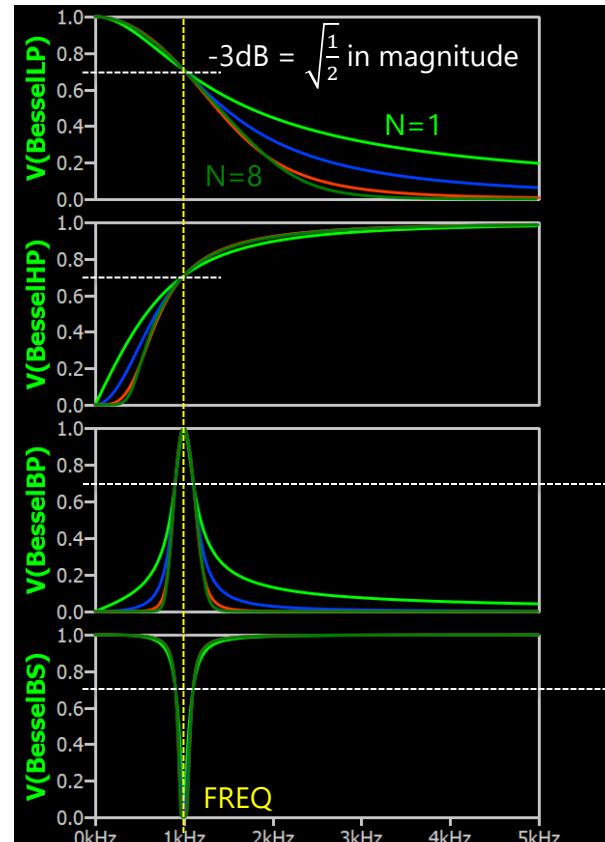
BesselLP
B1
V=V(in)
Laplace=BesselLP(N,Freq)

BesselHP
B3
V=V(in)
Laplace=BesselHP(N,Freq)

BesselBP
B2
V=V(in)
Laplace=BesselBP(N,Freq,BW)

BesselBS
B4
V=V(in)
Laplace=BesselBS(N,Freq,BW)

```
.plot V(BesselBS)  
.plot V(BesselBP)  
.plot V(BesselHP)  
.plot V(BesselLP)
```



B. Laplace – Filter Functions (Butterworth Filter)

Qspice : B Source - Laplace - Butterworth Filter.qsch

```
.param FREQ=1K  
.param BW=200  
.ac lin 1000 1 5K  
.step param N list 1 2 4 8
```



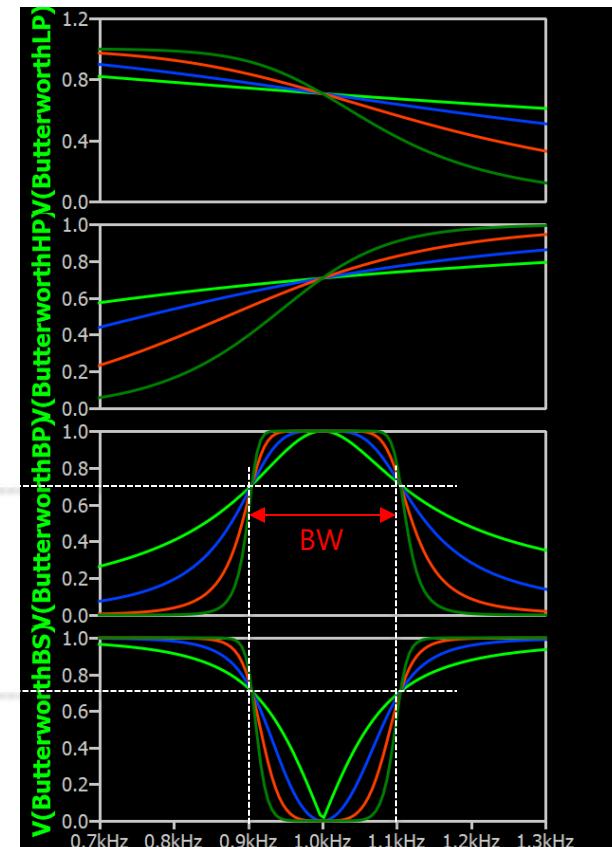
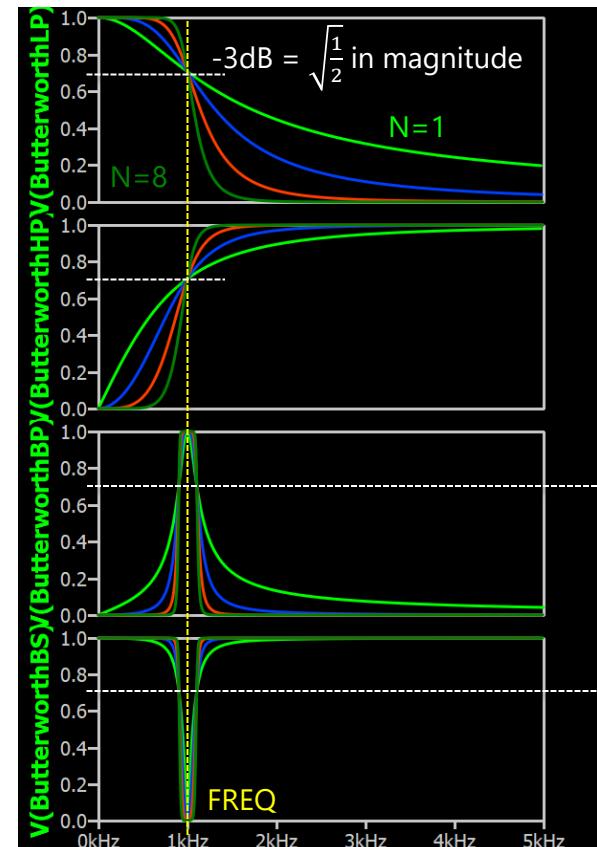
ButterworthLP
B1
V=V(in)
Laplace=ButterworthLP(N,Freq)

ButterworthHP
B3
V=V(in)
Laplace=ButterworthHP(N,Freq)

ButterworthBP
B2
V=V(in)
Laplace=ButterworthBP(N,Freq,BW)

ButterworthBS
B4
V=V(in)
Laplace=ButterworthBS(N,Freq,BW)

```
.plot V(ButterworthBS)  
.plot V(ButterworthBP)  
.plot V(ButterworthHP)  
.plot V(ButterworthLP)
```



B. Laplace – Filter Functions (Chebyshev Filter)

Qspice : B Source - Laplace - Chebyshev Filter.qsch

```
.param FREQ=1K  
.param BW=200  
.param dB=1  
.ac lin 1000 1 5K  
.step param N list 1 2 4 8
```



ChebyshevLP

B1
V=V(in)
Laplace=ChebyshevLP(N,Freq,dB)

Example : 1dB ripple

ChebyshevHP

B3
V=V(in)
Laplace=ChebyshevHP(N,Freq,dB)

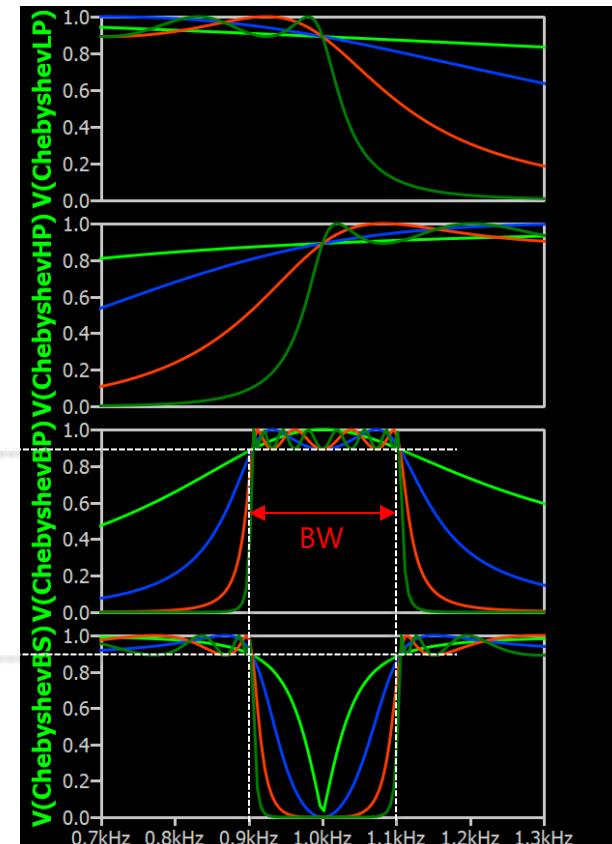
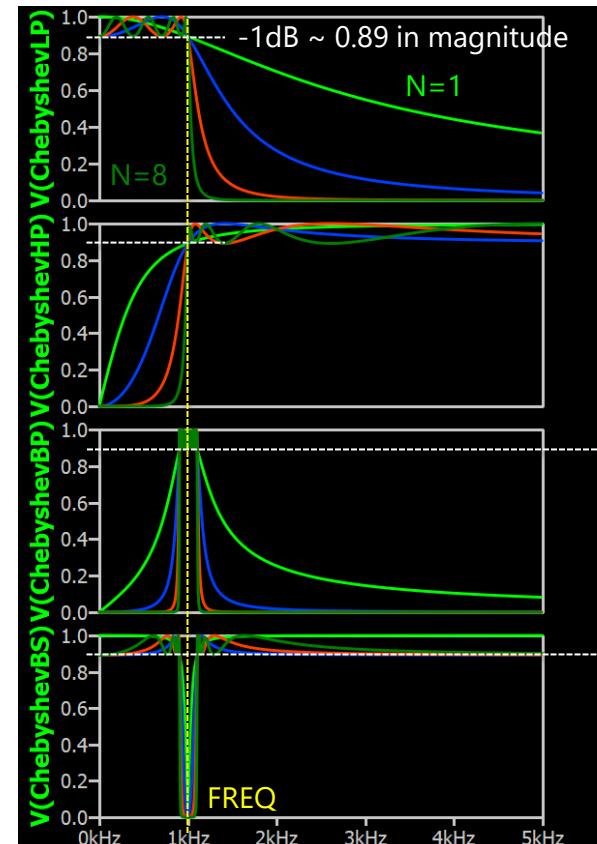
ChebyshevBP

B2
V=V(in)
Laplace=ChebyshevBP(N,Freq,dB,BW)

ChebyshevBS

B4
V=V(in)
Laplace=ChebyshevBS(N,Freq,dB,BW)

```
.plot V(ChebyshevBS)  
.plot V(ChebyshevBP)  
.plot V(ChebyshevHP)  
.plot V(ChebyshevLP)
```



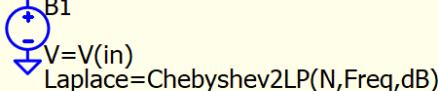
B. Laplace – Filter Functions (Chebyshev2 Filter)

Qspice : B Source - Laplace - Chebyshev2 Filter.qsch

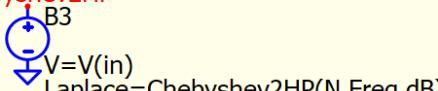
```
.param FREQ=1K  
.param BW=200  
.param dB=20  
.ac lin 1000 1 5K  
.step param N list 1 2 4 8
```



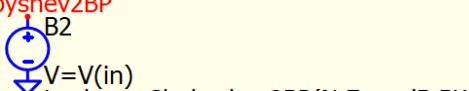
Chebyshev2LP Example : 20dB rejection



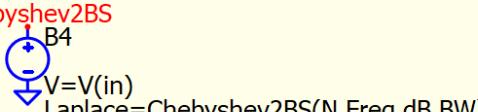
Chebyshev2HP



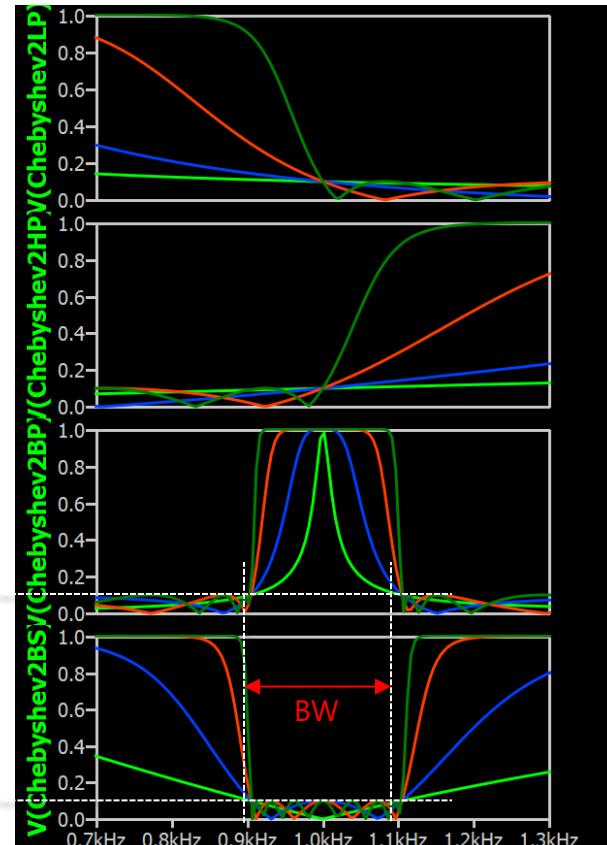
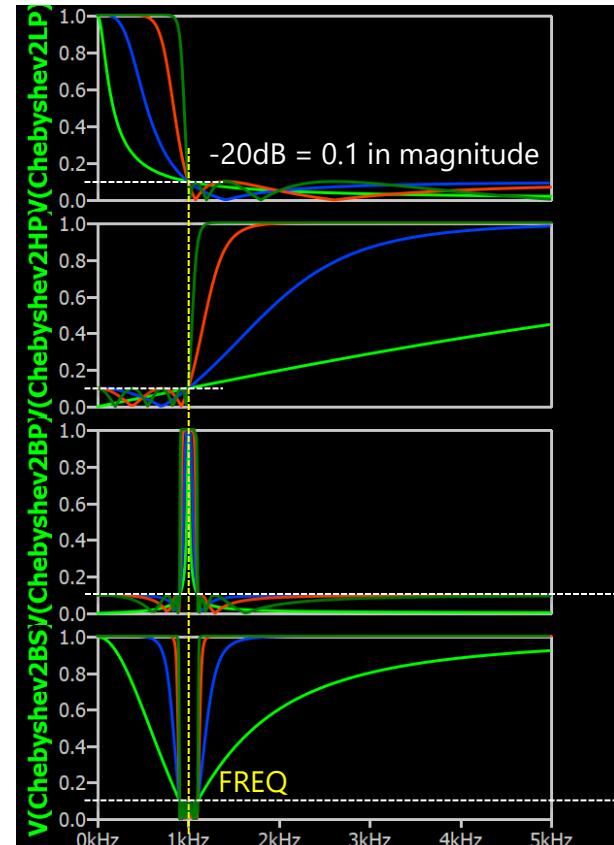
Chebyshev2BP



Chebyshev2BS



```
.plot V(Chebyshev2BS)  
.plot V(Chebyshev2BP)  
.plot V(Chebyshev2HP)  
.plot V(Chebyshev2LP)
```

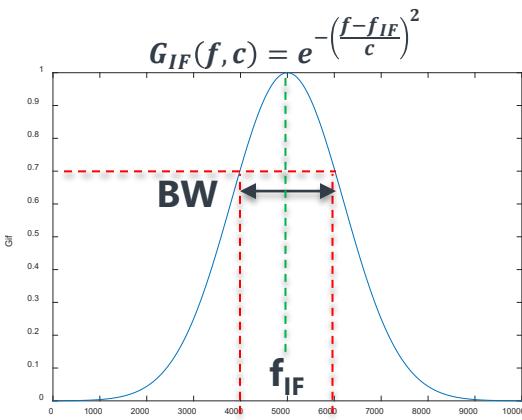


B. Laplace – Filter Functions (HELP : Laplace s-Domain Support)

Qspice : B Source - Laplace - GaussianBP.qsch | Gaussian_Filter_Formula.m

- Example of Laplace Gaussian Band Pass Filter

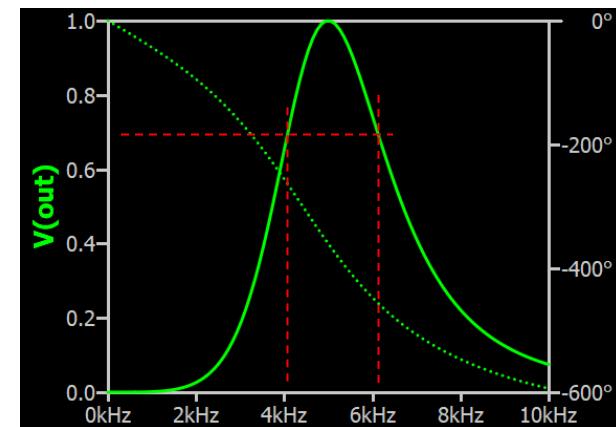
- General form of Gaussian filter frequency response : $G_{IF}(f, c) = e^{-\left(\frac{f-f_{IF}}{c}\right)^2}$
 - f_{IF} is center frequency
- Bandwidth (BW) is defined as $f_{BW} = f_{IF} \pm \frac{BW}{2}$ where $G_{IF}(f_{BW}, c) = \frac{1}{\sqrt{2}} \approx 0.7071$
 - Bandwidth refers to the width of spectral band at half of its maximum power or $\frac{1}{\sqrt{2}}$ of its maximum voltage
- Therefore, $c = \frac{BW}{2\sqrt{-\ln(\frac{1}{\sqrt{2}})}} \text{ or } BW = 2\sqrt{-\ln(\frac{1}{\sqrt{2}})} \times c$
- Qspice : GaussianBP(N, Freq, BW), where N is filter order, Freq is f_{IF} and BW is Bandwidth



```
.param fif=5000
.param c=1699
.param BW=2*sqrt(-log(1/sqrt(2)))*c
.option listparam
.in V1
.out B1
V=V(in)
Laplace = GaussianBP(4,fif,BW)
.ac lin 100 1 10000
.plot V(out)

.meas Vmax max mag(V(out))
.meas fcenter FIND frequency WHEN mag(V(out))=Vmax
.meas fL FIND frequency WHEN mag(V(out))=Vmax/sqrt(2) rise=1
.meas fH FIND frequency WHEN mag(V(out))=Vmax/sqrt(2) fall=last
.meas BandWidth fH-fL
```

an example of 4th-order



B. Laplace – Filter Functions (HELP : Laplace s-Domain Support)

Qspice : B Source - Laplace - GaussianBP.qsch | Gaussian_Filter_Formula.m

- Appendix : Derive c from f_{IF} and BW
 - Substitute $G_{IF}(f_{BW}, c) = \frac{1}{\sqrt{2}}$ and $f_{BW} = f_{IF} \pm \frac{BW}{2}$ into general formula $G_{IF}(f, c) = e^{-\left(\frac{f-f_{IF}}{c}\right)^2}$
 - $\frac{1}{\sqrt{2}} = e^{-\left(\frac{f_{IF} \pm \frac{BW}{2} - f_{IF}}{c}\right)^2} = e^{-\left(\frac{\pm \frac{BW}{2}}{c}\right)^2} = e^{-\left(\frac{BW}{2c}\right)^2}$
 - $\ln\left(\frac{1}{\sqrt{2}}\right) = -\left(\frac{BW}{2c}\right)^2 \rightarrow \sqrt{-\ln\left(\frac{1}{\sqrt{2}}\right)} = \frac{BW}{2c}$
 - $BW = 2\sqrt{-\ln\left(\frac{1}{\sqrt{2}}\right)} \times c \text{ OR } c = \frac{BW}{2\sqrt{-\ln\left(\frac{1}{\sqrt{2}}\right)}}$

Compare 2nd Order LPF of Gaussian, Bessel and Butterworth

Qspice : B Source - Laplace - Compare.qsch

- 2nd Order LPF
 - Compare Gaussian, Bessel and Butterworth Low-Pass 2nd order filter
 - Butterworth : maximally flat magnitude response
 - Bessel : maximally flat group delay
 - Gaussian : no overshoot in step response
 - Butterworth 2nd order filter yields identical response of 2nd order system with damping factor $\zeta = 0.707$
 - 2nd order LPF with 40dB/decade of gain roll-off after cutoff frequency

```
.param Fc=10K
.param N=2
.ac dec 100 100 100K
.in V1
.out AC1
.plot V(GaussianLP) V(BesselLP) V(ButterworthLP)
+V(FirstOrderLP) V(SecondOrderLP)

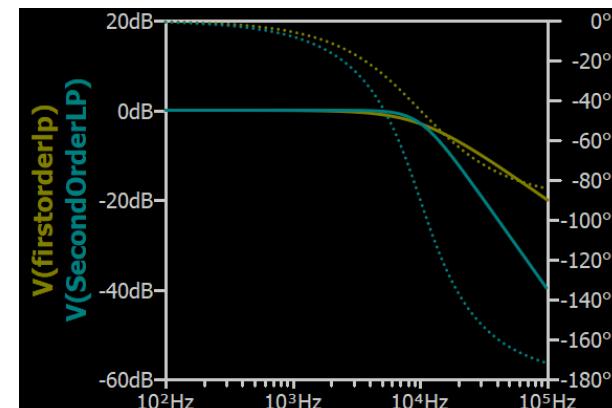
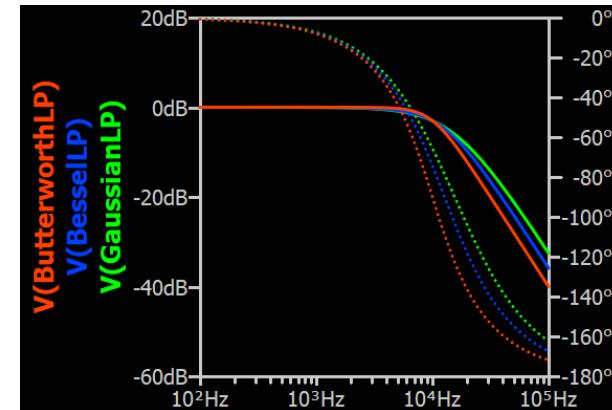
GaussianLP
.in E1
.out Laplace=GaussianLP(N,Fc)

BesselLP
.in E2
.out Laplace=BesselLP(N,Fc)

ButterworthLP
.in E3
.out Laplace=ButterworthLP(N,Fc)

FirstOrderLP
.in E4
.out Laplace=a/(s+a)
.param a=2*pi*Fc

SecondOrderLP
.in E5
.out Laplace=wn^2/(s^2+2*zeta*wn*s+wn^2)
.param wn=2*pi*Fc
.param zeta=.707
.zeta=0.7 : equivalent to 2nd order ButterworthLP
```

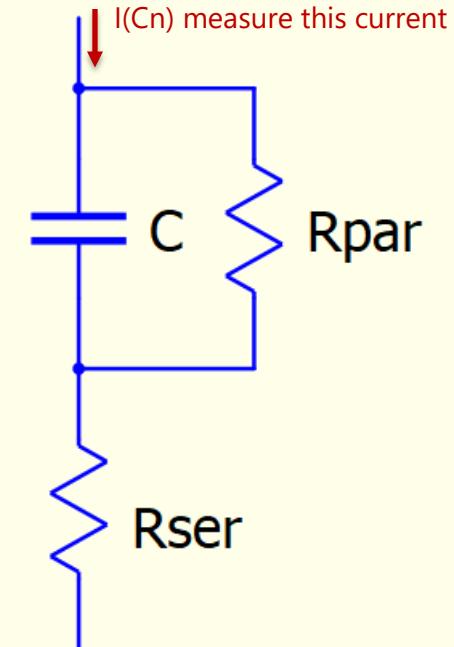


C. Capacitor

C. Capacitor Instance Parameters in Qspice HELP

Capacitor Instance Parameters			
Name	Description	Units	Default
CAPACITANCE	Capacitance(not usually labeled)	F	0.0
CSAT	Capacitance asymptotically approached in saturation	F	10% of CAPACITANCE
IC	Initial voltage on capacitor	V	0.0
L ¹	Length	m	0.0
M	Number of identical parallel devices		1.0
RPAR	Equivalent parallel resistance	Ω	Infinite
RSER	Equivalent series resistance	Ω	0.0
SATFRAC	Fractional drop in capacitance at VSAT		0.7
TC	Polynomial temperature coefficient list		0.0
TC1	1st order temperature coefficient	$^{\circ}\text{C}^{-1}$	0.0
TC2	2nd order temperature coefficient	$^{\circ}\text{C}^{-2}$	0.0
TEMP	Instance temperature	$^{\circ}\text{C}$	Circuit temperature
TNOM	Temperature capacitance was measured(aka TREF)	$^{\circ}\text{C}$	Circuit TNOM
VSAT	Voltage that drops capacitance to SATFRAC×CAPACITANCE	V	Infinite
W ¹	Width	m	0.0

Rpar and Rser in C model



C. Capacitor – Rser and Rpar in Qspice and Ltspice

Qspice : Capacitor - model verification.qsch

- Qspice vs LTspice

- Beware that Qspice and LTspice treat capacitor Rser and Rpar differently
- Capacitor conversion from LTspice to Qspice in .subckt or netlist

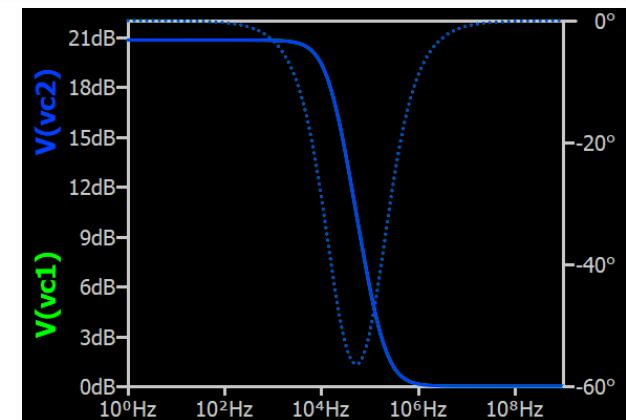
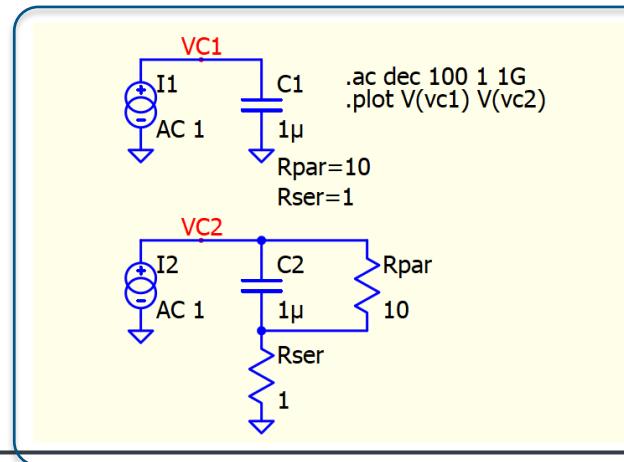
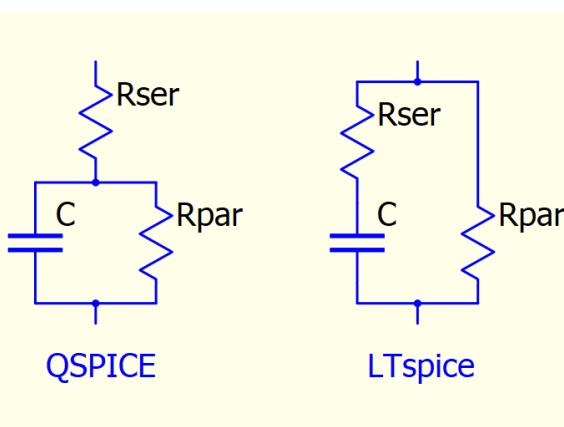
- Ltspice

`Cn N001 N002 <capacitance> Rser=x Rpar=y`

- Qspice

`Cn N001 N002 <capacitance> Rser=x`

`RparCn N001 N002 y`

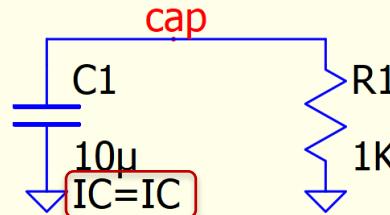


C. Instance Params : IC and M

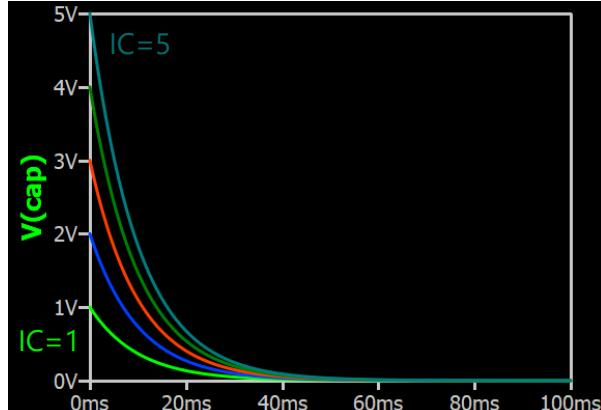
Qspice : Capacitor - IC.qsch | C Capacitor - M.qsch

- **IC**

- IC : Initial Condition
- **Default IC=0**
- Use in transient analysis for initial voltage of capacitor at time=0s

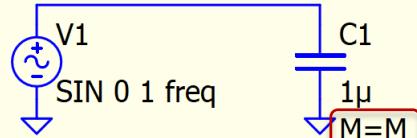


```
.step param IC 1 5 1  
.tran 100m  
.plot V(cap)
```



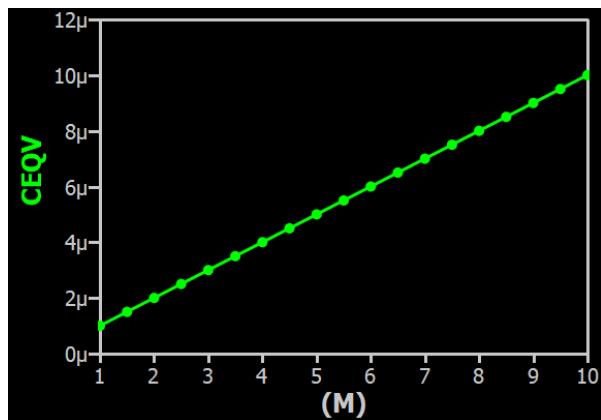
- **M**

- M : Number of identical parallel devices
- **Default M=1**



```
.param freq=1K  
.step param M 1 10 .5  
.tran 1/freq
```

Calculate equivalent capacitance in .tran analysis
.meas Ip max I(C1)
.meas Ceqv Ip/1/2/pi/freq
.meas plot Ceqv

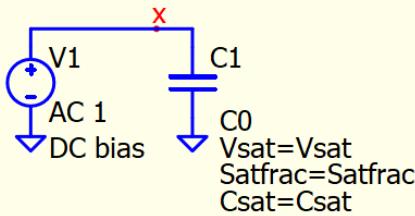


C. Instance Params : Vsat, Csat, Satfrac : For Nonlinear Capacitance

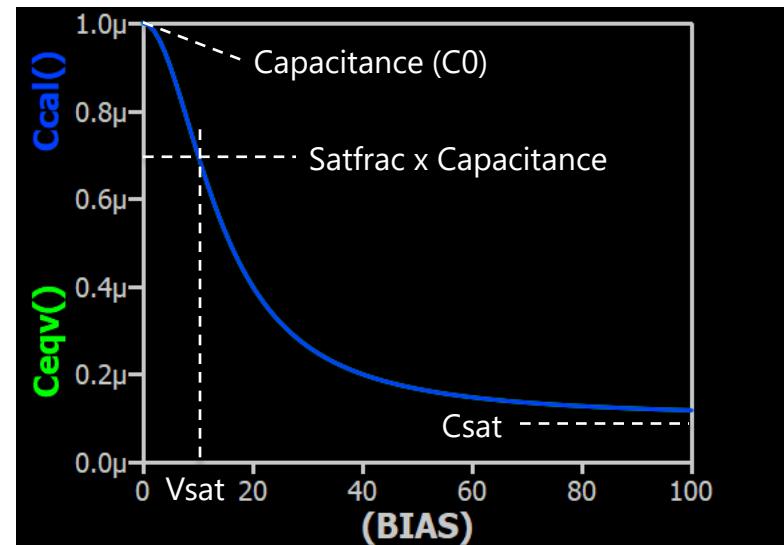
Qspice : C Capacitor - Csat Satfrac and Vsat - Eqn.qsch

- Formula of Vsat, Csat and Satfrac, with Capacitance= C_0

$$C(V_{DC}) = \frac{1}{\left(\frac{V_{DC}^2}{k} + \frac{1}{C_0 - C_{sat}}\right)} + C_{sat} \quad \text{where } k = \frac{V_{sat}^2}{\left(\frac{1}{C_0 \text{SATFRAC} - C_{sat}} - \frac{1}{C_0 - C_{sat}}\right)}$$



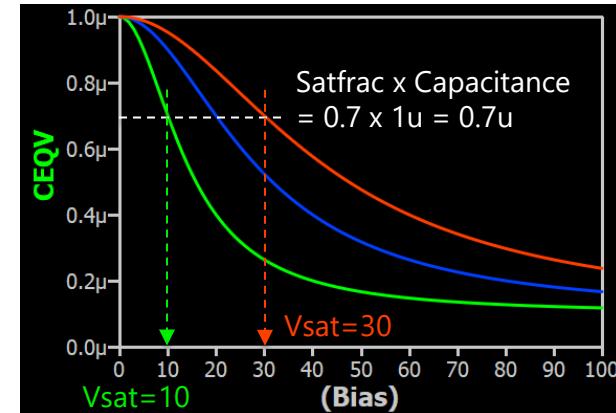
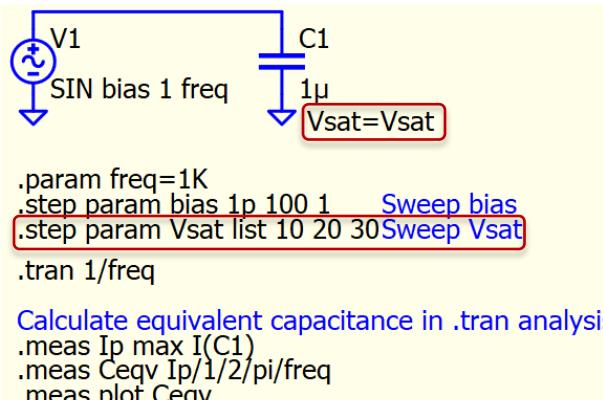
```
.param frq=1Meg
.step param bias 0 100 1
.ac list frq
.func Cequiv() I(V1)/1/2/pi/frq
.plot Cequiv() Ccal()
.param k1=Vsat^2/(1/(C0*Satfrac-Csat)-1/(C0-Csat))
.func Ccal() 1/((bias^2/k1)+1/(C0-Csat))+Csat
```



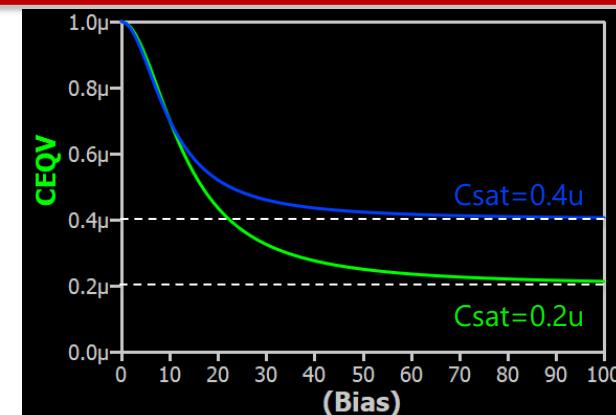
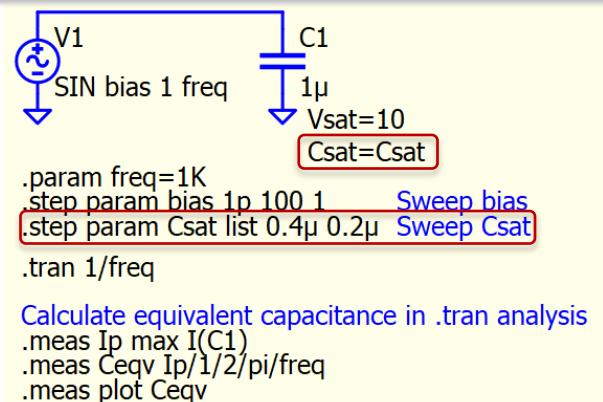
C. Instance Params : Vsat, Csat, Satfrac : For Nonlinear Capacitance

Qspice : C Capacitor - Vsat (.tran).qsch ; C Capacitor - Csat (.tran).qsch

- Vsat and Satfrac
 - Vsat : Voltage that drops capacitance to **Satfrac x Capacitance**
 - Satfrac : Fractional drop in capacitance at Vsat
 - **Default Vsat=Infinite (V)**
 - **Default Satfrac=0.7**



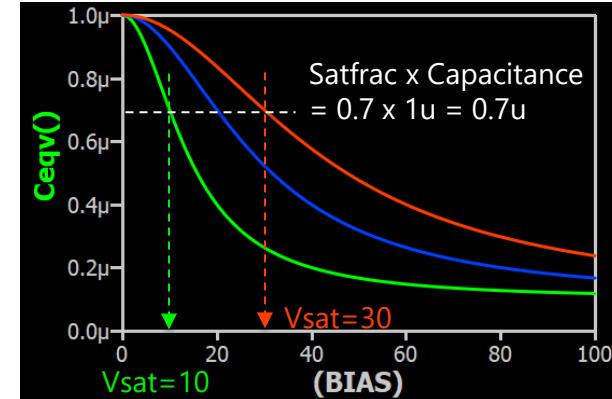
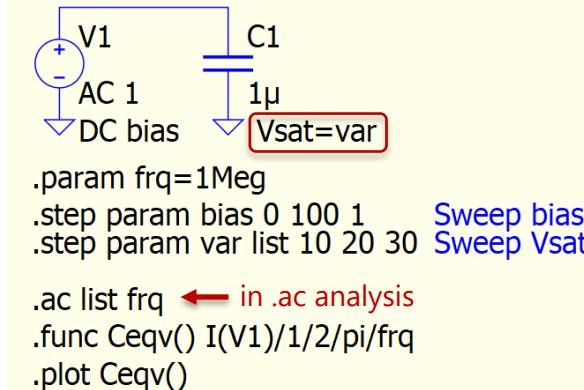
- Csat
 - Csat : Capacitance asymptotically approached in saturation
 - ** Vsat must be set to see its effect
 - **Default Csat=10% of C**



C. Instance Params : Replicate previous graph with .ac analysis

Qspice : C Capacitor - Vsat (.ac).qsch

- Vsat and Satfrac
 - Vsat : Voltage that drops capacitance to **Satfrac x Capacitance**
 - Satfrac : Fractional drop in capacitance at Vsat
 - **Default Vsat=Infinite (V)**
 - **Default Satfrac=0.7**



- Capacitance equation
 - $X_C = \frac{1}{\omega C} = \frac{V_c}{I_c}$
 - $C = \frac{|I_c|}{|V_c| \omega} = \frac{|I_c|}{2\pi f |V_c|}$
 - This formula is used to calculate equivalent capacitance (small signal model) at different capacitor bias voltage

C. Instance Params : Thermal Parameter

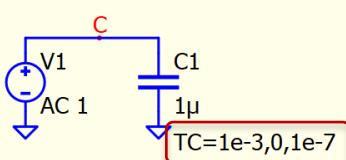
- Thermal Equation of Capacitor
 - $C = C_{nom} \times (1 + \sum TC_n (T - T_{nom})^n)$
 - Where T is operating temperature TEMP, which uses as `.temp`, `.dc temp` or `.step param temp`
 - Thermal Instance Parameter and equation relationship
 - TC : Polynomial of Thermal Coefficient where $TC = <TC1>, <TC2>, <TC3>, <TC4>, \dots$
 - $TC_n = TC_n$
 - TC1, TC2 : 1st and 2nd order Thermal Coefficient
 - $TC1 = TC_1$ and $TC2 = TC_2$
- TEMP : Instance Temperature (Simulation Temperature of this capacitor)
- TNOM : Temperature Capacitance was measured
 - C_{nom} : Capacitance C @ TNOM

C. Instance Params : TC, TC1 and TC2

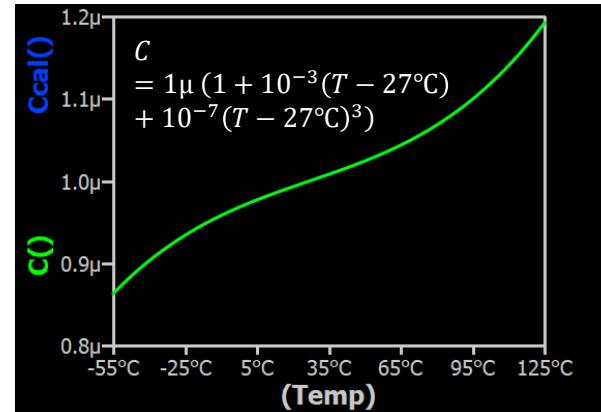
Qspice : C Capacitor - TC.qsch | C Capacitor - TC1 TC2.qsch

- TC

- TC : Temperature Coefficient list
- Default TC=0**
- TC can be defined as
 $TC = <\text{value1}>, <\text{value2}> \dots$
- Formula
- $C = C_{nom} (1 + \sum TC_n (T - T_{nom})^n)$
- In this example, Default T_{nom} is used, which is 27°C .

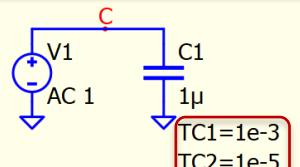


```
.step param TEMP -55 125 1
.param frq=1Meg
.ac list frq
.func imZ() V(C)/I(C1)
.func C() -1/2/pi/frq/imZ()
.plot C() Ccal()
.func Ccal() 1μ*(1+1e-3*(TEMP-27)+1e-7*(TEMP-27)^3)
```

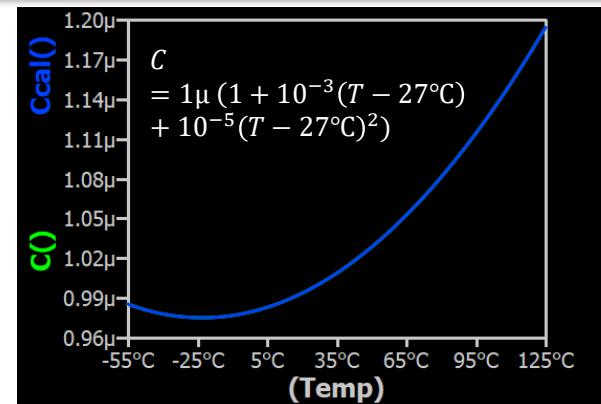


- TC1 and TC2

- TC1 : 1st order temp coeff
- TC2 : 2nd order temp coeff
- $C = C_{nom} (1 + TC_1 (T - T_{nom}) + TC_2 (T - T_{nom})^2)$



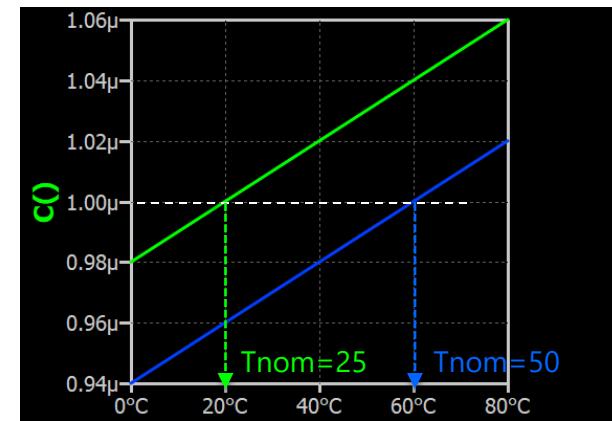
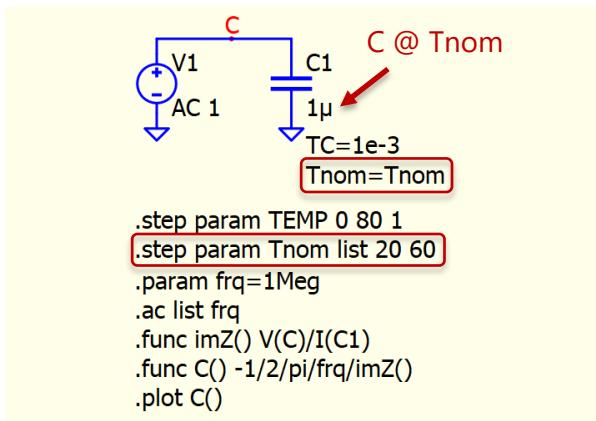
```
.step param TEMP -55 125 1
.param frq=1Meg
.ac list frq
.func imZ() V(C)/I(C1)
.func C() -1/2/pi/frq/imZ()
.plot C() Ccal()
.func Ccal() 1μ*(1+1e-3*(TEMP-27)+1e-5*(TEMP-27)^2)
```



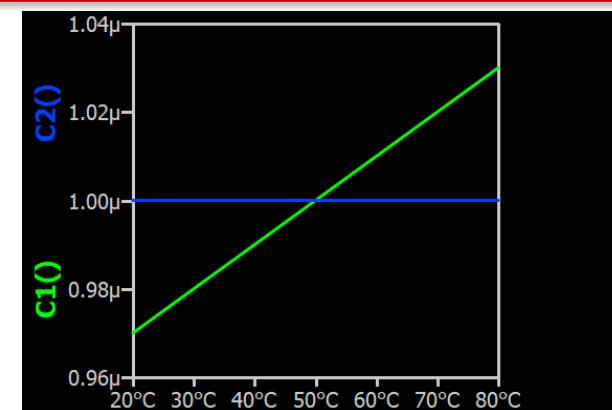
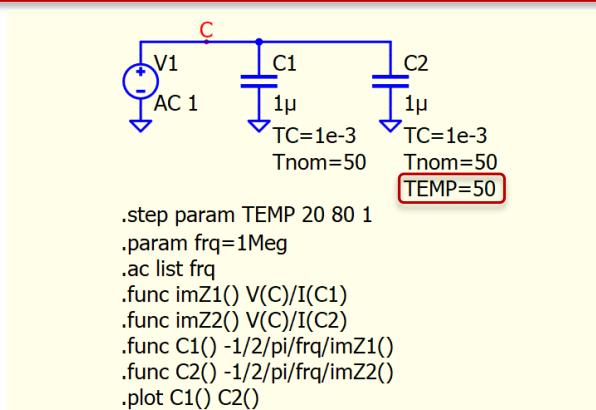
C. Instance Params : Tnom and TEMP

Qspice : C Capacitor - Tnom.qsch | C Capacitor - Temp.qsch

- Tnom
 - Nominal temperature
 - **Default Tnom=27°C**
 - Capacitance Cnom in thermal equation is measured at Tnom (i.e. at Tnom, calculated capacitance equal model capacitance)



- TEMP
 - Instance temperature
 - **Default TEMP=Circuit Temp**
 - With TEMP, capacitor is fixed at this temperature in regardless change of circuit temperature



C. Capacitor – Behavioral Capacitance ($C=$)

Qspice : C Capacitor - Behavioral C.qsch

- $C=$
 - Behavioral Capacitance
 - **Format : $C=<\text{equation}>$**
 - Qspice supports behavioral capacitance for a given equation which can include any circuit node voltage or device current

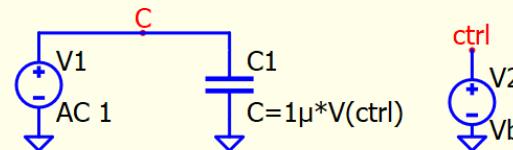
Arbitrary Capacitance-Based Capacitance Device

QSPICE also supports a behavioral capacitance where an equation for charge is not available. To use it, give an equation for the capacitance is given. The equation can include any circuit node voltages or device currents of devices modeled as Thévenin equivalents(V-, L-, E-, or H-devices). An original numerical method is used to track the charge on the capacitance not before seen in SPICE programs.

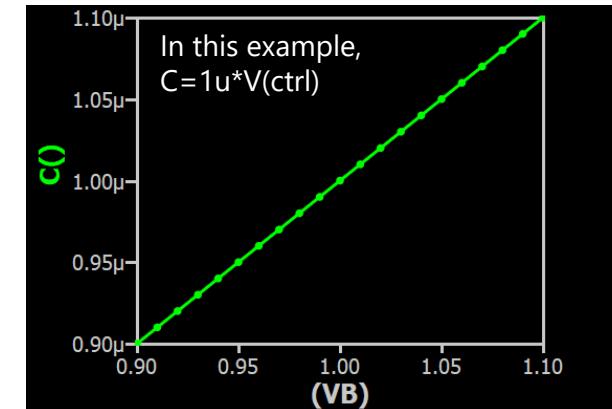
Syntax: Cnnn N1 N2 C=<expression> [additional instance parameters]

Name	Description	Units	Default
C	Equation of charge	Coulomb	
IC	Initial voltage on capacitor	V	0.0
M	Number of identical parallel devices		1.0
RPAR	Equivalent parallel resistance	Ω	Infinite
NOISELESS	Ignore the noise contribution from RPAR		not set
TEMP	Instance temperature	$^{\circ}\text{C}$	Circuit temperature

Behavioral Capacitor Model with $C=<\text{eqn}>$



```
.step param Vb 0.9 1.1 0.01
.param frq=1Meg
.ac list frq
.func C() -1/2/pi/frq/(im(V(C)/I(C1)))
.plot C()
```



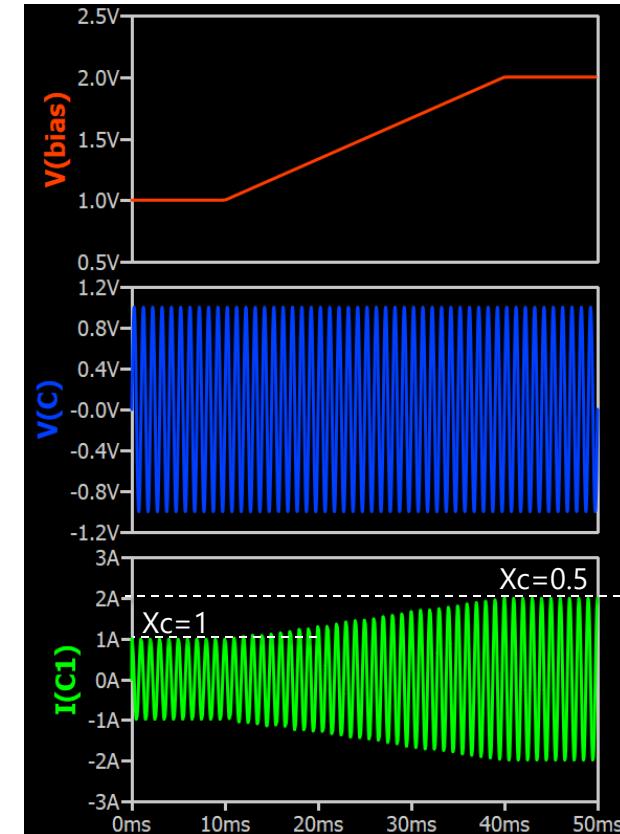
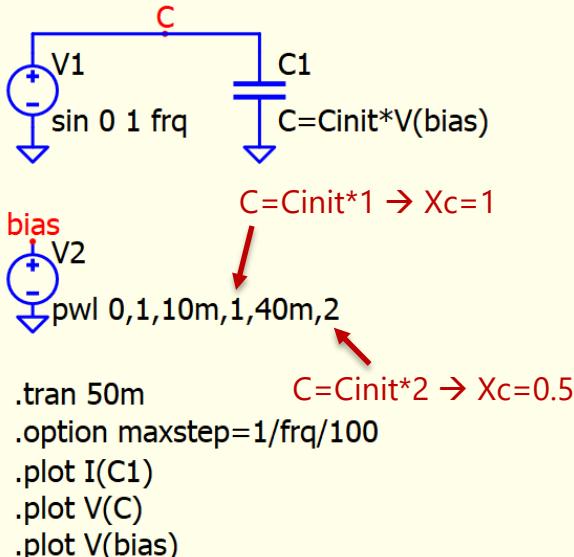
C. Capacitor – Behavioral Capacitance ($C=$)

Qspice : C Capacitor - Behavioral C (.tran).qsch

- $C=$
 - Behavioral Capacitance
 - This schematic demonstrate using behavioral capacitance model in transient (.tran) analysis
 - This example assume capacitance is linear to its bias voltage

Behavioral Capacitor model with $C=<\text{eqn}>$

```
.param frq=1K
.param Xc=1
.param Cinit=1/2/pi/frq/Xc
```



C. Capacitor – Arbitrary Charge-Based Capacitance (Q=)

Qspice : C Capacitor - Q - Basic.qsch

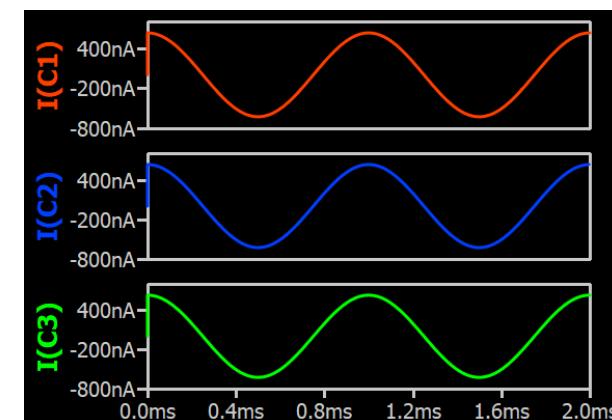
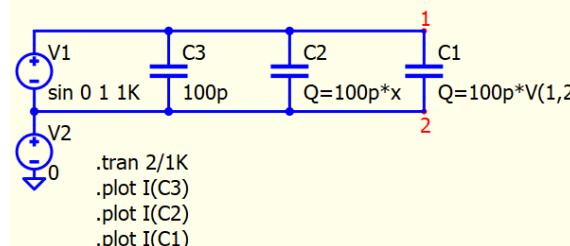
- $Q =$
 - Arbitrary Charge-Based Capacitance Device model
 - **Format : $Q = <\text{equation}>$**
 - Basic relationship of charge, capacitance and voltage is
$$Q = C \times V$$
 - This formula only correct if C is not depending on capacitor voltage
- In Qspice, special variable **x**, represent voltage across the device
- For example, a 100pF constant capacitance can be written as
- $Q = 100p \cdot x$
- $Q = 100p \cdot V(1,2)$
 - If node name are defined

Arbitrary Charge-Based Capacitance Device

QSPICE supports a charge-based arbitrary capacitance device. To use it, give an equation for the charge on the capacitor. The equation can include any circuit node voltages or device currents of devices modeled as Thévenin equivalents(V-, L-, E-, or H-devices). It supports transcapacitance which is common in charge-conserving transistor charge models.

Syntax: Cnnn N1 N2 Q=<expression> [additional instance parameters]

Name	Description	Units	Default
Q	Equation of charge	Coulomb	
IC ²	Initial charge or voltage on capacitor	Coulomb or Volts	
M	Number of identical parallel devices		1.0
RPAR	Equivalent parallel resistance	Ω	Infinite
NOISELESS	Ignore the noise contribution from RPAR		not set
TEMP	Instance temperature	$^{\circ}\text{C}$	Circuit temperature



C. Capacitor – Arbitrary Charge-Based Capacitance ($Q=$)

Qspice : C Capacitor - Q - VoltageDependentC.qsch

- Voltage dependent C

- If C is a function of capacitor voltage V_c

$$Q = \int C(V_c) dV_c$$

- If $C(V_c) = m_c V_c + c_0$

- $Q = \int (m_c V_c + c_0) dV_c$

- $Q = \frac{1}{2} m_c (V_c)^2 + c_0 V_c$

- It is important that if capacitor voltage is in capacitance formula, charge formula requires this integration**

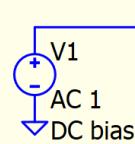
- This implementation can directly be done with $C=<\text{eqn}>$ format

- If $C = m_c V_{bias} + c_0$

- As voltage is only an external parameter but not capacitor voltage, no integration is required

- $Q = CV = (m_c V_{bias} + c_0) V_c$

Three Equivalent Implementation

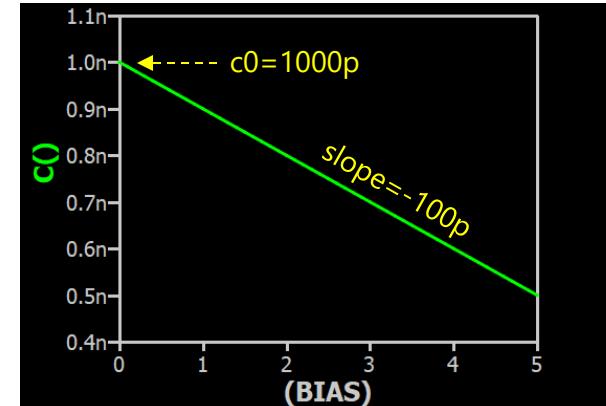


a

C1

$$\begin{aligned}Q &= 1/2 * mc * x^{**2} + c0 * x \\Q &= (mc * bias + c0) * x \\C &= mc * x + c0\end{aligned}$$

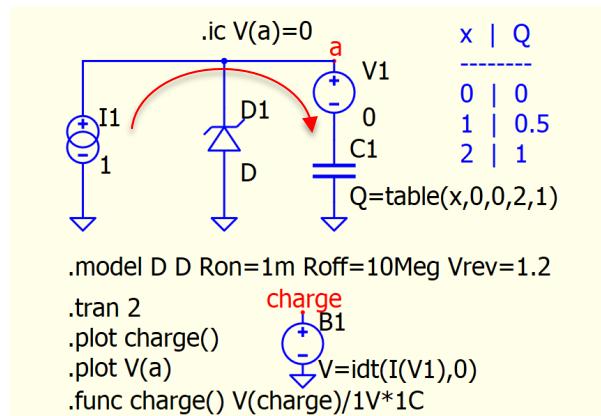
```
.param frq=1Meg           .param c0=1000p
.ac list frq              .param mc=-100p
.step param bias 0 5 0.1
.func C() -1/2/pi/frq/imag(V(a)/I(C1))
.plot C()
```



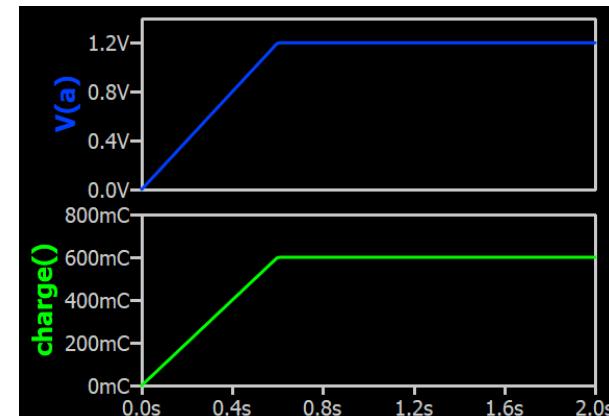
C. Capacitor – Arbitrary Charge-Based Capacitance ($Q=$)

Qspice : C Capacitor - Q - QModel-Charge.qsch

- Special – Charge model
 - $Q=<\text{eqn}>$ can be used to define a voltage dependent charge device
 - Charge example**
 - $Q=\text{table}(x,0,0,2,1)$ defines a charge model where the voltage across the capacitor symbol determines the total charge in the device at that voltage (i.e. 0V, 0C ; 2V, 1C)
 - The current flowing into C_1 accumulates the charge Q and results in $V(a)$
 - Diode D_1 is set as a zener diode to clamp the charging voltage only up to 1.2V, ensuring that the charge does not exceed the table() limitations



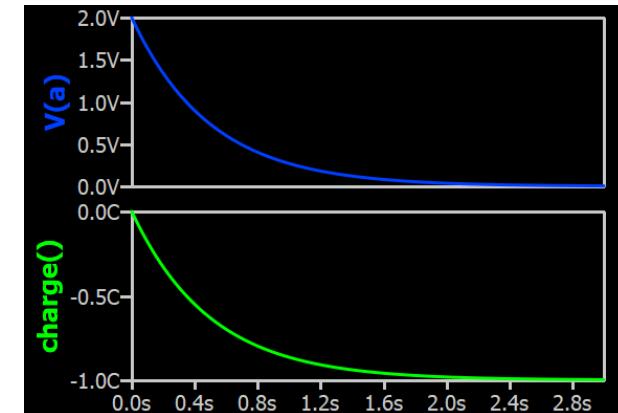
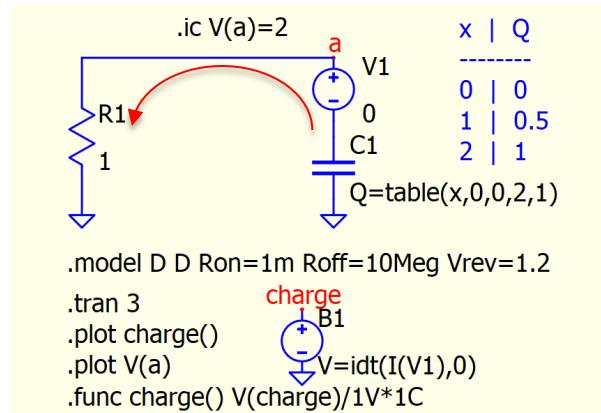
- Initial condition (.ic) is given for $V(a)=0$ to force C_1 to have no charge ($Q=0C$) at $t=0$ in .tran



C. Capacitor – Arbitrary Charge-Based Capacitance ($Q=$)

Qspice : C Capacitor - Q - QModel-DisCharge.qsch

- Special – Charge model
 - **Discharge Example**
 - In this example, with an initial $V(a)=2$, $C1$ has an initial charge $Q=1C$ at $t=0$
 - The discharge current is determined by $V(a)$ and load, which results in drawing of charge from $C1$ over time
 - Consequently, maximum amount of charge that can be drawn is the initial charge determined by the initial condition
 - One application of a voltage-dependent charge device is simulating a battery, where $3.6C=1mAh$



- Initial condition (.ic) is given for $V(a)=2$ to force $C1$ to have full charge ($Q=1C$) at $t=0$ in .tran

C. Capacitor – Why Qspice not include parasitic inductance (ESL or Lser)

- Answer by Mike Engelhardt in Qspice forum
 - <https://forum.qorvo.com/t/parasitic-inductance-of-passive-components-lser/15608/3>
 - <https://forum.qorvo.com/t/esl-in-capacitor-model/16944/2>

 Engelhardt Oct '23

I did not implement Lser for capacitors because it usually caused more harm than good.

A series inductance is not a great way to model the internal inductance of a capacitor. For foil capacitors, you really need a ladder of series R & L driving a distributed capacitance, though I've been able to match the complex impedance within a few degrees over several decades of frequency with just two lumps.

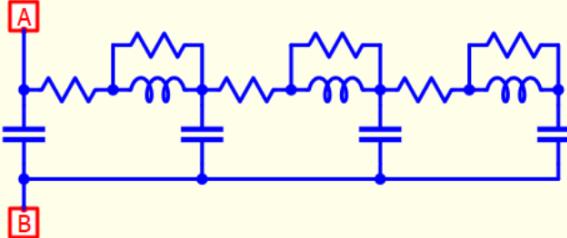
For ceramic capacitors, modeling their internal inductance is not really meaningful without modeling the parasitics of the PCB.

Anyway, when I watch people using Lser, it only increases simulation times without actually improving the integrity of the simulation.

Your mileage will vary,
-Mike



For any type of film capacitor, it is more of a distributed $R_{ser}/R_{par}/L_{ser}/C_{par}$ network. Poorly described with a single lumped series L. For a ceramic capacitor, a series inductance can be a more viable representation, but usually adds no accuracy to the simulation since connecting inductance often dominates. It was a mistake to support Lser as a part of a capacitor in LTspice. It did much more harm than good. Anyway, below would be a viable equivalent schematic of a film capacitor in lumped constants between nodes A and B:



The diagram shows a ladder network of resistors (wavy lines) and capacitors (parallel lines) between two nodes, A and B. The top row consists of four resistor-capacitor (RC) pairs connected in series. The bottom row consists of three capacitor-resistor (CR) pairs connected in parallel with the top row. Node A is at the top left, and node B is at the bottom right. This represents a distributed model of a capacitor's internal behavior.

Notice there is nowhere a series inductance.

D. Diode

Model Parameters

Diode Model Parameters in Qspice HELP

Diode Instance Parameters

Name	Description	Units	Default
AREA	Relative area		1.0
M	Number of identical parallel devices		1.0
TEMP	Instance temperature	°C	Circuit temperature

Diode Model Parameters

Name	Description	Units	Default
AF	Flicker noise exponent	1.	1.0
BV	Reverse breakdown voltage	V	Infinite
CJO	Zero-bias junction capacitance	F	0.0
EG	Activation energy	eV	1.11
FC	Forward-bias depletion capacitance coefficient		0.5
GMAX	Maximum conductivity(straight-line extension)	Ω	10000.
GP	Parallel conductivity added in lieu of global Gmin	Ω	0.
IBV	Current at breakdown voltage	A	1e-10
IBVL	Low-level reverse breakdown knee current	A	0.0
IKF	High-injection knee current	A	1e308
IS	Saturation current	A	1e-14
ISR	Recombination current parameter	A	0.0
KF	Flicker noise coefficient		0.0
M	Grading coefficient		0.5

N	Emission coefficient	1.0
NBV	Reverse breakdown emission coefficient	1.0
NBVL	Low-level reverse breakdown emission coefficient	1.0
NR	ISR emission coefficient	2.0
RS	Series resistance	Ω
TBV1	1st order BV temperature coefficient	°C⁻¹
TBV2	2nd order BV temperature coefficient	°C⁻²
TIKF	IKF temp coefficient	°C⁻¹
TNOM	Parameter measurement temperature(aka TREF)	°C
TGP1	1st order GP temperature coefficient¹	°C⁻¹
TGP2	2nd order GP temperature coefficient¹	°C⁻²
TRS1	1st order RS temperature coefficient¹	°C⁻¹
TRS2	2nd order RS temperature coefficient¹	°C⁻²
TT	Transit-time²	sec
VJ	Junction potential	V
VP	dQ/dt damping parameter for diffusion charge	0.01
XTI	Saturation current temperature exponent	3.0

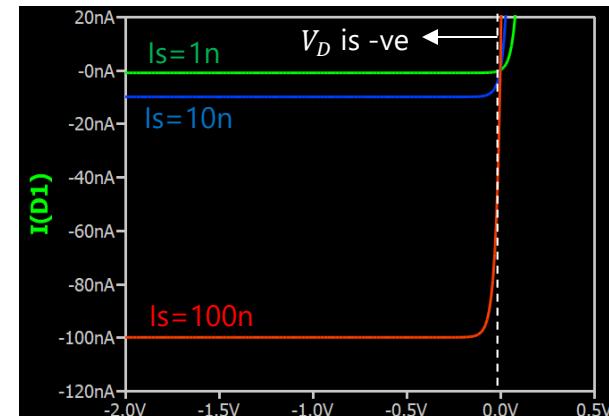
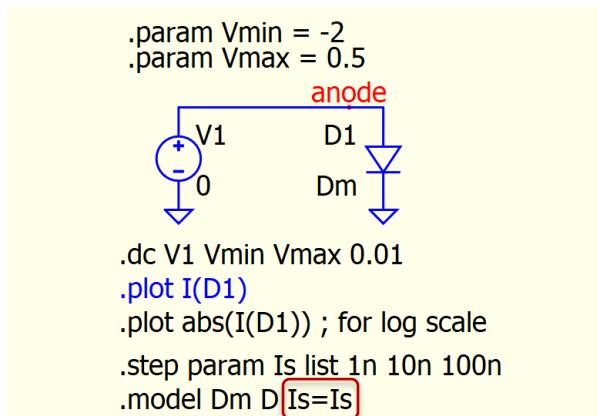
.model parameters only for information display but no electrical behavior

- MFG : manufacturer name
- Vrev : Peak reverse voltage
- Iave : Average current rating

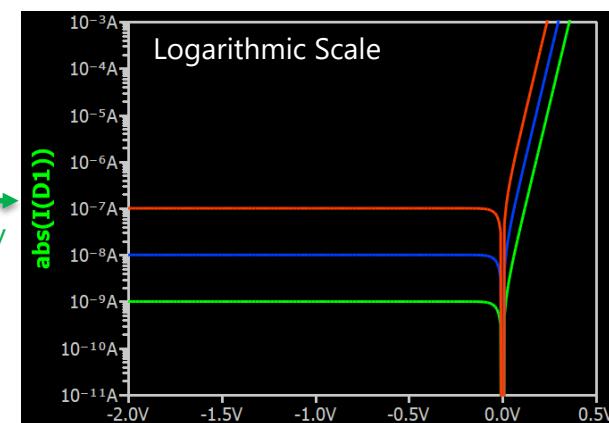
D. Diode (I-V Characteristic) : IS

Qspice : Dmodel - IS.qsch

- IS
 - Is : Saturation current
 - Saturation current in reverse region
 - **Default IS = 1e-14A**
 - $I_D = I_s \left(e^{\frac{qV_D}{nkT}} - 1 \right)$
 - $\frac{kT}{q}$ is called thermal voltage
 - q : electronic charge
 - 1.602176487e-19 Coulomb
 - k : Boltzmann constant
 - 1.380649e-23 J/K
 - T : Temperature in Kelvin
 - $T = T_{celsius} + 273.15^\circ C$
 - V_D : Diode voltage
 - n : emission coefficient



$|I_d|$ →
Magnitude only

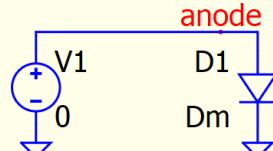


D. Diode (I-V Characteristic) : N and Rs

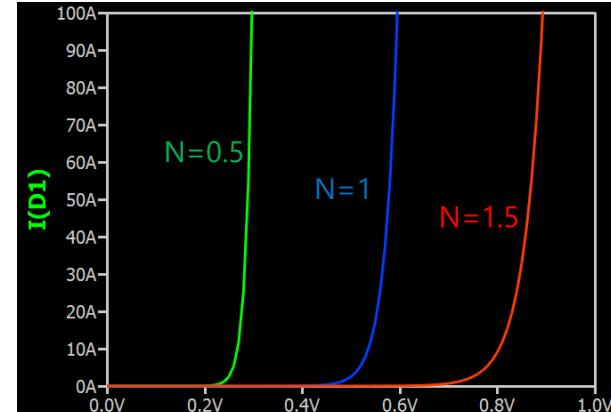
Qspice : Dmodel - N.qsch / Dmodel - Rs.qsch

- N
 - n : emission coefficient
 - **Default N=1**
 - $I_D = I_s (e^{\frac{qV_D}{nkT}} - 1)$
 - N affects threshold voltage in forward bias

```
.param Vmin = 0  
.param Vmax = 1  
.step param n list 0.5 1 1.5
```

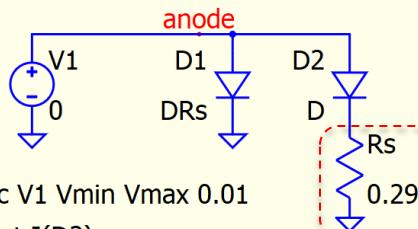


```
.dc V1 Vmin Vmax 0.01  
.plot I(D1)  
.model Dm D Is=10n N=n
```

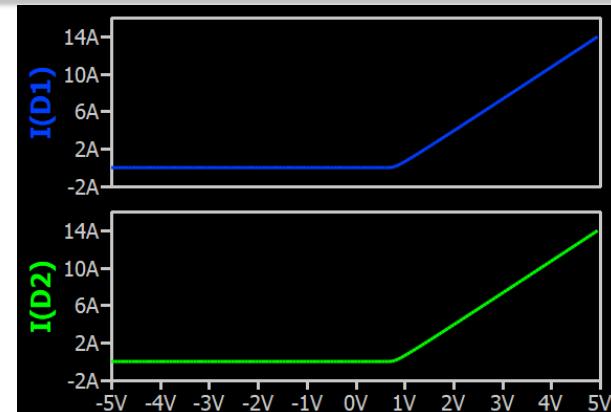


- RS
 - Rs : Series resistance
 - **Default RS=0Ω**
 - This is equivalent to resistor added in series to diode
 - ** V_D in $I_D = I_s (e^{\frac{qV_D}{nkT}} - 1)$ formula only represent diode model voltage drop. If RS is added, diode model anode to cathode voltage is $V_D + I_D R_s$

```
.param Vmin = -5  
.param Vmax = 5
```



```
.dc V1 Vmin Vmax 0.01  
.plot I(D2)  
.plot I(D1)  
.model DRs D Rs=0.29
```

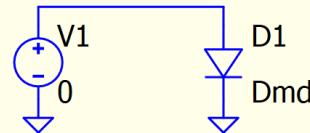


D. Diode (I-V Characteristic - Thermal) : EG

Qspice : Dmodel - EG.qsch

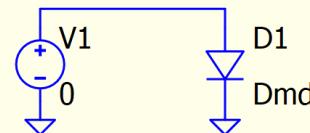
- Eg
 - Activation energy
 - **Default Eg=1.1eV**
 - Eg is temperature dependence factor which modifies effective value of IS (saturation current) and VJ (junction potential)
- Eg has no effect at Tnom
- Example
 - Eg=1.1 or Eg=3.4 with identical I-V characteristic at Tnom=27°C
 - I-V characteristic which different temperature dependence with different Eg

EG: Activation energy / Energy Gap

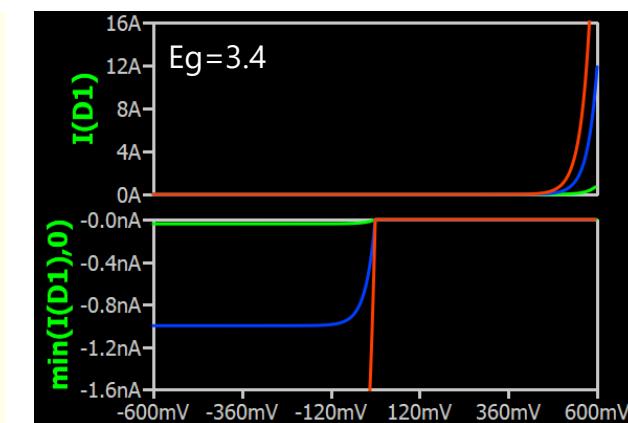
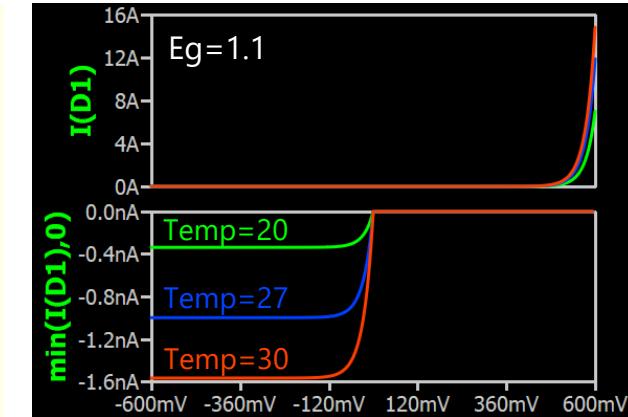


```
.dc V1 -.6 .6 0.001 temp list 20 27 30  
.model Dmdl D Is=1n N=1 Eg=1.1  
.model Dmdl D Is=1n N=1 Eg=3.4  
.plot min(I(D1),0)  
.plot I(D1)
```

EG: Activation energy / Energy Gap



```
.dc V1 -.6 .6 0.001 temp list 20 27 30  
.model Dmdl D Is=1n N=1 Eg=1.1  
.model Dmdl D Is=1n N=1 Eg=3.4  
.plot min(I(D1),0)  
.plot I(D1)
```



D. Diode (I-V Characteristic - Thermal) : EG

- EG : Activation energy / Energy Gap / Bandgap
 - Germanium : 0.67eV
 - Silicon : 1.11eV
 - GaN : 3.4eV
 - Ultra-fast Recovery :
 - Silicon-Schottky : 0.69eV
 - SiC-Schottky : 3.2eV

Forward Voltage (VF) or Threshold @ Specified Forward Current (IF)

** Condition : Recombination current is not used : ISR=0

- Background
 - In datasheet, it generally specify forward threshold by defining as forward voltage drop (VF) at a specified forward current level (IF)
- Diode model include RS
 - $V_{D,ext} = V_D + I_D R_S$ and $I_d = I_s \left(e^{\frac{qV_D}{nkT}} - 1 \right)$
 - where $V_{D,ext}$ is diode voltage drop including series resistance
 - Re-arrange I_d formula : $V_D = n \frac{kT}{q} \ln \left(\frac{I_d}{I_s} + 1 \right)$
 - Therefore,
 - $V_{D,ext} = n \frac{kT}{q} \ln \left(\frac{I_d}{I_s} + 1 \right) + I_D R_S$
 - Substitute $V_{D,ext}$ by VF and I_D by IF and Qspice .model parameters
 - $V_F = N \frac{kT}{q} \ln \left(\frac{IF}{IS} + 1 \right) + IF \times RS$
 - where Qspice default temperature T is 27°C = 300.15K
 - $k=1.380649e-23 \text{ J/K}$ and $q=1.602176487e-19 \text{ C}$ $\rightarrow \frac{kT}{q} = 2.5865e-2$
 - Equation can be simplified with $IF \gg IS$ where $\left(\frac{IF}{IS} + 1 \right) \rightarrow \frac{IF}{IS}$
- Simplified Results in Qspice
 - $VF = 2.5865e-2 * N * \ln(IF/IS) + IF * RS$
 - IS, N and RS are in diode .model
 - IF is specified forward current for diode threshold (e.g. IF=100mA)

Approximation formula to calculate Vd threshold at IF, @27oC
.param VF=2.5865e-2*N*ln(IF/IS)+IF*RS

Model Parameters

.param IS=5n

.param N=2

.param Rs=.58

IF forward Current

.param IF=100m

.dc V1 0 1 0.01

.plot I(D1)

.options listparam

.model 1N4148 D Is=5n Rs=.58 n=2

+Cjo=.87p m=.025 tt=7.9n Nbv=2000

+BV=100 IBV=150n XTI=6

+mfg="ON Semiconductor"

+Vrev=100 Iave=200m

Output Window

TEMP = 27 "CKTTEMP"

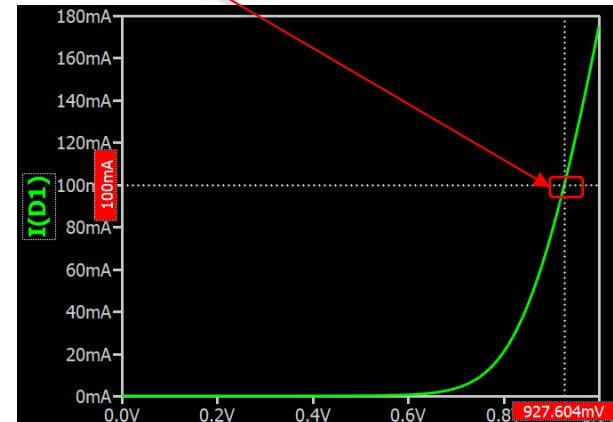
IS = 5N "5N"

N = 2 "2"

RS = 580M ".58"

VF = 927.646M "2.5865E-2*N*LN(IF/IS)+IF*RS"

IF = 100M "100M"



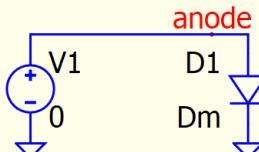
D. Diode (I-V Characteristic : Breakdown) : BV, IBV and NBV

Qspice : Dmodel - BV IBV.qsch / Dmodel - NBV.qsch

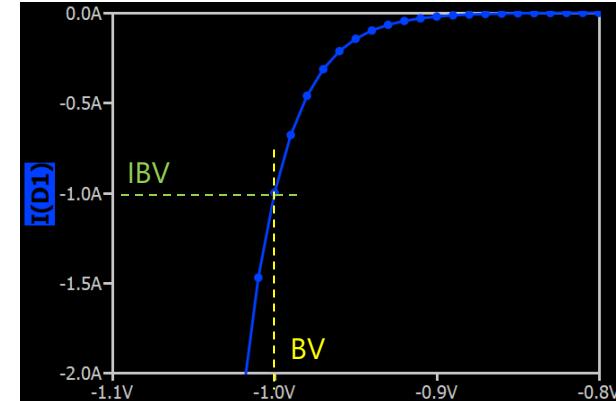
- BV and IBV

- BV : Breakdown Voltage
- IBV : Current at breakdown voltage
- Default BV=Infinite**
- Default IBV=1e-10**
- Breakdown region eqn
 - $I_D = -IBV e^{-\frac{q(BV+V_D)}{kT}}$

```
.param Vmin = -1.2  
.param Vmax = 0
```



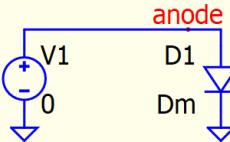
```
.dc V1 Vmin Vmax 0.01  
.plot I(D1)  
.model Dm D BV=1 IBV=1
```



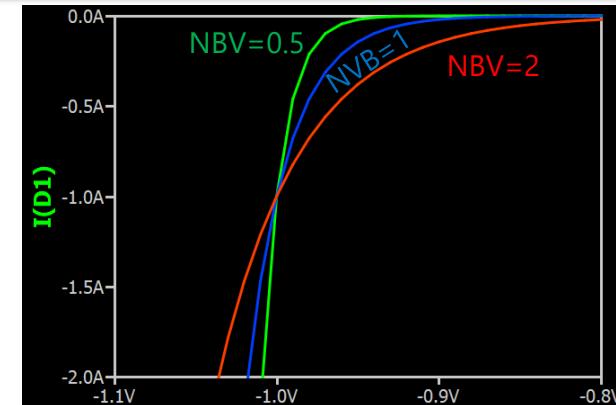
- NBV

- NBV : Reverse breakdown emission coefficient
- Default NBV=1**
- Change the sharpness of breakdown reverse current

```
.param Vmin = -1.2  
.param Vmax = 0
```



```
.dc V1 Vmin Vmax 0.01  
.plot I(D1)  
.model Dm D BV=1 IBV=1 NBV=nbv  
.step param nbv list 0.5 1 2
```



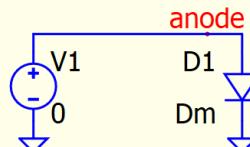
D. Diode (I-V Characteristic : Breakdown) : IBVL and NBVL

Qspice : Dmodel - IBVL.qsch / Dmodel - NBVL.qsch

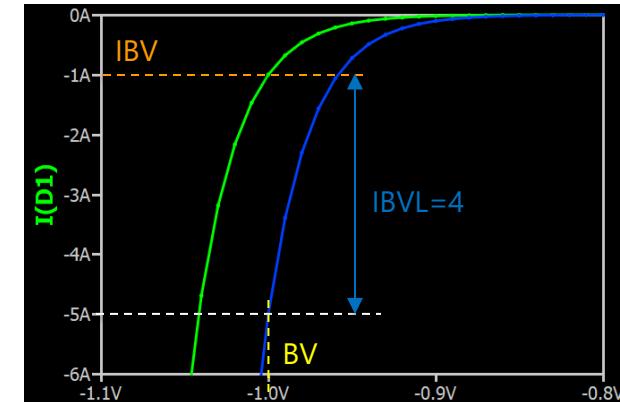
- **IBVL**

- IBVL : low-level reverse breakdown knee current
- **Default IBVL=1**

```
.param Vmin = -1.2  
.param Vmax = -0.6
```



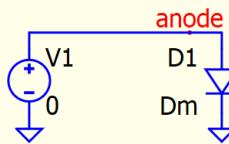
```
.dc V1 Vmin Vmax 0.01  
.plot I(D1)  
.step param ibvl list 0 4  
.model Dm D BV=1 IBV=1 IBVL=ibvl
```



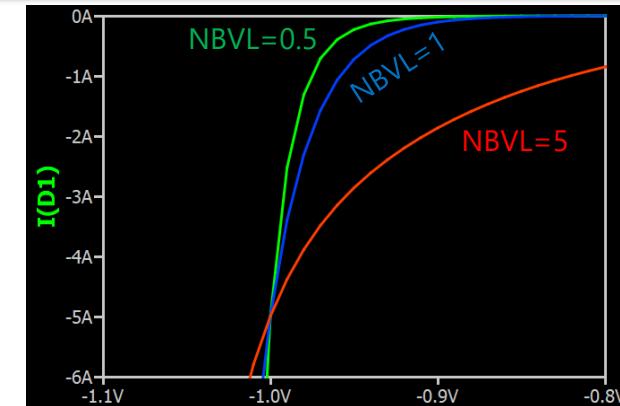
- **NBVL**

- NBVL : Low-level reverse breakdown emission coefficient
- **Default NBVL=1**
- Change the sharpness of breakdown reverse current when IBVL is used

```
.param Vmin = -1.2  
.param Vmax = -0.6
```



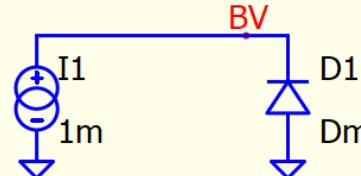
```
.dc V1 Vmin Vmax 0.01  
.plot I(D1)  
.step param nbvl list 0.5 1 5  
.model Dm D BV=1 IBV=1 IBVL=4  
+NBVL=nbvl
```



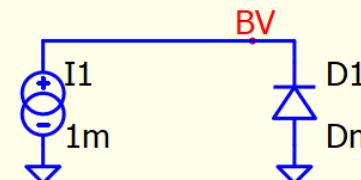
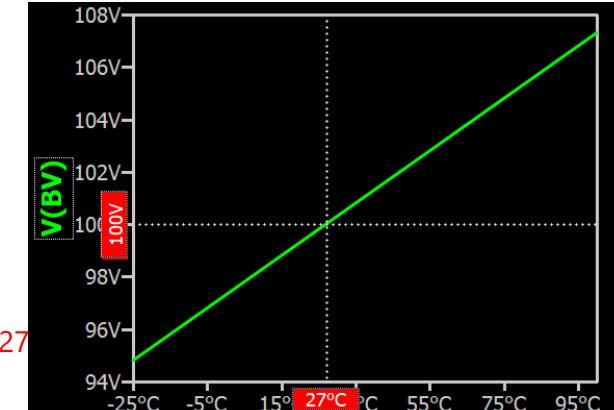
D. Diode (I-V Characteristic : Breakdown) : TBV1, TBV2

Qspice : Dmodel - TBV1.qsch | Dmodel - TBV2.qsch

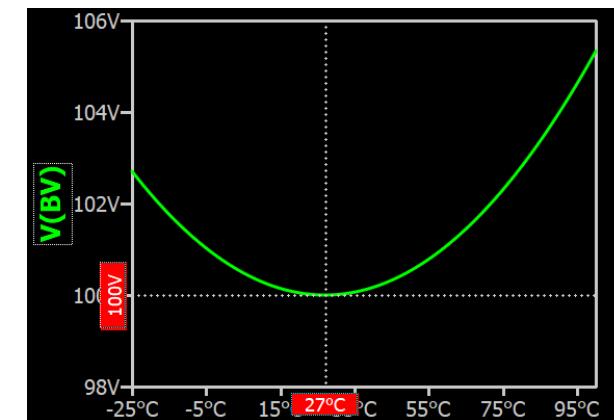
- TBV1, TBV2 (tempco)
 - TBV1 : 1st order BV temperature coefficient
 - TBV2 : 2nd order BV temperature coefficient
 - **Default TBV1=0**
 - **Default TBV2=0**
 - Equation
 - $BV(T) = BV \times (1 + TBV1 \times \Delta T + TBV2 \times \Delta T^2)$
 - where $\Delta T = T - T_{nom}$
 - In default T_{nom} is 27°C



```
.plot V(BV)
.model Dm D BV=100 IBV=1m
+TBV1=1m
Breakdown=
.dc temp -25 100 1 100V@1mA,Tnom=27
.option reltol=1μ
```



```
.plot V(BV)
.model Dm D BV=100 IBV=1m
+TBV2=10μ
.dc temp -25 100 1
.option reltol=1μ
```

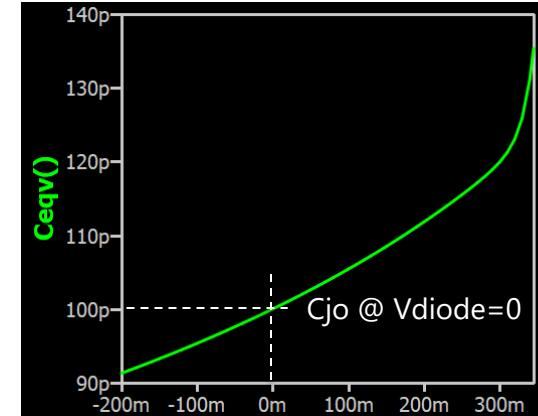


D. Diode (Capacitance) : CJO and M

Qspice : Dmodel - CJO.qsch / Dmodel - M.qsch

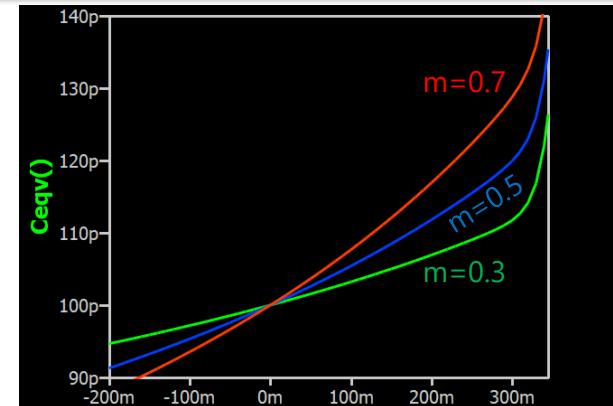
- CJO
 - Cjo : Zero-bias junction capacitance
 - **Default CJO=0F**
- $C_j = \frac{C_{jo}}{\left(1 - \frac{V_D}{\Phi_0}\right)^m}$
 - Φ_0 : Junction potential (VJ)
 - m : Grading coefficient
 - M = 0.33 : linearly graded
 - M = 0.5 : abrupt junction

```
.param Vmin = -0.2
.param Vmax = 0.345
.param f = 1K
.anode
V1 D1 Dm
DC var AC 1 .model Dm D Cjo=100p
.func Zim() imag(V(anode)/I(D1))
.func Ceqv() -1/2/pi/f/Zim()
.ac list f
.step param var Vmin Vmax 0.01
.plot Ceqv()
```



- M
 - m : grading coefficient
 - **Default M=0.5**
- $C_j = \frac{C_{jo}}{\left(1 - \frac{V_D}{\Phi_0}\right)^m}$
 - m : Grading coefficient
 - M = 0.33 : linearly graded
 - M = 0.5 : abrupt junction

```
.param Vmin = -0.2
.param Vmax = 0.345
.param f = 1K
.anode
V1 D1 Dm
DC var AC 1 .model Dm D Cjo=100p m=m
.func Zim() imag(V(anode)/I(D1))
.func Ceqv() -1/2/pi/f/Zim()
.ac list f
.step param var Vmin Vmax 0.01
.plot Ceqv()
.step param m list 0.3 0.5 0.7
```



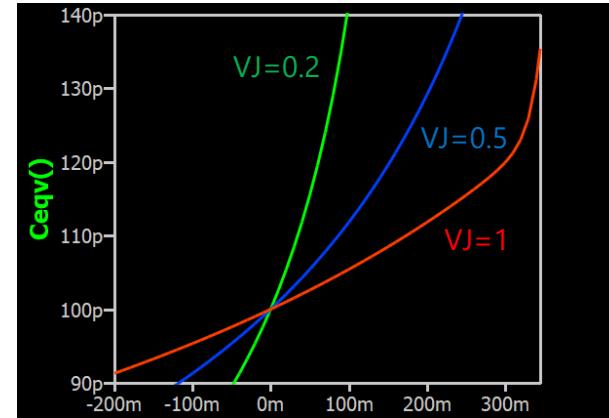
D. Diode (Capacitance) : VJ and FC

Qspice : Dmodel - VJ.qsch / Dmodel - FC.qsch

- VJ
 - V_j : Junction potential
 - **Default $VJ=1V$**
 - $C_j = \frac{C_{jo}}{\left(1 - \frac{V_D}{\Phi_0}\right)^m}$
 - Φ_0 : Junction potential (V_j)
 - may range from 0.2 to 1V

```
.param Vmin = -0.2
.param Vmax = 0.345
.param f = 1K
.anode
V1          .model Dm D Cjo=100p [VJ=vj]
DC var      .func Zim() imag(V(anode)/I(D1))
AC 1        .func Ceqv() -1/2/pi/f/Zim()
D1          Dm
```

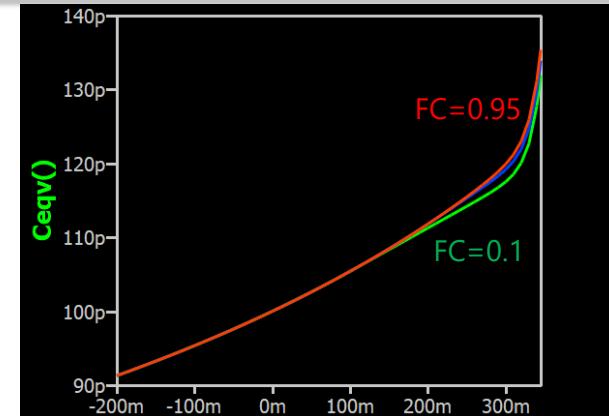
.ac list f
.step param var Vmin Vmax 0.01
.plot Ceqv()
.step param vj list 0.2 0.5 1



- FC
 - F_c : Forward-bias depletion capacitance coefficient
 - **Default $FC=0.5$**
 - A factor between 0 and 0.95 (limit by 0.95 in Qspice), which determines how the junction capacitance is calculated when the junction is forward-biased

```
.param Vmin = -0.2
.param Vmax = 0.345
.param f = 1K
.anode
V1          .model Dm D Cjo=100p [FC=fc]
DC var      .func Zim() imag(V(anode)/I(D1))
AC 1        .func Ceqv() -1/2/pi/f/Zim()
D1          Dm
```

.ac list f
.step param var Vmin Vmax 0.01
.plot Ceqv()
.step param fc list 0.1 0.2 0.95

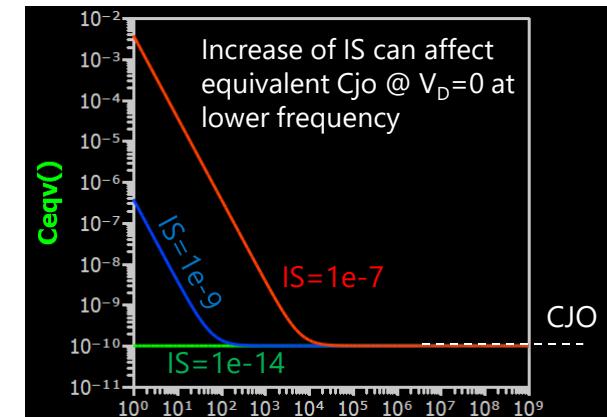


D. Diode (Capacitance) : Parameters can affect Cj in frequency

Qspice : Dmodel - Cj Effect – IS.qsch

- Params can affect Cj in frequency
 - IS : Saturation current

```
.param Vmin = -0.2
.param Vmax = 0.345
.param f = 1K
.anode
V1
DC 0
AC 1
D1
Dm
.func Zim() imag(V(anode)/I(D1))
.func Ceqv() -1/2/pi/f/Zim()
.ac list f
.step dec param f 1 1G 10
.plot Ceqv()
.model Dm D Cjo=100p Is=Is
.step param Is list 1e-14 1e-9 1e-7
```



D. Diode (Reverse Recovery) : TT

Qspice : Dmodel - TT.qsch

- TT : Transit-time
 - TT is not the time it takes a diode to turn off. It's a measure of **the amount of charge that needs to be pulled out to turn the diode off**. Specifically, **the stored charge is TT times the forward current**. For the product to be Coulomb, the dimensions of TT must be time
 - $Q_{store} = I_{fwd} \times tt$, where tt is transit-time in Qspice
 - **Default TT=0s** [equivalent to disable reverse recovery from TT]
 - ** If TT is used, Rs must be non-zero (it can affect forward I-V characteristic if Rs=0 with a finite TT or vice versa)

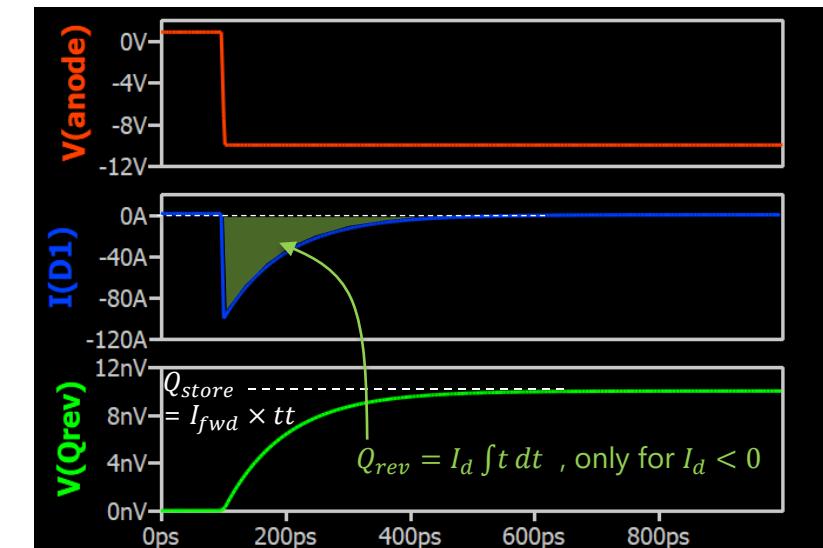
```
.param v fwd=0.834  
.param v rev=-10  
.param td=0.1n  
.param p=1p  
  
anode  
V1 D1 Dm  
pwl 0 v fwd td v fwd td+p v rev
```

```
.tran 10*td  
.option maxstep=p/50  
.plot V(Qrev)  
.plot I(D1)  
.plot V(anode)
```

```
.param tt=10n  
.model Dm D Rs=1e-12 Tt=tt
```

Mike's comment : If TT isn't zero,
Rs can not be zero, because of the
way QSPICE handles dQ/dT soft recovery.

Integral negative diode current : $Q = I \cdot t$
Qrev
B1
V=idt(uramp(I(V1)),0)



D. Diode (Reverse Recovery) : VP

Qspice : Dmodel - VP.qsch

- VP : dQ/dt damping parameter for diffusion charge
 - Default VP=0.01
 - Beware that default value of VP in LTspice is 0 but in Qspice is 0.01, it can lead to different simulation results
 - It affect current amplitude at reverse recovery
 - Increase VP can reduce current amplitude but increase reverse recovery time (soft recovery)

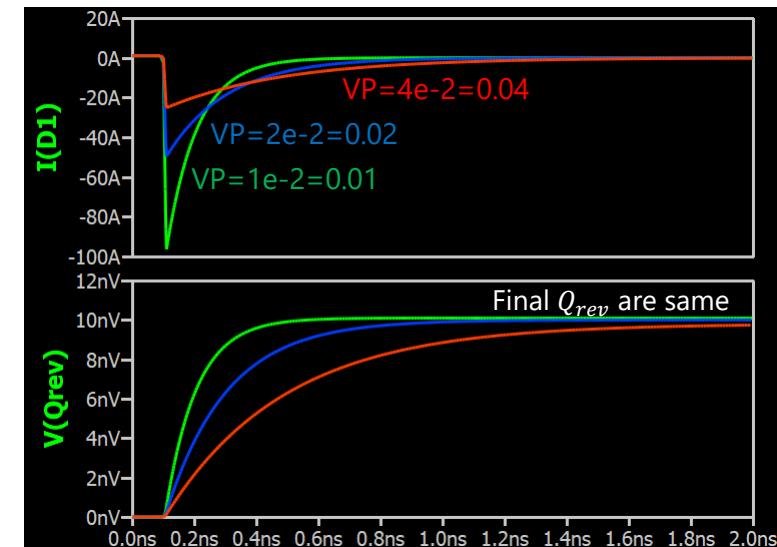
```
.param vfwd=0.834
.param vrev=-10
.param td=0.1n
.param p=1p
.anode
V1
D1
Dm
.pwl 0 vfwd td vfwd td+p vrev
.tran 20*td
.option maxstep=p/50
.plot V(Qrev)
.plot I(D1)
.step param vp list 1e-2 2e-2 4e-2
.param tt=10n
.model Dm D Rs=1e-12 Tt=tt Vp=vp
```

Integral negative diode current : $Q = \int I(V) dt$

Q_{rev}

$B1$

$V = idt(uramp(I(V1)), 0)$

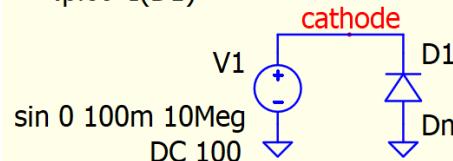


D. Diode (Reverse Recovery) : VP

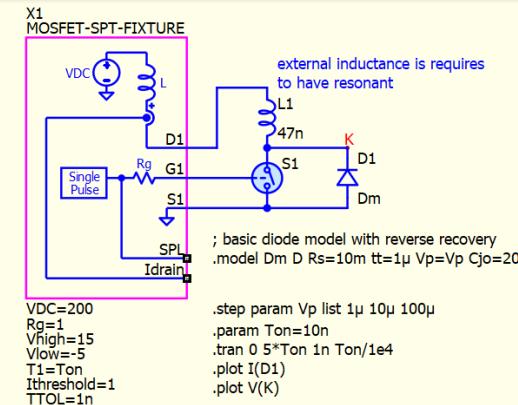
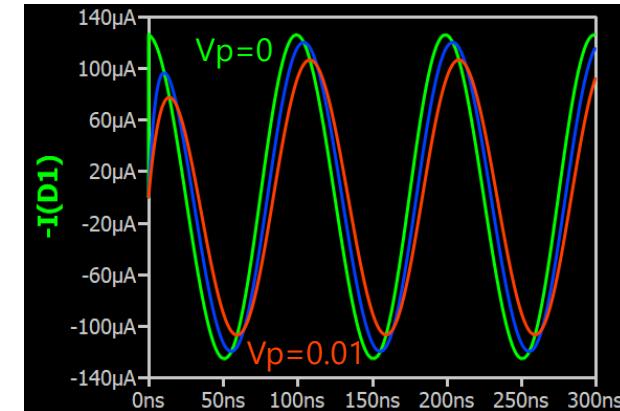
Qspice : Dmodel - VP (in tran).qsch | Dmodel - VP (ringing).qsch

- V_p and Ringing
 - When reviewing V_p, it doesn't show any effect in AC analysis but only in the transient response, and it's also damped with C_{jo}
 - The top schematic suggests that a higher V_p results in higher resistance (which may require more significant damping)
 - The bottom schematic is a single-pulse test to study voltage ringing when the switch is OFF. Resonance occurs between the external inductance L1 and the junction capacitance of D1. With a higher V_p, there is higher damping, and the ringing profile is damped more quickly

```
.step param Vp list 0 0.005 0.01
.tran 3/10Meg
.plot -I(D1)
```



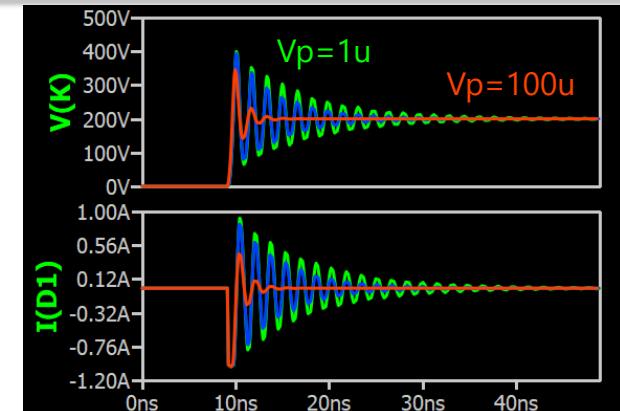
```
.model Dm D Rs=10m tt=1μ Vp=Vp Cjo=20p
```



```
VDC=200
Rg=1
Vhigh=15
Vlow=-5
T1=Ton
Ithreshold=1
TTOL=1n

; basic diode model with reverse recovery
.model Dm D Rs=10m tt=1μ Vp=Vp Cjo=20p

.step param Vp list 1μ 10μ 100μ
.param Ton=10n
.tran 0 5*Ton 1n Ton/1e4
.plot I(D1)
.plot V(K)
```



D. Diode (Reverse Current) : CJO

Qspice : Dmodel - CJO (RR).qsch

- CJO : Zero-bias junction capacitance

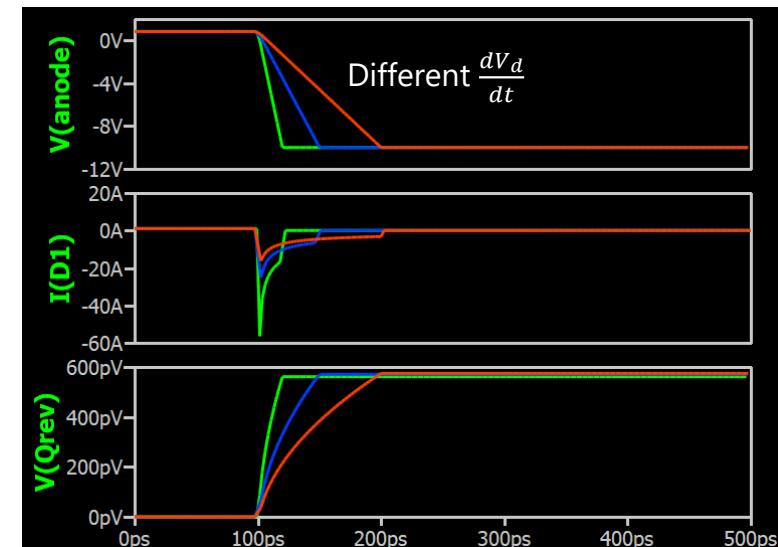
- Junction capacitance leads to reverse current but **NOT** a reverse recovery
 - For example, SiC has negligible reverse recovery and its model may not have TT but with CJO
- Reverse charge depends on junction capacitance and reverse voltage
- $\frac{dV_d}{dt}$ or $\frac{dI_d}{dt}$ are factors that can affect peak reverse current in reverse recovery

```
.param vfwd=0.834  
.param vrev=-10  
.param td=0.1n  
.param p=duratio  
  
anode  
V1 D1 Dm  
  
.tran 5*td  
.option maxstep=p/50  
.plot V(Qrev)  
.plot I(D1)  
.plot V(anode)  
  
.step param duration list 20p 50p 100p  
.model Dm D CJO=100p
```

pwl 0 vfwd td vfwd td+p vrev

Integral negative diode current : $Q = I \cdot t$

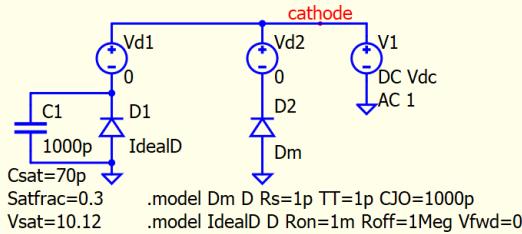
Q_{rev}
B1
 $V = idt(uramp(I(V1)), 0)$



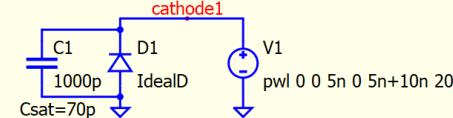
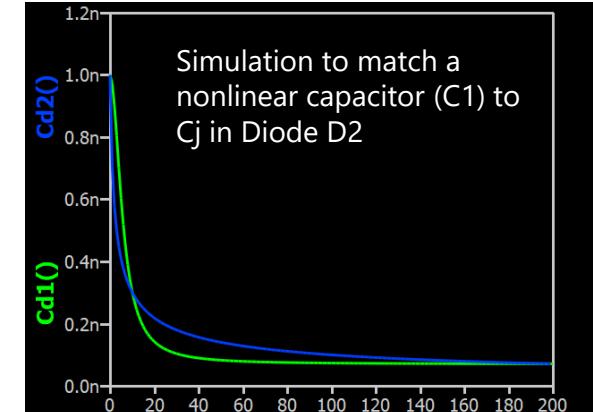
D. Diode (Reverse Current) : CJ

Qspice : Dmodel - CJ in Reverse Voltage.qsch / Dmodel - CJ in Reverse Current.qsch

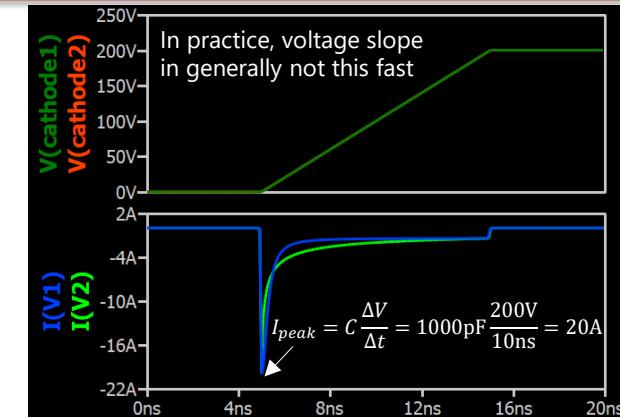
- C_j Explanation
 - Junction capacitance is in parallel to diode and its capacitance is a function of diode reverse voltage
- Simulation #1
 - C_j is nonlinear to diode reverse voltage, to model C_j, it requires use non-linear capacitor with C_{sat}, Satfrac and V_{sat}
- Simulation #2
 - Apply reverse voltage can simulate capacitor charging current, which act like reverse recovery but actually is not
 - Reverse recovery rely on forward current as reverse recovery is to remove this charge to turn off the diode
 - However, C_j is simply a mechanism that current charging a nonlinear capacitor



```
.ac list f      .param f=1Meg      .plot Cd1() Cd2()
.step param Vdc 0 200 1
.func imZD1() imag(V(drain)/I(Vd1))
.func imZD2() imag(V(drain)/I(Vd2))
.func Cd1() -1/2/pi/f/imZD1()
.func Cd2() -1/2/pi/f/imZD2()
```



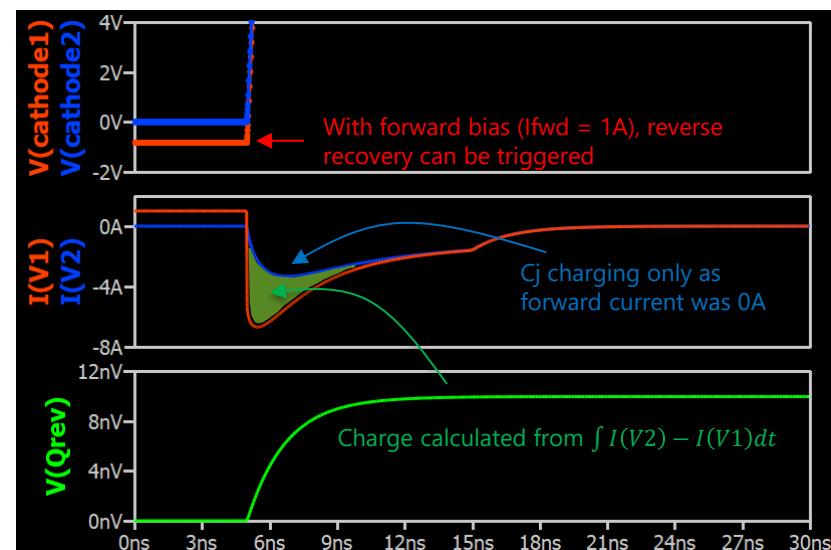
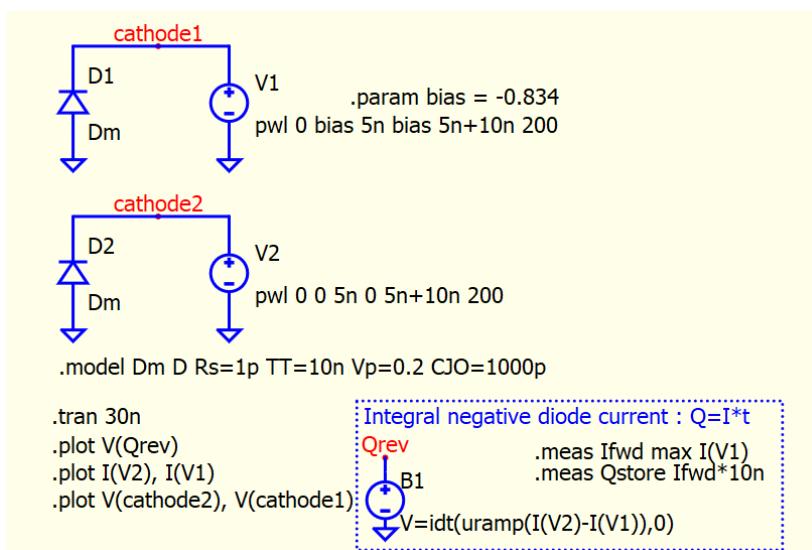
```
.tran 20n
.plot I(V2), I(V1)
.plot V(cathode2), V(cathode1)
.model Dm D Rs=1p TT=1p CJO=1000p
.model IdealD D Ron=1m Roff=1Meg Vfwd=0
```



Explanation of Reverse Recovery and Junction Capacitance Charging

Qspice : Dmodel - Reverse Recovery and Capacitor Charging.qsch

- Explanation of Reverse Recovery and Junction Capacitance Charging Current
 - Reverse Recovery is only preset if forward current is applied
 - If forward current is 0A, reverse current is by charging junction capacitance (C_j), and this is a nonlinear capacitance (capacitance decrease with cathode voltage) and therefore, reverse current reduce over time even ramp of cathode voltage is constant, where $I_d = C_j \frac{dV_d}{dt}$, where C_j is a function of V_d
 - If forward current is preset, reverse recovery is applied (TT and Vp). A extra portion of reverse current is generated and this extra charge equal $I_{fwd} \times TT$



PSPICE Static Model Params (Recombination) : IKF, ISR and NR

** Modeling of Recombination Current

- Semiconductor Device Modeling with SPICE (Section 1.9.1)

- $$I_D = (K_{hli} I_F + K_{gen} I_R) - I_{Breakdown}$$

- $$I_F = I_S \left(e^{\frac{qV_D}{nkT}} - 1 \right)$$

- I_S is IS : Saturation current (default = 1e-14A)
 - n is N : Emission coefficient (default = 1)

- $$K_{hli} = \sqrt{\frac{I_{KF}}{I_{KF} + I_F}} \text{ for } I_{KF} > 0 : \text{ If } I_{KF} \rightarrow \infty, K_{hli} = 1$$

- I_{KF} is IKF : High injection knee current (default = 1e308)

- $$I_R = I_{SR} \left(e^{\frac{qV_D}{n_R kT}} - 1 \right)$$

- I_{SR} is ISR : Recombination current parameter (default = 0A)
 - n_R is NR : ISR emission coefficient (default = 2)

- $$K_{gen} = \sqrt{\left[\left(1 - \frac{V_D}{\Phi_0} \right)^2 + 0.005 \right]^m}$$

- m is M : Grading coefficient (default = 0.5)
 - Φ_0 is VJ : Junction potential (default = 1V)

- $$I_{Breakdown} = IBV e^{-\frac{q(BV+V_D)}{kT}}$$

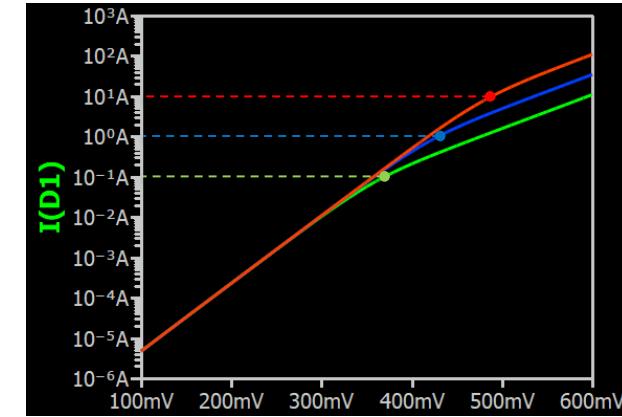
- IBV is IBV : Current at breakdown voltage (default = 1e-10A)
 - BV is BV : Reverse breakdown voltage (default = Infinite V)
 - This is equation to describe breakdown region. Reverse leakage is described in $(K_{hli} I_F + K_{gen} I_R)$

D. Diode (Recombination) : IKF

Dmodel - IKF.qsch

- IKF
 - IKF : High injection knee current
 - **Default IKF=1e308**
 - $I_D = (K_{hli}I_F + K_{gen}I_R)$
 - $I_D = K_{hli}I_F$
 - $K_{hli} = \sqrt{\frac{I_{KF}}{I_{KF}+I_F}}$ for $I_{KF} > 0$
 - The modification effect may only be easily observed with ID in log scale

```
.param Vmin = 0.1
.param Vmax = 0.6
.anode
V1 0 D1 Dm
.DC V1 {Vmin} {Vmax} 0.01
.PLOT I(D1)
.step param ikf list 0.1 1 10
.model Dm D Is=100n N=1 IKF={ikf}
```



D. Diode (Recombination) : ISR

Qspice : Dmodel - ISR.qsch

- ISR
 - Isr : Recombination current parameter
 - **Default ISR=0A**
 - $I_D = (K_{hli}I_F + K_{gen}I_R)$
 - In following examples, I_F is forced to be negligible as compare to $K_{gen}I_R$ in calculating I_D

$$\bullet I_D = K_{gen} I_{SR} \left(e^{\frac{qV_D}{n_R kT}} - 1 \right)$$

$$\bullet K_{gen} = \sqrt{\left[\left(1 - \frac{V_D}{\Phi_0} \right)^2 + 0.005 \right]^m}$$

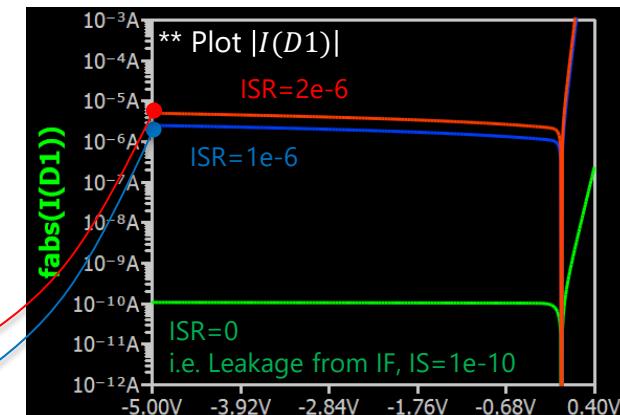
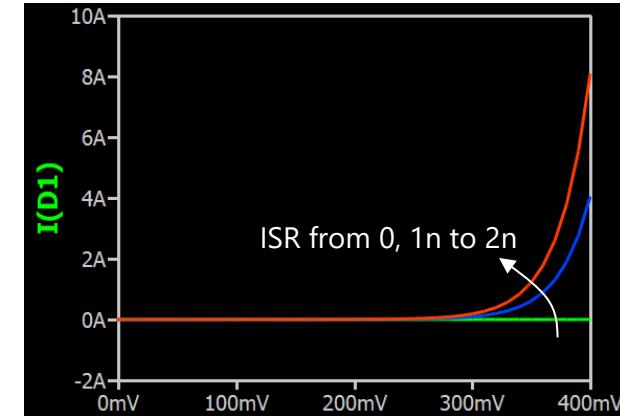
• n_R is NR : default = 2V

• m is M : default = 0.5

• Φ_0 is VJ : default = 1V

```
.param Vmin = -5
.param Vmax = 0.4
.anode
V1 0 D1
Dm
.dc V1 Vmin Vmax 0.01
.plot I(D1)
.step param isr list 0 1e-6 2e-6
.model Dm D Is=1e-10 N=2 ISR=isr NR=1
Is and N are set to equivalent with negligible If as
compare to Ir to demonstrate Ir effect
```

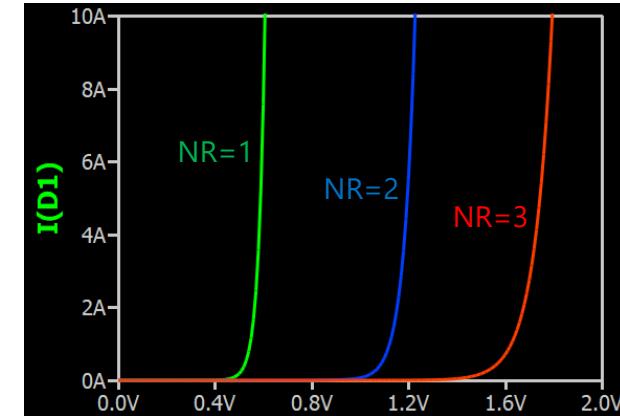
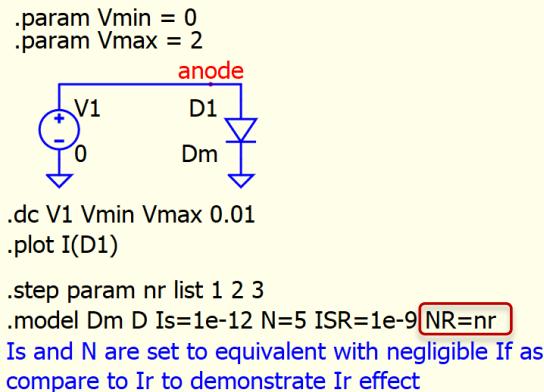
- Remark
 - As IS is set to 1e-12, in reverse and ISR=0, leakage current can still be observed through equation I_F
 - @ $V_D = -5V$
 - Therefore
 - ID for ISR=2e-6, $VD=-5$ is $4.8990\mu A$
 - ID for ISR=1e-6, $VD=-5$ is $2.4495\mu A$



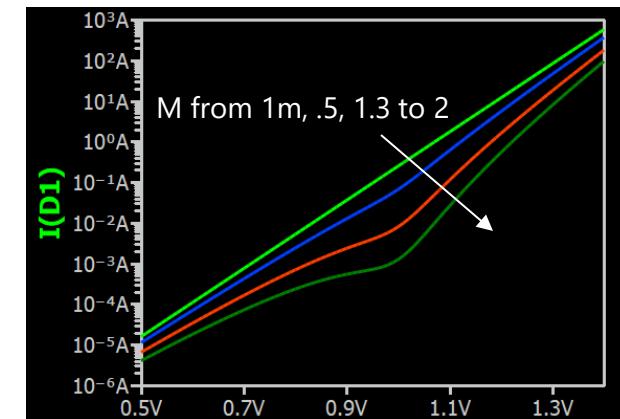
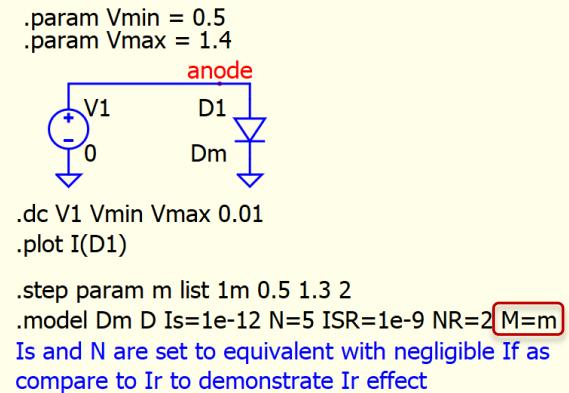
D. Diode (Recombination) : NR and M

Qspice : Dmodel - NR.qsch / Dmodel - M (Recombination).qsch

- NR
 - Nr : ISR emission coefficient
 - **Default NR=2**
 - $I_D = K_{gen} I_{SR} \left(e^{\frac{qV_D}{n_R kT}} - 1 \right)$



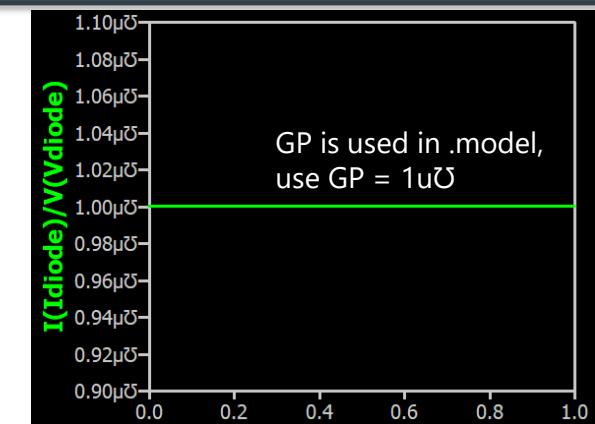
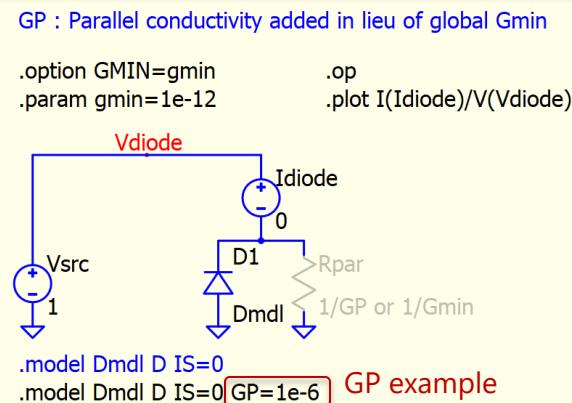
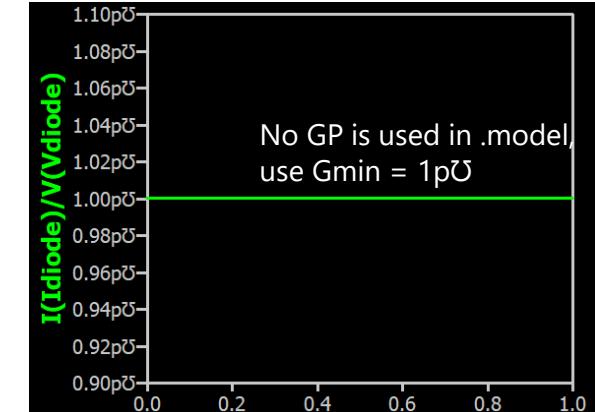
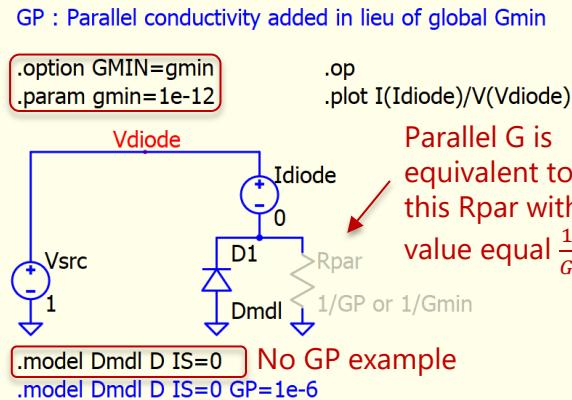
- M
 - M : Grading coefficient
 - **Default M=0.5**
 - $K_{gen} = \sqrt{\left[\left(1 - \frac{V_D}{\Phi_0} \right)^2 + 0.005 \right]^M}$
 - The modification effect may only be easily observed with ID in log scale



D. Diode (Parallel Conductance) : GP, TGP1 and TGP2

Qspice : Dmodel - GP.qsch

- GP
 - Parallel conductivity added in lieu of global Gmin
 - **Default GP=0**
 - If GP is not used, Gmin is used
 - Otherwise, GP is used
 - Gmin is minimum conductance (G) in parallel to PN junction (Default gmin in .option is 1e-12)
 - This parallel conductance is tested in this section by placing a diode in reverse with IS=0, thereby eliminating any contribution from the reverse current of the diode. This allows us to solely observe the effect of the parallel conductance.



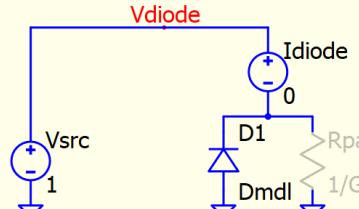
D. Diode (Parallel Conductance) : GP, TGP1 and TGP2

Qspice : Dmodel - GP - TGP1 TGP2.qsch

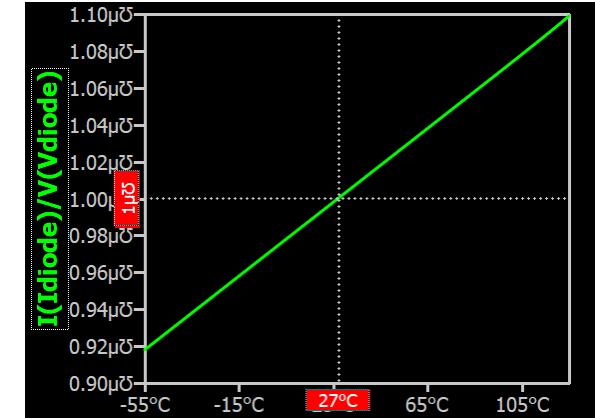
- TGP1, TGP2 (tempco)
 - TGP1 : 1st order GP temperature coefficient
 - TGP2 : 2nd order GP temperature coefficient
 - **Default TGP1=0**
 - **Default TGP2=0**
 - Equation
 - $G(T) = GP \times (1 + TGP1 \times \Delta T + TGP2 \times \Delta T^2)$
 - where $\Delta T = T - T_{nom}$
 - In default T_{nom} is 27°C

GP : Parallel conductivity added in lieu of global Gmin

```
.option GMIN=gmin          .dc temp -55 125 1
.param gmin=1e-12           .plot I(Idiode)/V(Vdiode)
```

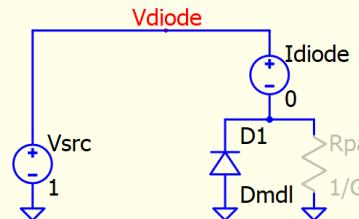


```
.model Dmdl D GP=1e-6 TGP1=1e-3 TGP2=0
```

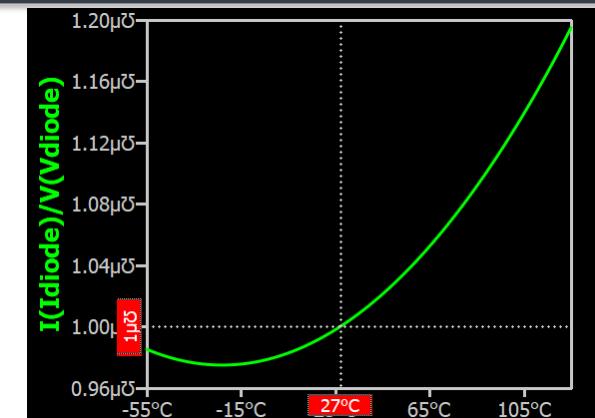


GP : Parallel conductivity added in lieu of global Gmin

```
.option GMIN=gmin          .dc temp -55 125 1
.param gmin=1e-12           .plot I(Idiode)/V(Vdiode)
```



```
.model Dmdl D GP=1e-6 TGP1=1e-3 TGP2=1e-5
```



Textbook Diode Modeling

Qspice : Dmodel (Textbook).qsch

- Textbook Diode Modeling

- $I_S = 1e-12$ to minimize leakage current when diode is OFF

- $R_S = \frac{\Delta V_D}{\Delta I_D}$ in the region that diode is ON

- $N = \frac{V_F - I_F \times R_S}{0.025865 \times \ln\left(\frac{I_F}{I_S}\right)}$: IF is current at VF

- Assume textbook requirement is

- Threshold = 0.7V and $\frac{\Delta I_D}{\Delta V_D} = 100A/V$

- By $V_F = \frac{I_F}{\frac{\Delta I_D}{\Delta V_D}} + \text{Threshold}$

- We use a higher IF to determine the curve,

- @ $I_F=20A \rightarrow V_F = \frac{20A}{100A/V} + 0.7V = 0.9V$

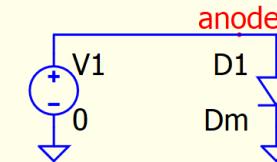
- Therefore

- $I_S = 1e-12$

- $R_S = 1/100 = 0.01$

- $N = \frac{0.9 - 20 \times 0.01}{0.025865 \times \ln\left(\frac{20}{1e-12}\right)} = 0.883$

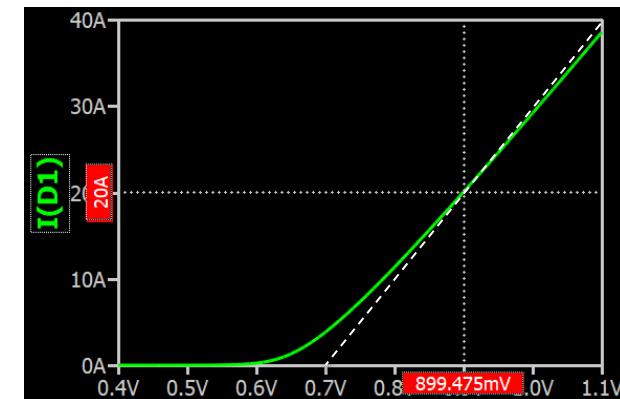
```
.param Vmin = 0.4  
.param Vmax = 1.2
```



```
.dc V1 Vmin Vmax 0.01
```

```
.plot I(D1)
```

```
.model Dm D Is=1e-12 N=0.883 RS=0.01
```



D. Diode

Behavioral Diode Model Parameters

Behavioral Diode Model Parameters in Qspice HELP

QSPICE includes a simplified, behavioral, diode. To use those device equations, specify a non-zero RON.

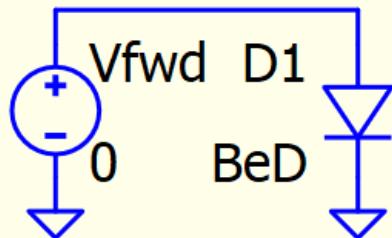
Behavioral Diode Model Parameters

Name	Description	Units	Default
RON	Forward on resistance	Ω	0.0 ³
ROFF	Off resistance	Ω	1./GMIN
RZEN	Breakdown resistance(aka RREV)	Ω	RON
VFWD	Forward voltage drop	V	0.0
VREV	Reverse voltage drop	V	Infinite
EPSILON	Width of quadratic region splining off and on regions	V	0.0
REVEPSILON	Width of quadratic region splining breakdown and on regions	V	0.0
CJO	Shunt capacitance	F	0.0

- ^{3]} The value of zero means don't use these equations, but the conventional SPICE semiconductor diode equations.

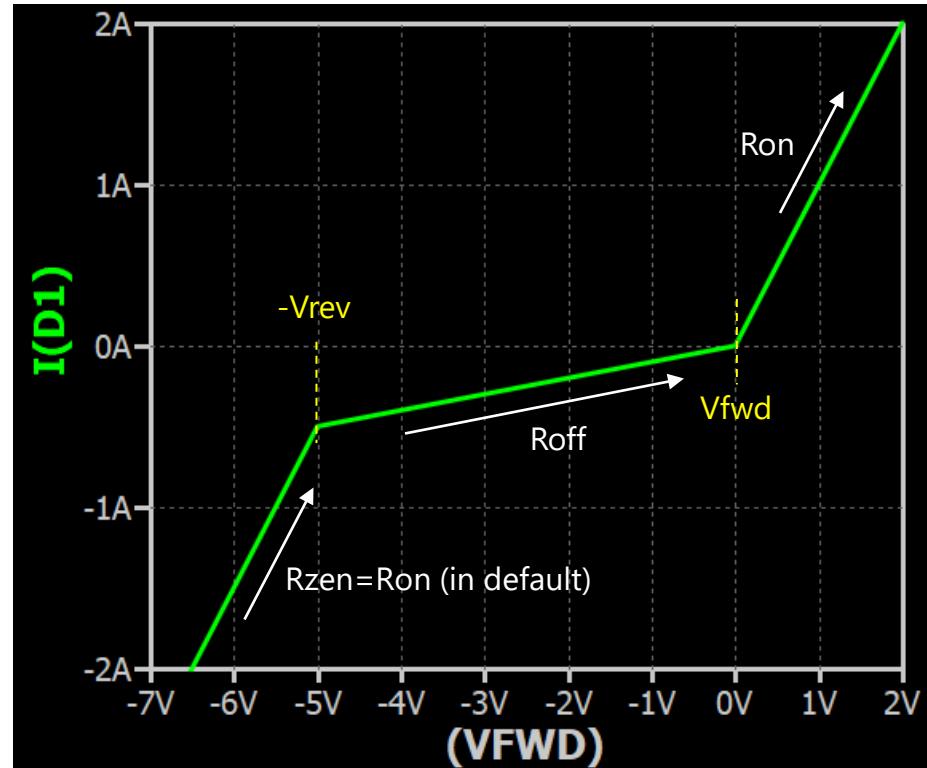
Behavioral Diode Model Params : Ron, Roff, Vfwd, Vrev

Qspice : Behavioral D Model - Ron Roff Vfwd Vrev.qsch



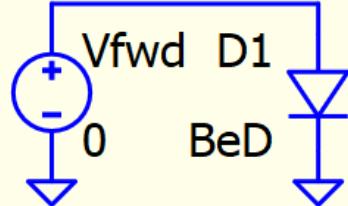
```
.model BeD D Ron=1 Roff=10  
+ Vfwd=0 Vrev=5  
+ Epsilon=0 Revepsilon=0  
.dc Vfwd -7 2 0.1  
.plot I(D1)
```

** Important Note : Refer to Qspice help, Ron must include and set to non-zero to activate behavioral diode model



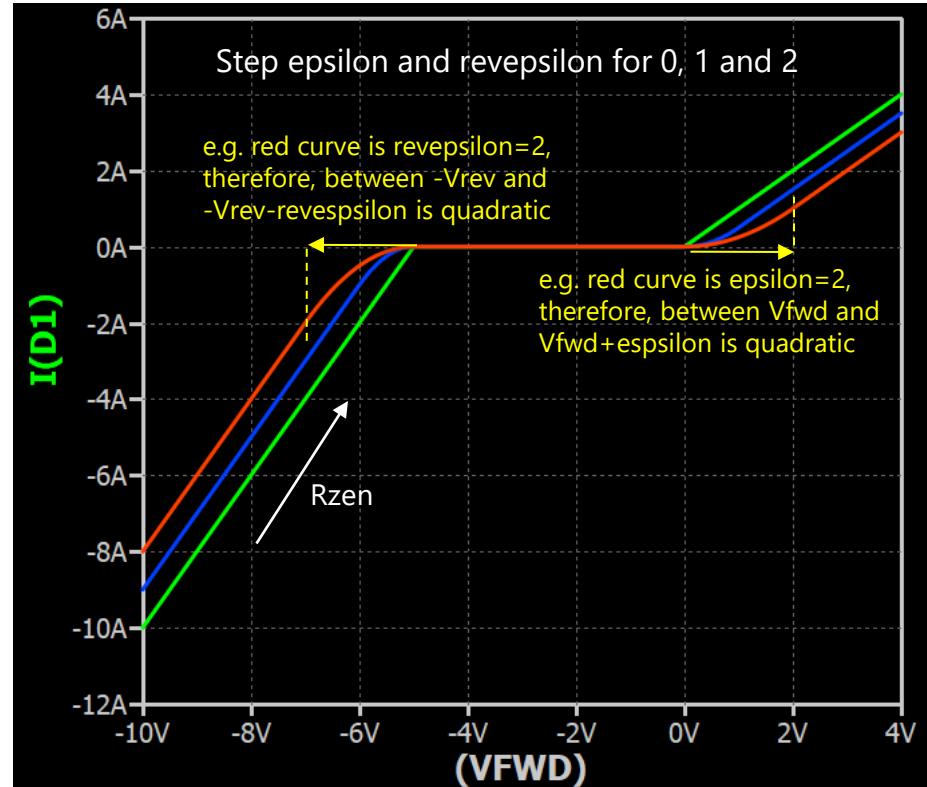
Behavioral Diode Model Params : Rzen, Epsilon, Revepsilon

Qspice : Behavioral D Model - Epsilon Revepsilon Rzen.qsch



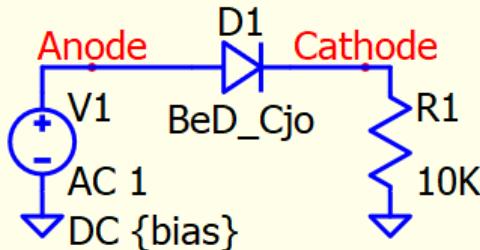
```
.model BeD D Ron=1 Roff=10Meg  
+ Vfwd=0 Vrev=5  
+ Epsilon={val} Revepsilon={val}  
+ Rzen=0.5  
  
.step param val 0 2 1  
.dc Vfwd -10 4 0.1  
.plot I(D1)
```

* Roff change to 10Meg in this study



Behavioral Diode Model Params : Cjo

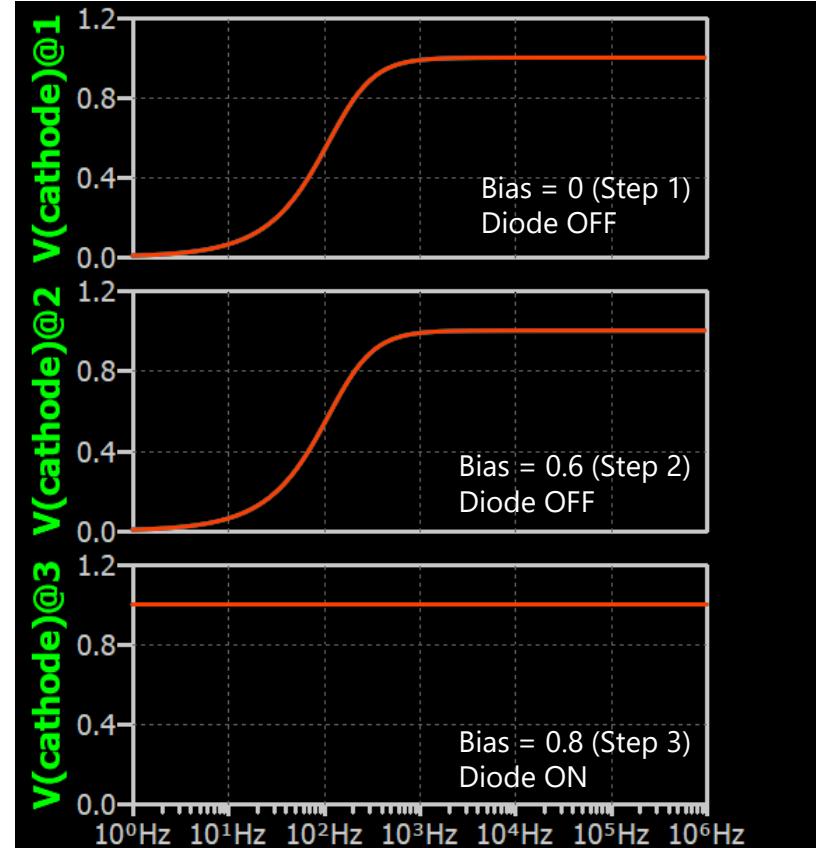
Qspice : Behavioral D Model - Cjo.qsch



```
.param bias = 0  
.model BeD_Cjo D Ron=1m Roff=1Meg  
+ Vfwd=0.7 Cjo=100n
```

Cjo is equivalent to a capacitor parallel the behavioral diode

```
.ac dec 100 1 1Meg  
.step param bias list 0 0.6 0.8  
  
.plot V(cathode)@3  
.plot V(cathode)@2  
.plot V(cathode)@1
```



D. Diode

Ideal Diode .model

Ideal Diode Model

Qspice : Special - IdealD (.dc).qsch / Special - IdealID (.tran).qsch

- Ideal Diode Characteristic
 - No breakdown current
 - No reverse leakage
 - No capacitance effect
 - No reverse recovery
- Ideal Model with Diode .model
 - .model Idlm D N=0.01 RS=10 μ ; Vf@1A=0
 - .model Idlm D N=0.36 RS=10 μ ; Vf@1A=0.3
 - .model Idlm D N=0.85 RS=10 μ ; Vf@1A=0.7
 - ** RS is added to prevent gmin stepping without series resistance
- Ideal Model with Behavioral Diode .model
 - .model IdealD D Vfwd=0 Ron=1m Roff=100Meg ; Vf,th=0
 - .model IdealD D Vfwd=0.3 Ron=1m Roff=100Meg ; Vf,th=0.3
 - .model IdealD D Vfwd=0.7 Ron=1m Roff=100Meg ; Vf,th=0.7

E. F. G. H. Dependent Sources

E. F. G. H. Dependent Sources

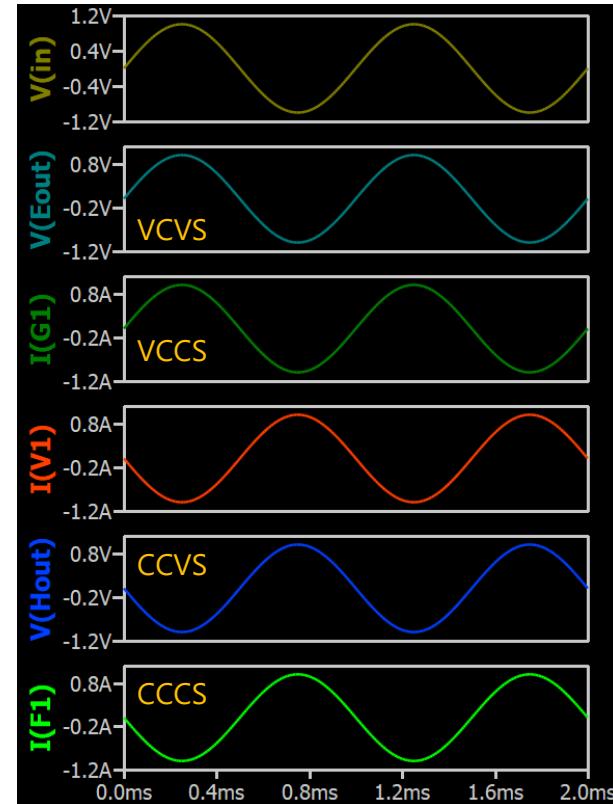
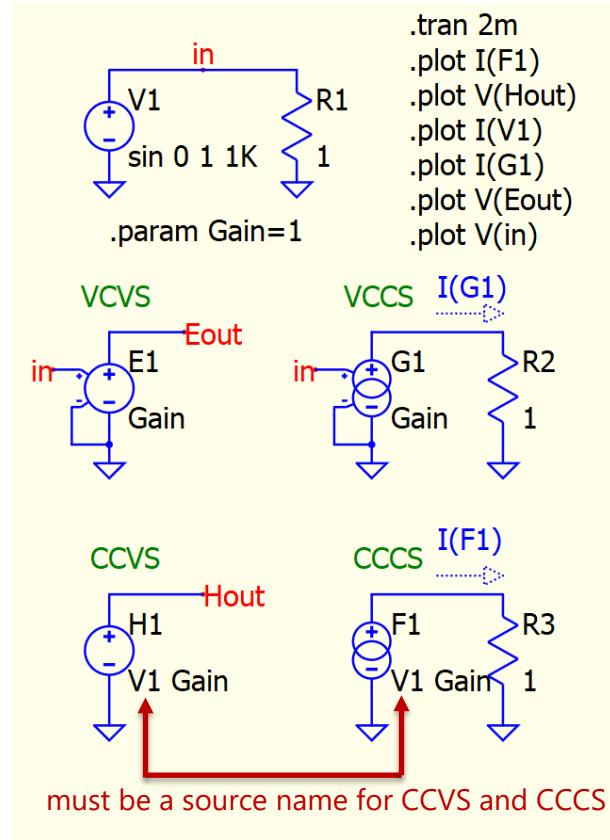
Qspice : Dependent Source - Basic.qsch

- Dependent Sources

- [E] VCVS : Voltage Dependent Voltage Source
- [G] VCCS : Voltage Dependent Current Source
- [H] CCVS : Current Dependent Voltage Source
- [F] CCCS : Current Dependent Current Source

- ** Current Dependent Src

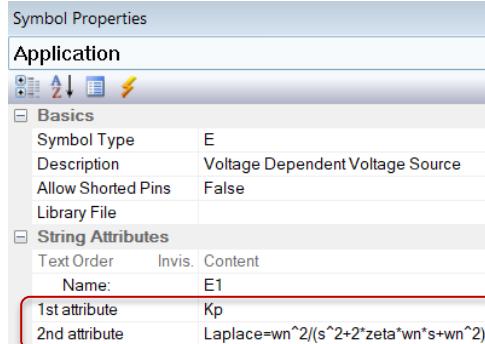
- For current dependent source, the current sense has to come from voltage source instead of current source
- A 0V voltage source can be used for current sensing
- (Explanation from Mike Engelhardt) Sensing the current in a current is complicated because it might have VSAT or VSAT2, meaning it's not a current source



E-, G-source : Laplace

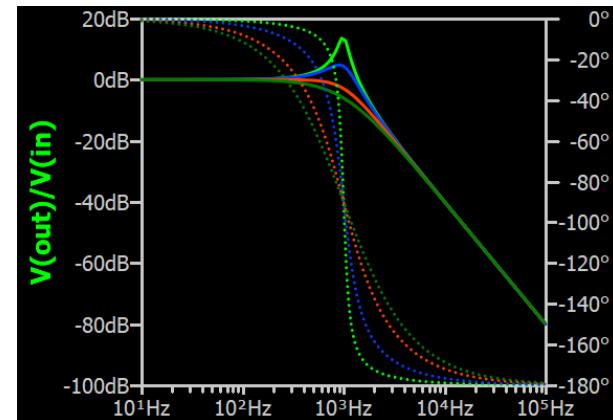
Qspice : E-source - Laplace.qsch | G-source - Laplace.qsch

- Laplace
 - E- and G-source support Laplace function in format
Ennn N+ N- NC+ NC- <gain> LAPLACE=<s-domain expression>
 - Adding a new attribute **Laplace=<expression>**



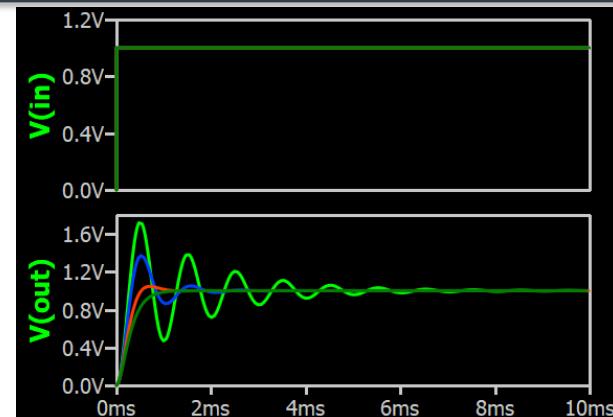
.param f=1K .param wn=2*pi*f
.param zeta=.3 .param Kp=1
.param Kp=1
in out
V1 E1 Kp Overall Gain
AC 1 Laplace=wn^2/(s^2+2*zeta*wn*s+wn^2)
PWL 0 0 1n 1
.step param zeta list 0.1 0.3 0.7 1
.ac dec 100 f/100 f*100 .tran 10/f
.plot ac V(out)/V(in) .plot tran V(out)
.plot tran V(in)

Frequency Domain



.param f=1K .param wn=2*pi*f
.param zeta=.3 .param Kp=1
.param Kp=1
in out
V1 E1 Kp Overall Gain
AC 1 Laplace=wn^2/(s^2+2*zeta*wn*s+wn^2)
PWL 0 0 1n 1
.step param zeta list 0.1 0.3 0.7 1
.ac dec 100 f/100 f*100 .tran 10/f
.plot ac V(out)/V(in) .plot tran V(out)
.plot tran V(in)

Time Domain

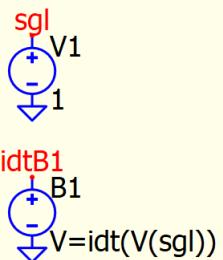


E-Source Alternative Syntax – Equivalent to Behavioral Voltage Src

Qspice : E-source - alternative syntax.qsch

- E-source

- Syntax: Exxx n+ n- value={<expression>}
 - Must use curly braces
- This is an alternative syntax of the behavioral voltage source (B-source)
- This syntax only available in using text, no Qspice symbol support this format, for Qspice to backward compatible to legacy spice syntax



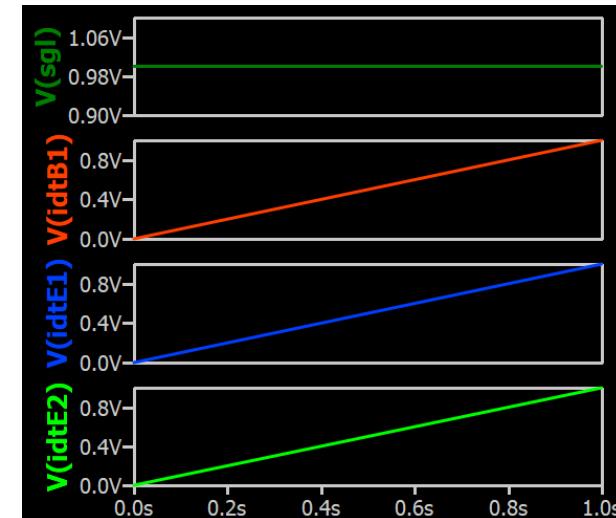
```
.tran 1  
.plot V(idtE2)  
.plot V(idtE1)  
.plot V(idtB1)  
V=idt(V(sgl))  
.plot V(sgl)
```

Syntax: Exxx n+ n- value={<expression>}
alternative syntax of the behavioral source with E-source
formula must have curly bracket {}

E1 idtE1 0 Value {idt(V(sgl))} ← Can accept without "="

E2 idtE2 0 Value={idt(V(sgl))}

```
V1 sgl 0 1  
B1 idtB1 0 V=idt(V(sgl))  
E1 idtE1 0 Value {idt(V(sgl))}  
E2 idtE2 0 Value={idt(V(sgl))}  
.plot V(idtE2)  
.plot V(idtE1)  
.plot V(idtB1)  
.plot V(sgl)  
.tran 1  
.end
```

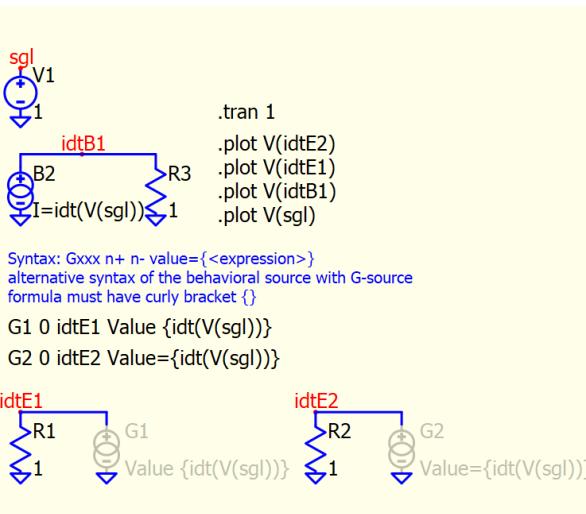


G-Source Alternative Syntax – Equivalent to Behavioral Current Src

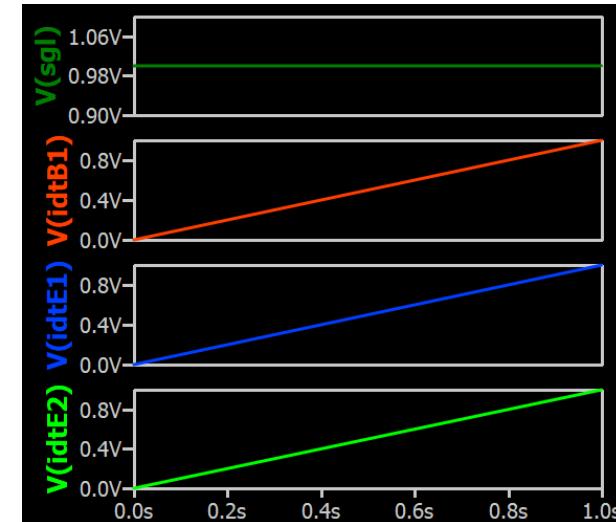
Qspice : G-source - alternative syntax.qsch

- G-source

- Syntax: $G_{xxx} n+ n- \text{ value}=\{\text{<expression>}\}$
 - Must use curly braces
- This is an alternative syntax of the behavioral current source (B-source)
- This syntax only available in using text, no Qspice symbol support this format, for Qspice to backward compatible to legacy spice syntax



```
V1 sgl 0 1
R1 idtE1 0 1
R2 idtE2 0 1
R3 idtB1 0 1
B2 0 idtB1 I=idt(V(sgl))
G1 0 idtE1 Value {idt(V(sgl))}
G2 0 idtE2 Value={idt(V(sgl))}
.plot V(idtE2)
.plot V(idtE1)
.plot V(idtB1)
.plot V(sgl)
.tran 1
.end
```



Polynomial Functions POLY [Legacy Syntax]

Qspice : E-source - POLY(1)-1.qsch | E-source - POLY(1)-2.qsch

- Polynomial Functions

- POLY(NDIM)

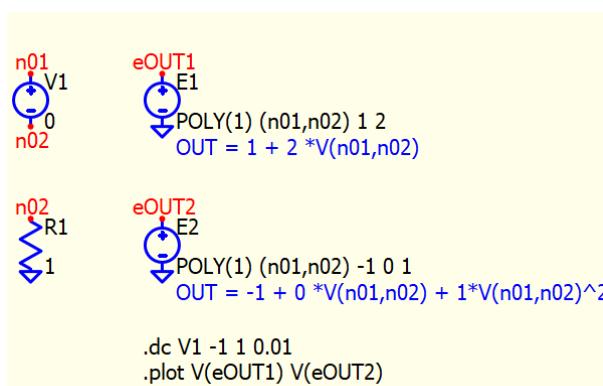
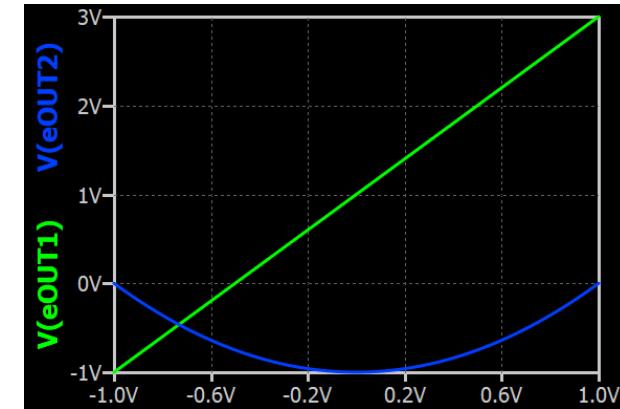
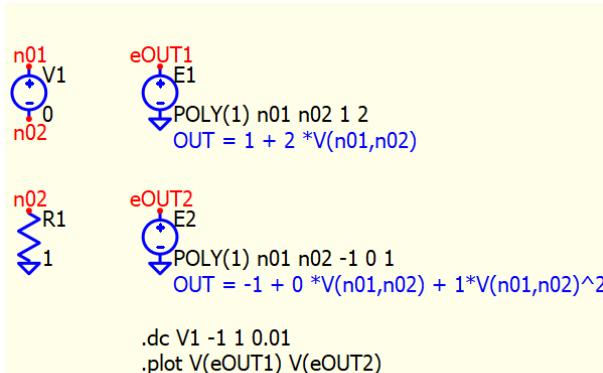
- Polynomial equations in n-dimension
 - Detail can be found in Hspice user guide, for n-dimensional definition
 - Poly supports E-, F-, G-, H- element statement

- One-dimensional eqn

- POLY(1) n01 n02 1 2 or
 - POLY(1) (n01,n02) 1 2
 - POLY(1) is one-dimension
 - n01 n02 is voltage difference between $V(n01,n02)$
 - 1 2 are polynomial coefficient : $1 + 2 * V(n01,n02)$

- E-source netlist syntax

- POLY() requires legacy e-source syntax
 - E1 eOUT1 0 POLY(1) n01 n02 1 2



Polynomial Functions POLY [Legacy Syntax]

Qspice : E-source - POLY(2).qsch

- Polynomial Function

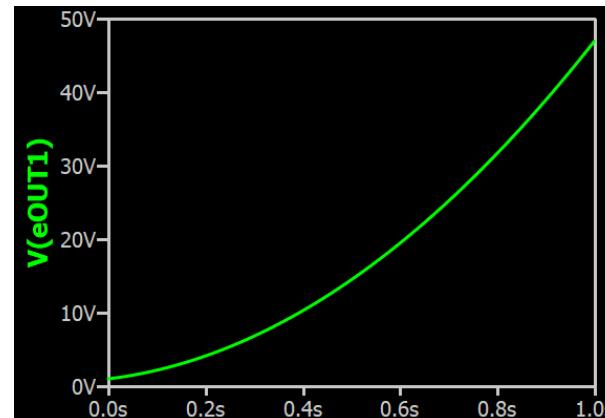
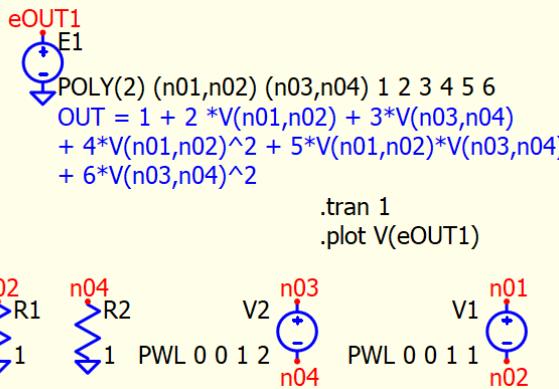
- POLY(NDIM) can be n-dimensional

- Two-dimension example

- POLY(2) (n01,n02) (n03,n04) 1 2 3 4 5 6
 - where (n01,n02) and (n03,n04) are two differentiate voltages
 - 1 2 3 4 5 6 are polynomial coefficient, and with formula as

$$1 + 2 *V(n01,n02) + 3*V(n03,n04)+ 4*V(n01,n02)^2 + 5*V(n01,n02)*V(n03,n04)+ 6*V(n03,n04)^2$$

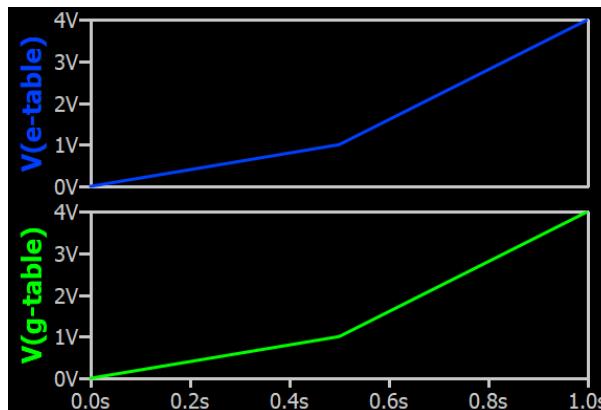
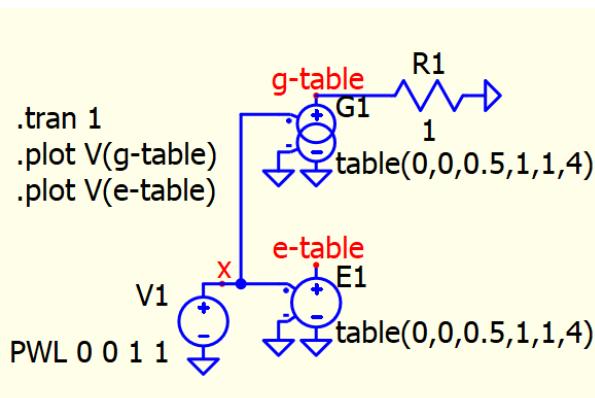
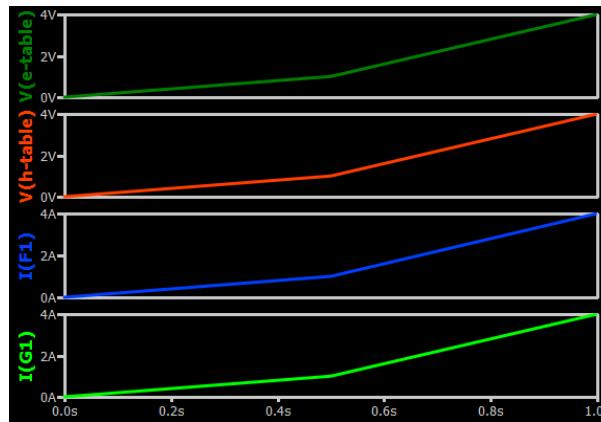
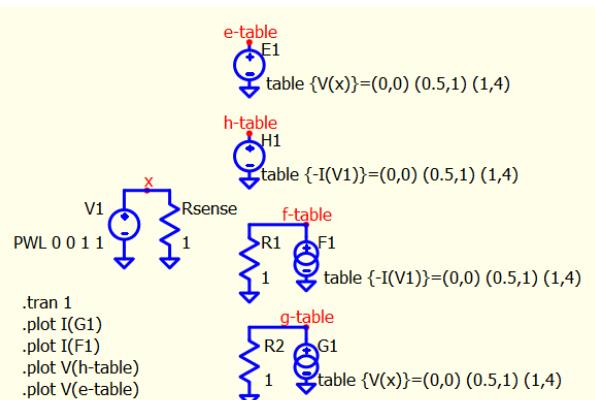
zero order term, first order term, second order term, ...



Archaic PSpice Table for E-, H-, F-, G- and LTspice Table for E-source

Qspice : Dependent Source - Archaic PSpice Table.qsch

- Archaic PSpice Table
 - This is an archaic syntax from Pspice in format of $\text{table } \{\text{node/current}\}=(x_0,y_0) (x_1,y_1) (x_2,y_2)\dots$
- LTspice E-source Table
 - Even number elements table is supported in LTspice



I. Current Source

I. Current Source : Instance Parameters

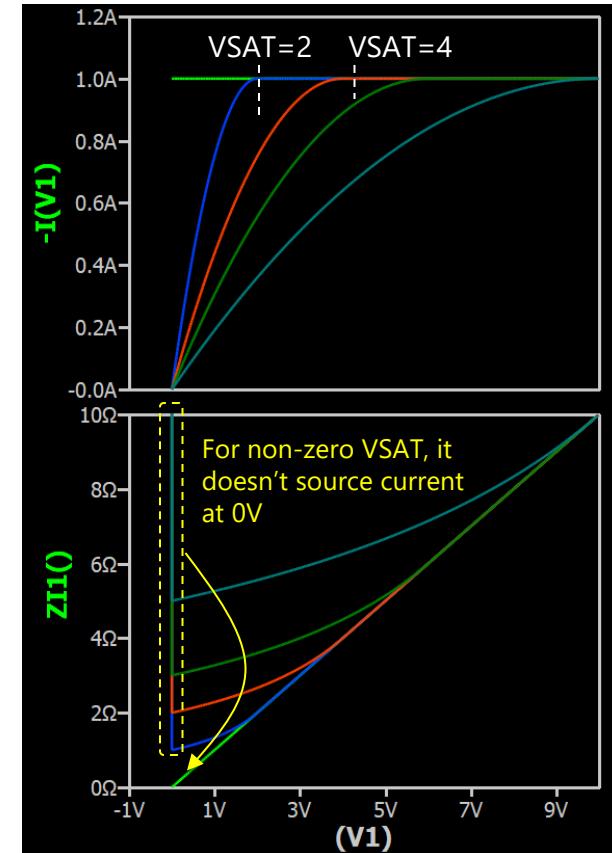
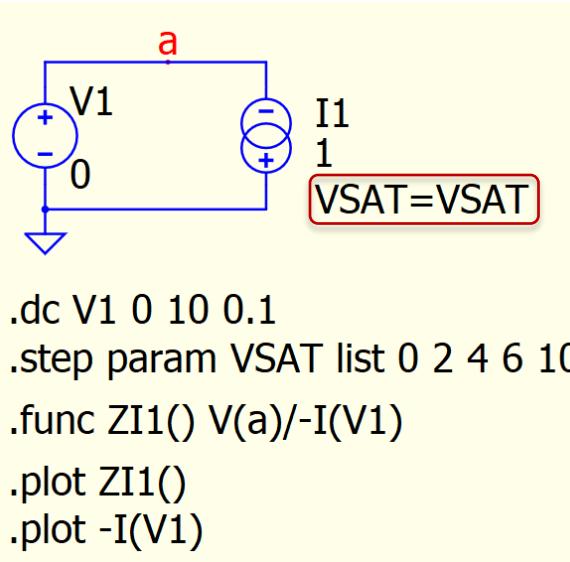
Current Source Instance Parameters

Name	Description	Units	Default
AC	AC magnitude, optionally followed by phase angle	A, °	0.
ACMAG	AC magnitude	A	0.
ACPHASE	AC phase	°	0.
DC	DC value of source	A	0.
EXP	Exponential source description		
LOG	Interpolate between PWL and CHIRP points		(not set)
PULSE	Pulse description		
PWL	Piecewise linear description		
SFFM	Single frequency FM description		
SINE	Sinusoidal source description(aka SIN)		
TIMECTRL	Time step control, one of NONE, LIMITS ¹ , BREAKS ² , or BOTH	String	LIMITS
VSAT ³	Don't source power(example)	V	0.
VSAT2	Don't source power and don't conduct in reverse	V	0.
XTRAP	Extrapolate beyond PWL and CHIRP points		(not set)

I. Instance Params : VSAT

Qspice : I Source - VSAT.qsch

- VSAT
 - VSAT : Does **NOT** source power
 - **Default VSAT=0V**
 - Characteristic
 - Only source current when $V(-ve,+ve) > VSAT$
 - Current source doesn't source current at $V(-ve,+ve)=0$
 - Equivalent to Open
 - Not to apply negative voltage between $V(-ve,+ve)$ as it will have current flow out from -ve
 - Usage
 - Useful for modeling an active load or a bias current in a macromodel



I. Instance Params : VSAT - Formula

- VSAT – formula

- Assume

- v is voltage across source
 - I_{limit} is value of source
 - V_{sat} is value of VSAT

- If $v < V_{SAT}$

- $I = \frac{-I_{limit}}{(V_{sat})^2} (v - V_{sat})^2 + I_{limit}$

- Slope of I at $v=0$: $\frac{2 I_{limit}}{V_{sat}}$

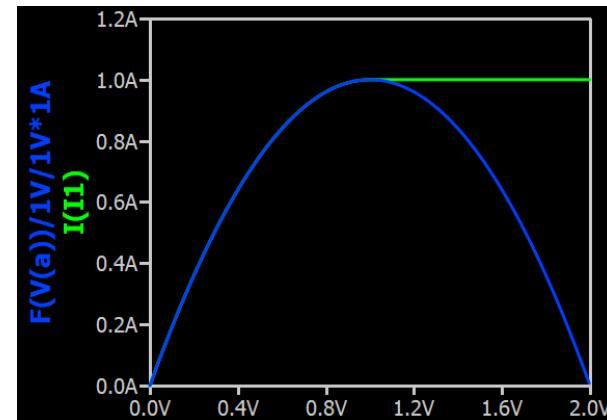
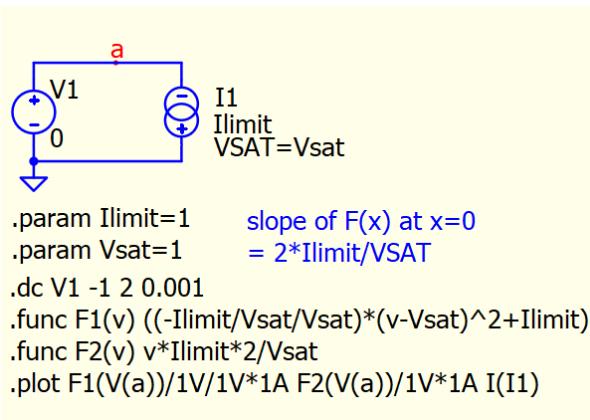
- Therefore, $R = \frac{V_{sat}}{2 I_{limit}}$

- If $v \geq V_{SAT}$

- $I = I_{limit}$

- If $v < 0$

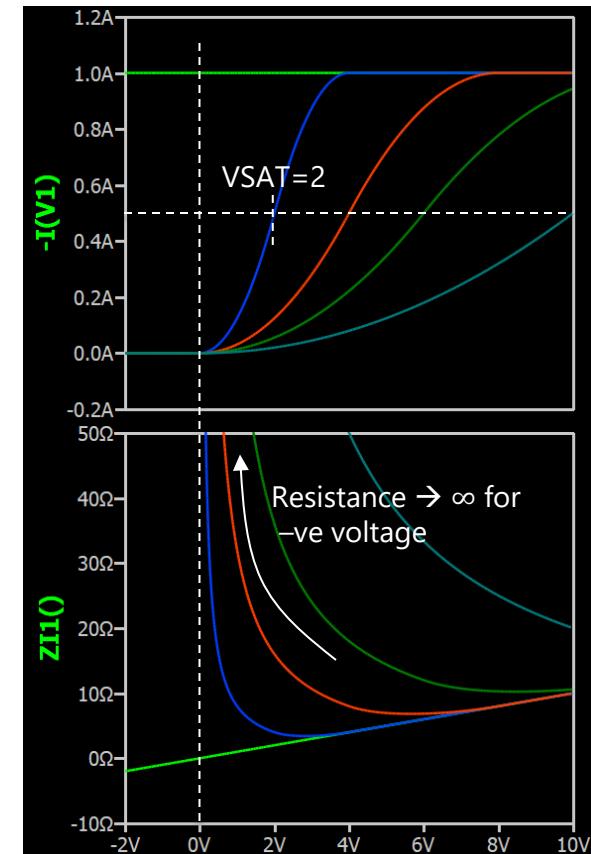
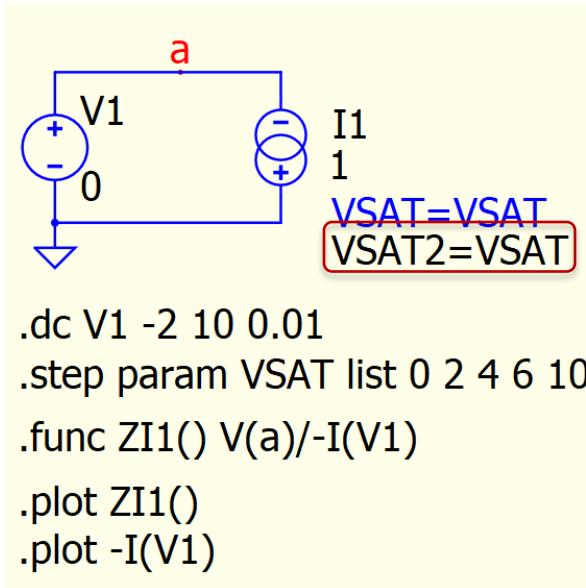
- $\frac{I}{v} = \frac{2 I_{limit}}{V_{sat}} \rightarrow I = \frac{2 v I_{limit}}{V_{sat}}$



I. Instance Params : VSAT2

Qspice : I Source - VSAT2.qsch

- VSAT2
 - VSAT2 : Does **NOT** source power and **NOT** conduct in reverse
 - **Default VSAT2=0V**
 - If VSAT2 is used, VSAT will be overridden



I. Instance Params : VSAT and VSAT2 – Comparison

Qspice : I Source - VSAT VSAT2.qsch

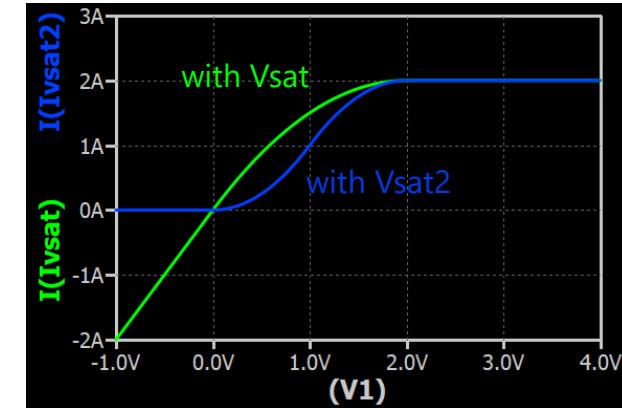
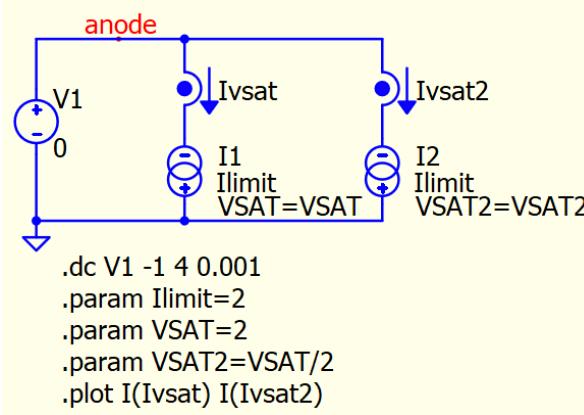
- VSAT and VSAT2

- VSAT

- @VSAT, Current limit to current source value
 - Conducts in reverse direction
 - Current and voltage is in tanh-shaped curve

- VSAT2

- @VSAT2*2, Current limit to current source value
 - Doesn't conduct in reverse direction
 - Current and voltage is in S-shaped curve



I. Instance Params : VSAT and VSAT2 – Ilimit Diode (Usage)

Qspice : I Source - VSAT VSAT2 - Ilimit Diode.qsch

- VSAT and VSAT2 – Usage example

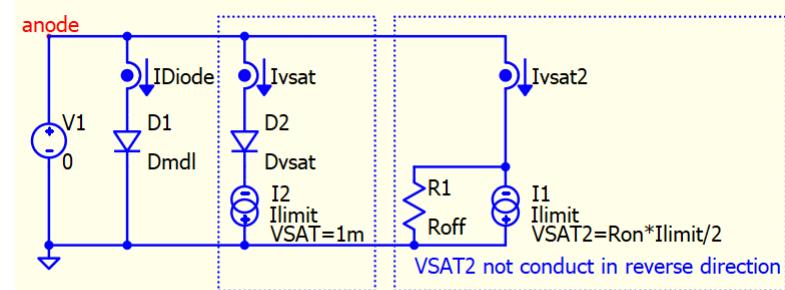
- Current limit diode
- A diode with current limit can be emulated with
 - Current source with VSAT and a Diode
 - Current source with VSAT2 and a parallel R

- Current source with VSAT

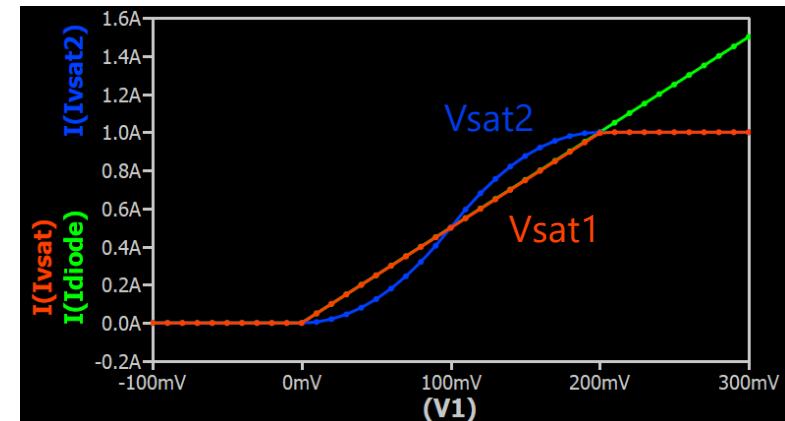
- VSAT don't source power but can conduct in reverse, therefore, it requires a diode to block reverse conduction

- Current source with VSAT2

- VSAT2 don't source power and not conduct in reverse. However, as its ideally not draw current in reverse direction, a parallel resistor is added to emulate Roff value



```
.param Ron=200m  
.param Roff=10Meg  
.param Ilimit=1  
.model Dvsat D Ron=Ron Roff=Roff Vfwd=0  
.model Dmdl D Ron=Ron Roff=Roff Vfwd=0  
.dc V1 -0.1 0.5 0.01  
.plot I(Idiode) I(Ivsat2) I(Ivsat)
```



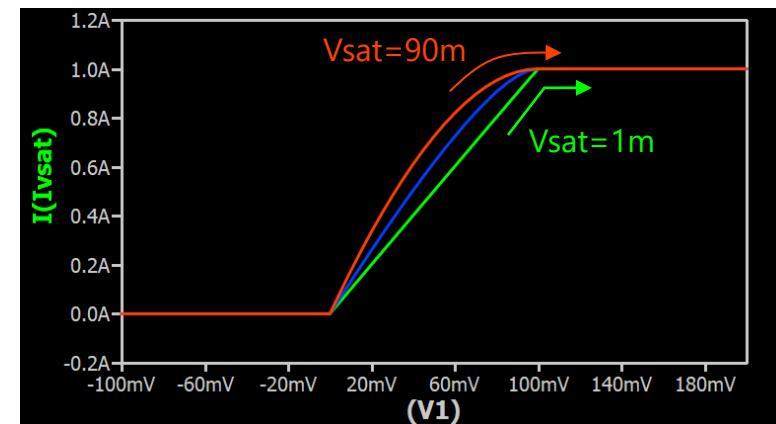
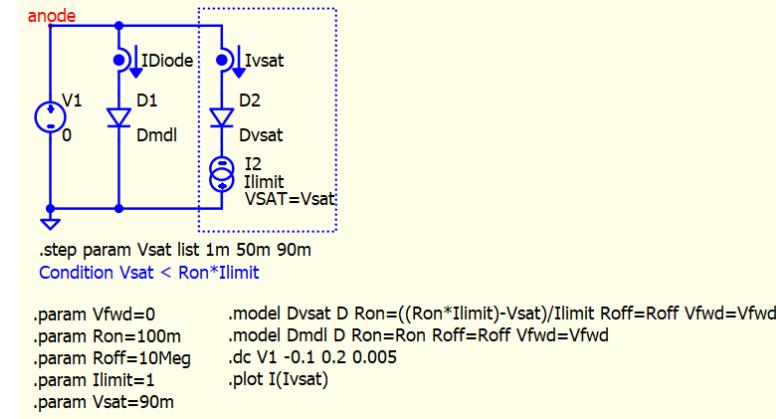
I. Instance Params : VSAT – Ilimit Diode with Smooth approach

Qspice : I Source - VSAT - Ilimit Diode.qsch

- VSAT – Usage

- The current limit diode with a smooth approach uses VSAT to control how the current limit is approached
- In this setup, the aim is to maintain the current limit point. Therefore, if VSAT is specified, the Ron of the diode should be

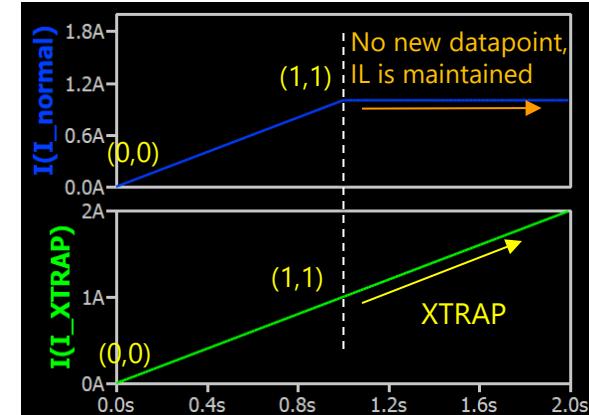
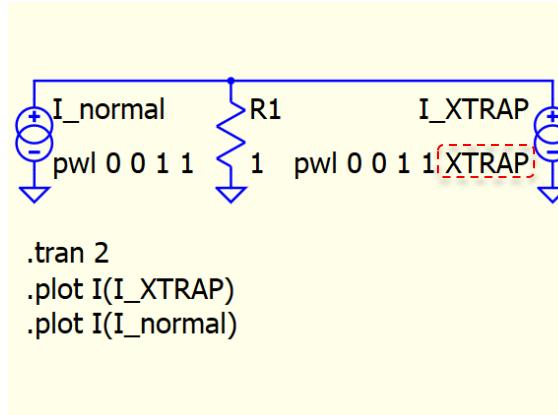
$$\frac{R_{on} \times I_{limit} - V_{sat}}{I_{limit}}$$



I. Instance Params : XTRAP

Qspice : I Source - XTRAP.qsch

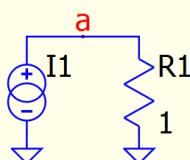
- XTRAP
 - Xtrap : Extrapolate beyond PWL and CHIRP points
 - **Default XTRAP is not set**
 - Without XTRAP, the current stays constant with the value specified with the last time point



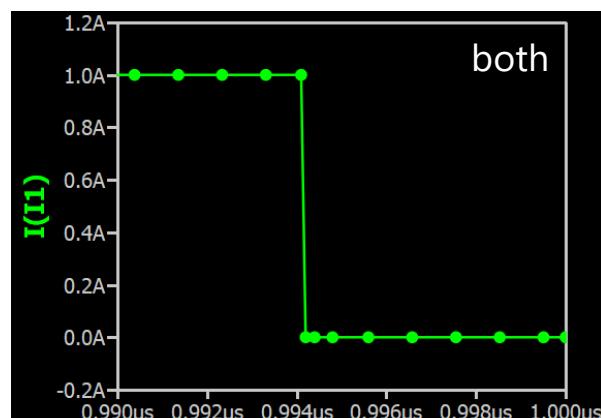
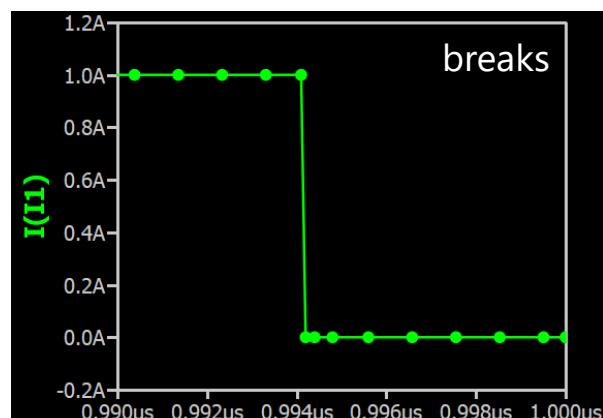
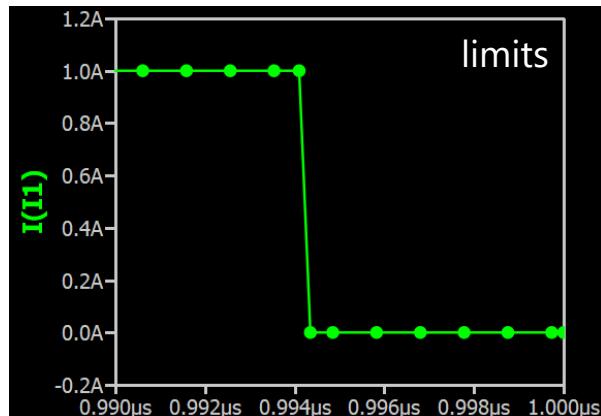
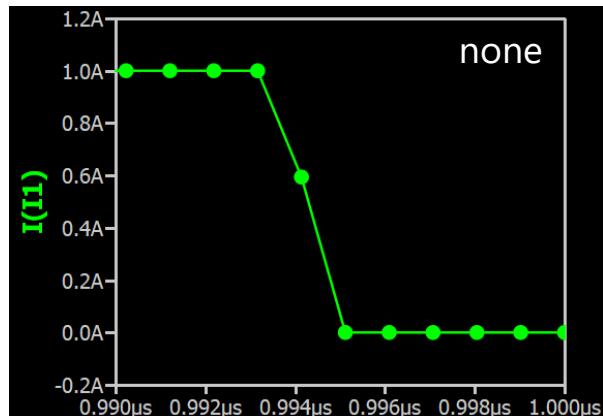
I. Instance Params : Timectrl

Qspice : I Source - TIMECTRL.qsch

- TIMECTRL
 - Timectrl : Time step control
 - None
 - Limits
 - Breaks
 - Both
 - **Default Timectrl=Limits**



```
pulse 0 1 4n 0 0 10n 20n  
Timectrl=none  
Timectrl=limits  
Timectrl=breaks  
Timectrl=both  
.tran 1000n  
.plot I(I1) I(I2)
```



J. JFET Transistor

J. JFET Model Parameters

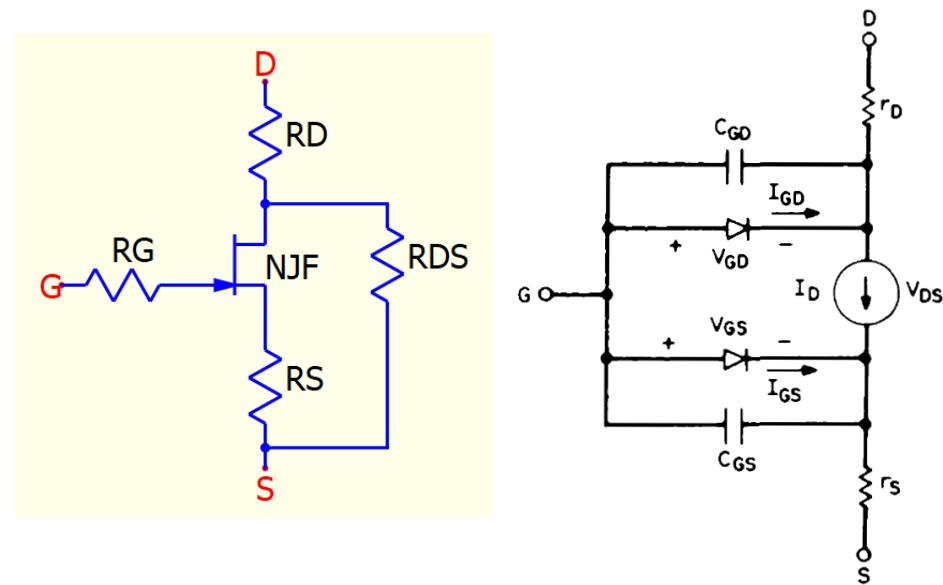
JFET Model Parameters

Name	Description	Units	Default
AF	Flicker noise exponent		1.0
ALPHA	Impact ionization coefficient	V ⁻¹	0.0
B	Doping tail parameter		0.0
BETA	Transconductance parameter	A/V ²	0.0001
BETATCE	Exponential transconductance parameter temperature coefficient	%/°C	0.0
CGD	Zero-bias G-D junction capacitance	F	0.0
CGDO	G-D overlap(i.e. constant) capacitance	F	0.0
CGS	Zero-bias G-S junction capacitance	F	0.0
CGSO	G-S overlap(i.e. constant) capacitance	F	0.0
ETA	Philips, et al.-style subthreshold conduction parameter		0.0
ETATC1	ETA first order tempco	°C ⁻¹	0.0
ETATC2	ETA second order tempco	°C ⁻²	0.0
FC	Forward bias junction fit parameter		0.5
GDSNOI	Shot noise coefficient for nlev = 3	A	2.0
GMAX	Maximum junction conductivity(straight line extension)	Ω	10K
IS	Junction saturation current	A	1e-14
ISR	Recombination current parameter	A	0.0
KF	Flicker Noise Coefficient		0.0
LAMBDA	Channel length modulation parameter	V ⁻¹	1.0
M	Grading coefficient(aka MJ)		0.5
N	Emission coefficient		1.0

NLEV	Noise level(equation selector)		0.0
NR	ISR emission coefficient		2.0
PB	Gate junction potential	V	1.0
RD	Drain resistance	Ω	0.0
RDTC1	RD first order tempco	°C ⁻¹	0.0
RDTC2	RD second order tempco	°C ⁻²	0.0
RDS	Additional Drain-Source leakage resistance	Ω	infinite
RDSTC1	RDS first order tempco	°C ⁻¹	0.0
RDSTC2	RDS second order tempco	°C ⁻²	0.0
RG	Gate resistance	Ω	0.0
RGTC1	RG first order tempco	°C ⁻¹	0.0
RGTC2	RG second order tempco	°C ⁻²	0.0
RONX	Channel conductivity multiplier in linear region ¹		1.0
RS	Source resistance	Ω	0.0
RSTC1	RS first order tempco	°C ⁻¹	0.0
RSTC2	RS second order tempco	°C ⁻²	0.0
TNOM	Parameter measurement temperature(aka TREF)	V	0.0
VK	Impact ionization knee current	V	0.0
VT0	Threshold Voltage(aka VTO or VT)	V	-2.0
VTOTC	Threshold Voltage temperature coefficient	°C ⁻¹	1.0
XTI	Saturation current temperature exponent		3.0

J. JFET Basic Equation

- JFET equation in Semiconductor Device Modeling with SPICE Section 3.1.3)
 - For channel pinched off : $V_{GS} - V_{T0} \leq 0$
 - $I_D = 0$
 - For saturated region : $0 \leq V_{GS} - V_{T0} \leq V_{DS}$
 - $I_D = \beta (V_{GS} - V_{T0})^2 (1 + \lambda V_{DS})$
 - For linear region : $0 < V_{DS} < V_{GS} - V_{T0}$
 - $I_D = \beta [2 (V_{GS} - V_{T0}) - V_{DS}] (1 + \lambda V_{DS})$

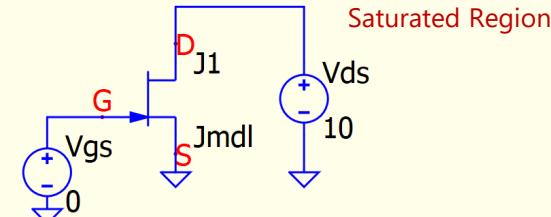


J. JFET Instance Params (I_D) : BETA and VTO

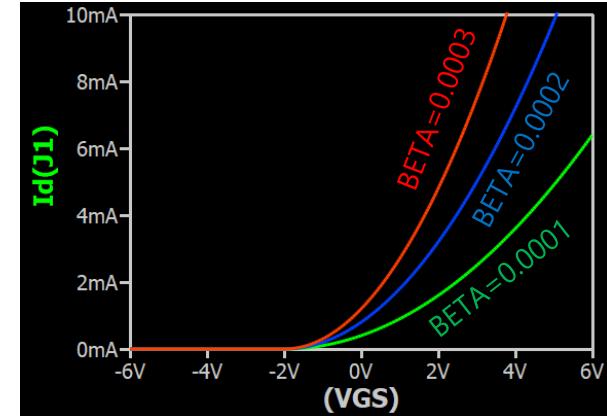
Qspice : JFET - Beta.qsch / JFET - VTO.qsch

- BETA

- Beta : Transconductance parameter
- Default BETA=0.0001A/V²**
- $I_D = \beta (V_{GS} - V_{T0})^2 (1 + \lambda V_{DS})$

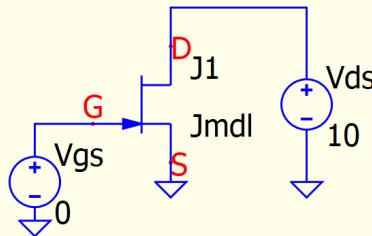


```
.dc Vgs -5 5 0.01  
.plot Id(J1)  
.step param beta list 0.0001 0.0002 0.0003  
.model Jmdl NJF BETA={beta}
```

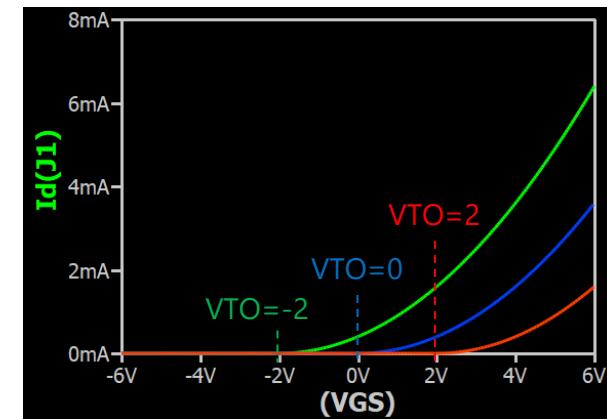


- VTO (aka VT0 or VT)

- VTO : Threshold Voltage
- Default VTO=-2V**
- $I_D = \beta (V_{GS} - V_{T0})^2 (1 + \lambda V_{DS})$



```
.dc Vgs -6 6 0.01  
.plot Id(J1)  
.step param vto list -2 0 2  
.model Jmdl NJF VTO={vto}
```

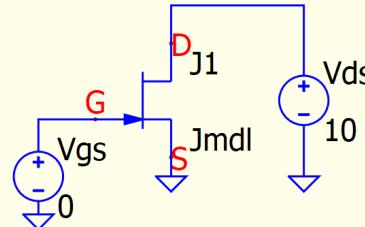


J. JFET Instance Params (I_D) : Lambda and Betatce

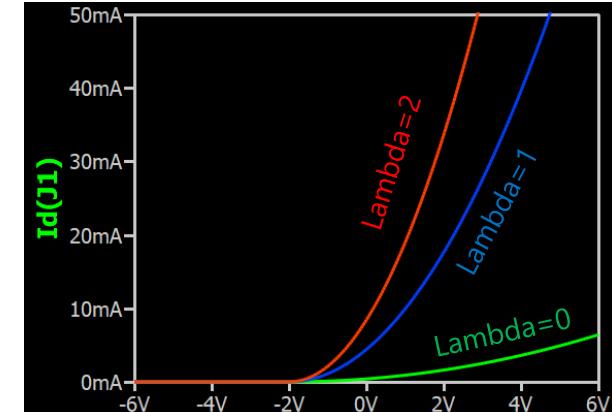
Qspice : JFET - LAMBDA.qsch / JFET - BETATCE.qsch

LAMBDA

- Lambda : Channel length modulation parameter
- **Default LAMBDA=0V⁻¹**
- $I_D = \beta (V_{GS} - V_{T0})^2 (1 + \lambda V_{DS})$

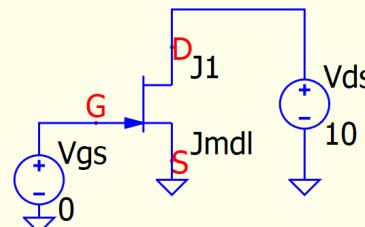


```
.dc Vgs -6 6 0.01  
.plot Id(J1)  
.step param lambda list 0 1 2  
.model Jmdl NJF LAMBDA={lambda}
```

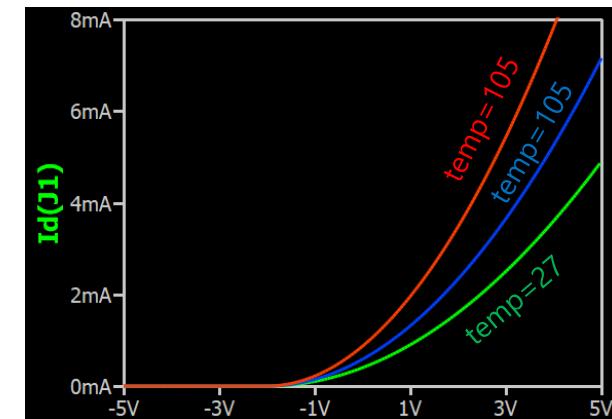


BETATCE

- Betatce : Exponential transconductance parameter temperature coefficient
- **Default BETATCE=0 (%/°C)**



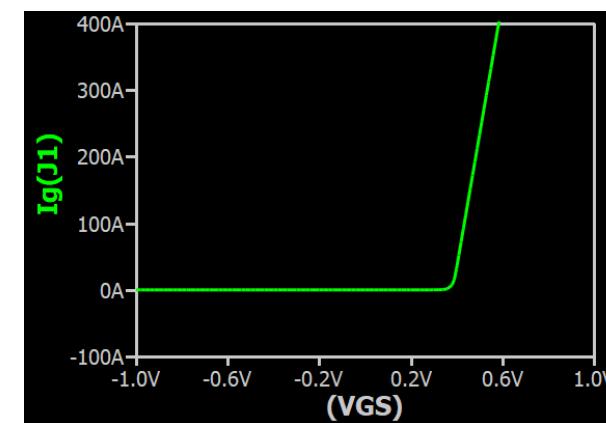
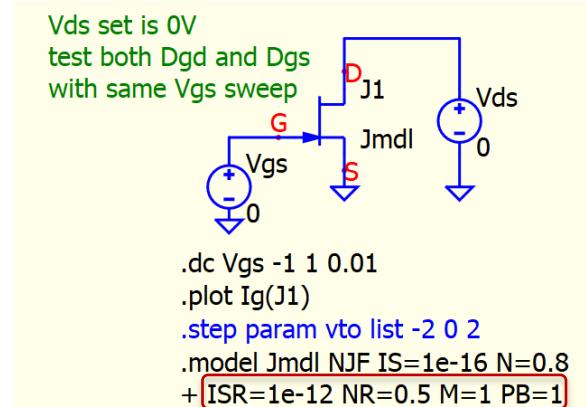
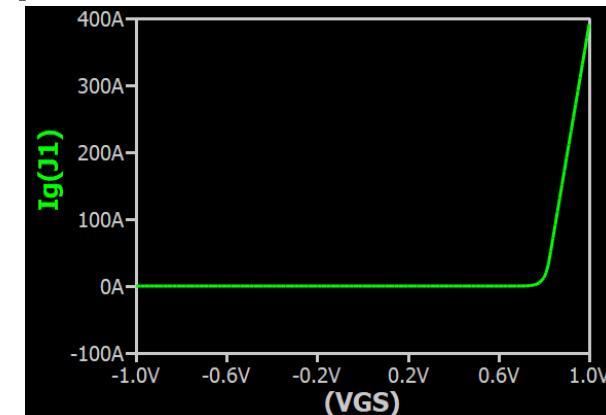
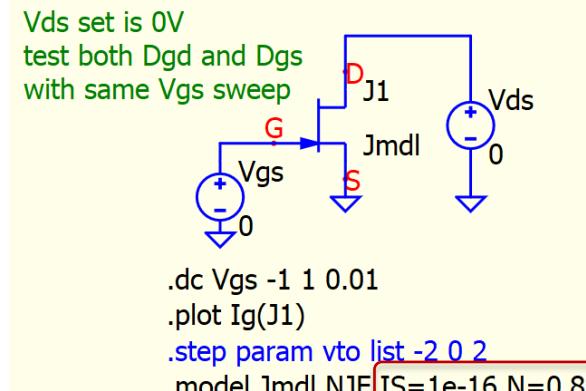
```
.dc Vgs -5 5 0.01  
.temp 27 65 105  
.plot Id(J1)  
.param betatce=1  
.model Jmdl NJF BETATCE={betatce}
```



J. JFET Instance Params (Diode GD and GS) : IS, N, ISR, NR, M and PB

Qspice : JFET - IS N.qsch / JFET - ISR NR M PB.qsch

- Two diodes in JFET
 - DGD and DGS
 - These two diodes are identical model, with recombination current included
 - $I_{GS} = (I_F + K_{gen}I_R) + I_I$
 - $I_F = I_S (e^{\frac{qV_D}{nKT}} - 1)$
 - $I_R = I_{SR} (e^{\frac{qV_D}{n_RKT}} - 1)$
 - $K_{gen} = \sqrt{\left[\left(1 - \frac{V_D}{\Phi_0} \right)^2 + 0.005 \right]^m}$
 - I_S : IS Junction saturation current (Default IS=1e-14A)
 - n : N emission coefficient (Default N=1)
 - I_{SR} : ISR Recombination current parameter (Default ISR=0A)
 - n_R : ISR emission coefficient (Default NR=1)
 - m : M Grading coefficient (Default M=0.5)
 - Φ_0 : PB Gate junction potential (Default PB=1V)
- Diode modeling refer to D. Diode section



J. JFET Instance Params (Diode GS) : Alpha and VK

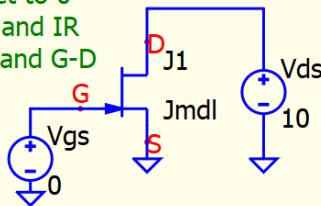
Qspice : JFET - ALPHA VK.qsch

- Gate Source Diode in Pspice model

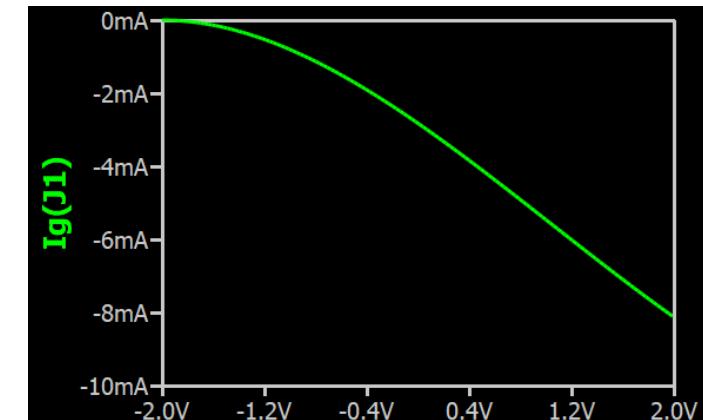
$$I_{GS} = (I_F + K_{gen} I_R) + I_I$$

- $I_I = \alpha I_D [V_{DS} - (V_{GS} - V_{TO})] e^{\frac{V_k}{V_{DS} - (V_{GS} - V_{TO})}}$ for $0 < G_{GS} - V_{TO} < V_{DS}$
- $I_I = 0$ otherwise
- α is ALPHA Impact ionization coefficient (Default ALPHA=0V⁻¹)
- V_k is VK Impact ionization knee current (Default VK=0V)

IS and ISR set to 0
to disable IF and IR
in diode G-S and G-D



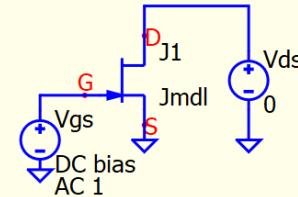
```
.dc Vgs -2 2 0.01
.plot Ig(J1)
.step param vto list -2 0 2
.model Jmdl NJF IS=0 ISR=0
+ALPHA=1 VK=1
```



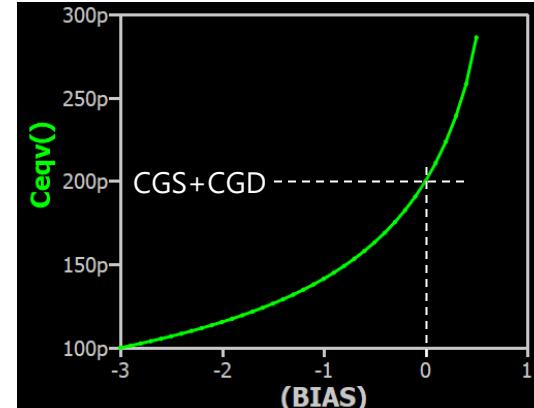
J. JFET Instance Params (Capacitance) : CGS, CGD, CGSO, CDSO

Qspice : JFET - CGS CGD.qsch / JFET - CGSO CGDO.qsch

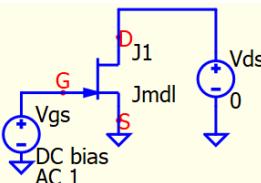
- CGS, CGD
 - CGS : Zero-bias G-D junction capacitance
 - CGD : Zero-bias G-S junction capacitance
 - **Default CGS=0F**
 - **Default CGD=0F**
 - This example simulate with both CGS and CGD



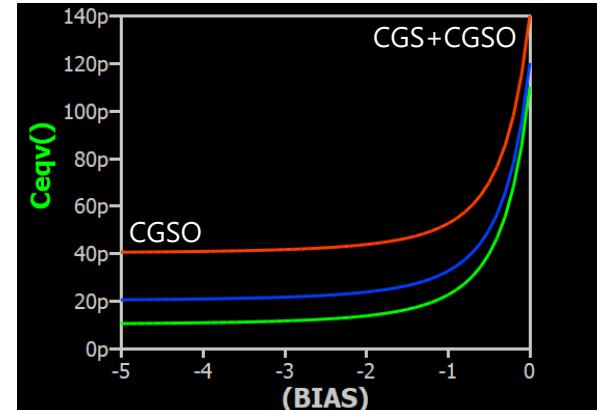
```
.ac list f      .param f=1Meg  
.step param bias -3 0.5 0.1  
.func imZ() imag(V(G)/Ig(J1))  
.func Ceqv() -1/2/pi/f/imZ()  
.plot Ceqv()  
.model Jmdl NJF CGS=100p CGD=100p
```



- CGSO, CDSO
 - CGSO : G-S overlap capacitance
 - CDSO : D-S overlap capacitance
 - **Default CGSO=0F**
 - **Default CDSO=0F**
 - This example only simulate CGS with CGSO. GDSO has same effect but to affect CDSO
 - ** M, PB, FC has effect in junction capacitance, refer to D. diode



```
.ac list f      .param f=1Meg  
.step param bias -5 0 0.1  
.func imZ() imag(V(G)/Ig(J1))  
.func Ceqv() -1/2/pi/f/imZ()  
.plot Ceqv()  
.step param cgso list 10p 20p 40p  
.model Jmdl NJF M=3 CGS=100p CGSO=cgso
```



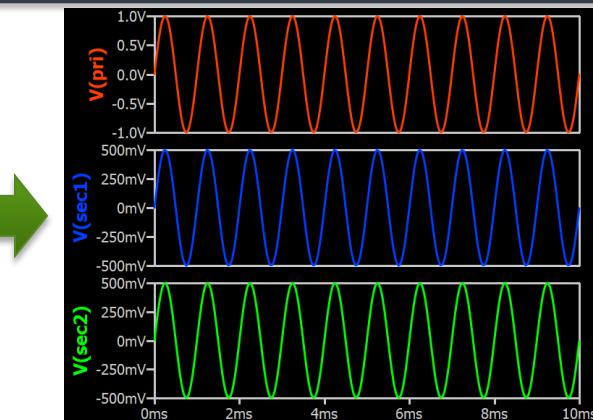
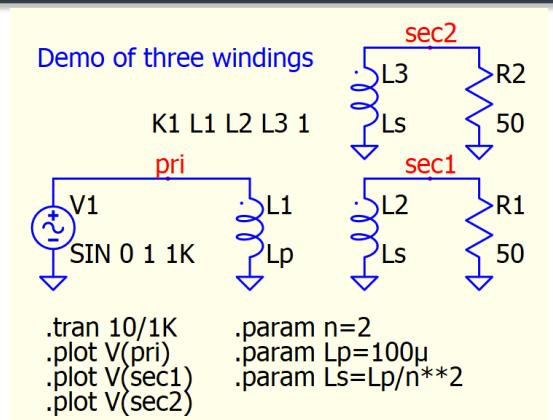
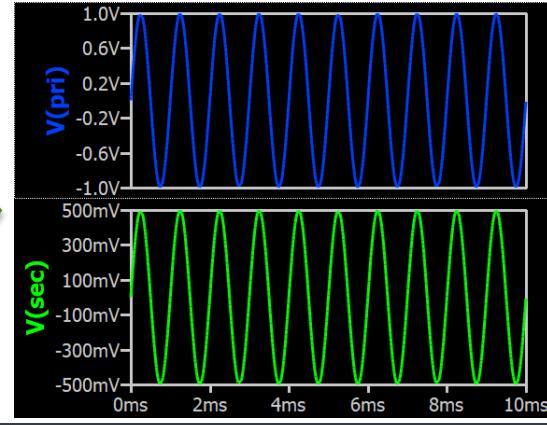
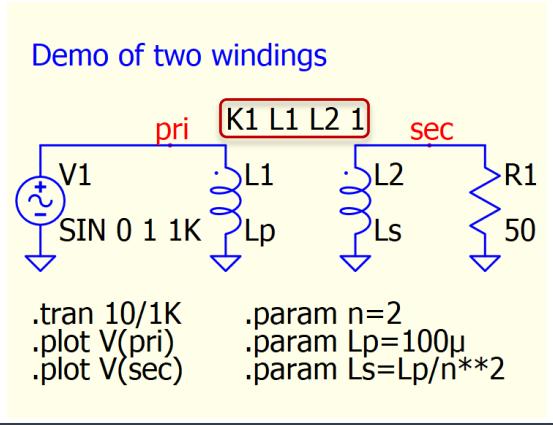
K. Mutual Inductance

K. Mutual Inductance – Basic Formula

Qspice : K Mutual - Two Winding.qsch / K Mutual - Three Winding.qsch

- K Mutual Inductance
 - K : coupling coefficient of coupled inductors
 - Ideal coupling : 1
 - Range : -1 to 1
 - Two or more coupled inductors are required
 - Press L two times to get symbol with a dot notation (not necessary, but recommend for current direction)

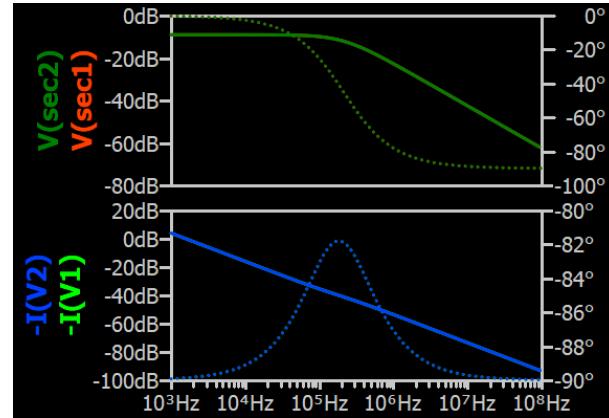
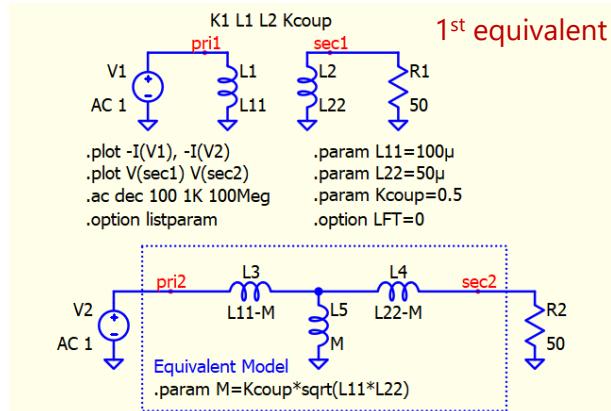
- Inductors as Transformer
 - $\frac{L_p}{N_p^2} = \frac{L_s}{N_s^2}$ and $n = \frac{N_p}{N_s}$
 - N is number of turns
 - $L_p = n^2 L_s$ or $L_s = \frac{1}{n^2} L_p$
 - In practice, L_p is measured from primary with secondary open



K. Mutual Inductance – Mutual and Self Inductance Model

Qspice : K Mutual - Self and Mutual Model.qsch

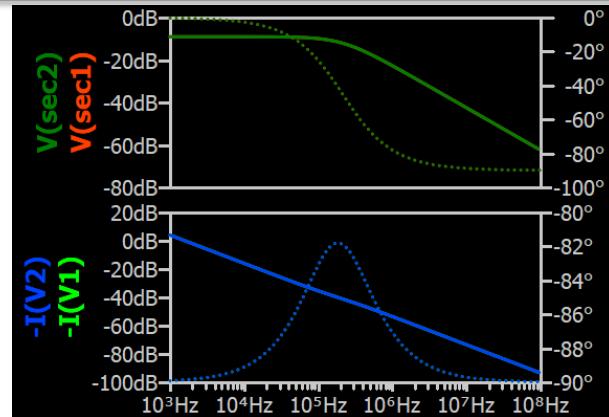
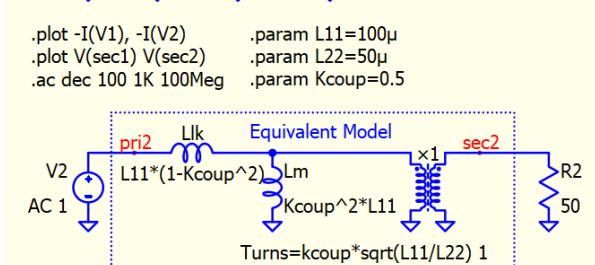
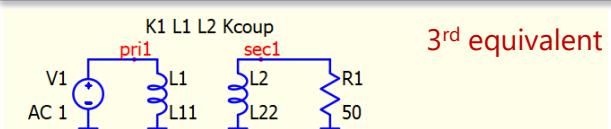
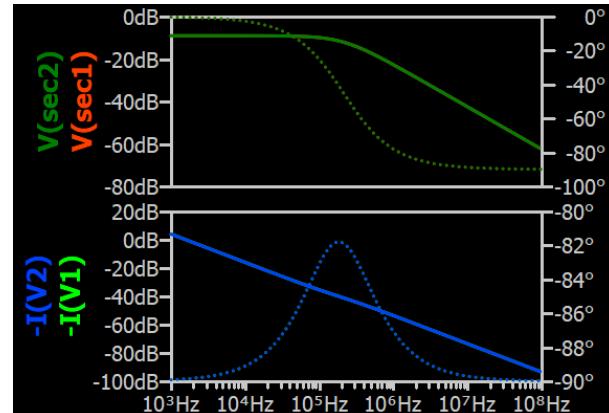
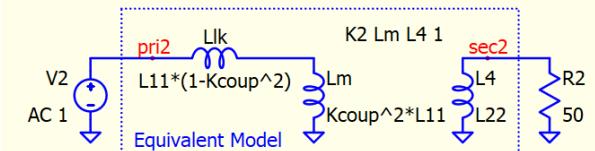
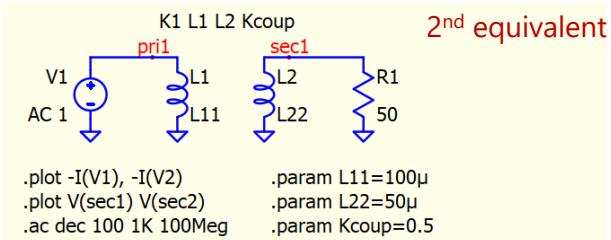
- K Mutual Inductance
 - T equivalent circuit of mutually coupled inductors: Mutual (M) and Self (L_{11} , L_{22}) model
 - $M = K\sqrt{L_{11} L_{22}}$
 - $L_1 = L_{11} - M$
 - $L_2 = L_{22} - M$
 - This is the mathematical form of mutual inductance, and a more common model with magnetizing inductance (L_m) and leakage (L_{lk}) equivalent is shown on the next slide



K. Mutual Inductance – Magnetizing and Leakage Model

Qspice : K Mutual - Lleakage - Model1.qsch | K Mutual - Lleakage - Model2.qsch

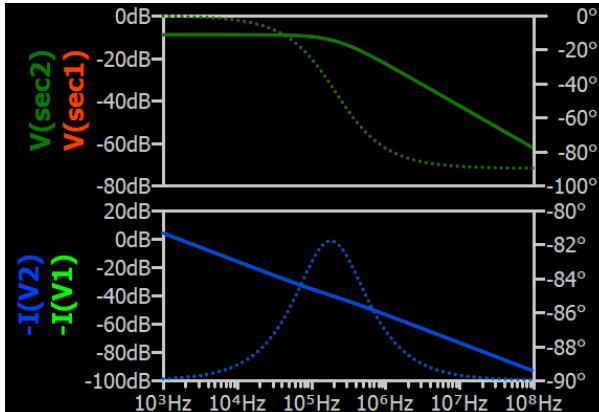
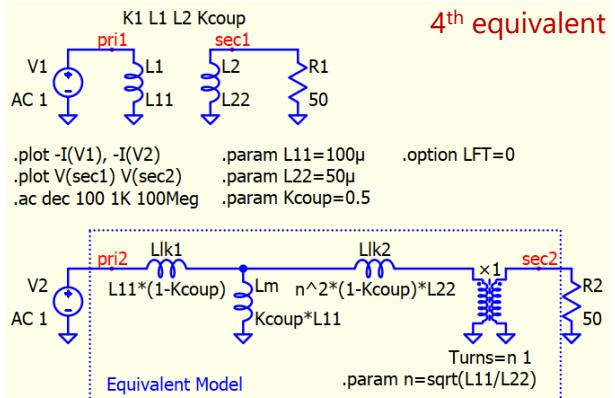
- K Mutual Inductance
 - This is equivalent model with magnetizing inductance (L_m), leakage inductance (L_{lk}) and equivalent turn ratio
 - $L_m = L_{11} K^2$
 - $L_{leakage} = L_{11} \sqrt{1 - K^2}$
 - $n = K \sqrt{\frac{L_{11}}{L_{22}}}$
 - In this example, two type of equivalent models are shown to explain effect of K in coupled inductors, one with K coupling 1 and one with ideal transformer



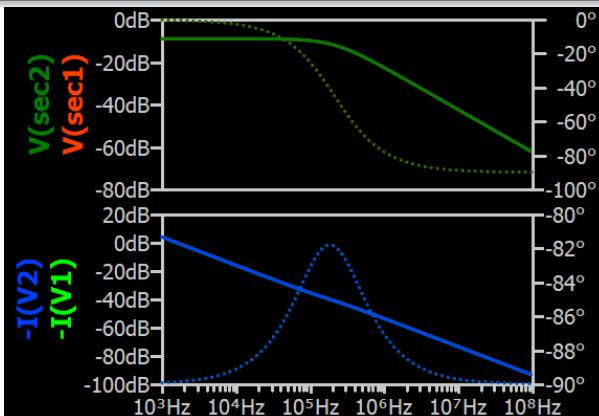
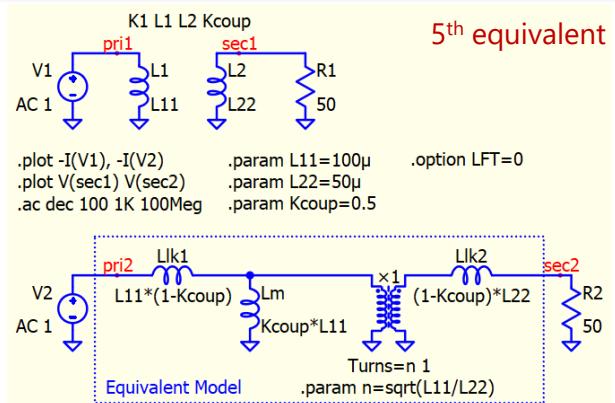
K. Mutual Inductance – Magnetizing and Leakage Model

Qspice : K Mutual - Lleakage – Model3.qsch | K Mutual - Lleakage – Model4.qsch

- K Mutual Inductance
 - Two leakage equivalent model with both in primary
 - $L_m = L_{11} K$
 - $L_{lk1} = L_{11}(1 - K)$
 - $L_{lk2} = n^2 L_{22}(1 - K)$
 - $n = \sqrt{\frac{L_{11}}{L_{22}}}$



- K Mutual Inductance
 - Two leakage equivalent model with one in primary and one in secondary
 - $L_m = L_{11} K$
 - $L_{lk1} = L_{11}(1 - K)$
 - $L_{lk2} = L_{22}(1 - K)$
 - $n = \sqrt{\frac{L_{11}}{L_{22}}}$



L. Inductor

Inductor L Instance Parameters in Qspice HELP

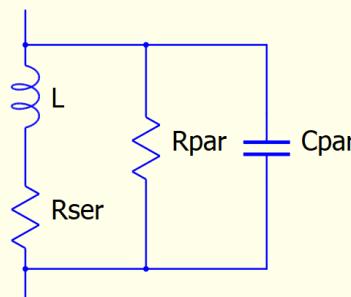
Inductor Instance Parameters			
Name	Description	Units	Default
AU	Wire or stripline is made of gold		
AL	Wire or stripline is made of aluminum		see below
AG	Wire or stripline is made of silver		
BEND	Fractional inductance correction for wire bend or proximity effects		1.
CPAR	Parallel capacitance	F	0.
CU	Wire or stripline is made of copper		see below
DIAMETER	Diameter of wire or air coil	m	
FREQUENCY	Frequency at Q. Also used to compute Rser due to skin effect		
HEIGHT	Height of PCB stripline above ground plane	m	
IC	Initial current if uic is specified on .tran statement	A	none
INDUCTANCE	Inductance of inductor	H	0.0
ISAT	Current causing inductance to drop to SATFRAC×INDUCTANCE	A	Infinite
LENGTH	Length of wire, stripline, or air coil	m	
LSAT	Inductance asymptotically approached in saturation	H	10% of INDUCTANCE
M	Number of parallel inductors		1.0
NI	Wire is made of nickel		
Q	Quality factor at FREQUENCY		
RPAR ¹	Equivalent parallel resistance	Ω	INDUCTANCE÷(15.91×GMIN)
RSER	Equivalent series resistance	Ω	0.0
SATFRAC	Fractional drop in inductance at ISAT		0.7
TC ²	Temperature coefficient polynomial		(none)
TC1 ²	1st order temperature coefficient		0.0
TC2 ²	2nd order temperature coefficient		0.0
THICK	Thickness of stripline on top of a PCB	m	0.0
TNOM	Temperature inductance was measured(aka TREF)	°C	Circuit TNOM
TURNs	Number of turns of an air coil		
VERBOSE	Print wire L, Rser, Rpar results on the console		(not set)
WIDTH	Width of stripline on top of a PCB	m	

Arbitrary Inductance Device

QSPICE also supports an arbitrary inductance device. To use it, give an equation for the flux due to the inductor. The equation can include any circuit node voltages or device currents of devices modeled as Thévenin equivalents(V-, L-, E-, or H- devices). In the interest of completeness, the device supports transinductance.

Syntax: Cnnn N1 N2 Q=<expression> [additional instance parameters]

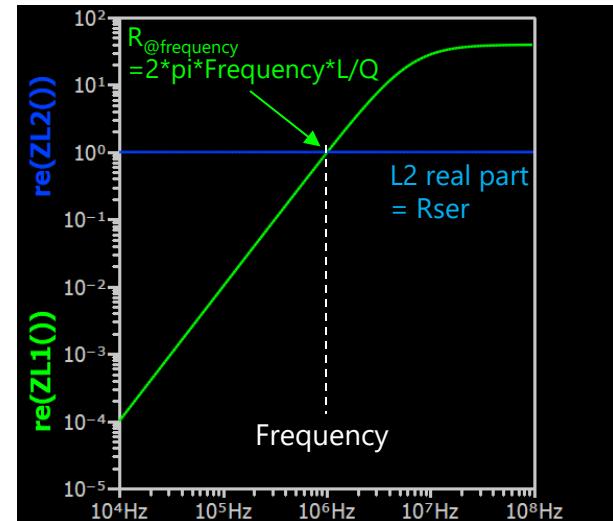
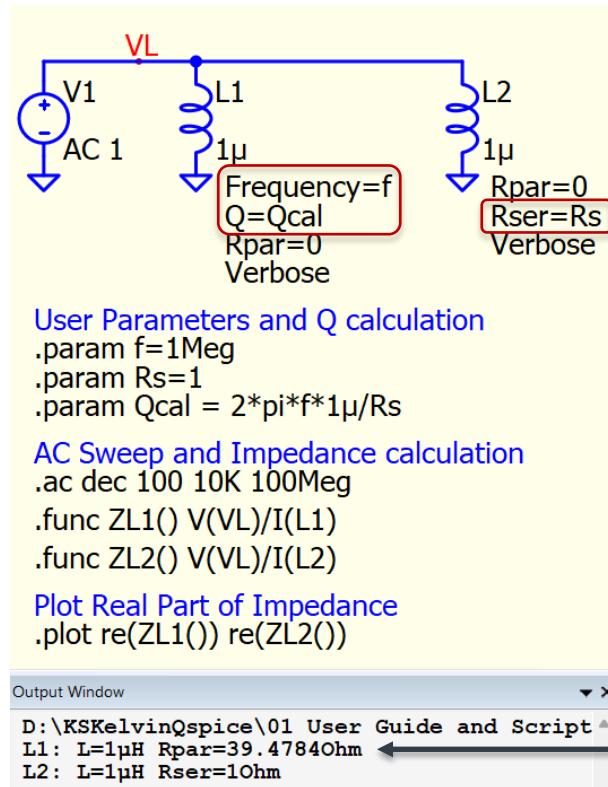
Name	Description	Units	Default
FLUX	Equation of flux(aka F)	Weber	
IC	Initial flux due to the inductor	Weber	
M	Number of identical parallel devices		1.0
RPAR	Equivalent parallel resistance	Ω	Infinite
RSER	Equivalent series resistance	Ω	0.0
NOISELESS	Ignore the noise contribution from RPAR and RSER		not set
TEMP	Instance temperature	°C	Circuit temperature



L. Instance Params : Rser, Frequency and Q

Qspice : L - Rser Frequency Q.qsch

- Rser
 - Equivalent series resistance
 - **Default Rser=0**
- Frequency and Q
 - Frequency : Frequency at Q, also used to compute Rser due to skin effect
 - Q : Quality factor at FREQUENCY
 - Formula to calculate Rpar if Rser=0 (ignore Rpar)
 - $Q = \frac{X_L}{R} = \frac{2\pi f L}{R}$
 - It doesn't actually give a changing Rser according to frequency, but rather a modeling technique uses Rpar to simulate an equivalent change in Rser with respect to frequency



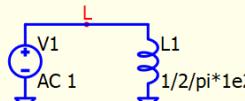
If Q and Frequency are set, Rpar is ignored and forced a value based on these two instance parameters

L. Instance Params : Rpar – Equivalent Parallel Resistance

Qspice : L - Rpar Prove.qsch | L - Rpar (.option LFT).qsch

- **Rpar**
 - Equivalent parallel resistance
 - **Default**
 $Rpar = Inductance / (1.591 * Gmin)$
 - 1.591 is only an approximate, exact formula is $\frac{1}{2\pi}$
 - Gmin is minimum conductance in .option, default gmin=1e-12
 - To calculate Rpar, .ac analysis is used, with Lp-Rp model ($Rpar=Rp$)
 - Impedance : $Z = \frac{V}{I}$
 - Admittance : $Y = \frac{I}{V} = G + jB$
 - $Rp = \frac{1}{G}$, where G is conductance
- **.option LFT for Rpar**
 - If circuit option LFT is sent, then the resistance is chosen to be equal to the reactance at frequency LFT
 - Rpar formula changed to $Rpar = 2\pi * LFT * Inductance$

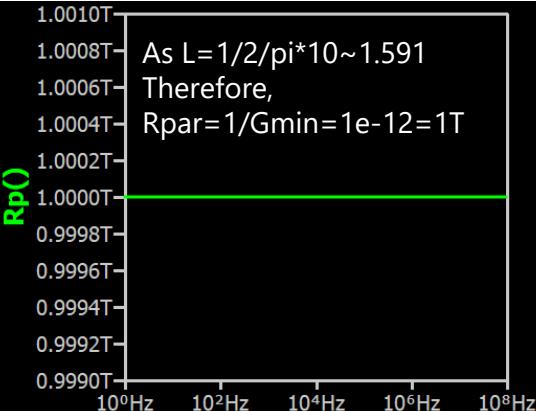
Default Rpar=Inductance/(15.91*Gmin) in HELP
**15.91 is approximation from $1/2/\pi*1e2$



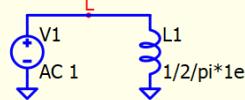
.ac dec 100 1 100Meg
Default gmin in Qspice is 1e-12
.option gmin=1e-12

** Inductor always with a default Rpar!

Formula to calculate Rp from Lp-Rp model
.func Z() V(L)/I(L1)
.func Y() 1/Z()
.func Rp() 1/re(Y())
.plot Rp()



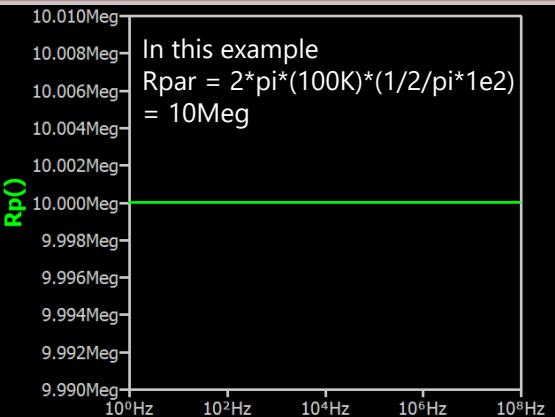
Default Rpar=Inductance/(15.91*Gmin) in HELP
**15.91 is approximation from $1/2/\pi*1e2$



.option LFT=100K
.ac dec 100 1 100Meg
Default gmin in Qspice is 1e-12
.option gmin=1e-12

** If .option LFT is used, Rpar formula is changed

Formula to calculate Rp from Lp-Rp model
.func Z() V(L)/I(L1)
.func Y() 1/Z()
.func Rp() 1/re(Y())
.plot Rp()

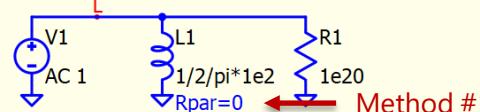


L. Instance Params : Rpar – Equivalent Parallel Resistance

Qspice : L - Rpar=0 Gmin=1e-308 LFT=1e308.qsch

- Force Rpar to Infinite
 - Three method to force Rpar to infinite
 - #1 : Set Rpar=0
 - This method is mentioned by Mike Engelhardt in Qspice forum and equivalent to force it infinite
 - #2 : Set gmin=1e-308
 - If 1/gmin is infinite, Rpar becomes infinite
 - This is useful if multiple inductors in schematic, but be caution it may introduce other problem if non-linear device in schematic when gmin is required
 - #3 : Set .option LFT=1e308
 - As LFT is infinite, Rpar becomes infinite
 - I suggest to use this method
 - OR .option LFT=0 also is infinite
 - In this demonstration, a R1 is added in parallel for visual aids, as Rpar=infinite will return NaN in Rp() and nothing is shown in waveform viewer

Default Rpar=Inductance/(15.91*Gmin) in HELP
**15.91 is approximation from $1/2/\pi \cdot 10^2$



.ac dec 100 1 100Meg

Default gmin in Qspice is 1e-12

.option gmin=1e-308

.option LFT=1e308

Method #1

Method #2 : $R_{par} = \text{Inductance}/(1591 \cdot G_{min})$

Method #3 : $R_{par} = 2 \cdot \pi \cdot LFT \cdot \text{Inductance}$

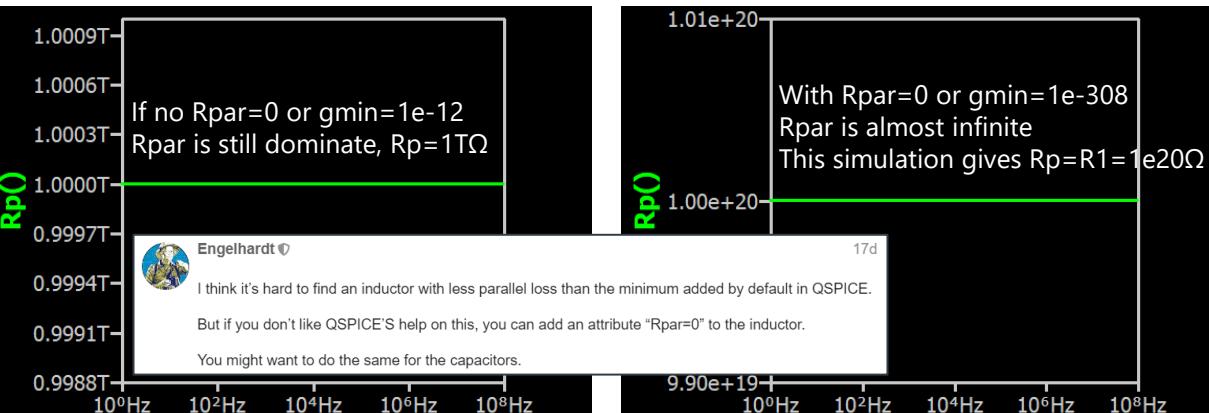
Formula to calculate Rp from Lp-Rp model:

.func Z() V(L)/-I(V1)

.func Y() 1/Z()

.func Rp() 1/re(Y())

.plot Rp()



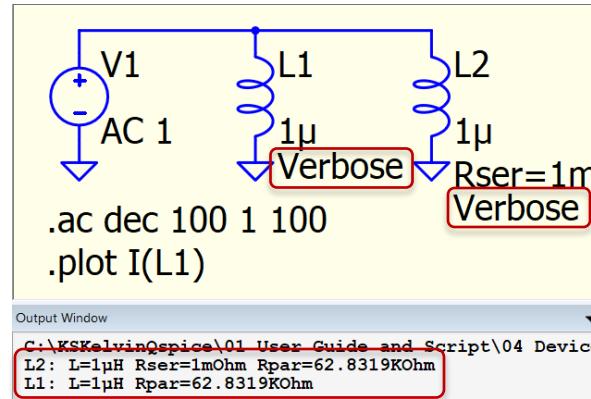
L. Instance Params : Rpar – Equivalent Parallel Resistance

- Quick Summary of Rpar
- Rpar formula priority
 - 1st priority : If **Rpar=<value>**, Rpar is set to equal <value>
 - 2nd priority : If **.option LFT** is set and **without Rpar**, Rpar = $2\pi LFT \cdot \text{Inductance}$
 - 3rd priority : If **no .option LFT** and **without Rpar**, Rpar = Inductance/(15.91*gmin)
- Force Rpar to Infinite
 - #1 : Set Instance parameter Rpar=0 individually
 - #2 : Set **.option LFT=0**
 - #3 : Set **.option gmin=1e-308** (not recommend)

L. Instance Params : Verbose

Qspice : L - Verbose.qsch

- Verbose
 - Print wire L, Rpar, Rser results on the console
 - It will not print $R_{ser}=0$ or $R_{par}=0$



L. Instance Params : Diameter, Length, Turns for Air Coil

- Diameter, Length and Turns for Air Coil
 - Diameter (d): Diameter of Air Coil in meter
 - Length (l): Length of Air Coil in meter / Wire diameter if $\frac{l}{d} < 0.002$
 - Turns (N): Number of Turns of Air Coil
 - If LENGTH, DIAMETER and TURNS are specified, the inductance of an air coil is computed

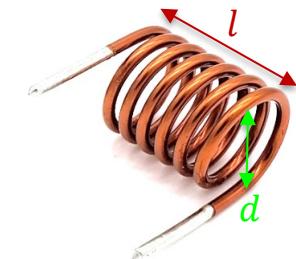
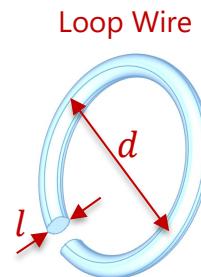
- Formula
 - For LENGTH / DIAMETER ratios less than .002, the equation for the self inductance of a loop of wire is used
 - $$L = N^2 \frac{D}{2} \mu_0 \left[\ln \left(\frac{8d}{l} \right) - 2 \right]$$
where d is coil diameter in meter and l is diameter of wire
 - For intermediate LENGTH / DIAMETER ratios, the elliptical integration is performed

$$L = K_{nagaoka} \left(\frac{\mu_0 \pi \left(\frac{d}{2} \right)^2 N^2}{l} \right)$$

where $K_{nagaoka}$ is Nagaoka's coefficient, an extremely accurate approximation formula is Lundin's formula

$$K_{nagaoka} \sim \frac{2}{\pi} \left(\frac{l}{d} \right) \left[\frac{\left(\ln \left(\frac{4d}{l} \right) - \frac{1}{2} \right) \left(1 + 0.383901 \left(\frac{l}{d} \right)^2 + 0.017108 \left(\frac{l}{d} \right)^4 \right)}{1 + 0.258952 \left(\frac{l}{d} \right)^2} + 0.093842 \left(\frac{l}{d} \right)^2 + 0.002029 \left(\frac{l}{d} \right)^4 + 0.000801 \left(\frac{l}{d} \right)^6 \right]$$

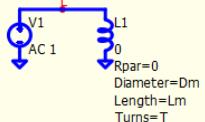
- For LENGTH / DIAMETER ratios greater than 0.6, the ARRL formula is used
 - $$L(\mu\text{H}) = \frac{d^2 N^2}{18d + 40l}$$
 where d and l are in inch (1 inch = 0.0254 meter)



L. Instance Params : Diameter, Length, Turns for Air Coil

Qspice : L - Length Diameter Turns for Air Coil.qsch

Air Coil with Length Diameter and Turns in Qspice



For Air Coil, Diameter refer to Diameter of Coil
 $.param \text{inch2m}=0.0254$; inch to meter conversion
 $.param \text{Linch}=\text{var}$; Length in inch
 $.param \text{Dinch}=1$; Diameter in inch
 $.param \text{T}=10$; number of Turns
 $.param \text{Lm}=\text{Linch}*\text{inch2m}$; Length in meter
 $.param \text{Dm}=\text{Dinch}*\text{inch2m}$; Diameter in meter



As Dinch set to 1, L/D Ratio = Linch
 $.step \text{param var } 0.0001 \ 1 \ 0.001$
 $.step \text{dec param var } 0.0001 \ 1 \ 20$

$.param f=10\text{Meg}$
 $.ac \text{list } f$
 $.func \text{imZ}() \text{ im}(V(L)-I(V1))$
 $.func \text{LsQspice}() \text{ imZ}() / 2 \pi / f$

$$\text{L(Dratio)} = \text{AC} (\text{Dinch}^2 * \text{T}^2) / (18 * \text{Dinch} + 40 * \text{Linch}) * 1e-6$$

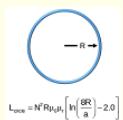
ARRL formula : for Length/Diameter ratios > 0.6
<https://www.arrl.org/files/file/Technology/tis/info/pdf/9708033.pdf>
 $L(\mu\text{H}) = (\text{Dinch}^2 * \text{T}^2) / (18 * \text{Dinch} + 40 * \text{Linch}) * 1e-6$

$$L(\mu\text{H}) = \frac{d^2 n^2}{18d + 40f}$$

$$\text{Loop} \quad V4$$

$$\text{AC } \text{T}^2 * (\text{Dm}/2)^2 * \text{u0}^0 * (\log(8 * \text{Dm}/2 / (\text{Lm}/2)) - 2)$$

Self Inductance of a loop of wire : for Length/Diameter ratios < 0.002
https://learnemc.com/EXT/calculators/Inductance_Calculator/L-circle.html
 $L(\text{Henry}) = \text{T}^2 * (\text{Dm}/2)^2 * \text{u0}^0 * (\log(8 * \text{Dm}/2 / (\text{Lm}/2)) - 2)$



R=radius of coil=Dm/2
 a =radius of wire=Lm/2
 ur =relative permeability=1 for air

$$L_{\text{loop}} = N^2 R^2 \mu_0 \mu_r \left[\ln \left(\frac{8R}{a} \right) - 2 \right]$$

$$\text{Lnagaoka} \quad V6$$

$$\text{AC } \text{KN}^0 = (\text{u0}^0 * \text{pi}^0 * (\text{Dm}/2)^2 * \text{T}^2) / \text{Lm}$$

Inductance with Nagaoka's coefficient
 $L(\text{Henry}) = \text{KN}^0 * (\text{u0}^0 * \text{pi}^0 * (\text{Dm}/2)^2 * \text{T}^2) / \text{Lm}$
 $; \text{Lundin's formula for short coils (approximation of Nagaoka's coefficient)}$
 $.param \text{DL}=\text{Dm}/\text{Dm}$
 $.param \text{LD}=\text{Lm}/\text{Dm}$
 $.param \text{KN}^0=2/\pi^0 * \text{LD}^2 * ((\ln(4 * \text{DL}) - 0.5) * (1 + 0.383901 * \text{LD}^2 + 0.017108 * \text{LD}^4) / (1 + 0.258952 * \text{LD}^2) + 0.093842 * \text{LD}^2 + 0.002029 * \text{LD}^4 - 0.000801 * \text{LD}^6)$
 $; \text{equivalent to LariL (Wheeler's formula, approximation of Nagaoka's coefficient)}$
 $.param \text{KN}=1/(1+0.9 * \text{Dm}/\text{Lm})$

Final Inductance Formula

The total inductance of a solenoid with N turns is:

$$L = K_N \cdot \left(\frac{\mu_0 \pi a^2 N^2}{l} \right)$$

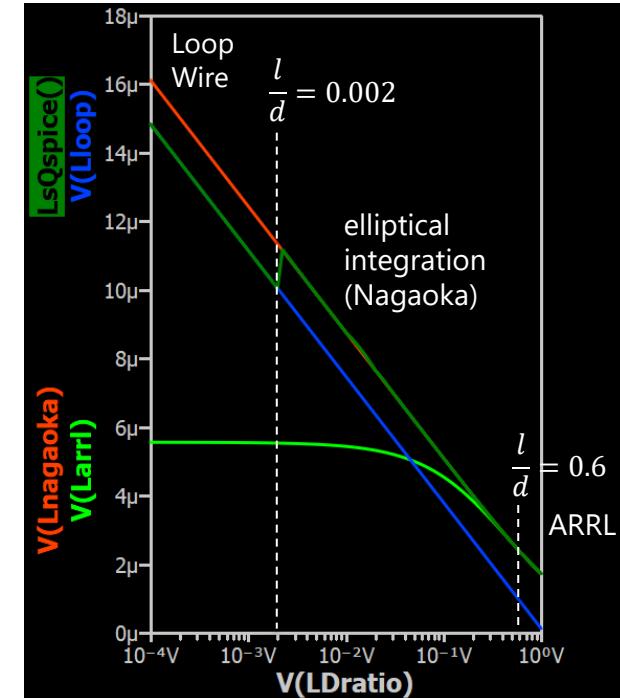
KN: Nagaoka's coefficient

An extremely accurate approximation formula for Nagaoka's coefficient is due to Richard Lundin
[Reference https://www.g3ynh.info/zdocs/magnetics/Solenoids.pdf](https://www.g3ynh.info/zdocs/magnetics/Solenoids.pdf)

Lundin's formula for short coils ($D \geq l$). Max. error: < 2 ppM (0.0002%).

$$K_{LS} = \frac{2}{\pi} \left(\ell / D \right) \left[\frac{[\ln(4 D / \ell) - 1/2] [1 + 0.383901 (\ell / D)^2 + 0.017108 (\ell / D)^4]}{[1 + 0.258952 (\ell / D)^2]} \right]$$

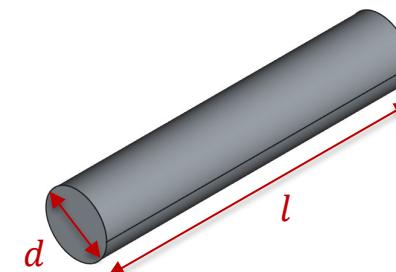
$$+ 0.093842 (\ell / D)^2 + 0.002029 (\ell / D)^4 - 0.000801 (\ell / D)^6 \right]$$



$$\frac{\text{Length}}{\text{Diameter}} = \frac{l}{d}$$

L. Instance Params : Diameter and Length for Straight Wire

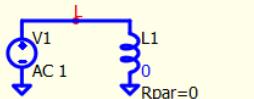
- Diameter, Length and Turns for Straight Wire
 - Diameter (d): Diameter of Wire in meter
 - Length (l): Length of Wire in meter
 - If LENGTH are DIAMETER are specified, but not TURNS, it computes the inductance and resistance of a straight wire per Grover's Inductance Calculations. Wire material defaults to Aluminum is this case.
- Formula
 - Grover's Inductance Calculation
 - $$L(\mu\text{H}) = 0.002 l \left[\ln\left(\frac{4l}{d}\right) - 1 + \frac{\mu_r}{4} + \frac{d}{2l} \right]$$
 - where l and d are in centimeters (cm)
 - Resistance Calculation
 - $$R = \rho \frac{l}{A} = \rho \frac{l}{\pi \left(\frac{d}{2}\right)^2}$$
 - where $\rho(T) = \rho_0 [1 + \alpha(T - T_0)]$
 - For Aluminum, resistivity $\rho_0 = 2.615 \times 10^{-8} \Omega\text{m}$ and temperature coeff $\alpha = 3.9 \times 10^{-3} \text{K}^{-1}$ @ $T_0 = 20^\circ\text{C}$
 - https://en.wikipedia.org/wiki/Electrical_resistivity_and_conductivity



L. Instance Params : Diameter and Length for Straight Wire

Qspice : L - Length Diameter for Wire.qsch

Wire with Length and Diameter in Qspice



For Wire (no Turn is defined), Diameter refer to Wire Diameter
.param Lm=var ; Length in meter
.param Dm= 0.002; Diameter in meter
.param Lcm=Lm*100
.param Dcm=Dm*100

.param f=10Meg
.ac list f
.step dec param var 0.01 0.1 20

.func Rser() re(V(L)/-I(V1))
.func imZ() im(V(L)/-I(V1))
.func LsQspice() imZ()/2/p/f
.plot Rser() V(res)
.plot LsQspice() V(Lcal)
;.plot V(LDratio)



.param resistivity=2.615*1e-8*(1+3.9e-3*(Temp-20))

$$R = \rho \frac{\ell}{A}$$

https://en.wikipedia.org/wiki/Electrical_resistance_and_conductance
https://en.wikipedia.org/wiki/Electrical_resistivity_and_conductivity

$$\rho(T) = \rho_0 [1 + \alpha(T - T_0)]$$



.option saveparams
.option LISTPARAM

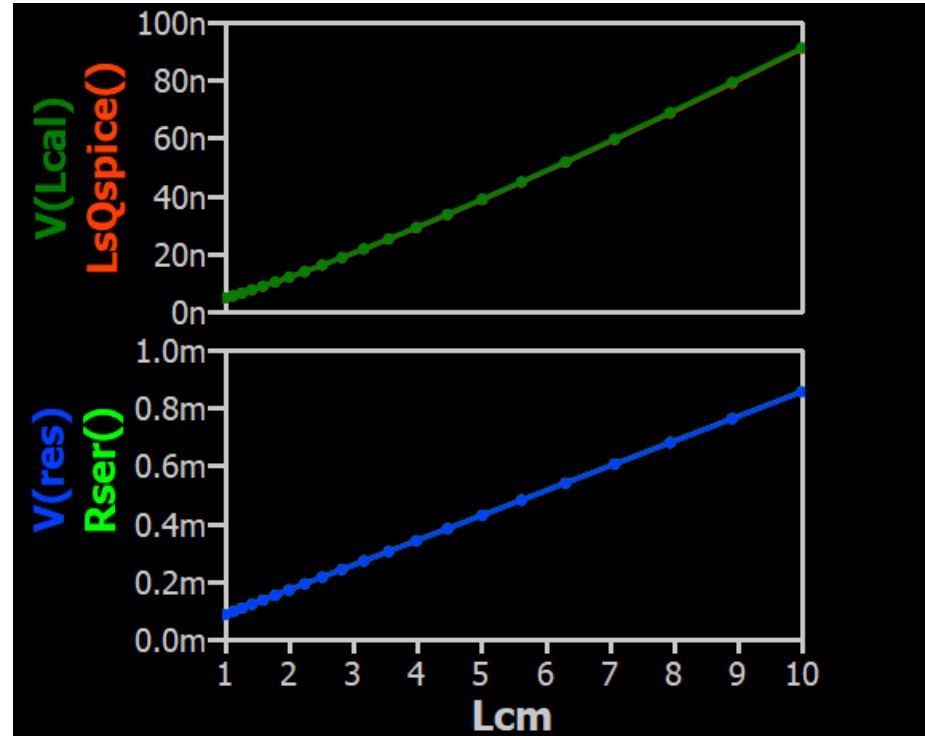
Grover's Inductance Formula for a Straight Wire

The self-inductance L of a straight round wire is given by:

$$L = 0.002l \left[\ln \left(\frac{4l}{d} \right) - 1 + \frac{\mu_r}{4} + \frac{d}{2l} \right] \quad (\text{in } \mu\text{H})$$

Where:

- * L = Self-inductance in **microhenries** (μH)
- * l = Length of the wire in **centimeters** (cm)
- * d = Diameter of the wire in **centimeters** (cm)
- * μ_r = Relative permeability of the wire material (≈ 1 for copper, aluminum, and other non-magnetic materials)

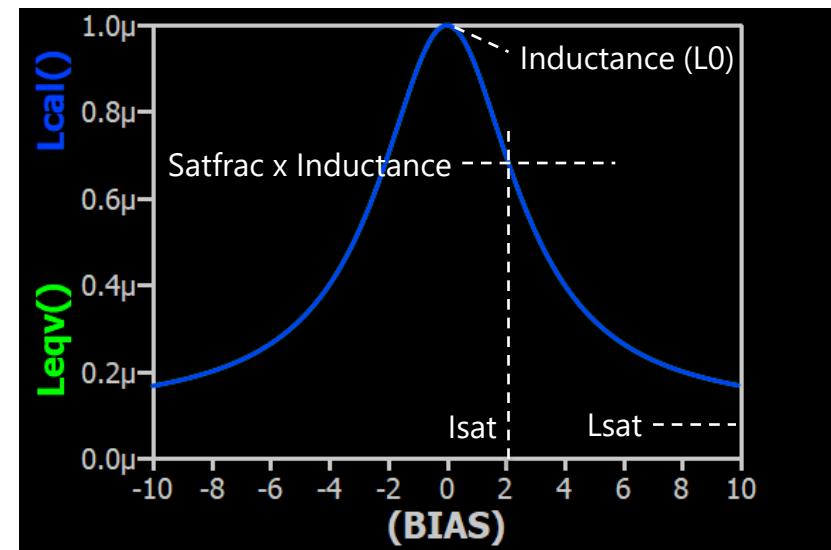
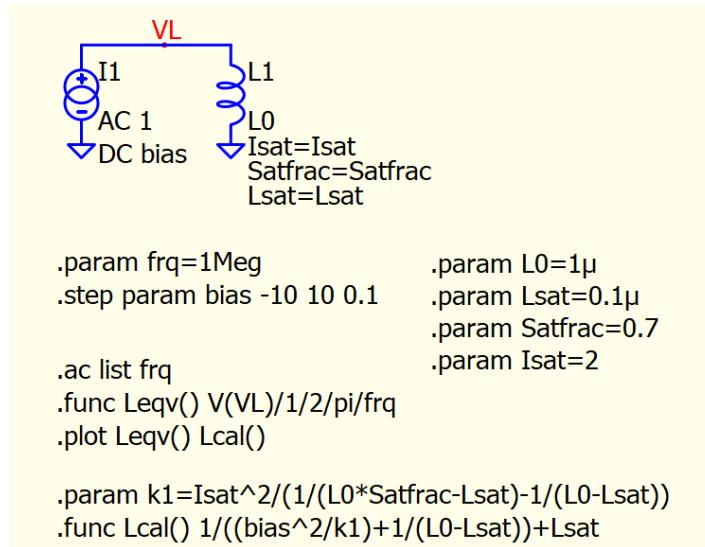


L. Instance Params : Isat, Lsat, Satfrac : For Nonlinear Inductance

Qspice : L - Lsat Satfrac Isat - Eqn.qsch

- Formula of Isat, Lsat and Satfrac, with Inductance= L_0

- $$L(I_{DC}) = \frac{1}{\left(\frac{I_{DC}^2}{k} + \frac{1}{L_0 - L_{sat}}\right)} + L_{sat} \quad \text{where } k = \frac{I_{sat}^2}{\left(\frac{1}{L_0 \text{SATFRAC} - L_{sat}} - \frac{1}{L_0 - L_{sat}}\right)}$$

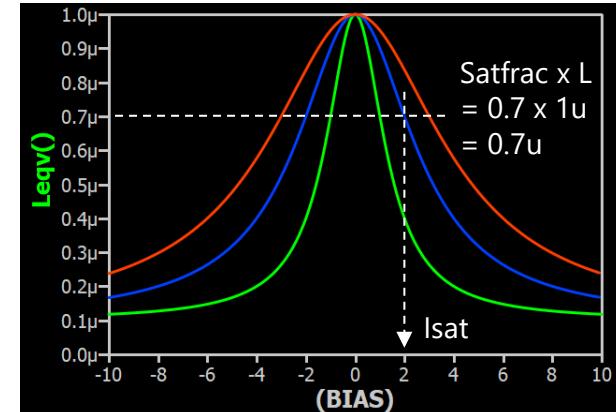
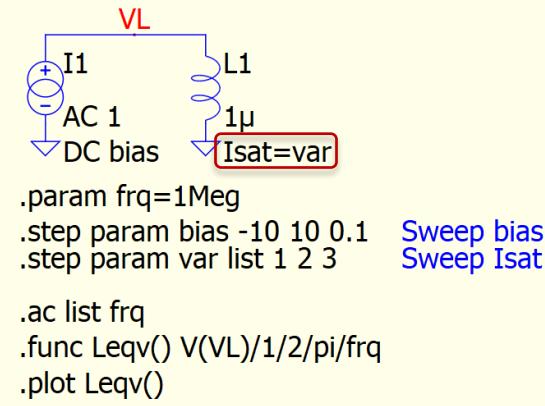


L. Instance Params : Isat, Lsat, Satfrac - For Nonlinear Inductance

Qspice : L - Isat Satfrac (.ac).qsch

- Isat and Satfrac

- Isat : Current that drops inductance to Satfrac x Inductance
- Satfrac : Fractional drop in inductance at Isat
- **Default ISAT=0**
 - i.e. ISAT is infinite
- **Default SATFRAC=0.7**
- Nonlinear effect is applied to +ve and -ve current direction

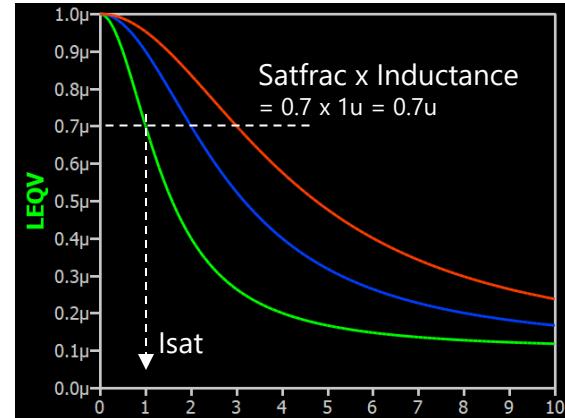
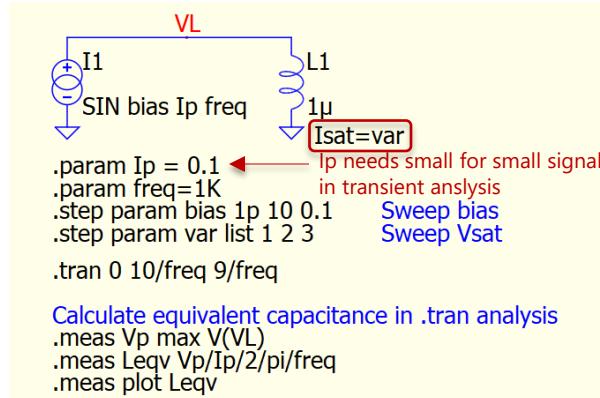


- Inductance equation
 - $X_L = \omega L = \frac{V_L}{I_L}$
 - $L = \frac{|V_L|}{|I_L|} \frac{1}{\omega} = \frac{|V_L|}{2\pi f |I_L|}$
 - These formulas are used to calculate the equivalent inductance (small signal model) at different inductor bias currents

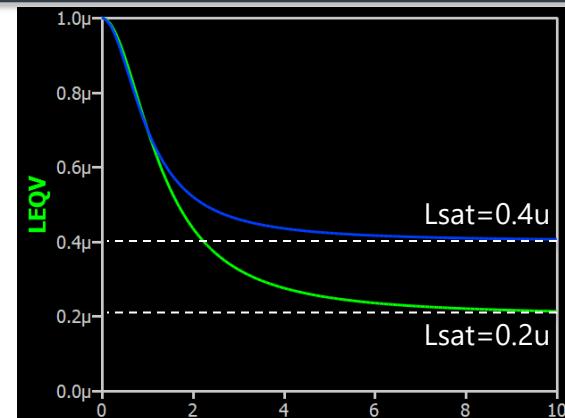
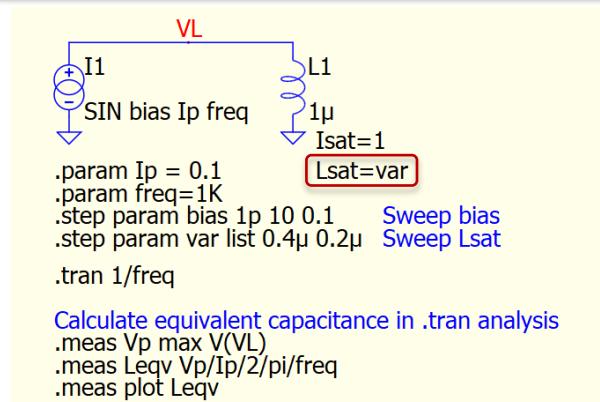
L. Instance Params : Isat, Lsat, Satfrac - For Nonlinear Inductance

Qspice : L - Isat Satfrac (.tran).qsch | L - Lsat (.tran).qsch

- Isat and Satfrac
 - **Isat** : Current that drops inductance to **Satfrac x Inductance**
 - **Satfrac** : Fractional drop in inductance at Isat
 - **Default ISAT=0**
 - **Default SATFRAC=0.7**



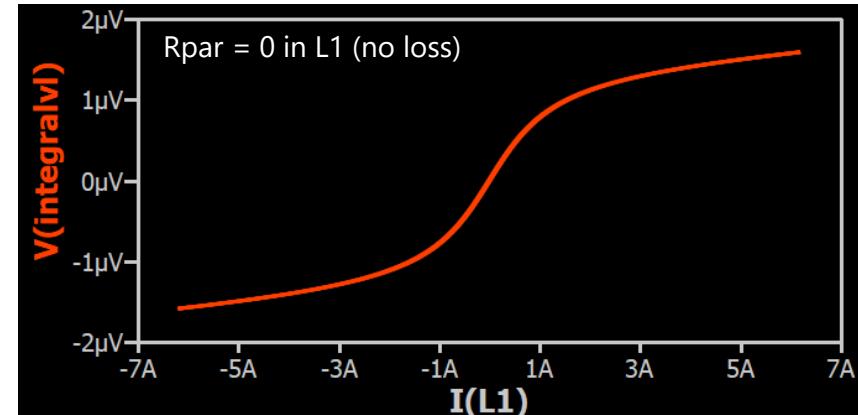
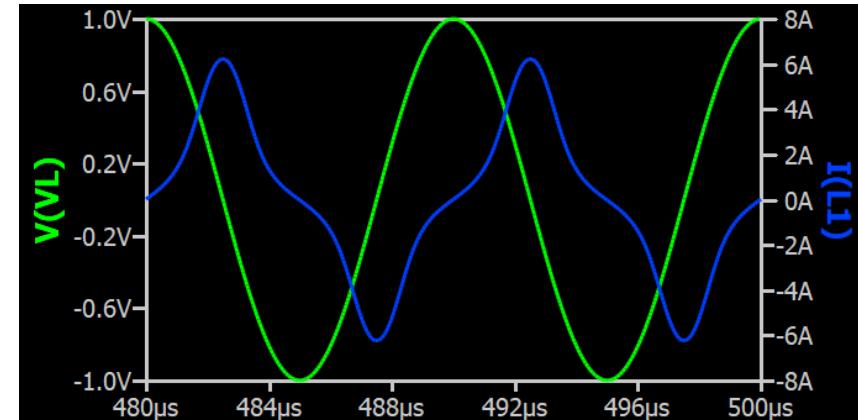
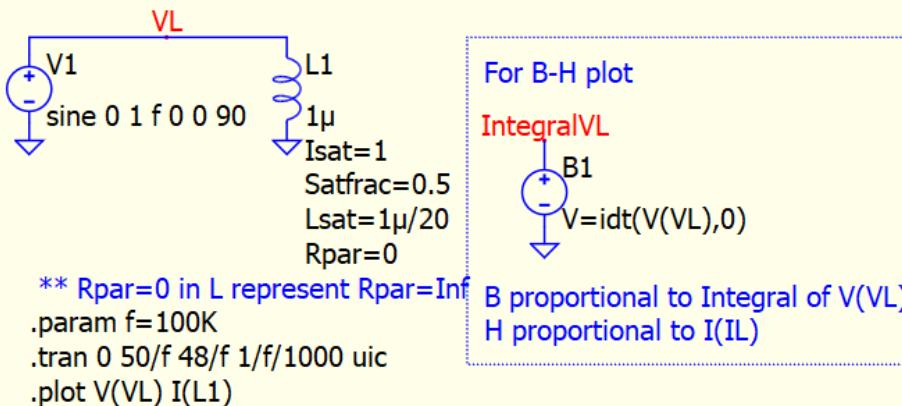
- Lsat
 - **Lsat** : Inductance asymptotically approached in saturation
 - Isat must be set to non-zero to take effect



Demonstration of Nonlinear Inductor (model B-H saturation)

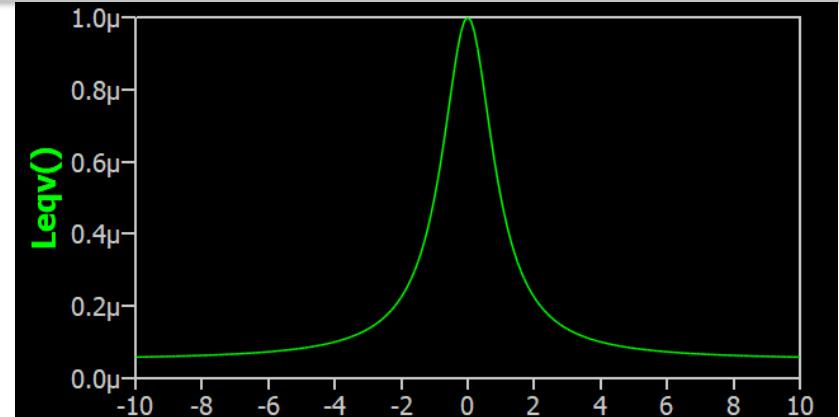
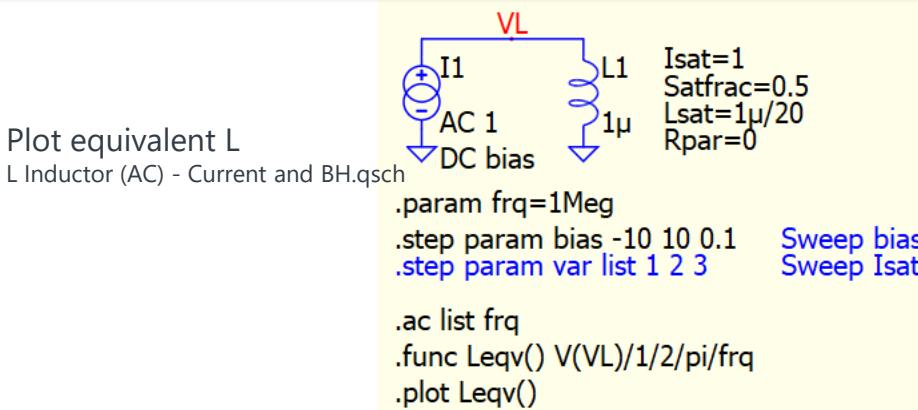
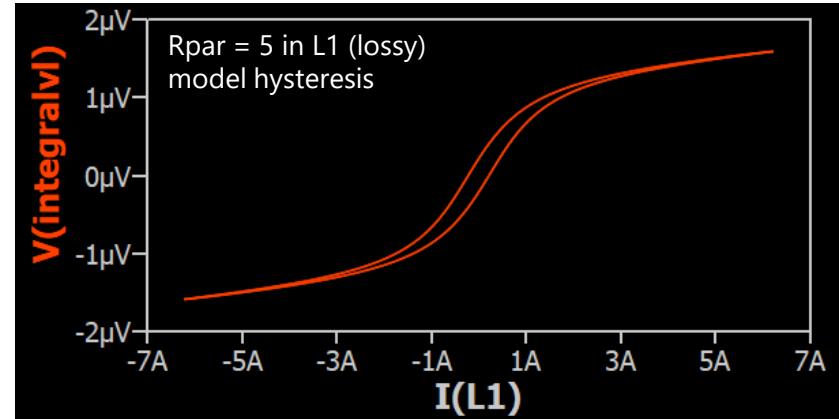
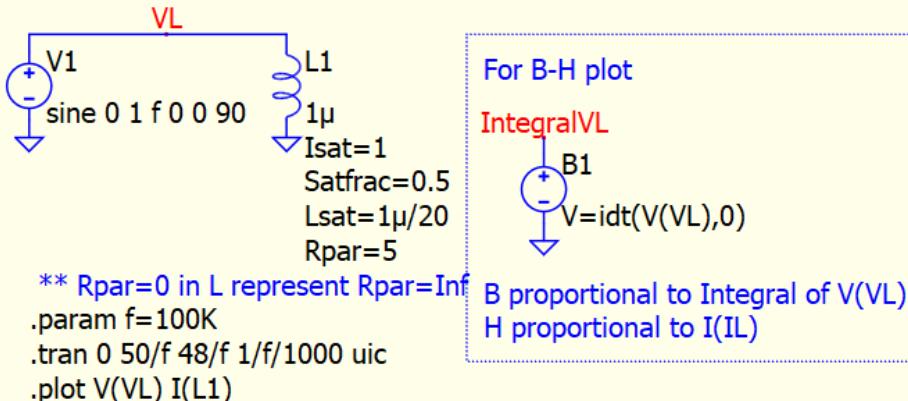
Qspice : L Inductor Nonlinear (Tran) - Current and BH.qsch

- Nonlinear Inductor in Qspice
 - This simulation demonstrate transient of non-linear inductor with a sinusoidal voltage source, which can expect a non-linear current
- Reference : B and H for inductor
 - Magnetic field intensity $H = \frac{B}{\mu} = \frac{N}{l_e} i_L$ (unit : A/m)
 - Magnetic field strength $B = \frac{\phi_B}{A_e} = \frac{1}{N A_e} \int v_L dt$ (unit : Tesla)
 - B also called magnetic flux density



Demonstration of Nonlinear Inductor (model B-H hysteresis : lossy)

Qspice : L Inductor Nonlinear (Tran) - Current and BH.qsch



Comment about Saturation and Hysteresis of B-H curve

Isat, Lsat and Satfrac are key parameters in Qspice for non-linear inductance modeling. General interest is just a saturating include. This represents inductor saturating when current reach certain level (i.e. inductance drop according to current, which can model by Isat, Lsat and Satfrac). This characteristic is saturation in B-H curve.

For hysteresis of B-H curve, it can generally model with a parallel resistor Rpar to represent loss.

May be some people prefer to model non-linear inductor with other way, but I found Qspice definition generally good enough as L vs bias current can be easily measure practically.

L. Arbitrary Inductance : Inductor modeling with FLUX

- Arbitrary Inductance Device

- To use arbitrary inductance device, give an equation for the flux due to the inductor

Arbitrary Inductance Device

QSPICE also supports an arbitrary inductance device. To use it, give an equation for the flux due to the inductor. The equation can include any circuit node voltages or device currents of devices modeled as Thévenin equivalents(V-, L-, E-, or H- devices). In the interest of completeness, the device supports transinductance.

Syntax: Cnnn N1 N2 Q=<expression> [additional instance parameters]

Name	Description	Units	Default
FLUX	Equation of flux(aka F)	Weber	
IC	Initial flux due to the inductor	Weber	
M	Number of identical parallel devices		1.0
RPAR	Equivalent parallel resistance	Ω	Infinite
RSER	Equivalent series resistance	Ω	0.0
NOISELESS	Ignore the noise contribution from RPAR and RSER		not set
TEMP	Instance temperature	$^{\circ}\text{C}$	Circuit temperature

- Basic Formula between Inductance and Flux

- [equation 1] : Inductor Voltage v_L and Flux ϕ relationship : $v_L = \frac{d\phi}{dt}$
- [equation 2] : Inductor Voltage v_L and Inductor Current i_L relationship : $v_L = L \frac{di_L}{dt}$
- By $\frac{d\phi}{dt} = L \frac{di_L}{dt} \rightarrow \text{Inductance } L = \frac{d\phi}{di_L}$

L. Arbitrary Inductance : Inductor modeling with FLUX

Qspice : Flux formula.qsch

- Qspice inductor modeling with Flux ϕ
 - For linear inductor, if $\frac{d\phi}{di_L}$ is constant $\rightarrow \phi = L i_L$
 - Therefore, $L = \frac{\phi}{i_L}$
 - In Qspice : $\text{flux} = L * I(\text{Lnnn})$
 - where Lnnn is the name attribute of inductor symbol (e.g. L1, L2)
 - For nonlinear inductor, ϕ can be expressed by
$$\phi = a \tanh(b i_L)$$
 - Therefore, $L = \frac{d\phi}{di_L} = \frac{d a \tanh(b i_L)}{di_L} = a b (1 - \tanh^2(b i_L))$
 - In Qspice : $\text{flux} = a * \tanh(b * I(\text{Lnnn}))$
 - $L_{0A} @ 0A = a * b$: L_{0A} is inductance at 0A
 - b determines the spread
 - If $b i_L$ is large, $(1 - \tanh^2(b i_L)) = 0$, therefore, this formula will give 0H inductance when i_L is large
 - Nonlinear and linear flux formula can be combined to handle 0H inductance when i_L is large
 - In Qspice : $\text{flux} = a * \tanh(b * I(\text{Lnnn})) + L_{\min} * I(\text{Lnnn})$
 - $L_{\min} @ i_L = \infty$: L_{\min} : L_{\min} is inductance at current $\rightarrow \infty$
 - $L_{0A} @ i_L = 0$: $a * b - L_{\min}$
 - b determines the spread

$$L = \frac{d\phi}{di_L} = \text{dflux}()$$

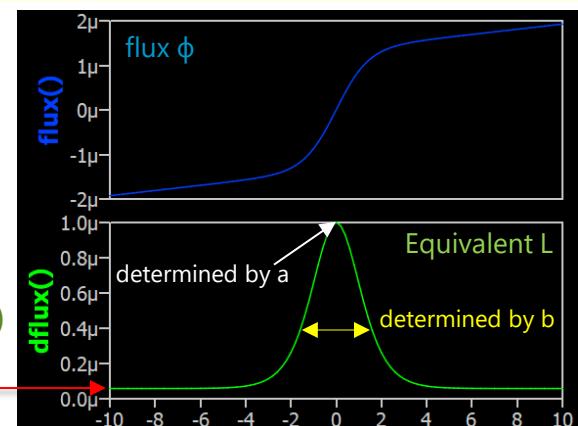
determined by L_{\min}

Assume flux formula is $a * \tanh(b * IL) + L_{\min} * IL$
.param b = 0.7
.param L0A = 1u
.param a = (L0A-Lmin)/b
.param Lmin=58n
.func flux() a*tanh(b*IL)+Lmin*IL

Calculate Inductance by dflux/dIL
.func dflux() D(flux())

Sweep IL from -10A to 10A

.step param IL -10 10 0.2
.op
.plot dflux()
.plot flux()

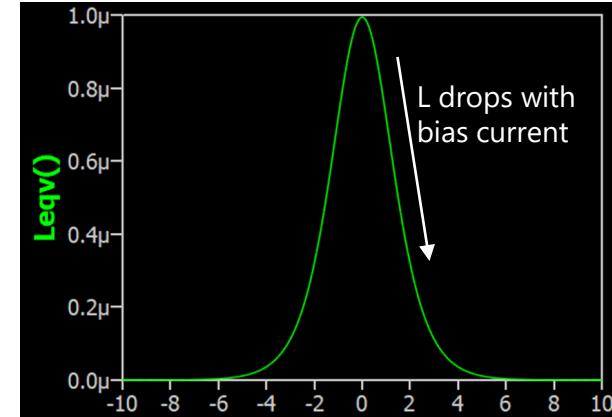
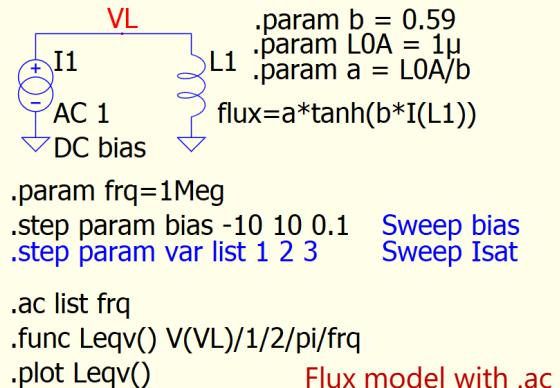


L. Arbitrary Inductance : Inductor modeling with FLUX

Qspice : L Inductor (AC) - Flux eqn 1.qsch ; L Inductor Nonlinear (Tran) - Flux eqn 1.qsch

- Flux model

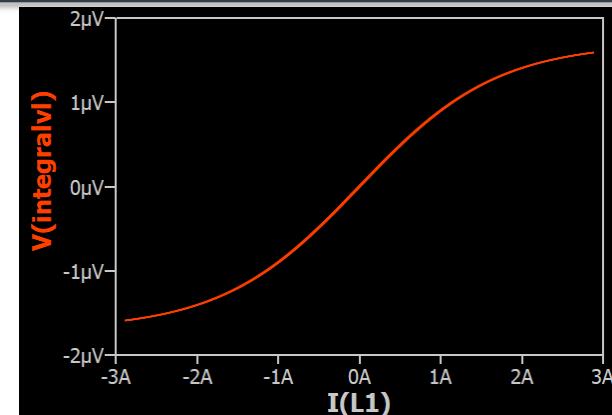
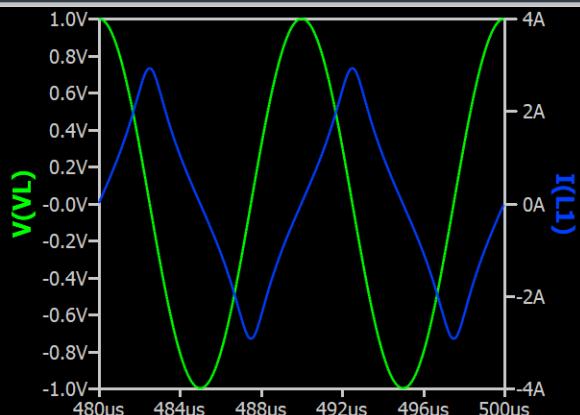
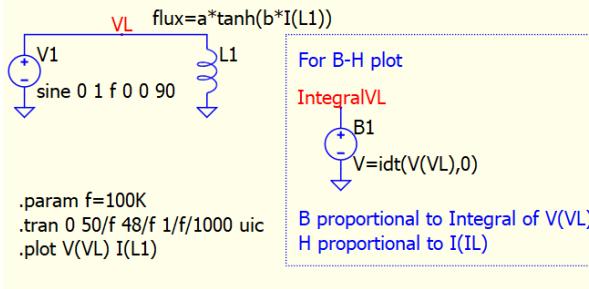
- .ac is used to calculate inductance and bias current relationship
- .tran is used to demonstrate non-linear inductor current and proportional B-H curve
- Flux equation uses format
 - $\text{Flux} = a * \tanh(b * I(L1))$



Flux model with .tran

```

.param b = 0.59
.param L0A = 1μ
.param a = L0A/b
flux=a*tanh(b*I(L1))
  
```



L. Arbitrary Inductance : Inductor modeling with FLUX – Nonlinear with Lmin

Qspice : L Inductor (AC) - Flux eqn 2.qsch ; L Inductor Nonlinear (Tran) - Flux eqn 2.qsch

- Flux Model

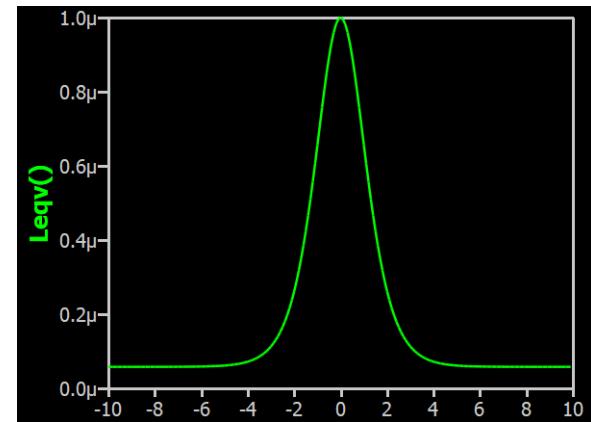
- This is nonlinear flux model include Lmin at large inductor current
- Flux equation uses format
 - Flux = $a \tanh(b \cdot I(L1)) + Lmin \cdot I(L1)$

```

.param b = 0.7
.param LOA = 1μ
.param a = (LOA-Lmin)/b
.param Lmin=58n
flux=a*tanh(b*I(L1))+Lmin*I(L1)

.param frq=1Meg
.step param bias -10 10 0.1 Sweep bias
.step param var list 1 2 3 Sweep Isat

.ac list frq
.func Leqv() V(VL)/1/2/pi/frq
.plot Leqv()
  
```



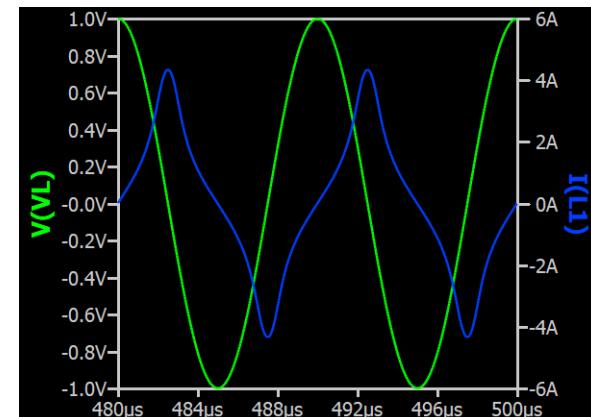
```

.param b = 0.7
.param LOA = 1μ
.param a = (LOA-Lmin)/b
.param Lmin=58n
flux=a*tanh(b*I(L1))+Lmin*I(L1)

.sine V1 0 1 f 0 0 90
.VL

.param f=100K
.tran 0 50/f 48/f 1/f/1000 uic
.plot V(VL) I(L1)

For B-H plot
IntegralVL
B1
V=idt(V(VL),0)
B proportional to Integral of V(VL)
H proportional to I(IL)
  
```

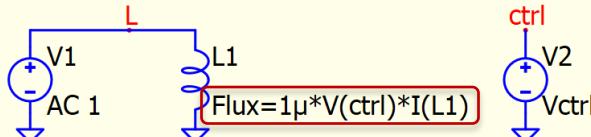


L. Behavioral Inductance with Flux (Arbitrary Inductance)

Qspice : L - Behavioral L with Flux (.ac).qsch | L - Behavioral L with Flux (.tran).qsch

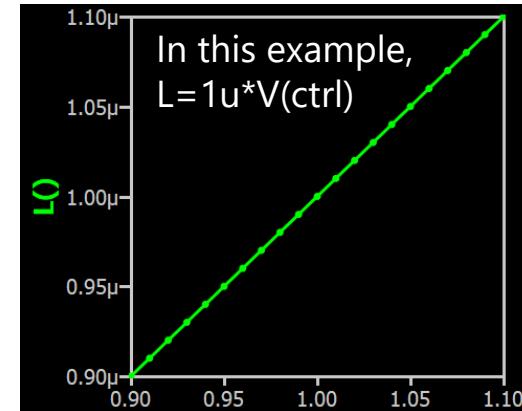
- Behavioral L with Flux ϕ
 - For linear inductor
 - $\phi = L \times I_L$
 - Therefore
 - Flux = <eqn> * I(Ln)
 - <eqn> is inductance equation supports voltage node or device current
 - Ln is name attribute of inductor
 - .ac and .tran example of arbitrary inductance device controlled by external voltage source

Linear Inductor flux formula : Flux = L * IL
Therefore, behavioral inductance model can be derived using Arbitrary Inductance Device

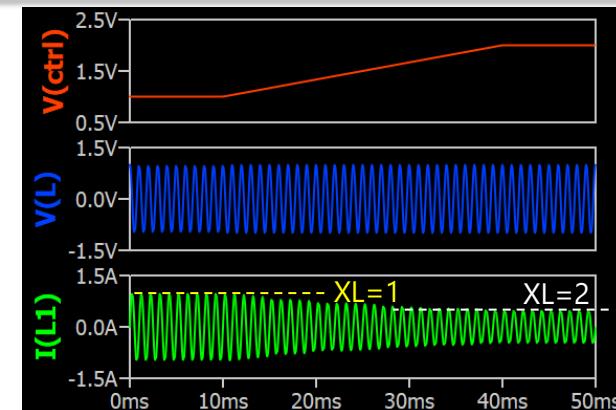
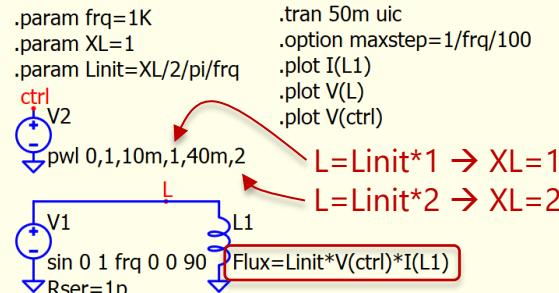


```
.step param Vctrl 0.9 1.1 0.01
.param frq=1Meg
.ac list frq
```

```
.func imZ() V(L)/I(L1)
.func L() imZ()^2/pi/frq
.plot L()
```



Linear Inductor flux formula : Flux = L * IL
Therefore, behavioral inductance model can be derived using Arbitrary Inductance Device



M. MOSFET

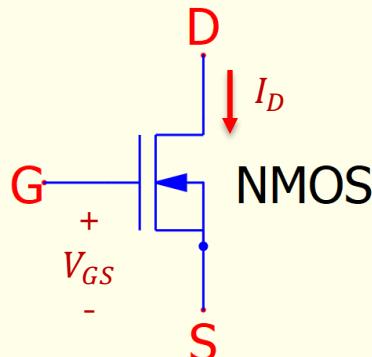
Model Levels for

NMOS | PMOS

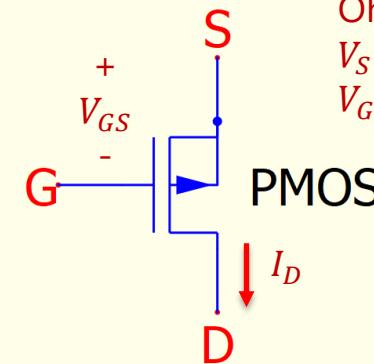
M. MOSFET

- MOSFET Syntax
 - Mnnn D G S B <model> [instance parameters]
 - D : Drain
 - G : Gate
 - S : Source
 - B : Substrate or Body
 - .model <model> <NMOS|PMOS|VDMOS> [model parameters]

M. MOSFET NMOS and PMOS



Ohmic / Saturation
 $V_D > V_S$
 V_{GS} is +ve ($V_G > V_S$)



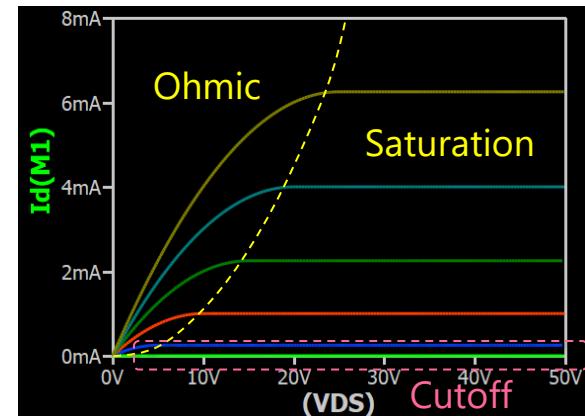
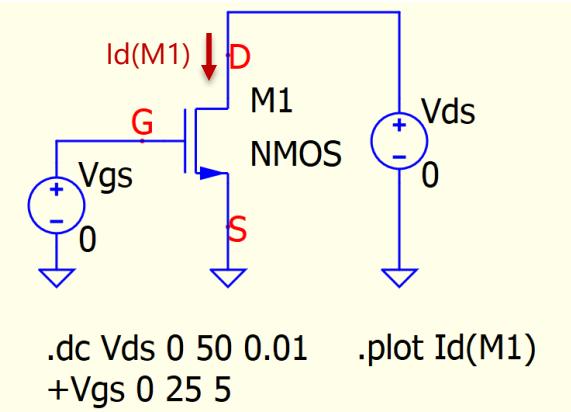
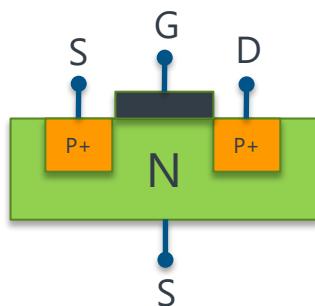
Ohmic / Saturation
 $V_S > V_D$
 V_{GS} is -ve ($V_G < V_S$)

Voltage Relations	NMOS
$V_{GS} > V_{th}$ $V_{DS} < V_{GS} - V_{th}$	Ohmic (Triode/Linear)
$V_{GS} > V_{th}$ $V_{DS} \geq V_{GS} - V_{th}$	Saturation (Active)
$V_{GS} < V_{th}$	Cutoff

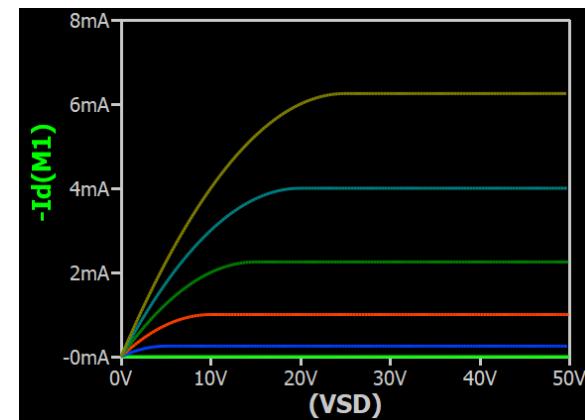
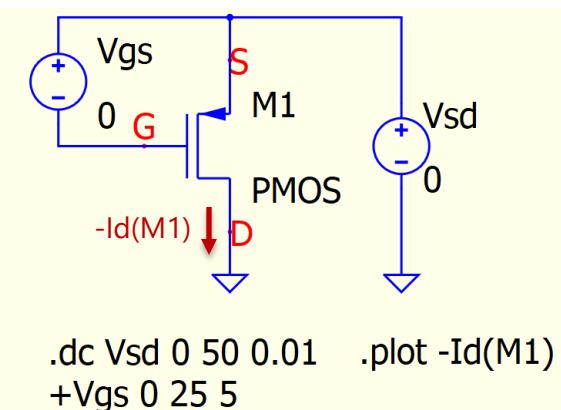
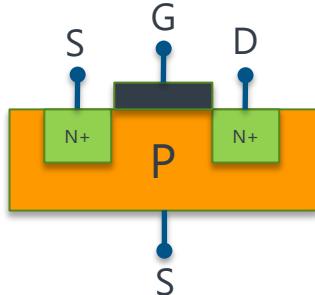
M. MOSFET NMOS and PMOS Ohmic, Saturation and Cutoff

Qspice : NMOS PMOS (Basic) - Region NMOS.qsch | NMOS PMOS (Basic) - Region PMOS.qsch

- NMOS
 - N-channel MOSFET



- PMOS
 - P-channel MOSFET



MOSFET Model Levels for NMOS | PMOS | VDMOS

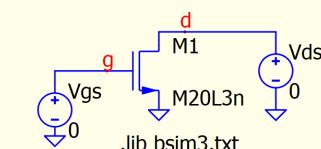
- Model Levels for MOSFET (NMOS | PMOS | VDMOS)
 - Monolithic N- or P- Mosfet : **.model <model> <NMOS|PMOS> Level=X**
 - Parameter Level specifies the model to be used for NMOS | PMOS
 - Vertical Double Diffused Power Mosfet : **.model <model> VDMOS <nchan|pchan>**
 - In default as nchan if not specify

Level	Name	Model	Comment
1	MOS1	Shichman-Hodges	MOS is generic model typically assumes long-channel devices
2	MOS2	Grove-Frohman	VDMOS is extended from MOS1 with additional parameters Cgdmin, Cgdmax, ETA, Ronx
3	MOS3	Semi-empirical model	MOS2 and MOS3 are intermediate and advance model of MOS1
-4	BSIM1	Berkeley Short-Channel IGFET Model	CMOS short-channel IGFET model, initial version, rarely used
-5	BSIM2		level number set negative in Qspice
7, 8, 49,53	BSIM3v3.3.2		Channel length down to 0.25um
14, 54	BSIM4v4.8.1		Channel length below 0.25um
5, 12, 55	Modified EKV v2.6	Enz-Krummenacher-Vittoz	CMOS
2010	SiC FET	Qorvo SiC FET	

NMOS MOS3 and BSIM3 Examples

Qspice : MOS Level=3 - MOS3.qsch | MOS Level=7 - BSIM3.qsch | bsim3.txt

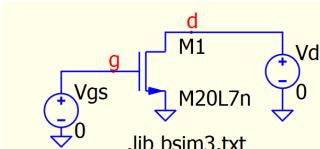
NMOS Level=3 MOS3 Example



```
.dc Vgs -5 5 0.01 Vds list 5  
.plot Id(M1)
```

```
.MODEL M20L3n NMOS LEVEL=3 PHI=0.700000 TOX=3.9100E-08 XJ=0.200000U TPG=1  
+ VTO=0.7909 DELTA=-7.2550E-01 LD=3.6790E-07 KP=5.4712E-05  
+ UO=619.5 THETA=4.2250E-02 RSH=3.2310E+01 GAMMA=0.5350  
+ NSUB=6.7240E+15 NFS=5.9090E+11 VMAX=1.9770E+05 ETA=8.3090E-02  
+ KAPPA=4.1240E-01 CGDO=4.8737E-10 CGSO=4.8737E-10  
+ CGBO=3.4582E-10 CJ=1.3214E-04 MJ=6.0852E-01 CJSW=5.2994E-10  
+ MJSW=2.5789E-01 PB=4.0492E-01
```

NMOS Level=7 BSIM3 Example



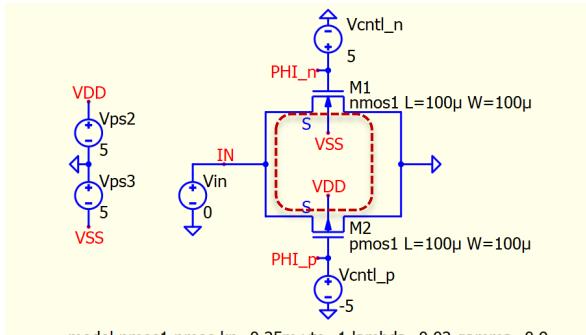
```
.dc Vgs -5 5 0.01 Vds list 5  
.plot Id(M1)
```

```
.MODEL M20L7n NMOS ( LEVEL= 7  
*+VERSION = 3.1  
+ TNOM = 27 TOX = 3.91E-8  
+ NCH = 8E16 VT0 = 0.7582903  
+ K1 = 1.2842394 K2 = -0.2616765 K3 = 1.8658079  
+ K3B = -6.3342185 W0 = 1E-8 NLX = 1E-9  
+ DVT0W = 0 DVT1W = 0 DVT2W = 0  
+ DVT0 = 0.8232689 DVT1 = 0.2798469 DVT2 = -0.3303888  
+ U0 = 686.7089438 UA = 2.30065E-9 UB = 1.887526E-19  
+ UC = 1.641046E-11 VSAT = 1.1522865 A0 = 0.4530999  
+ AGS = 0.143591 BO = 2.190772E-6 B1 = 3.8864578E-6  
+ KETA = -0.0185064 AI = 0 A2 = 1  
+ RDWSW = 915.0226945 PRWG = 1.170682E-4 PRWB = -1.258108E-7  
+ WR = 1 WINT = 1.354538E-7 LINT = 3.631938E-7  
+ *XL = 0 XW = 0  
+  
+ DWB = 8.367884E-8 VOFF = -0.0371577 DWG = -4.259528E-8  
+ CIT = 0 CDSC = 0 NFACTOR = 0.0171325  
+ CDSCD = 0  
+ CDSCB = 2.06298E-5 ETAB = 2.972609E-3 CDSCD = 0  
+ DSUB = 6.9559E-3 PCLM = 4.6075119 ETAB = -6.934953E-4  
+ PDIBLC2 = 1.404855E-6 PDIBLCB = -1E-3 DROUT = 0.5839765  
+ PSCLBE1 = 2.291228E10 FSCBE2 = 6.675847E-7 PVAG = 1.8770406  
+ DELTA = 0.01 MOBMOD = 1 PRT = 0  
+ UTE = -1.5 KT1 = -0.11 KT1L = 0  
+ KT2 = 0.022 UAI = 4.31E-9 UBI = -7.61E-18  
+ UCI = -5.6E-11 AT = 3.3E4 WL = 0  
+ WLN = 1 WW = 0 WNN = 1  
+ NWL = 0 LL = 0 LLN = 1  
+ LW = 0 LWN = 1 LWL = 0  
+ CAPMOD = 2 XPART = 0.4 CGDO = 3.54E-10  
+ CGSO = 3.54E-10 CGBO = 0 CJ = 1.321396E-4  
+ FB = 0.4049196 MJ = 0.6085241 CJSW = 5.299371E-10  
+ PBSW = 0.6552296 MJSW = 0.2578664 PVTH0 = 0.0684739  
+ PRDSW = -1.8E3 PK2 = -0.0812182 WKETA = 0.0246323  
+ LKETA = -4.18493E-3 PAGS = 0.536 )
```

MOSFET model : Substrate or Body (B) Terminal

Qspice : MOSFET-CMOS-Substance (Circuit Example).qsch

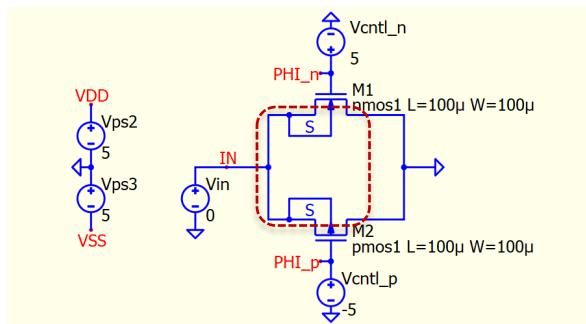
- Substrate or Body (B)
 - MOSFET syntax
 - Mnnn D G S B <model>
 - For Power MOSFET, the B terminal is typically connected to the Source terminal
 - In CMOS, the substrate may be connected to potential different to Source terminal
 - This circuit example demonstrates how the substrate potential can affect the electronic characteristics of this CMOS amplifier
 - Reference :
https://www.ece.mcgill.ca/~grobber4/SPICE/SPICE_Decks/1st_Edition_LTSPICE/chapter5/Chapter%205%20MOSFETs%20web%20version.html



```

.model nmos1 nmos kp=0.25m vto=1 lambda=0.02 gamma=0.9
.model pmos1 pmos kp=0.25m vto=-1 lambda=0.02 gamma=0.9
.dic Vin -5 1.1m
.plot V(IN)/Is(M1) V(IN)/Is(M2) V(IN)/(Is(M1)+Is(M2))

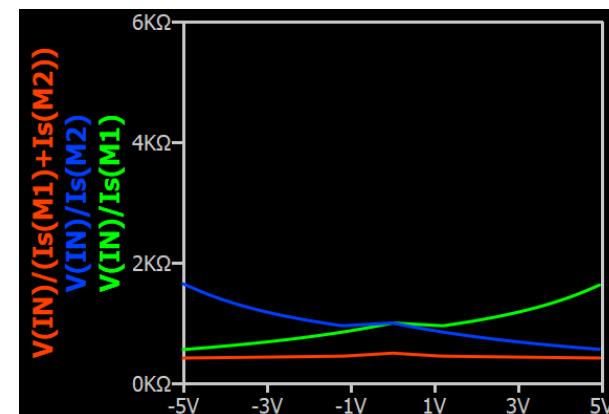
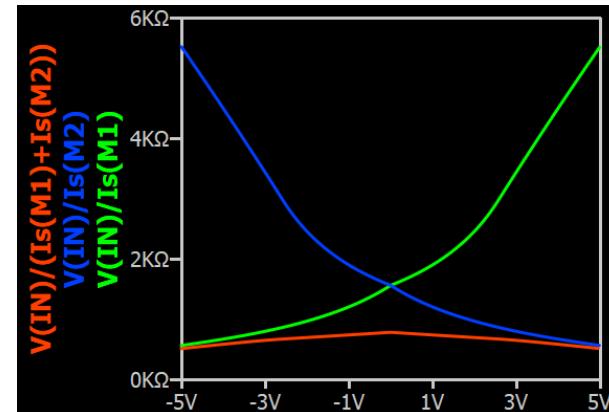
```



```

.model nmos1 nmos kp=0.25m vto=1 lambda=0.02 gamma=0.5
.model pmos1 pmos kp=0.25m vto=-1 lambda=0.02 gamma=0.5
.dc Vin -5 1.1m
.plot V(IN)/Is(M1) V(IN)/Is(M2) V(IN)/(Is(M1)+Is(M2))

```



Low Power IC (Integrated Circuit) MOSFET

- Modeling of Low Power CMOS
 - In Qspice, model is commonly found as : BSIM3 , BSIM4
- Low Power IC (Integrated Circuit) MOSFET
 - NMOS (Monolithic N-channel MOS)
 - PMOS (Monolithic P-channel MOS)
 - CMOS (Complementary Metal-Oxide-Semiconductor) : Employs both n-type and p-type MOSFETs (NMOS and PMOS) to achieve low power consumption and high noise immunity for the integration of digital logic, analog circuits and memory on a single chip
 - SOI (Silicon-On-Insulator MOSFET) : Utilizes a thin layer of silicon on top of an insulating layer to improve device performance and reducing parasitic capacitance for high-frequency performance and reduces cross-talk in mixed signal designs

High-Power MOSFET

- Modeling of High-Power MOSFET
 - In Qspice, model is commonly found as : VDMOS , Level=2010 (SiC) and .SUBCKT
- Common High-Power MOSFET
 - VDMOS (Vertical Double-Diffused MOSFET): Most common high-power MOSFET variant and widely used in a range of applications, including power supplies, motor control, and audio amplifiers
 - LDMOS (Laterally Diffused MOSFET): High-power applications such as RF power amplifiers, wireless communication systems, and automotive systems.
 - Trench MOSFET: Commonly used in power conversion, motor drives, and automotive applications
 - GaN HEMT (Gallium Nitride High Electron Mobility Transistor): High-frequency and high-power applications such as RF power amplifiers, radar systems, and wireless power transfer.
 - SiC MOSFET (Silicon Carbide MOSFET): High-power and high-temperature applications due to their excellent performance, particularly in electric vehicles, renewable energy systems, and industrial power electronics
 - DMOS (Double-Diffused MOSFET)
 - UMOS (Unipolar MOSFET)

M. MOSFET (VDMOS)

**** VDMOS with different
default as monolithic MOSFET
models, it popularly used in
board level SMPS**

MOSFET Level 1 (MOS1) Model Parameters

MOSFET Level 1 Model Parameters

Name	Description	Units	Default
A	Additional non-linear Gate-drain capacitance abruptness		1.0
AD	Default drain area	m ²	0.0
AF	Flicker noise exponent		1.0
AS	Default source area	m ²	0.0
BV	Body diode breakdown voltage	V	Infinite
CAPOP ⁹	Charge(Capacitance) model		1
CBD	Zero-bias B-D junction capacitance	F	0.0
CBS	B-S junction capacitance	F	0.0
CGBO	Gate-bulk overlap capacitance	F/m	0.0
CGDMAX ⁸	Maximum additional non-linear Gate-Drain capacitance	F	0.0
CGDMIN ⁸	Minimum additional non-linear Gate-Drain capacitance	F	0.0
CGDO	Gate-drain overlap capacitance	F/m	0.0
CGS	Gate-source capacitance that doesn't scale with width	F	0.0
CGSO	Gate-source overlap capacitance	F/m	0.0
CJ	Bottom junction cap per area	F/m ²	0.0
CJO	Body diode zero-bias capacitance	F	0.0
CJSW	Side junction cap per length	F/m	0.0
EG	Body diode activation energy	V	1.11
ETA ⁸	Philips, et al.-style subthreshold conduction parameter		0.0
ETA2	Inverse sharpness of Ids limit for non-square law variation		0.0
FC	Forward bias junction fit parameter		0.5
GAMMA	Bulk threshold parameter	V ^{1/2}	0.0
GDSNOI	Channel shot noise coefficient(nlev=3)		1.0
GM	Ideal transconductance for non-square law variation	Ω	none
GMAX	Maximum bulk PN conductivity	Ω	1000.
IBV	Body diode current at breakdown voltage	A	1e-10
IDS	Maximum drain current for non-square law variation	A	Infinite
IGSS	Gate-Source leakage current	A	0.0
IS	Bulk junction saturation current	A	1e-14
JS	Bulk junction saturation current density	A/m ²	0.0
K1 ¹⁰	Threshold voltage sensitivity to bulk node	V ^{1/2}	0.25
KF	Flicker noise coefficient		0.0
KP	Transconductance parameter	A/V ²	2e-5
L	Default channel length	m	1e-5
LAMBDA	Channel length modulation	V ⁻¹	0.0

LD	Lateral diffusion	m	0.0
MJ	Bulk grading coefficient		0.5
MJSW	Side grading coefficient		0.33
N	Bulk(or Body) PN junction emission coefficient		1.0
NBV	Body diode breakdown emission coefficient		1.0
NLEV	Noise level(equation selector)		0
NRD	Default drain squares		0.0
NRB	Default bulk squares		0.0
NRG	Default gate squares		0.0
NRS	Default source squares		0.0
NSS	Surface state density	cm ⁻²	0.0
NSUB	Substrate doping	cm ⁻²	0.0
PB	Bulk junction potential	V	1.0
PD	Default drain perimeter	m	0.0
PHI	Surface potential	V	1.0
PCHAN	Flag to specify vertical PMOS geometry		not set
PS	Default source perimeter	m	0.0
RB	Bulk resistance	Ω	0.0
RD	Drain resistance	Ω	0.0
RDS	Additional Drain-Source shunt resistance	Ω	Infinite
RG	Gate resistance	Ω	0.0
RON	Ron at zero drain current for non-square law variation	Ω	none
RONX ⁸	Channel conductivity multiplier in linear region		1.0
RS	Source resistance	Ω	0.0
RSH	Sheet resistance	Ω/□	0.0
TNOM	Parameter measurement temperature(aka TREF)	°C	27.0
TOX	Oxide thickness	m	1.5e-8
TPG	Gate type		1
TT	Bulk PN junction Transit-time	sec	0.0
U0	Surface mobility(aka U0)	cm ² /V×sec	600
VDMOS ⁸	Flag to specify vertical geometry		not set
VFB ¹⁰	Flat band voltage	V	-1.0
VIGSS	Voltage at which IGSS was measured	V	5.0
VJ	Body diode Junction potential	V	1.0
VT0	Threshold voltage(aka VTO)	V	0.0
W	Default channel width	m	1e-5
XPART ¹⁰	Channel charge partitioning		1
XTI	Body diode Saturation current temperature exponent		3

Display purpose only

- Vds : VDS rating
 - Ids : IDS rating
 - Qq : Gate charge
 - Mfg : manufacturer
 - Ron : On Resistance
- * It no longer for display purpose if both GM and Ron are set positive, for non-square law variation

Linear and Saturation Region : VTO, KP, W, L, LD, LAMBDA

- From Semiconductor Device Modeling with Spice (Section 4.2.3)

- Linear region : $V_{GS} > V_{TH}$ and $V_{DS} < V_{GS} - V_{TH}$
 - $I_{DS} = \frac{KP}{L-2X_{jl}} \left(V_{GS} - V_{TH} - \frac{V_{DS}}{2} \right) V_{DS} (1 + \lambda V_{DS})$
- Saturation region : $V_{GS} > V_{TH}$ and $V_{DS} > V_{GS} - V_{TH}$
 - $I_{DS} = \frac{KP}{2} \frac{W}{L-2X_{jl}} (V_{GS} - V_{TH})^2 (1 + \lambda V_{DS})$
- where
 - V_{TH} : **VTO**, zero-bias threshold voltage (Default VTO=0 V)
 - KP : **KP**, transconductance (Default KP=1 A/V²)
 - W : **W**, channel width (Default W=1e-4 m)
 - L : **L**, channel length (Default L=1e-4 m)
 - X_{jl} : **LD**, lateral diffusion (Default LD=0 m)
 - λ : **LAMBDA** channel-length modulation (Default LAMBDA=0 V⁻¹)

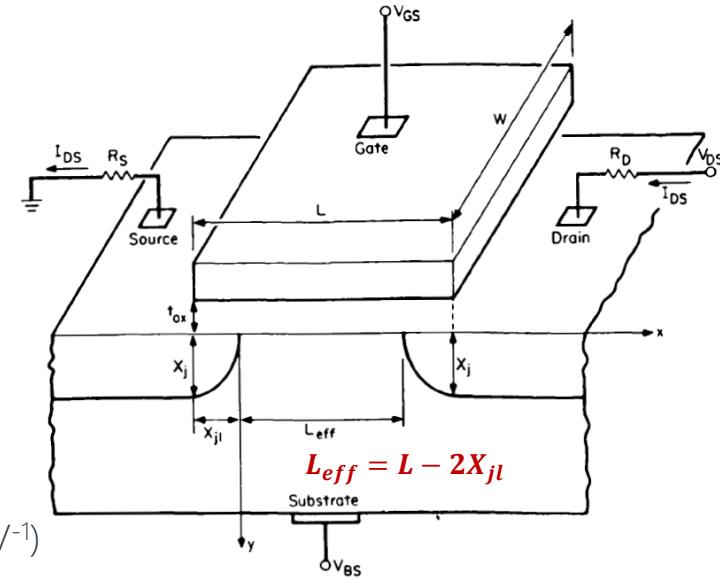


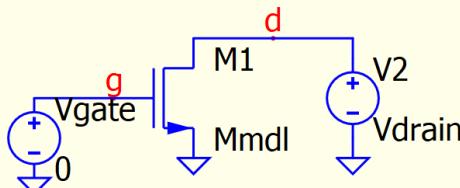
Figure 4-1 Structure of the MOST.

M. MOSFET Params : VTO, KP

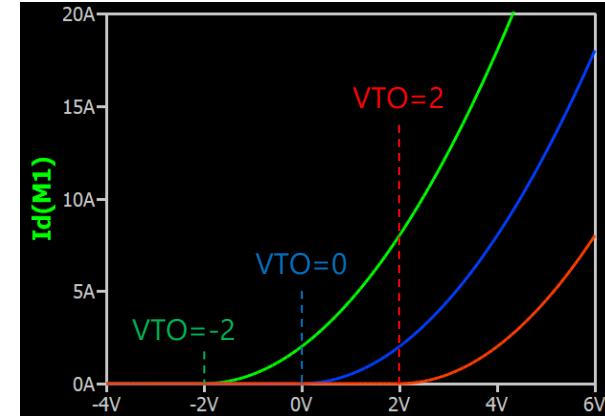
Qspice : VDMOS - VTO.qsch | VDMOS - KP.qsch

- VTO
 - VTO : Zero-bias threshold voltage
 - **Default VTO=0V**

```
.param Vdrain=10  
.dc Vgate -4 6 0.1 .plot Id(M1)
```

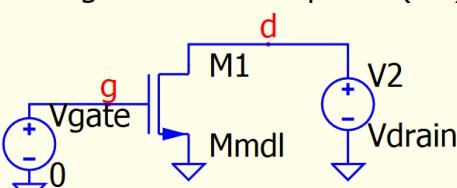


```
.step param vto list -2 0 2  
.model Mmdl VDMOS VTO=vto
```

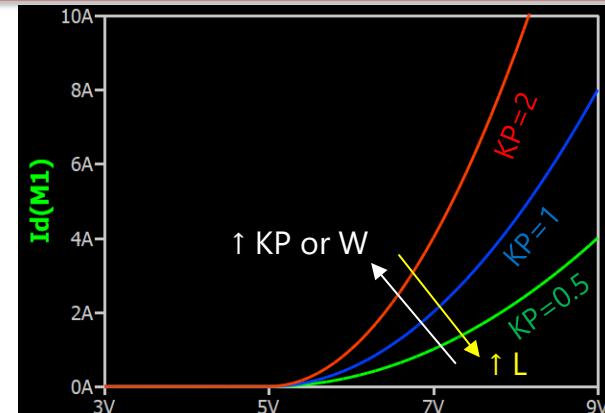


- KP
 - KP : transconductance
 - **Default KP=1 A/V²**
 - Factor : $\frac{KP}{2} \frac{W}{L-2X_{jl}}$
(for saturation region)

```
.param Vdrain=10  
.dc Vgate 3 9 0.1 .plot Id(M1)
```



```
.step param kp list 0.5 1 2  
.model Mmdl VDMOS VTO=5 KP=kp
```

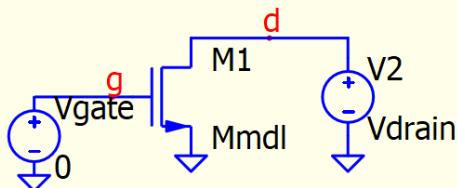


M. MOSFET Params : W and L

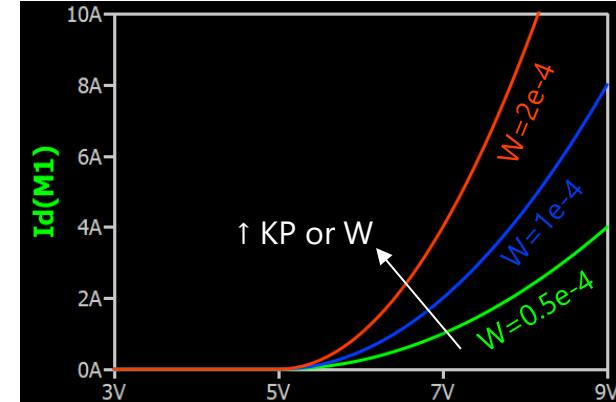
Qspice : VDMOS - W.qsch | VDMOS - L.qsch

- W and L
 - W : Channel Width
 - Default value in
.option DEFW=<value>
with default = 1e-4
(100um)
 - L : Channel Length
 - Default value in
.option DEFLEN=<value>
with default = 1e-4
(100um)
 - Effect is similar to KP Factor : $\frac{KP}{\frac{W}{2} L - 2X_{jl}}$
(for saturation region)

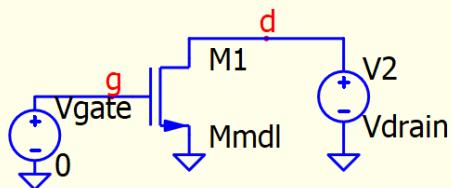
```
.param Vdrain=10  
.dc Vgate 3 9 0.1 .plot Id(M1)
```



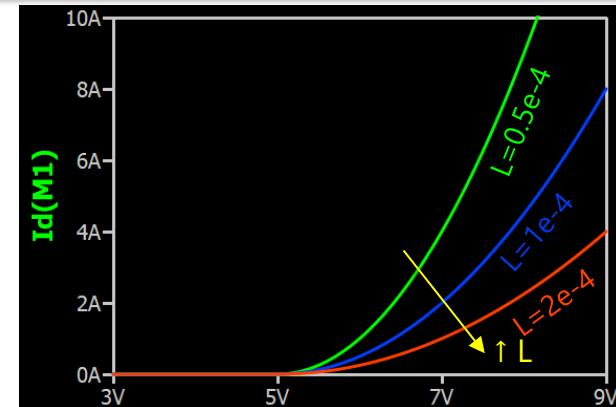
```
.step param w list 0.5e-4 1e-4 2e-4  
.model Mmdl VDMOS VTO=5 W=w
```



```
.param Vdrain=10  
.dc Vgate 3 9 0.1 .plot Id(M1)
```



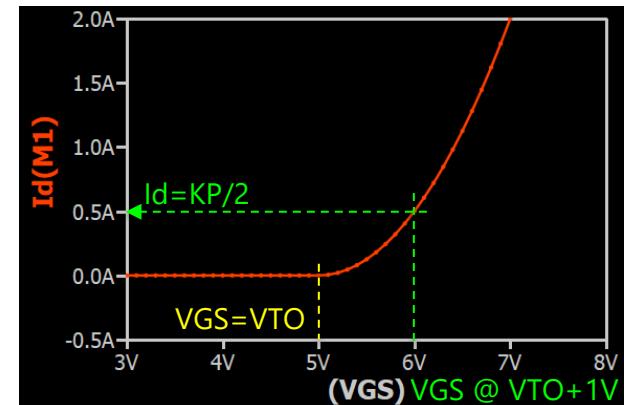
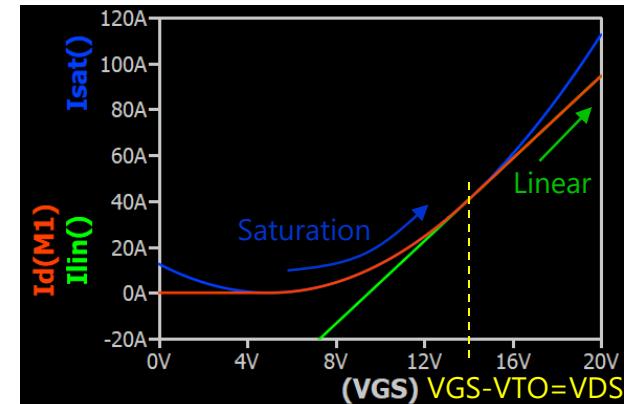
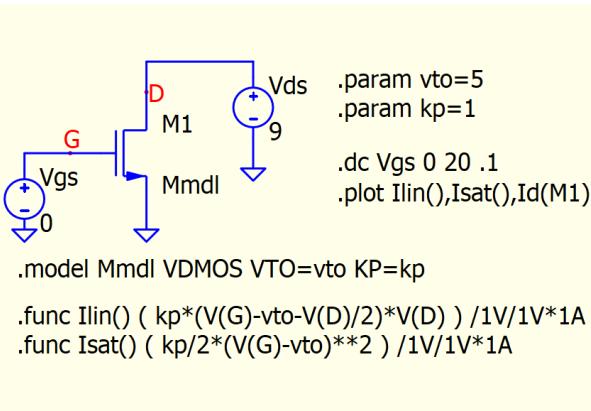
```
.step param l list 0.5e-4 1e-4 2e-4  
.model Mmdl VDMOS VTO=5 L=l
```



M. MOSFET Params : VTO, KP only (textbook modeling)

Qspice : MOSFET - VTO KP only.qsch

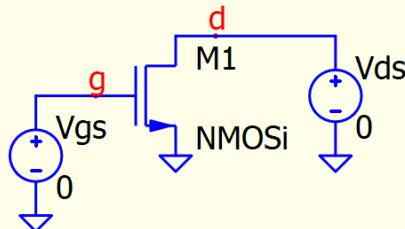
- VTO and KP only
 - A model only with user defined VTO and KP
 - Where W=L, LD=0 and Lambda=0
 - Saturation Region
 - $I_{DS} = \frac{KP}{2}(V_{GS} - VTO)^2$
 - $\therefore \frac{dI_{DS}}{dV_{GS}} = KP \times (V_{GS} - VTO)$
 - Linear Region
 - $I_{DS} = KP \left(V_{GS} - VTO - \frac{V_{DS}}{2} \right) V_{DS}$
 - $\therefore \frac{dI_{DS}}{dV_{GS}} = KP \times V_{DS}$
 - The transition between Linear and Saturation region at $V_{GS} - VTO = V_{DS}$
 - In this example, VTO=5V and VDS=9V, therefore, transition at VGS=14V



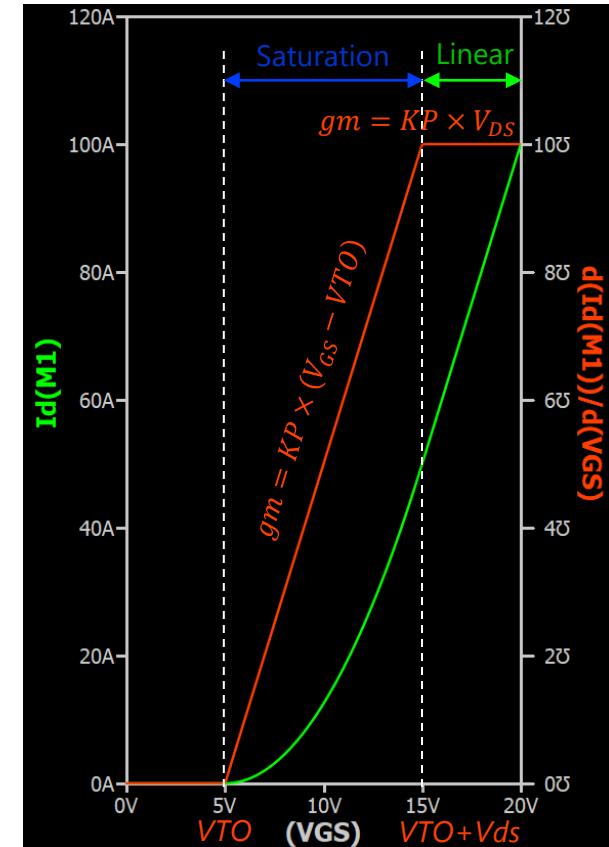
M. MOSFET Params : VTO, KP only – gm (textbook modeling)

Qspice : VDMOS - VTO KP only (gm).qsch

- VTO and KP only (gm)
 - Transconductance (gm) is defined by $gm = \frac{\Delta I_d}{\Delta V_{gs}}$
 - Saturation Region
 - $I_{DS} = \frac{KP}{2}(V_{GS} - VTO)^2$
 - $\therefore gm = \frac{dI_{DS}}{dV_{GS}} = KP \times (V_{GS} - VTO)$
 - Linear Region
 - $I_{DS} = KP \left(V_{GS} - VTO - \frac{V_{DS}}{2} \right) V_{DS}$
 - $\therefore gm = \frac{dI_{DS}}{dV_{GS}} = KP \times V_{DS}$



```
.model NMOSi VDMOS VTO=5 KP=1
.dc Vgs 0 20 0.01 Vds list 10
.plot Id(M1) d(Id(M1))/d(VGS)
```



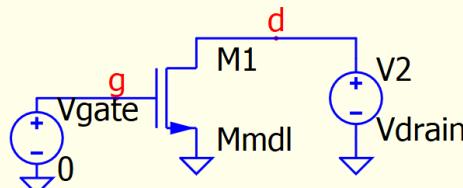
M. MOSFET Params : LD and ETA

Qspice : VDMOS - LD.qsch / VDMOS - ETA.qsch

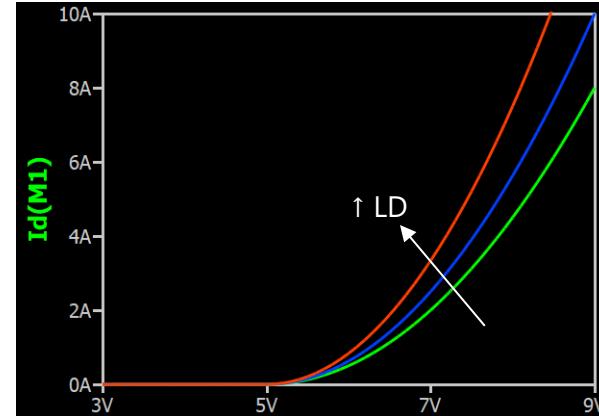
- LD

- LD : lateral diffusion
- **Default LD=0 m**
- $L_{eff} = L - 2LD > 0$
 - Therefore, $0 < LD < \frac{L}{2}$

```
.param Vdrain=10  
.dc Vgate 3 9 0.1 .plot Id(M1)
```



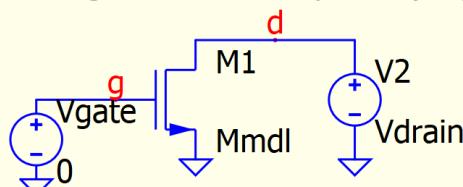
```
.step param Id list 0 0.1e-4 0.2e-4  
.model Mmdl VDMOS VTO=5 LD=Id
```



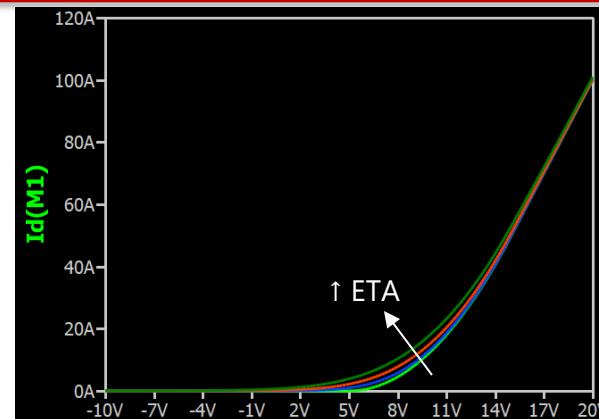
- ETA (aka ksubthres)

- ETA : V_{ds} dependence of threshold voltage
- Static feedback on threshold voltage
- **Default ETA=0**
- Qspice accepts parameter name as ETA or ksubthres

```
.param Vdrain=10  
.dc Vgate -10 20 0.1.plot Id(M1)
```



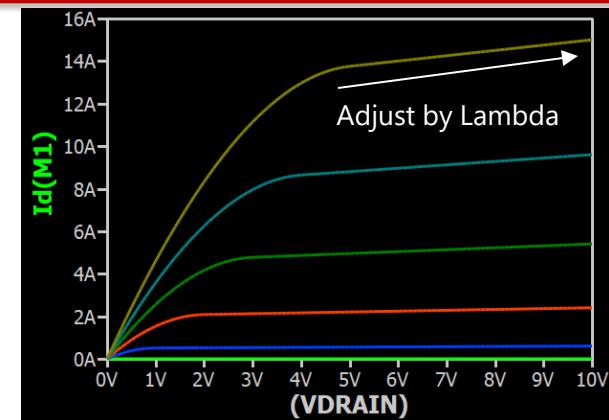
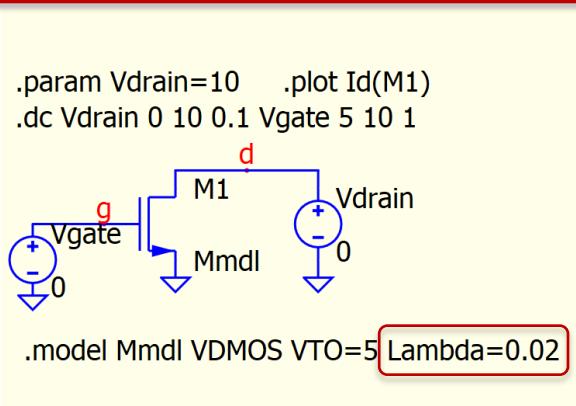
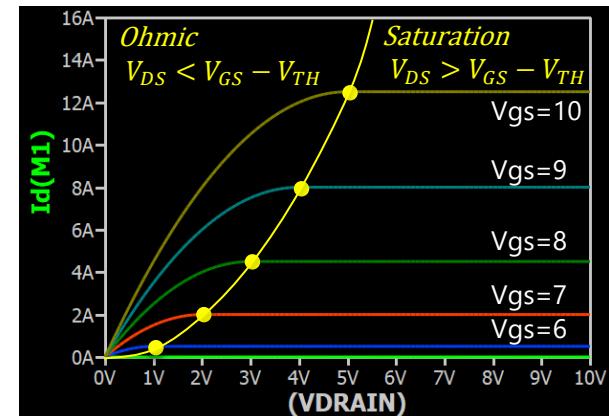
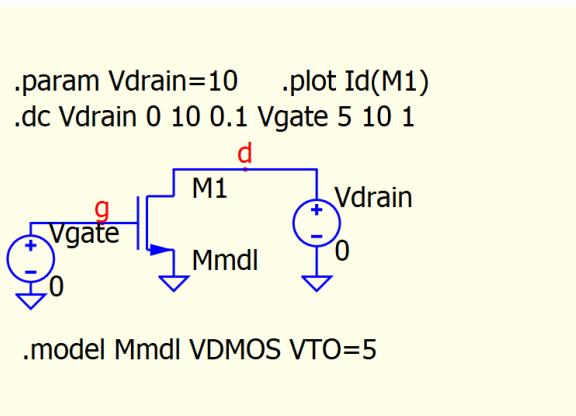
```
.step param eta list 0 2 3 4  
.model Mmdl VDMOS VTO=5 ETA=eta
```



M. MOSFET Params : Lambda

Qspice : VDMOS -LAMBDA.qsch

- LAMBDA
 - Labmda : channel-length modulation
 - Default LAMBDA=0 V⁻¹**

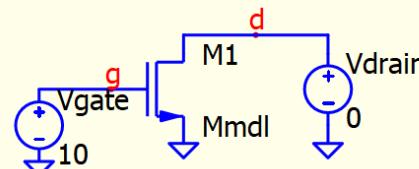


M. MOSFET Params : RONX

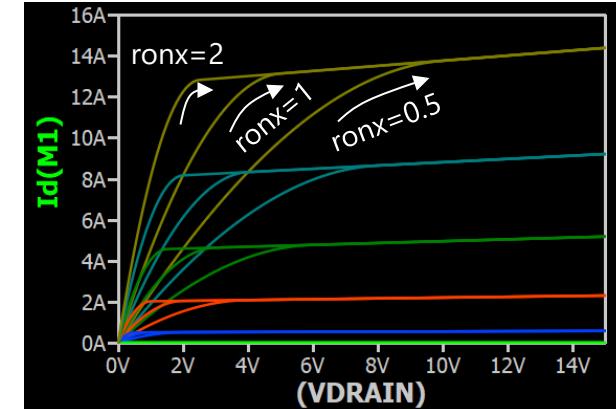
Qspice : VDMOS - RONX.qsch

- RONX
 - Ronx : Channel conductivity multiplier in linear region
 - Default Ronx=1**

```
.param Vdrain=10      .plot Id(M1)
.dc Vdrain 0 15 0.1 Vgate 5 10 1
```



```
.step param ronx list 0.5 1 2
.model Mmdl VDMOS VTO=5
+Lambda=0.01 RONX=ronx
```



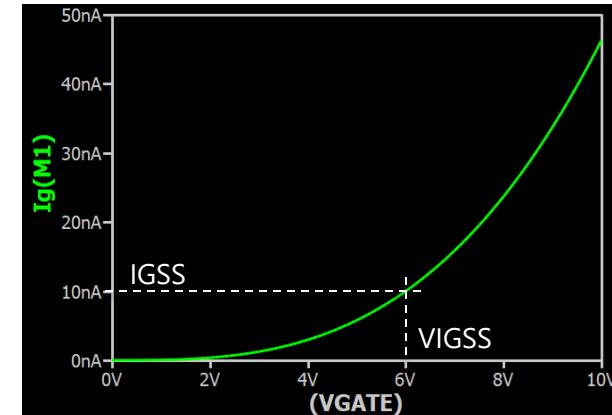
M. MOSFET Params (Gate) : VIGSS and IGSS, RG and CGS

Qspice : VDMOS - VIGSS IGSS.qsch / VDMOS - Rg Cgs.qsch

VIGSS and IGSS

- Vigss : Voltage at which IGSS was measured
- Igss : Gate-Source leakage current
- **Default VIGSS=5V**
- **Default IGSS=0A**
- Parameters to determine Igss in relates to Vgs

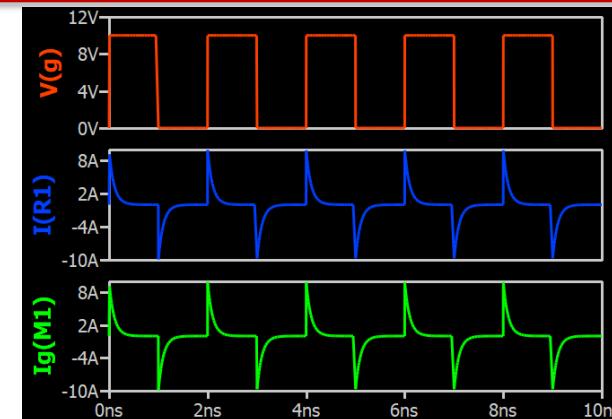
```
.param Vdrain=10  
.dc Vgate 0 10 0.1 .plot Ig(M1)  
  
M1  
Vgate  
V2  
Vdrain  
  
.model Mmdl VDMOS VTO=5 VIGSS=6 IGSS=10n
```



RG and CGS

- Rg : Gate Resistance
- Cgs : Gate-source capacitance that doesn't scale with width
- **Default RG=0Ω**
- **Default CGS=0F**
- Parameters to determine gate resistance and capacitance

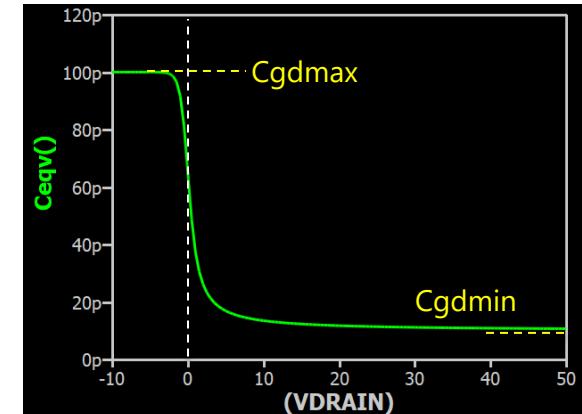
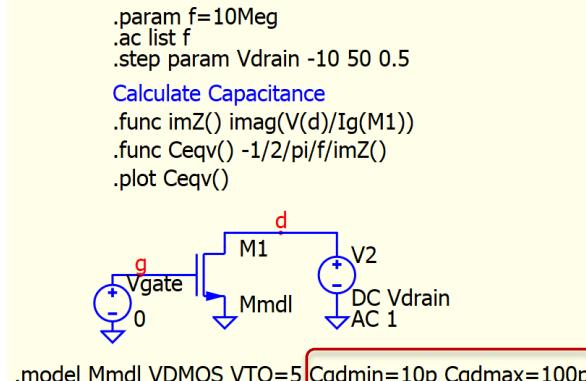
```
Equivalent Rg and Cgs  
R1  
C1  
100p  
d  
M1  
Mmdl  
V2  
Vdrain  
  
pulse 0 10 0 0 0 1n 2n  
.model Mmdl VDMOS VTO=5 Rg=1 Cgs=100p
```



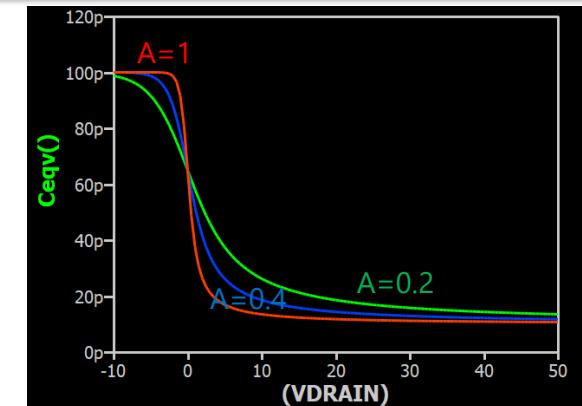
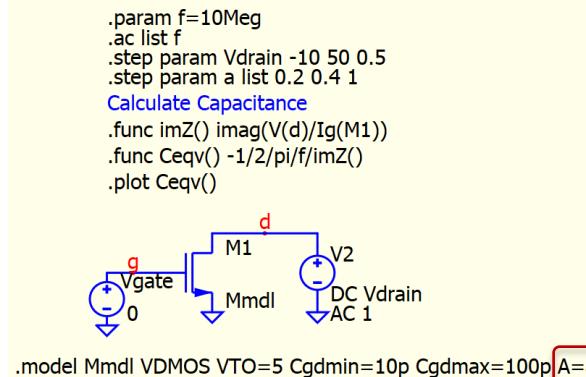
M. MOSFET Params (Cgd) : CGDMAX, CGDMIN and A

Qspice : VDMOS - CGDMIN CGDMAX.qsch / VDMOS - A.qsch

- CGDMAX and CGDMIN
 - Cgdmax : Maximum additional non-linear Gate-Drain capacitance
 - Cgdmin : Minimum additional non-linear Gate-Drain capacitance
 - Default Cgdmax=0F**
 - Default Cgdmin=0F**
 - Model non-linear of Cgd capacitance



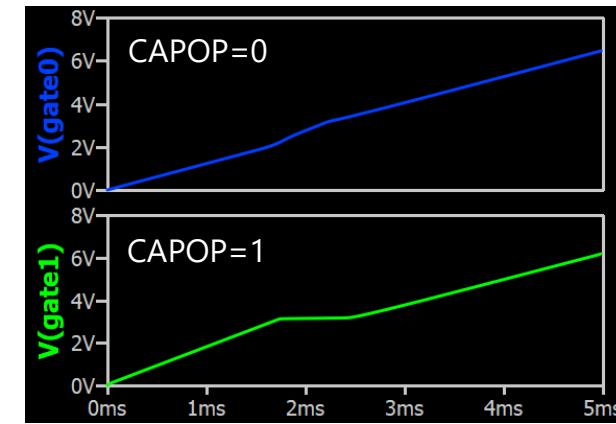
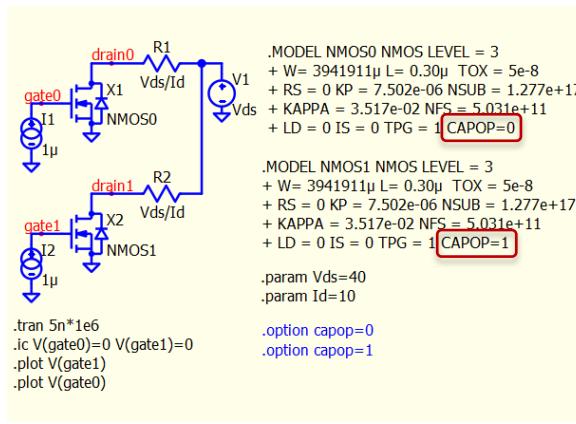
- A
 - A : Additional non-linear Gate-drain capacitance abruptness
 - Default A=1**
 - Modify Cgd vs drain voltage profile



M. MOSFET Params (Gate Charge Model) : CAPOP

Qspice : VDMOS - CAPOP.qsch

- CAPOP
 - Charge (Capacitance) model
 - **Default CAPOP=0**
 - CAPOP=0 is SPICE Meyer Capacitance Model
 - CAPOP=1 is BSIM1 Charge Model
 - For LTspice/Pspice NMOS model
 - They use the BSIM1 charge model. If the gate charge profile deviates from expectations, add CAPOP=1 or use .option CAPOP=1 to override the NMOS model.



- **Mike Engelhardt Comment about Gate Charge Model**

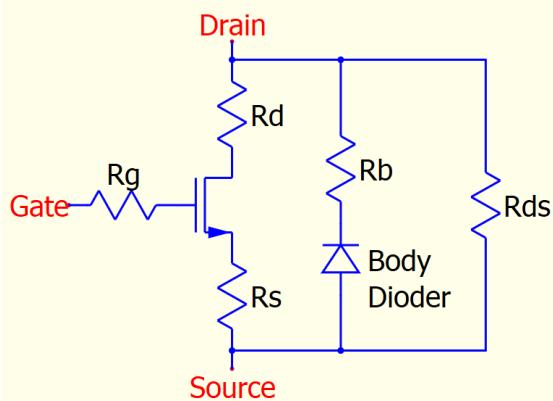
The deal is that the BSIM1 model got popular because it conserved charge. However, it was usually only a contrived circuit that would show that the Meyer model didn't conserve charge, so it really wasn't usually important. All the same, it became a fashion statement to have charge-conserving model, since it took an advanced SPICE programmer to change the charge model in the source code. There was a time when most SPICE vendors couldn't do it. However, it turned out that the Meyer model was often more like the true behavior. While LTspice and PSpice jumped onto the BSIM1 fashion statement, other SPICE programs, like HSPICE, supported both.

M. MOSFET Params : R_b, R_d, R_{ds}, R_g, R_s, Body Diode

Qspice : VDMOS - BodeDiode.qsch / VDMOS - R_b R_d R_{ds} R_g R_s.qsch

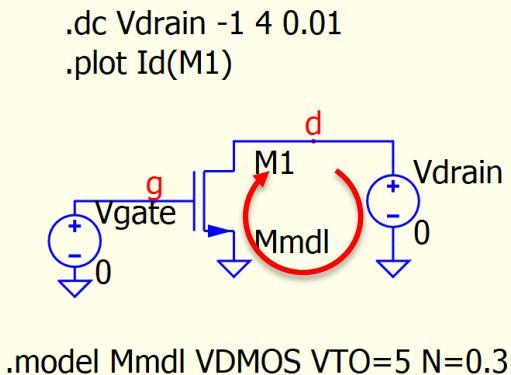
- Resistance

- R_b : Bulk resistance
- R_d : Drain resistance
- R_{ds} : Additional Drain-Source shunt resistance
- R_g : Gate resistance
- R_s : Source resistance
- **Default R_b=R_d=R_g=R_s=0Ω**
- **Default R_{ds}=Infinite**



- Body Diode

- Parameters same as diode model parameters
- Refer to section : D. Diode Diode Model Parameters
- This ideal diode also provide C_{ds} capacitance with its junction capacitance C_j



M. MOSFET : Drain-Source Turn ON Resistance : R_{ds,on}

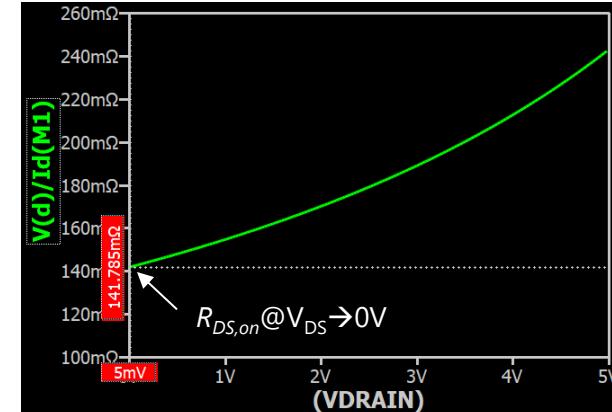
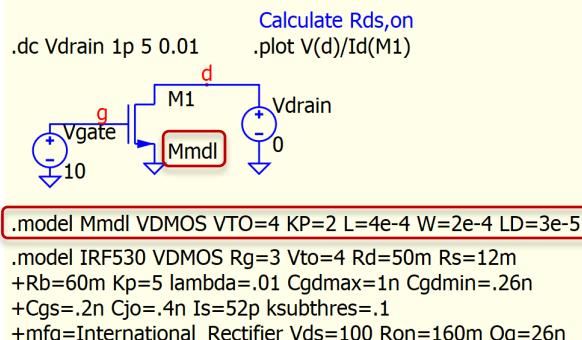
- R_{ds,on}
 - This is not a direct parameters, but as to design SMPS, you may want to know what parameters in MOSFET model affect R_{ds,on}
 - In SMPS, R_{ds,on} is generally considered as MOSFET fully turn ON, where
 - $V_{GS} \gg V_{TH}$ and V_{DS} is low, MOSFET in linear region, and $R_{DS,ON} = \frac{V_{DS}}{I_{DS}}$
 - Linear region formula : $V_{GS} > V_{TH}$ and $V_{DS} < V_{GS} - V_{TH}$
 - $I_{DS} = KP \frac{W}{L-2X_{jl}} \left(V_{GS} - V_{TH} - \frac{V_{DS}}{2} \right) V_{DS} (1 + \lambda V_{DS})$
 - Therefore
 - $R_{DS,ON} = \frac{V_{DS}}{I_{DS}} = \frac{V_{DS}}{KP \frac{W}{L-2X_{jl}} \left(V_{GS} - V_{TH} - \frac{V_{DS}}{2} \right) V_{DS} (1 + \lambda V_{DS})} = \frac{1}{KP \frac{W}{L-2X_{jl}} \left(V_{GS} - V_{TH} - \frac{V_{DS}}{2} \right) (1 + \lambda V_{DS})}$
 - $\lim_{V_{DS} \rightarrow 0} R_{DS,ON} = \frac{1}{KP \frac{W}{L-2X_{jl}} \left(V_{GS} - V_{TH} - \frac{0}{2} \right) (1 + \lambda \times 0)} = \frac{1}{KP \frac{W}{L-2X_{jl}} (V_{GS} - V_{TH})} = \frac{L-2X_{jl}}{KP \cdot W \cdot (V_{GS} - V_{TH})}$
 - With Qspice model parameters, also consider drain and source resistance in model
 - @ $V_{DS} \rightarrow 0V$: $R_{DS,ON} = \frac{L-2 LD}{KP \cdot W \cdot (V_{GS} - VTO)} + RD + RS$
 - Where in default, VTO=0, KP=1, L=1e-4, W=1e-4, LD=0, RD=0 and RS=0

M. MOSFET : Drain-Source Turn ON Resistance : R_{ds,on}

Qspice : VDMOS - (Calculate) Rdson.qsch

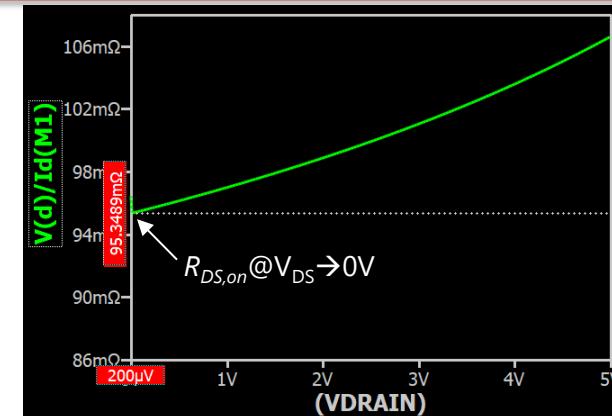
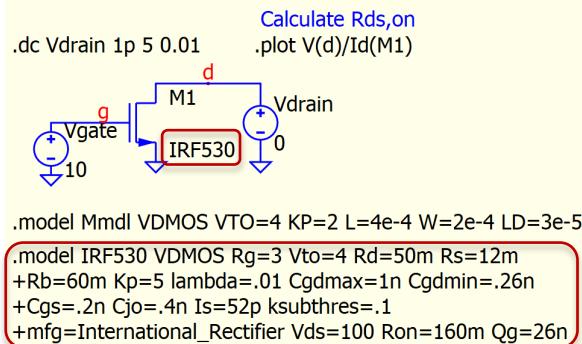
- Example #1

- Mmdl model
- @V_{DS} → 0V
- $R_{DS,ON} = \frac{L-2 LD}{KP \cdot W \cdot (V_{GS}-VTO)} + RD + RS$
 - RD=0
 - RS=0
- $R_{DS,ON} @ V_{DS} \rightarrow 0V = 141.67m\Omega$



- Example #2

- Mmdl model
- @V_{DS} → 0V
- $R_{DS,ON} = \frac{L-2 LD}{KP \cdot W \cdot (V_{GS}-VTO)} + RD + RS$
 - VTO=4, KP=5
 - L=W=1e-4, LD=0
 - RD=50m
 - RS=12m
- $R_{DS,ON} @ V_{DS} \rightarrow 0V = 95.33m\Omega$



M. MOSFET
Level 2010 (SiC FET)
Model

M. MOSFET Level 2010 (SiC FET) Model

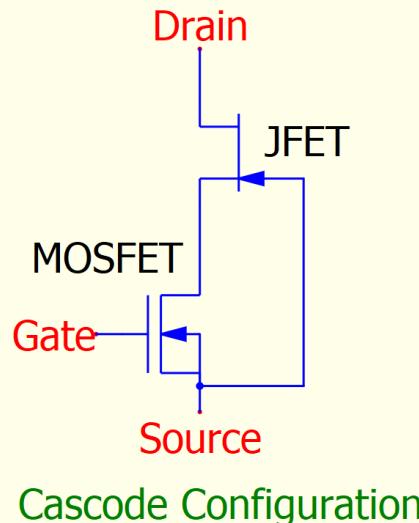
Name	Description	Units	Default
A	Sharpness of transition from CGDMIN to CGDMAX		1.0
AF	MOSFET Flicker noise exponent		1.0
BV	Body diode breakdown voltage	V	Infinite
CGDMAX	Gate-Drain maximum capacitance	F	0.0
CGDMIN	Gate-Drain minimum capacitance	F	0.0
CGS	Gate-source overlap capacitance	F	0.0
CJO	Body diode zero bias cap	F	0.0
EG	Body diode activation energy	V	1.11
ETA	Philips, et al. style subthreshold conduction	V	0.0
ETATC1	ETA first order tempco	V/ $^{\circ}$ C ⁻¹	0.0
ETATC2	ETA second order tempco	V/ $^{\circ}$ C ⁻²	0.0
GDSNOI	Noise equation selector		0
GMAX	Maximum conductivity of any PN junction	Ω	1000.
IBV	Body diode current at breakdown voltage	A	1e-10
IGSS	Gate-Source leakage	A	0.0
IS	Body diode saturation current	A	1e-14
JAF	JFET Flicker Noise Exponent		1.0
JALPHA	JFET impact ionization coefficient	V ⁻¹	0.0
JB	JFET doping tail parameter		0.0
JBETA	JFET transconductance parameter	A/V ²	1.0
JBETATCE	JFET exponential temperature coefficient	%/ $^{\circ}$ C	0.0
JCGD	JFET G-D junction cap	F	0.0
JCGDO	JFET G-D junction cap	F	0.0
JCGS	JFET G-S junction capacitance	F	0.0
JCGSO	JFET G-S junction capacitance	F	0.0
JFC	JFET forward bias junction fit parm.		0.5
JIS	JFET gate junction saturation current	A	1e-14
JISR	JFET recombination current parameter	A	0.0
JKF	JFET Flicker Noise Coefficient		0.0
JLAMBDA	JFET Channel length modulation parameter	V ⁻¹	0.0

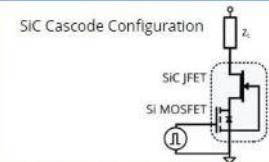
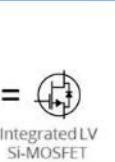
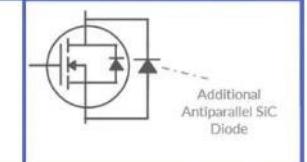
JM	JFET grading coeff.		0.5
JN	JFET emission coefficient		1.0
JNR	JFET Is _s emission coefficient		2.0
JPB	JFET gate junction potential	V	1.0
JRD	JFET drain ohmic resistance	Ω	0.0
JRDS	JFET additional Drain-Source leakage resistance	Ω	Infinite
JRDSTC1	JFET RDS first order tempco	$^{\circ}$ C ⁻¹	0.0
JRDSTC2	JFET RDS second order tempco	$^{\circ}$ C ⁻²	0.0
JRDTCl	JFET RD first order tempco	$^{\circ}$ C ⁻¹	0.0
JRDTC2	JFET RD second order tempco	$^{\circ}$ C ⁻²	0.0
JRG	JFET gate ohmic resistance	Ω	0.0
JRGTC1	JFET RG first order tempco	$^{\circ}$ C ⁻¹	0.0
JRGTC2	JFET RG second order tempco	$^{\circ}$ C ⁻²	0.0
JRONX	JFET channel conductivity multiplier in linear region		1.0
JRS	JFET source ohmic resistance	Ω	0.0
JRSTC1	JFET RS first order tempco	$^{\circ}$ C ⁻¹	0.0
JRSTC2	JFET RS second order tempco	$^{\circ}$ C ⁻²	0.0
JVK	JFET ionization knee current	V	0.0
JVTO	JFET threshold voltage	V	-2.0
JVTOTC	JFET V _{t0} 's temperature coefficient	$^{\circ}$ C ⁻¹	0.0
JXTI	JFET IS temperature coefficient		3.0
KF	MOSFET flicker noise coefficient		0.0
KP	MOSFET transconductance parameter	A/V ²	1.0
LAMBDA	MOSFET channel length modulation	V ⁻¹	0.0
LG	JFET gate inductance	H	0.0
LGRPAR	JFET gate inductance parallel loss	Ω	Infinite
LMID	Inductance making cascode connection	H	0.0
LS	MOSFET Source Inductance	H	0.0
LSRPAR	MOSFET Source inductance parallel loss	Ω	Infinite
LMIDRPAR	Cascode inductance parallel loss	Ω	Infinite
M	Body diode grading coefficient(MJ)	Ω	0.5

N	Bulk diode emission coefficient		1.0
NBV	Body diode breakdown emission coeff		1.0
NLEV	MOSFET noise equation selecctor		0
NOISELESS	The device does not contribute to a noise		
RB	Body diode serial resistance	Ω	0.0
RBTC1	MOSFET RB first order tempco	$^{\circ}$ C ⁻¹	0.0
RBTC2	MOSFET RB second order tempco	$^{\circ}$ C ⁻²	0.0
RD	MOSFET drain ohmic resistance	Ω	Infinite
RDS	MOSFET drain-source shunt resistance	Ω	
RDTC1	MOSFET RD first order tempco	$^{\circ}$ C ⁻¹	0.0
RDTC2	MOSFET RD second order order	$^{\circ}$ C ⁻²	0.0
RG	MOSFET gate resistance	Ω	0.0
RONX	MOSFET conductivity multiplier in linear region		1.0
RS	MOSFET Source ohmic resistance	Ω	0.0
RSTC1	MOSFET RS first order tempco	$^{\circ}$ C ⁻¹	0.0
RSTC2	MOSFET RS second order tempco	$^{\circ}$ C ⁻¹	0.0
THETA	MOSFET V _{gs} dependence on mobility ala mos3	V ⁻¹	0.0
TNOM	Parameter measurement temperature(TNOM)	$^{\circ}$ C	27.0
TRB1	MOSFET RB first order tempco	$^{\circ}$ C ⁻¹	0.0
TRB2	MOSFET RB second order tempco	$^{\circ}$ C ⁻²	0.0
TRD1	MOSFET RD first order tempco	$^{\circ}$ C ⁻¹	0.0
TRD2	MOSFET RD second order tempco	$^{\circ}$ C ⁻²	0.0
TRS1	MOSFET RS first order tempco	$^{\circ}$ C ⁻¹	0.0
TRS2	MOSFET RS second order tempco	$^{\circ}$ C ⁻²	0.0
TT	Bulk diode transit time	s	0.0
VJ	Body diode Junction potential	V	1.0
VTO	MOSFET threshold voltage	V	0.0
VTOTC	MOSFET's VTO tempco	$^{\circ}$ C ⁻¹	0.0
XTI	Body diode Saturation current temperature exp.		3.0

M. MOSFET Level 2010 (SiC FET) Cascode Configuration

- MOSFET Level 2010 (SiC FET) Model Parameters
 - SiC Cascode Configuration for SiC FET, which contains a JFET and MOSFET
 - SiC JFET
 - Si MOSFET
 - Reference
 - "Cascode Configuration Eases Challenges of Applying SiC JFETs", United SiC, App Note USCI_AN0004
 - "Origins of SiC FETs and their evolution towards the perfect switch", United SiC, White Paper

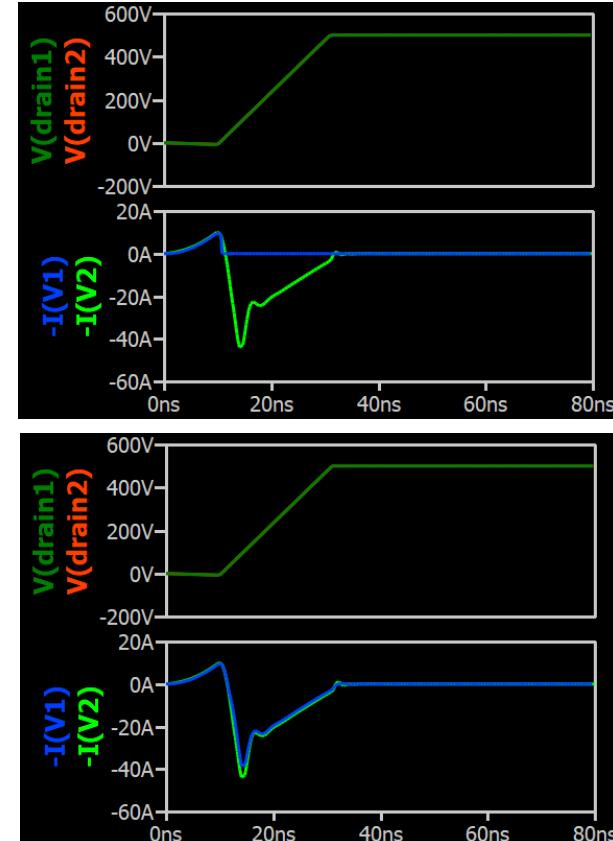
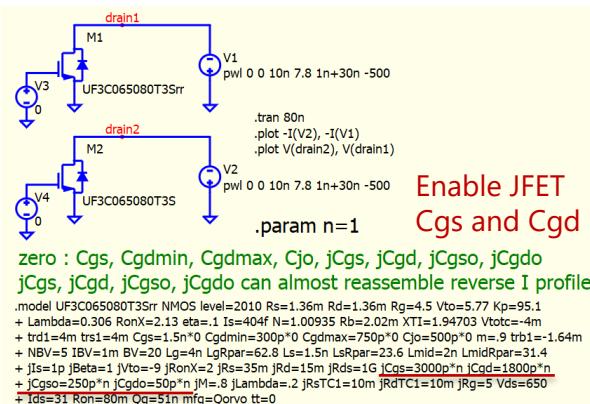
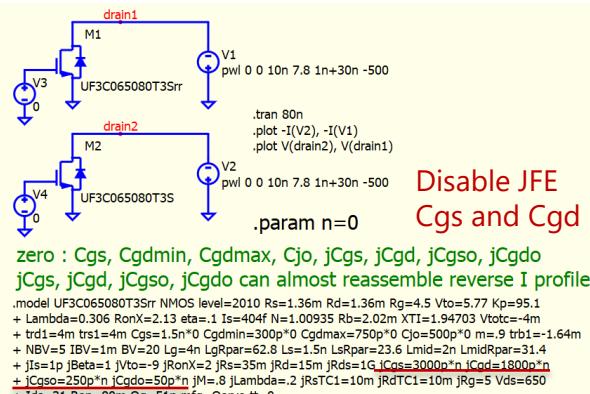


Normally Off UnitedSiC Cascode		Normally Off Typical SiC MOSFET
SiC Cascode Configuration	 = 	
Die Size	(Smaller) $R_{DS(A)} \sim 1.75 \text{ m}\Omega\cdot\text{cm}^2$	(Larger) $R_{DS(A)} \sim 3.1 - 4.5 \text{ m}\Omega\cdot\text{cm}^2$
Gate Drive	(Standard) $V_{GS} = 0V$ to $12V$ OR (SiC) $V_{GS} = -10V$ to $20V$	$V_{GS} = -5V$ to $20V$
Threshold	$V_{GS(TH)} = 5V$ Typical	$V_{GS(TH)} = 2.2V$ Typical
Intrinsic Diode	Low Qrr, +10% Over Temperature	High Qrr, High VF 3X Over Temperature
Avalanche	Yes	Yes
Short Circuit	Yes	Low

M. MOSFET Level 2010 : Reverse Current

Qspice : UF3C065080T3S Reverse Current.qsch

- Reverse Current
 - TT (transit-time) is set to 0 in Qorvo SiC FET, reverse recovery of bulk diode is not be used
 - Reverse current is only contributed by junction capacitance in SiC FET
 - From Qorvo SiC FET model, reverse current is mainly contributed from Cgs and Cgd of JFET
 - $jCgs, jCgso$
 - $jCgd, jCgdo$



O. Lossy Transmission Line

O. Lossy Transmission Line Model Parameters

- Lossy Transmission Line
 - Syntax: Onnn L+ L- R+ R- <model>
 - .model <model> LTRA [model parameters]

Lossy Transmission Line Model Parameters

Name	Description	Units	Default
C	Capacitance per meter	F/m	0.0
COMPACTABS	Abstol for history straight line checking		ABSTOL
COMPACTREL	Reltol for history straight line checking		RELTOL
G	Conductance per meter	S/m	0.0
L	Inductance per meter	H/m	0.0
LEN	Length of line	m	0.0
LININTERP	Use linear interpolation	Boolean	false
MIXEDINTERP	Use linear interpolation if quadratic results look unacceptable	Boolean	false
NOCONTROL	No timestep control	Boolean	false
NOSTEPLIMIT	Don't always limit timestep to 0.8*(delay of line)	Boolean	false
QUADINTERP	Use quadratic interpolation	Boolean	false
R	Resistance per meter	Ω/m	0.0
STEPLIMIT	Always limit timestep to 0.8*(delay of line)	Boolean	false
TRUNCDONTCUT	Don't limit timestep to keep impulse response calculation errors low	Boolean	false
TRUNCNR	Use N-R iterations for step calculation in LTRAttrunc	Boolean	false

O. Lossy Transmission Line – Fundamental Formula

- Formula of L, C, LEN, Zo and Physical LEN

- L, C, LEN are in unit length

- $Z_o = \sqrt{\frac{L}{C}}$

- It is common that C and Zo are given in datasheet
- Therefore, $L = Z_o^2 \times C$

- $TD = LEN \times \sqrt{L \times C} = LEN \times Z_o \times C$

- By $TD = \frac{LEN_{physical}}{c \times VOP} \rightarrow LEN_{physical} = LEN \times Z_o \times C \times c \times VOP$

- c : speed of light (3×10^8 m/s)
- VOP : velocity of propagation

- Example from RG393

- $Z_o = 50, C = 94.46 \text{ pF/m}, VOP = 0.695$

- If unit length LEN=1

- $L = 241.15 \text{ nH/m}$

- $TD = 4.723 \text{ ns}$

- $LEN_{physical} = 0.9847\text{m}$

Electrical Specifications

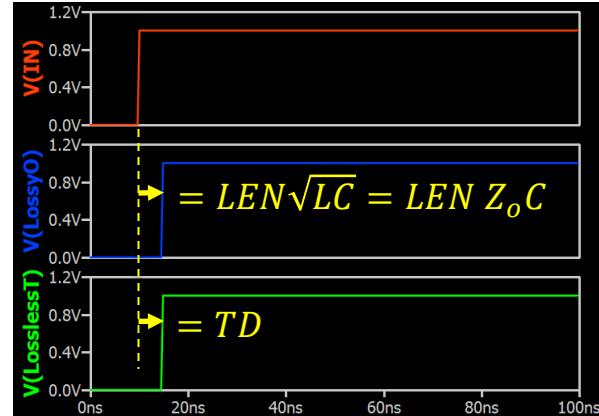
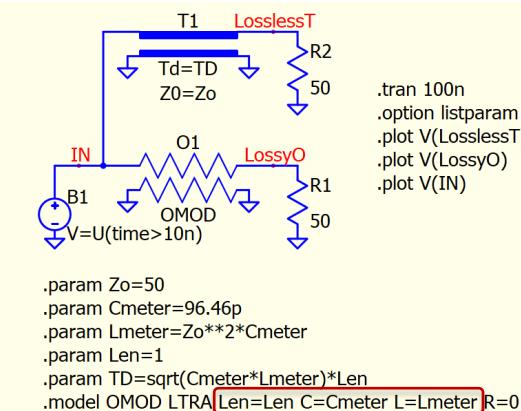
Description	Minimum	Typical	Maximum	Units
Frequency Range	DC		10	GHz
Impedance		50		Ohms
Velocity of Propagation		69.5		%
Nominal Capacitance		29.4 [96.46]		pF/ft [pF/m]

Example : Electrical Specification of RG393
(from Pasternack datasheet)

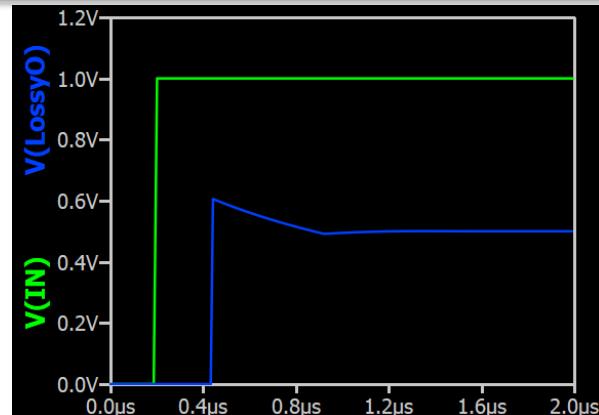
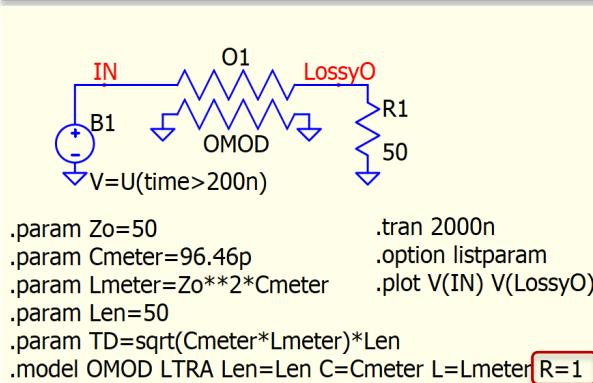
O. Model Params : C, L, LEN and R

Qspice : Oline - Len C and L.qsch | Oline - R.qsch

- C, L and LEN
 - C : C per unit length
 - L : L per unit length
 - LEN : Number of Unit Length
 - ** this is not physical length



- R
 - R : Resistance per unit length
 - In this example, R=1Ω/m and LEN=50, total series resistance is 50Ω

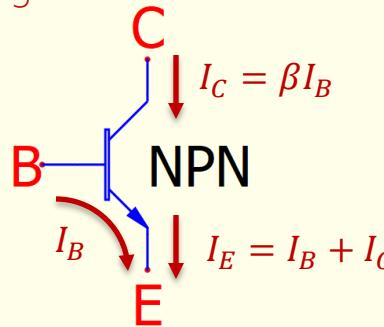


Q. Bipolar Transistor
(Level 1, Extended
SPICE Gummel-Poon)

Q. Bipolar Transistor : NPN and PNP

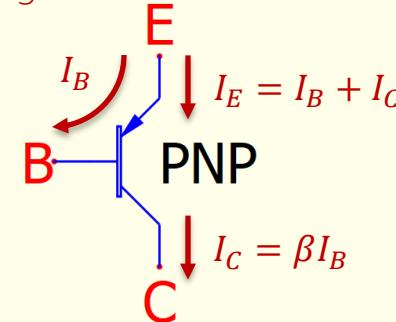
Active Region

$$V_C > V_E$$



Active Region

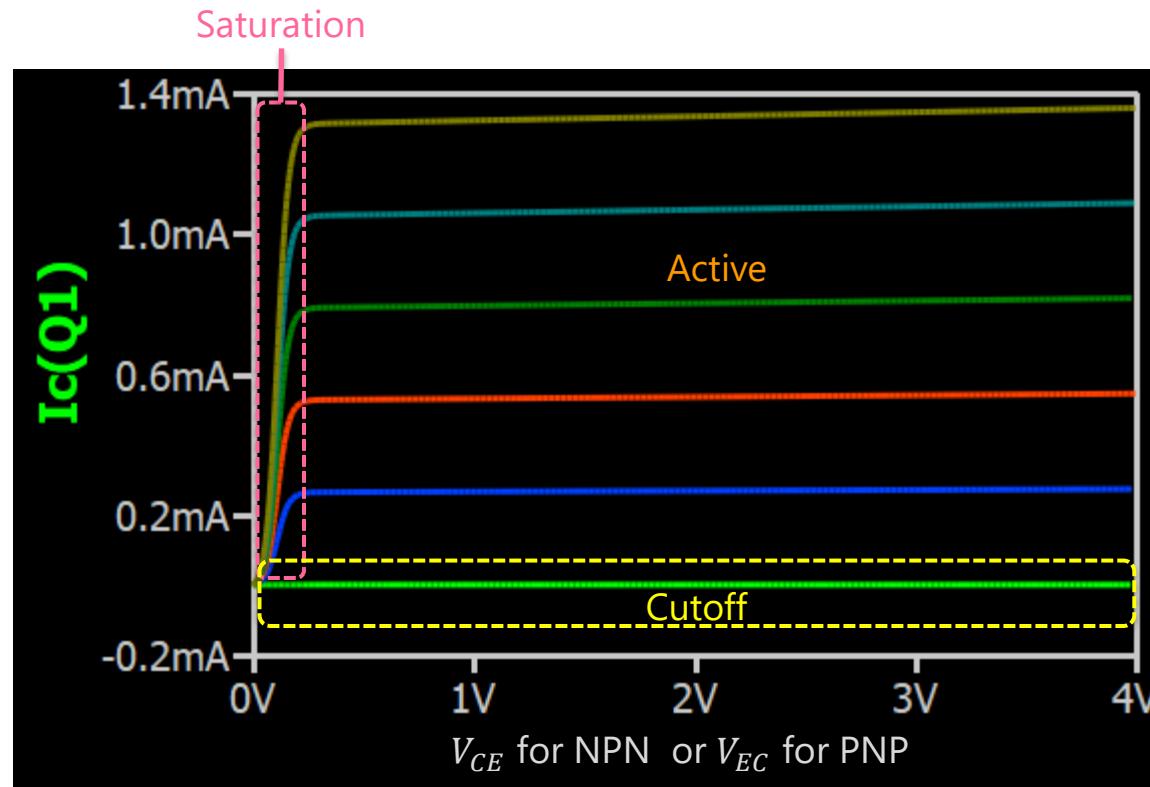
$$V_E > V_C$$



Voltage Relations	NPN
$V_E < V_B < V_C$	Active
$V_E < V_B > V_C$	Saturation
$V_E > V_B < V_C$	Cutoff
$V_E > V_B > V_C$	Reverse

Voltage Relations	PNP
$V_E < V_B < V_C$	Reverse
$V_E < V_B > V_C$	Cutoff
$V_E > V_B < V_C$	Saturation
$V_E > V_B > V_C$	Active

Q. Bipolar Transistor : NPN and PNP : Active, Saturation and Cutoff



Q. Bipolar Transistor Model Parameters

Bipolar Transistor Model Parameters			
Name	Description	Units	Default
AF	Flicker Noise Exponent		1.0
BF	Ideal forward beta		100.
BR	Ideal reverse beta		1.0
BRS	Substrate parasitic transistor beta		.0
CJC	Zero bias B-C capacitance	F	0.0
CJE	Zero bias B-E capacitance	F	0.0
CJS	Zero bias C-S capacitance(aka CCS)	F	0.0
CN ¹	Tempco for hole mobility	°C ⁻¹	NPN: 2.42 PNP: 2.20
D ¹	Tempco for scattering-limited hole carrier velocity	V ⁻¹	NPN: 0.87 PNP: 0.52
EG	Energy gap for IS temp. dependency	V	1.11
FC	Forward bias junction fit parameter		0.5
GAMMA ¹	Epitaxial region doping factor		1e-11
GMAX	Maximum PN conductivity(straight-line extension)	Ω	1000.
IBC	B-C saturation current	A	0.0
IBE	B-E saturation current	A	0.0
IKF	Forward beta roll-off corner current(aka IK)	A	Infinite
IKR	Reverse beta roll-off corner current	A	Infinite
IRB	Current for base resistance=(rb+rbm)/2	A	Infinite
IS	Saturation Current	A	1e-16
ISC	B-C leakage saturation current(aka C4)	A	IS
ISE	B-E leakage saturation current(aka C2)	A	IS
ISS	Substrate junction saturation current	A	0.0 ²
ITF	High current dependence of TF	A	0.0
KF	Flicker Noise Coefficient		0.33
MJC	B-C junction grading coefficient(aka MC)		0.33
MJE	B-E junction grading coefficient(aka ME)		0.0
MJS	Substrate junction grading coefficient(aka MS)		0.0
NC	B-C leakage emission coefficient		2.0
NE	B-E leakage emission coefficient		1.5
NEPI ¹	Epitaxial region emission coefficient		1.0
NF	Forward emission coefficient		1.0
NK	High-current roll-off coefficient		0.5
NR	Reverse emission coefficient		1.0
NS	Substrate junction emission coefficient		1.0
PTF	Excess phase	°	0.0
QCO ¹	Epitaxial region charge factor	C	0.0

QUASIMOD ¹	Temperature dependence equation selector		0
RB	Zero bias base resistance	Ω	0.0
RBM	Minimum base resistance	Ω	0.0
RC	Collector resistance	Ω	0.0
RCO ¹	Epitaxial region resistance	Ω	0.0
RE	Emitter resistance	Ω	0.0
SUBS	Set to 2 to specify lateral geometry.		1
TF	Ideal forward transit time	sec	0.0
TNOM	Parameter measurement temperature(aka TREF)	°C	27.0
TR	Ideal reverse transit time	sec	0.0
TRB1	Rb 1st order temperature coefficient	°C ⁻¹	0.0
TRB2	Rb 2nd order temperature coefficient	°C ⁻²	0.0
TRBM1	Rbm 1st order temperature coefficient(aka TRM1)	°C ⁻¹	0.0
TRBM2	Rbm 2nd order temperature coefficient(aka TRM2)	°C ⁻²	0.0
TRC1	RC 1st order temperature coefficient	°C ⁻¹	0.0
TRC2	RC 2nd order temperature coefficient	°C ⁻²	0.0
TRE1	RE 1st order temperature coefficient	°C ⁻¹	0.0
TRE2	RE 2nd order temperature coefficient	°C ⁻²	0.0
TVAF1	1st order tempco of forward Early voltage	V/°C ¹	0.0
TVAF2	2nd order tempco of forward Early voltage	V/°C ²	0.0
TVAR1	1st order tempco of reverse Early voltage	V/°C ¹	0.0
TVAR2	2nd order tempco of reverse Early voltage	V/°C ²	0.0
VAF	Forward Early voltage(aka VA)	V	Infinite
VAR	Reverse Early voltage(aka VB)	V	Infinite
VG ¹	Extrapolated band gap voltage at 0°K	V	1.206
VJC	B-C junction built in potential(aka PC)	V	0.75
VJE	B-E junction built in potential(aka PE)	V	0.75
VJS	Substrate junction built in potential(aka PS)	V	0.75
VO ¹	Carrier mobility knee voltage	V	10.0
VTF	Voltage giving VBC dependence of TF	V	Infinite
XCJC	Fraction of CJC connected to intrinsic base ³		1.0
XCJC2	Fraction of CJC connected to intrinsic base ⁴		1.0
XCJS	Fraction of CJS connected internally to RE		1.0
XTB	Forward and reverse beta temp. exp.		0.0
XTF	Coefficient for bias dependence of TF		0.0
XTI	Temperature exponent for IS(aka PT)		3.0

Q. Bipolar Transistor Basic Equation

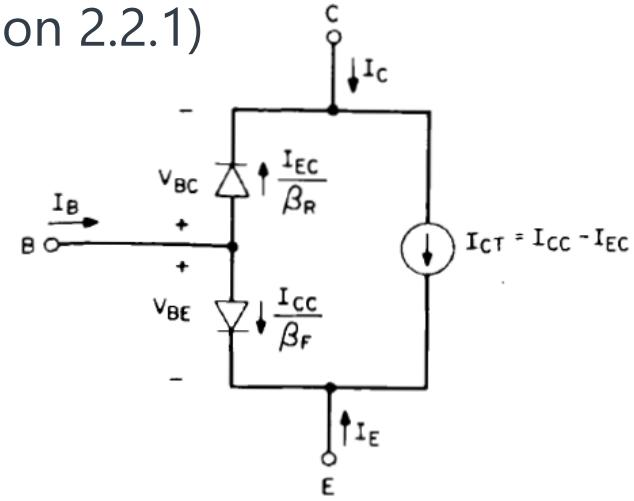
- Semiconductor Device Modeling with SPICE (Section 2.2.1)

- Basic equation from BJT Ebers-Moll static model

- $$\begin{aligned} I_C &= I_{CT} - \frac{I_{EC}}{\beta_R} \\ I_{CT} &= I_{CC} - I_{EC} \\ I_E &= -\frac{I_{CC}}{\beta_F} - I_{CT} \\ I_B &= \frac{I_{CC}}{\beta_F} + \frac{I_{EC}}{\beta_R} \end{aligned}$$

- Therefore

- $$\begin{aligned} I_C &= I_{CC} - I_{EC} - \frac{I_{EC}}{\beta_R} = \beta_F \left(I_B - \frac{I_{EC}}{\beta_R} \right) - I_{EC} - \frac{I_{EC}}{\beta_R} \\ I_C &= \beta_F I_B - \left(\frac{\beta_F}{\beta_R} + \frac{1}{\beta_R} + 1 \right) I_{EC} \end{aligned}$$



- Understanding (Approximation approach)

- If $V_{CE} \gg V_{BE}$, diode D_{BC} is reverse bias, where $I_{EC} \rightarrow 0$

- $$I_C = \beta_F I_B$$

- If $V_{BE} \gg V_{CE}$, diode D_{BC} is forward bias, and with $\beta_F \gg \beta_R$

- $$I_C = \beta_F I_B - \frac{\beta_F}{\beta_R} I_{EC}$$

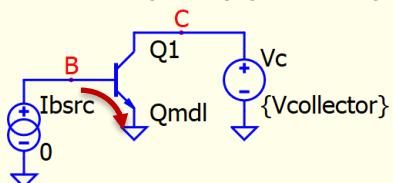
Q. Bipolar Params ($\beta_F | \beta_R$) : BF and BR

Qspice : NPN - BF.qsch / NPN - BR.qsch

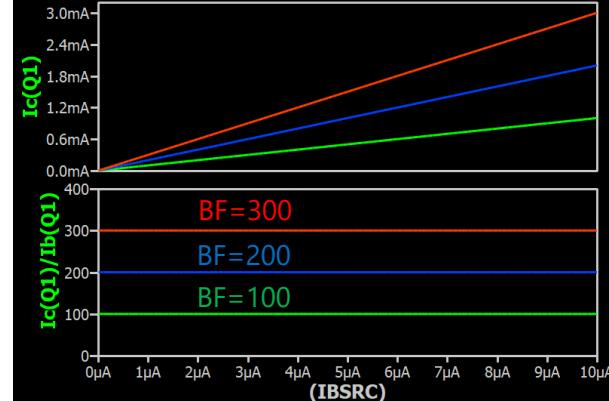
- BF
 - BF : Ideal forward beta
 - **Default BF=100**
 - $I_C = \beta_F I_B$ for $V_{CE} \gg V_{BE}$

- BR
 - BR : Ideal reverse beta
 - **Default BR=1**
 - $I_C = \beta_F I_B - \frac{\beta_F}{\beta_R} I_{EC}$
 - This effect is mainly observed when both BE and BC diode are forward bias, where collector-emitter voltage is close to 0V
 - Smaller β_R , larger $\frac{\beta_F}{\beta_R}$ and less I_C and less $\frac{I_C}{I_B}$

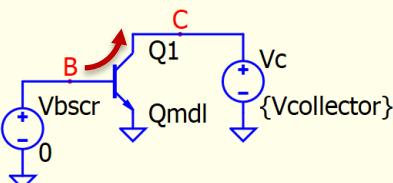
Sweep Parameters
`.param Vcollector = 10
.param Ibmax=10μ
.dc Ibsrc 10n {Ibmax} {Ibmax/100}`



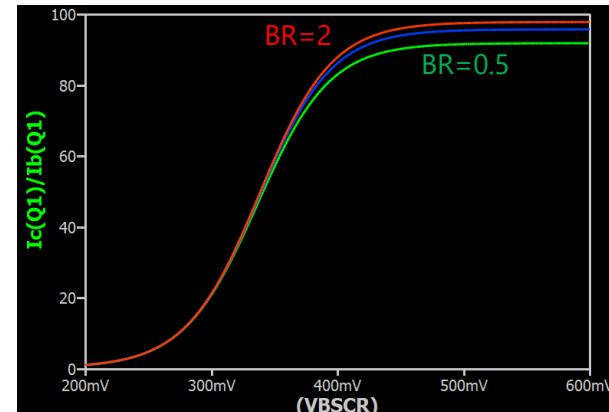
`.step param bf list 100 200 300
.model Qmdl NPN BF={bf}`



Sweep Parameters
`.param Vcollector = 0.2
.param Ibmin=10p
.dc Vbscr 0.2 0.6 0.001`



`.step param br list 0.5 1 2
.model Qmdl NPN BR={br} BF=100`

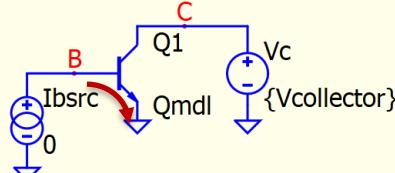


Q. Bipolar Params ($\beta_F | \beta_R$) : IKF

Qspice : NPN - IKF.qsch / NPN - IKR.qsch

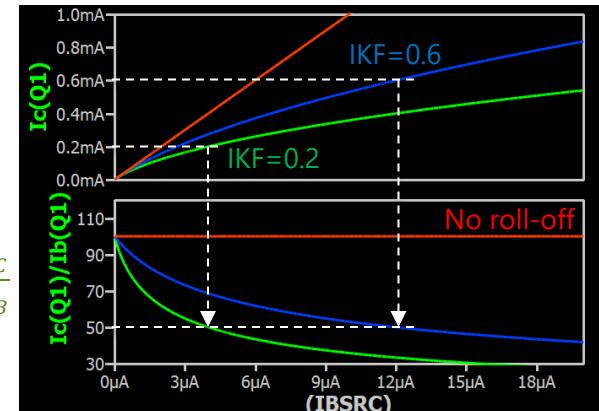
- IKF
 - IKF : Forward beta roll-off corner current
 - Default IKF=infinite (A)**
 - IKF models β_F drop when I_C increase
 - IKF is I_C where effective BF (β_F) reduces multiple approximate by $(1-NK)$
 - This example Default NK=0.5, effective BF is reduced to 50 @ $I_C=IKF$

Sweep Parameters
`.param Vcollector = 10
.param Ibmax=20μ
.dc Ibsrc 10n {Ibmax} {Ibmax/100}`



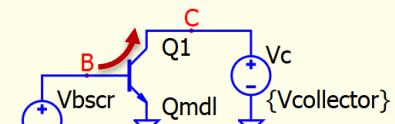
`.step param ikf list 0.2m 0.6m 1e12
.model Qmdl NPN BF=100 IKF=ikf`

$$\beta_{F,\text{eff}} = \frac{I_C}{I_B}$$

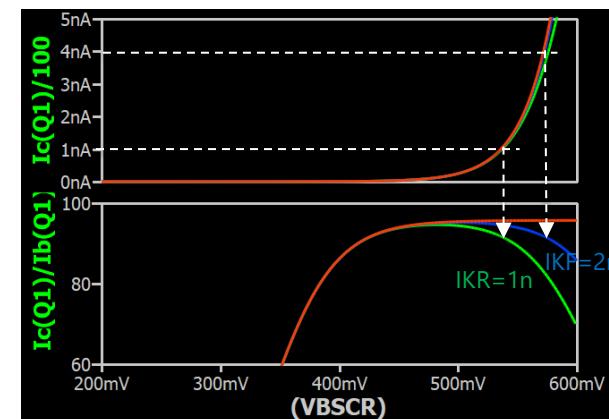


- IKR
 - IKR : Reverse beta roll-off corner current
 - Default IKR=infinite (A)**
 - IKR models β_F drop when I_C increase
 - IKR is $\frac{I_C}{BF}$ where effective BR (β_R) reduces to 50%

Sweep Parameters
`.param Vcollector = 0.2
.param Ibmin=10p
.dc Vbscr 0.2 0.6 0.001`



`.param bf=100
.step param ikr list 1n 4n 1e12
.model Qmdl NPN BR=1 BF=bf IKR=ikr`



Q. Bipolar Params ($\beta_F | \beta_R$) : NK

Qspice : NPN - NK.qsch

- NK
 - NK : High-current roll-off coefficient
 - **Default NK=0.5**
 - This parameter is only be used if IKF not infinite

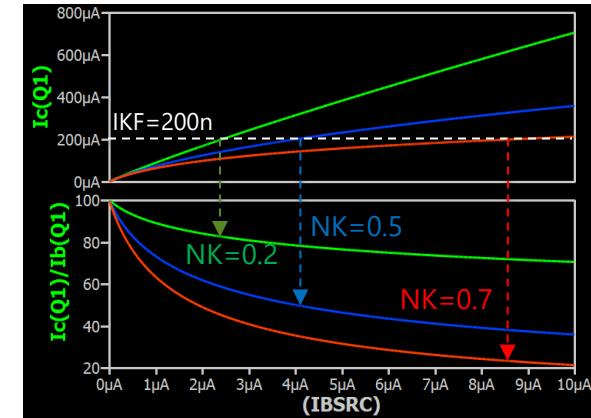
Sweep Parameters
.param Vcollector = 10
.param Ibmax=20 μ
.dc Ibsrc 10n {Ibmax} {Ibmax/100}

Plot Output
.plot Ic(Q1)/Ib(Q1)
.plot Ic(Q1)

Diagram

Model Definition

```
.step param nk list 0.2 0.5 0.7  
.model Qmdl NPN BF=100 IKF=200 $\mu$  NK=nk
```



Q. Bipolar Params (B-E and B-C Diode) : IS, NF, BF, NR, BR

Qspice : NPN - IS NF BF.qsch / NPN - IS NR BR.qsch

- Diode of Base-Emitter
 - **IS** : Saturation Current
 - **NF** : Forward emission coefficient
 - Default **IS** = $1e-16A$
 - Default **NF** = 1
 - In this case

$$I_B = I_{BE} = \frac{I_{CC}}{\beta_F}$$

$$= \frac{1}{\beta_F} I_S \left(e^{\frac{qV_{BE}}{n_F kT}} - 1 \right)$$

where β_F is **BF**

- Diode of Base-Collector
 - **IS** : Saturation Current
 - **NR** : Reverse emission coefficient
 - Default **IS** = $1e-16A$
 - Default **NR** = 1
 - In this case

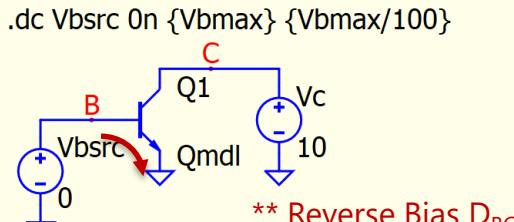
$$I_B = I_{BC} = \frac{I_{EC}}{\beta_R}$$

$$= \frac{1}{\beta_R} I_S \left(e^{\frac{qV_{BE}}{n_R kT}} - 1 \right)$$

where β_R is **BR**

Sweep Parameters
.param Vcollector = 10

.param Vbmax=0.8
.dc Vbsrc 0n {Vbmax} {Vbmax/100}

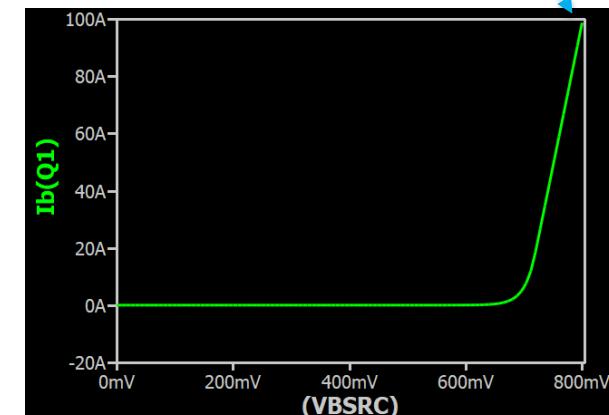
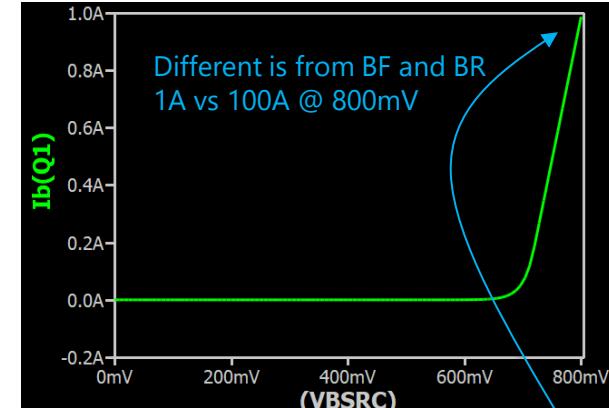
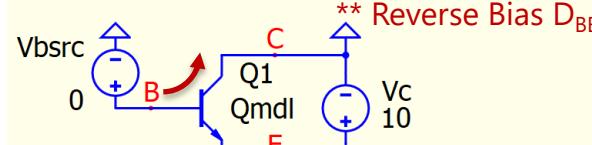


.model Qmdl NPN IS=1e-16 NF=0.7 BF=100

Sweep Parameters
.param Vcollector = 10

.param Vbmax=0.8
.dc Vbsrc 0n {Vbmax} {Vbmax/100}

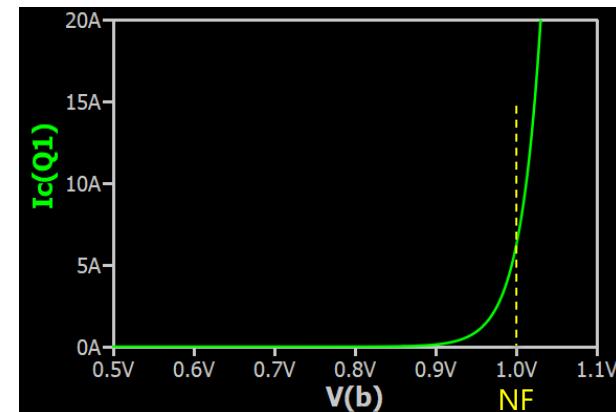
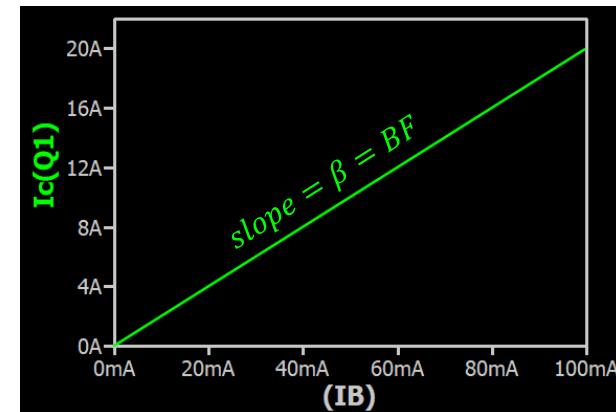
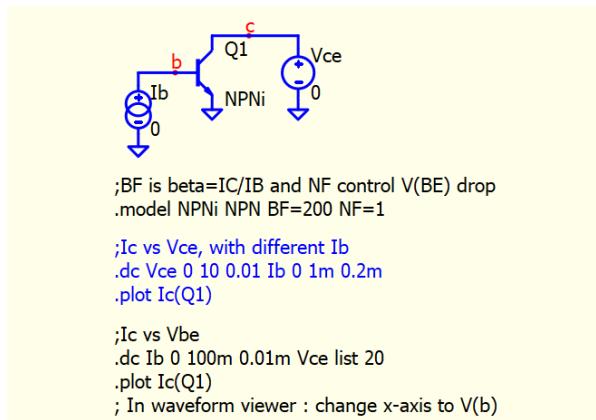
.model Qmdl NPN IS=1e-16 NR=0.7 BR=1



Q. Bipolar – BF and NF only (textbook modeling)

Qspice : NPN - BF NF only.qsch

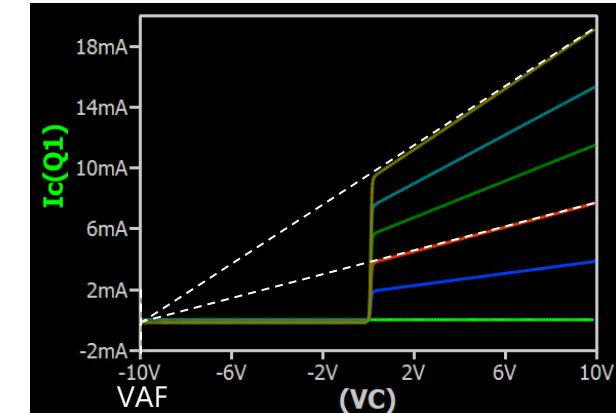
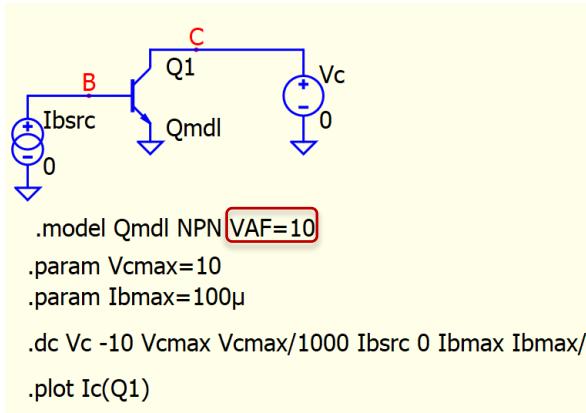
- BF and NF only
 - A model only with BF and NF
 - $BF = \beta = \frac{\Delta I_C}{\Delta I_B}$
 - Base-emitter is diode characteristic, and its voltage drop can be determined by NF



Q. Bipolar Params (Active Region) : VAF

Qspice : NPN - VAF.qsch

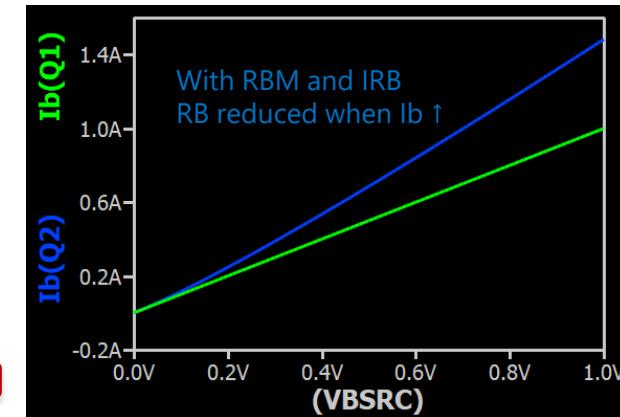
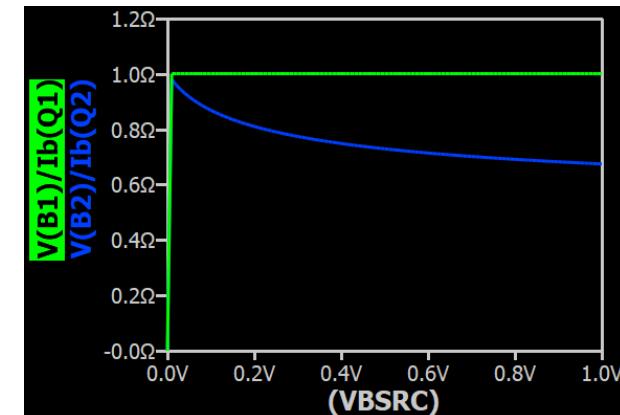
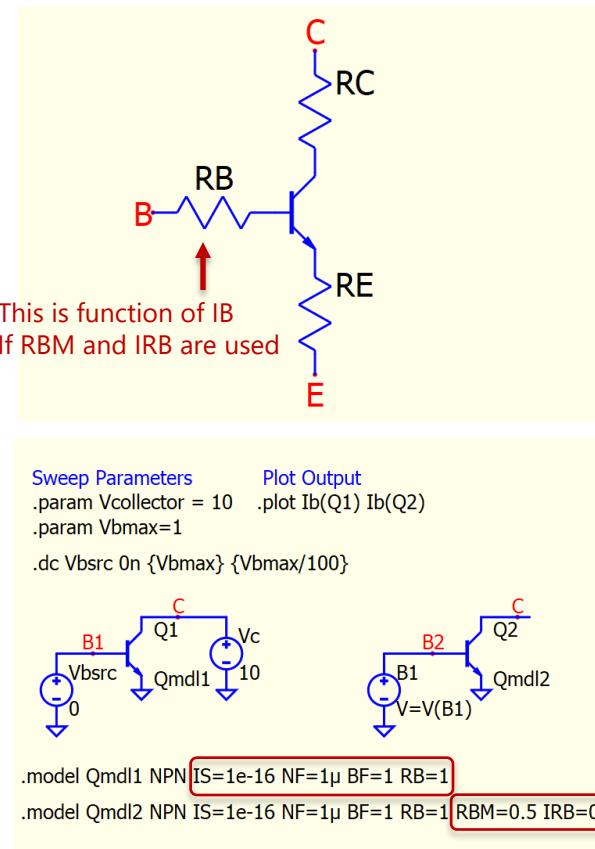
- VAF (aka VA)
 - VAF : Forward Early voltage
 - **Default VAF=Infinite (V)**



Q. Bipolar Params (Resistance): RB, RC, RE, RBM and IRB

Qspice : NPN - RBM IRB.qsch

- RB, RC, RE
 - RB : Zero bias base resistance
 - If RBM and IRB are used, base resistor is current dependent
 - RC : Collector resistance
 - RE : Emitter resistance
- RBM, IRB
 - RBM : Minimum base resistance
 - IRB : Current for base resistance
 - $R_{B, \text{effective}} @ \text{IRB} = \frac{1}{2} (\text{RB} + \text{RBM})$
 - Default RBM=0Ω
 - Default IRB=Infinite(A)
 - If these 2 parameters are defined, base resistance is function of base current



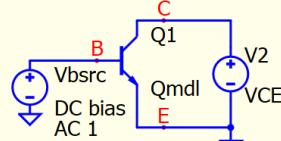
Q. Bipolar Params (Capacitance) : CJC, MJC, VJC, CJE, MJE, VJE

- B-C, B-E and C-S nonlinear capacitance
 - Model nonlinear capacitor between B-C, B-E and C-S
 - Parameters definition follows diode CJO, M and VJ
 - $C_j = \frac{C_{JO}}{\left(1 - \frac{V_D}{V_J}\right)^M}$
 - CJO : Zero-bias junction capacitance
 - M : Grading coefficient
 - VJ : Junction potential

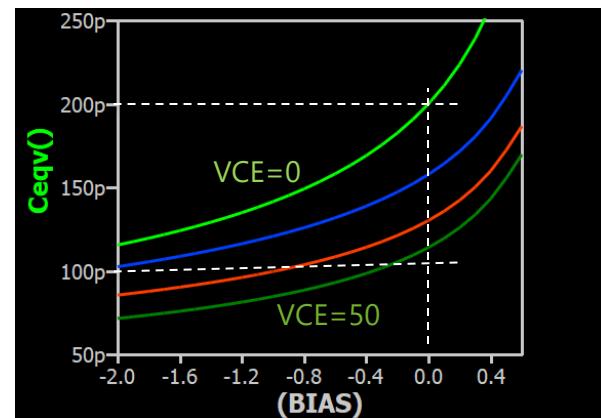
Diode	CJO	M	VJ
B-C	CJC	MJC	VJC
B-E	CJE	MJE	VJE
C-S	CJS	MJS	VJS

- Refer to D. Diode CJO, M and VJ
 - C-S normally not modeled

```
.param f=1Meg          Calculate Equivalent C
.ac list f
.step param bias -2 0 6 0 1
.func Zim() imag(V(B)/Ib(Q1))
.func Ceqv() -1/2/pi/f/Zim()
.plot Ceqv()
```



```
.model Qmdl NPN CJE=100p MJE=0.5 VJE=1  
+CJC=100p MJC=0.5 VJC=1
```



Q. Bipolar Params (Capacitance) : CJC, MJC, VJC, CJE, MJE, VJE

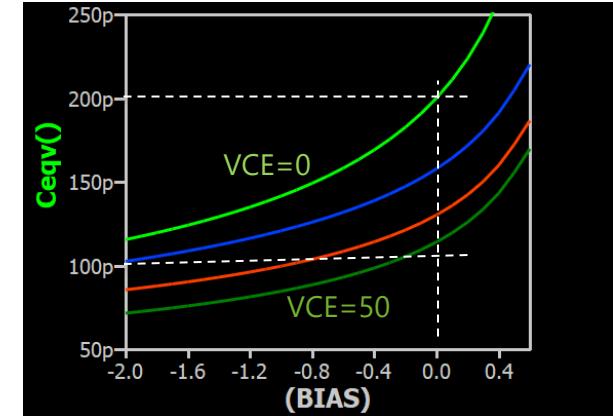
Qspice : NPN - CJC CJE.qsch

- B-C, B-E and C-S nonlinear capacitance
 - Model nonlinear capacitor between B-C, B-E and C-S
 - Parameters definition follows diode CJO, M and VJ
 - $C_j = \frac{C_{JO}}{(1 - \frac{V_D}{V_J})^M}$
 - CJO : Zero-bias junction capacitance
 - M : Grading coefficient
 - VJ : Junction potential
- Simulation
 - This simulation modeled both capacitance between B-C and B-E. Change VCE to observe overall capacitance effect in different bias voltage

```
.param f=1Meg
.ac list f
.step param bias -2 0.6 0.1
.step param VCE list 0 2 10 50
.model Qmdl NPN CJE=100p MJE=0.5 VJE=1
+CJC=100p MJC=0.5 VJC=1
```

Calculate Equivalent C

```
.func Zim() imag(V(B)/Ib(Q1))
.func Ceqv() -1/2/pi/f/Zim()
.plot Ceqv()
```



Diode	CJO	M	VJ
B-C	CJC	MJC	VJC
B-E	CJE	MJE	VJE
C-S	CJS	MJS	VJS

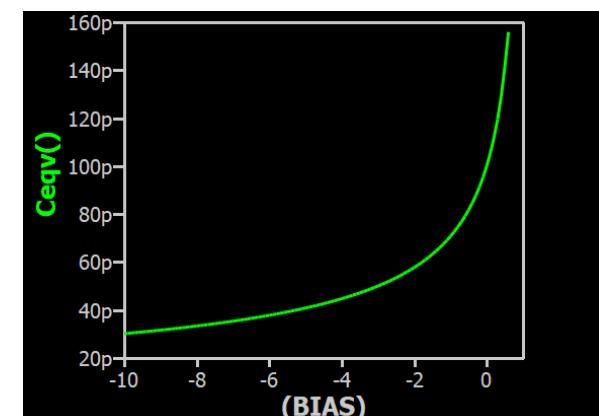
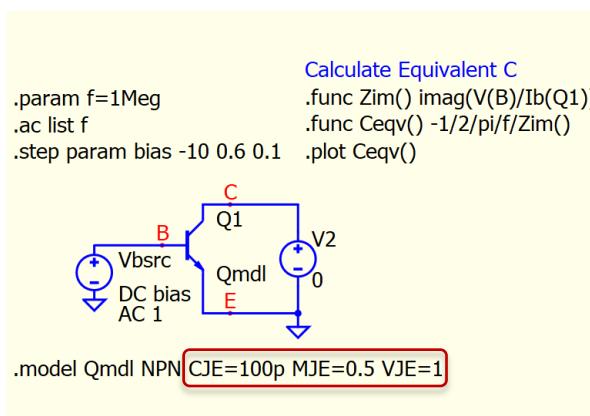
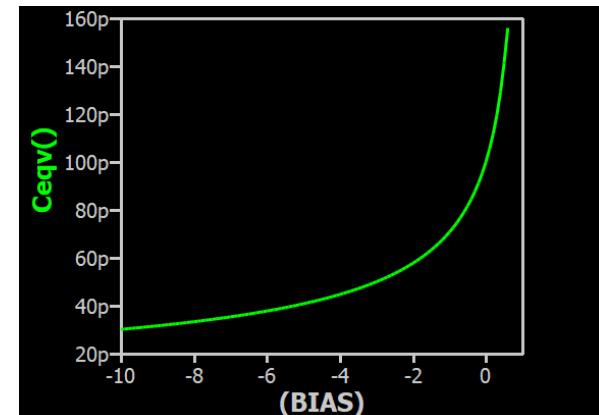
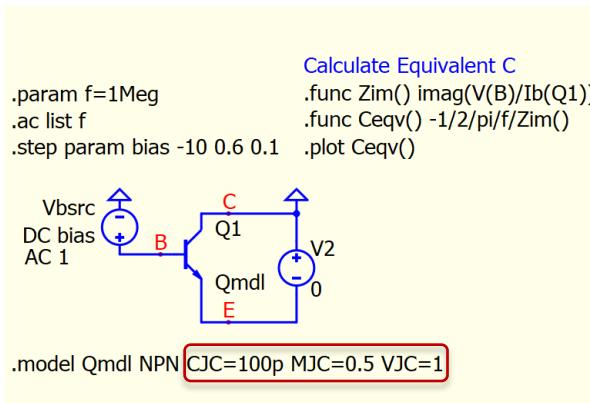
- Refer to D. Diode CJO, M and VJ
- C-S normally not modeled

- Simulation Explanation
 - CBC and CBE are nonlinear capacitance of junction diode from B to C and B to E
 - Higher reverse voltage, lower capacitance
 - Therefore, increase VCE, increase reverse voltage of B-C and affects B-C capacitance. However, increase VCE doesn't affect VBE, and no effects in B-E capacitance
 - @VB=0V, when VCE=0, no reverse in both B-C and B-E and capacitance equals CJC+CJE

Q. Bipolar Params (Capacitance) : CJC, MJC, VJC, CJE, MJE, VJE

Qspice : NPN - CJC MJC VJC.qsch / NPN - CJE MJE VJE.qsch

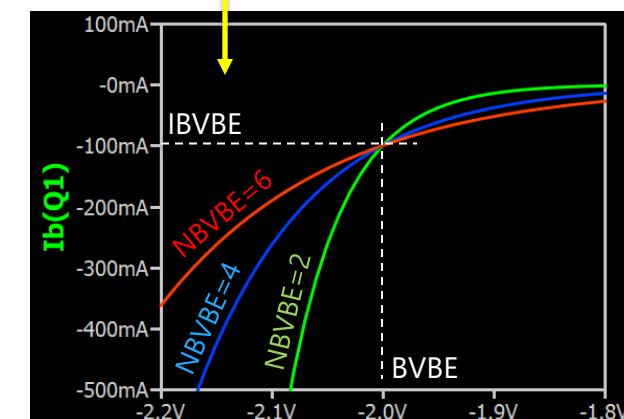
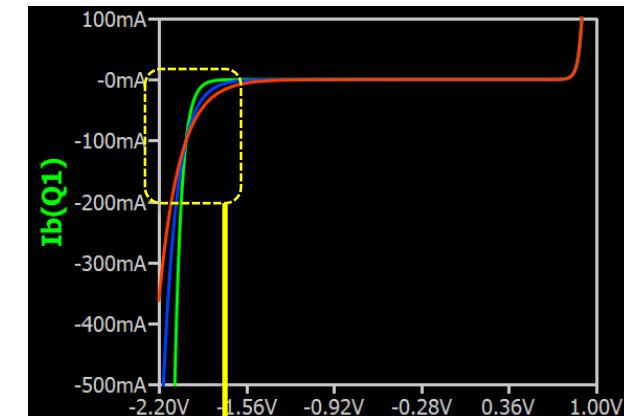
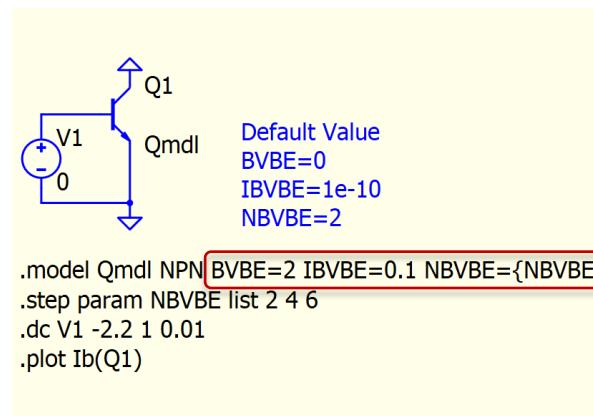
- CJC, MJC and VJC
 - CJC : Zero bias B-C capacitance
 - MJC : B-C junction grading coefficient
 - VJC : B-C junction built in potential
 - **Default CJC=0F**
 - **Default MJC=0.5**
 - **Default VJC=1**
- CJE, MJE and VJE
 - CJE : Zero bias B-E capacitance
 - MJE : B-E junction grading coefficient
 - VJE : B-E junction built in potential
 - **Default CJE=0F**
 - **Default MJE=0.5**
 - **Default VJE=1**



Q. Bipolar Params (B-E Diode) : BVBE, IBVBE, NBVBE

Qspice : NPN - BVBE IBVBE NBVBE.qsch

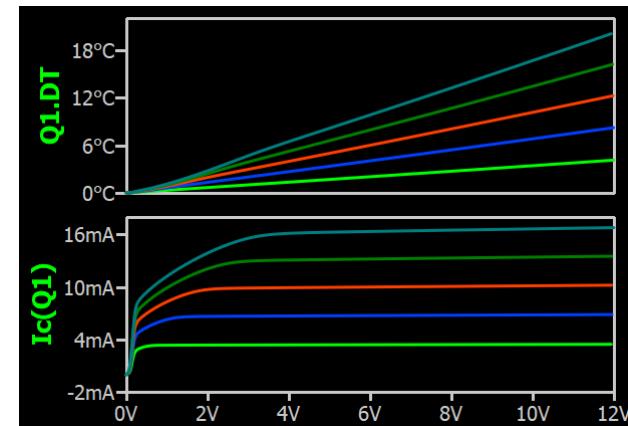
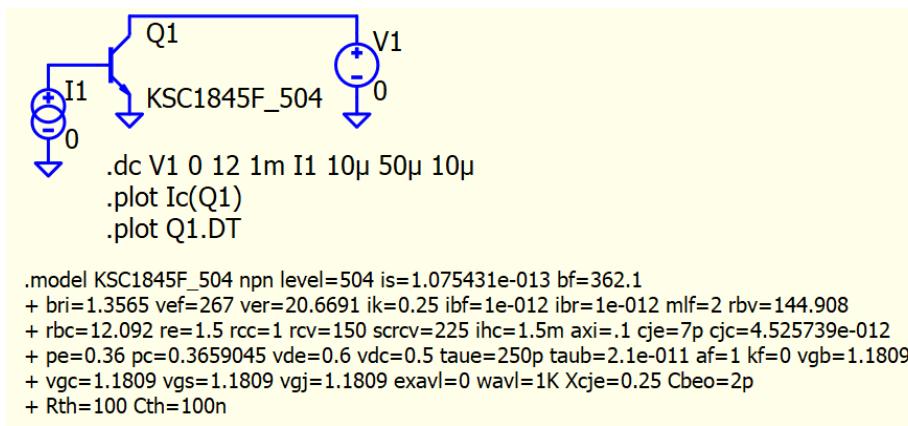
- BVBE, IBVBE, NBVBE
 - Qspice Revision History
 - 12/30/2023 Implemented bipolar B-E breakdown via model parameters BVBE, IBVBE, AND NBVBE.
 - Parameters to model breakdown characteristic of Base-Emitter Diode
 - BVBE : Reverse breakdown voltage
 - IBVBE : Current at breakdown voltage
 - NBVBE : Emission coefficient
 - **Default BVBE=0V**
 - **Default IBVBE=1e-10A**
 - **Default NBVBE=2**



**Q. Bipolar Transistor
(MEXTRAM 504, Level
504 by NXP)**

Q. Transistor (MEXTRAM 504)

- MEXTRAM 504
 - Release history
 - Qspice Revision History : 11/29/2023 Released a MEXTRAM 504(rev 12) implementation
 - <https://forum.qorvo.com/t/models-of-transistors-with-thermal-parameters/15809/5>
 - Comment from Mike Engelhardt
 - It's the current minor revision level, 12
 - It should be able to properly run that model without the collector current going to zero
 - If no self heating node is specified, it will make one called Q1.DT. Otherwise you can make a five terminal transistor and connect a thermal network the temperature node



R. Resistor

R. Resistor Instance and Model Parameters

- Resistor Syntax
 - Rnnn N1 N2 [<model>] <resistance> [instance parameters]
 - .model <model> RES [model parameters]

Resistor Instance Parameters

Name	Description	Units	Default
AC	Value to use for .ac analysis	Ω	RESISTANCE
B ¹	NTC coefficient		0.0
CTH ²	Thermal capacitance	F	Model value
RESISTANCE	Resistance	Ω	none
RTH ²	Thermal resistance	Ω	Model value
L ³	Length	m	1.0
W ³	Width	m	1.0
M	Number of identical parallel instances		1.0
SHORTED	Don't issue a warning if culled due to being shorted out		not set
TC ¹	Polynomial temperature coefficient list		0.0
TC1 ¹	1nd order temperature coefficient	$^{\circ}\text{C}^{-1}$	0.0
TC2 ¹	2nd order temperature coefficient	$^{\circ}\text{C}^{-2}$	0.0
TEMP	Instance temperature	$^{\circ}\text{C}$	Circuit temperature
TNOM	Temperature resistance was measured(aka TREF)	$^{\circ}\text{C}$	Circuit TNOM

Resistor Model Parameters

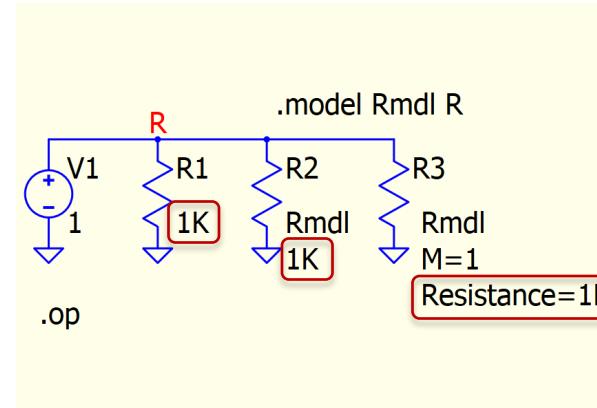
Name	Description	Units	Default
CTH ²	Thermal capacitance	F	0.0
DEFW	Default device width	m	1.0
NARROW ³	Narrowing of resistor	m	1.0
R	Resistor multiplier		1.0
RTH ²	Thermal resistance	Ω	0.0
RSH ³	Sheet resistance(for instance L and W)	Ω/\square	0.0
TC ¹	Polynomial temperature coefficient list		0.0
TC1 ¹	1nd order temperature coefficient	$^{\circ}\text{C}^{-1}$	0.0
TC2 ¹	2nd order temperature coefficient	$^{\circ}\text{C}^{-2}$	0.0
TCE ¹	Exponential order temperature coefficient	$^{\circ}\text{C}/^{\circ}\text{C}$	0.0
TNOM	Parameter measurement temperature(aka TREF)	$^{\circ}\text{C}$	Circuit TNOM

R. Resistor – Resistance

Qspice : R - Resistance.qsch

- Resistance

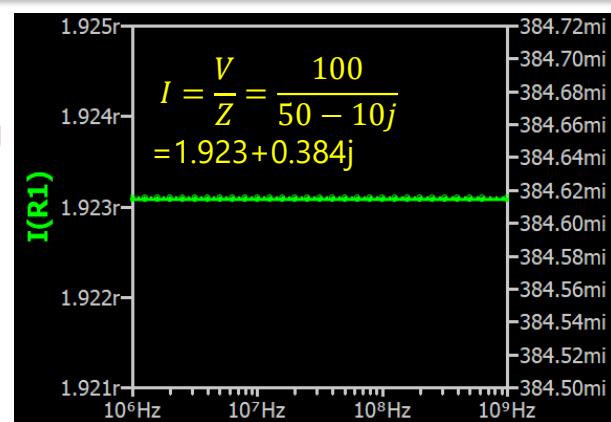
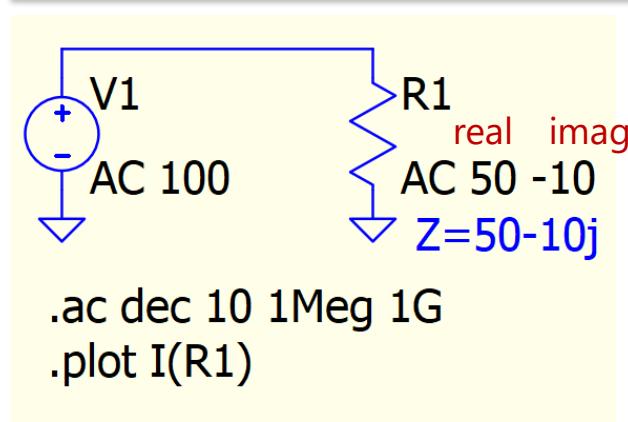
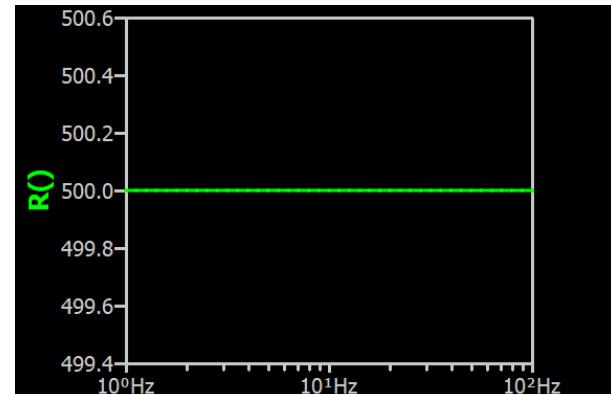
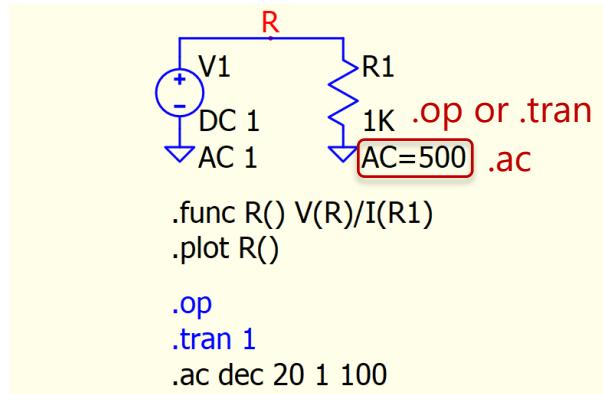
- Resistance can be specified in three way
 - #1 : Resistance value at 1st attribute
 - #2 : <model> at 1st attribute and resistance value at 2nd attribute
 - #3 : <model> at 1st attribute, resistance value after 2nd attribute with **Resistance=<value>**



R. Resistor – AC in .ac (ac analysis)

Qspice : R - AC.qsch | R - AC complex impedance.qsch

- AC [RE]
 - Value to use for .ac analysis
 - **Default = RESISTANCE**
 - In this example, .op/.tran treats R1 as 1Kohm, while .ac treats R1 as 500ohm
 - Therefore, a single resistor can have two distinct resistance values in its DC and AC analysis. One common use case is in the op-amp open-loop gain test circuit
- AC [RE] [IM]
 - AC resistance can be defined as complex impedance $R + jX$
 - **Syntax : AC [RE] [IM]**



R. Resistor – AC without Resistance (.op)

Qspice : R - AC complex impedance (.op).qsch

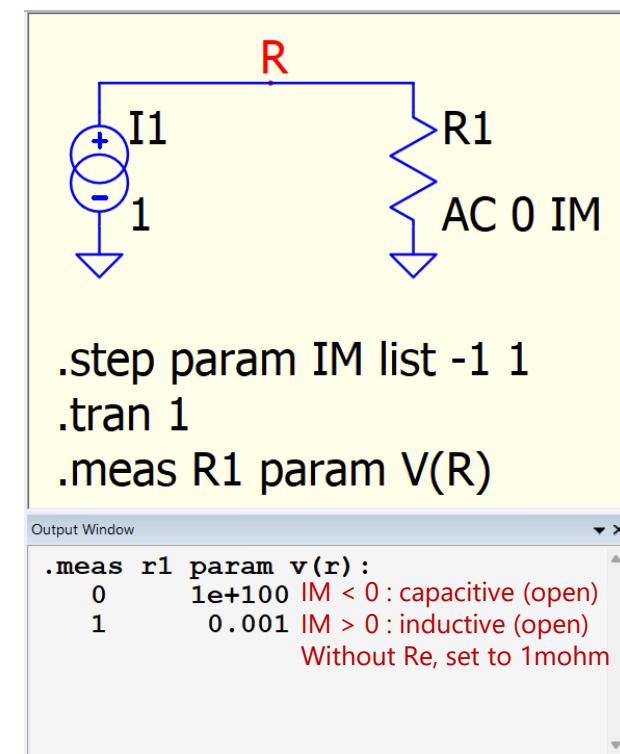
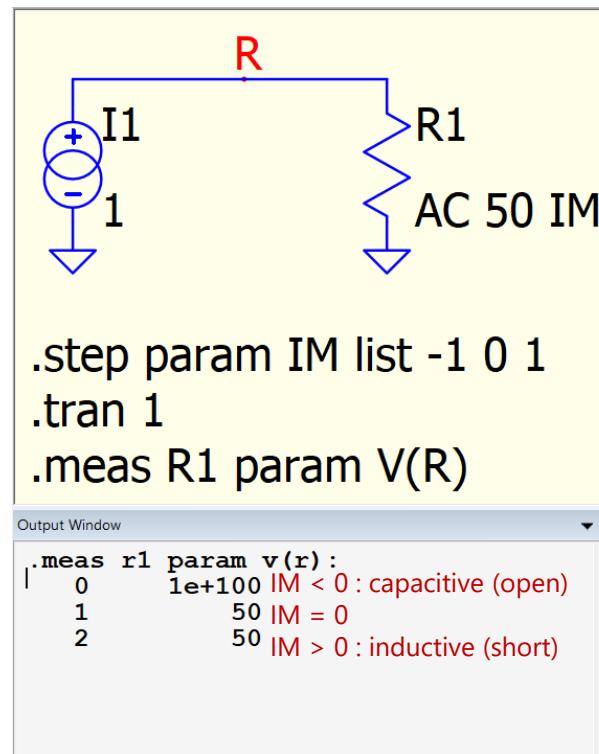
- AC Impedance without Resistance

- If RESISTANCE is not specified but only AC is defined, its DC resistance behaves as follows

- AC Re Im

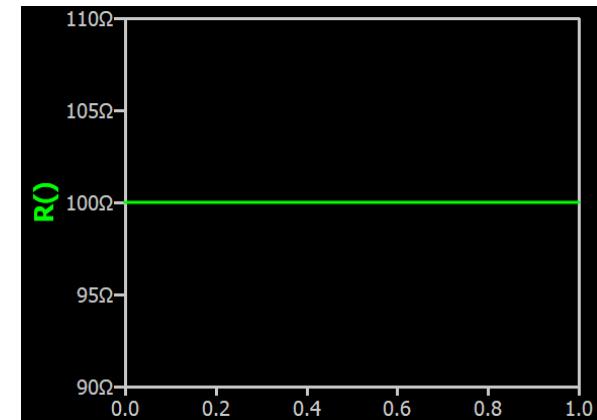
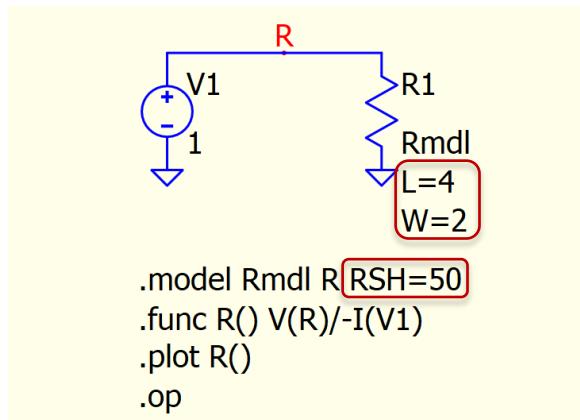
- If $Im < 0$, it is capacitive and acts as open circuit ($R=1e100$)
- If $Im > 0$, it is inductive and acts as short circuit, Re is in series and therefore, DC resistance set to Re

- This allows a complex impedance to provide a DC solution based on its reactance



R. Resistor – RSH, L and W (Sheet Resistance)

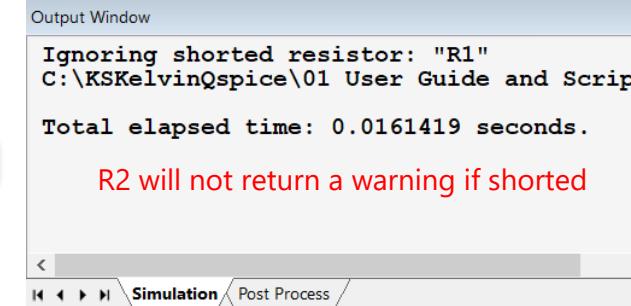
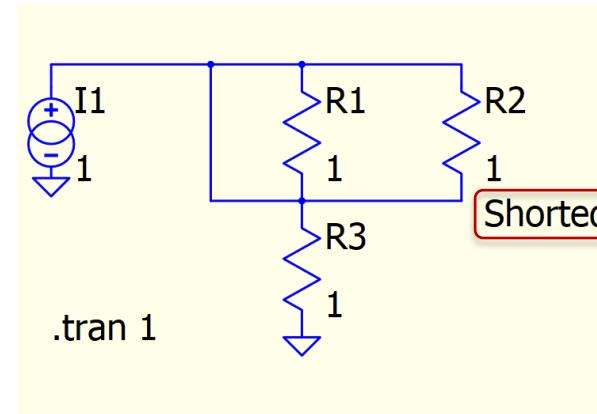
- RSH, L and W
 - RSH : Sheet Resistance
 - L : Length
 - W : Width
 - RSH is model param and L, W are instance params
 - If RESISTANCE is defined, RSH, L and W will be ignored
 - Sheet resistance is the resistance of a square of the conductive thin film with uniform thickness
 - $R = R_{SH} \times \frac{L}{W}$



R. Resistor – Shorted

Qspice : R - Shorted.qsch

- Shorted
 - Don't issue a warning if culled due to being shorted out
 - **Default : not set**



R. Resistor – Thermal Related Parameters

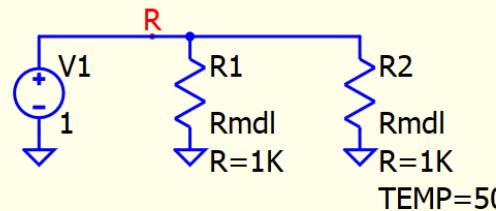
- Resistor Thermal Equation
 - Formula with temperature coefficient list TC : $R_T = R_{nom}(1 + \sum T_{Cn}\Delta T^n)$
 - TC supports a list of temperature coefficient in format TC=<value1>,<value2>,<value3>,...
 - Formula with temperature coefficient TC1 and TC2 : $R_T = R_{nom}(1 + T_{C1}\Delta T + T_{C2}\Delta T^2)$
 - Formula with temperature coefficient TCE : $R_T = R_{nom} e^{\frac{T_{CE}}{100}\Delta T}$
 - Formula with B (or beta) for thermistor (NTC) : $R_T = R_{nom} e^{B \times \left(\frac{1}{T} - \frac{1}{T_{nom}}\right)}$
 - where Temperature in above formula is in Kelvin, $\Delta T = T - T_{nom}$

R. Resistor Params : Tnom and TEMP (Instance param), TC

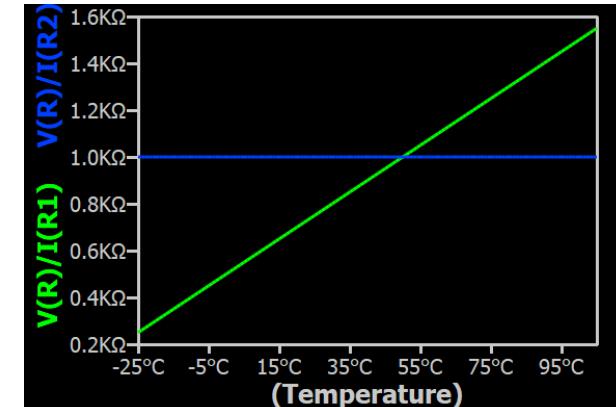
Qspice : R - TNOM TEMP.qsch | R - TC.qsch

- Tnom and TEMP

- Tnom : Parameter measurement temperature
- TEMP : Instance temperature in Instance param
- Default Tnom=27oC**
- Default TEMP=Circuit** Temperature
- In this example, instance temperature TEMP is assigned for R2, which forced R2 at TEMP instead of circuit temperature

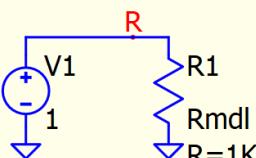


```
.model Rmdl R Tnom=50 TC1=0.01  
.plot V(R)/I(R1) V(R)/I(R2)  
.dc Temp -25 105 1  
.op
```

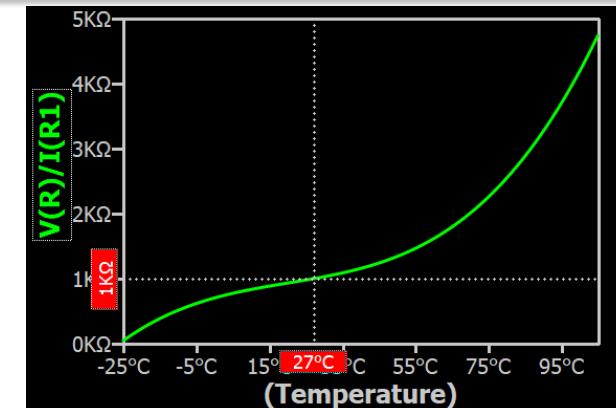


- TC

- TC : Polynomial temperature coefficient list
- Default TC=0**
- $TC = <value1>, <value2>, <value3>, \dots$
- $R_T = R_{nom}(1 + \sum T_{Cn} \Delta T^n)$
- where Tnom is R defined measurement temperature



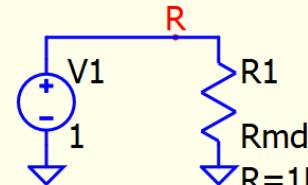
```
.model Rmdl R TC=0.01,0.0001,0.000005  
.plot V(R)/I(R1)  
.dc Temp -25 105 0.001  
.op
```



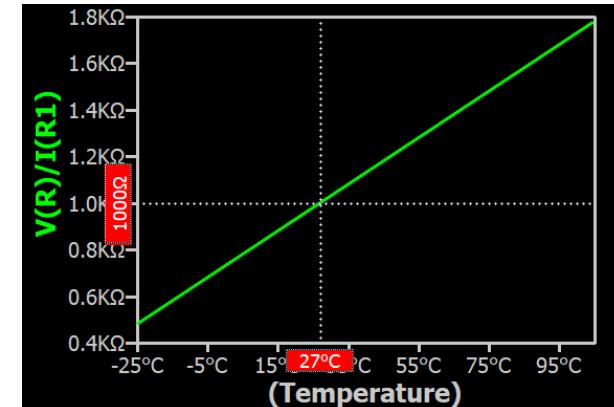
R. Resistor Params : TC1 and TC2

Qspice : R - TC1.qsch / R - TC2.qsch

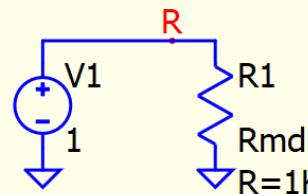
- TC1
 - TC1 : 1nd order temperature coefficient
 - **Default TC1=0°C⁻¹**
 - $RT = R*(1+TC1*(T-Tnom))$
 - Where Tnom is R defined measurement temperature



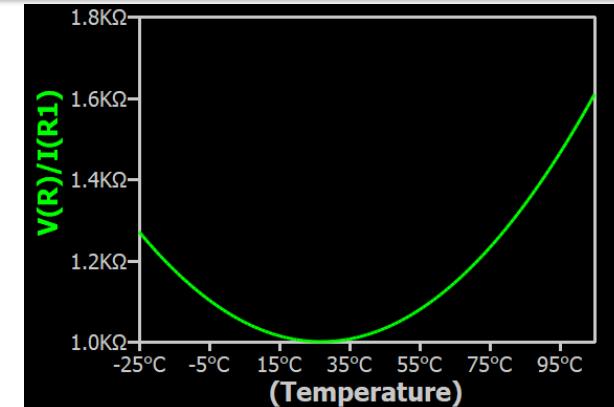
```
.model Rmdl R TC1=0.01  
.plot V(R)/I(R1)  
.dc Temp -25 105 0.001  
.op
```



- TC2
 - TC2 : 2nd order temperature coefficient
 - **Default TC2=0°C⁻²**
 - $RT = R*(1+TC2*(T-Tnom)^2)$
 - TC2 and TC1 can be used together



```
.model Rmdl R TC2=0.0001  
.plot V(R)/I(R1)  
.dc Temp -25 105 1  
.op
```

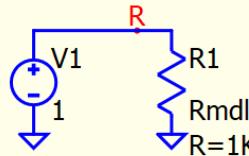


R. Resistor Instance Param : TCE and B for Thermistor (NTC)

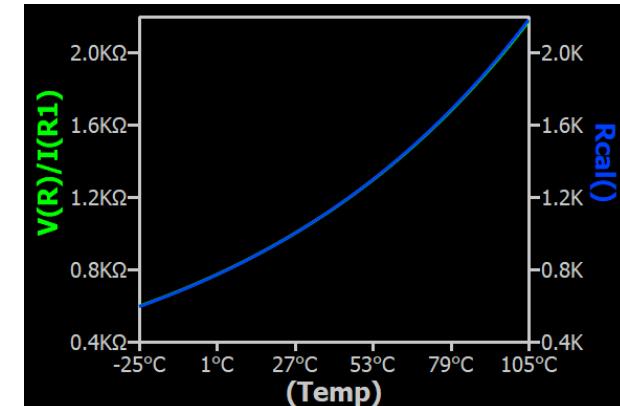
Qspice : R - TCE.qsch | R - B.qsch

TCE

- TCE : Exponential order temperature coefficient
- Default TCE=0%/ $^{\circ}\text{C}$**
- $R_T = R_{\text{nom}} \times \exp((T-T_{\text{nom}})/100 \times \text{TCE})$
- If TCE is specified non-zero, TC1 and TC2 are ignored

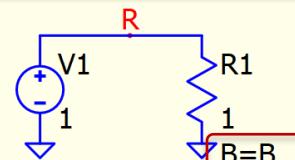


```
.model Rmdl R TCE=1  
.plot V(R)/I(R1) Rcal()  
.dc Temp -25 105 1  
.step param Temp -25 105 1  
.op  
.func Rcal() 1K*exp((Temp-27)/100*1)
```

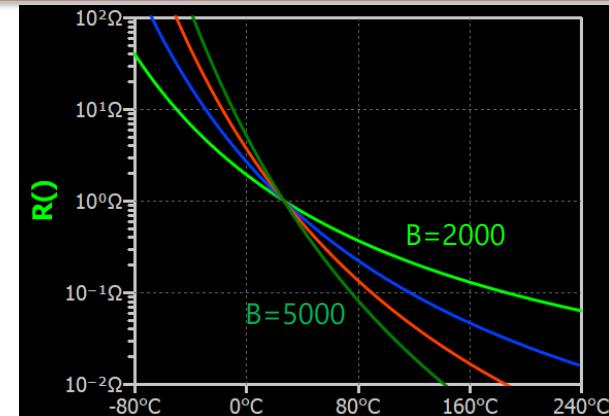


B (for thermistor/NTC)

- NTC Coefficient
- Default B=0**
- $R_T = R_{\text{nom}} e^{B \times \left(\frac{1}{T} - \frac{1}{T_{\text{nom}}} \right)}$
- Refer to TDK NTC thermistor general information, B value of common NTC materials range is from 3000 to 5000



```
.op  
.step param TEMP -80 240 1  
.step param B 2000 5000 1000  
.func R() V(R)/I(R1)  
.plot R()
```

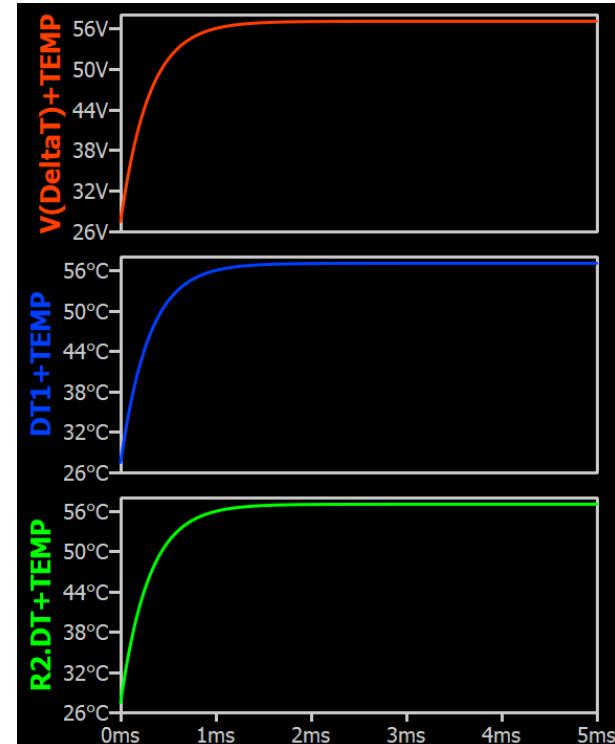
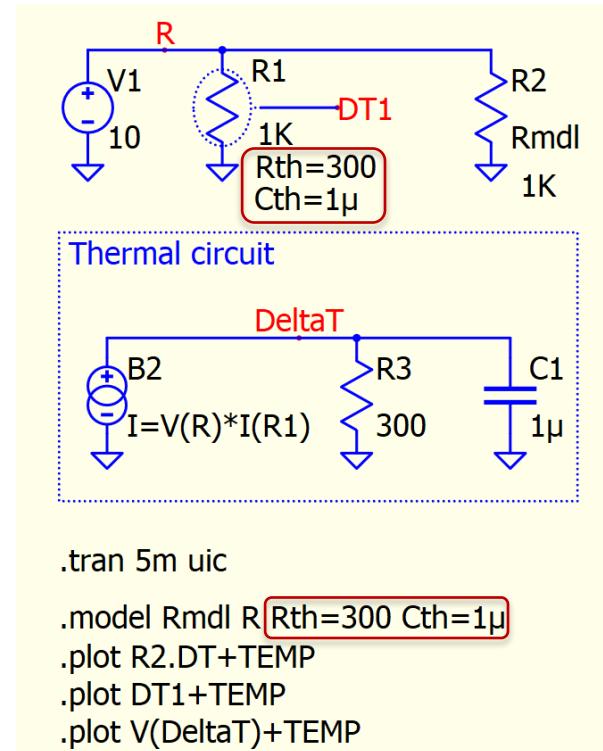


R. Resistor Params : Rth and Cth (Thermal Resistance and Capacitance)

Qspice : R - Rth Cth (Thermal Circuit).qsch

- RTH and CTH

- RTH : Thermal Resistance
- CTH : Thermal Capacitance
- **Default RTH=0**
- **Default CTH=0**
- Cth and Cth supports both instance (e.g. R1) and .model parameters (e.g. R2)
- DT (Delta Temperature ΔT)
 - DT is rise in temperature above ambient of resistor
 - 3-terminal resistor (3rd node in square braces) : ΔT is returned at 3rd node
 - 2-terminal resistor : ΔT is returned as Rnnn.DT

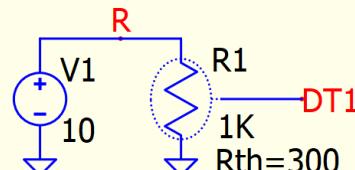


R. Resistor Params : Rth and Cth (Thermal Resistance and Capacitance)

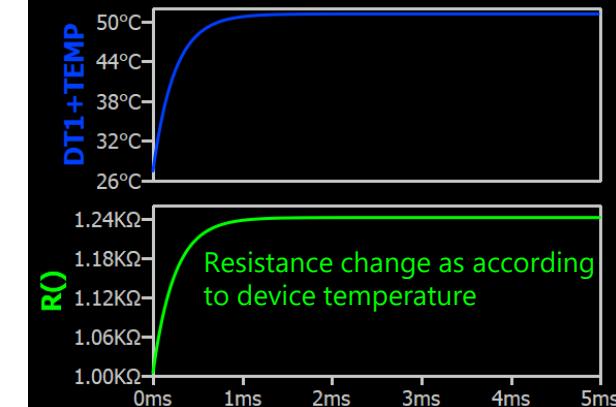
Qspice : R - Rth Cth (Self Heating).qsch

- Self Heating Resistor
 - This is an example for resistor with positive temperature coefficient
 - Resistance and temperature relationship is controlled by temperature coefficient TC. Rth and Cth determine thermal response to achieve self heating simulation

Self Heating Resistor Modeling

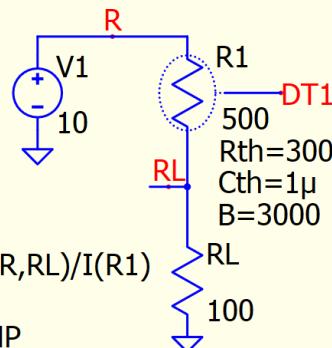


```
.tran 5m uic  
.func R() V(R)/I(R1)  
.plot R()  
.plot DT1+TEMP
```

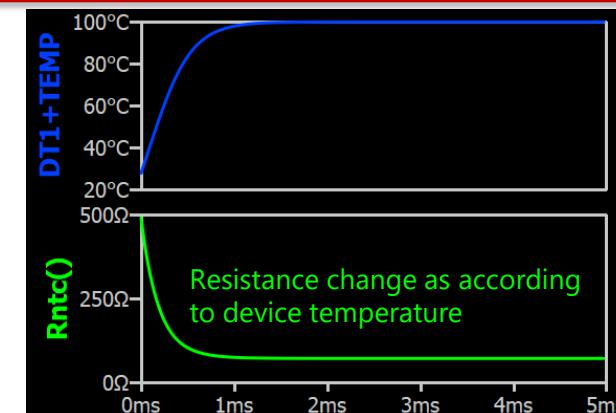


- Thermistor Self Heating
 - This example use NTC coefficient B to setup a thermistor for self heating simulation
 - A loading resistor is required to limit maximum current for simulation to reach steady state with using NTC

Self Heating Thermistor Modeling



```
.tran 5m uic  
.func Rntc() V(R,RL)/I(R1)  
.plot Rntc()  
.plot DT1+TEMP
```

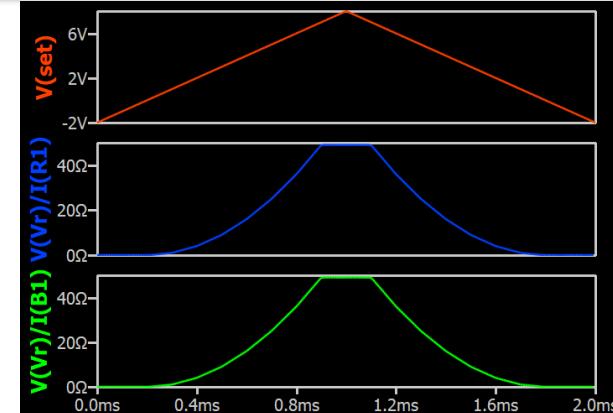
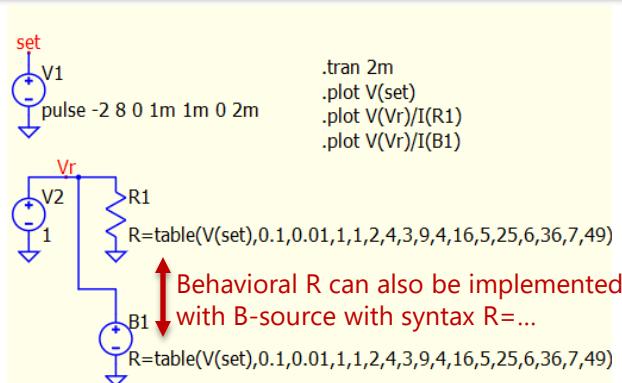
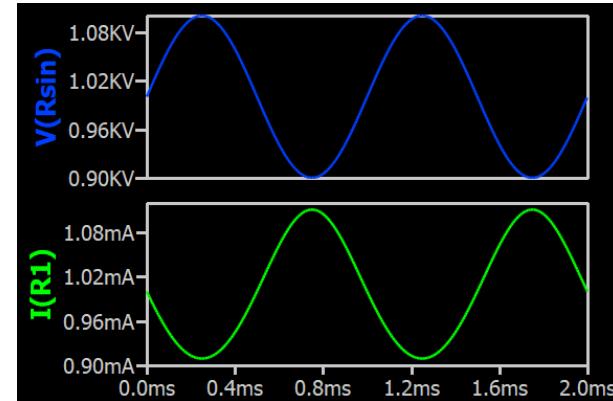
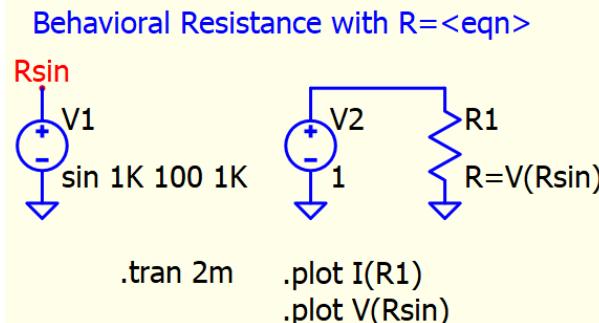


R. Resistor : Behavioral Resistor (R)

Qspice : R - Behavioral R Demo 01.qsch | R - Behavioral R Demo 02.qsch

- Behavioral Resistor

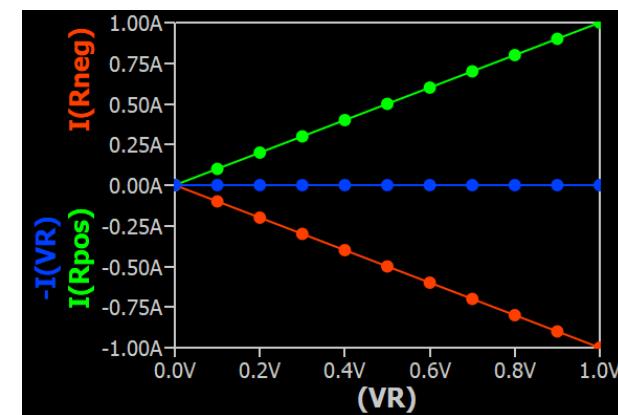
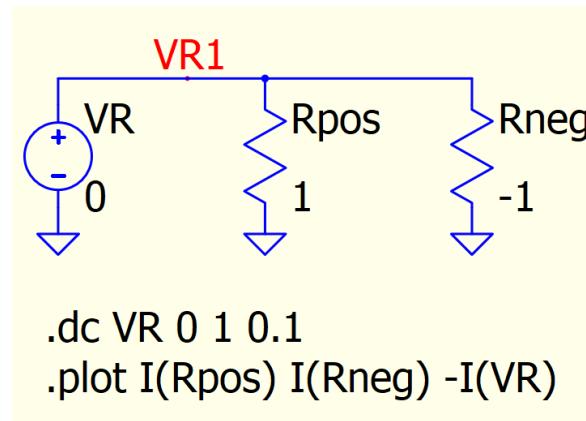
- To use formula or voltage/current node for resistance, type $R=...$ in $<\text{val}>$ of resistor
- Also support to use Behavioral source (B) for behavioral resistor
 - Replace $V=\text{expression}$ with $R=\text{expression}$



R. Resistor : Negative Resistance

Qspice : R - negative.qsch

- Negative Resistance
 - Qspice supports negative resistance with warning message returned in output window
 - Application example : negative resistor can be used in parallel a positive resistor to cancel the supply current



S. Voltage Controlled Switch

S. Voltage Controlled Switch Instance Parameter

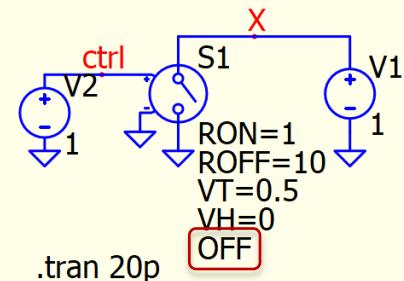
- S. Voltage Controlled Switch Instance Parameters
 - Syntax: Snnn N1 N2 NC+ NC- <model> [instance parameters]

Voltage Controlled Switch Instance Parameters

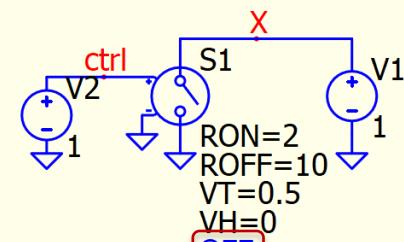
ETA ¹	Sub-/Over-Threshold conduction	V	model value
M	Number of parallel devices		1.0
OFF	Switch is initially off	Boolean	false
ON	Switch is initially on	Boolean	false
ROFF	Off resistance	Ω	model value
RON	On resistance	Ω	model value
TTOL	Temporal tolerance	sec	model value
VH	Hysteresis voltage	V	model value
VT	Threshold voltage	V	model value

S. Switch Instance Param – ON and OFF

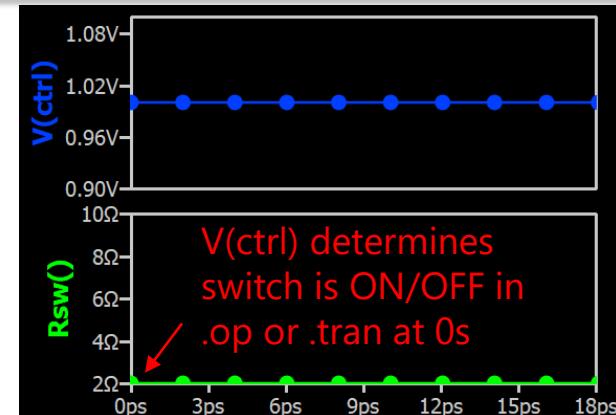
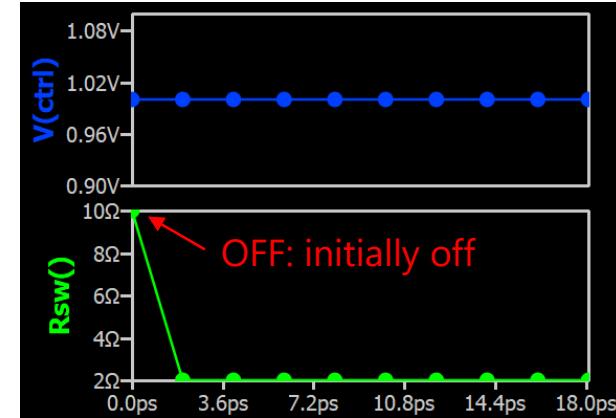
- ON and OFF
 - ON/OFF : Switch is initially ON or OFF
 - The behavior of switch is not depended on input control level in operating point (.op or .tran at 0s) if ON or OFF is assigned



```
.tran 20p  
.func Rsw() V(x)/-I(V1)  
.plot Rsw()  
.plot V(ctrl)
```



```
.tran 20p  
.func Rsw() V(x)/-I(V1)  
.plot Rsw()  
.plot V(ctrl)
```



S. Voltage Controlled Switch Model Parameters

- S. Voltage Controlled Switch Model Parameters
 - It is common to define Switch Model for usage as multiple switches in a schematic
 - Syntax: Snnn N1 N2 NC+ NC- <model> [instance parameters]
 - .model <model> **SW** [model parameters]
 - .model <model> **VSWITCH** [model parameters]

Voltage Controlled Switch Model Parameters

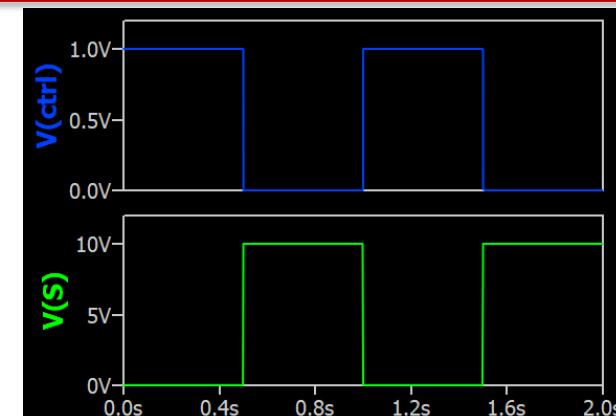
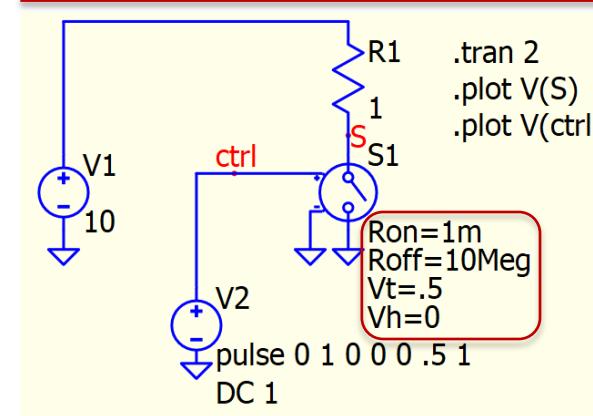
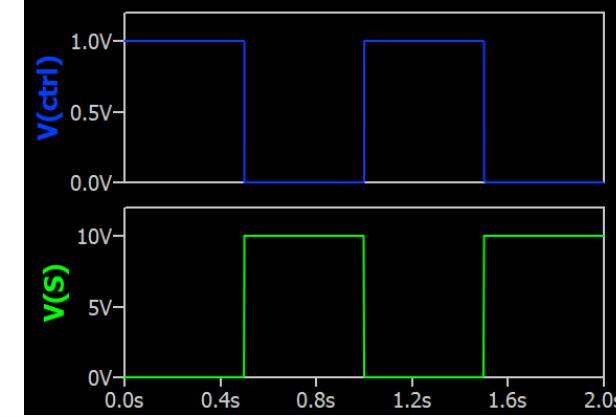
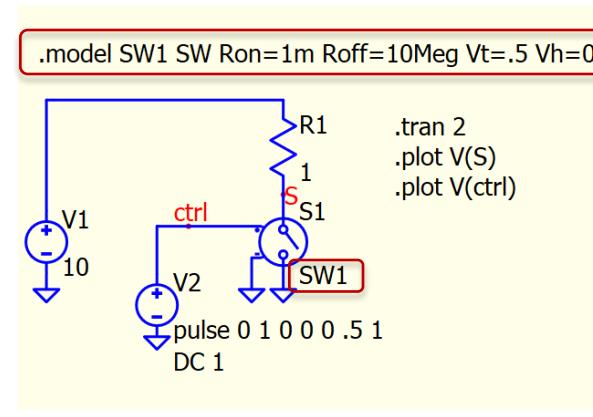
Name	Description	Units	Default
ETA ¹	Sub-/over-Threshold conduction	V	0.0
ROFF	Off resistance	Ω	1e6
RON	On resistance	Ω	1.0
TTOL	Temporal tolerance	sec	1e308
VH	Hysteresis voltage	V	0.0
VOFF	Voltage when open	V	0.0
VON	Voltage when closed	V	0.0
VT	Threshold voltage	V	0.0

S. Voltage Controlled Switch – How to Define a Switch

Qspice : Switch - Define1.qsch | Switch - Define2.qsch

- Define a Switch

- A switch must have R_{on} , R_{off} , V_t , and V_h defined by the user, as there are no default values for these parameters in Qspice

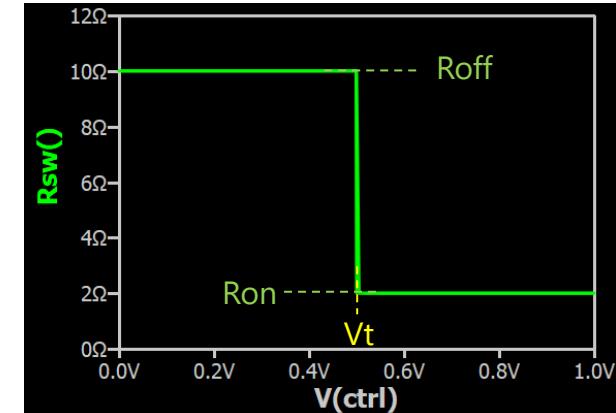


S. Switch Model Params : Ron, Roff, VT, Von and Voff

Qspice : Switch - Ron Roff VT VH.qsch | Switch - Ron Roff Von Voff.qsch

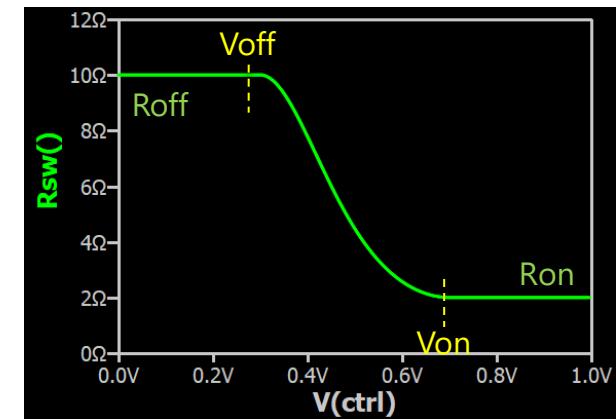
- RON, ROFF, VT, VH
 - Ron : On resistance
 - Roff : Off resistance
 - VT : Threshold voltage
 - **Default RON=1Ω**
 - **Default ROFF=1MegΩ**
 - **Default VT=0V**
 - Qspice must include VH

```
.model mSW SW Ron=2 Roff=10 Vt=0.5 VH=0
          SW
          S1
          mSW
ctrl
V1
I1
1
pulse 0 1 0 1 1 1 4
.tran 4
.func Rsw() V(sw)/I(I1)
.plot Rsw()
.plot V(ctrl)
```



- VON, VOFF
 - Von : Voltage when closed
 - Voff : Voltage when open
 - **Default VON=0V**
 - **Default VOFF=0V**
 - switch smoothly transitions between on and off
 - If VT and VH are specified, VON and VOFF are ignored

```
.model mSW SW Ron=2 Roff=10 Von=0.7 Voff=0.3
          SW
          S1
          mSW
ctrl
V1
I1
1
pulse 0 1 0 1 1 1 4
.tran 4
.func Rsw() V(sw)/I(I1)
.plot Rsw()
.plot V(ctrl)
```



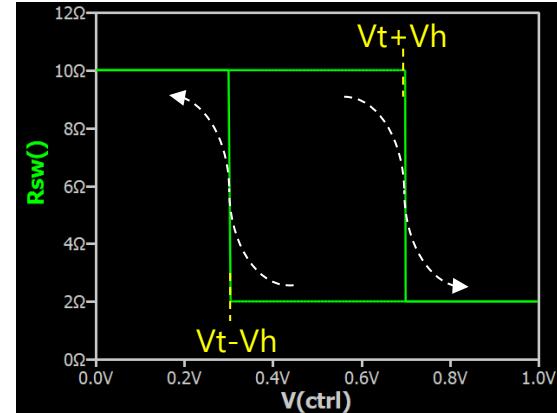
S. Switch Model Params : VH – Hysteresis and Smooth Transition

Qspice : Switch - VH.qsch

- VH
 - V_h : Hysteresis voltage
 - **Default VH=0V**
 - Switch is ON
 - $V(+)-V(-) > V_t + V_h$
 - Switch is OFF
 - $V(+)-V(-) < V_t - V_h$
 - Switch maintain its state
 - $V_t - V_h < V(+)-V(-) < V_t + V_h$

```
.model mSW SW Ron=2 Roff=10 Vt=0.5 Vh=0.2  
+ve Vh
```

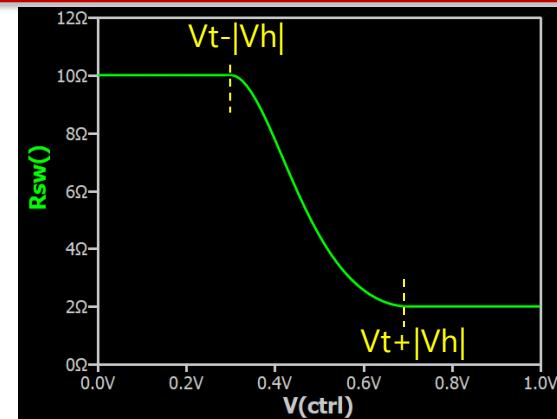
```
pulse 0 1 0 1 1 1 4  
.tran 4  
.func Rsw() V(sw)/I(I1)  
.plot Rsw()  
.plot V(ctrl)
```



- VH (-ve explained)
 - Negative V_h
 - If V_H is negative the switch smoothly transitions between on and off

```
.model mSW SW Ron=2 Roff=10 Vt=0.5 Vh=-0.2  
-ve Vh
```

```
pulse 0 1 0 1 1 1 4  
.tran 4  
.func Rsw() V(sw)/I(I1)  
.plot Rsw()  
.plot V(ctrl)
```



S. Switch Model Params : Von and Voff Relationship

Qspice : Switch - Ron Roff Von Voff.qsch | Switch - Ron Roff Von Voff (Inv).qsch

- **VON, VOFF**

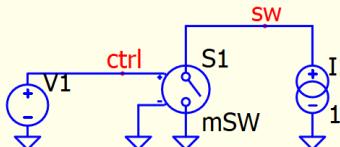
- For VON > VOFF

- Standard smooth transition switch pattern
 - Switch is OFF when input < Voff and Switch is ON when input > Von

- For VON < VOFF

- Switch pattern is inversed with smooth transition
 - Switch is OFF when input > Voff and Switch is ON when input < Von

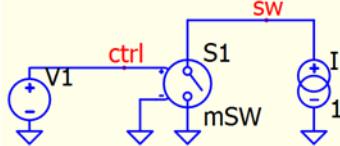
```
.model mSW SW Ron=2 Roff=10 Von=0.7 Voff=0.3
```



```
pulse 0 1 0 1 1 1 4
```

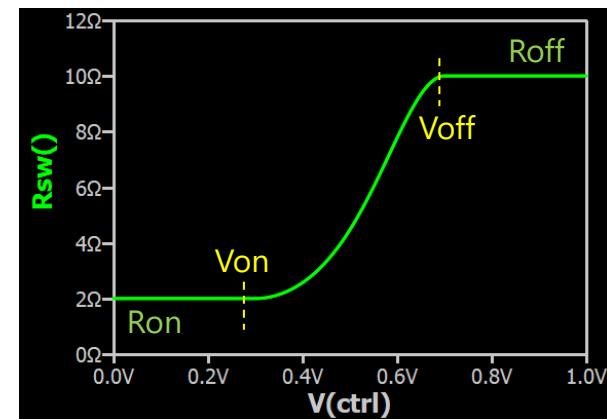
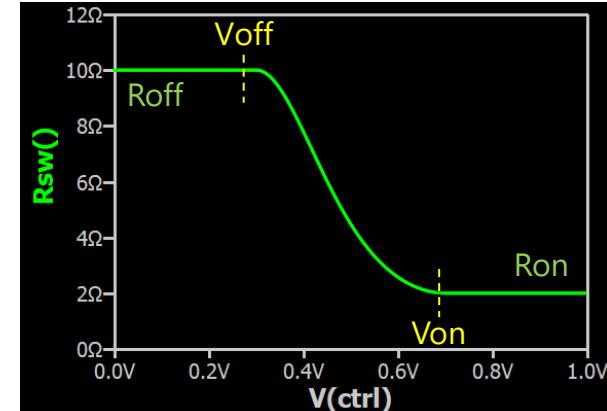
```
.tran 4  
.func Rsw() V(sw)/I(I1)  
.plot Rsw()  
.plot V(ctrl)
```

```
.model mSW SW Ron=2 Roff=10 Von=0.3 Voff=0.7
```



```
pulse 0 1 0 1 1 1 4
```

```
.tran 4  
.func Rsw() V(sw)/I(I1)  
.plot Rsw()  
.plot V(ctrl)
```



S. Switch Model Params : (VON, VOFF) vs (VT, VH)

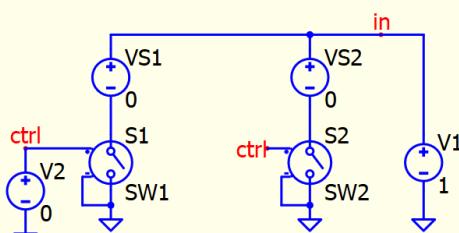
- Change between [VON, VOFF] and [VT, VH]
 - Important Note
 - VH must be negative for smooth transition
 - VON and VOFF is Pspice syntax which must be a smooth transition
 - For VON > VOFF
 - $VT = (VON + VOFF)/2$
 - $VH = |VON - VOFF|/2 * (-1)$: negative for smooth transition
 - For discrete switching action, remove the negative sign, but this is not an exact equivalent change.
 - For VON < VOFF
 - VT and VH : Same as above
 - Swap ROFF and RON

S. Switch Model Params : (VON, VOFF) vs (VT, VH)

Qspice : Switch - VH neg formula.qsch

- Formula in Smooth Transition Region

- $L_m = \ln(\sqrt{R_{ON} \times R_{OFF}})$, $L_r = \ln\left(\frac{R_{ON}}{R_{OFF}}\right)$, $V_m = \frac{V_{ON}+V_{OFF}}{2}$, $V_d = V_{ON} - V_{OFF}$
- For $V_{ON} > V_{OFF}$: $R_s = e^{L_m + \frac{3 L_r (V_c - V_m)}{2 V_d} - 2 L_r \left(\frac{V_c - V_m}{V_d}\right)^3}$
- For $V_{ON} < V_{OFF}$: $R_s = e^{L_m - \frac{3 L_r (V_c - V_m)}{2 V_d} + 2 L_r \left(\frac{V_c - V_m}{V_d}\right)^3}$

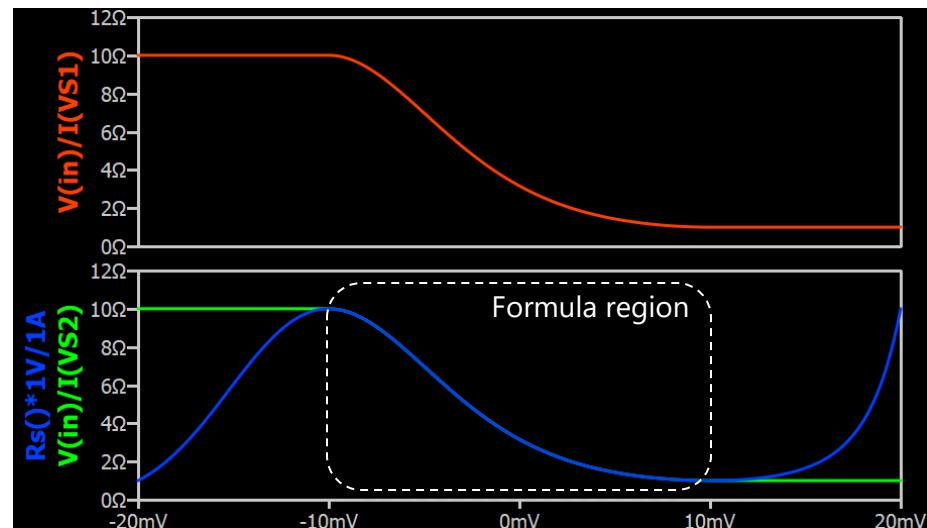


```

.model SW1 SW ROFF=10 RON=1 Vt=Vt Vh=Vh
.model SW2 SW ROFF=Roff RON=Ron VOFF=Voff VON=Von
.dc V2 -20m 20m 0.1m

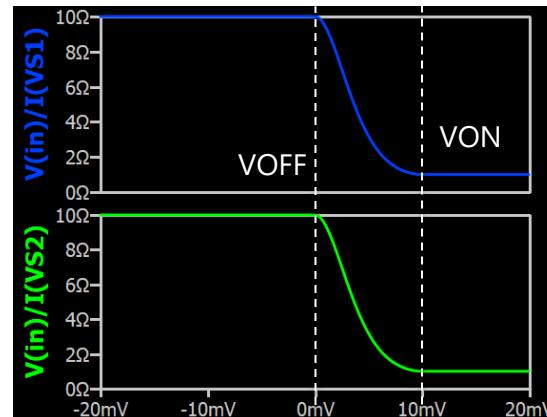
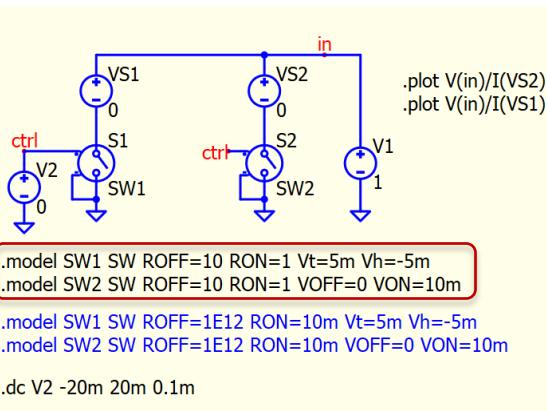
.func Lm() ln(sqrt(Ron*Roff))
.func Lr() ln(Ron/Roff)
.func Vm() (Von+Voff)/2
.func Vd() (Von-Voff)
.func Rs() exp(Lm() + 3*Lr()*(V(ctrl)-Vm())/(2*Vd()) - 2*Lr()*((V(ctrl)-Vm())/Vd())^3)

```

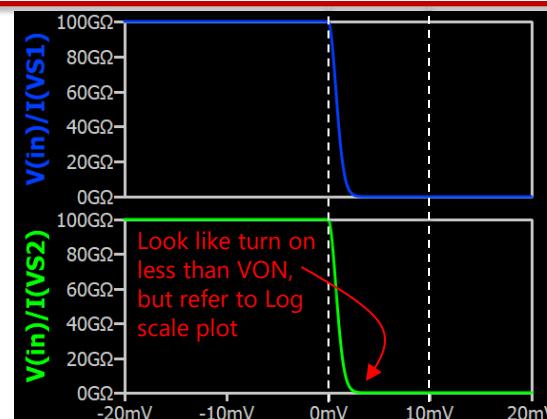
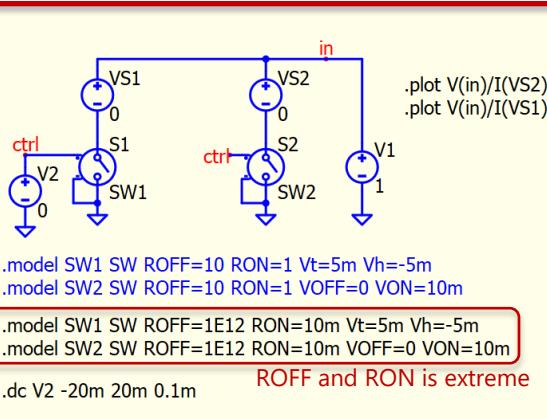


S. Switch Model Params : (VON, VOFF) vs (VT, VH)

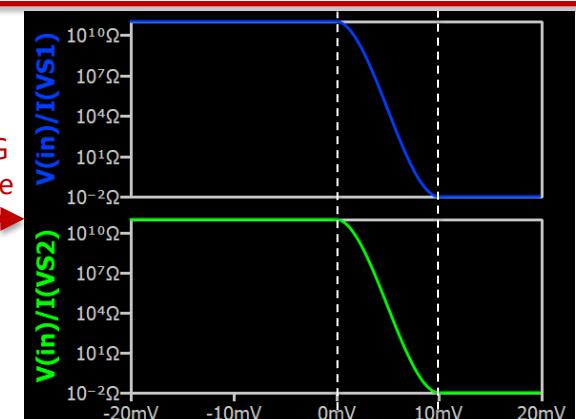
Qspice : Switch - Von Voff VT VH.qsch



- Two examples



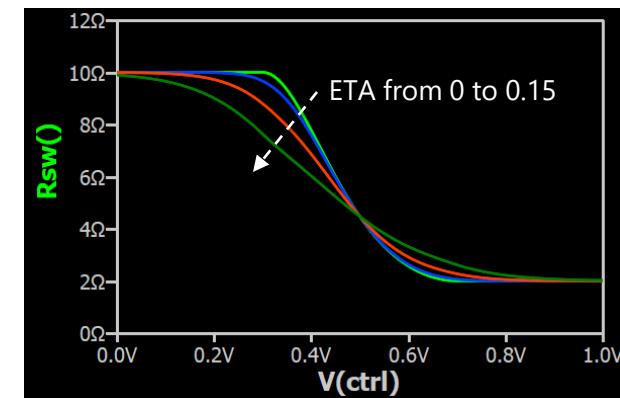
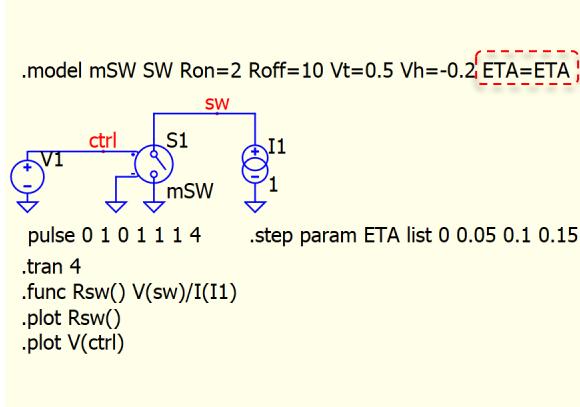
LOG Scale



S. Switch Model Params : ETA

Qspice : Switch - ETA.qsch

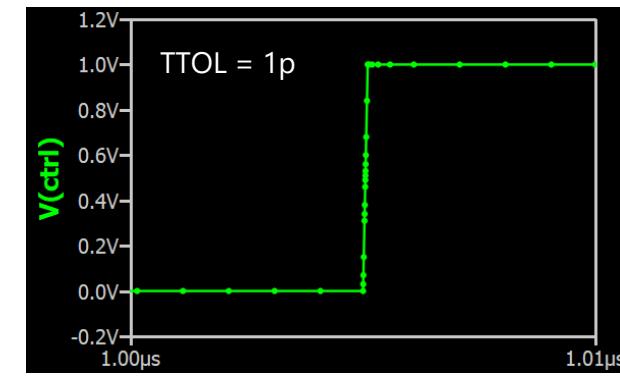
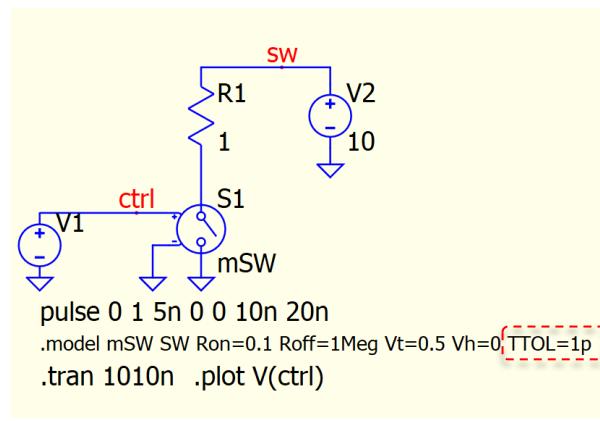
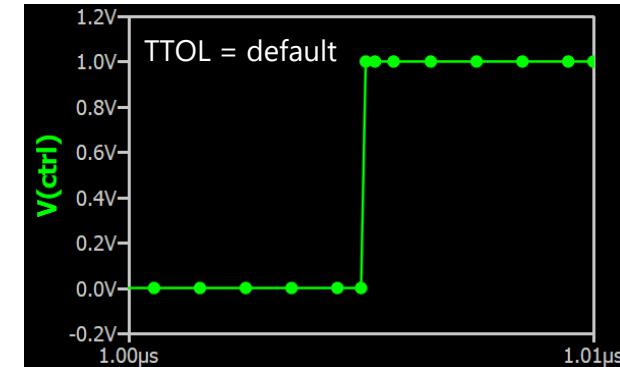
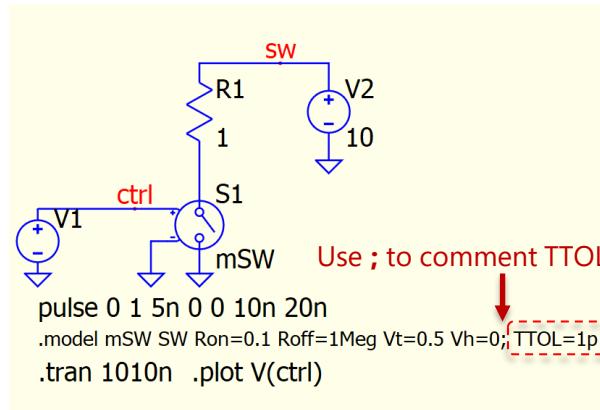
- **ETA**
 - ETA : Sub-/over- Threshold conduction
 - **Default ETA=0**
 - Prevents the control voltage Jacobian from vanishing
 - Only effective in smoothly transitions cases
 - Vt is negative
 - Von / Voff are used without Vt and VH



S. Switch Model Params : TTOL

Qspice : Switch - TTOL.qsch

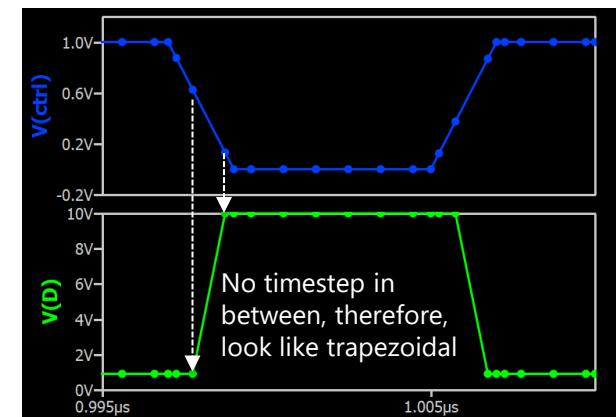
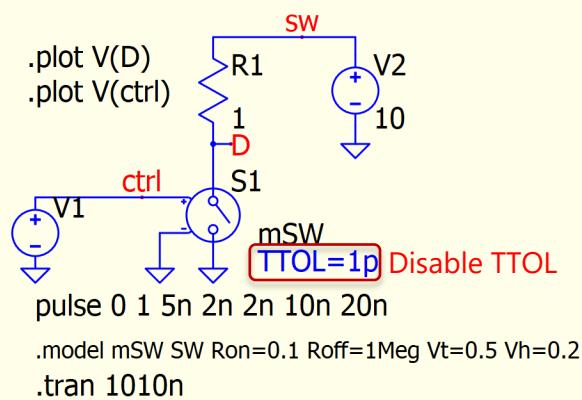
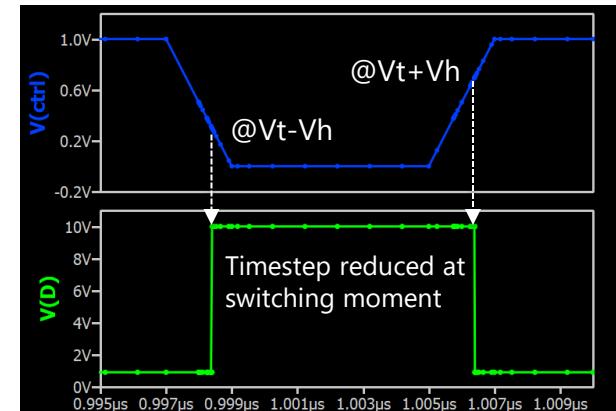
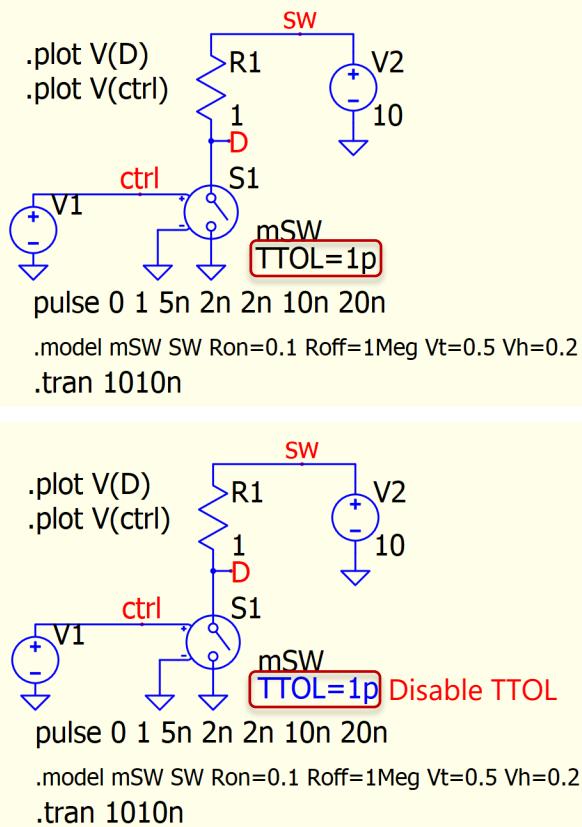
- TTOL
 - Ttol : Temporal tolerance
 - **Default TTOL=1e308s**
 - Equivalent to Disable TTOL
- Usage
 - TTOL allows one to determine how accurately the switch time should be found
 - It only affect timestep at switch time, which maintain simulation speed but improve accuracy at switching or logic action



S. Switch Model Params : TTOL (with VH hysteresis)

Qspice : Switch - TTOL with VH.qsch

- TTOL
 - Ttol : Temporal tolerance
 - **Default TTOL=1e308s**
- With Vh hysteresis
 - Extra timestep occurs according to switching action with hysteresis
 - ** TTOL may not response to negative Vh (-Vh) as switch is defined as smoothly transit instead of discrete state



T. Lossless Transmission Line

T. Lossless Transmission Line Instance Parameters

- T. Lossless Transmission Line
 - Syntax: Tnnn L+ L- R+ R- Zo=<value> Td=<value>

Lossless Transmission Line Instance Parameters

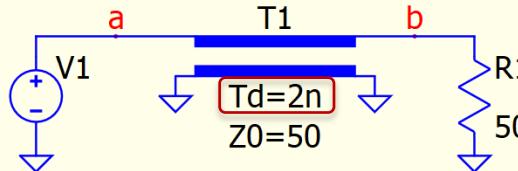
Name	Description	Units	Default
F ¹	Frequency	Hertz	1GHz
NL ¹	Normalized length at frequency given	wavelengths	1/4
TD	Transmission delay	sec	
Z0	Characteristic impedance(aka ZO)	Ω	50Ω

T. Instance Params : TD, F and NL

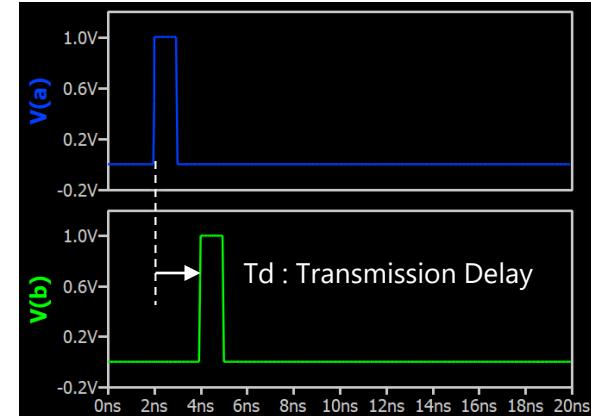
Qspice : Tline - Td.qsch ; Tline - F NL.qsch

- TD and Z0

- Td : Transmission delay
- Z0 : Characteristic Impedance
- **Default TD is not set**
- **Default Z0=50Ω**
- This simulation terminate transmission line with characteristic impedance Z0, no reflection of signal
- If TD is given, F and NL are ignored



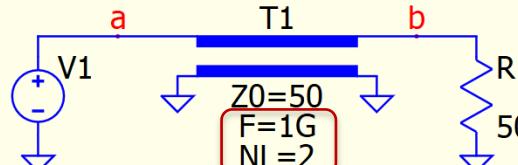
```
pulse 0 1 2n 0 0 1n 1 1  
.options MAXSTEP=1p  
.tran 20n  
.plot V(b)  
.plot V(a)
```



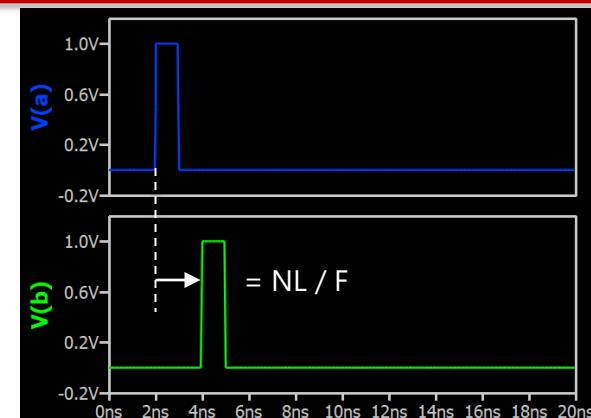
- F and NL

- F : Frequency
- NL : Normalized length at frequency given
- **Default F=1GHz**
- **Default NL=1/4**
- Equation

$$\begin{aligned} NL &= \frac{Length}{Wavelength} = \frac{Length}{\lambda} \\ \text{Period per } \lambda &: T = \frac{1}{F} \\ \text{Delay } T_d &= T \times \frac{Length}{\lambda} = \frac{NL}{F} \end{aligned}$$



```
pulse 0 1 2n 0 0 1n 1 1  
.options MAXSTEP=1p  
.tran 20n  
.plot V(b)  
.plot V(a)
```



T. Instance Params : TD – Fundamental Formula

- Calculate Transmission Delay (TD) of Coaxial Cable (Ideal Case)

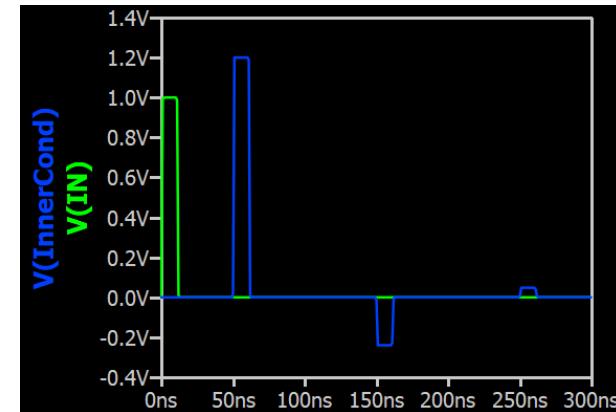
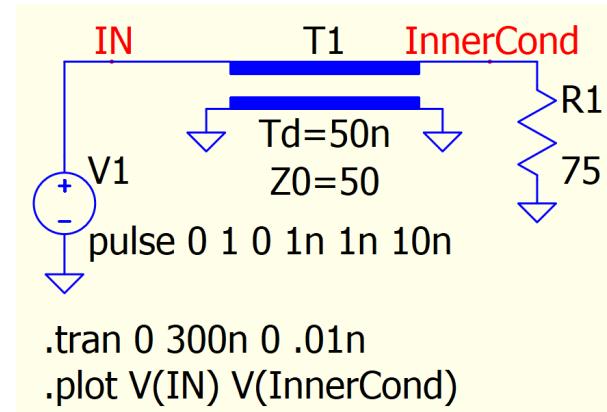
- $$TD = \frac{LEN_{physical}}{c \times VOP}$$

- TD : Transmission Delay
- c : Speed of Light = 3×10^8 m/s
- VOP : Velocity of Propagation, which depends on dielectric material
 - TFE : 0.69
 - Polyethylene : 0.66
- LEN_{physical} : Physical Length of coaxial cable (meter)

T. Concept – Common Misconception

Qspice : Tline - Common Misconception.qsch

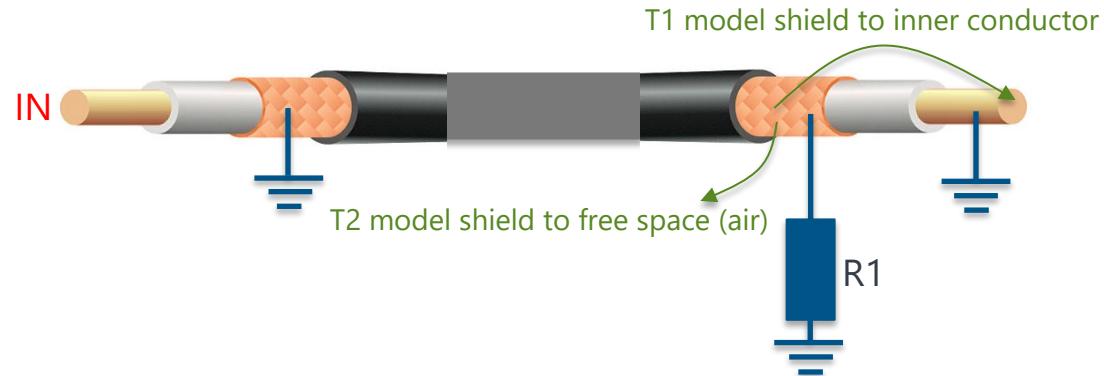
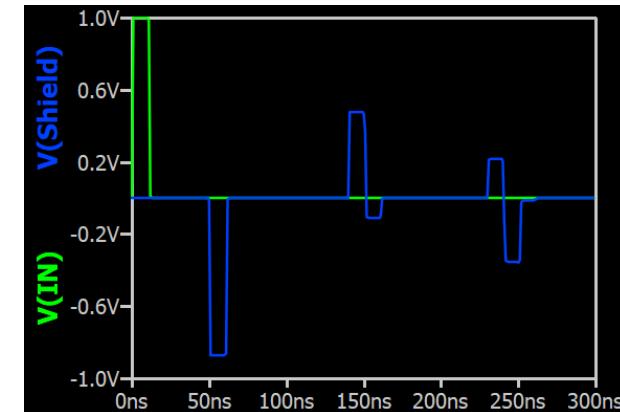
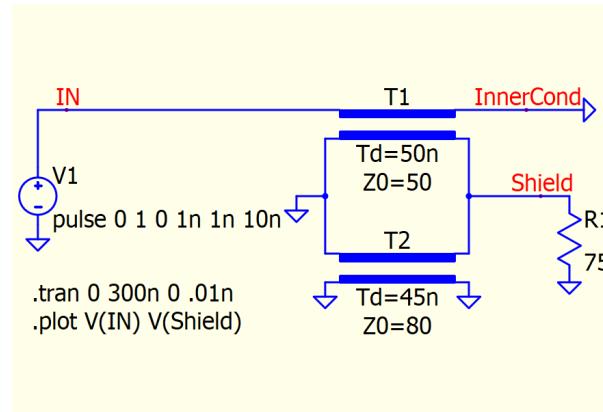
- Misconception
 - There is common misconception about the spice transmission line element
 - Neither Lossless nor Lossy transmission line models a length of coax
 - It only models one mode of the cable, i.e. internal node / inner conductor
 - That means usually both ends of the transmission line is ground



T. Example of Propagate with Shield

Qspice : Tline - Example - Propagate with Shield.qsch

- Propagate with Shield
 - Since a cable has many modes and wires, modeling a piece of coax requires two transmission line elements
 - Typically, different modes propagate at different speeds (through air instead of the cable dielectric), and are usually not uniform in real-world applications. This is why modeling other than the internal mode is usually meaningless



V. Voltage Source

V. Voltage Source - Instance Params

- V. Voltage Source
 - Syntax: Vnnn N+ N- <voltage> [Additional Instance Parameters]

Name	Description	Units	Default
AC	AC magnitude, optionally followed by phase angle	V, °	0.
ACMAG	AC Magnitude	V	0.
ACPHASE	AC Phase	°	0.
DC	DC source value	V	0.
EXP	Exponential source description		
LOG	Interpolate between PWL and CHIRP points		(not set)
NOISELESS	Ignore noise from RSER, if any.		(not set)
PULSE	Pulse description		
PWL	Piecewise linear description		
RSER	Internal series resistance		
SFFM	Single frequency FM description		
SINE	Sinusoidal source description(aka SIN)		
TD	Additional delay	s	0.
TIMECTRL	Time step control, one of NONE, LIMITS ¹ , BREAKS ² , BOTH	String	LIMITS
TIME_SCALE_FACTOR	Multiplies times, divides frequencies		1.
TRIGGER	Condition to start CHIRP, EXP, PULSE, PWL, or SFFM		
VALUE_SCALE_FACTOR	Multiplies voltages		1.
XTRAP	Extrapolate beyond PWL and CHIRP points		(not set)

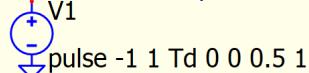
V. Voltage Source (Instance Params) : DC

Qspice : Vsource - DC (.tran).qsch | Vsource - DC (.op).qsch

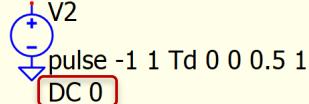
- DC
 - DC source value
 - This DC value is used in .dc analysis
 - Special Usage
 - For pulse in .tran, DC value can define voltage before Tdelay or at 0s
 - Without DC, this voltage equals <value1> in pulse
 - ** If without DC, .op in default to use source t=0 voltage as its DC solution

```
.tran 3  
.param Td=1  
.plot V(pulse-WithDC)  
.plot V(pulse-WithoutDC)
```

pulse-WithoutDC Special Usage in Pulse

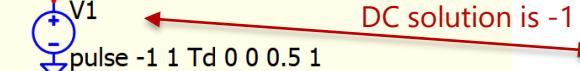


pulse-WithDC

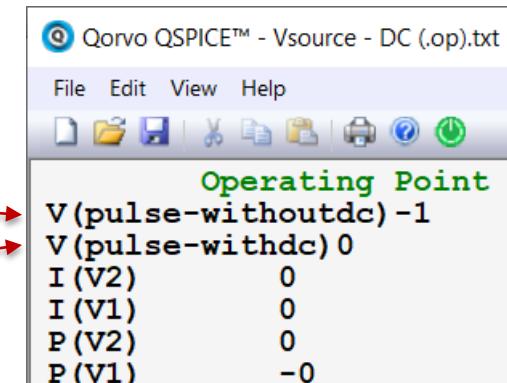
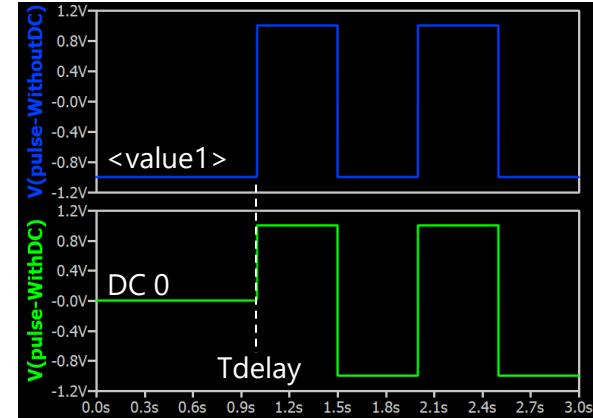
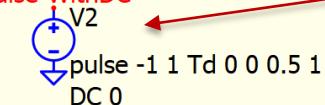


```
.op  
.param Td=1  
.plot V(pulse-WithDC)  
.plot V(pulse-WithoutDC)
```

pulse-WithoutDC Special Usage in Pulse



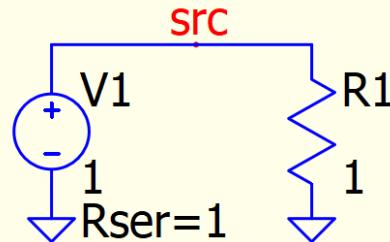
pulse-WithDC



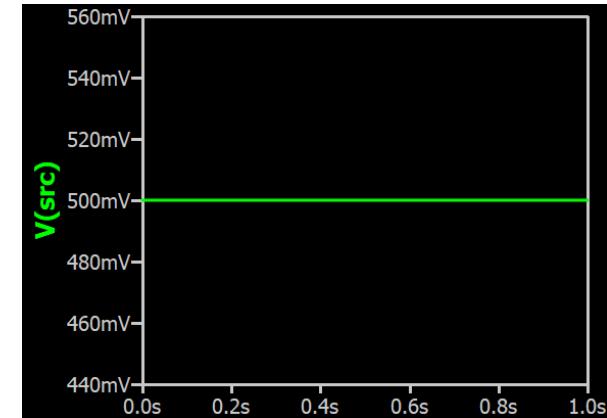
V. Voltage Source (Instance Params) : Rser

Qspice : Vsource - Rser.qsch

- Rser
 - Internal series resistance
 - **Default Rser=0**



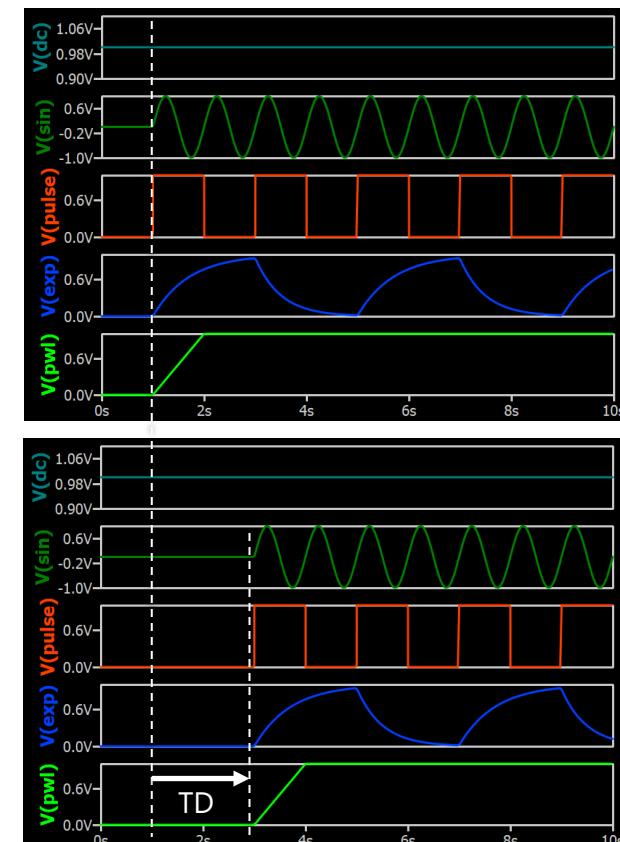
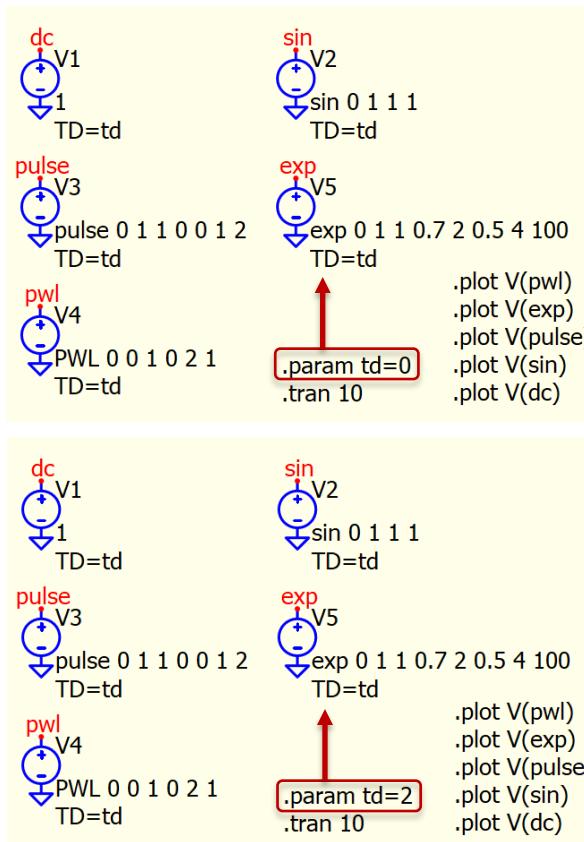
.tran 1
.plot V(src)



V. Voltage Source (Instance Params) : TD

Qspice : Vsource - TD.qsch

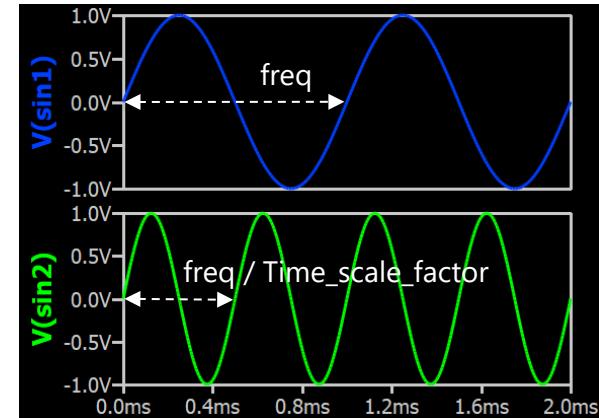
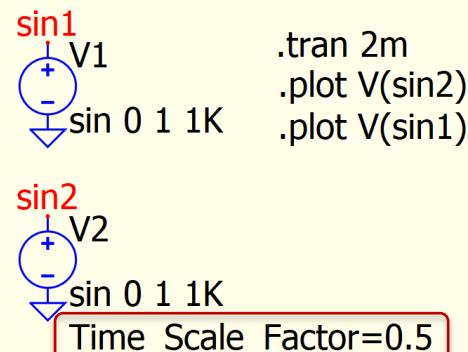
- TD
 - Additional Delay
 - **Default TD=0**
 - Additional delay for source with delay parameter (e.g. sin, pulse, exp, pwl etc...)
 - This doesn't apply to DC source



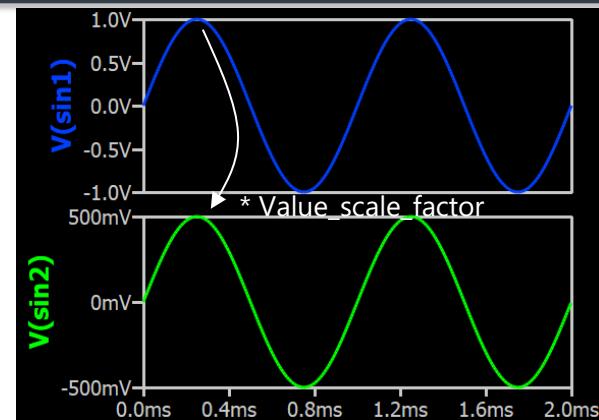
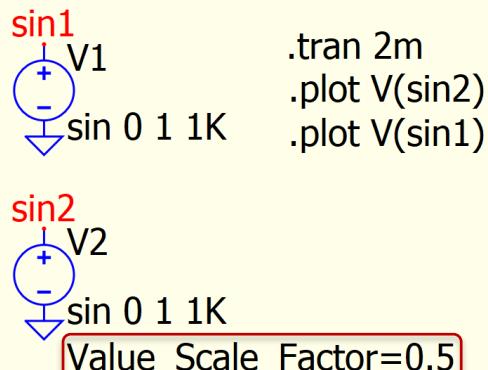
V. Voltage Source (Instance Params) : Time / Value_Scale_Factor

Qspice : Vsource - Time_Scale_Factor.qsch | Vsource - Value_Scale_Factor.qsch

- TIME_SCALE_FACTOR
 - Time_Scale_Factor : multiples times, divides frequency
 - Default Time_Scale_Factor=1



- VALUE_SCALE_FACTOR
 - Value_Scale_Factor : multiples voltages
 - Default Value_Scale_Factor=1



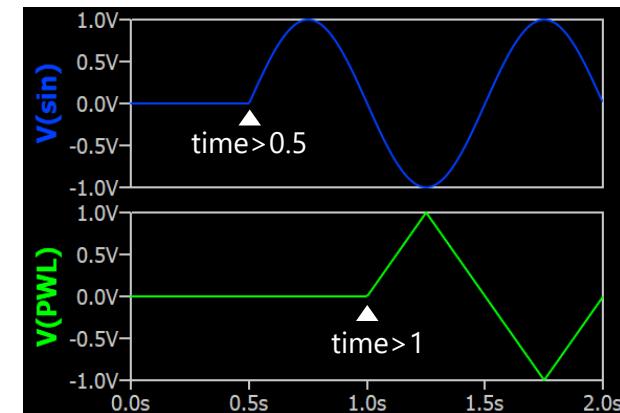
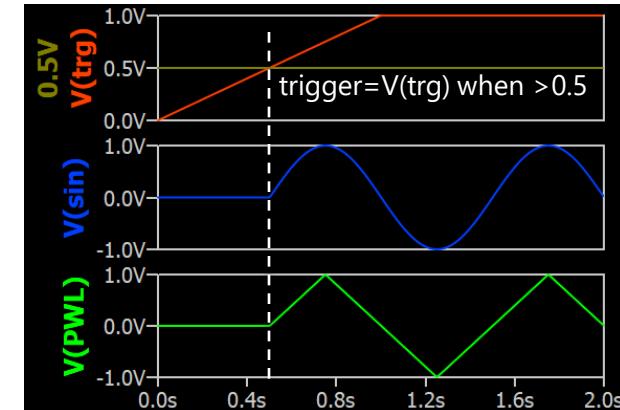
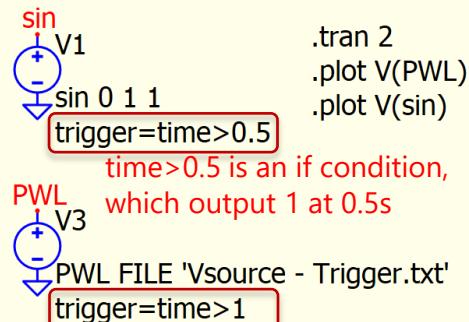
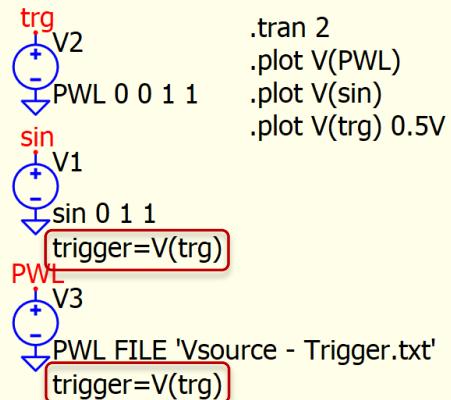
V. Voltage Source (Instance Params) : Trigger

Qspice : Vsource - Trigger - 01 Basic.qsch | Vsource - Trigger - 02 Time fcn.qsch

- **TRIGGER**

- Trigger : Condition to start the SINE, PWL, PULSE, EXP or SFFM wave (aka TRIG)
- If function defined in Trigger > 0.5 (rising edge trigger), sources start to output
- *Vsource-Trigger.txt* in this example

```
Vsource - Trigger.txt
1 0 0
2 0.25 1
3 0.75 -1
4 1.25 1
5 1.75 -1
6 2.25 1
7 2.5 0
8
```

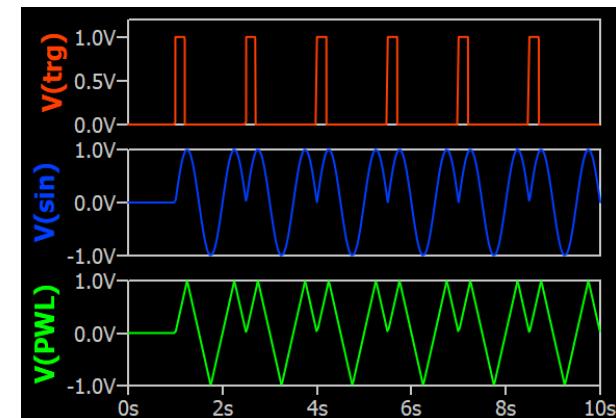
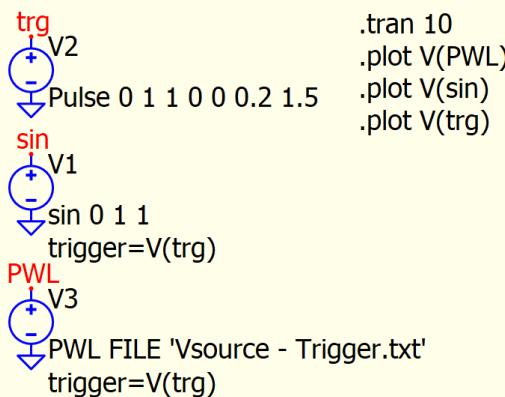
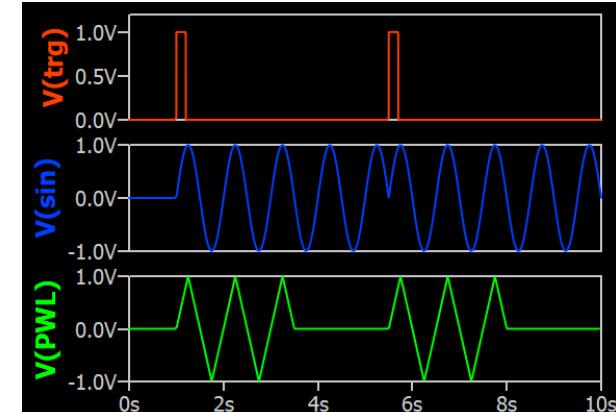
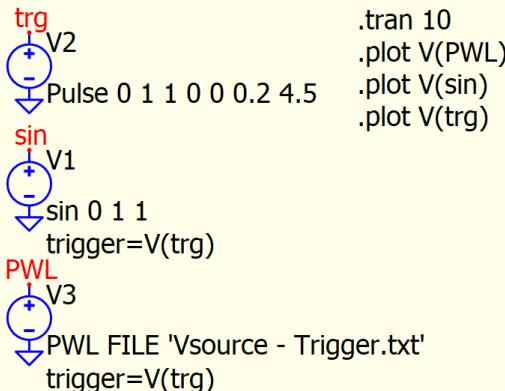


V. Voltage Source (Instance Params) : Trigger

Qspice : Vsource - Trigger - 03 Multiple.qsch

- **TRIGGER**

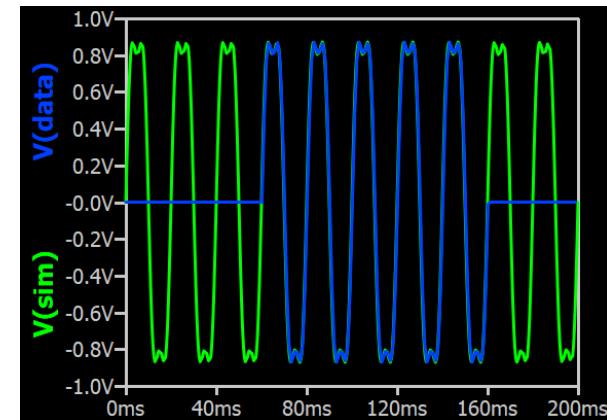
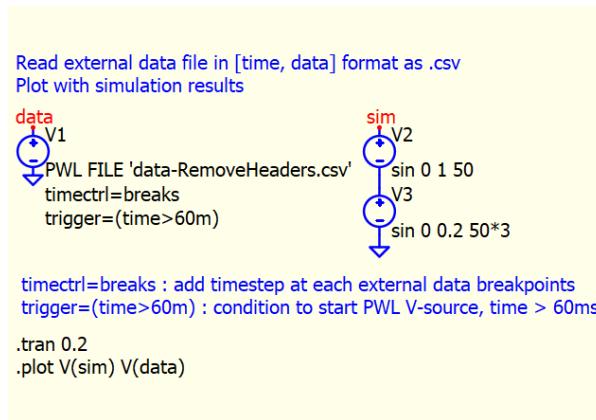
- Re-trigger the source at each trigger rising edge
- Trigger will not turn off the source once its triggered, only for re-trigger



V. Voltage Source (Instance Params) : Trigger – Data Overlap

Qspice : \Vsource Trigger PWL File\Vsource - Trigger - PWL File.qsch

- Trigger with PWL File
 - An application of the Trigger is to overlay external data with simulated results, aligning the external data in time with the simulated results for comparison
 - In this example, a V-source with a **PWL FILE** is utilized to load a 2-column .csv data file
 - **Trigger** is employed to initiate the source output at a user-defined timestamp



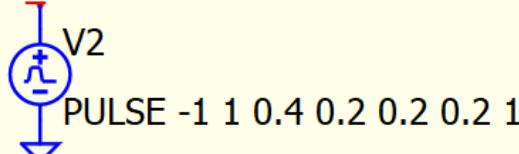
	data-RemoveHeaders.csv
1	0, 0
2	0.001, 0.470821351
3	0.002, 0.777997472
4	0.003, 0.870820431
5	0.004, 0.833498947
6	0.005, 0.8
7	0.006, 0.833500243
8	0.007, 0.870820305
9	0.008, 0.77799289
10	0.009, 0.470811773
11	0.01, -1.17543E-05
12	0.011, -0.470830929
13	0.012, -0.778002053
14	0.013, -0.870820557

V. Voltage Source : PULSE (Pulse)

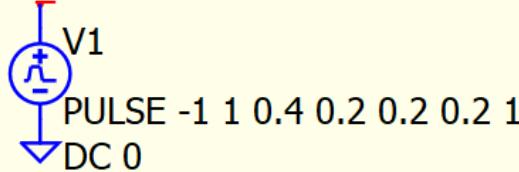
Qspice : Vsource PULSE.qsch

Syntax : PULSE V1 V2 Td Trise Tfall Ton Tperiod

pulse_woDC

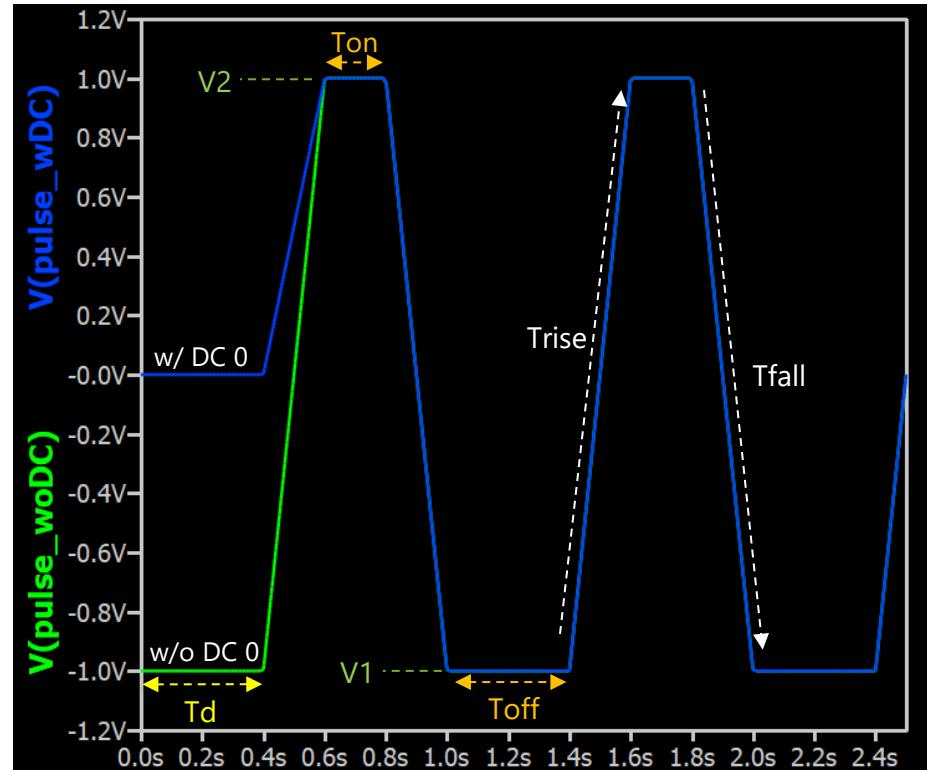


pulse_wDC



.tran 2.5

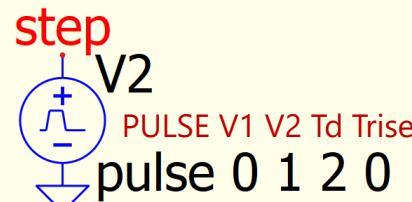
.plot V(pulse_woDC) V(pulse_wDC)



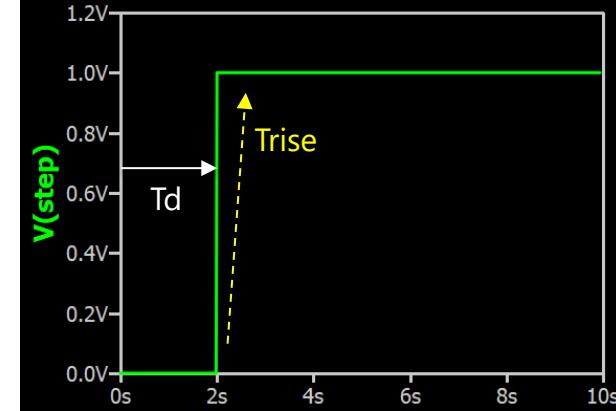
V. Voltage Source : PULSE (Pulse) : Usage

Qspice : Vsource PULSE-Step.qsch / Vsource PULSE-SinglePulse.qsch

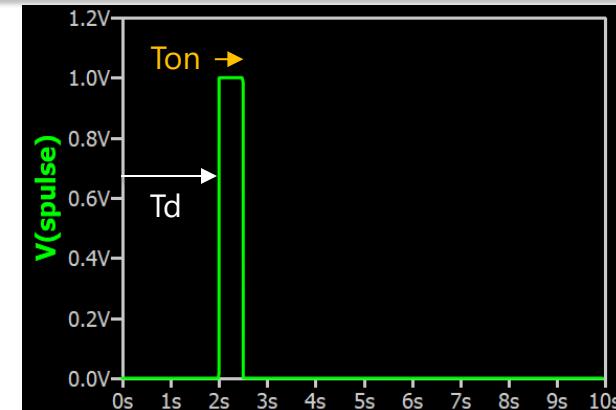
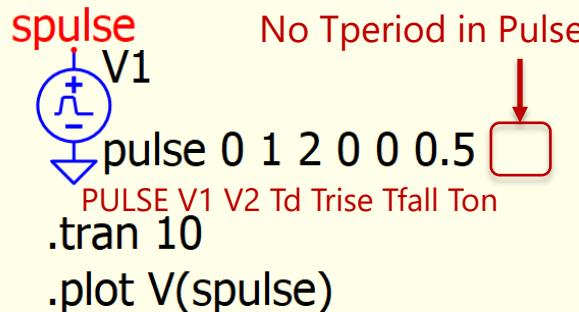
- Step with PULSE
 - To generate a step, only input value to **Trise**



.tran 10
.plot V(step)



- Single Pulse with PULSE
 - To generate a single non-repeating pulse, does not specify **Tperiod** in pulse statement

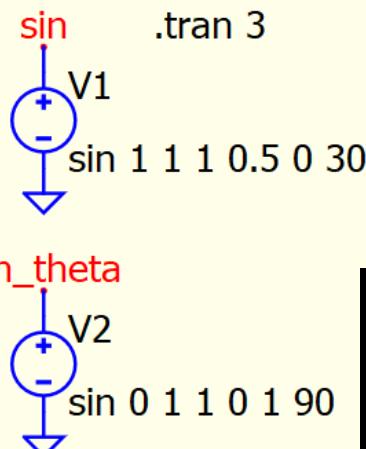


V. Voltage Source : SIN (Sine Wave)

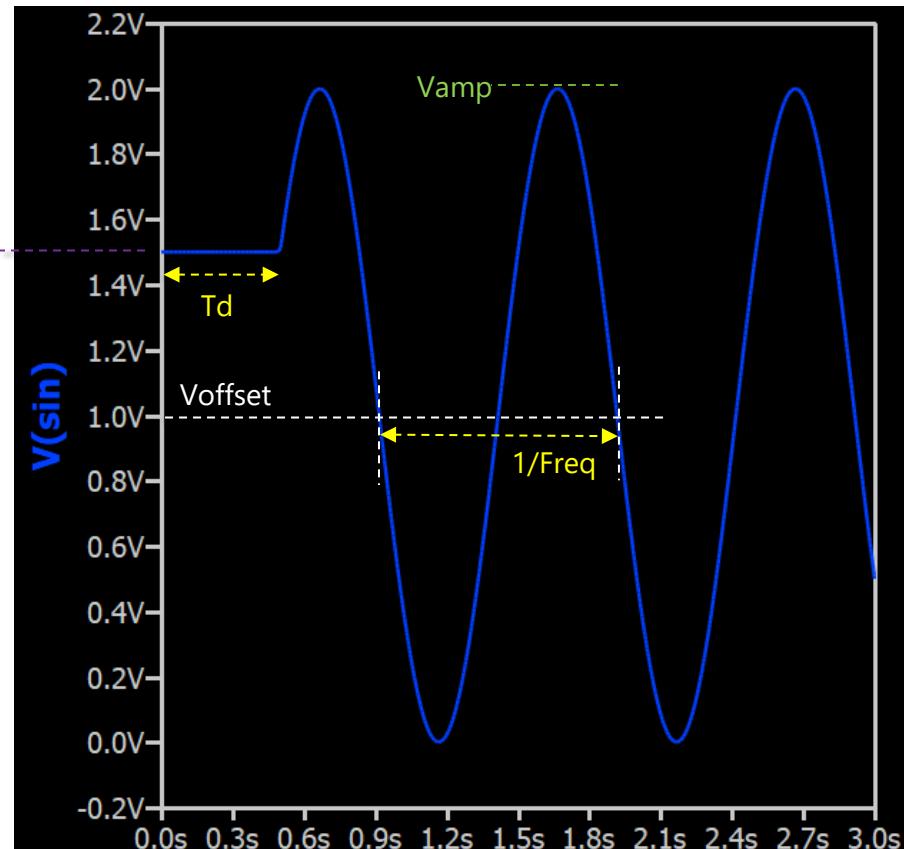
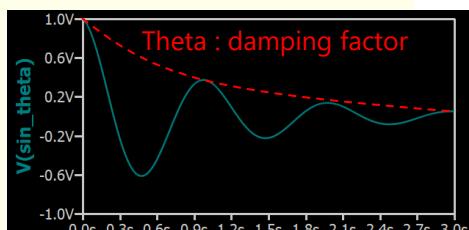
Qspice : Vsource SIN.qsch

$$\text{Phi} : \text{level} = V_{\text{offset}} + \sin(\text{Phi}) \\ \text{e.g.: } = 1 + \sin(30^\circ) = 1.5$$

Syntax : SIN Voffset Vamp Freq Td Theta Phi



Amplitude drops by
 $e^{-1} \sim 0.378$ in $1/\Theta$ seconds



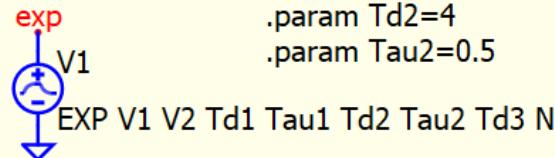
V. Voltage Source : EXP (Exponential)

Qspice : Vsource EXP.qsch

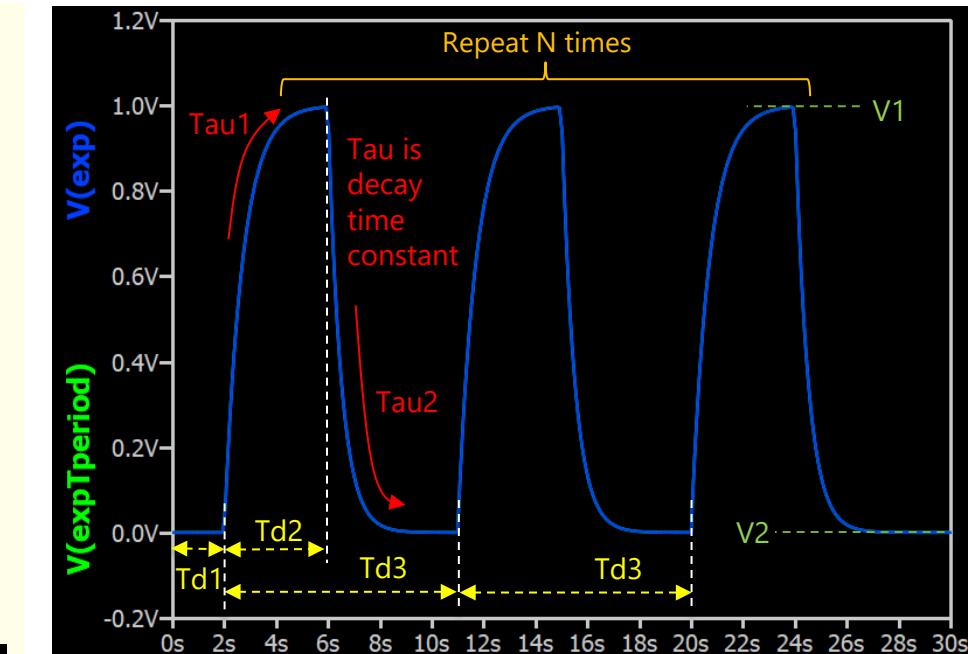
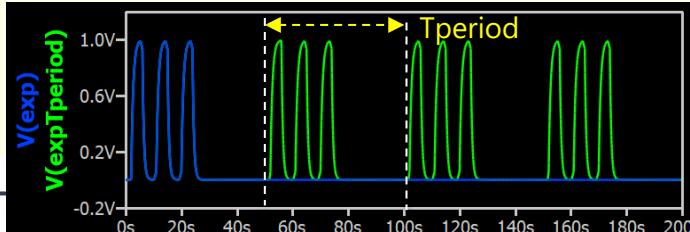
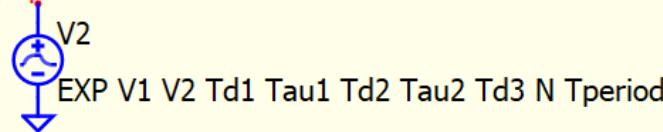
```
.plot V(expTperiod) V(exp)
```

Syntax : EXP V1 V2 Td1 Tau1 Td2 Tau2 Td3 N Tperiod

```
.tran 200          .param V1=0          .param Td3=9  
.param V2=1          .param N=3  
.param Td1=2          .param Tperiod=50  
.param Tau1=0.7  
.param Td2=4  
.param Tau2=0.5
```



expTperiod



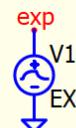
V. Voltage Source : EXP (Exponential) – N

Qspice : Vsource EXP-N.qsch

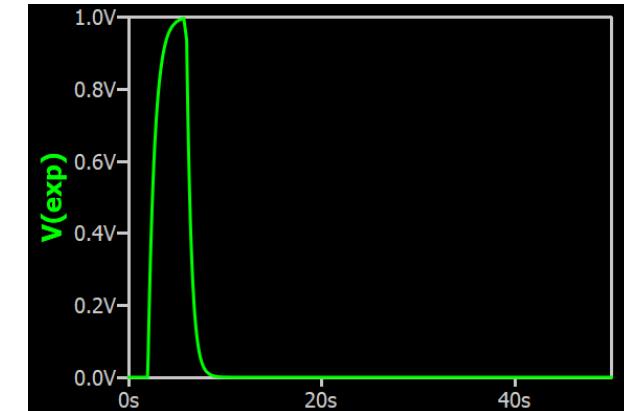
- EXP – N
 - N is Number of times repeated in cycles
 - If N is omitted or set to 0, it force N=1
 - ** EXP will not automatically repeat forever

Syntax : EXP V1 V2 Td1 Tau1 Td2 Tau2 Td3 N Tperiod

```
.tran 200          .param V1=0      .param Td3=9  
.plot V(exp)       .param V2=1      .param N=0  
                   .param Td1=2  
                   .param Tau1=0.7  
                   .param Td2=4  
                   .param Tau2=0.5
```



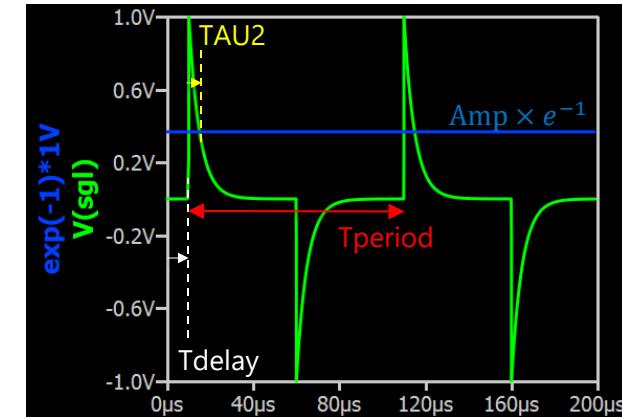
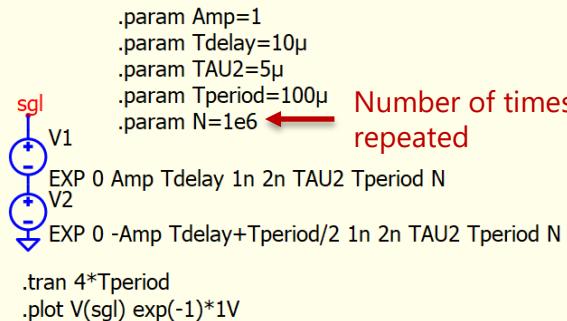
EXP V1 V2 Td1 Tau1 Td2 Tau2 Td3 N



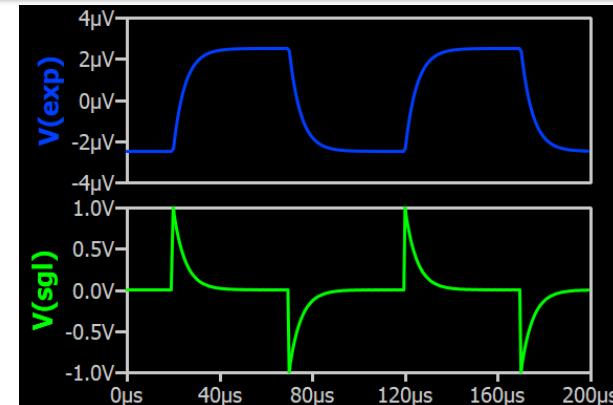
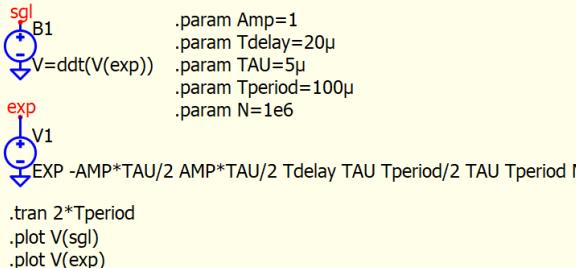
V. Voltage Source : EXP (Exponential) – Derivative Profile

Qspice : Vsource EXP-DualSrc | Vsource EXP-DDT.qsch

- EXP (derivative)
 - Two exponential sources can be used to generate this exponential derivative signal



- EXP (derivative)
 - Use Behavioral source with derivative function ddt() to process exponential source output

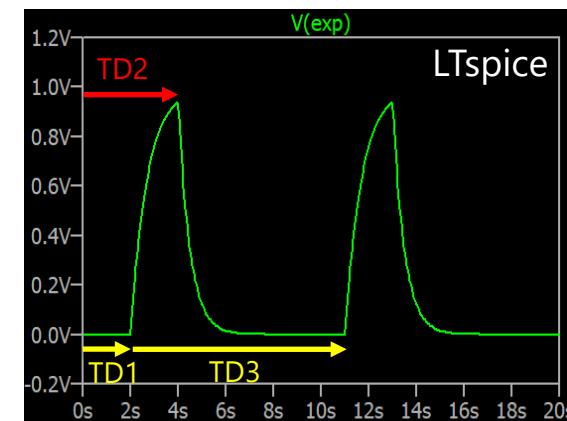
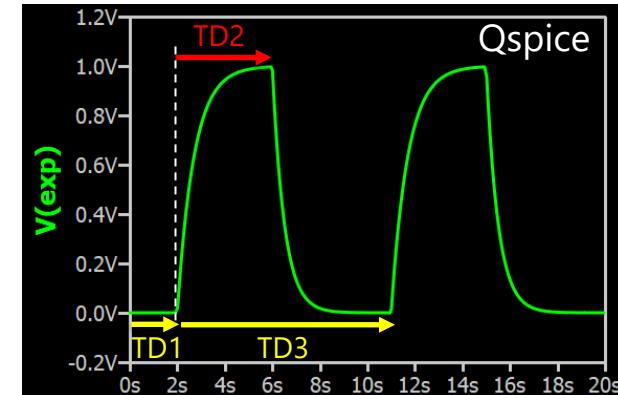


V. Voltage Source : EXP (Exponential) – LTspice vs Qspice

Qspice : Vsource EXP-CompareSPICE.cir

- LTspice vs Qspice
 - Definition of EXP are different between SPICE
 - TD2 : LTspice begin its count from 0s, and Qspice begin by follow TD1
 - N : If omitted in LTspice but with TD3, EXP source in default repeat forever
 - To convert LTspice syntax into Qspice for EXP
 - Assume you have a LTspice EXP source
 - TD2 (Qspice) = TD2 - TD1
 - If N is omitted in LTspice, force a large value for N, for example, N=1e6

```
* Netlist to Run in SPICE
V1 exp 0 EXP {V1} {V2} {Td1} {Tau1} {Td2} {Tau2} {Td3} {N}
.tran 20
.param V1=0
.param V2=1
.param Td1=2
.param Tau1=0.7
.param Td2=4
.param Tau2=0.5
.param Td3=9
.param N=2
.end
```



V. Voltage Source : SFFM (Single Frequency FM)

Qspice : Vsource SFFM.qsch

- SFFM
 - Single Frequency FM
 - It is hard to observe FM signal in time domain waveform
 - FFT is used to explain its nature

Syntax : SFFM Voff Vamp Fcar MDI Fsig

SFFM



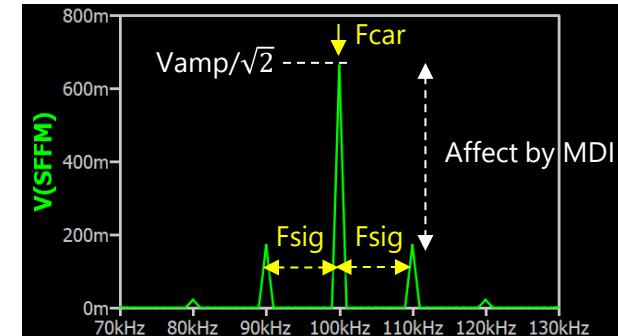
V1

SFFM 0 1 100K 0.5 10K

.tran 10/10K

.plot V(SFFM)

FFT magnitude is RMS

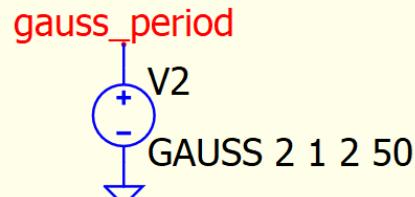
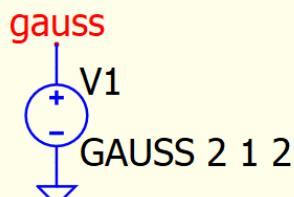


- Transient formula
 - $V_{off} + V_{amp} \cdot \sin(2\pi \cdot F_{car} \cdot time) + MDI \cdot \sin(2\pi \cdot F_{sig} \cdot time)$

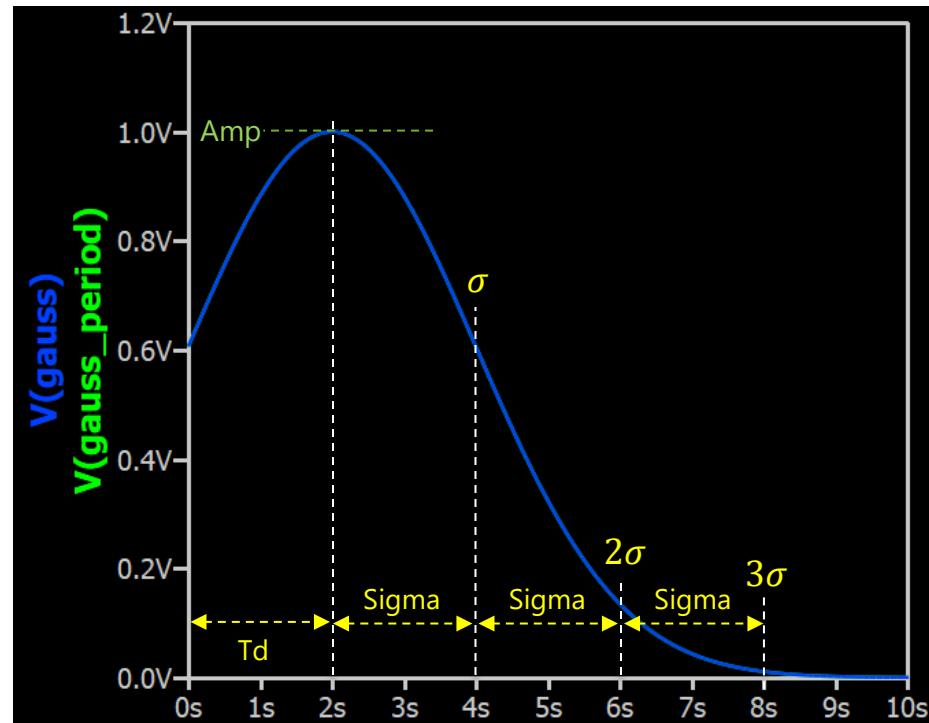
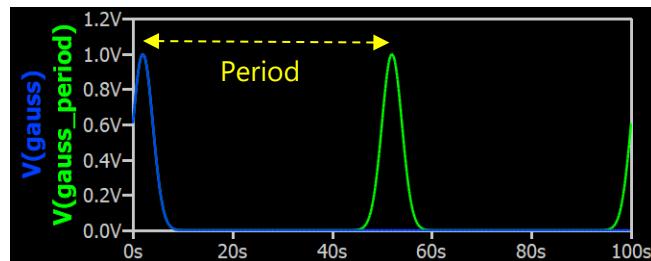
V. Voltage Source : GAUSS (Gaussian Pulse)

Qspice : Vsource GAUSS.qsch

Syntax : GAUSS Td Amp Sigma [Period]



```
.plot V(gauss_period) V(gauss)
.tran 100
```

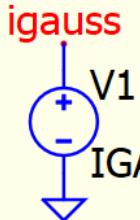


V. Voltage Source : IGAUSS (Imaginary Gaussian Pulse)

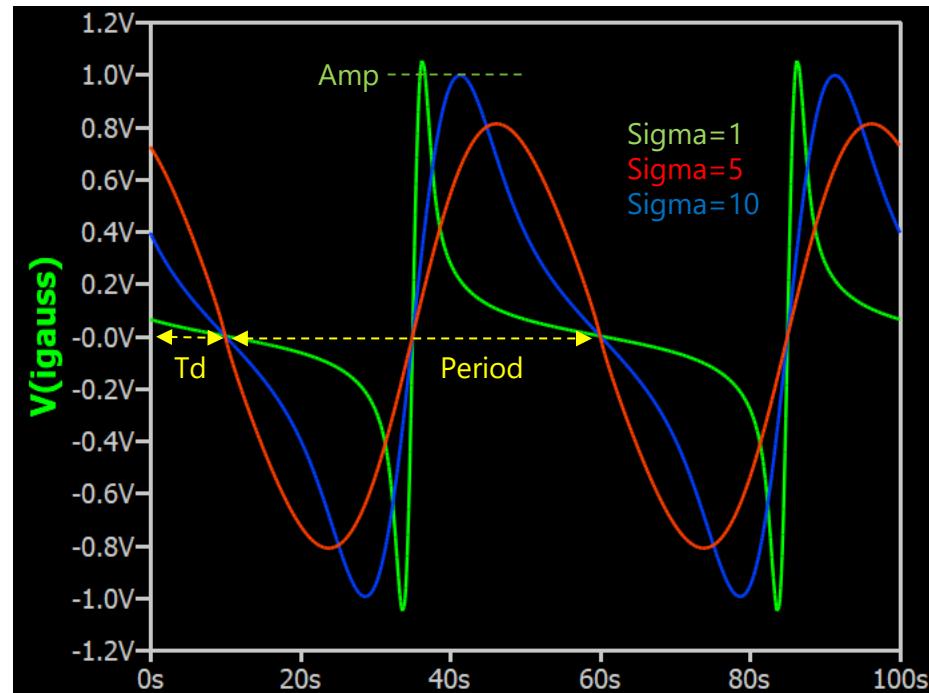
Qspice : Vsource IGAUSS.qsch

Syntax : IGAUSS Td Period Sigma Amp

```
.step param Sigma list 1 5 10
```



```
.plot V(igauss)  
.tran 100
```



V. Voltage Source : CHIRP (Piece-wise Linear Chirp)

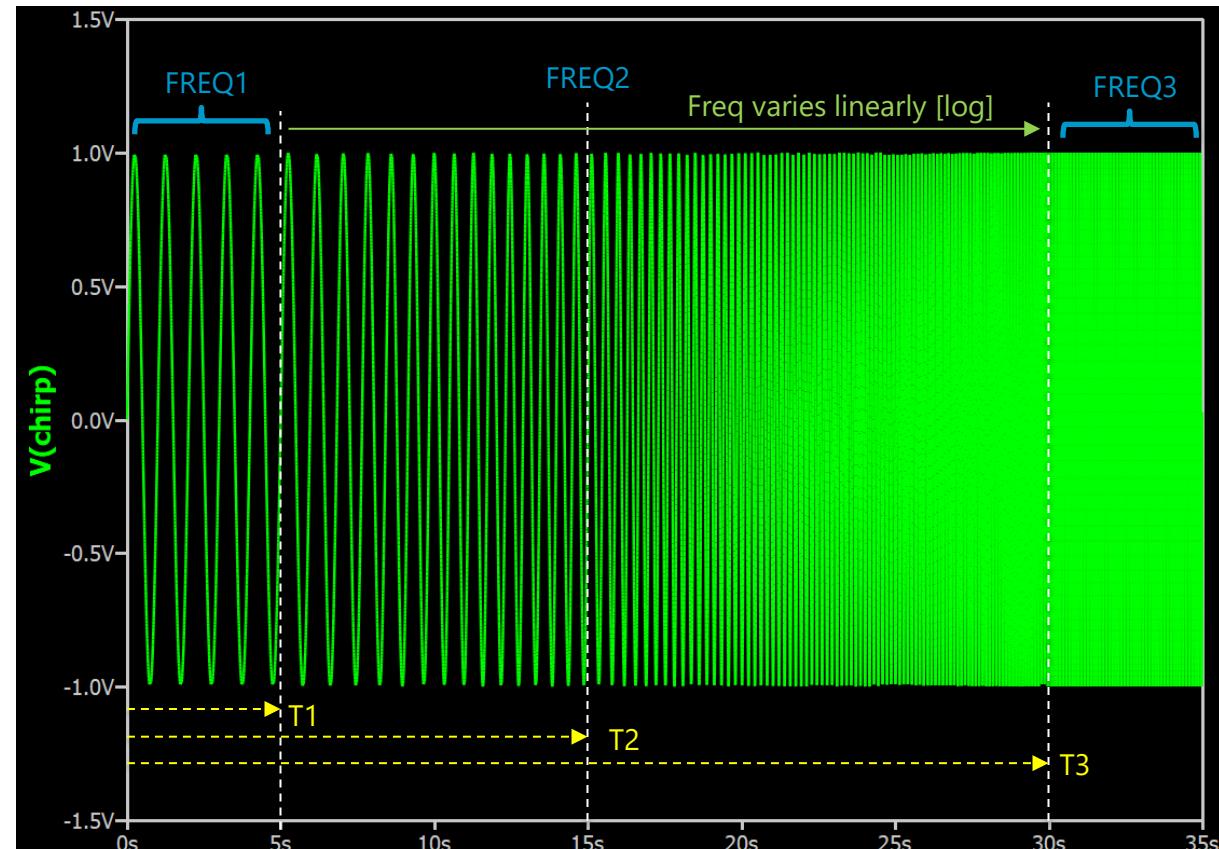
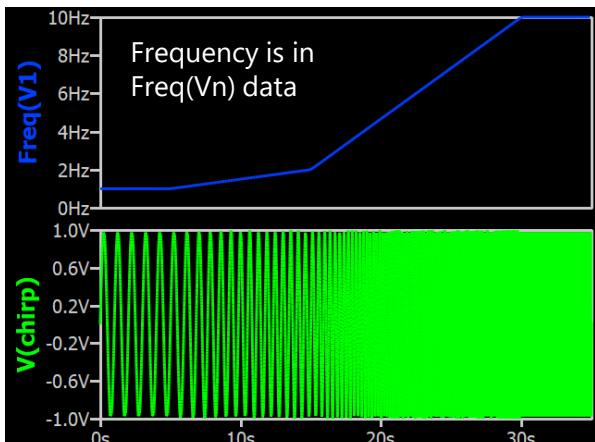
Qspice : Vsource Chirp.qsch

Syntax : CHIRP(AMP T1 FREQ1 T2 FREQ2 [...]) [LOG] [XTRAP]

```
.param AMP=1  
.param T1=5  
.param FREQ1=1  
.param T2=15  
.param FREQ2=2  
.param T3=30  
.param FREQ3=10  
  
chirp  
V1  
-  
CHIRP AMP T1 FREQ1 T2 FREQ2 T3 FREQ3
```

```
.plot V(chirp)  
.tran 35  
.options MAXSTEP=1/1000
```

XTRAP : extrapolate beyond PWL and CHIRP points



V. Voltage Source : CHIRP (Piece-wise Linear Chirp) – REPEAT

Qspice : Vsource CHIRP-Repeat.qsch

- CHIRP Repeat Syntax

- Vnnn N+ N- CHIRP **REPEAT FOREVER** t1 f1 t2 f2 t3 f3... **ENDREPEAT**
- Vnnn N+ N- CHIRP **REPEAT FOR N** t1 f1 t2 f2 t3 f3... **ENDREPEAT**
- N is number of times
- ** last voltage value of a repeated CHIRP will be ignored and forced to be equal to the first value

Syntax :

```
CHIRP [REPEAT] [FOR N] (AMP T1 FREQ1 T2 FREQ2 [...]) [LOG] [XTRAP] [ENDREPEAT]  
CHIRP [REPEAT] [FOREVER] (AMP T1 FREQ1 T2 FREQ2 [...]) [LOG] [XTRAP] [ENDREPEAT]
```

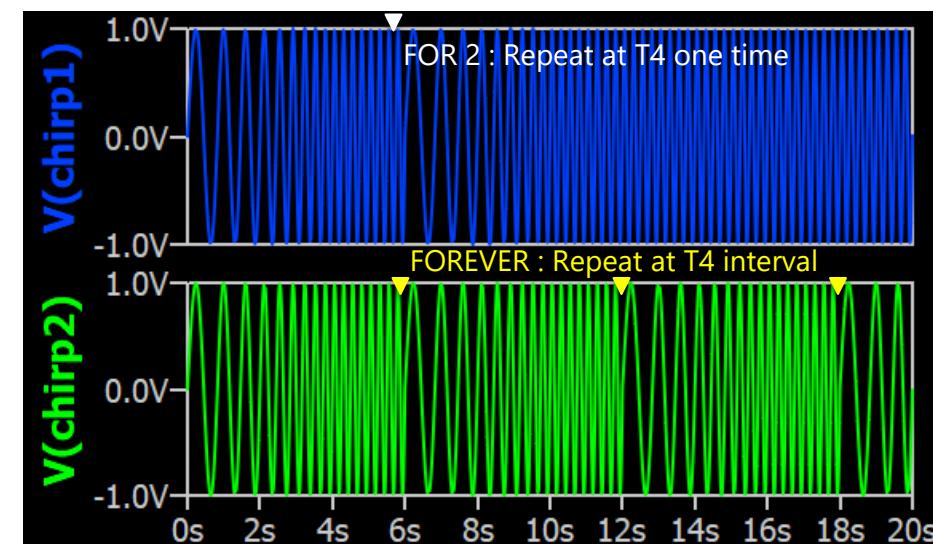
```
.param AMP=1          .plot V(chirp2)  
.param T1=0            .plot V(chirp1)  
.param FREQ1=1  
.param T2=2            .tran 20  
.param FREQ2=2          .options MAXSTEP=1/1000  
.param T3=4  
.param FREQ3=4  
.param T4=6  
.param FREQ4=4
```

chirp1
V1

```
CHIRP REPEAT FOR 2 AMP T1 FREQ1 T2 FREQ2 T3 FREQ3 T4 FREQ4 ENDREPEAT
```

chirp2
V2

```
CHIRP REPEAT FOREVER AMP T1 FREQ1 T2 FREQ2 T3 FREQ3 T4 FREQ4 ENDREPEAT
```



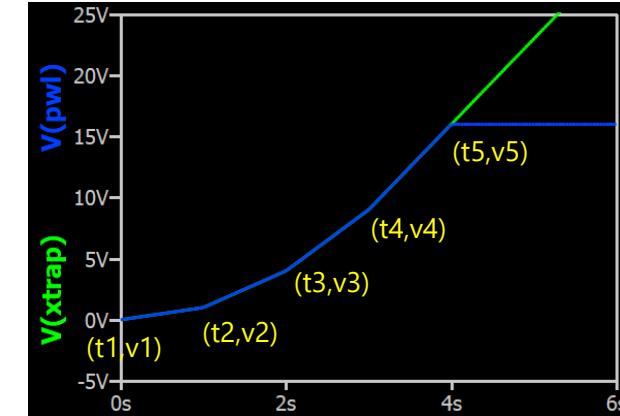
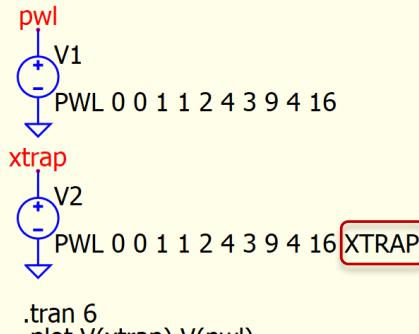
V. Voltage Source : PWL (Piece-wise Linear) – XTRAP and LOG

Qspice : Vsource PWL-Xtrap.qsch/ Vsource PWL-Log.qsch

- Basic Usage and XTRAP

- Standard
 - last value will hold
- Xtrap
 - Extrapolate beyond PWL and CHIRP points

Syntax : PWL(t1 v1 t2 v2 t3 v3...) [LOG] [XTRAP]

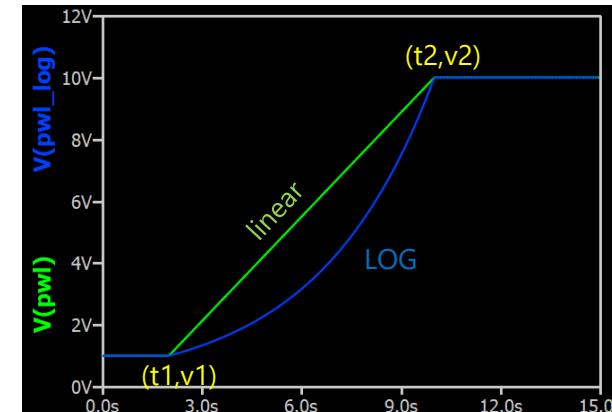
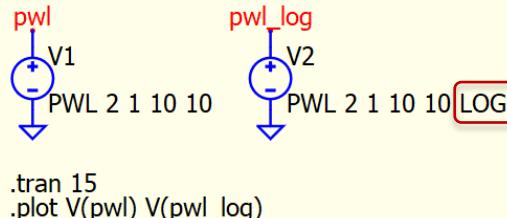


- LOG

- Log

- Interpolate between PWL and CHIRP points with log function

Syntax : PWL(t1 v1 t2 v2 t3 v3...) [LOG] [XTRAP]



V. Voltage Source : PWL (Piece-wise Linear) – REPEAT

Qspice : Vsource PWL-Repeat.qsch

- PWL Repeat Syntax

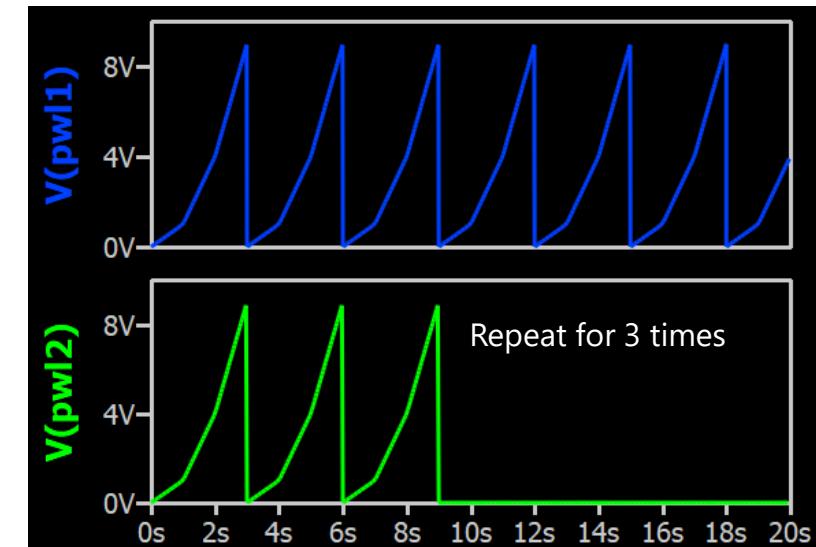
- Vnnn N+ N- PWL **REPEAT FOREVER** t1 v1 t2 v2 t3 v3... **ENDREPEAT**
- Vnnn N+ N- PWL **REPEAT FOR N** t1 v1 t2 v2 t3 v3... **ENDREPEAT**
 - N is number of times
 - ** last voltage value of a repeated PWL will be ignored and forced to be equal to the first value**

Syntax : PWL FOR[EVER] [times] (t1 v1 t2 v2 t3 v3...) Endrepeat

pwl1
V1
PWL REPEAT FOREVER (0 0 1 1 2 4 3 9 3+1p 0) Endrepeat

pwl2
V2
PWL REPEAT FOR 3 (0 0 1 1 2 4 3 9 3+1p 0) Endrepeat

.tran 20
.plot V(pwl2)
.plot V(pwl1)

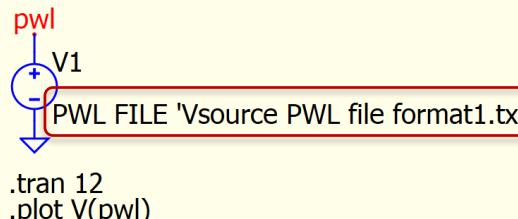


V. Voltage Source : PWL (Piece-wise Linear) – Load from file

Qspice : Vsource PWL file.qsch / Vsource PWL-Repeat-File.qsch

- PWL : load from file
 - Filename is quoted with single ('filename') or double ("filename") quotation
 - Delimiter can be space [] or comma [,]
 - To load from file, syntax is
 - FILE filename
 - Support repeat syntax, but remember last value in repeated PWL is ignored and forced to be equal to first value
- File internal format
 - Only two column of data
 - 1st column is TIME
 - 2nd column is Voltage/Current
 - No headers line

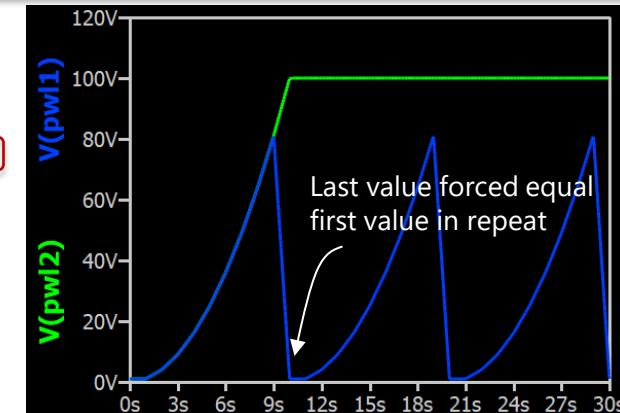
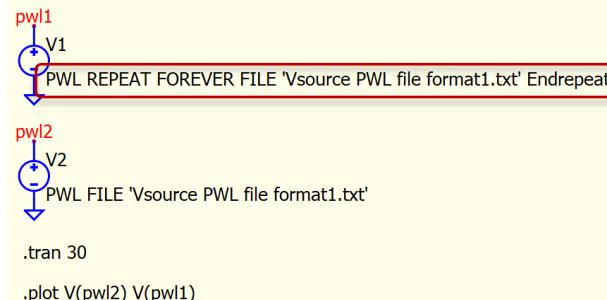
Syntax : PWL FILE file.txt [LOG] [XTRAP]



1	1	1
2	2	4
3	3	9
4	4	16
5	5	25
6	6	36
7	7	49
8	8	64
9	9	81
10	10	100

1	1,1
2	2,4
3	3,9
4	4,16
5	5,25
6	6,36
7	7,49
8	8,64
9	9,81
10	10,100

Syntax : PWL FOR[EVER] [times] FILE 'filename' Endrepeat

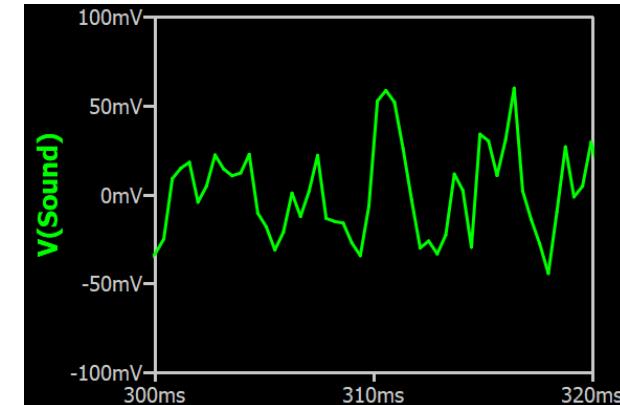


V. Voltage Source : PWL (Piece-wise Linear) – Load from file

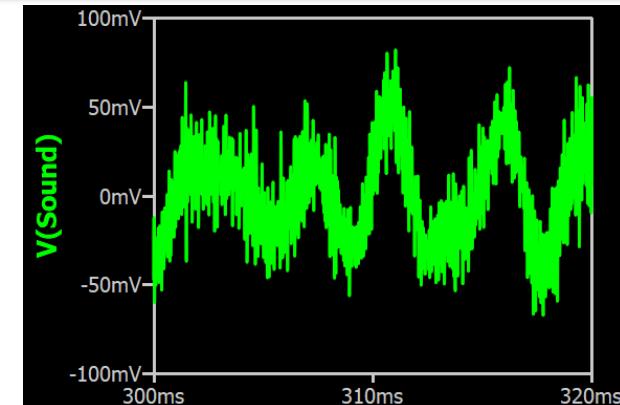
Qspice : Vsource - PWL with TimeCtrl.qsch | SoundScope0.csv

- PWL : load from file
 - In default,
timectrl=none for PWL
V-source. Timestep is determined by simulator and not necessary to follow data file breakpoints
 - To fully re-assemble data file, **timectrl=breaks** is required to guarantee simulation take at least one timestep at each break point

Sound
V1
PWL File 'SoundScope0.csv'
.tran 0.4
.plot V(Sound)



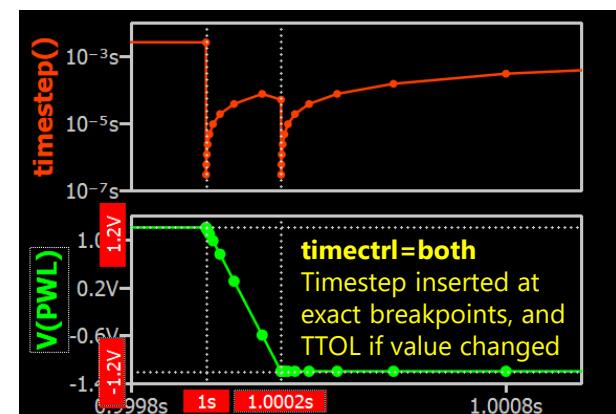
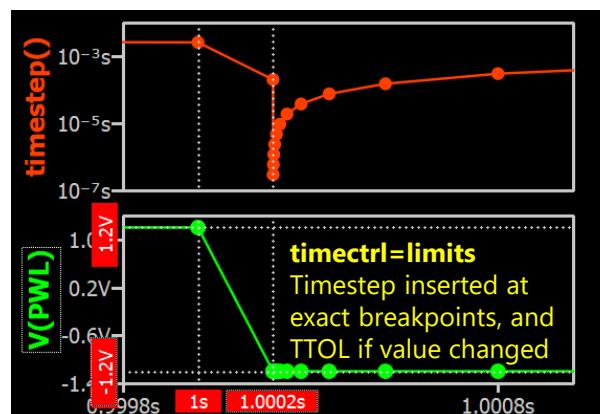
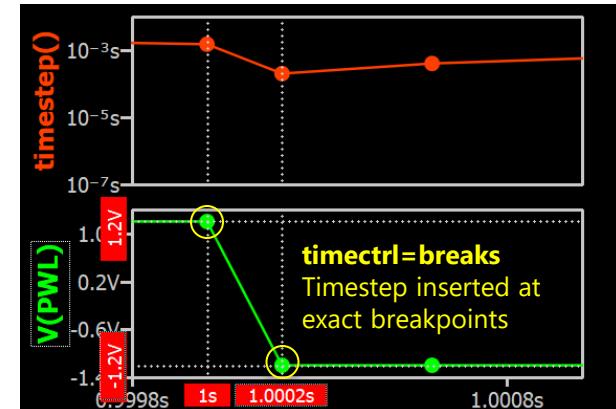
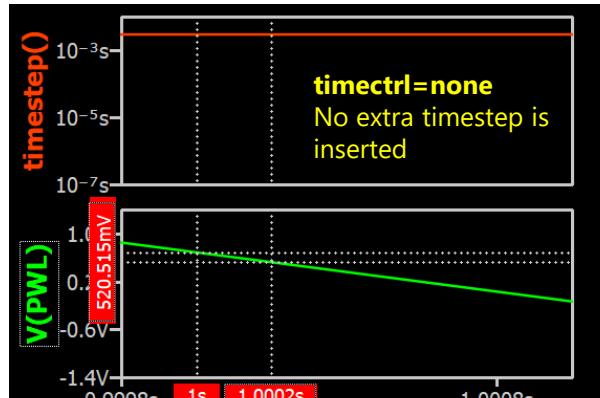
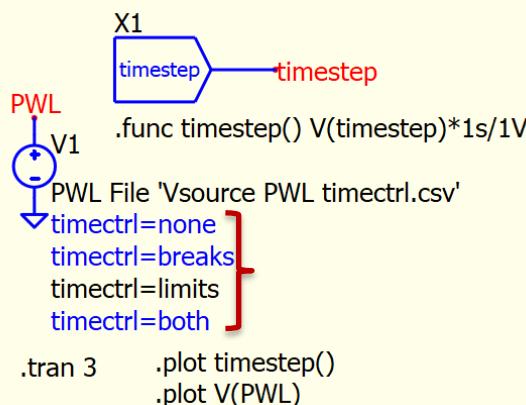
Sound
V1
PWL File 'SoundScope0.csv'
timectrl=breaks
.tran 0.4
.plot V(Sound)



V. Voltage Source : PWL (Piece-wise Linear) – Timectrl

Qspice : Vsource PWL timectrl.qsch | Vsource PWL break.csv

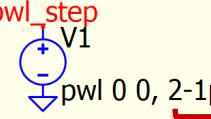
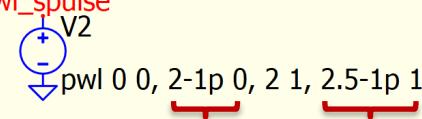
- Timectrl in PWL
 - There are three type of time control in V-source to control simulation timestep
 - None
 - Breaks
 - Limits
 - Both

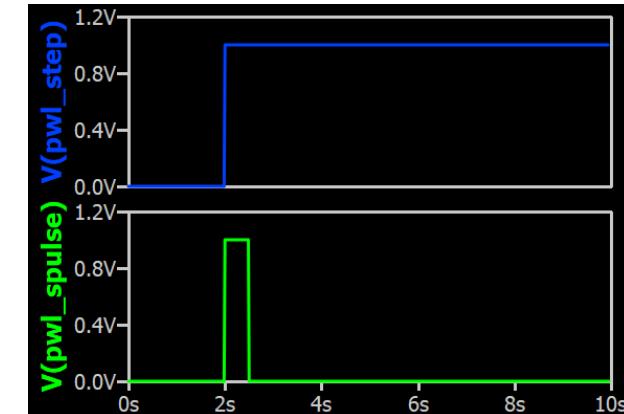


V. Voltage Source : PWL (Piece-wise Linear) – Usage

Qspice : Vsource PWL-SinglePulse.qsch

- Step and Single Pulse
 - To create step or single pulse with PWL, a datapoint to maintain voltage level before change is required
 - For example, if change occurs at 2s, define an extra datapoint at 1.999s or (2-1p)

pwl_step

pwl_spulse

.tran 10
.plot V(pwl_spulse)
.plot V(pwl_step)



W. Current Controlled
Switch

W. Current Controlled Switch

- W. Current Controlled Switch

- Syntax: Wnnn N1 N2 <name> <model> [ttol=<value>]
 - The current sensed in device <name> controls the switch. The sense device can be any element internally represented as a Thévenin equivalent circuit, i.e., E-, H-, L-, or V-devices.
- .model <model> CSW [model parameters]

Current Controlled Switch Model Parameters

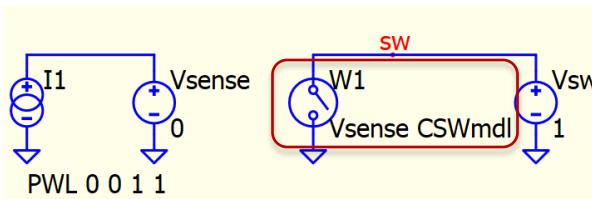
Name	Description	Units	Default
IH	Hysteresis current	A	0.0
IOFF	Current when open	A	0.0
ION	Current when closed	A	0.0
IT	Threshold current	A	0.0
M	Number of parallel devices		1.0
ROFF	Open resistance	Ω	1/Gmin
RON	Closed resistance	Ω	1.0
TTOL	Temporal tolerance	sec	1e308

If IT and IH are specified, ION and IOFF are ignored. If IH is negative the switch smoothly transitions between on and off.

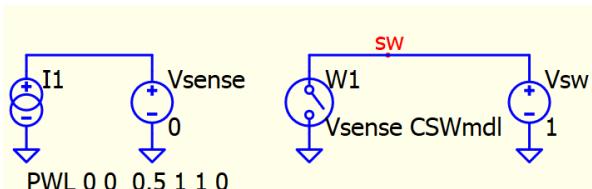
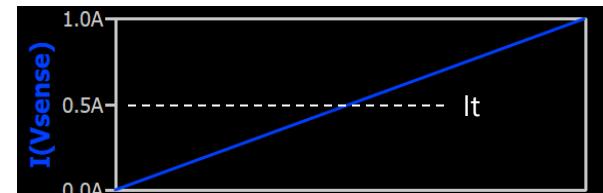
W. Current Controlled Switch

Qspice : CSW - RON ROFF IT IH.qsch | CSW - IH -ve.qsch

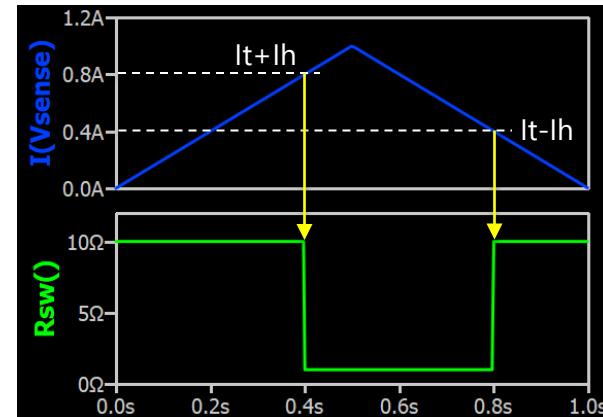
- RON, ROFF, IT, IH
 - RON : Closed resistance
 - ROFF : Open resistance
 - IT : Threshold current
 - IH : hysteresis current
 - Discrete transition with +ve IH, smoothly transition with -ve IH
 - **Default Ron=1**
 - **Default Roff=1/Gmin**
 - **Default It=0**
 - **Default Ih=0**



```
.model CSWmdl CSW Ron=1 Roff=10 It=0.5 Ih=0
.func Rsw() V(sw)/-I(Vsw)
.tran 1
.plot Rsw()
.plot I(Vsense)
```



```
.model CSWmdl CSW Ron=1 Roff=10 It=0.6 Ih=0.2
.func Rsw() V(sw)/-I(Vsw)
.tran 1
.plot Rsw()
.plot I(Vsense)
```

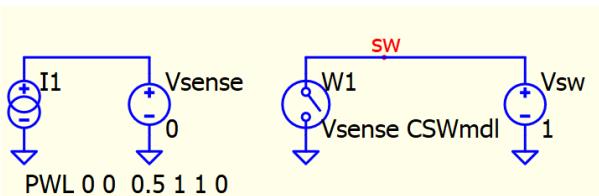


W. Current Controlled Switch

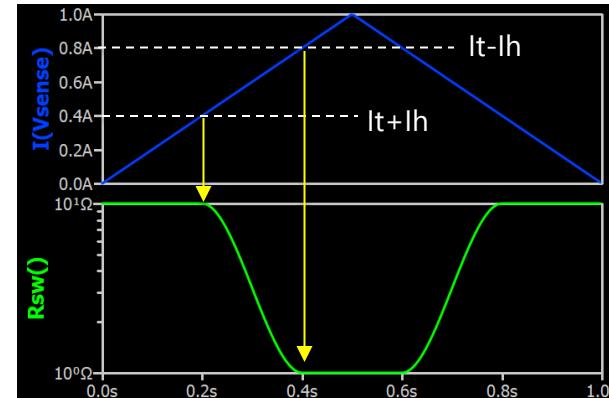
Qspice : CSW - IH -ve.qsch | CSW - ION IOFF.qsch

- IH

- IH : hysteresis current
- IH is negative the switch smoothly transitions between on and off

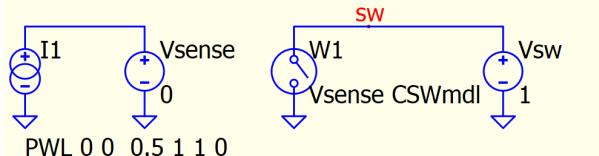


```
.model CSWmdl CSW Ron=1 Roff=10 It=0.6 Ih=-0.2
.func Rsw() V(sw)/-I(Vsw)
.tran 1
.plot Rsw()
.plot I(Vsense)
```

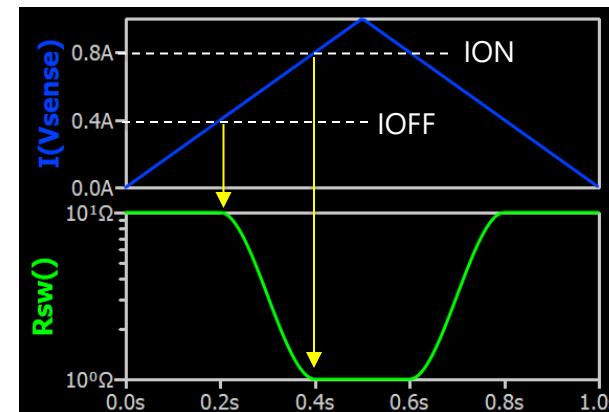


- ION and IOFF

- ION : Current when closed
- IOFF : Current when open
- If ION and IOFF are specified, the switch smoothly transitions between on and off



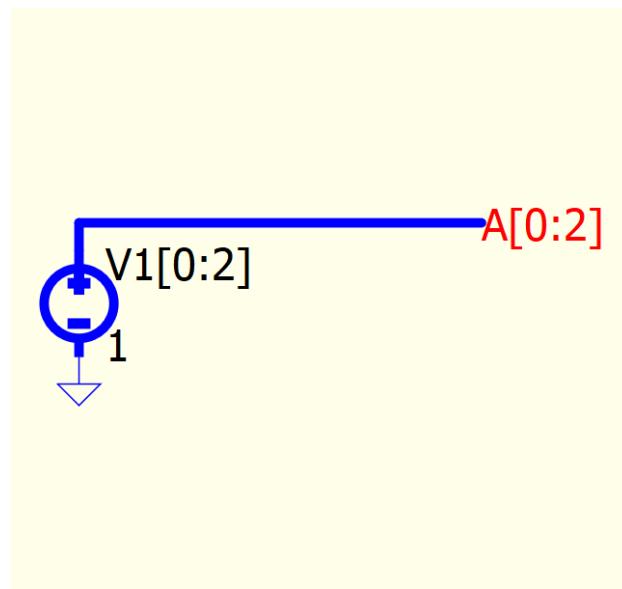
```
.model CSWmdl CSW Ron=1 Roff=10 Ion=0.8 Ioff=0.4
.func Rsw() V(sw)/-I(Vsw)
.tran 1
.plot Rsw()
.plot I(Vsense)
```



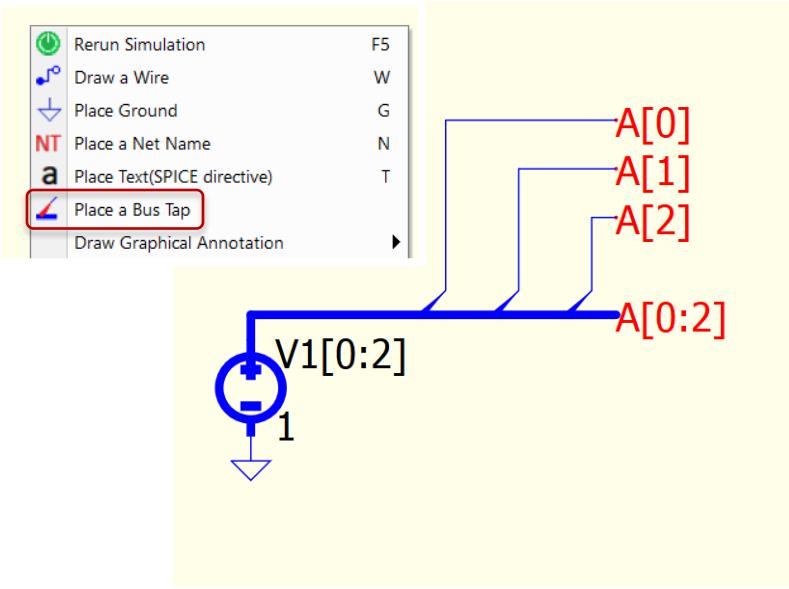
W. Wire and BUS

W. Wire and Bus : Introduction

- [1] Place a voltage source and draw wire
- [2] Rename voltage source to V1[0:2]
- [3] Place net name as A[0:2]
- [4] Now, the wire becomes a bus, in this example, A[m] is voltage of V1[m]

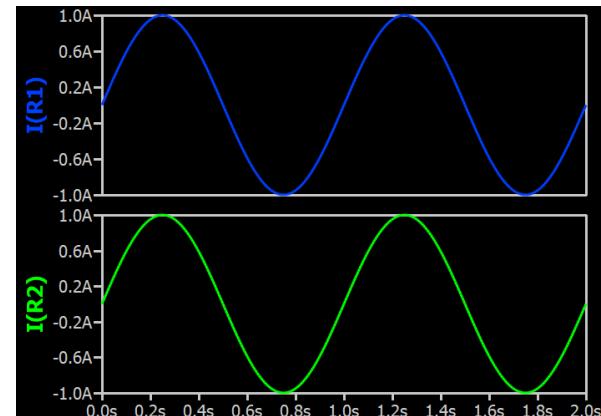
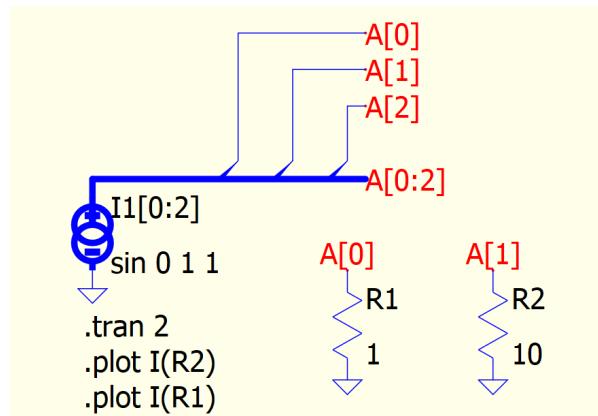


- [5] Right click and select "Place a Bus Tap"
- [6] Place net name to bus tap wire



W. Wire and Bus : Bus for V/I-source

- Identical I-source
 - File : BUS - Isrc.qsch
 - Bus can replicate current or voltage source for multiple nodes usage
 - In netlist
 - $I1[0] 0 A[0] \sin 0 1 1$
 - $I1[1] 0 A[1] \sin 0 1 1$
 - $I1[2] 0 A[2] \sin 0 1 1$
 - ** .dc for $I1$ won't work as three I-sources $I1[0]$, $I1[1]$ and $I1[2]$ in this format



W. Wire and Bus : Bus for Ø-Device

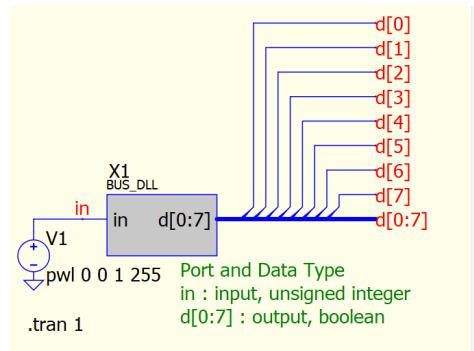
- Bus for Ø-Device

- Files

- BUS_DLL.qsch
 - bus_dll.cpp
 - bus_dll.dll

- Procedure

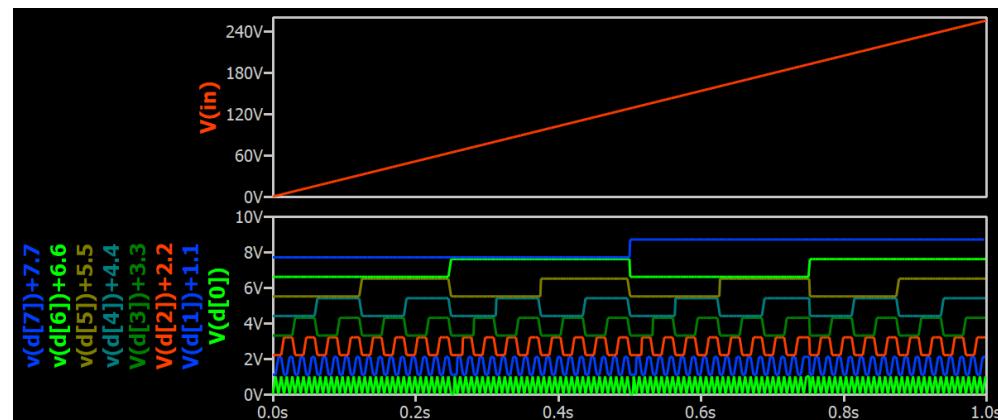
- For Ø-Device, defines an output port with label name with net[0:7] array
 - To output bit, data type should be boolean
 - In template generation, it creates variable name as `net_0_` for bit 0 and `net_7_` for bit 7



```
.plot V(d[0]) V(d[1])+1.1 V(d[2])+2.2 V(d[3])+3.3  
+V(d[4])+4.4 V(d[5])+5.5 V(d[6])+6.6 V(d[7])+7.7  
.plot V(in)
```



```
extern "C" __declspec(dllexport) void bus_dll  
{  
    unsigned int in = data[0].ui; // input  
    bool &d_0 = data[1].b; // output  
    bool &d_1 = data[2].b; // output  
    bool &d_2 = data[3].b; // output  
    bool &d_3 = data[4].b; // output  
    bool &d_4 = data[5].b; // output  
    bool &d_5 = data[6].b; // output  
    bool &d_6 = data[7].b; // output  
    bool &d_7 = data[8].b; // output  
  
    // Implement module evaluation code here:  
    d_0 = in & 0x01;  
    d_1 = in & 0x02;  
    d_2 = in & 0x04;  
    d_3 = in & 0x08;  
    d_4 = in & 0x10;  
    d_5 = in & 0x20;  
    d_6 = in & 0x40;  
    d_7 = in & 0x80;  
}
```



Y. Piezoelectric Crystal

Y. Piezoelectric Crystal : Instance Parameters

- Y. Piezoelectric Crystal
 - Syntax: Ynnn N+ N- <frequency1> dF=<value> Ctot=<value> [Q=<value>]

Piezoelectric Crystal Instance Parameters

Name	Description	Units	Default
CTOT	Total capacitance as measured at DC	F	
DF ¹	Difference between series and parallel resonant frequencies	Hz	
FREQUENCY ¹	Resonant frequency	Hz	
IC	Initial internal current(Used with UIC on the .tran)	A	0.0
M	Number of parallel devices		1.0
RSER	Series resistance	Ω	Determined from Q
Q	Quality factor		$1.6e13 \div \text{FREQUENCY}^2$

$$Q = \min(1e6, \frac{1e13}{\text{FREQUENCY}^2})$$

If default equation is used

- If DF > 0 : **Series Resonant Freq = Frequency** and **Parallel Resonant Freq = Frequency + DF**
- If DF < 0 : **Series Resonant Freq = Frequency + DF** and **Parallel Resonant Freq = Frequency**

Y. Piezoelectric Crystal : Formula and Equivalent Circuit

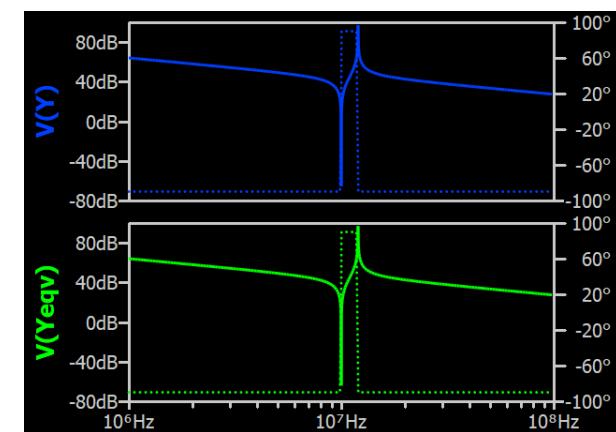
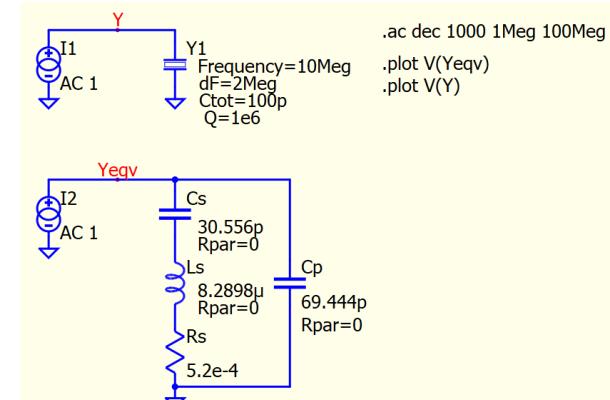
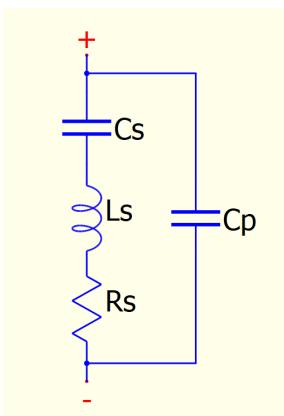
Qspice : Y Crystal - Equivalent Circuit.qsch

- Piezoelectric Crystal Equation

- Series Resonant Frequency : $f_s = \frac{1}{2\pi\sqrt{L_s C_s}}$
- Parallel Resonant Frequency : $f_p = \frac{1}{2\pi\sqrt{\frac{C_p C_s}{L_s C_p + C_s}}}$
- Crystal Oscillators Q-factor : $Q = \frac{X_L}{R} = \frac{2\pi f L_s}{R_s}$
- Total Capacitance : $C_{TOT} = C_p + C_s$

- Calculated Value

- $C_p = \left(\frac{f_s}{f_p}\right)^2 C_{TOT}$
- $C_s = C_{TOT} - C_p$
- $L_s = \frac{1}{(2\pi f_s)^2 C_s}$
- $R = \frac{2\pi f L}{Q}$



\tilde{A} -Device

Type : MultGmAmp

\tilde{A} -Device : MultGmAmp and RRopAmp

- \tilde{A} -Device
 - Syntax

$\tilde{A}nnn \ VDD \ VSS \ OUT \ IN- \ IN+ \ MULT+ \ MULT- \ IN-- \ IN++ \ EN \ \# \ \# \ \# \ \# \ \# \ \# \ \# \ <TYPE> \ [INSTANCE \ PARAMETERS]$

- \tilde{A} -device models a highly configurable Gm block than can mimic the behavior of a complementary MOSFET output, there are two types MULTGMAMP and RRROPAMP

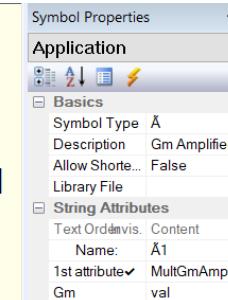
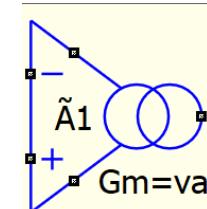
\tilde{A} -Device Types

TYPE	Behavior
MULTGMAMP	Multiplying Gm Amplifier
RRROPAMP	Rail-to-Rail Output OpAmp

MULTGMAMP is Operational Transconductance Amplifier (OTA)

RRopAmp is implemented with two Gm amps (MULTGMAMP) with an internal node for a Miller capacitor connected to the output

- To Identify what $<TYPE>$ a symbol is
- In Symbol Properties > 1st attribute
 - View > Netlist : from device syntax



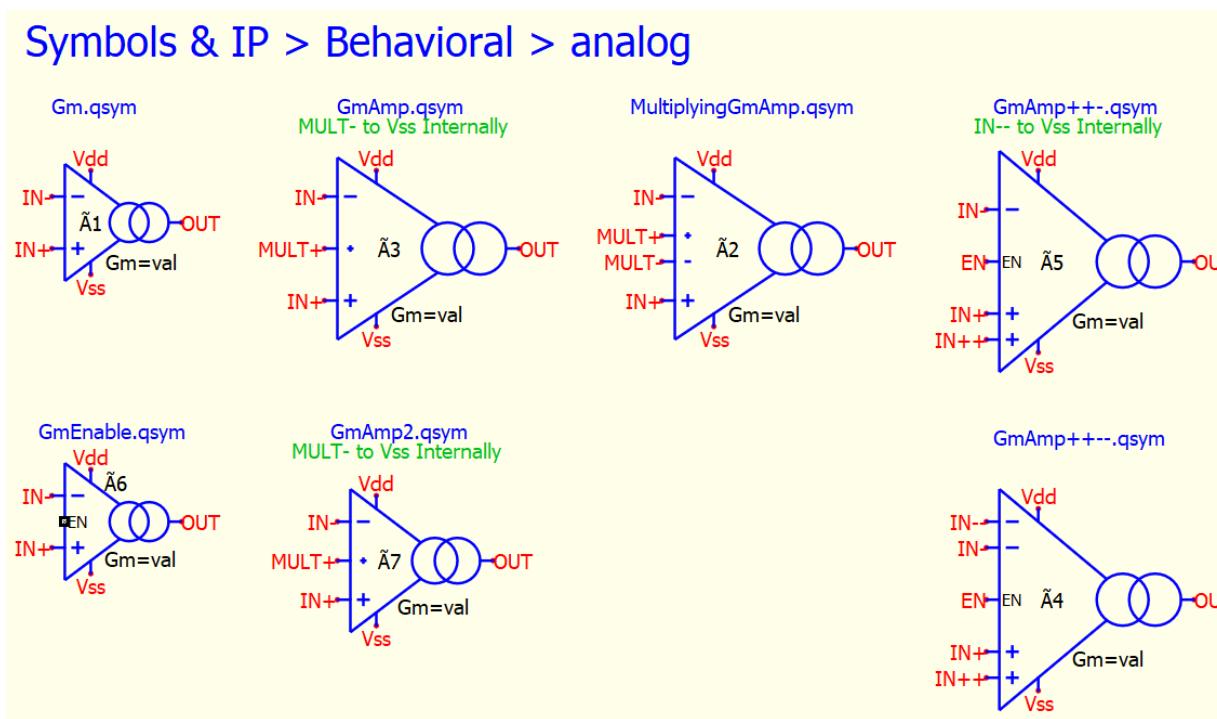
```
* C:\Users\kelvinleung\Documents\QSPICE\Untitled.qsch
A1 Y0 Y1 Y2 Y3 Y4 Y Y Y Y Y Y Y Y Y Y Y Y MultGmAmp Gm=val
.end
```

\tilde{A} -Device : MultGmAmp

- \tilde{A} -Device

- Syntax: **$\tilde{A}nnn\ VDD\ VSS\ OUT\ IN-\ IN+\ MULT+\ MULT-\ IN--\ IN++\ EN\ \#\#\#\#\#\#<TYPE>\ [INSTANCE\ PARAMETERS]$**
- Formula in Linear Region : $i_{out} = GM * V(MULT+, MULT-) * V(\min(IN+, IN++), \max(IN-, IN--))$

Symbols & IP > Behavioral > analog



À-Device Instance Parameters in MultGmAmp

À-Device Instance Parameters

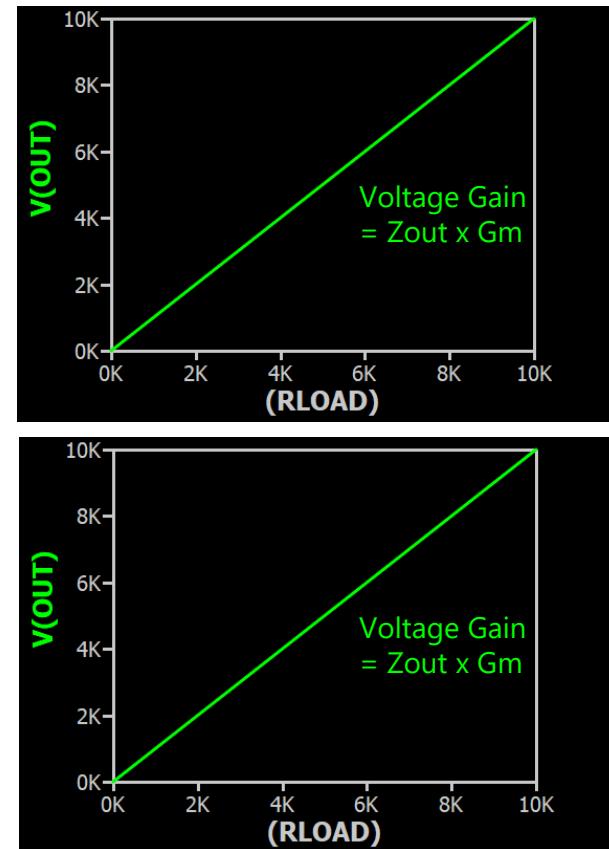
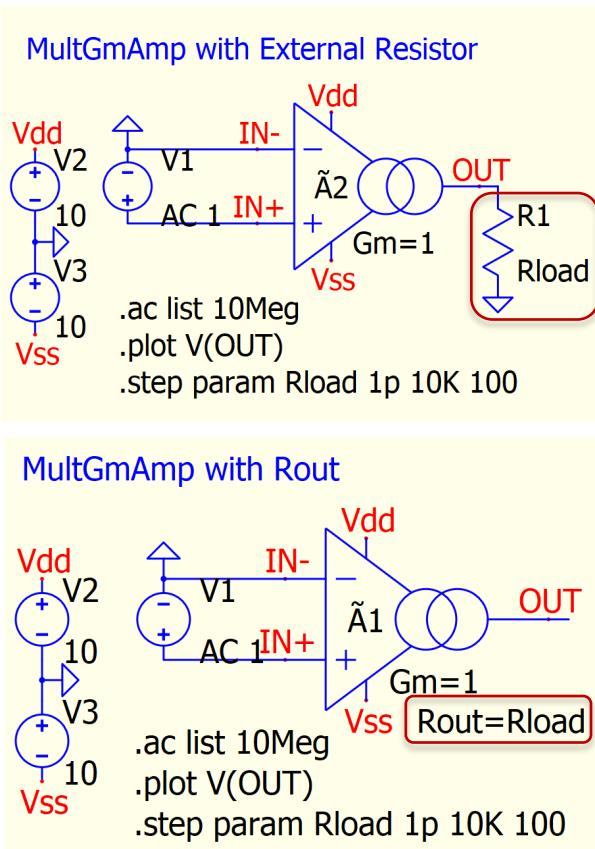
Name	Description
CAPINCM	Common eigenmode input capacitance
CAPINNM	Normal eigenmode input capacitance
CAPVDD	Capacitance from output to Vdd
CAPVSS	Capacitance from output to Vss
EN	Equivalent input voltage noise density
ENK	EN corner frequency
FT	3dB bandwidth of transconductance with no voltages slewing
GM	Ideal transconductance
IC	Initial condition of $V_{in} \times V_{mult}$
IN	Equivalent input current noise density
INF	Common mode input current noise density proportional to frequency
INK	IN corner frequency
IOUT	Maximum sourcing current
ISNK	Maximum sinking current
ISNKKNEE	Sharpness of Maximum sinking current limit
ISRC	Maximum sourcing current
ISRCKNEE	Sharpness of max sourcing current limit
M	Number of parallel devices

REF	Logic threshold for enable(from Vss)
ROUT	Additional impedance added to output(2*R to Vdd, 2*R to Vss)
TEMP	Instance temperature
TTOL	Temporal tolerance for enable & UVLO
UVLO	Minimum supply voltage
VCROSS	Cross conduction voltage range
VDSAT	Voltage where gm starts to switch over to a resistance
VDSAT1	Voltage where gm starts to switch over to a resistance(top FET)
VDSAT2	Voltage where gm starts to switch over to a resistance(bottom FET)
VINHIGH	Output range measured from positive rail
VINHIGHKNEE	Sharpness of positive input range limit
VINLOW	Input range measured from negative rail
VINLOWKNEE	Sharpness of negative input range limit
VOS1	Offset voltage for input
VOS2	Offset voltage for multiplying input
VOUTMAX	Maximum output voltage measured from negative rail
VOUTMIN	Minimum output voltage measured from negative rail

MultGmAmp : Basic

Qspice : Multgmamp - Basic.qsch

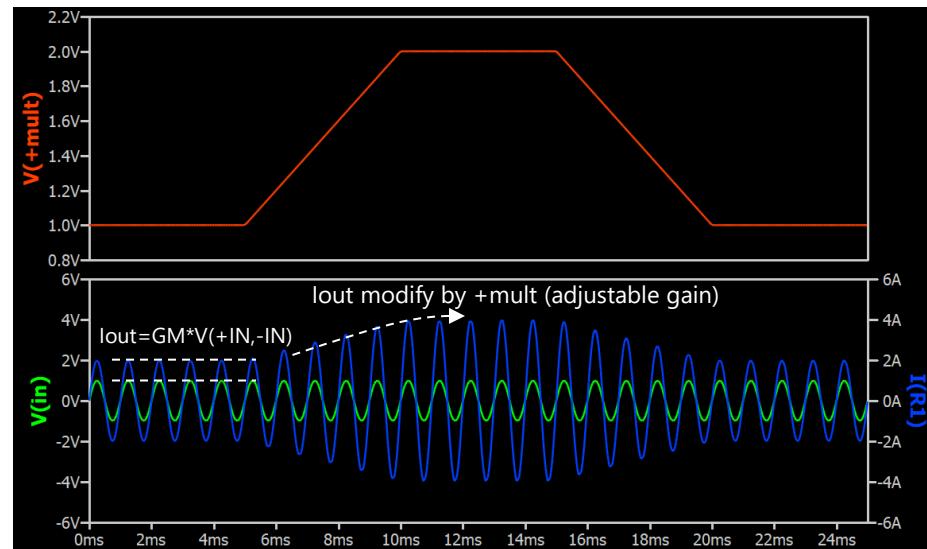
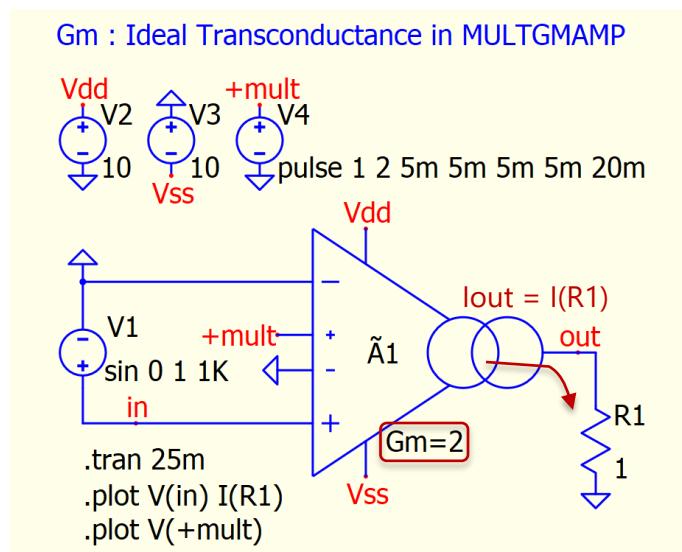
- Basic
 - MULTGMAMP is Operational Transconductance Amplifier (OTA) whose differential input voltage produces an output current
 - By simplify its input only with IN_- and IN_+
 - $I_{out} = Gm \times (V_{IN+} - V_{IN-})$
 - As OTA output a current, its output voltage depends on loading impedance, where
 $V_{out} = Z_{out} \times I_{out}$
 $V_{out} = Z_{out} Gm \times (V_{IN+} - V_{IN-})$
 - Z_{out} can be defined in two ways
 - External Resistor
 - Instance parameter R_{out}



MultGmAmp : GM (Ideal transconductance)

Qspice : Multgmamp – Gm with mult.qsch

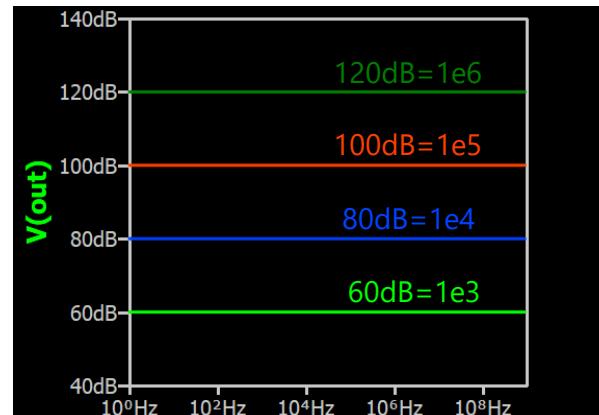
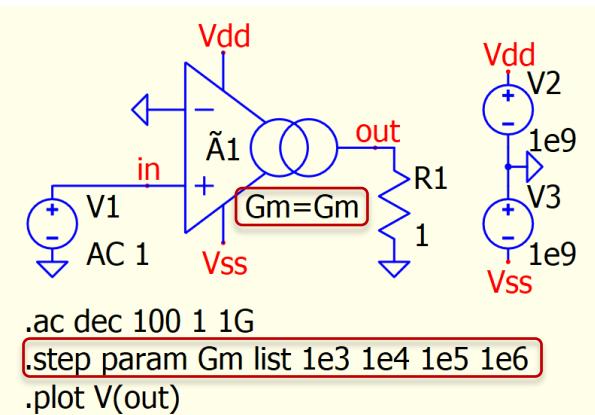
- Gm : Ideal transconductance
 - Operational Transconductance Amplifier (OTA) is an amplifier whose differential input voltage produces an output current
 - Without IN++ and IN--, formula is $\text{Iout} = \mathbf{GM} * V(\mathbf{MULT+}, \mathbf{MULT-}) * V(\mathbf{IN+}, \mathbf{IN-})$
 - Gm can be positive or negative, but must be non-zero



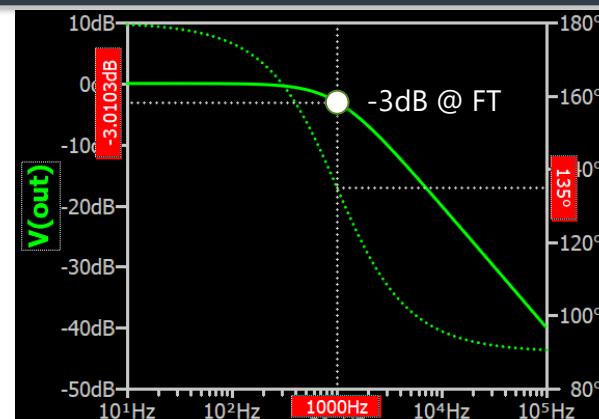
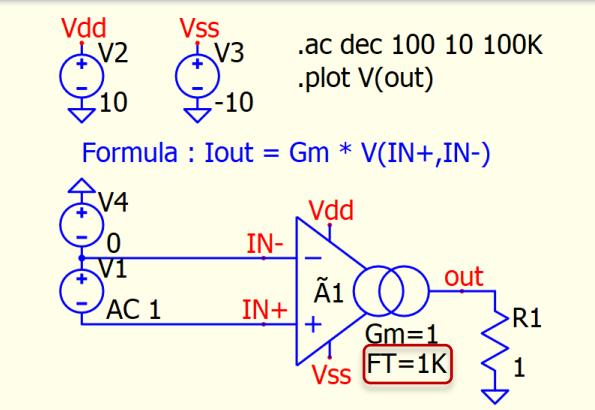
MultGmAmp : GM (Ideal transconductance) and FT (3dB Bandwidth)

Qspice : Multgmamp - Gm.qsch | Multgmamp - FT.qsch

- Gm Ideal transconductance
 - No default value of Gm
 - Must be user provided
 - MULTGMAMP has ideal frequency response if other instance parameters are in default



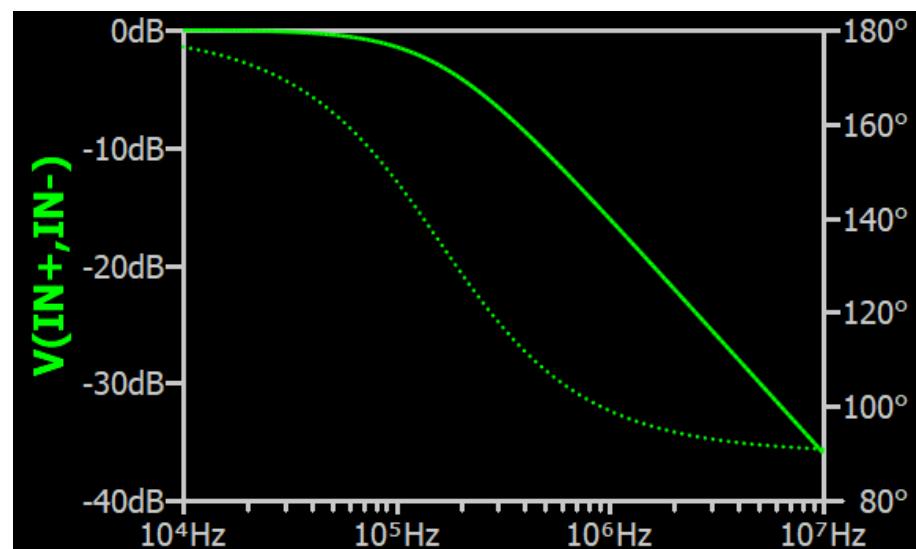
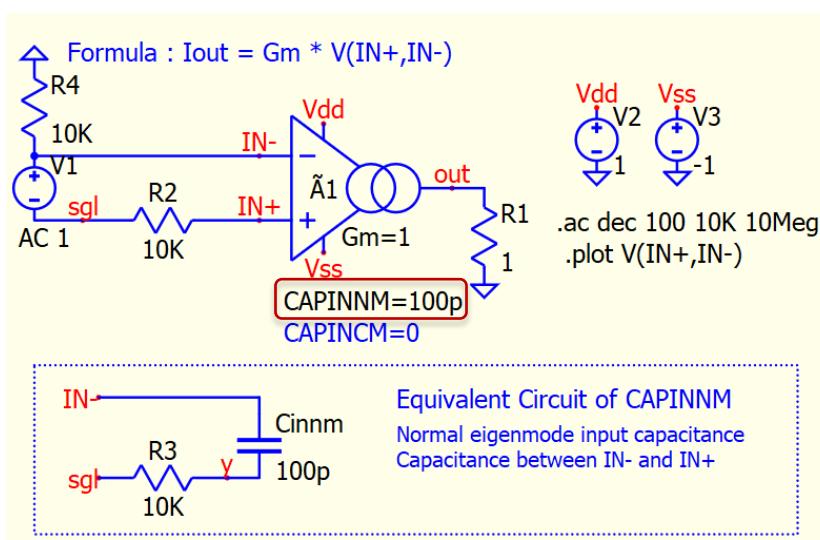
- FT (3dB Bandwidth)
 - FT : 3dB bandwidth of transconductance with no voltages slewing
 - **Default FT=0** (i.e. infinite)
 - FT is frequency



MultGmAmp : CAPINNM (Normal eigenmode input capacitance)

Qspice : Multgmamp - Capinnm.qsch

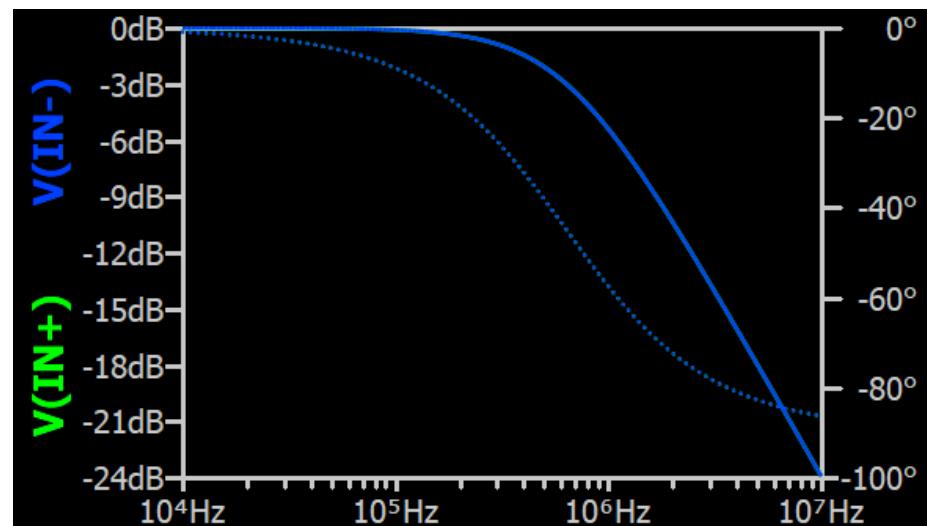
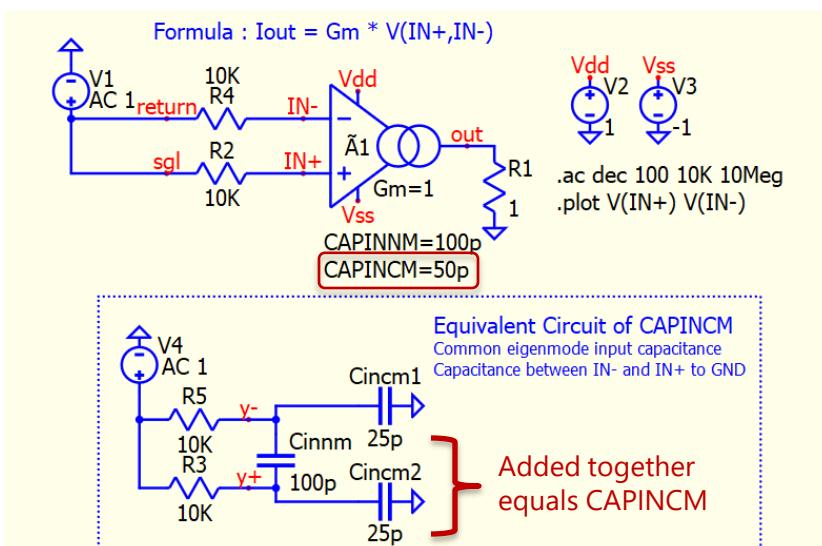
- CAPINNM : Normal eigenmode input capacitance
 - Default CAPINNM=0
 - This capacitance is between IN- and IN+



MultGmAmp : CAPINCM (Common eigenmode input capacitance)

Qspice : Multgmamp - Capincm.qsch

- CAPINCM : Common eigenmode input capacitance
 - Default CAPINCM=0
 - This capacitance is between IN- to GND and IN+ to GND

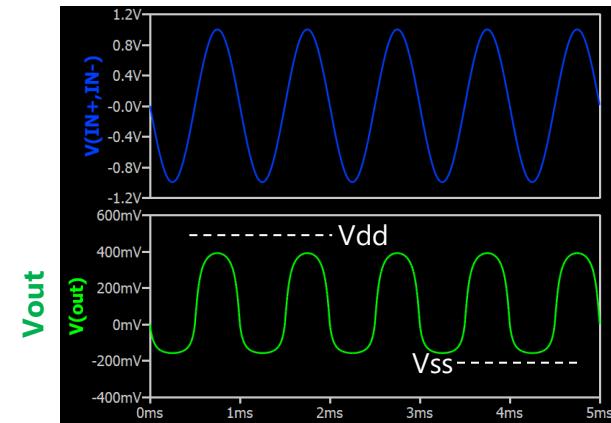
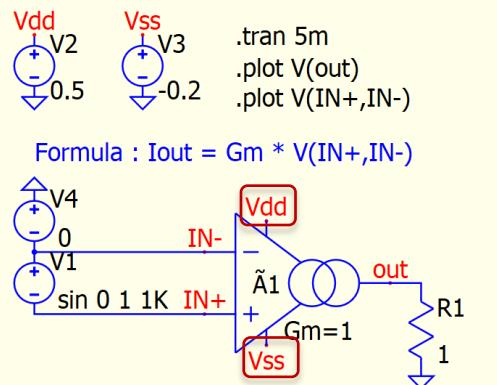


MultGmAmp : Vdd, Vss and VDSAT – Output Voltage Swing

Qspice : Multgmamp - Vdd Vss.qsch | Multgmamp - VDSAT.qsch

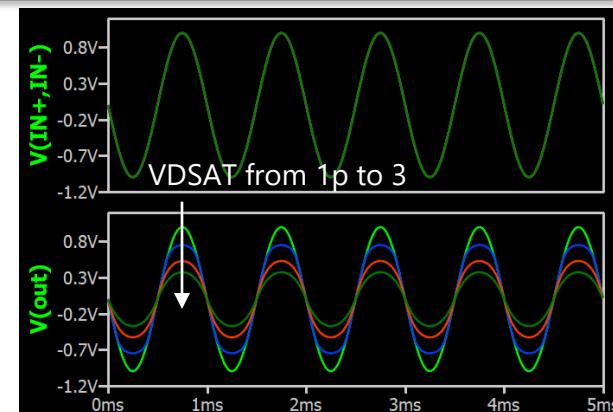
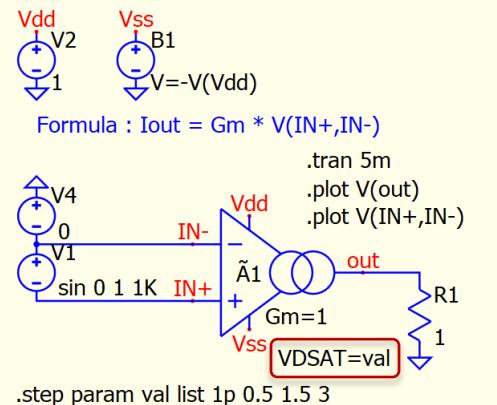
- Vdd and Vss Nets

- This is not parameters but Net 1 and Net 2 in \tilde{A} -Device
- Output voltage swing is limited by Vdd and Vss
- ** In this example, Vout cannot reach Vdd and Vss because Default VDSAT is set in Qspice



- VDSAT

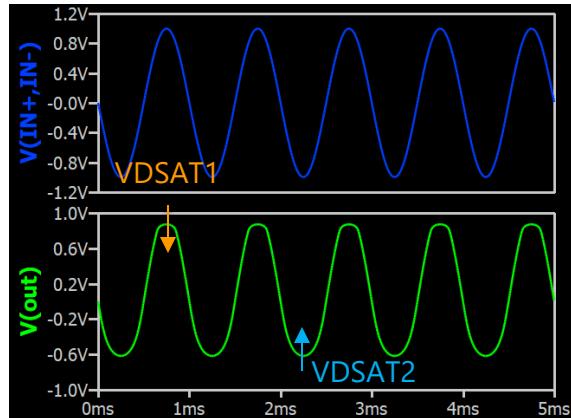
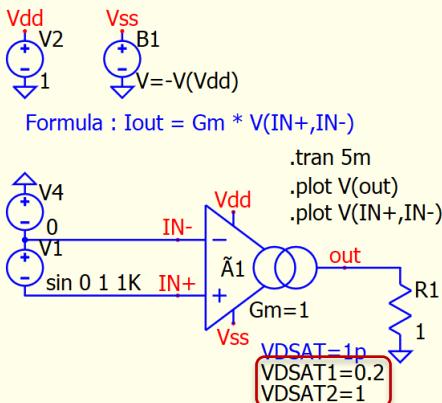
- Vdsat : Voltage where gm starts to switch over to a resistance
- Default VDSAT = 0.5**
- VDSAT=0** forced this value to default!
- Set VDSAT to small value (e.g. 1p) for no saturation**



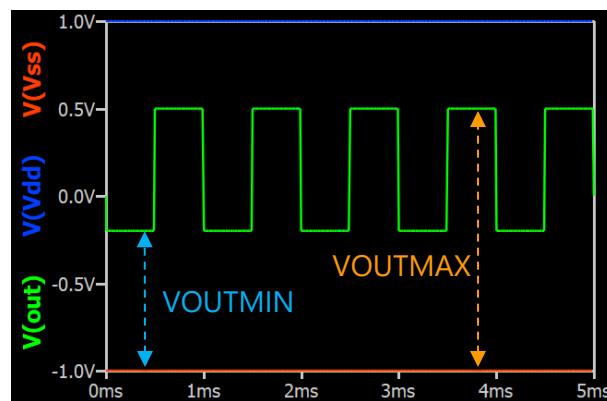
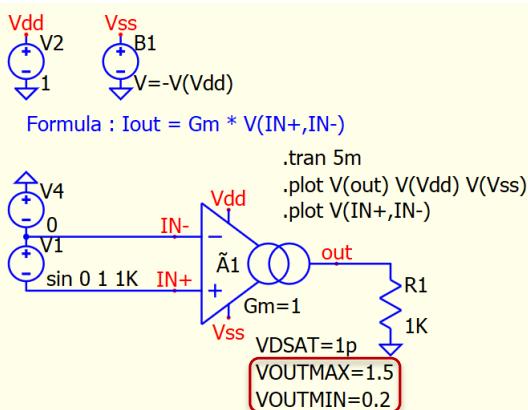
MultGmAmp : VDSAT1, VDSAT2, VOUTMIN, VOUTMAX – Output Voltage

Qspice : Multgmamp - VDSAT1 VDSAT2.qsch ; Multgmamp | VOUTMAX VOUTMIN.qsch

- VDSAT1 and VDSAT2
 - VDSAT1 : Voltage where gm starts to switch over to a resistance (top FET)
 - VDSAT2 : Voltage where gm starts to switch over to a resistance (bottom FET)
 - **Default VDSAT1=VDSAT**
 - **Default VDSAT2=VDSAT1**



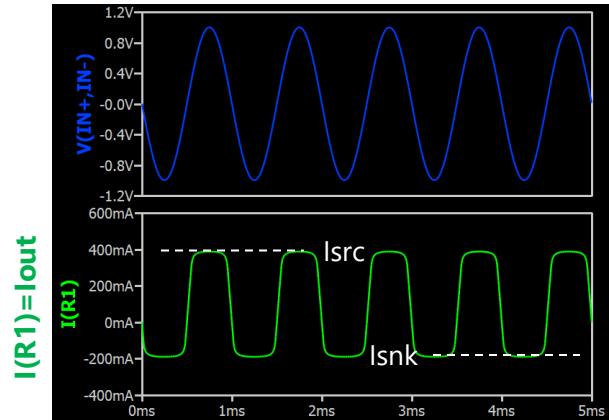
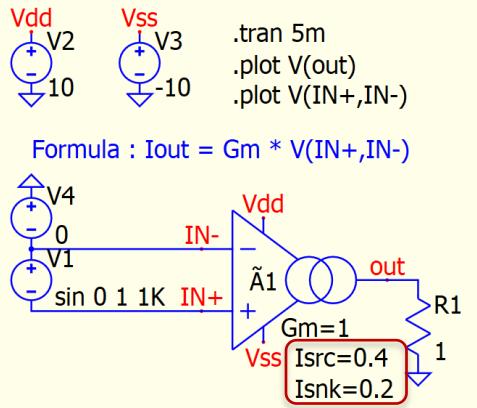
- VOUTMIN and VOUTMAX
 - Voutmin : Minimum output voltage measured from negative rail
 - Voutmax : Maximum output voltage measured from negative rail



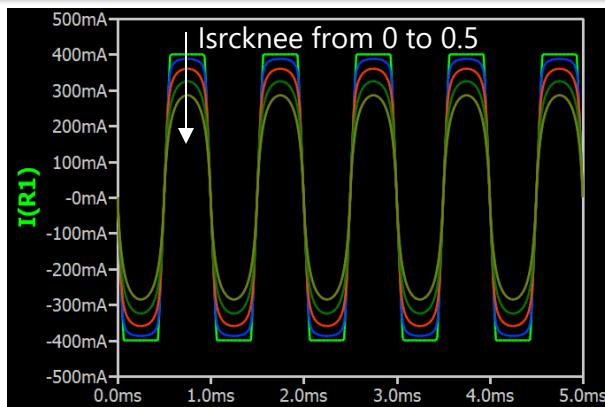
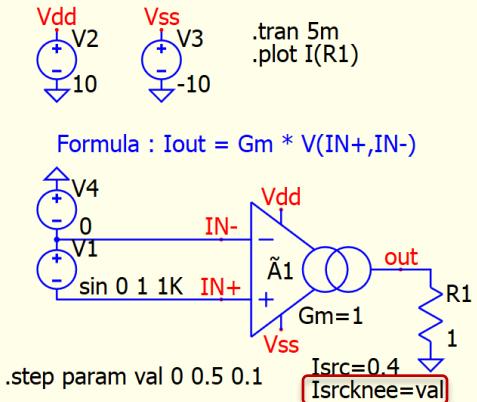
MultGmAmp : Iout/Isrc, Isnk – Output Sink and Src Current

Qspice : Multgmamp - ISRC ISNK.qsch ; Multgmamp - ISRCKNEE ISNKKNEE.qsch

- IOUT/ISRC and ISNK
 - Iout/Isrc : Maximum sourcing current (toward load)
 - Isnk : Maximum sinking current (toward device)
 - **Default ISRC = Infinite**
 - **Default ISNK = IOUT or ISRC**
 - i.e. if ISNK not specified, ISNK equals IOUT or ISRC



- ISRCKNEE and ISNKKNEE
 - Isrcknee : Sharpness of max sourcing current limit
 - Isnkknee : Sharpness of Maximum sinking current limit
 - **Default Isrckness = 0.1**
 - **Default Isnkkness = Isrcknee**
- Important Note
 - To have effect, must defines ISRC or ISNK



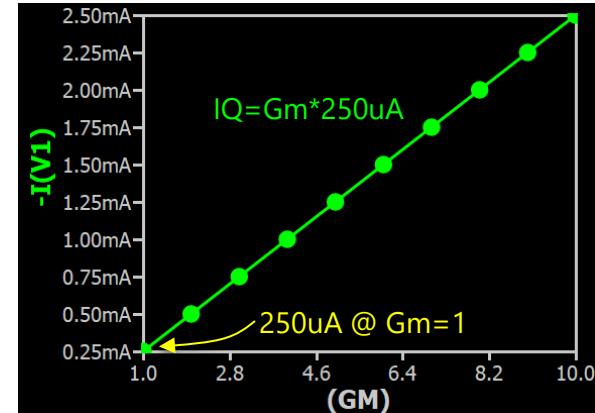
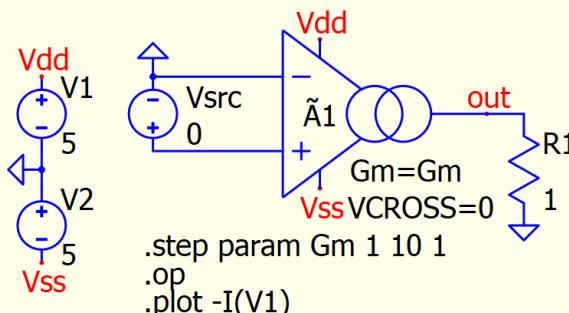
MultGmAmp : Quiescent Current and VCROSS

Qspice : Multgmamp - Quiescent Current.qsch | Multgmamp - VCROSS.qsch

Quiescent Current

- MultGmAmp runs in class AB, there is a quiescent current since both the top and bottom pass elements are conducting at the same time
- $I_Q = GM \times 250\mu A$
 - for $VCROSS=0$

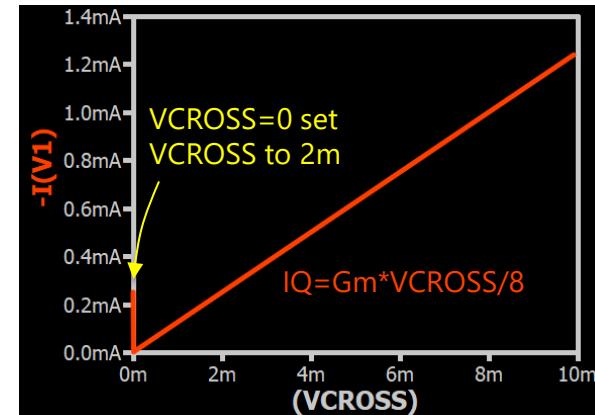
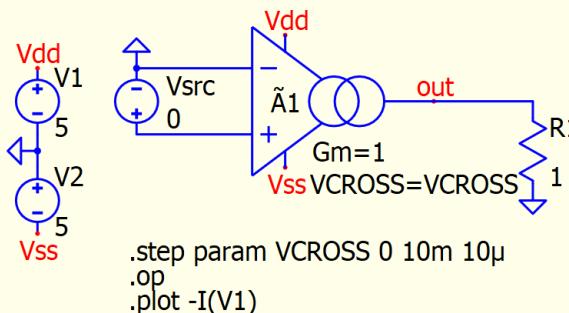
Quiescent Current in MultGmAmp



VCROSS

- Cross conduction voltage range
- Default $VCROSS=0$**
 - It will force $VCROSS=2mV$ if $VCROSS$ set to default or 0
 - If $VCROSS=1p$, quiescent current will $\sim 0A$
- $I_Q = GM \times \frac{VCROSS}{8}$

VCROSS in MultGmAmp (Quiescent Current)

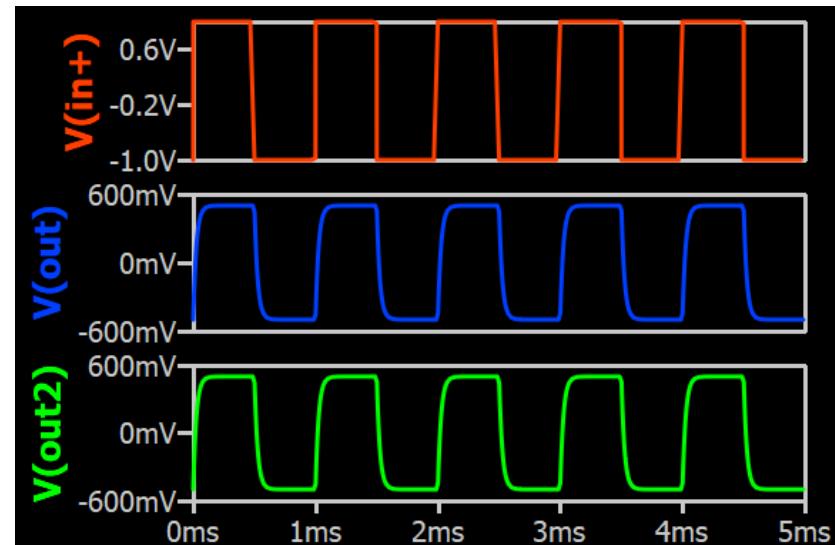
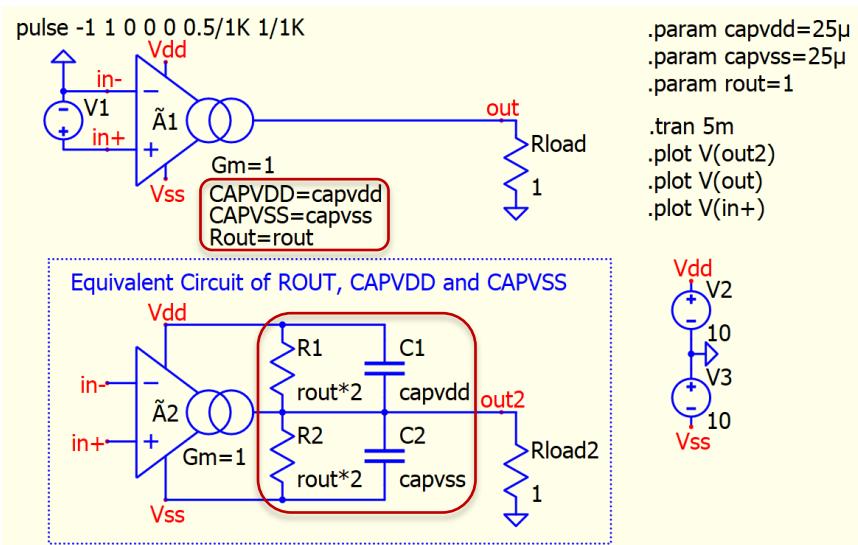


MultGmAmp : ROUT, CAPVDD and CAPVSS (Output Stage)

Qspice : Multgmamp - Rout CAPVDD CAPVSS.qsch

- ROUT, CAPVDD and CAPVSS

- ROUT : Additional impedance added to output ($2 \times R$ to Vdd, $2 \times R$ to Vss) (**Default ROUT=0, which is INF**)
- CAPVDD : Capacitance from output to Vdd (**Default CAPVDD=0**)
- CAPVSS : Capacitance from output to Vss (**Default CAPVSS=0**)

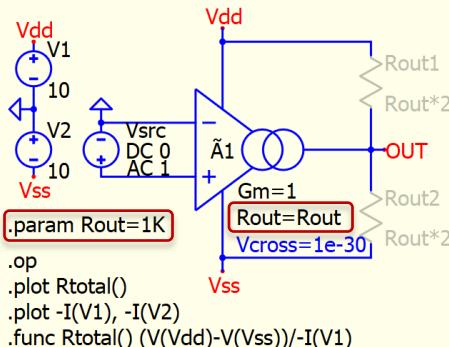


MultGmAmp : ROUT

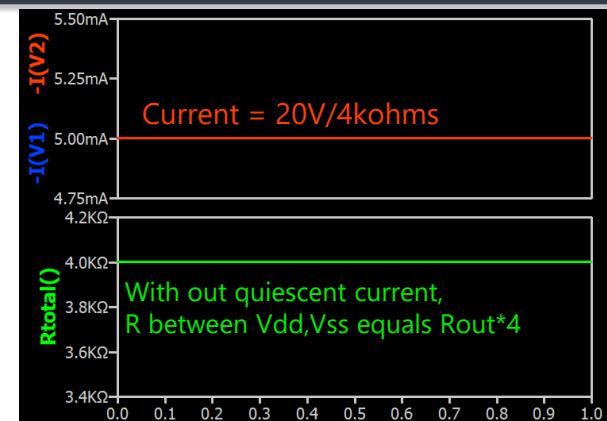
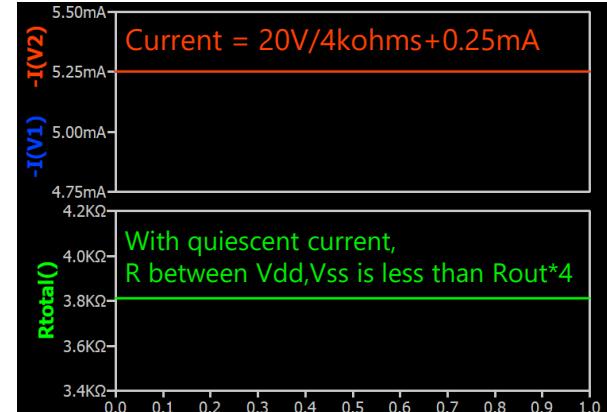
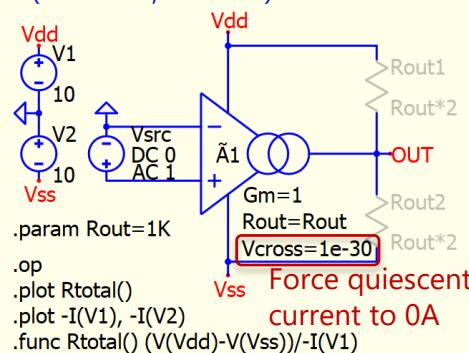
Qspice : Multgmamp - ROUT (.op).qsch

- ROUT
 - Additional impedance added to output ($2 \cdot R_{out}$ to V_{dd} , $2 \cdot R_{out}$ to V_{ss})
 - **Default ROUT=0**
 - ROUT equals infinite
 - In DC analysis, supply current equals current through ROUT + Quiescent Current
 - Total resistance between V_{dd} and V_{ss} is **Rout*4** if ignore quiescent current
 - In studying Rout, this section set $V_{cross}=1e-30$ to eliminate quiescent current

Rout : Additional impedance added to output
($2 \cdot R$ to V_{dd} , $2 \cdot R$ to V_{ss})



Rout : Additional impedance added to output
($2 \cdot R$ to V_{dd} , $2 \cdot R$ to V_{ss})



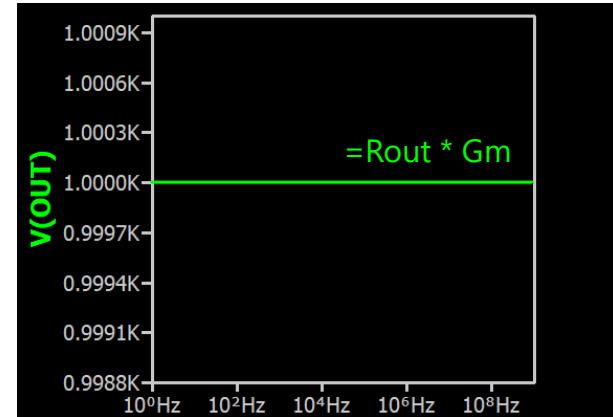
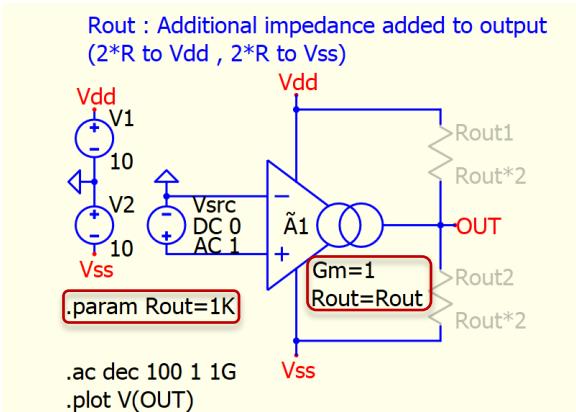
MultGmAmp : ROUT

Qspice : Multgmamp - ROUT (.ac).qsch

- ROUT in .ac
 - In .ac analysis, dc component likes quiescent current is ignored

Formula

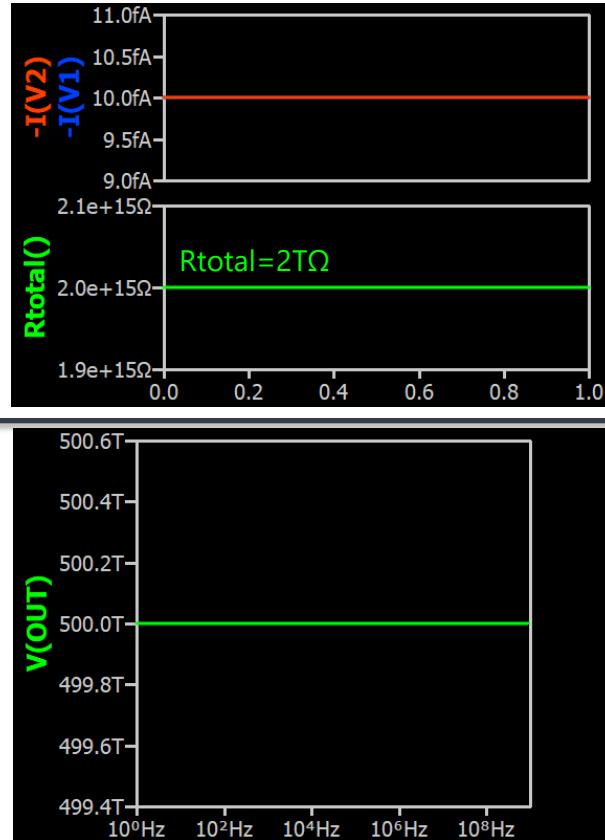
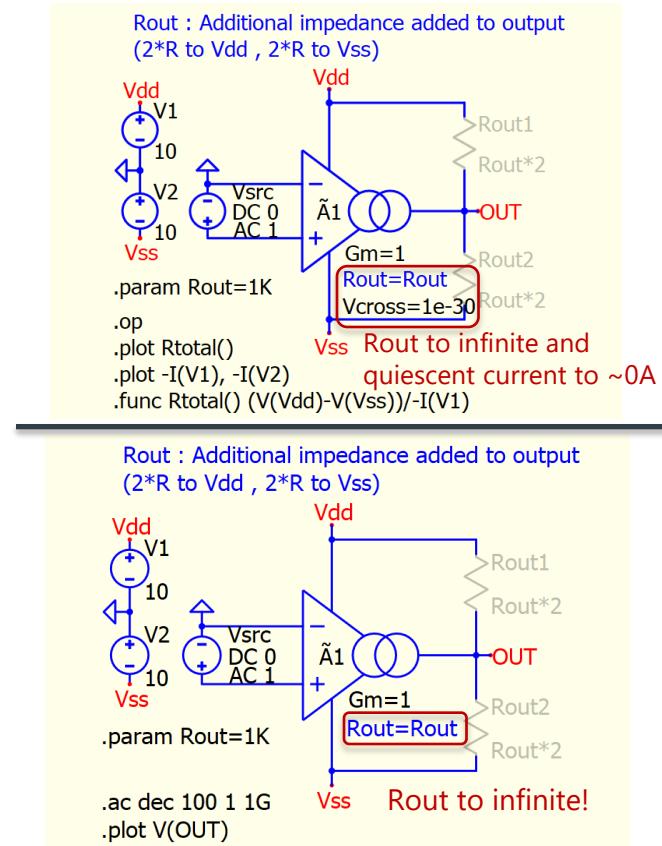
- By $I_{out} = Gm \times (V_{IN+} - V_{IN-})$
- With no external resistance,
 $V_{out} = Rout \times I_{out}$
 $= Rout \times Gm \times (V_{IN+} - V_{IN-})$
- .ac Set $V_{IN+} - V_{IN-} = 1$
- $V_{out} = Rout \times Gm$



MultGmAmp : ROUT=Infinite

Qspice : Multgmamp - ROUT - Infinite (.op).qsch | Multgmamp - ROUT - Infinite (.ac).qsch

- Case with $R_{out} = \text{Infinite}$
 - If R_{out} is set to default, R_{out} is infinite
 - However,
 - In .op analysis, R_{total} between V_{dd} to V_{ss} is $2T\Omega$
 - In .ac analysis, V_{out} will still read a finite value which is $500T$
 - In both cases, it proved Qspice with additional $1T\Omega$ between output- V_{dd} and output- V_{ss} by $R_{out} = \text{infinite}$

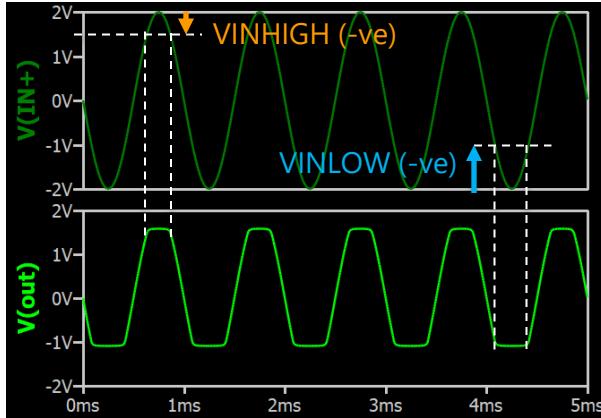
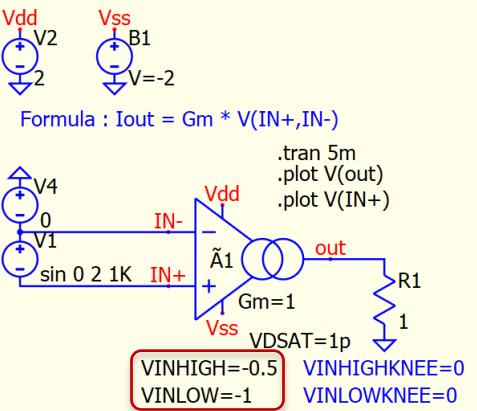


MultGmAmp : Input Range VINLOW, VINHIGH, VINHIGHKNEE, VINLOWKNEE

Qspice : Multgmamp - VINHIGH VINLOW.qsch ; Multgmamp - VINHIGHKNEE VINLOWKNEE.qsch

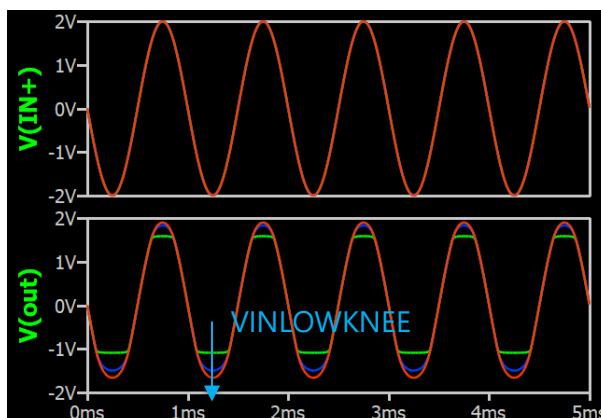
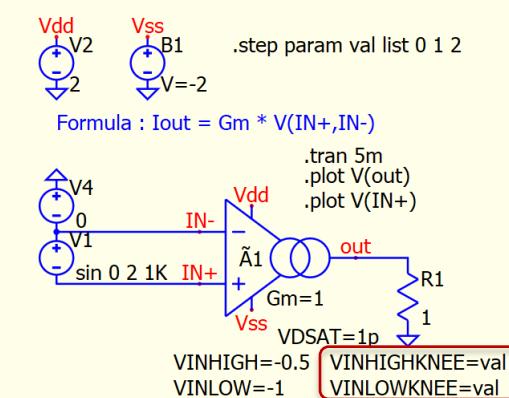
- VINLOW and VINHIGH

- VINLOW : Input range measured from negative rail
- VINHIGH : Input range measured from positive rail
- Default VINLOW=0**
- Default VINHIGH=0**
- Value is negative**, which limit input range. For example, VINHIGH=-0.5, Input HIGH is limited to $Vdd + VINHIGH = 2 - 0.5 = 1.5$ in this example



- VINLOWKNEE
VINHIGHKNEE

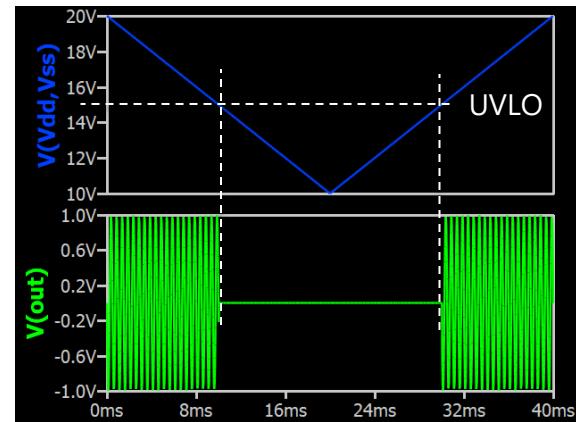
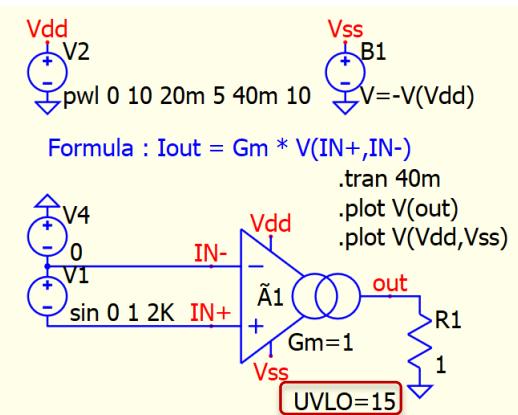
- Vinlowknee : Sharpness of negative input range limit
- Vinhightknee : Sharpness of positive input range limit
- Default VINHIGHKNEE=0**
- Default VINLOWKNEE=0**
- Increase KNEE soften the sharpness of input range, more output signal to come



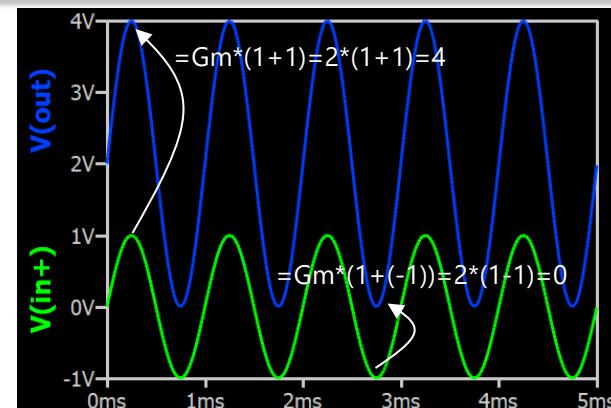
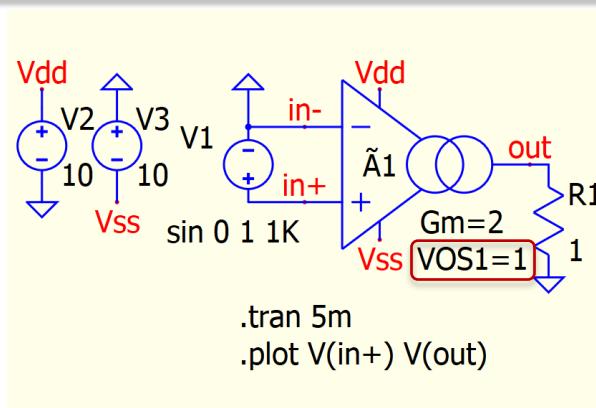
MultGmAmp : UVLO and VOS1

Qspice : Multgmamp - UVLO.qsch ; Multgmamp - VOS1.qsch

- UVLO
 - UVLO : Under Voltage Lock Out - Minimum supply voltage
 - Default UVLO is Infinite**
 - UVLO is compared to supply voltage = $V_{dd} - V_{ss}$



- VOS1
 - VOS1 : Offset voltage for input
 - $I_{out} = GM * (VOS1 + V(IN+, IN-))$
 - This parameter is not in HELP formula, but it actually implemented
 - Default VOS1 = 0**

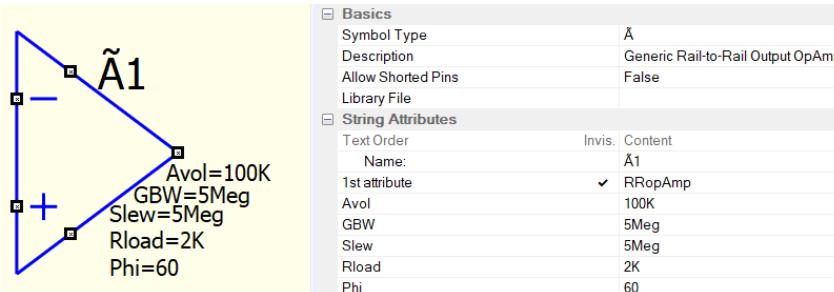


\tilde{A} -Device

Type : RRopAmp

\tilde{A} -Device RRopAmp Amplifier

- \tilde{A} -Device RRopAmp Amplifier
 - Syntax: $\tilde{A}nn VDD VSS OUT IN- IN+ MULT+ MULT- IN-- IN++ EN \# \# \# \# \# \# \# RROPAMP [INSTANCE PARAMETERS]$
 - A generic rail-to-rail output opamp specified by Avol, GBW, slew rate and phase margin. **It is implemented with two Gm amps** as above with an internal node for a Miller capacitor connected to the output.
 - Therefore, we can expect some parameters are carried from MULTGMAMP



Rail-to-Rail Output Op-Amp Device Instance Parameters		
Name	Description	Default
AVOL	Open loop voltage gain when loaded with RLOAD	1Meg
CAPINCM	Common eigenmode input capacitance	0.
CAPINNM	Normal eigenmode input capacitance	0.
CAPVDD	Capacitance from output to Vdd	0.
CAPVSS	Capacitance from output to Vss	0.01×CMILLER
CMILLER	Miller capacitance from output to internal node	1pF
EN	Equivalent input voltage noise density	0.
ENK	EN corner frequency	0.
GBW	Dominate pole gain-bandwidth	5MHz
IC	Initial condition of Vin × Vmult	none
IN	Equivalent input current noise density	0.
INF	Common mode input current noise density proportional to frequency	0.
INK	IN corner frequency	0.
IOUT	Maximum sourcing current	unlimited
ISNK	Maximum sinking current	unlimited
ISNKKNEE	Sharpness of max sinking current limit	0.1
ISRC	Max sourcing current	unlimited
ISRCKNEE	Sharpness of max sourcing current limit	0.1
M	Number of parallel devices	1.
PHI	Nominal phase margin ¹	90°
REF	Logic threshold for enable (from Vss)	(Vdd+Vss)÷2
RLOAD	Load resistance used in AVOL, GBW specification	2kΩ
ROUT	Unloaded output impedance	10kΩ
SLEW	Slew rate in Volts per second	5MV/s
TEMP	Instance temperature	circuit temperature
TTOL	Temporal tolerance for enable & UVLO	none
UVLO	Minimum supply voltage	0.
VCROSS	Cross conduction voltage range	1mV
VDSAT	Voltage where gm starts to switch over to a resistance	250mV
VDSAT1	Voltage where gm starts to switch over to a resistance (top FET)	250mV
VDSAT2	Voltage where gm starts to switch over to a resistance (bottom FET)	250mV
VINHIGH	Maximum valid input measured from positive rail	0.
VINHIGHKNEE	Sharpness of positive input range limit	100mV
VINLOW	Input range measured from negative rail	unlimited
VINLOWKNEE	Sharpness of negative input range limit	100mV
VOS1	Offset voltage for input	0.
VOUTMAX	Maximum output voltage measured from negative rail	unlimited
VOUTMIN	Minimum output voltage measured from negative rail	unlimited
ZX	Output impedance×(GBW÷frequency) in voltage follower configuration	100Ω

¹] The valid range is 90° to -55°. A positive number means it is unity-gain stable.

The RROPAMP device was the topic of [An Op-Ed on Op-Amp Modeling](#).

RropAmp : RLOAD, AVOL, GBW and PHI

Qspice : RROPAMP - AVOL.qsch | RROPAMP - GBW.qsch

RLOAD

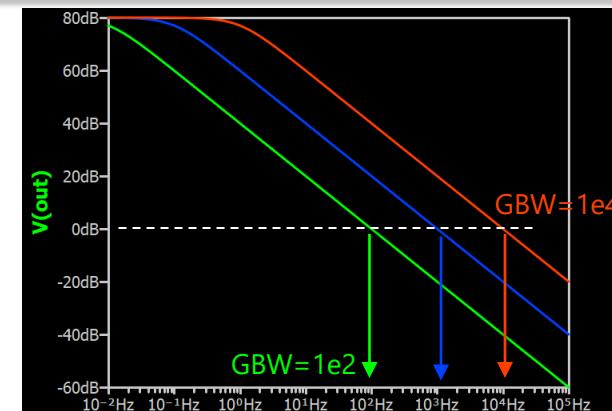
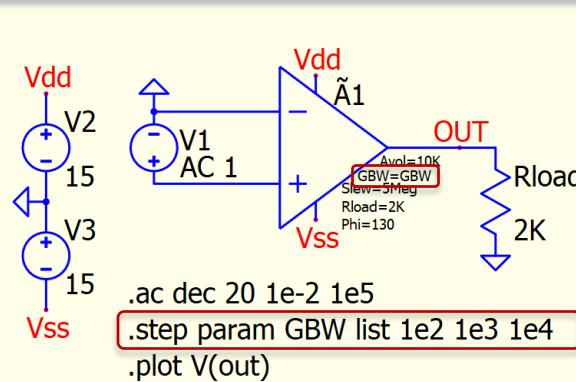
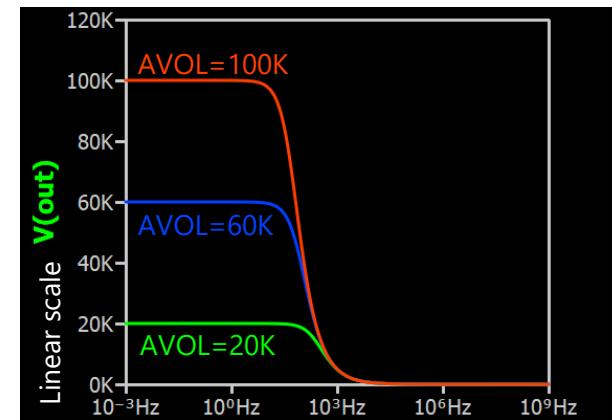
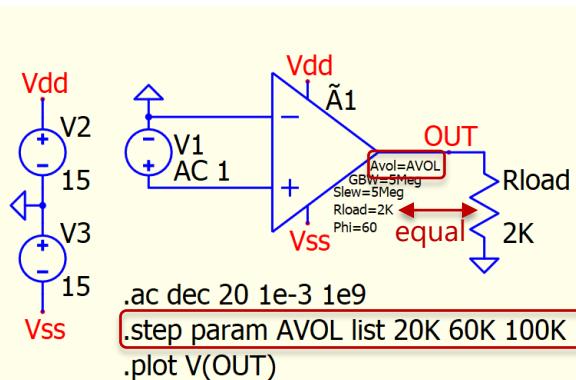
- Load resistance used in AVOL, GBW and PHI specification
- **Default Rload=2K**
- Output resistor with $R=Rload$ is required to obtain frequency response as according to these parameters specified

AVOL

- Open loop voltage gain when loaded with RLOAD
- **Default AVOL=1000k (120dB)**
- Gain = $\frac{V_{out}}{V_+ - V_-}$
- AVOL is gain @ 0Hz

GBW

- Dominant pole gain-bandwidth
 - GBW means frequency where gain of opamp falls to unity (0dB or 1)
- **Default GBW=5Meg**

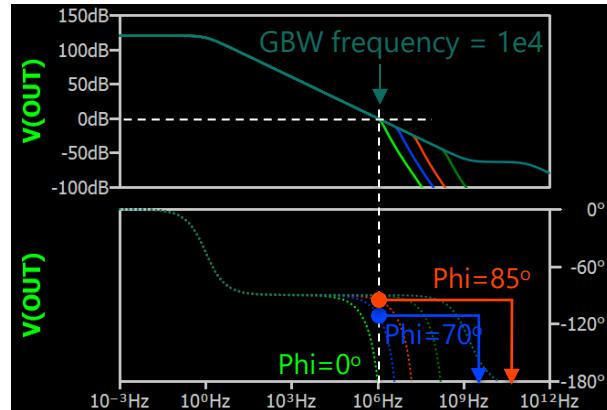
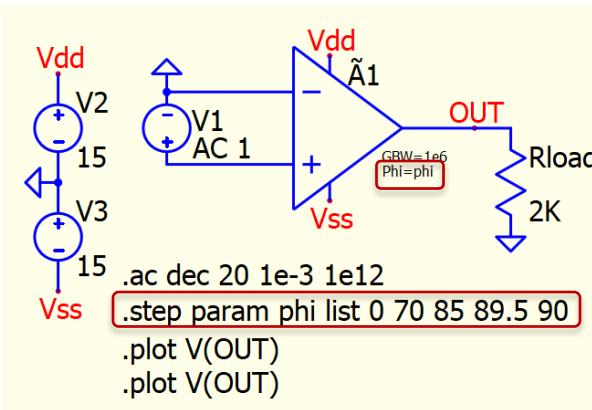


RropAmp : RLOAD, AVOL, GBW and PHI

Qspice : RROPAMP - PHI.qsch | RROPAMP - AVOL GBW PHI RLOAD.qsch

- PHI

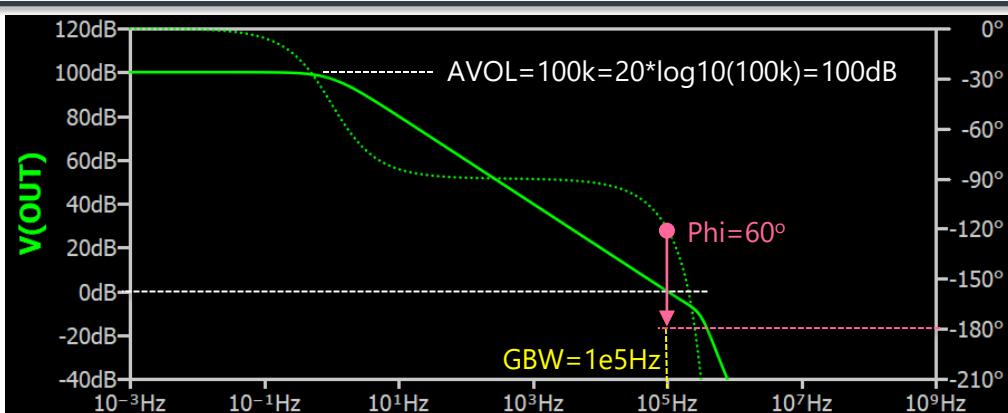
- Nominal phase margin
- **Default PHI=90°**
- Phase margin is measured of phase to -180° at 0dB (or at GBW frequency)
- PHI should limit in the range of $[-55^\circ, 90^\circ]$



- AVOL, GBW and PHI

- AVOL, GBW, PHI are three parameters to determine open-loop frequency response of RROPAMP
- RLOAD
 - These three parameters are specified at this Rload condition

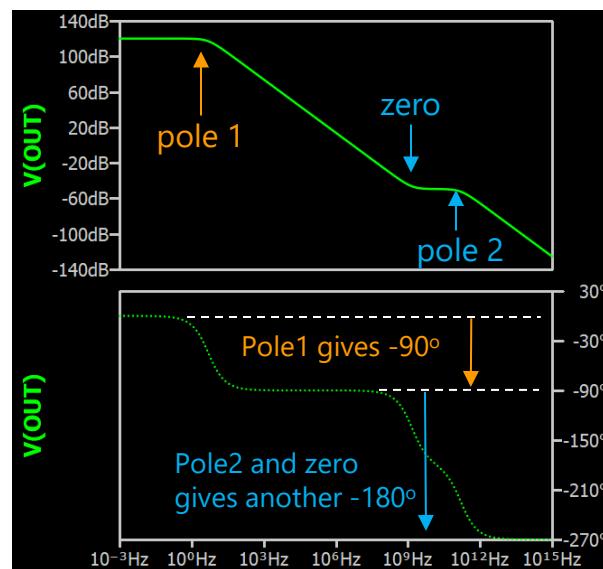
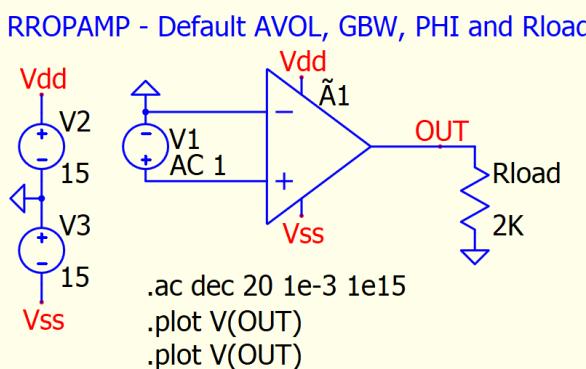
Avol=100K
GBW=1e5
Slew=5Meg
Rload=2K
Phi=60



RropAmp : RLOAD, AVOL, GBW and PHI

Qspice : RROPAMP - TF - Default.qsch

- RROPAMP models $G_{opamp}(s)$ with default value has 2nd order transfer function characteristic
 - Opamp Open-Loop Transfer Function : $v_{out} = G_{opamp}(s) (v_+ - v_-)$
 - RROPAMP model this transfer function with two poles and one zero
 - AVOL (open loop gain), GBW (gain bandwidth) and PHI (phase margin) are defined with RROPAMP loaded with a load resistor as RLOAD



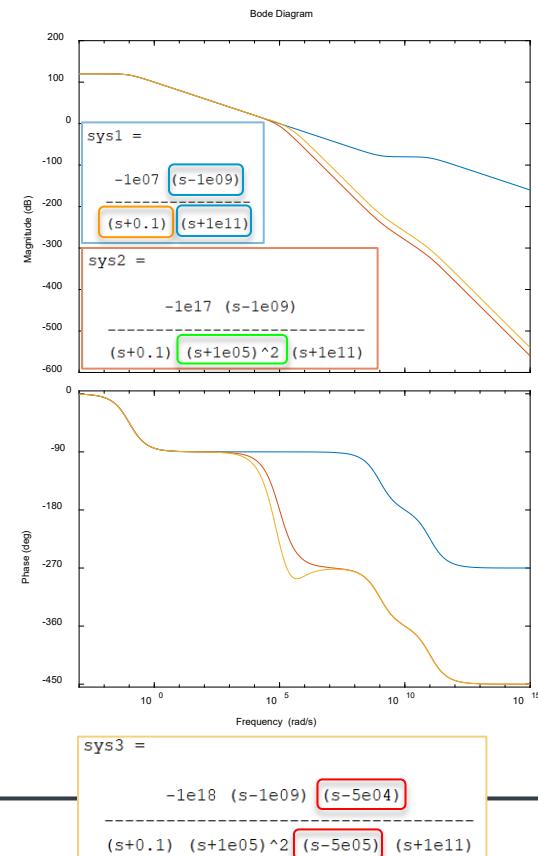
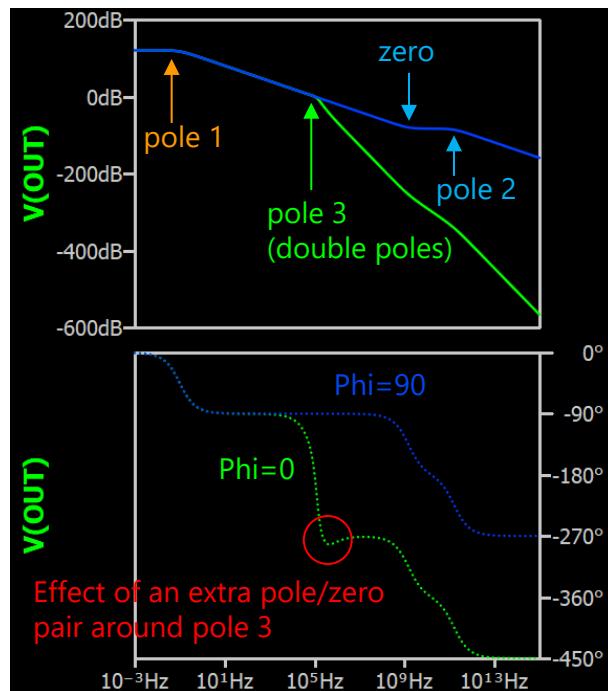
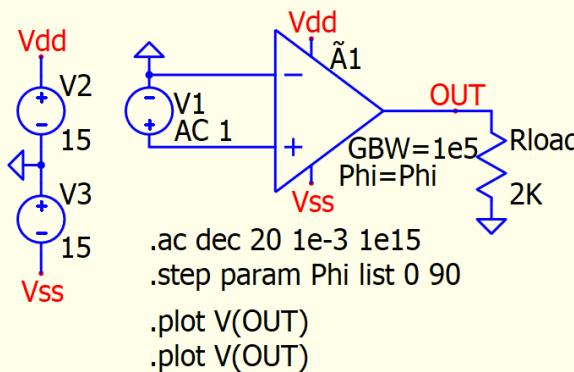
- **Pole 1**
 - Determine by **AVOL** and **GBW**
- **Pole 2 and Zero**
 - Determine by **Cmiller or ZX**
 - pole 2 and zero are apart by two decades which seems to be fixed by Qspice
 - PHI may add a double pole (pole 3) at crossover frequency to regulate phase margin

RropAmp : Two modeling mode

Qspice : RROPAMP - TF - Two Mode.qsch | RROPAMP - TF - Two Mode.m

- 2nd and 5th order
 - There are two mode in RropAmp behavior
 - 2nd order function with two poles and one zero
 - e.g. Phi=90 in this example
 - 5th order function with five poles and two zeros
 - e.g. Phi=0 in this example

RROPAMP - Default AVOL and Rload

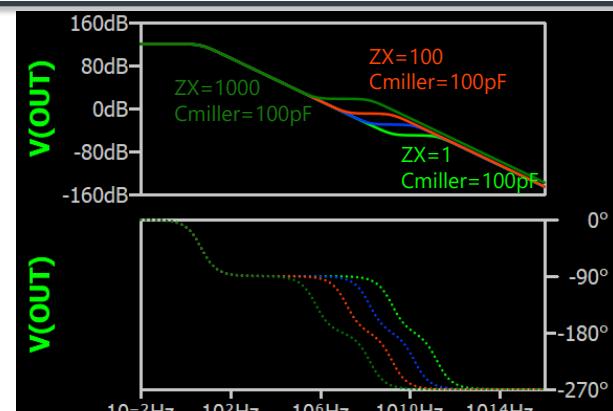
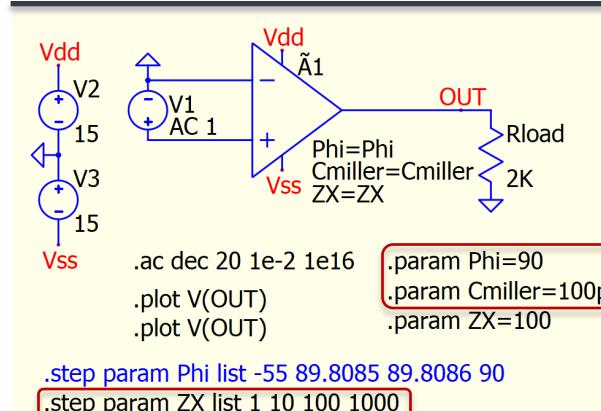
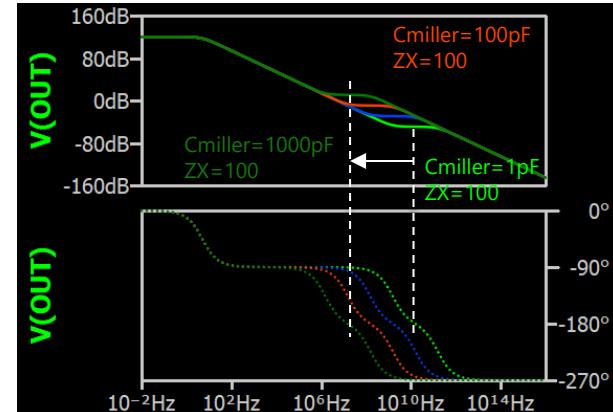
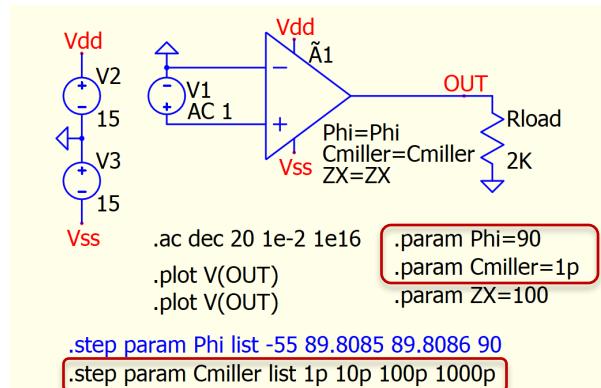


RropAmp : 2nd-Order with Cmiller and ZX

Qspice : RROPAMP - TF - 2nd-Order - Cmiller.qsch | RROPAMP - TF - 2nd-Order - ZX.qsch

- Mode : 2nd-Order
 - If $\Phi = 90^\circ$, open-loop bode is always behave as 2nd-order transfer function in regardless of Cmiller
 - Cmiller and ZX determines pole 2 and zero position
 - Cmiller : Miller capacitance from output to internal node
 - ZX : Output impedance in voltage follower configuration

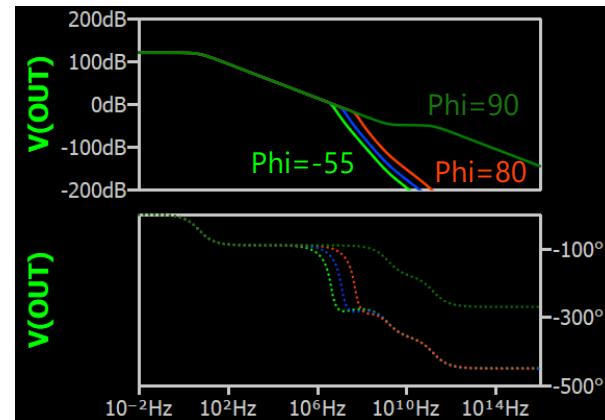
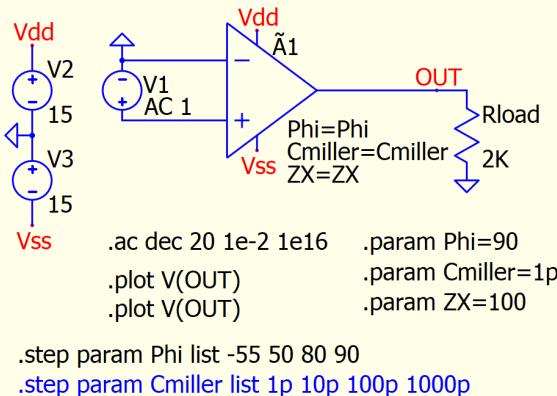
RropAmp with Default Phi=90, Cmiller=1p	mode 1 (2nd order)	mode 2 (5th order)
Default Cmiller=1p	Phi = [89.8086, 90]	Phi = [-55, 89.8085]
Default Phi=90	Cmiller = any value	---
Phi=80	Cmiller = [1p 52p]	Cmiller >= 53pF
Phi=60	Cmiller = [1p 172p]	Cmiller >= 173pF
Phi=40	Cmiller = [1p 356p]	Cmiller >= 357pF
Phi=20	Cmiller = [1p 822p]	Cmiller >= 823pF
Phi=10	Cmiller = [1p 1696p]	Cmiller >= 1697pF
Phi=0	---	Cmiller = any value



RropAmp : 5th-Order with PHI

Qspice : RROPAMP - TF - 5th-Order - Phi.qsch

- Mode : 5th-Order
 - If $\Phi < 89.8085^\circ$, depends on Cmiller (or ZX), bode in general will behave as 5th-order transfer function
 - The dominate pole with Φ is double poles (pole 3) near crossover frequency to regulate phase margin
 - It will give a 60dB/decade roll-off after crossover
 - Φ : Nominal phase margin

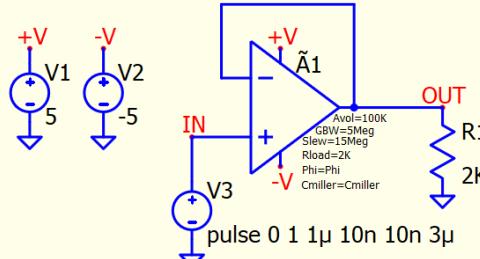


RropAmp with Default $\Phi=90$, $C_{miller}=1p$	mode 1 (2nd order)	mode 2 (5th order)
Default $C_{miller}=1p$	$\Phi = [89.8086, 90]$	$\Phi = [-55, 89.8085]$
Default $\Phi=90$	$C_{miller} = \text{any value}$	---
$\Phi=80$	$C_{miller} = [1p 52p]$	$C_{miller} \geq 53pF$
$\Phi=60$	$C_{miller} = [1p 172p]$	$C_{miller} \geq 173pF$
$\Phi=40$	$C_{miller} = [1p 356p]$	$C_{miller} \geq 357pF$
$\Phi=20$	$C_{miller} = [1p 822p]$	$C_{miller} \geq 823pF$
$\Phi=10$	$C_{miller} = [1p 1696p]$	$C_{miller} \geq 1697pF$
$\Phi=0$	---	$C_{miller} = \text{any value}$

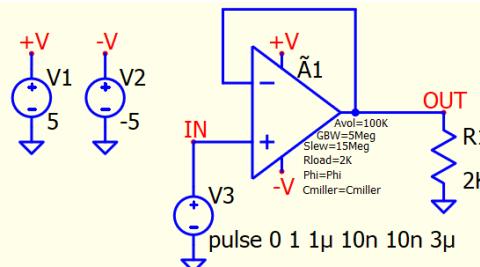
RropAmp : Phi and Cmiller in Time Domain

Qspice : RROPAMP - PHI Cmiller (.tran).qsch

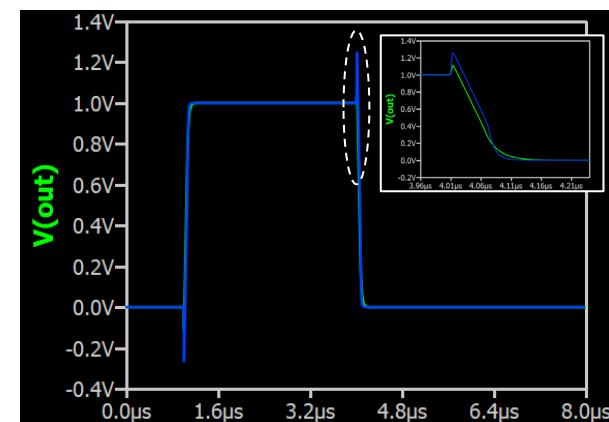
- Phi and Cmiller in Time Domain
 - Phi is the phase margin, which determines the transient response of the system. For example, 30 degrees will result in overshoot, while 90 degrees will lead to critically damped behavior
 - Cmiller is the Miller capacitance, which may introduce a reverse direction step in the output



```
.tran 0 8μ 0 1n .param Phi=90  
.plot V(out) .param Cmiller=1p  
.step param Phi list 30 45 90  
.step param Cmiller list 1p 100p
```



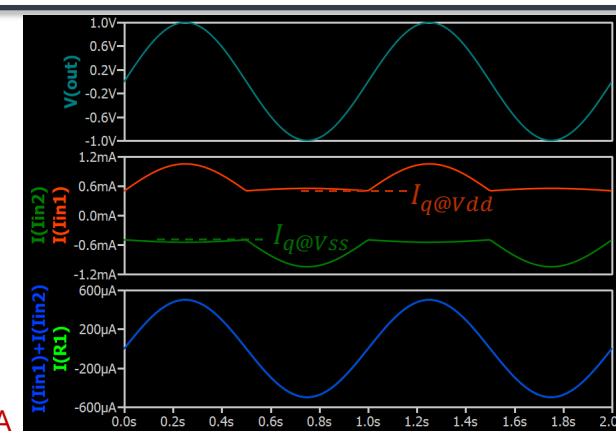
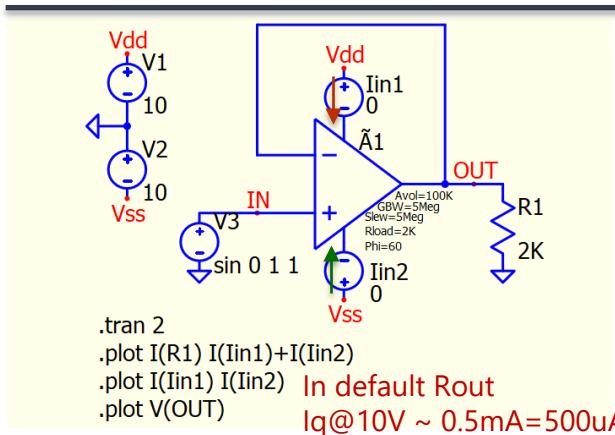
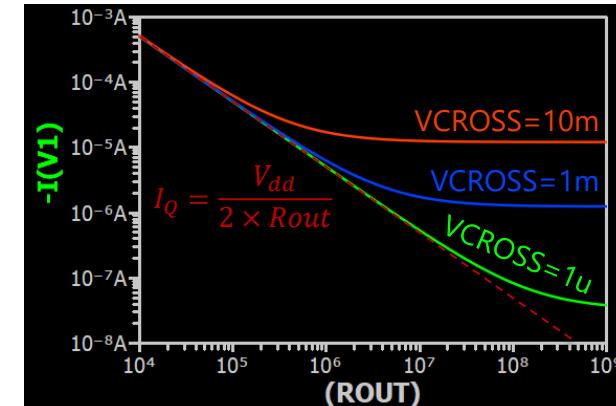
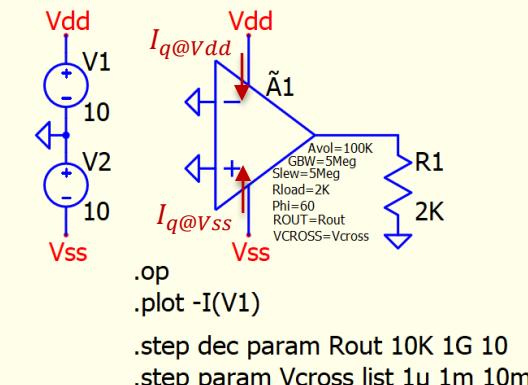
```
.tran 0 8μ 0 1n .param Phi=90  
.plot V(out) .param Cmiller=1p  
.step param Phi list 30 45 90  
.step param Cmiller list 100p 200p
```



RRopAmp – ROUT and VCROSS (Quiescent Current)

Qspice : RROPAMP - Rout Vcross (Quiescent Current).qsch

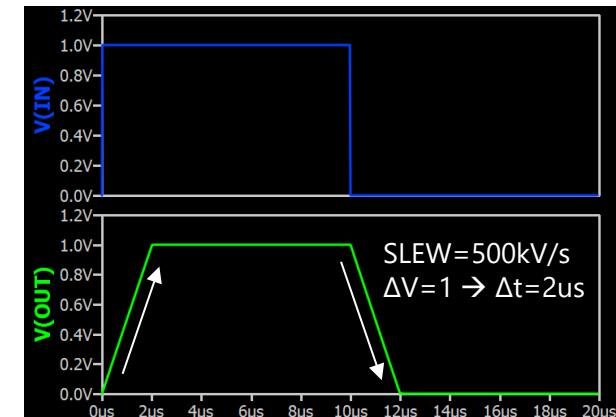
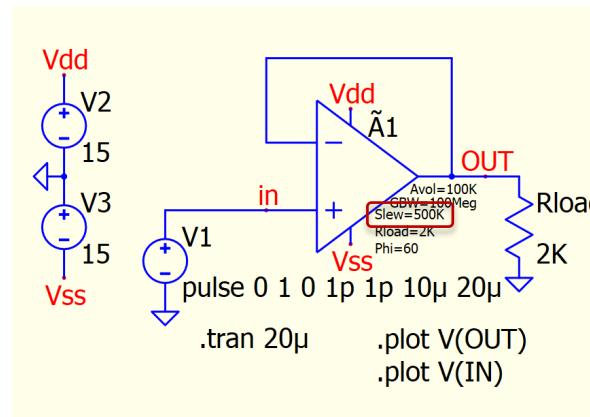
- Rout and VCROSS
 - ROUT : unloaded output impedance (default = 10kohms)
 - VCROSS : Cross conduction voltage range (default = 1mV)
 - Rout and Vcross affect quiescent current level
 - Quiescent current (I_q) is normally dominated by Rout, where $I_{q@V_{dd}} = \frac{V_{dd}}{2R_{out}}$ and $I_{q@V_{ss}} = \frac{V_{ss}}{2R_{out}}$ and VCROSS control minimum quiescent current if Rout approaching a very large value



RropAmp : SLEW

Qspice : RROPAMP - SLEW.qsch

- SLEW
 - Slew rate in Volts per second (maximum change rate)
 - $SLEW = \frac{\Delta V_{out}}{\Delta t}$
- **Default SLEW=5Meg**



¥-Device

¥-Device

- ¥-Device
 - Syntax:

`¥nnn N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15 N16 <TYPE> [INSTANCE PARAMETERS]`

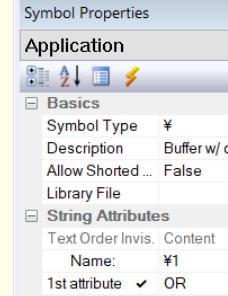
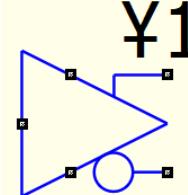
- The ¥-device supplies gate, flop, and a few other types of functional behaviors. The device expects exactly 16 pins, but not all are used. Unused pins must be specified connected to "¥".

¥-Device Types

TYPE	Behavior
AND	AND gate
CLOCKSYNC	Selects between a Sync and Clock inputs
D-FLOP	D-type flip-flop
EXTOSC	Oscillator programmed with an external resistor
HMITT	Schmitt trigger
JK-FLOP	JK-type flip-flop
MONOSTABLE	Retriggerable monostable
OR	OR gate
PS-FLOP	SMPS flip-flop
RS-FLOP	RS-type flip-flop
T-FLOP	Toggle flip-flop
XOR	XOR gate
Φ-DET	Phase/Frequency detector

To Identify what <TYPE> a symbol is

- In Symbol Properties > 1st attribute
- View > Netlist : from device syntax

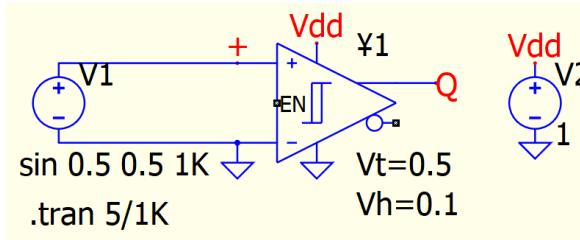


```
* C:\Users\kelvinleung\Documents\QSPICE\Unt
¥1 ¥0 ¥1 ¥2 ¥3 ¥4 ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ OR
.end
```

¥-Device : How EN pin works

Qspice : ¥-device - EN.qsch

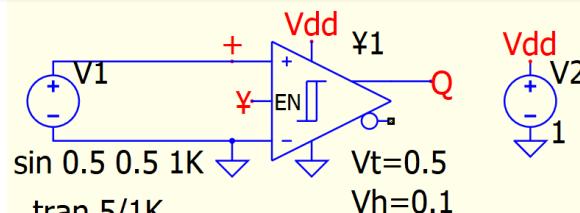
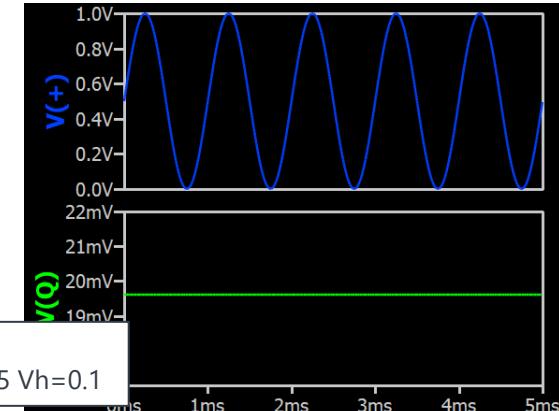
- How EN works
 - If EN is preset in symbol, normally, a signal is used to control its status. Here discuss two situations
 - #1 No connect to EN pin
 - Netlist assign ¥n as net name. EN pin is active to monitor EN signal. In this case, no output will generate
 - #2 Assign EN pin with net ¥
 - ¥ net name is equivalent to unassigned in ¥-device
 - Unassigned EN pin will be considered as Default ENABLE (except Latch)
 - Therefore, a symbol can be created with pin assigned to ¥ for their default



Example : No Connection

Netlist

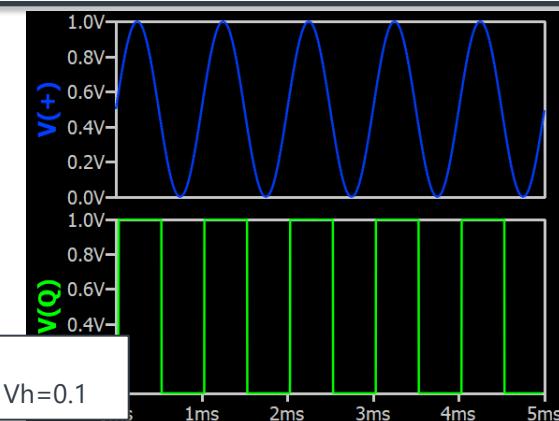
¥1 Vdd 0 Q ¥0 + 0 ¥1 ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ HMITT Vt=0.5 Vh=0.1



Example : Assign EN with net ¥

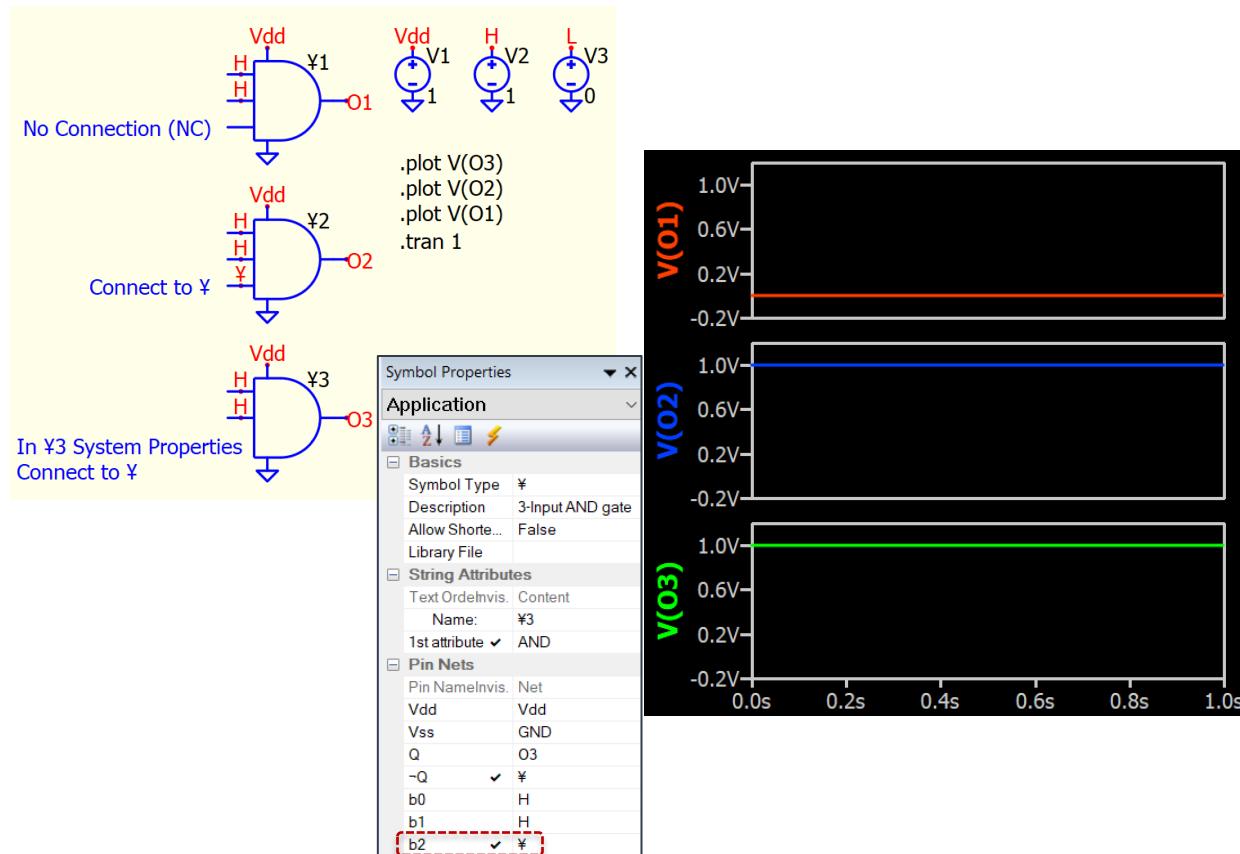
Netlist

¥1 Vdd 0 Q ¥0 + 0 ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ HMITT Vt=0.5 Vh=0.1



¥-Device : Net ¥

- Net ¥
 - Unconnected Pin and Pin connected to Net ¥ is different
 - Net ¥ is a special net name, in HELP of ¥-Device, it specified that unused pins must be specified connected to "¥"
 - Two methods to force a pin to ¥
- This is an example of AND gate
 - Device ¥1 : NC pin is equivalent to 0V and AND gate output 0
 - Device ¥2 : ¥ is assigned as Net in schematic and this input specified unused for this AND gate
 - Device ¥3 : ¥ is assigned but in System Properties, this input specified unused and become invisible in symbol



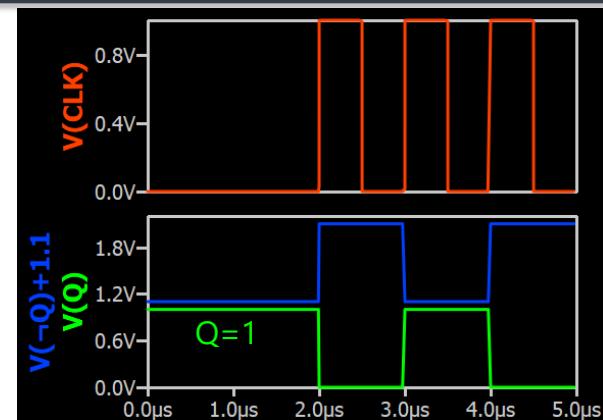
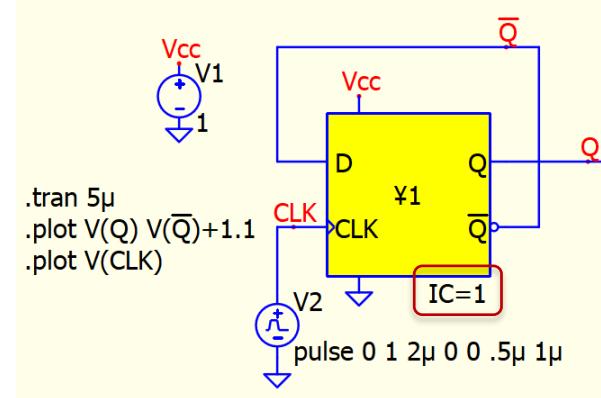
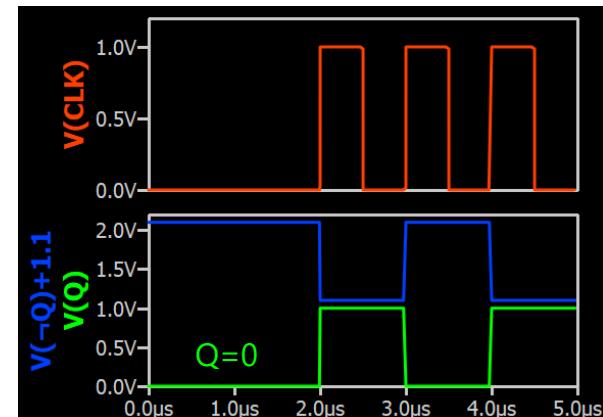
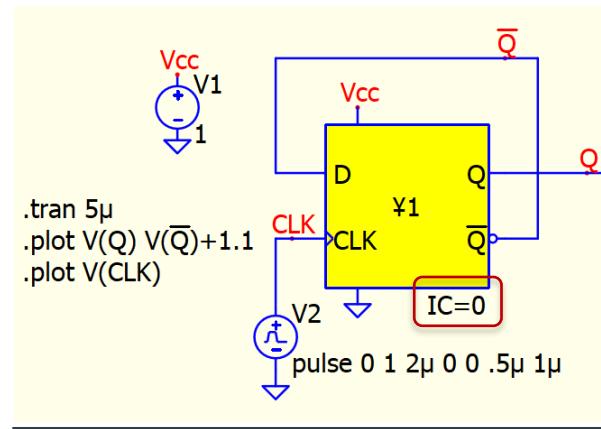
¥-Device : HELP > Simulator > Device Reference > ¥-Device
 Common Instance Parameters (example from AND Gate Instance Param)

Name	Description	Units	Default
CAPVDD	Capacitance from an output to Vdd	F	0.
CAPVSS	Capacitance from an output to Vss	F	0.
IC	Initial condition(needed, e.g., for a ring osc)		
M	Number of parallel devices		1.
REF	Logic reference voltage	V	$(Vdd + Vss) \div 2$
RSINK	Resistance to Vss when output high	Ω	RSRC
RSRC	Resistance to Vdd when output low(aka ROUT)	Ω	100.
TD	Delay(aka TD1)	s	0.
TD2	Asymmetrical delay	s	TD
TEMP	Instance temperature	$^{\circ}C$	27.
TFALL	Fall time	s	0.
TRISE	Rise time	s	0.
TTOL	Temporal tolerance	s	$1\mu s$
UVLO	Minimum Vdd-Vss voltage to operate	V	0.
ZMULT	Impedance multiplier when biased half way		1.

¥ Instance Params : IC (Initial Condition)

Qspice : ¥-device - IC (gate).qsch

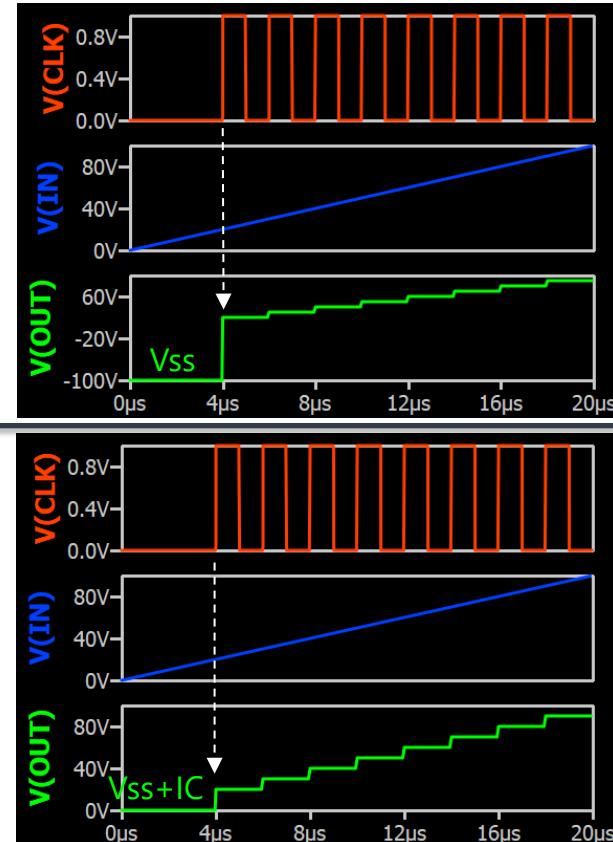
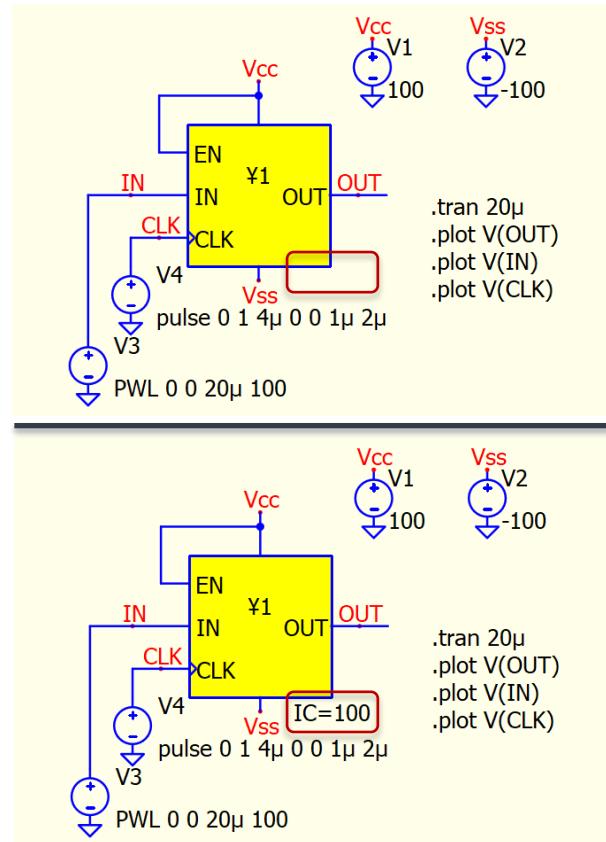
- IC
 - Initial Condition
 - ¥-device output follows initial condition at t=0s, which is needed in condition like logic gate setup to work as oscillator where its output stage is uncertain at t=0s



¥ Instance Params : IC (Initial Condition)

Qspice : ¥-device - IC (latch).qsch

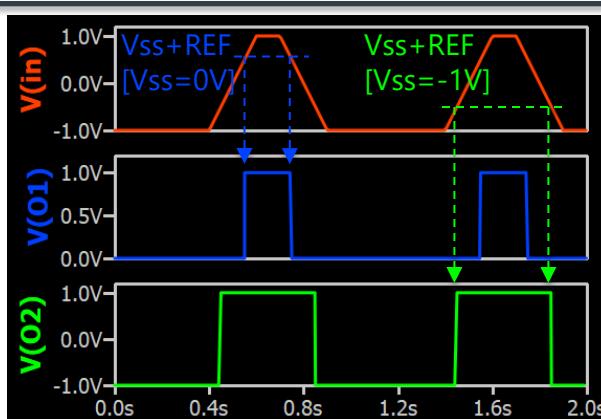
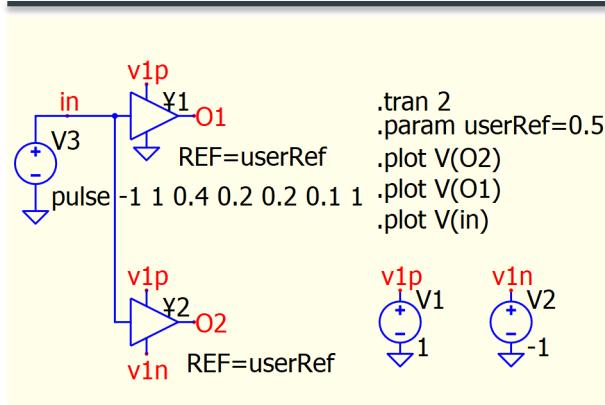
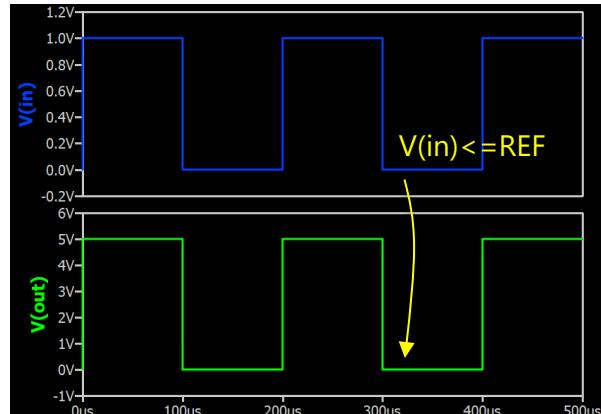
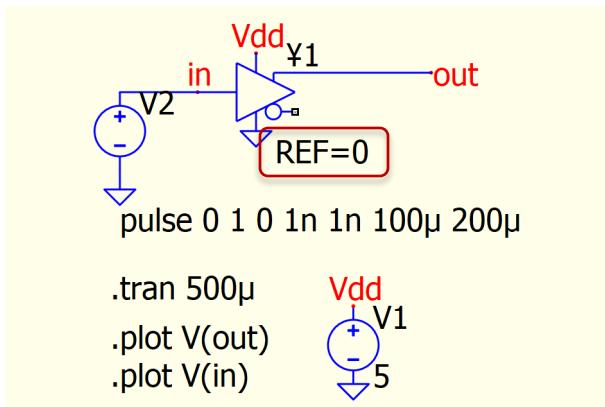
- IC for analog latch
 - An analog latch is a sample and hold device. Initial condition can also be applied to an analog latch, but it operates differently from a logic gate output
 - The latch samples and holds the input voltage at the rising clock edge. Before the first rising edge, its output voltage follows the Vss supply voltage if IC is not provided
 - IC is referenced to the Vss voltage. In this example, assigning IC=100 allows the output to be 0V (=Vss+IC) before the first clock rising edge.



Instance Params : REF (Logic Reference Voltage)

Qspice : -device - REF.qsch | -device - REF (Dual Supply).qsch

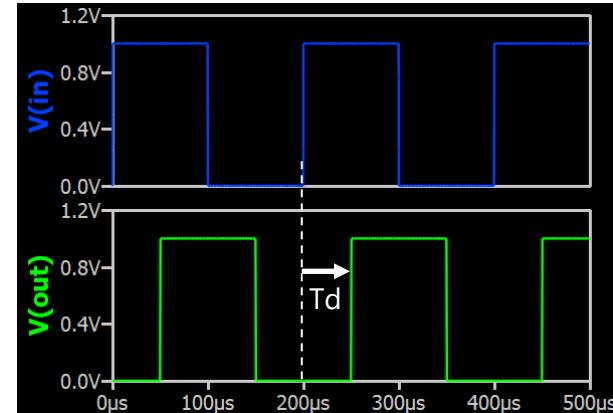
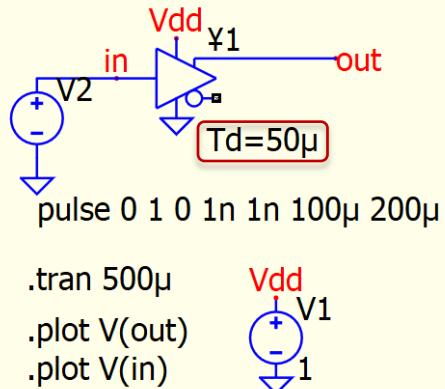
- REF
 - Logic Reference Voltage
 - **Default REF** = $\frac{Vdd + Vss}{2}$
 - Threshold definition
 - Low : $V(IN) \leq \text{REF}$
 - High : $V(IN) > \text{REF}$
 - Special Note
 - REF is not absolute voltage to node 0, but reference to VSS node
 - In this example, both devices have same REF, but as VSS with different potential, their logic threshold are different



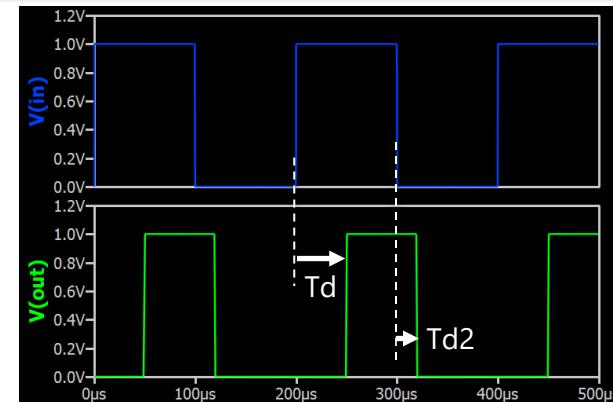
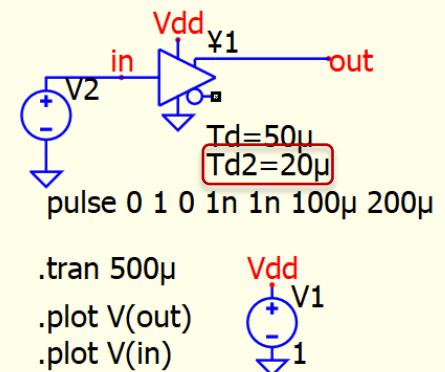
Instance Params : TD (Delay) and TD2 (Asymmetrical Delay)

Qspice : $\$$ -device - Td.qsch | $\$$ -device - Td Td2.qsch

- TD (Delay)
 - Td : Delay
 - **Default TD=0s**
 - ** If TD2 is not set, both rising and falling delay times are same
 - ** If TD > logic H duration or delay TD2 > logic L duration, output always LOW or HIGH



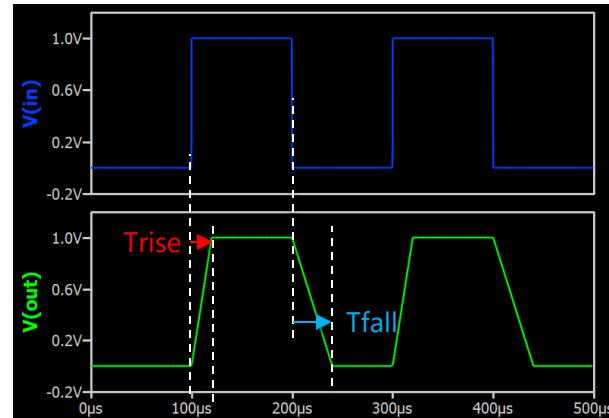
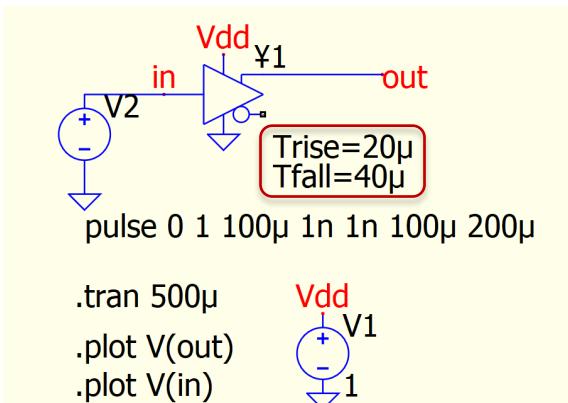
- TD2 (Asymmetrical delay)
 - TD2 : Asymmetrical delay
 - **Default TD2=TD**
 - If different delay times for rising and falling edge, set TD2 for falling edge delay time
(i.e. TD only control rising edge delay time)



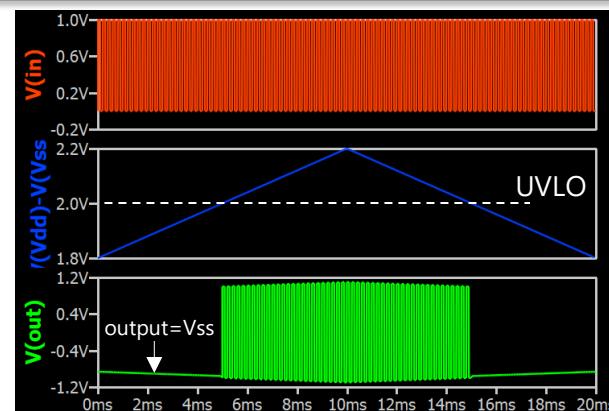
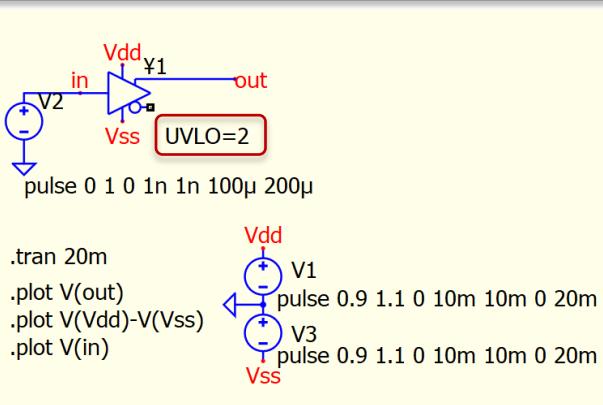
Instance Params : Trise (Rise time), Tfall (Fall time) and UVLO

Qspice : -device - Trise Tfall.qsch | -device - UVLO.qsch

- TRISE and TFALL
 - Trise : Rise time
 - Tfall : Fall time
 - **Default TRISE=0s**
 - **Default TFALL=0s**



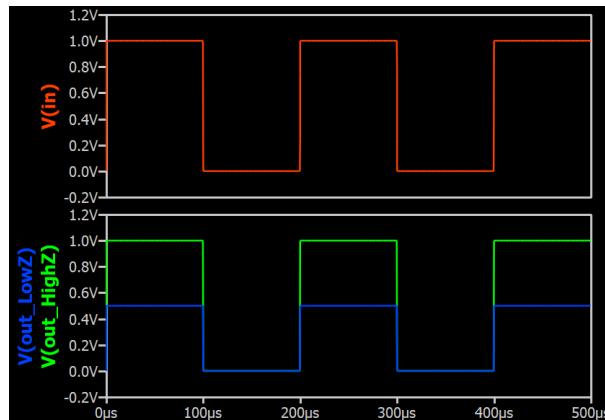
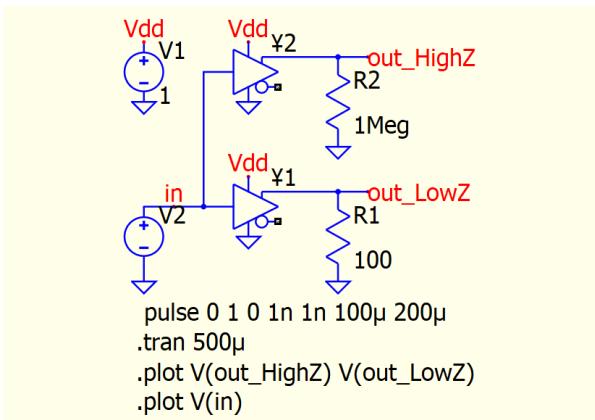
- UVLO
 - UVLO : Under Voltage Lock Out
 - **Default UVLO=0V**
 - UVLO : Minimum $V_{dd}-V_{ss}$ voltage to operate
 - Output only enabled when $V_{dd}-V_{ss} > \text{UVLO}$
 - Output equals V_{ss} when gate disable



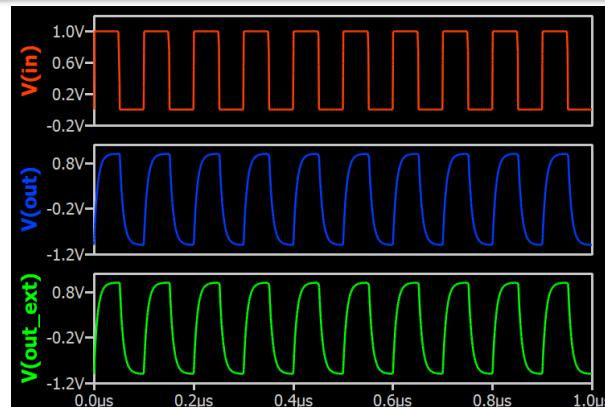
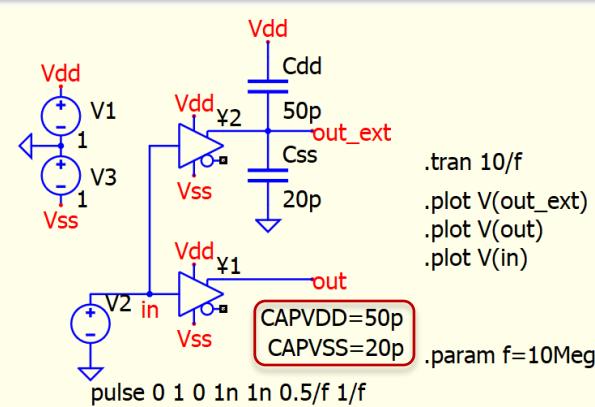
Instance Params : Output R and C (Rsink, Rsrc, Capvdd, CapVss)

Qspice : ¥-device - RSINK RSRC.qsch | ¥-device - Capvdd Capvss.qsch

- RSINK and RSRC
 - RSINK : Res to Vss when o/p high
 - RSRC : Res to Vdd when o/p low
 - **Default RSINK=100Ω**
 - **Default RSRC=100Ω**
- Example
 - This simulation shows loading effect when output to a 100 ohms, where o/p level is reduced to half of Vdd



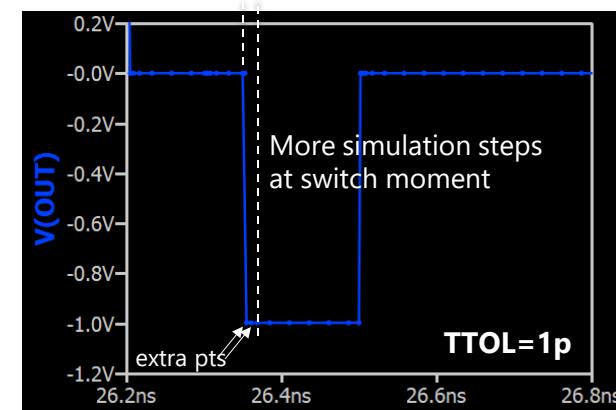
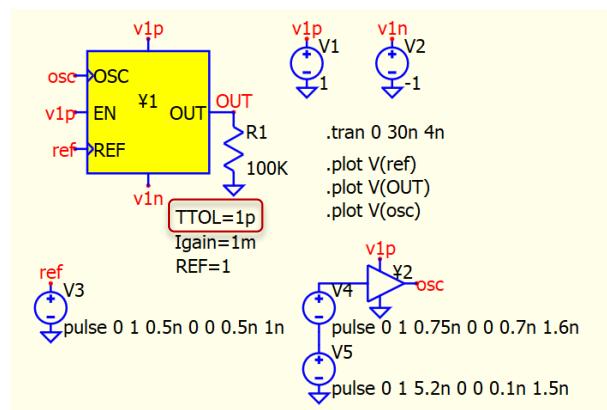
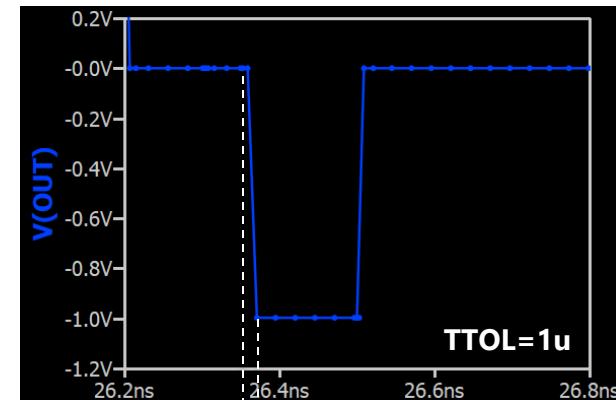
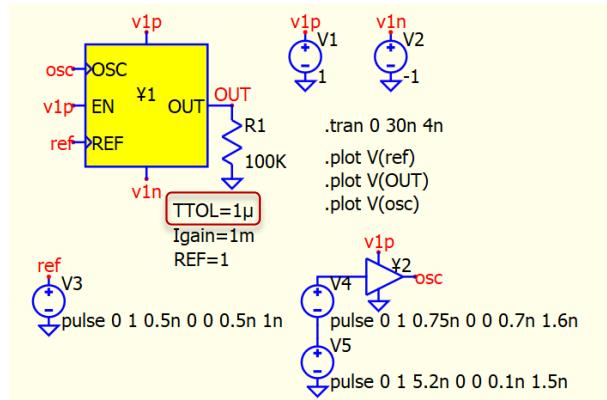
- CAPVDD, CAPVSS
 - Capvdd : Capacitance from an output to Vdd
 - Capvss : Capacitance from an output to Vss
- Explanation
 - Capacitance Capvdd/Capvss equivalent Cdd/Css this example
 - This demo can have frequency response because Rsink=Rsrc=100 internally



Instance Params : TTOL (Temporal Tolerance)

Qspice : \$-device - TTOL.qsch

- TTOL
 - TTOL : Temporal tolerance
 - **Default TTOL=1u**
 - TTOL allows one to determine how accurately the switch time should be found



¥ Truth Table : AND, OR, XOR

Qspice : Truth Table of AND OR XOR.qsch

- AND

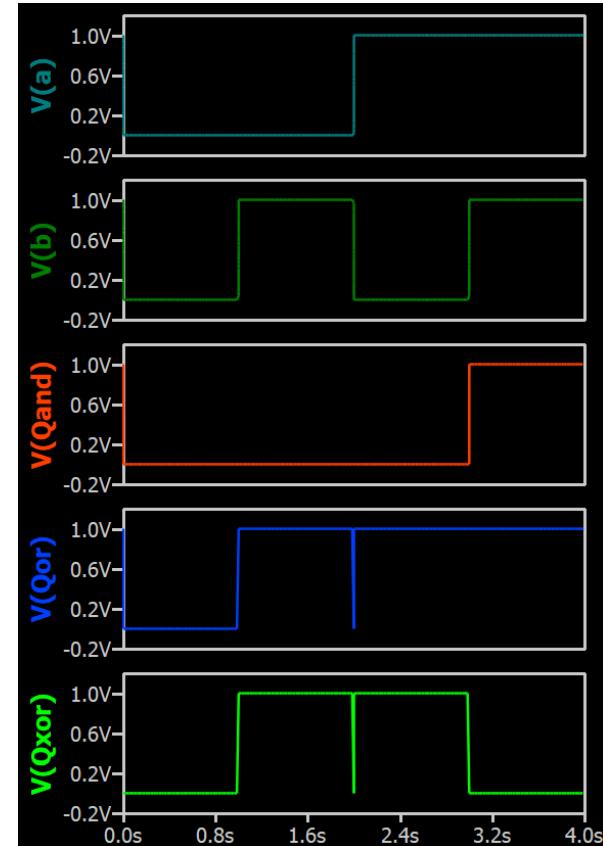
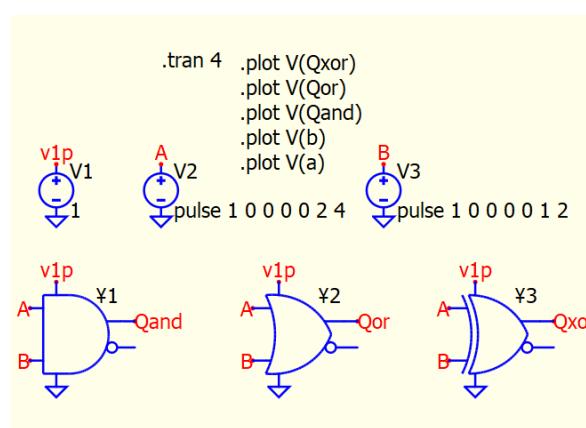
¥-Device AND			
A	B	Q	_Q
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

- OR

¥-Device OR			
A	B	Q	_Q
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

- XOR

¥-Device XOR			
A	B	Q	_Q
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1



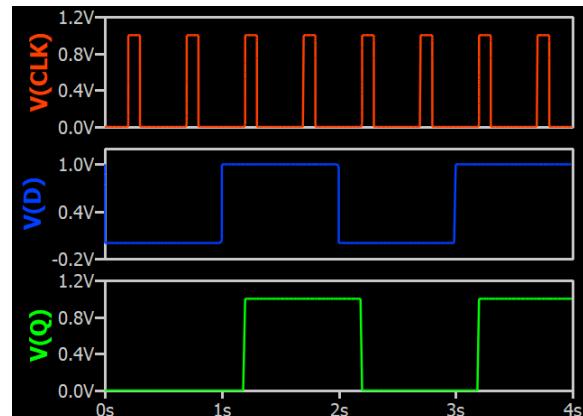
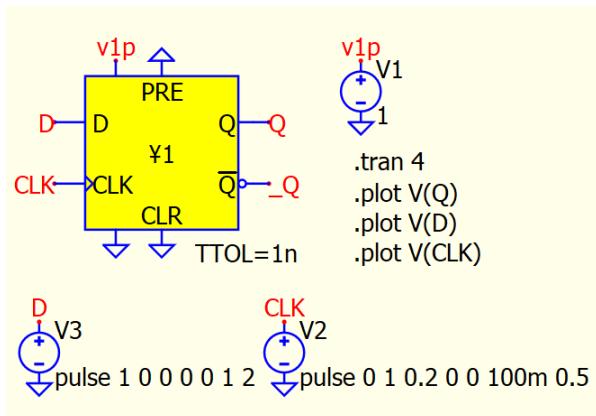
¥ Truth Table : D-FLOP and T-FLOP

Qspice : Truth Table of D-FLOP.qsch ; Truth Table of T-FLOP.qsch

- D-FLOP

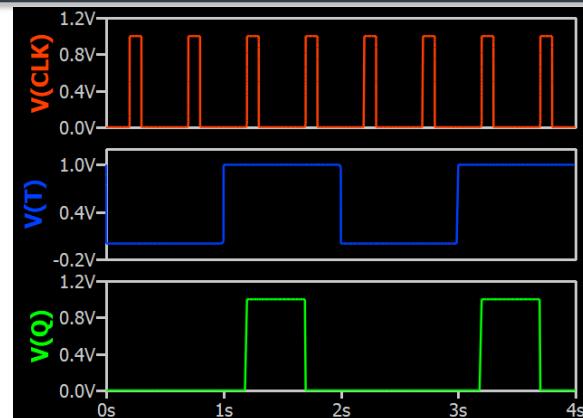
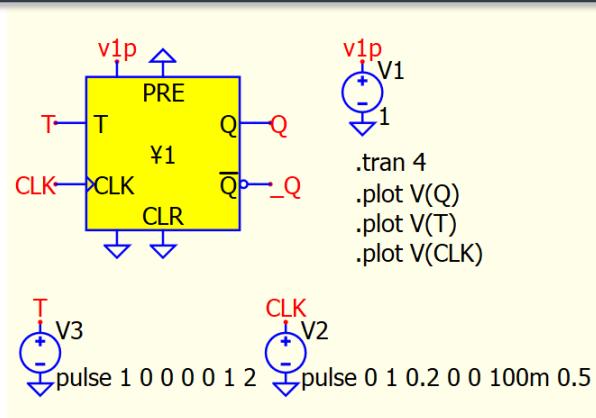
- PRE : Active High (=SET)
- CLR : Active High (=RESET)

¥-Device D-flop					
CLK	D	PRE	CLR	Q	\bar{Q}
↑	0	0	0	0	1
↑	1	0	0	1	0
x	x	0	0	Q	\bar{Q}
x	x	1	0	1	0
x	x	x	1	0	1



- T-FLOP

¥-Device T-flop				
CLK	T	PRE	CLR	Q_{n+1}
↑	0	0	0	0
↑	1	0	0	\bar{Q}_n
x	x	0	0	Q
x	x	1	0	1
x	x	x	1	0



¥ Truth Table : SR-FLOP and JK-FLOP

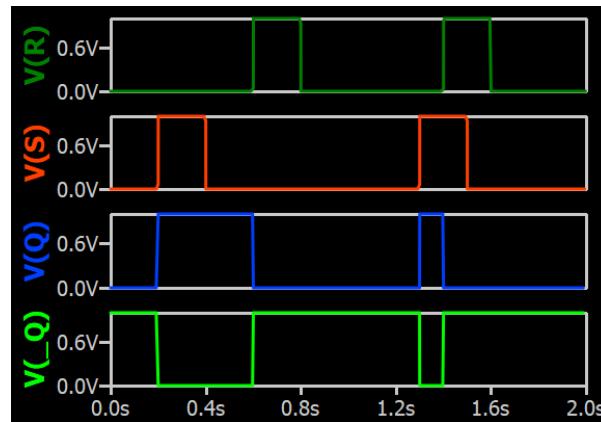
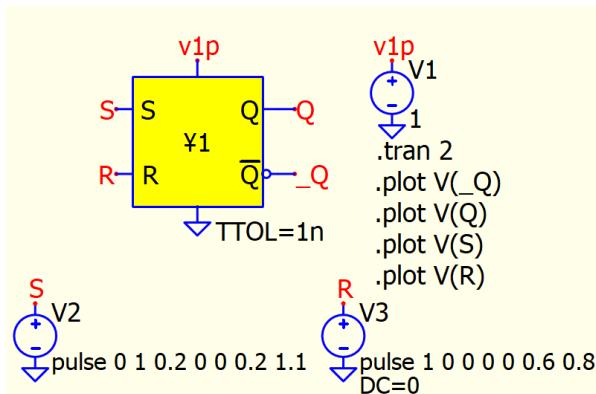
Qspice : Truth Table of SR-FLOP.qsch ; Truth Table of JK-FLOP.qsch

- SR-FLOP

R	S	Q	\bar{Q}
0	0	Unchange	
0	1	1	0
1	0	0	1
1	1	0	1

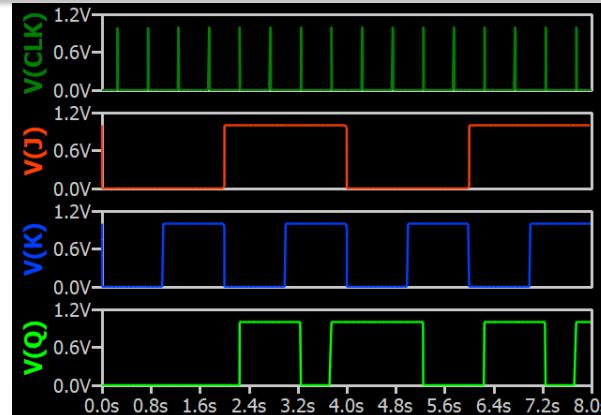
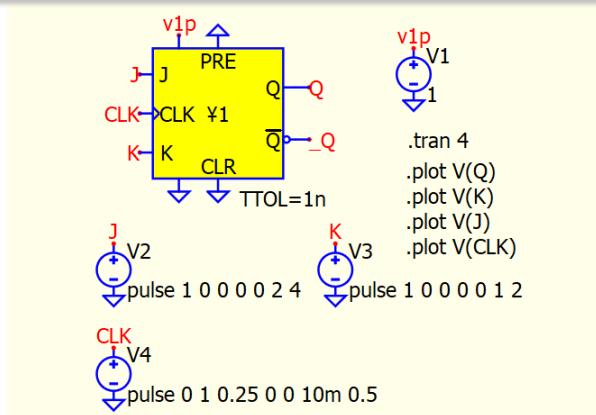


This state is generally declared as not allowed



- JK-FLOP

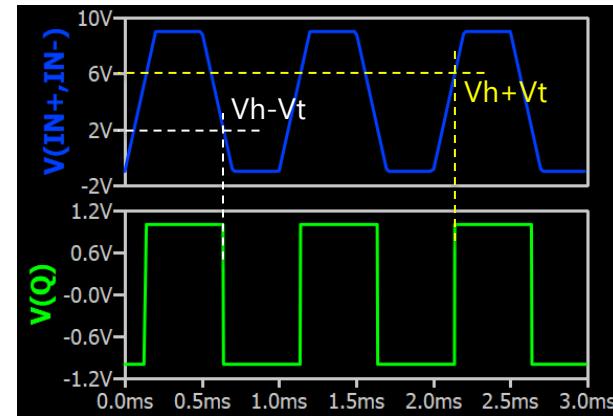
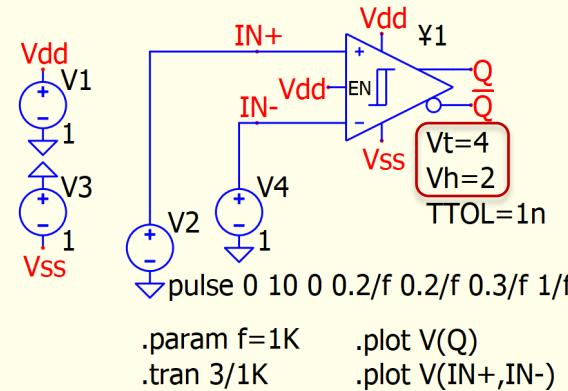
CLK	J	K	PRE	CLR	Q_{n+1}
↑	0	0	0	0	Q_n
↑	0	1	0	0	0
↑	1	0	0	0	1
↑	1	1	0	0	$_Q_n$
x	x	x	0	0	Q
x	x	x	1	0	1
x	x	x	x	1	0



¥-Device : Schmitt Trigger (HMITT) and Instance Parameters VT VH

Qspice : Type - HMITT (Vt Vh).qsch

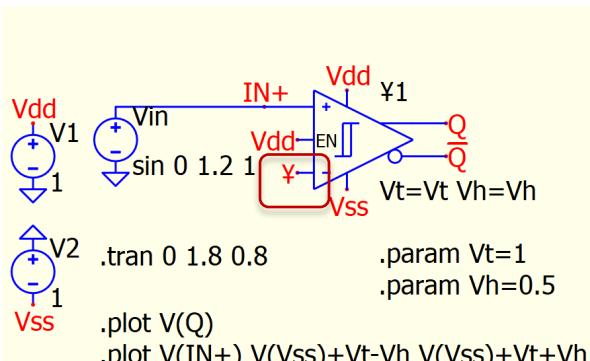
- Schmitt Trigger
 - HMITT is schmitt trigger logic input
- VT and VH
 - Vt : Threshold voltage
 - Vh : Half hysteresis voltage
 - Hysteresis for differential
 $V(IN+,IN-) = V(IN+) - V(IN-)$
 - True : $V(IN+,IN-) > VT + VH$
 - False : $V(IN+,IN-) < VT - VH$
 - Supply voltage Vdd and Vss only affect HMITT output but not input comparison



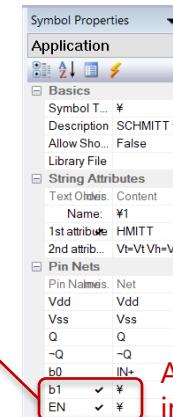
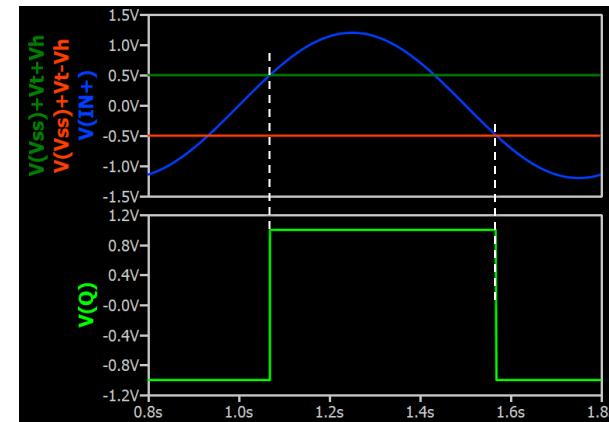
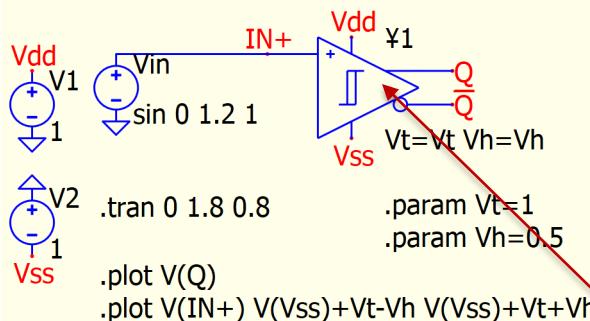
¥-Device : Schmitt Trigger - IN- to ¥ node, EN to ¥ node

Qspice : Type - HMITT (IN- to ¥).qsch | Type - HMITT (EN to ¥).qsch

- IN- connect to ¥
 - Schmitt trigger comparator allows IN- connect to ¥
 - This is equivalent IN- connect to Vss
 - Schmitt trigger output
TRUE : $V(IN+) > V(VSS) + VT + VH$
FALSE : $V(IN+) < V(VSS) + VT - VH$



Identical Setup

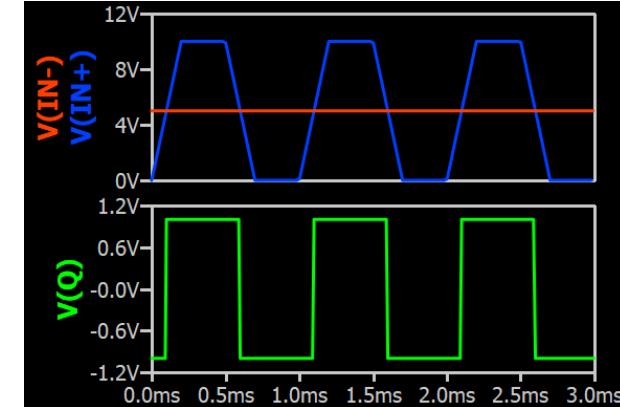
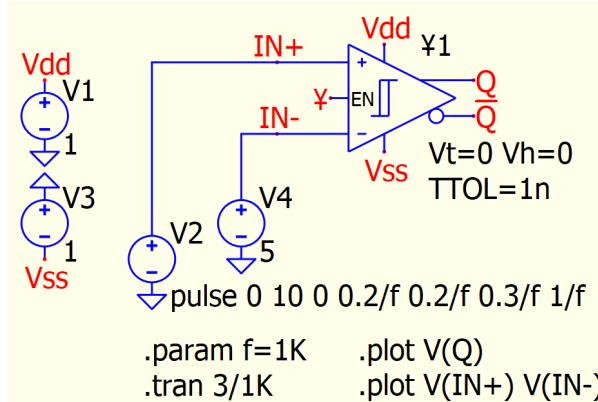


Assign ¥ in Pin Nets
in Symbol Properties

¥-Device : Schmitt Trigger – Simple Comparator

Qspice : Type - HMITT (Simple Comparator).qsch

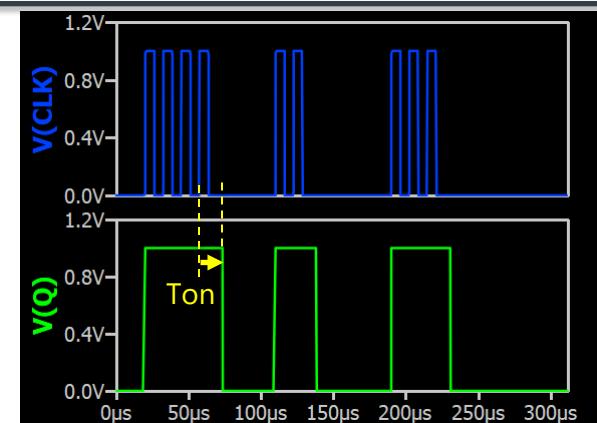
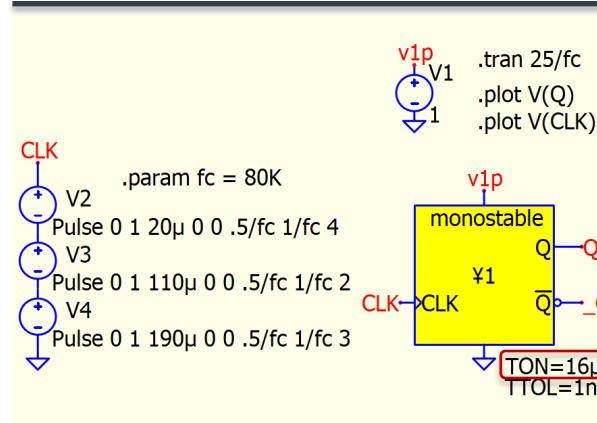
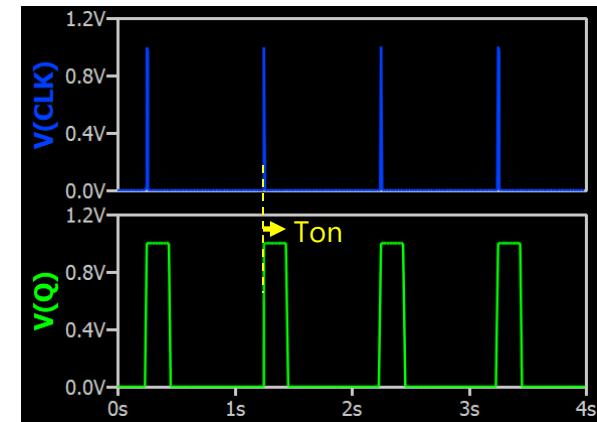
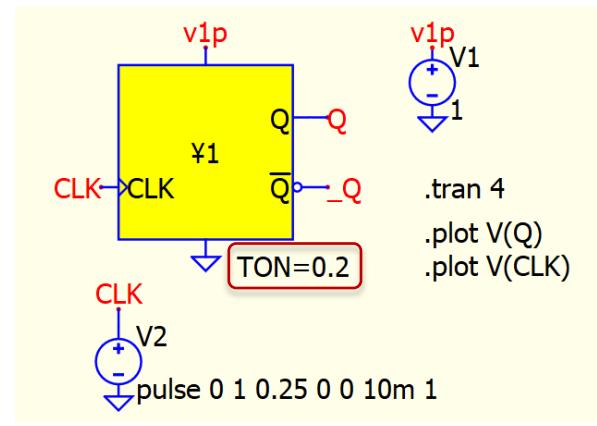
- Simple comparator
 - A simple comparator can be built with $V_t=0$ and $V_h=0$, which are the default values for V_t and V_h
 - It will only compare the output based on the relationship between $V(\text{IN}+)$ and $V(\text{IN}-)$



¥-Device : Monostable

Qspice : Type - Monostable 1.qsch | Type - Monostable 2.qsch

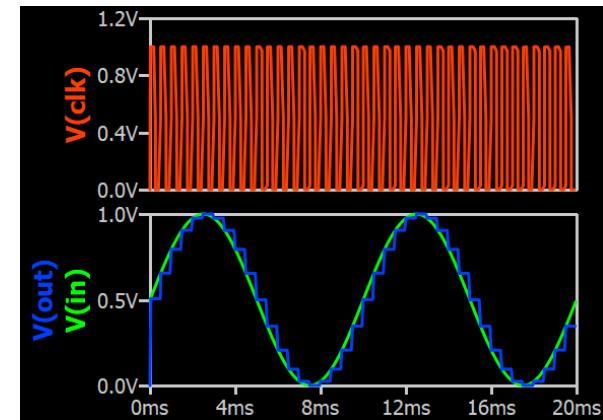
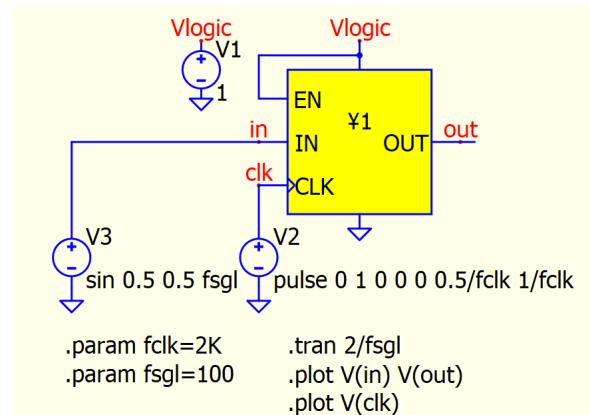
- Monostable
 - This is a Retriggerable Monostable (One-Shot)
 - TON : Monostable on time
 - **Default TON=0s**
 - Need to add TON attribute when Monostable Symbol is drag from Symbol & IP Browser in Qspice
 - Behavioral > flops > MonoStable
 - TON must be specified to use Monostable
 - Rising edge triggers a monostable pulse and reset from last pulse edge after Ton



¥-Device : and Latch (Sample and Hold S&H)

Qspice : Type - Latch (Sample and Hold).qsch

- Latch
 - Edge-sensitive Sample and Hold
 - The latch device is disabled if the EN pin is opened
 - ** Most Qspice default devices can be enabled by leaving the EN pin unconnected
 - Instance parameter IC
 - It can be used to define initial condition before first clock rising edge with OUT voltage equal
 $OUT = V_{ss} + IC$



¥ Truth Table : Φ -DET (Phi-Det) [Symbol : PhaseDetector in analog]

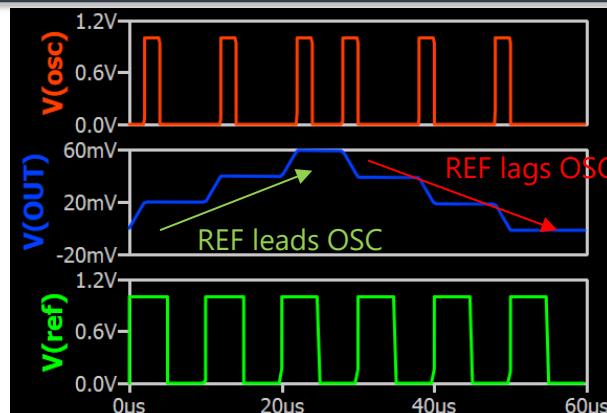
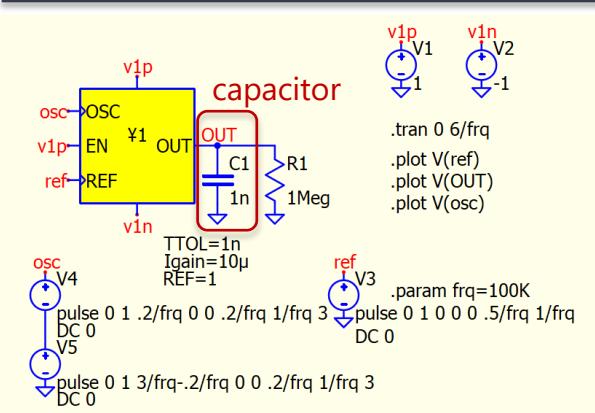
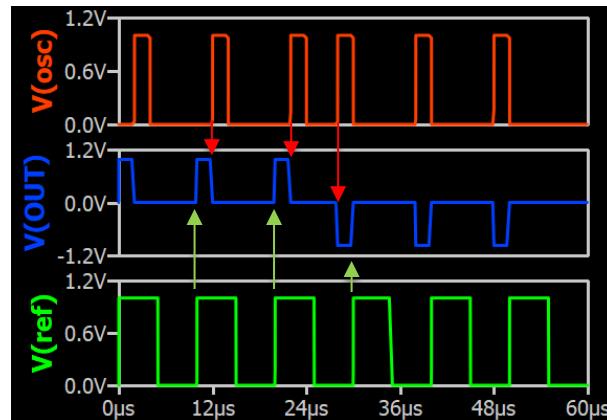
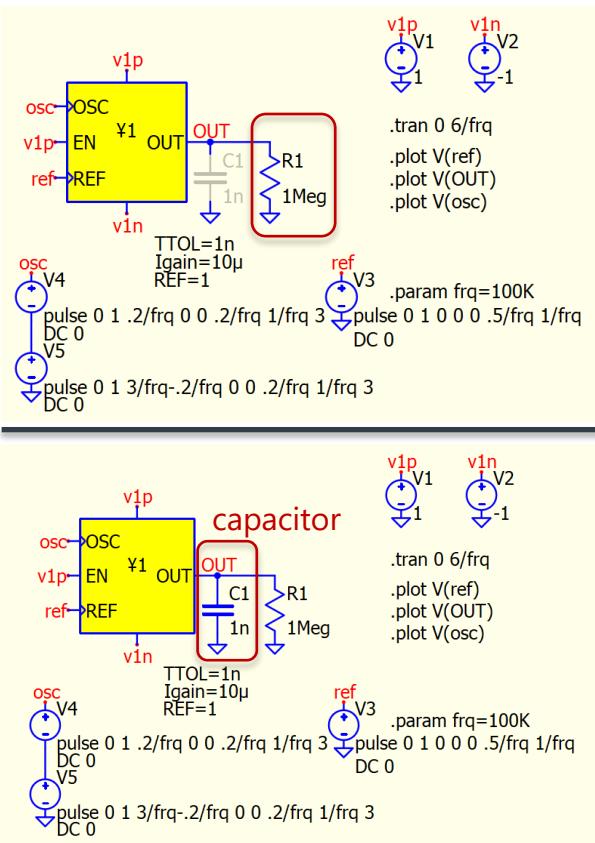
Qspice : Truth Table of Phi-Det 1.qsch | Truth Table of Phi-Det 2.qsch

- Φ -DET (Phi-Det)

- Phase/Frequency Detector (PFD)

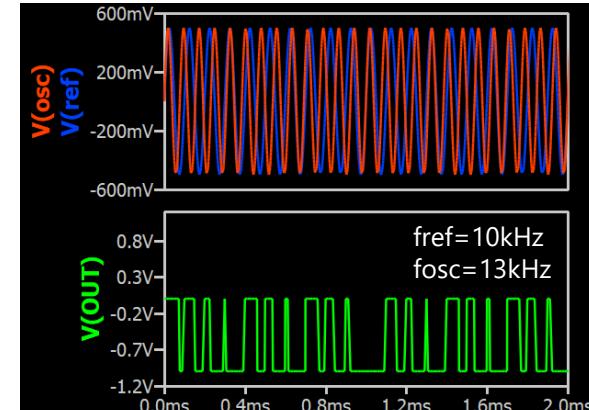
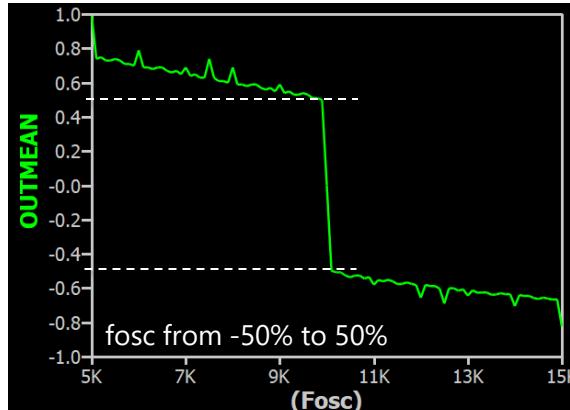
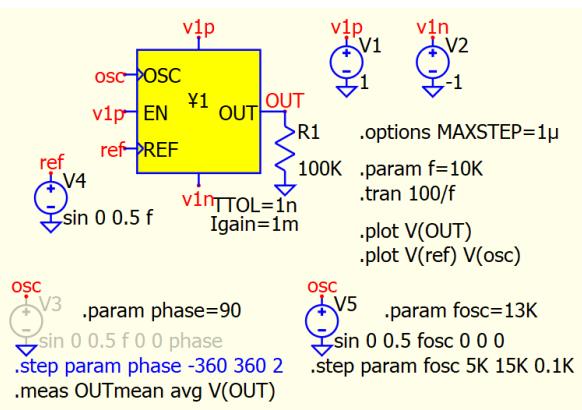
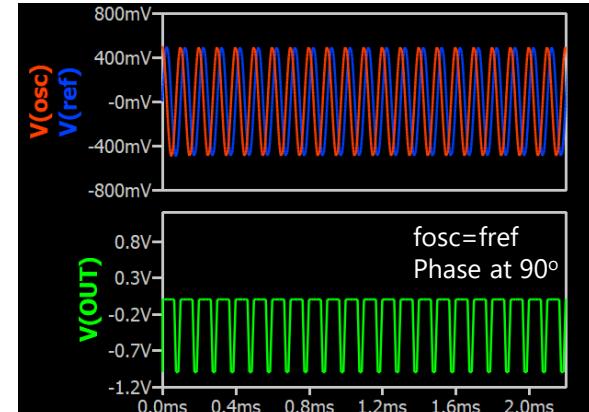
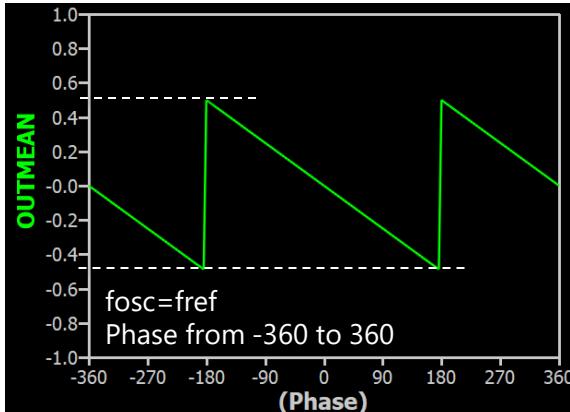
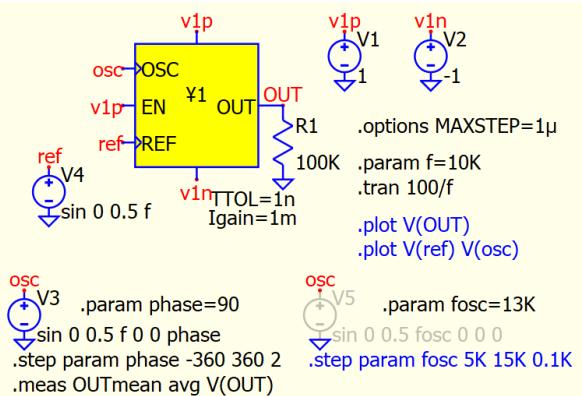
¥-Device Phi-Det		
REF	OSC	OUT
↑	x	Up
x	↑	Down
↑	↑	Unchange

- This device is an up down counter equivalent to Type-II PFD
- Edges on the REF input counts up (UP) and edges on the OSC counts down (DN)
- PFD output as a current source, tri-stage output (source, sink and high impedance)
- However, PFD output generally is connected to a capacitor (+series resistor) to output an integrated results to identify phase lead or lag



¥-Device Φ -DET (Phi-Det) : OSC and REF Relationship with OUT

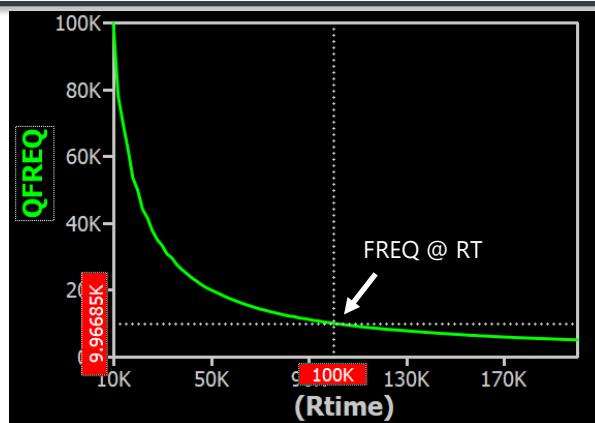
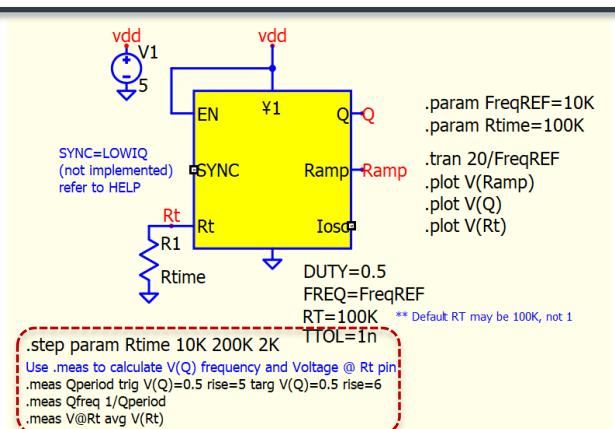
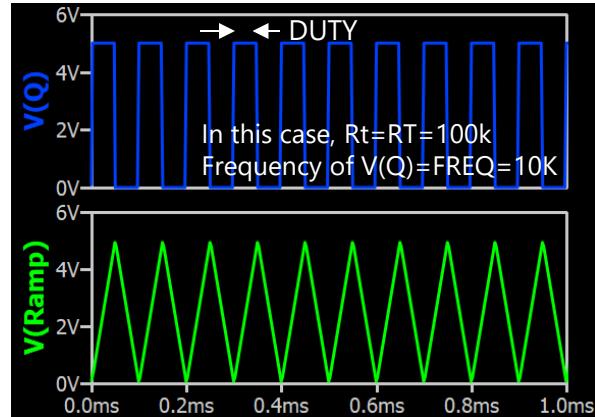
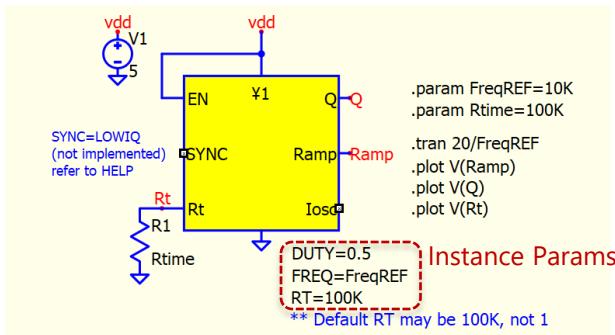
Qspice : Type - Phi-Det (Freq and Phase).qsch



¥-Device EXTOSC : Oscillator programmed with an external resistor

Qspice : Type - Extosc (Rtime .tran).qsch | Type - Extosc (Rtime .step).qsch

- EXTOSC
 - Oscillator programmed with an external resistor
 - Instance Parameters
 - DUTY : Oscillated V(Q) duty [Default DUTY = 0.9]
 - FREQ : Oscillated Frequency @ RT [Default FREQ=100kHz]
 - RT : Timing Resistance (Rtime) value at FREQ [Default RT=100kohms]
 - Symbol : **ExtOsc4** in analog
 - Oscillator frequency is controlled by external resistor with this symbol as no Vres1 and Vres2 pins
- V(Q) frequency vs Rtime
 - This simulation uses .step param Rtime and .meas to plot frequency of V(Q) vs Rtime relationship
 - It can observe that V(Q) frequency has inverse relationship with Rt
 - $\text{Freq of } V(Q) = \frac{RT}{Rtime} \times \text{FREQ}$

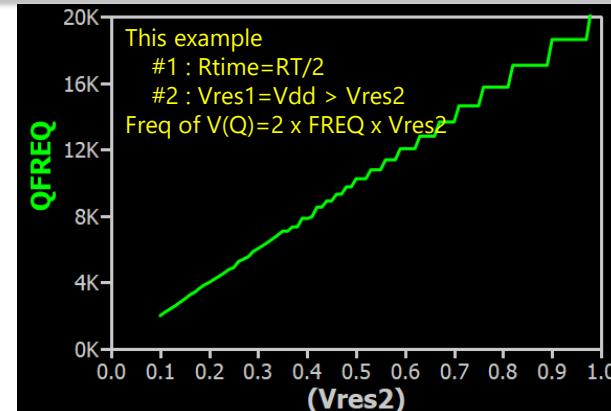
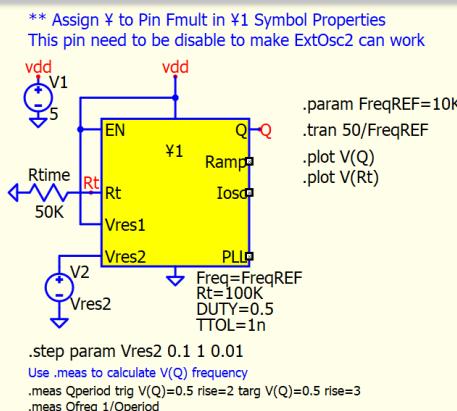
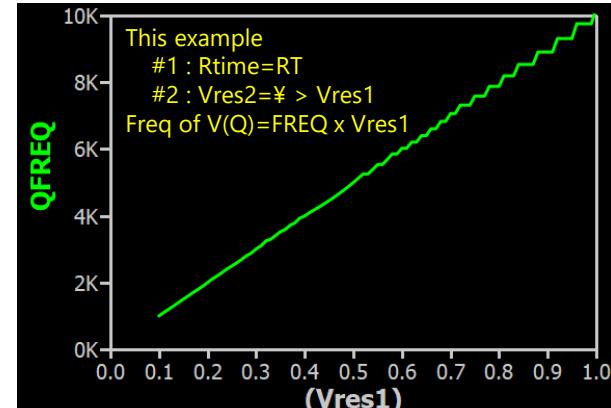
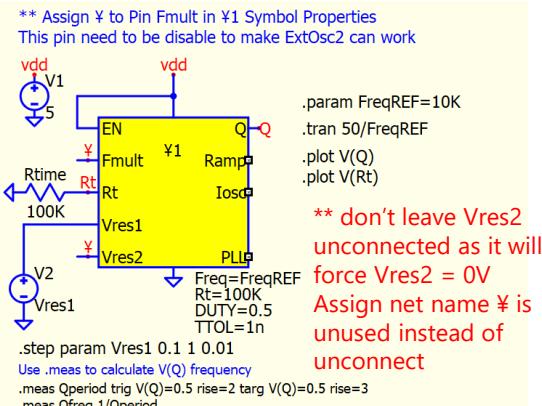


¥-Device EXTOSC : Oscillator programmed with an external resistor

Qspice : Type - Extosc (VRES1.step).qsch | Type - Extosc (VRES2.step).qsch

- V(Q) frequency vs Vres1/2
 - Symbol : **ExtOsc2** in analog
 - Oscillator frequency is controlled by voltage at Vres1 or Vres2 pins
 - This model with Vres1 and Vres2
 - Vres2 is not mentioned in Qspice HELP, but indeed there
 - Oscillator uses minimum value of Vres1 and Vres2 to determine output frequency
 - Instance Parameter VRT (voltage on timing resistor) can affect Frequency
 - Freq of V(Q)

$$= \frac{RT}{Rtime} \times FREQ \times \frac{\min(Vres1, Vres2)}{VRT}$$
 - Typically, the spare one would be used for frequency foldback when the output is shorted to ground
 - ** Fmult should connect to ¥ in using ExtOsc2 symbol
 - Method #1 : Connect to a Net ¥
 - Method #2 : Right Click symbol, System Properties, find Pin Name Fmult and assign Net as ¥, Fmult will be invisible and connect to ¥



¥-Device ULATOR – AM/FM Modulator

Qspice : Type - ULATOR (FREQ).qsch | Type - ULATOR (AMP).qsch

- ULATOR

- Modulator device generates a sine wave (OUT) with an amplitude controlled by the AMP input and frequency controlled by the FREQ input

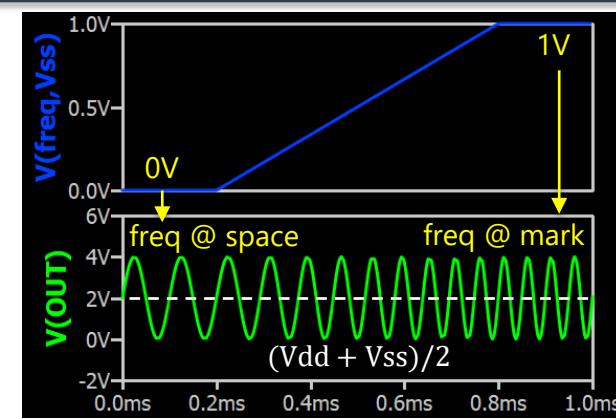
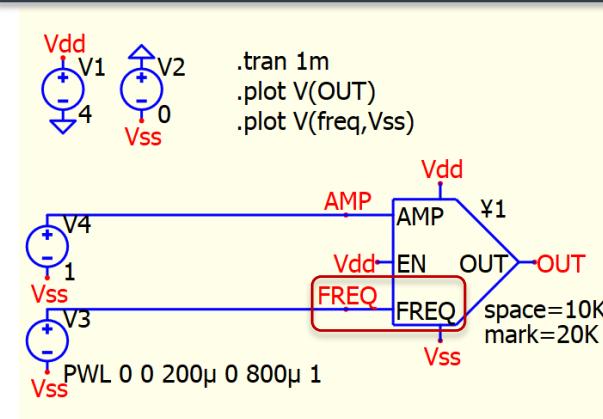
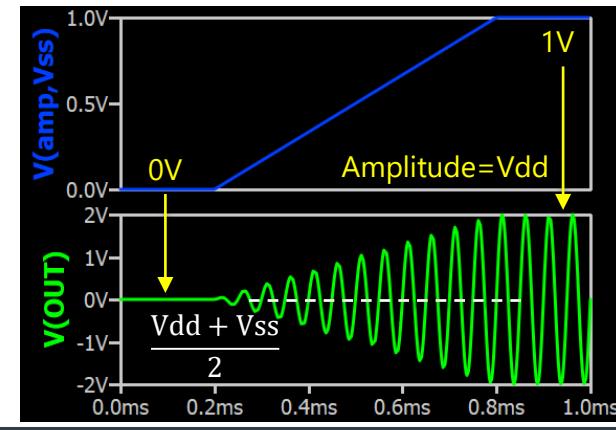
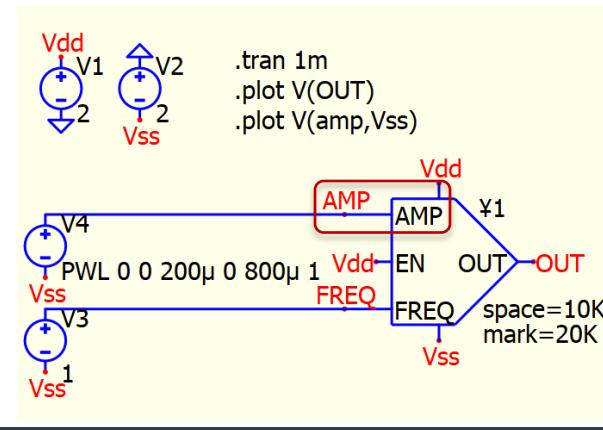
- Output DC at $\frac{Vdd+Vss}{2}$

- Output Amplitude

- 0V when $V(AMP, Vss) = 0V$
- Vdd when $V(AMP, Vss) = 1V$

- Output Frequency

- Frequency = <SPACE> when $V(FREQ, Vss) = 0V$
- Frequency @ <MARK> when $V(FREQ, Vss) = 1V$



¥-Device ULATOR – AM/FM Modulator

Qspice : Type - ULATOR (AMP extra).qsch | Type - ULATOR (FREQ extra).qsch

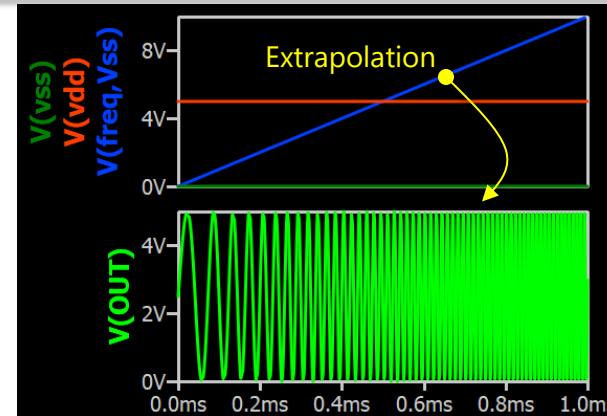
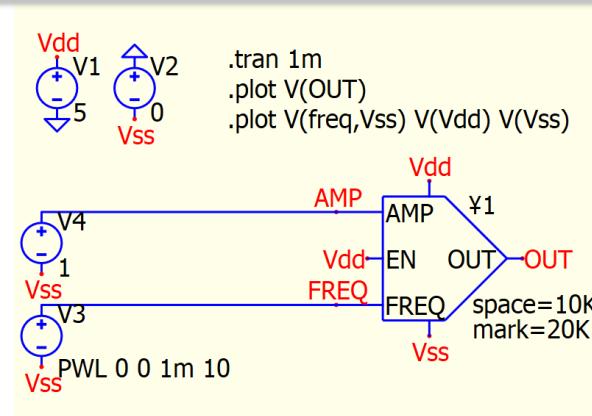
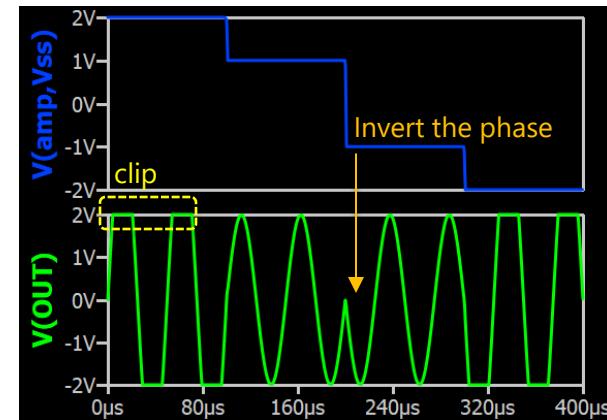
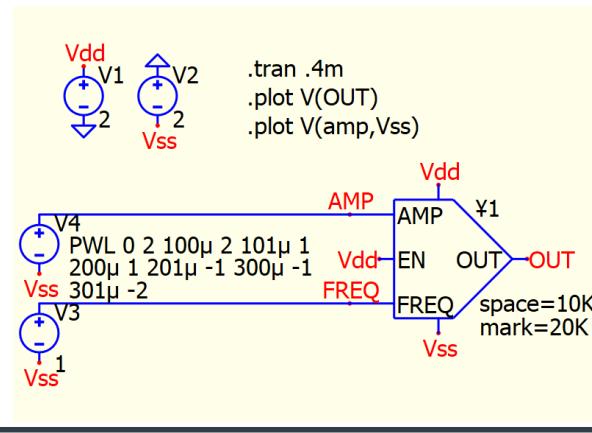
- ULATOR

- Output Amplitude

- If $V(AMP, Vss)$ excess $+/-1V$, clip the sine wave to Vdd/Vss
 - If $V(AMP, Vss)$ is negative, invert the phase

- Output Frequency

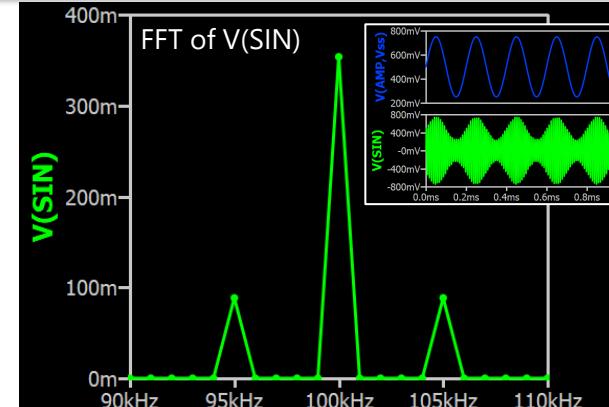
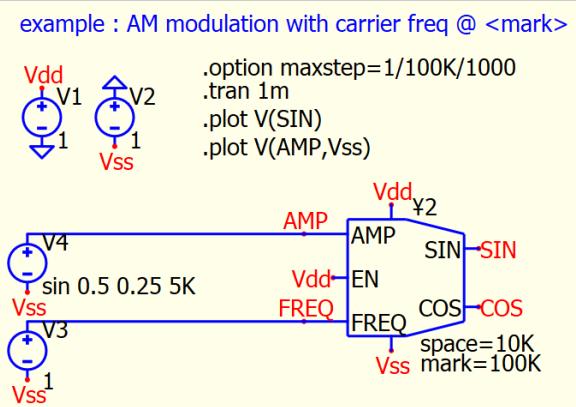
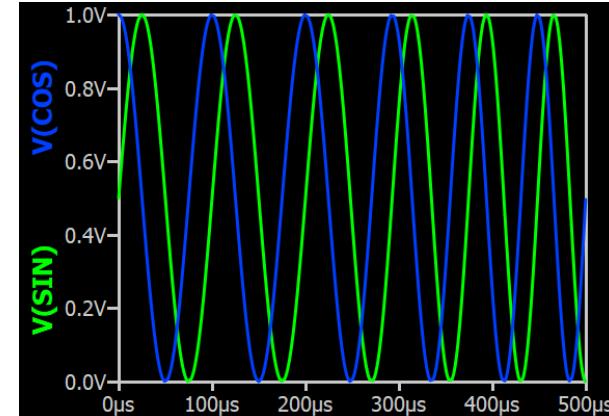
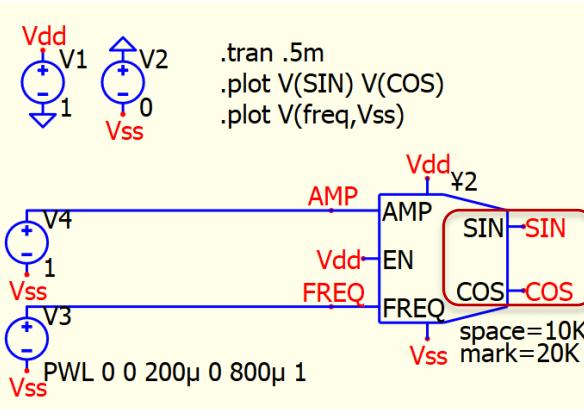
- $OUT_{frequency} = (mark - space) \times |V(FREQ, Vss)| + space$
 - Don't recommend to set $V(FREQ, Vss) < 0V$



¥-Device ULATOR – AM/FM Modulator

Qspice : Type - ULATOR (COS).qsch | Type - ULATOR (AM example).qsch

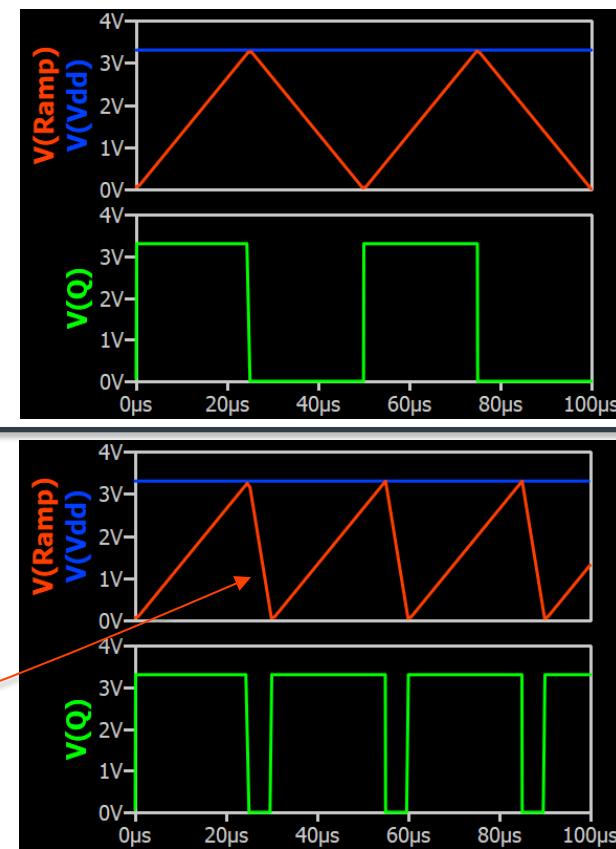
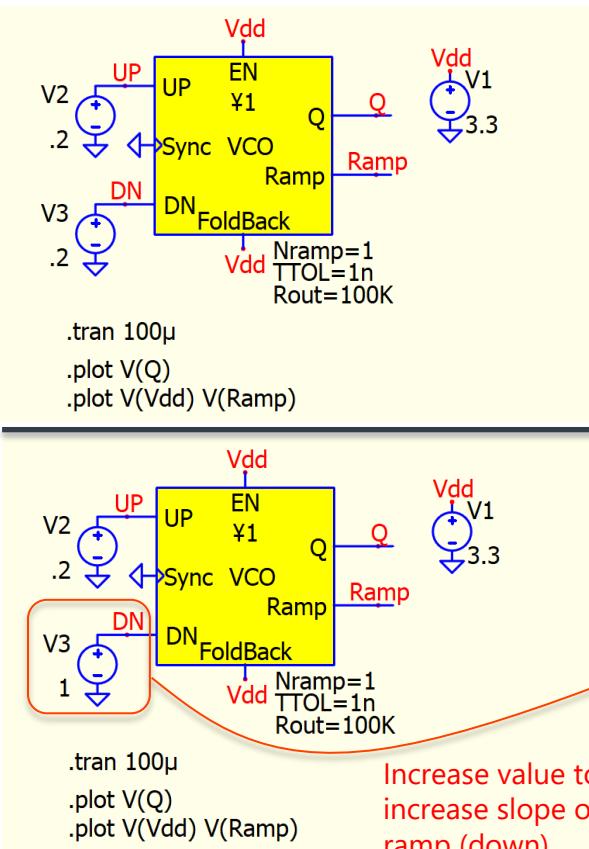
- ULATOR (COS)
 - Two outputs in ULATOR, which are SIN and COS
 - SignalGenerator1.qsym is single output with OUT = SIN
 - SignalGenerator2.qsym is dual outputs with SIN and COS



¥-Device VCO – Voltage-Controlled Oscillator

Qspice : Type - VCO - Basic.qsch

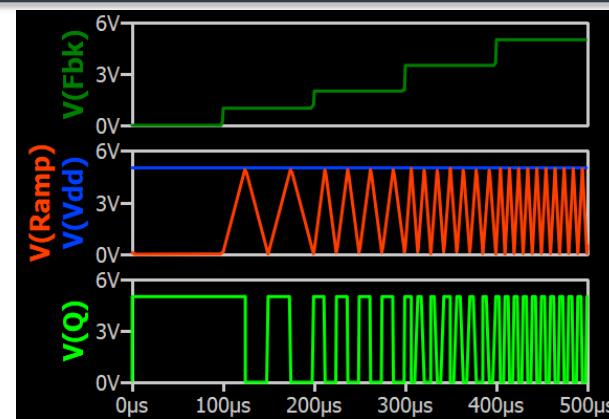
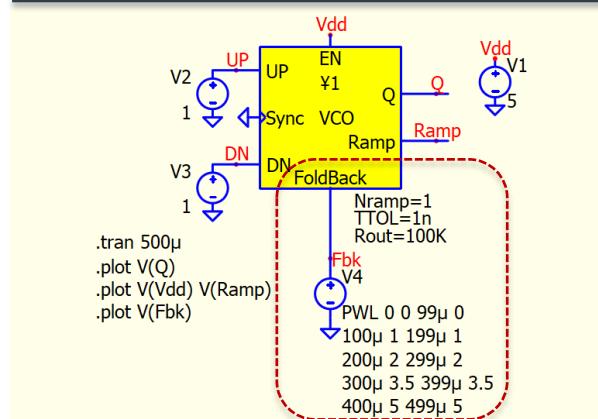
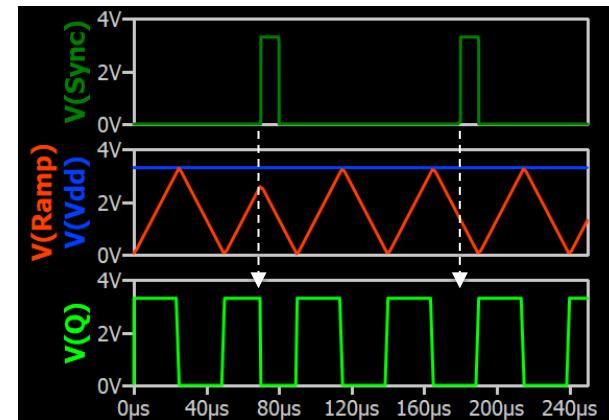
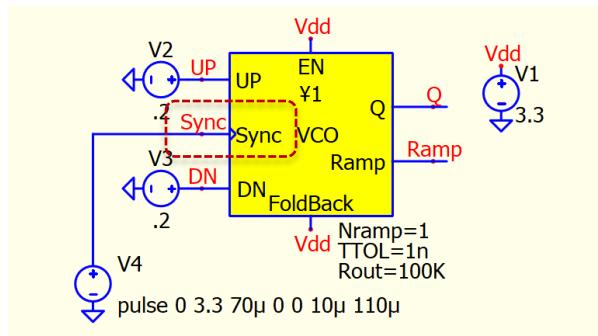
- VCO
 - Voltage-Controlled Oscillator (Undocumented in Help)
 - Basic Operating Example
 - This example is to setup a 1st order ramp signal with a slope controlled by voltage from UP/DN, and to output HIGH in the ramp up and LOW in the ramp down
 - UP/DN (pin) : Control the slope of Ramp Signal Up/Down
 - Fixing UP or DN can generate constant ON-time or OFF-time PWM
 - Ramp (pin) : Ramp signal created for VCO
 - Q (pin) : VCO output



¥-Device VCO – Voltage-Controlled Oscillator

Qspice : Type - VCO - Sync.qsch

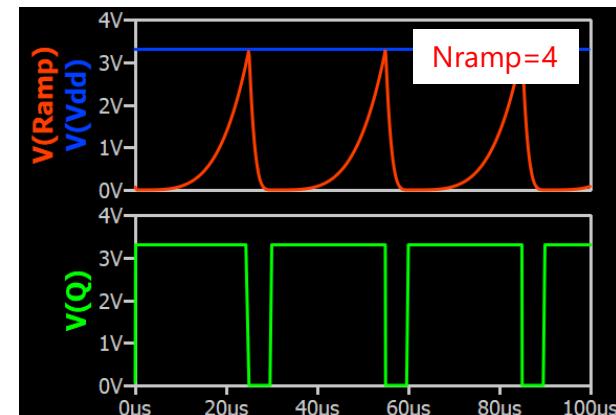
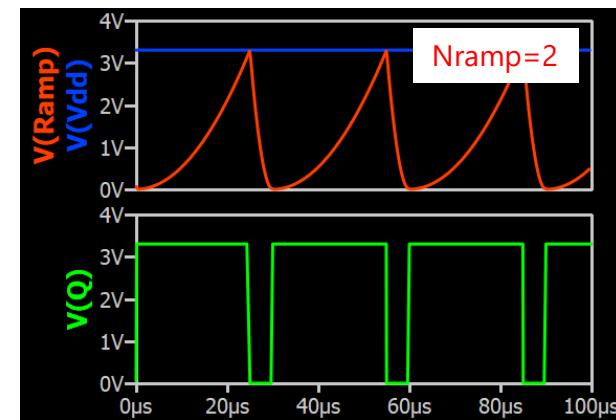
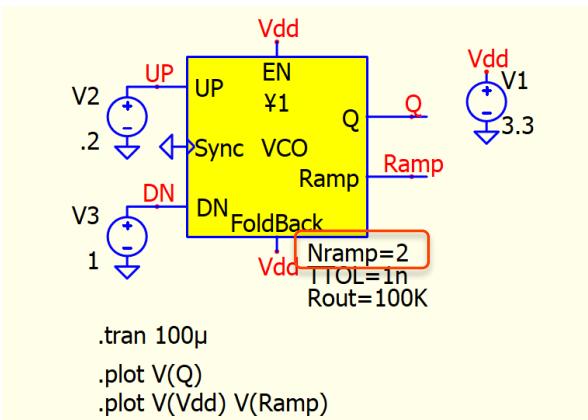
- VCO
 - Sync (pin) : Rising edge flips an up ramp (no effect during the down ramp)
 - FoldBack (pin) : Reduce its value to decrease the frequency generated by the VCO



¥-Device VCO – Voltage-Controlled Oscillator (Instance Param)

Qspice : Type - VCO - Nramp.qsch

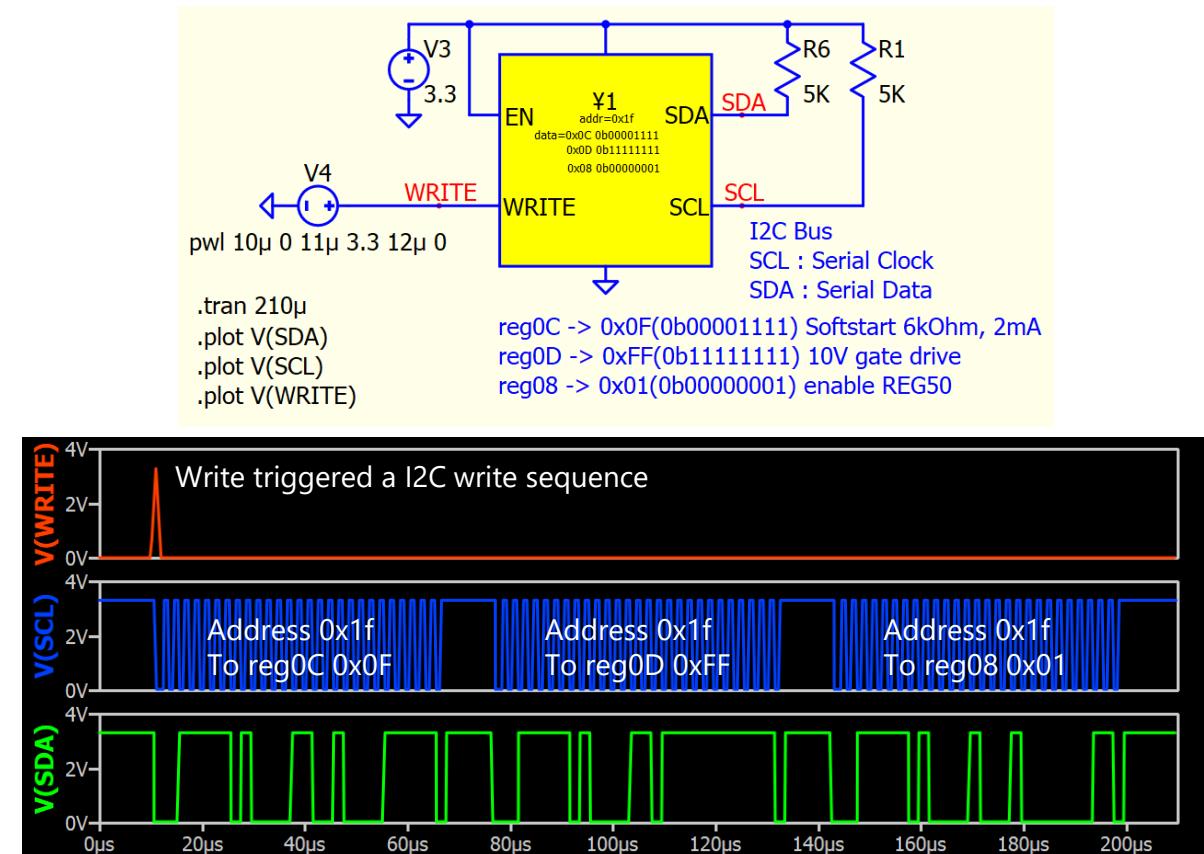
- VCO (Instance param)
 - Nramp : Order of Ramp
 - ** Does not affect the oscillated frequency, only the ramp shape



¥-Device I2C

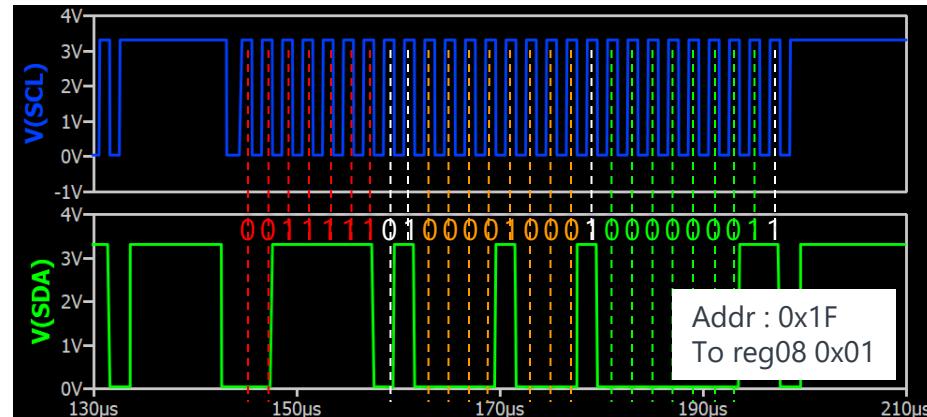
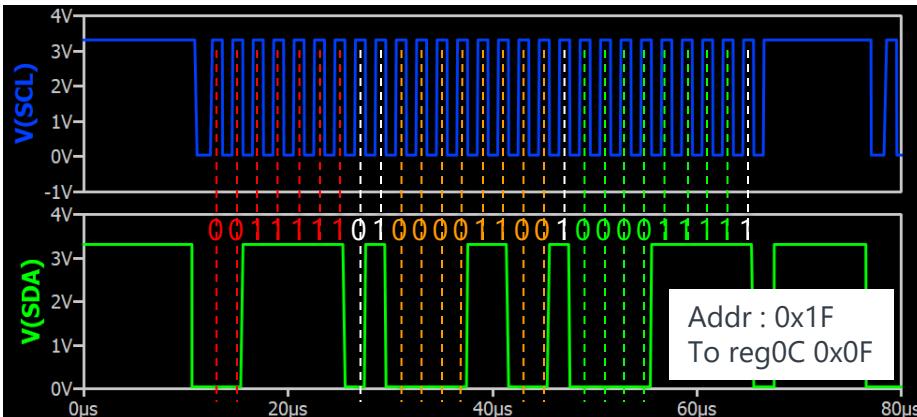
Qspice : Type - I2C.qsch

- I2C
 - Master I²C
 - Not in Qspice HELP
 - Write information into slave registers to instruct its to perform task
 - Rising edge from **WRITE** will trigger I2C ¥-Device to output I2C data which defined in **addr=** and **data=**
 - SCL : Serial Clock
 - SDA : Serial Data
- Syntax
 - ¥nnn VDD VSS SDA SCL EN
 - WRITE ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥ ¥
 - I2C addr=<addr>
 - data=<reg> <data> ...



¥-Device I2C

Qspice : Type - I2C.qsch



- ¥-Device I2C Timing Diagram

- I2C Timing
 - [Address 7-bits][Read/Write][ACK][REG][ACK][DATA][ACT]
 - To Register 0C , Data 0x0F (0b00001111)
 - 001111101000011001000011111
 - In HEX : [0x1f]01[0x0C]1[0b00001111]1
 - To Register 08, Data 0x01 (0b00000001)
 - 001111101000010001000000011
 - In HEX : [0x1f]01[0x08]1[0b00000001]1

¥1
addr=0x1f
data=0x0C 0b00001111
0x0D 0b11111111
0x08 0b00000001

Address
0x1F=0b0011111 (7bits)

¥-Device I2C : Reference of I2C Timing Diagram

- I2C Bus : Write

- Reference
 - Understanding the I2C Bus
 - Application Report SLVA704
 - <https://www.ti.com/lit/an/slva704/slva704.pdf>

Writing to a Slave On The I²C Bus

To write on the I²C bus, the master will send a start condition on the bus with the slave's address, as well as the last bit (the R/W bit) set to 0, which signifies a write. After the slave sends the acknowledge bit, the master will then send the register address of the register it wishes to write to. The slave will acknowledge again, letting the master know it is ready. After this, the master will start sending the register data to the slave, until the master has sent all the data it needs to (sometimes this is only a single byte), and the master will terminate the transmission with a STOP condition.

Figure 8 shows an example of writing a single byte to a slave register.

- Master Controls SDA Line
- Slave Controls SDA Line

Write to One Register in a Device

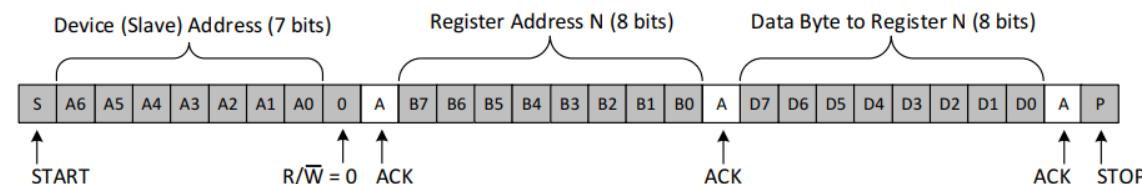


Figure 8. Example I²C Write to Slave Device's Register

Read=1

Write=0

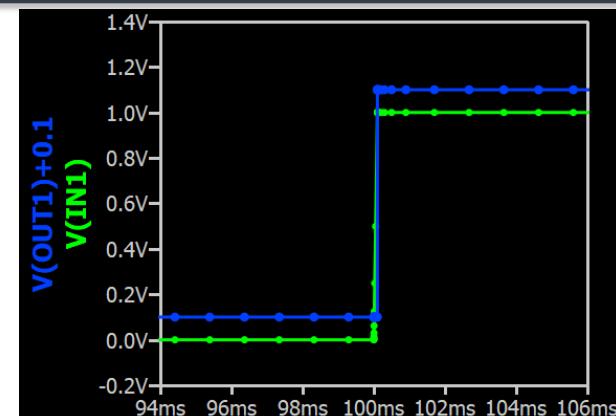
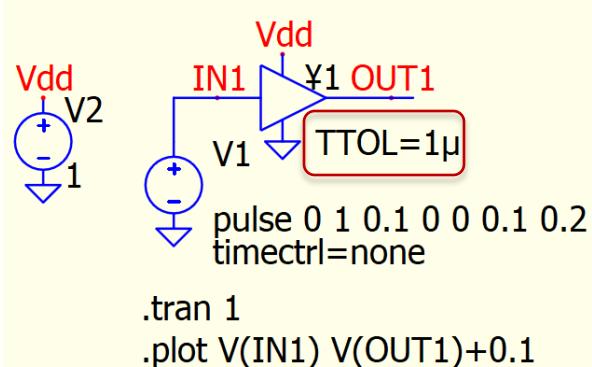
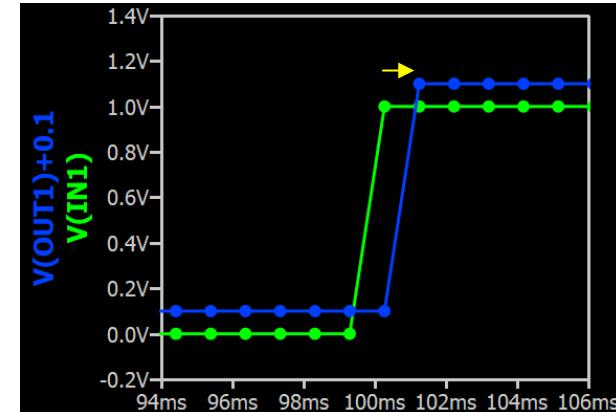
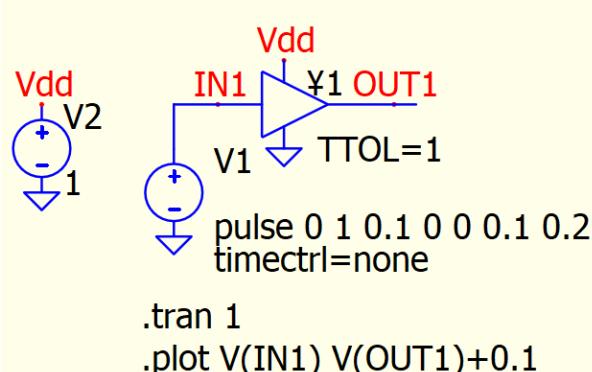
**¥-Device IN and OUT
Timing (nature of DLL
device)**

[This principle apply
to €-, £- and Ø-device]

¥-Device IN and OUT Timing

Qspice : \Input and Output Timing\INOUT-Timing-Demonstration.qsch

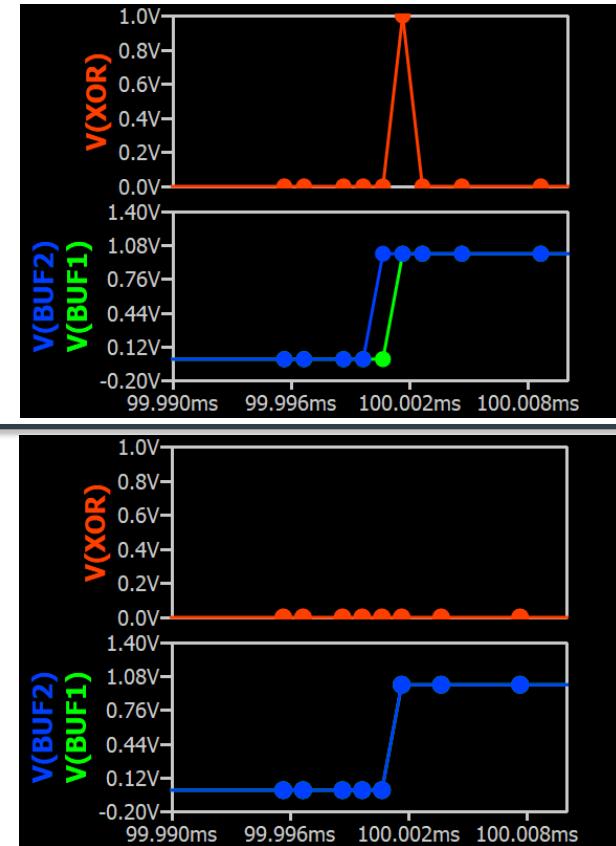
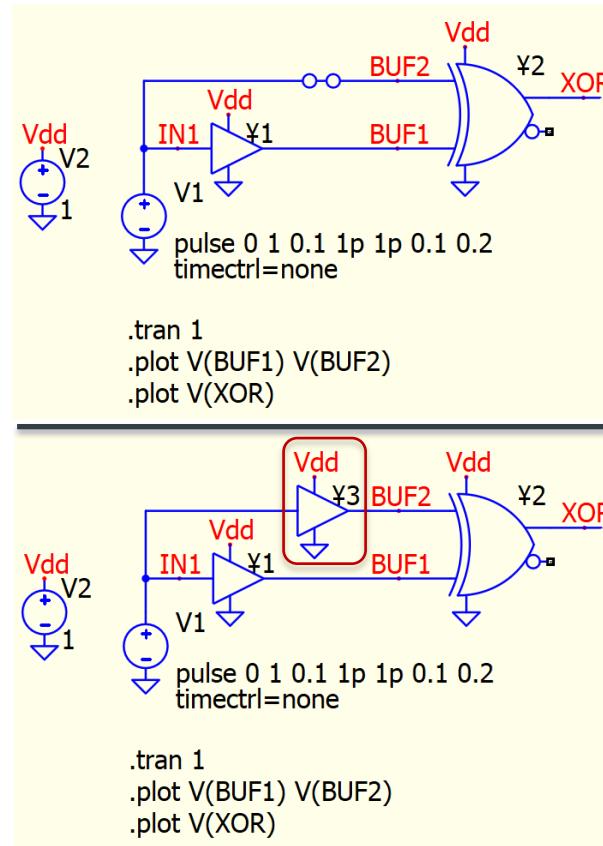
- ¥-Device Timing
 - ¥-device is a built-in DLL device (C++)
 - It's important to be aware that the output of the DLL device always has a one simulation timestep delay from its input
 - Simulation timestep is dynamic. In this example, if reduce TTOL in ¥1 to allow much smaller timestep when output toggle event occurs, the appearance of this one simulation timestep delay can be minimized to approximately the TTOL timestep



¥-Device IN and OUT Timing

Qspice : \Input and Output Timing\INOUT-Timing-XOR.example.qsch

- ¥-Device Timing
 - Impact – If multiple ¥-Devices are connected in series, such a delay can introduce a glitch-like profile
 - For example, this XOR gate receives two identical signals, either 0/0 or 1/1, and expects to always output 0. However, as buffer ¥1 introduces a one simulation timestep delay, the input of the XOR gate is no longer perfectly aligned
 - To resolve this, a buffer ¥1 should be added to compensate for this one simulation timestep delay nature



ϵ -Device

€-Device DAC Instance Params

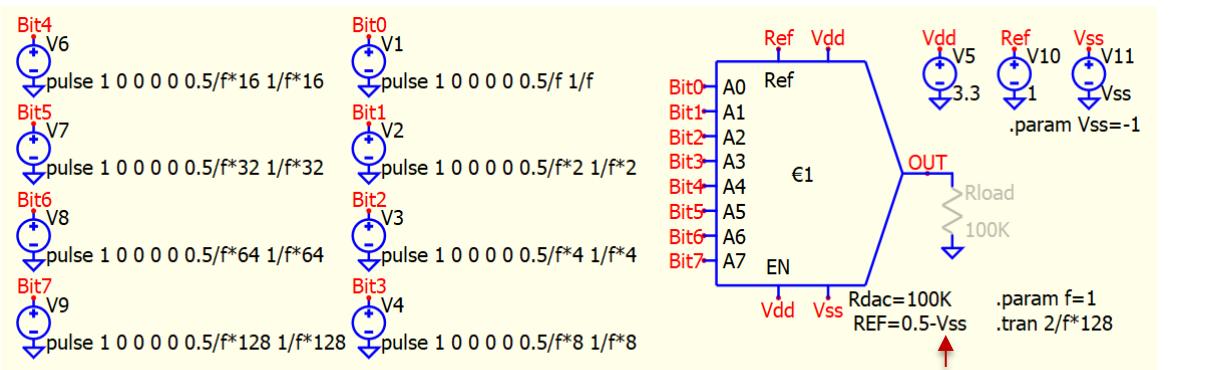
DAC Instance Parameters

Name	Description
BUFFER	If non-zero, then impedance of output buffer
CAPVDD	Capacitance from Vdd to Out
CAPVSS	Capacitance from Vss to Out
CODE	Non-zero maps from -n to n-1 to 0 to 2n-1
EN	Output buffer equivalent input voltage noise density
IC	Initial condition word on DAC
IN	Output buffer equivalent input current noise density
LSB	LSB = 0: output goes from 0 to Ref-lsb. LSB=1: output goes from lsb to Ref
M	Number of parallel devices
RDAC	R of R-2R DAC
REF	Logic reference voltage
TEMP	Instance temperature
TTOL	Temporal tolerance
UVLO	Minimum voltage to operate

€-Device DAC : Build-in Symbol DAC8.qsym

Qspice : DAC - Basic.qsch

- DAC
 - DAC : Digital to Analog Converter
 - Important Pins
 - Ref : the maximum voltage value that the DAC can reach
 - A7 : MSB of input word A
 - A0 : LSB of input word A
 - EN : Enable
 - Instance Params
 - Common instance params can refer to ¥-Device section (e.g. REF)



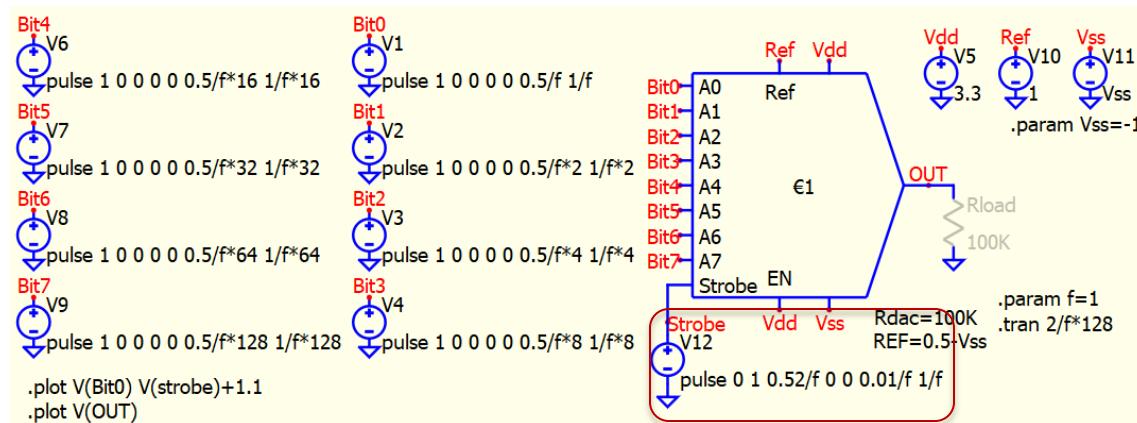
.plot V(Bit7)+7.7 V(Bit6)+6.6 V(Bit5)+5.5 V(Bit4)+4.4 V(Bit3)+3.3 V(Bit2)+2.2 V(Bit1)+1.1 V(Bit0) This REF is Instance Param
 .plot V(OUT) Logic Reference Voltage



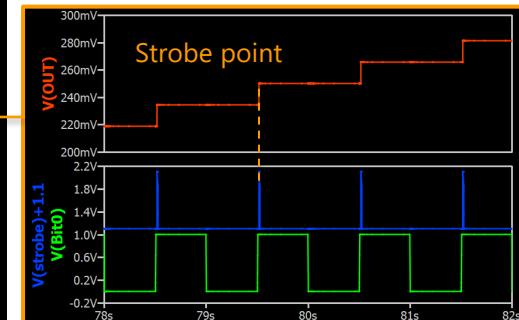
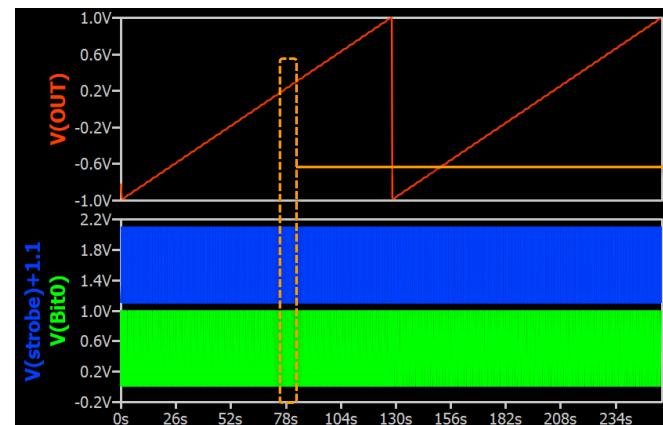
€-Device DAC : Build-in Symbol DAC8strobe.qsym

Qspice : DAC - Strobe.qsch

- Strobe
 - In practice, it named as load DAC strobe (LDAC)
 - This pin transfers all input register data to the DAC registers
 - It resolve jitter in V(out) in previous slide



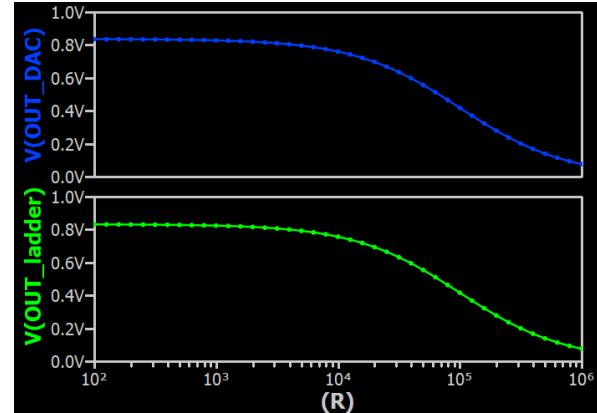
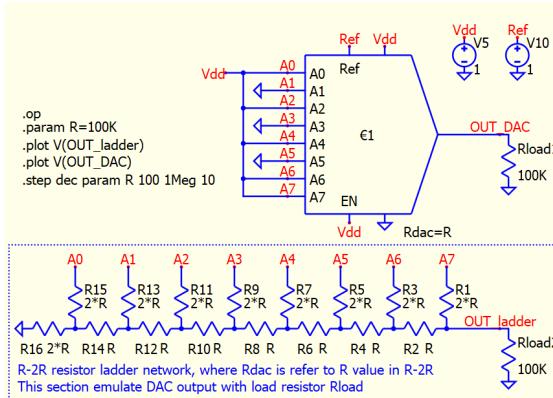
- Symbol
 - Build-in symbol with strobe is
 - DAC8strobe.qsym



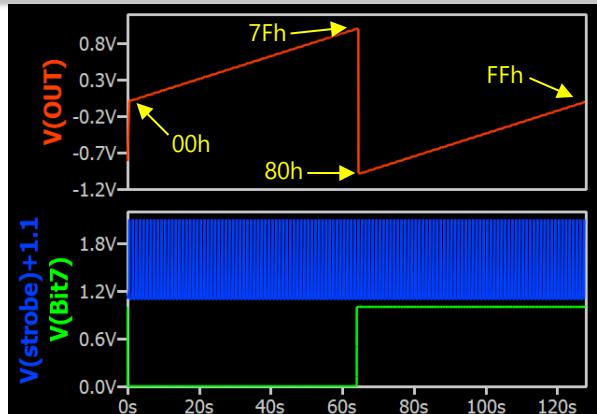
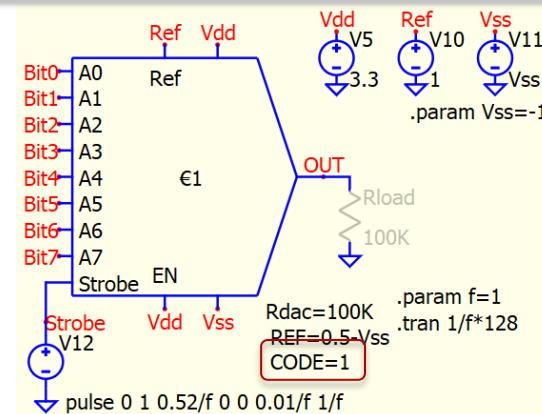
€-Device DAC Instance Params : RDAC, CODE

Qspice : DAC - RDAC.qsch ; DAC - CODE.qsch

- RDAC
 - Rdac : R of R-2R DAC
 - R-2R resistor ladder network is inexpensive solution for digital to analog conversion
 - Reference
 - https://en.wikipedia.org/wiki/Resist_ladder
 - Therefore, to prevent loading effect, a voltage buffer should be used in DAC output



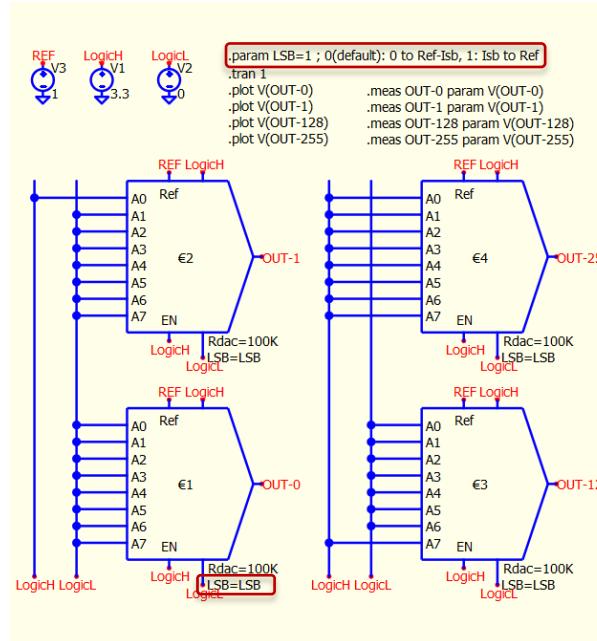
- CODE
 - Code : Non-zero maps from $-n$ to $n-1$ to 0 to $2n-1$
 - **Default CODE=0**
 - CODE=0 : Unsigned
 - CODE=1 : Signed - two's complement



€-Device DAC Instance Params : LSB

Qspice : DAC - LSB.qsch

- **LSB**
 - There are two types of R-2R DACs: one that reaches zero and one that does not (because the last resistor is connected to REF and not GND). The LSB is used to switch between the two types
- **Default : LSB=0**
- Formula
 - $V_{out} = V_{ref} \times \frac{D+LSB}{2^n}$
 - D is digital integer in n bits
 - $I_{sb} = \frac{V_{ref}}{2^n}$
- **LSB=0**
 - output from 0 to REF-lsb
- **LSB=1**
 - output from lsb to REF



Output Window **LSB=0 : 0 to REF-lsb**

```
.meas out-0 param v(out-0) : 0  
.meas out-1 param v(out-1) : 0.00390625  
.meas out-128 param v(out-128) : 0.5  
.meas out-255 param v(out-255) : 0.996094
```

Output Window **LSB=1 : lsb to REF**

```
.meas out-0 param v(out-0) : 0.00390625  
.meas out-1 param v(out-1) : 0.0078125  
.meas out-128 param v(out-128) : 0.503906  
.meas out-255 param v(out-255) : 1
```

R-2R resistor ladder network (digital to analog conversion)

- R-2R ladder network
 - $a_0 \dots a_{n-1}$ input
 - Logic 0 : 0V
 - Logic 1 : V_{ref}
 - Output formula
 - $V_{out} = V_{ref} \times \frac{D}{2^n}$
 - D is digital integer in n bits
 - By Assume $V_{ref} = 1V$ and 8-bits ($n=8$)
 - '00000000' (DEC=0)
 - $D=0 \rightarrow V_{out} = 0$
 - '00000001' (DEC=1)
 - $D=1 \rightarrow V_{out} = \frac{1}{2^8} = 0.003906$
 - '10000000' (DEC=128)
 - $D=128 \rightarrow V_{out} = \frac{128}{2^8} = 0.5$
 - '11111111' (DEC=255)
 - $D=255 \rightarrow V_{out} = \frac{255}{2^8} = 0.996$

R-2R resistor ladder network (digital to analog conversion) [\[edit\]](#)

Voltage Mode [\[edit\]](#)

A *voltage mode* R-2R resistor ladder network is shown in Figure 1. It produces an analog output voltage V_{out} from a digital *integer* D composed of n bits: a_{n-1} (*most significant bit*, MSB) through bit a_0 (*least significant bit*, LSB), which are driven from digital logic gates. The bit inputs are switched between 0 volts (*logic 0*) and V_{ref} volts (*logic 1*). The R-2R network causes these digital bits to be weighted

in their contribution to the output voltage V_{out} . Depending on which bits are set to 1 and which to 0, the output voltage V_{out} will have a corresponding *stepped value* between 0 and $\frac{2^n - 1}{2^n} V_{ref}$, where the minimal voltage step ΔV_{out} (corresponding to bit a_0) is $\frac{1}{2^n} V_{ref}$. The actual value of V_{ref} (and the voltage of logic 0) will depend on the type of technology used to generate the digital signals, and are ideally exact voltages.^[7]

For n bits, this R-2R DAC will convert the digital integer D into the output voltage V_{out} :

$$V_{out} = V_{ref} \times \frac{D}{2^n}$$

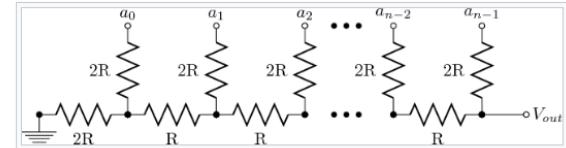


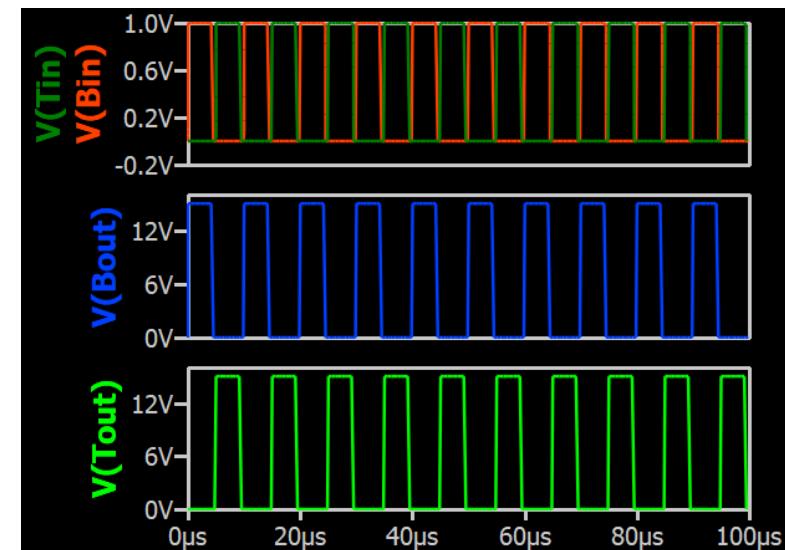
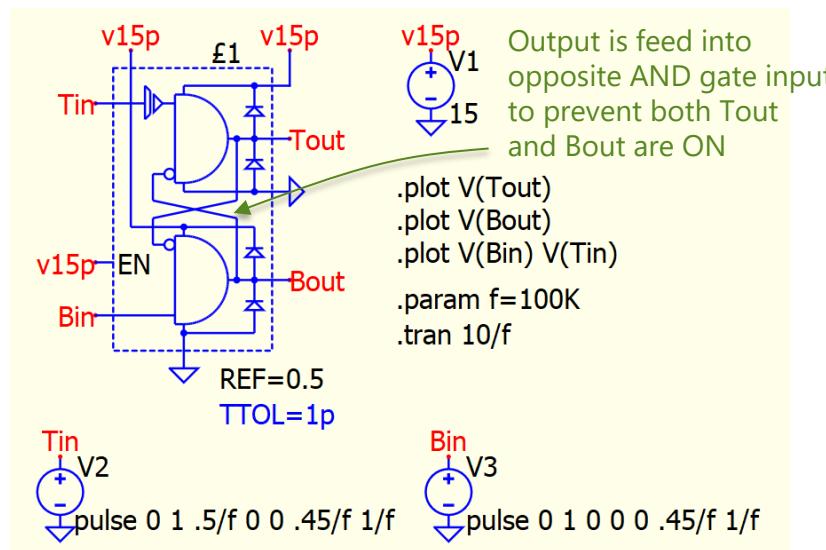
Figure 1: n-bit *voltage mode* R-2R resistor ladder DAC. Bit significance increases from left to right.

£-Device

f-Device : Dual Gate Driver

Qspice : Dual Gate Driver.qsch

- f-Device Dual Gate Driver
 - Syntax: fnnn Vdd Vss BOOST TOPGATE SW BOTGATE TCTRL BCTRL EN ¥ ¥ ¥ ... GATEDRIVER [INSTANCE PARAMETERS]
 - This behavioral device expects Top Input and Bottom Input has deadtime in between



£ [Dual Gate Driver] Instance Params

Gate Driver Instance Parameters

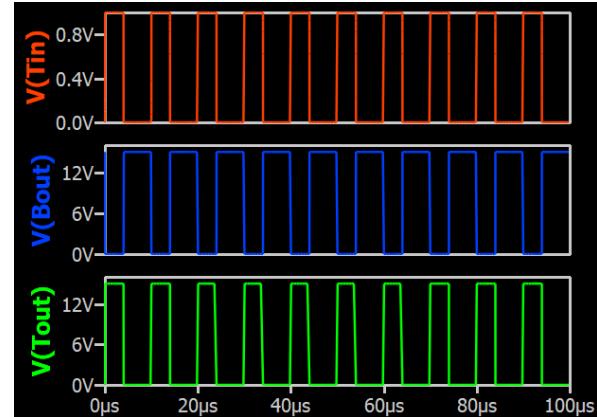
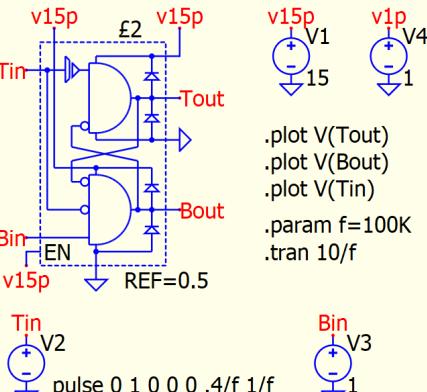
Name	Description	Units	Default
BOOST	Add logic such that the top FET is off if the bottom is controlled to be on		(not set)
BUCK	Add logic such that the bottom FET is off if the top is controlled to be on		(not set)
CAPVDD	Capacitance from a logic output to Vdd(or BOOST)	F	0.
CAPVSS	Capacitance from a logic output to Vss(or SW)	F	0.
M	Number of parallel devices		1.
REF	Logic reference for enable input	V	$(Vdd + Vss) \div 2$
ROFF1	Resistance used to pull bottom FET to VSS	Ω	RON1
RON1	Resistance used to pull bottom FET to VDD	Ω	1.
ROFF2	Resistance used to pull top FET to SW	Ω	ROFF
RON2	Resistance used to pull top FET to BOOST	Ω	RON1
RPASSIVE	Resistance used to pull both FETs' gates low when the driver is not enabled	Ω	1Meg
TEMP	Instance temperature	$^{\circ}C$	27.
TTOL	Temporal tolerance	s	
UVLO	Minimum Vdd-Vss voltage to operate	V	2.

£ [Dual Gate Driver] Instance Params - Buck and Boost

Qspice : Dual Gate Driver (Buck).qsch; Dual Gate Driver (Boost).qsch

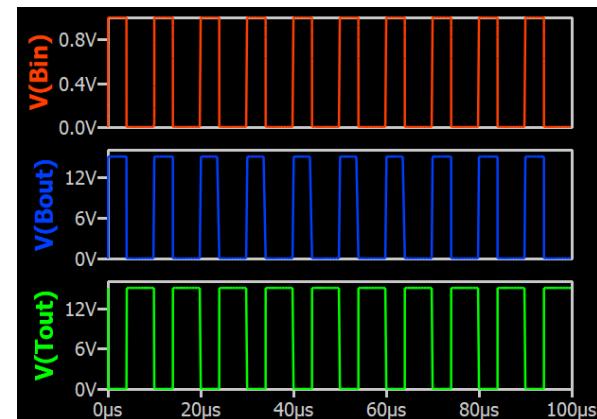
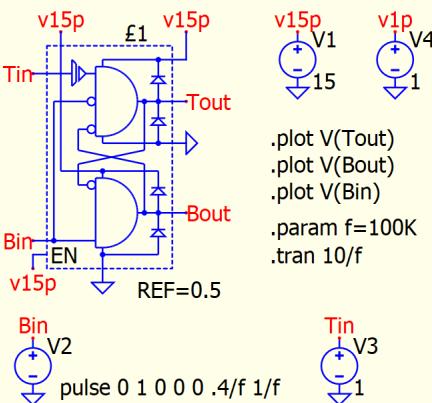
• BUCK

- Add logic such that the bottom FET is off if the top is controlled to on
- **Tin** is inverted to drive Bottom AND gate
 - Bin connects to LOW
- Switch of buck converter is generally at High Side



• BOOST

- Add logic such that the top FET is off if the bottom is controlled to be on
- **Bin** is inverted to drive Top AND gate
 - Tin connects to HIGH
- Switch of boost converter is generally at Low Side

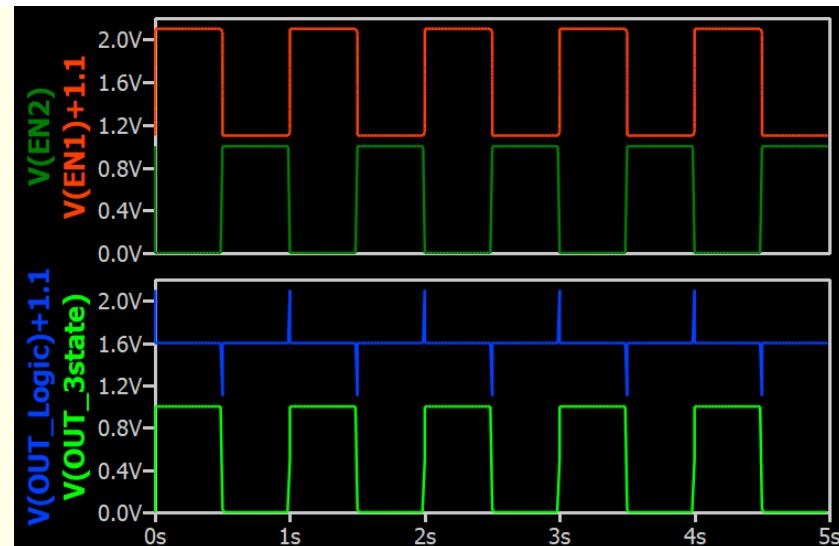
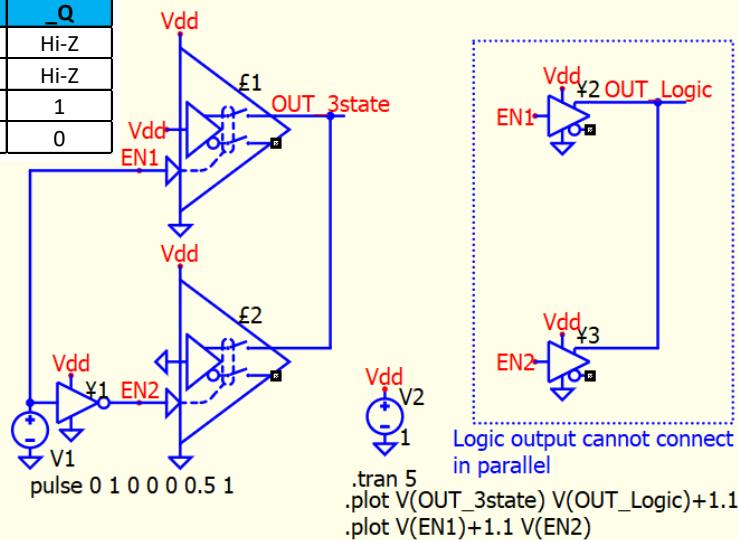


f-Device : Tri-state Buffer

- Tri-state Buffer

- Syntax: fnnn Vdd Vss Q Q̄ IN EN ¥ ¥ ¥ ... 3STATE [INSTANCE PARAMETERS]
- A tri-state output allows multiple circuits to share the same output line(s)
- Output into High-Z when disable

f-Device Tri-state Buffer			
EN	IN	Q	_Q
0	0	Hi-Z	Hi-Z
0	1	Hi-Z	Hi-Z
1	0	0	1
1	1	1	0



Ø-Device (.DLL)

Part 1: Overview

Fundamental about Ø-Device (DLL-Block)

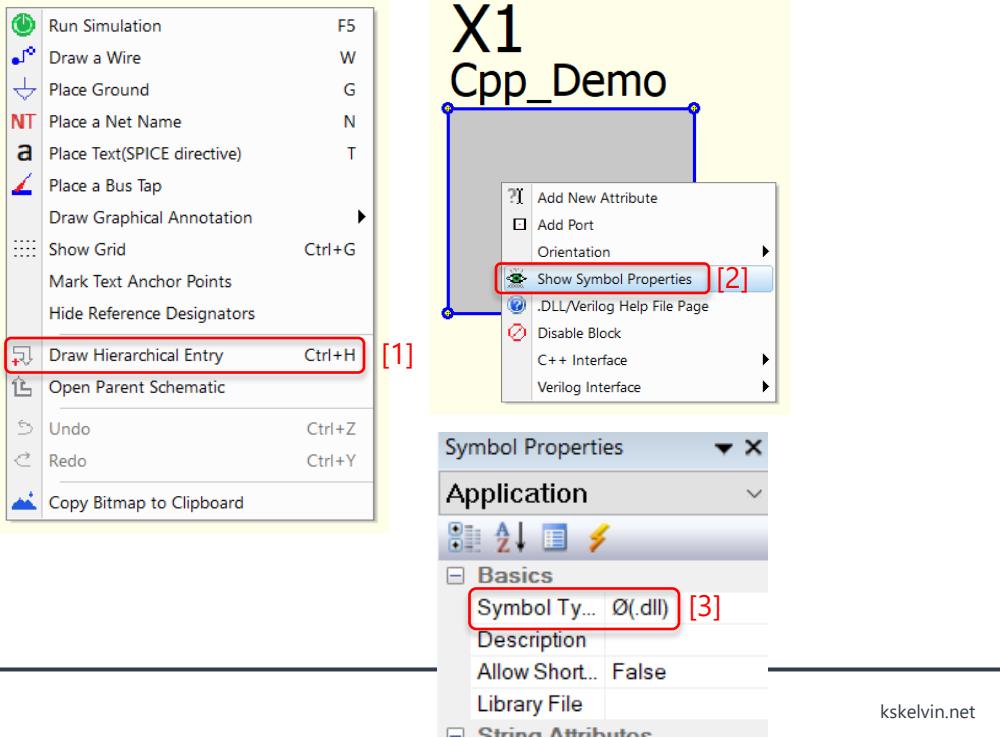
- Ø-Device
 - The Ø-Device is used for implementing complex algorithms with C++ or Verilog languages in QSPICE circuit simulation
 - It links a module defined in a .DLL (dynamic-link library), which is a file containing code and data. While .EXE files are standalone, .DLLs are shared libraries that offer reusable code for multiple applications
 - .DLLs can be written in C++ or Verilog (converted by Verilator)
 - The Ø-Device serves as a symbol for defining input/output ports and parameter data types
 - Input ports accept a node voltage (for Boolean data type, true/false is determined based on logic threshold REF)
 - Output port is emulated by a voltage source with Rout and Cout
 - Output port provides analog voltage, and the source voltage outputs a value according to the port's data type
 - For example, Boolean outputs VHIGH (default is 1V) and 0V, Unsigned Short output voltages ranging from 0 to 65536 (2^{16}) discrete levels, Float64 (double) outputs floating point voltages from -1.7e308 to +1.7e308
 - **.DLL always output a result delayed by one simulation timestep**
 - Unlike QSPICE's native devices (e.g. B-source), which computes equations with results in same simulation timestep
 - QSPICE's ¥-, €- and £- devices essentially function as .DLL devices and therefore also output results delayed by one simulation timestep

Workflow to Create C++ .DLL (Ø-Device)

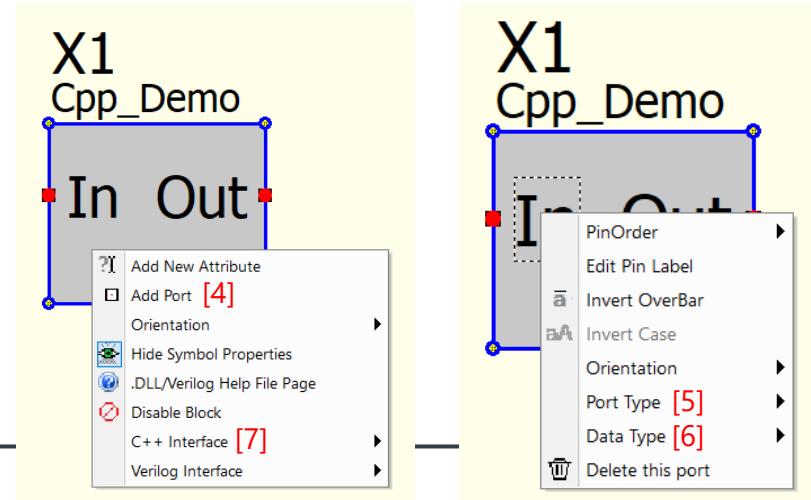
Qspice : \01 Overview\Workflow_Cpp\

Workflow to Create .DLL C++ (Ø-device)

- [1] In schematic, Right click > Draw Hierarchical Entry
- [2] Right click hierarchical block and Show Symbol Properties
- [3] In Symbol Type, change to Ø(.dll)



- [4] Right click hierarchical block > Add Port
 - Double click the port to assign name and change text location
- [5] Right click each port, select corresponding Port Type
 - For In, Port Type : Input
 - For Out, Port Type : Output
- [6] Right click each port, select corresponding Data Type
 - For In, Data Type : float (64 bit double)
 - For Out, Data Type : float (64 bit double)
- [7] Right click hierarchical block > C++ Interface
 - Create C++ Template > OK



Workflow to Create C++ .DLL (\emptyset -Device)

Qspice : \01 Overview\Workflow_Cpp\

Qorvo QSPICE™ - cpp_demo.cpp

File Edit View Help

// Automatically generated C++ file on Mon Sep 2 08:32:53 2024

// To build with Digital Mars C++ Compiler:

// dmc -mn -WD cpp_demo.cpp kernel32.lib

```
union uData
{
    bool b;
    char c;
    unsigned char uc;
    short s;
    unsigned short us;
    int i;
    unsigned int ui;
    float f;
    double d;
    long long int i64;
    unsigned long long int ui64;
    char *str;
    unsigned char *bytes;
};

// int DllMain() must exist and return 1 for a process to load the .DLL
// See https://docs.microsoft.com/en-us/windows/win32/dlls/dllmain for more information.
int __stdcall DllMain(void *module, unsigned int reason, void *reserved) { return 1; }

// #undef pin names lest they collide with names in any header file(s) you might include.
#define In
#define Out
#include <cmath> [8]
```

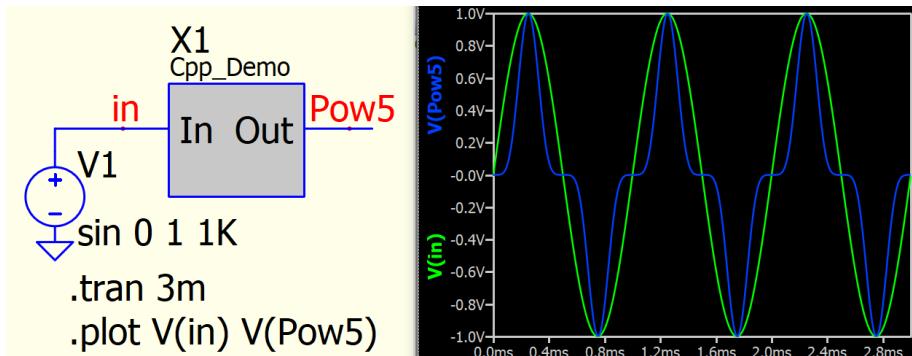
extern "C" __declspec(dllexport) void cpp_demo(void **opaque, double t, union uData *data)
{
 double In = data[0].d; // input
 double &Out = data[1].d; // output

```
// Implement module evaluation code here:
    Out = pow(In,5); [9]
```

"cpp_demo.dll" created successfully [10] : return created successfully

Compile DLL & Run F5
Compile DLL F6 [10]
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A

- [8] add `#include<cmath>` if math function is needed
- [9] Implement the function of the device below the comment
`// Implement module evaluation code here`
- [10] Right click > Compile DLL
 - If success, a successful statement in status bar
- [11] Run SPICE simulation that call the C++ device

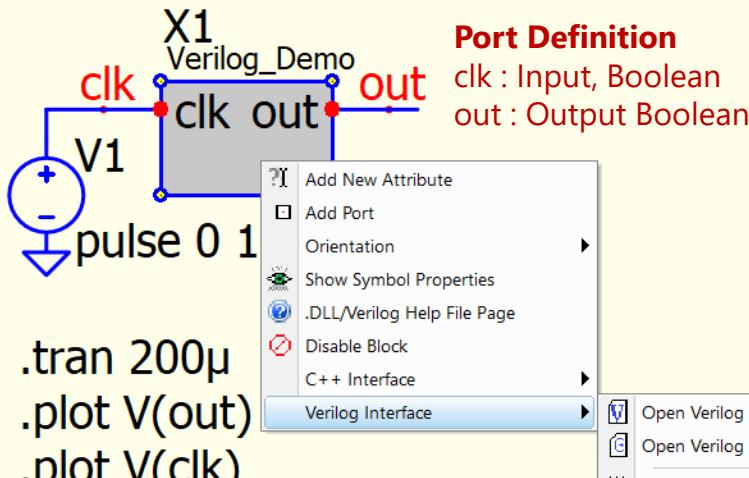


Workflow to Create Verilog .DLL (\emptyset -Device)

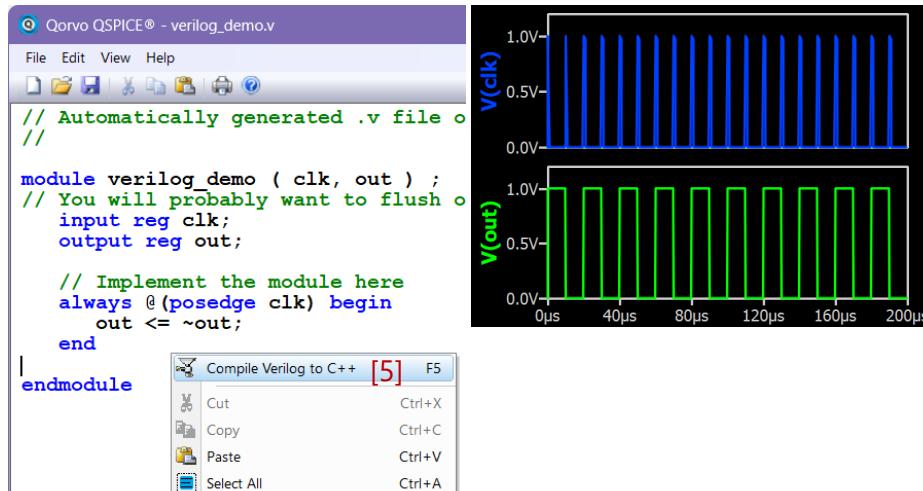
Qspice : \01 Overview\Workflow_Verilog\

Workflow to Create .DLL Verilog (\emptyset -device)

- [1] Follow workflow similar to C++ creation step [1]-[6]
- [2] Right click > Verilog Interface > Create C++ .DLL Main Template
 - This template is to define port in C++ .DLL to interface with Verilog. If add, remove, or change port/data type, need to re-create this template
- [3] Right click > Verilog Interface > Create Verilog Template



- [4] Add the code in Verilog module, this demo is to toggle output at every clock positive edge
- [5] Right click > Compile Verilog to C++
- [6] Go back to schematic and Run simulation
- [7] Output simulation results in waveform viewer



Ø-Device Instance Parameters

Ø-Device Instance Parameters

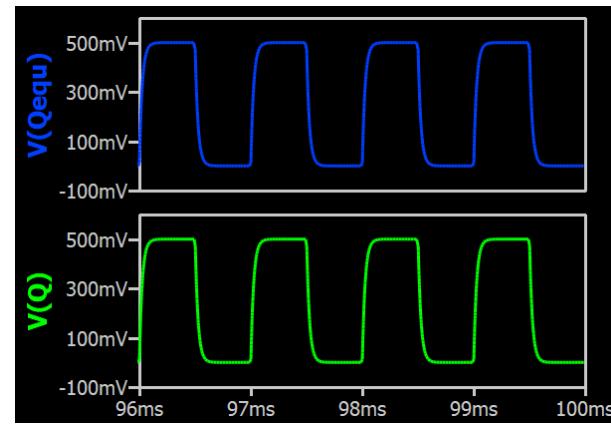
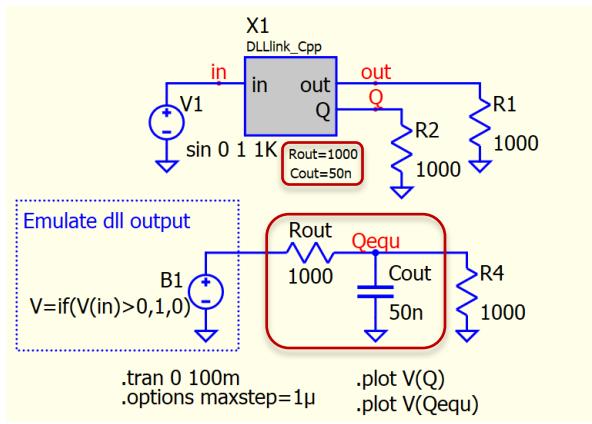
Name	Description	Units	Default
COUT	Output capacitance	C	0.
MAXTIMESTEP	Maximum time between evaluations	s	Infinite
REF	Logic threshold for inputs declared as boolean	V	.5
ROUT	Output impedance	Ω	1000.
VHIGH	Logic high level for outputs declared as boolean	V	1.
VLSB	ADC and DAC incremental voltage for an LSB change for multi-bit binary data types	V	1.

Ø-Device Instance Params : Rout / Cout and Vhigh

Qspice : \01 Overview\InstanceParams\DLLlink - Rout Cout.qsch | DLLlink - VHIGH.qsch

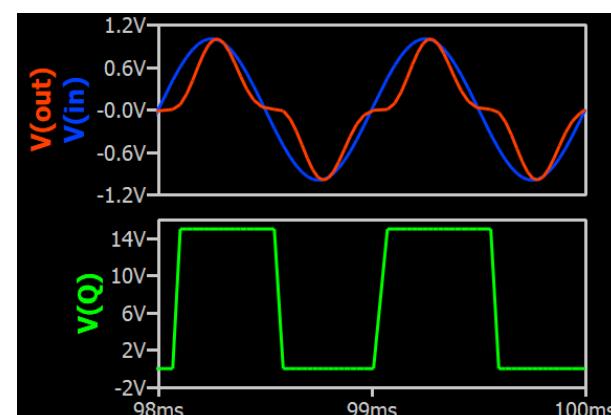
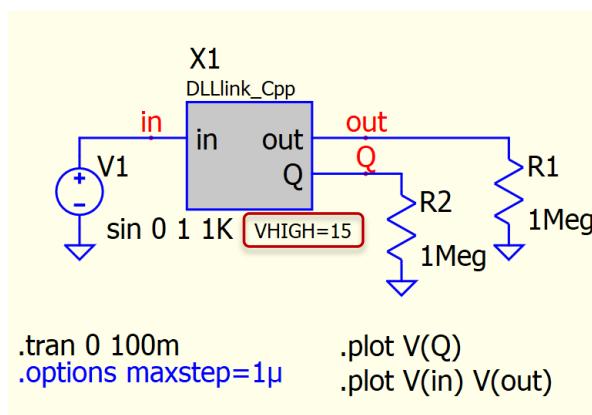
• ROUT and COUT

- Rout : Output Impedance
- Cout : Output Capacitance
- **Default ROUT=1000Ω**
- **Default COUT=0F**
- Example
 - In this example, it shown circuit arrangement of Rout and Cout in .dll device output



• VHIGH

- Vhigh : Logic high level for outputs declared as Boolean
- **Default VHIGH=1V**

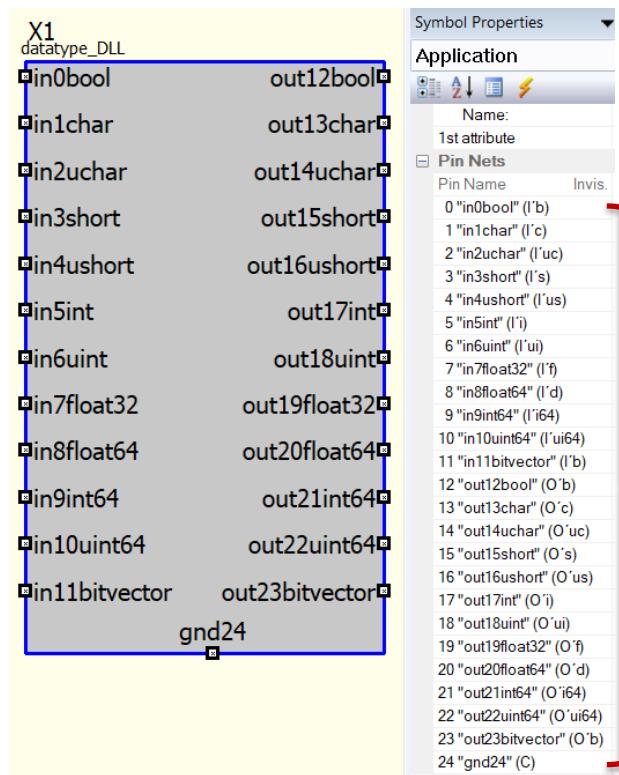


Ø-Device : Port Type and Data Type

Qspice : \01 Overview\DataType\datatype.qsch

- Port Type
 - Input
 - Output
 - DLL's ground

- Data Type
 - Boolean
 - Char
 - Unsigned char
 - Short
 - Unsigned short
 - Integer
 - Unsigned integer
 - Shortfloat (32 bit)
 - Float (64 bit)
 - Integer 64
 - Unsigned integer 64
 - Bit Vector



Symbol Properties pane:

- Application
- Pin Nets
- Port Type
- Data Type

Port Type dropdown:

- boolean
- char
- unsigned char
- short
- unsigned short
- integer
- unsigned integer
- shortfloat(32 bit)
- float(64 bit double)
- integer 64
- unsigned integer 64
- Bit Vector

Data Type dropdown:

- boolean
- char
- unsigned char
- short
- unsigned short
- integer
- unsigned integer
- shortfloat(32 bit)
- float(64 bit double)
- integer 64
- unsigned integer 64
- Bit Vector

Symbol properties code:

```
extern "C" __declspec(dllexport) void datatype_dll(struct sDATATYPE_I
```

Port Type	Data Type
in0bool	= data[0].b ; // input
in1char	= data[1].c ; // input
in2uchar	= data[2].uc ; // input
in3short	= data[3].s ; // input
in4ushort	= data[4].us ; // input
in5int	= data[5].i ; // input
in6uint	= data[6].ui ; // input
in7float32	= data[7].f ; // input
in8float64	= data[8].d ; // input
in9int64	= data[9].i64 ; // input
in10uint64	= data[10].ui64 ; // input
in11bitvector	= data[11].b ; // input
out12bool	= data[12].b ; // output
out13char	= data[13].c ; // output
out14 uchar	= data[14].uc ; // output
out15short	= data[15].s ; // output
out16 ushort	= data[16].us ; // output
out17int	= data[17].i ; // output
out18 uint	= data[18].ui ; // output
out19float32	= data[19].f ; // output
out20float64	= data[20].d ; // output
out21int64	= data[21].i64 ; // output
out22 uint64	= data[22].ui64 ; // output
out23bitvector	= data[23].b ; // output

The pin order, port and data types are now displayed in the pin line of the Symbol Properties pane

Ø-Device : Port Type and Data Type

Qspice : \01 Overview\DataType\datatype-range.qsch

DataType	C++ Declaration	Bits	union uData	Range in C++	Verilog Declaration
Boolean	bool	1	.b	true/false 0/1	reg
Char	char	8	.c	-128 to 127	byte
Unsigned char	unsigned char	8	.uc	0 to 255 (2^8)	byte unsigned
Short	short	16	.s	-32768 to 32767	shortint
Unsigned short	unsigned short	16	.us	0 to 65536 (2^{16})	shortint unsigned
Integer	int	32	.i	-2147483648 to 2147483647	integer
Unsigned integer	unsigned int	32	.ui	0 to 4294967295 (2^{32})	integer unsigned
Shortfloat (32 bits)	float	32	.f	-3.4e-38 to 3.4e38	shortreal
Float (64 bits)	double	64	.d	-1.7e-308 to 1.7e308	real
Integer 64	long long int	64	.i64	-9.22e18 to 9.22e18	longint
Unsigned integer 64	unsigned long long int	64	.ui64	0 to 1.84e19 (2^{64})	longint unsigned
Bit Vector [0:0]	bool	1	.b	true/false 0/1	reg [0:0]
Bit Vector [n:0], n=1 to 7	unsigned char	8	.uc	0 to 255 (2^8)	reg [n:0]
Bit Vector [n:0], n=8 to 15	unsigned short	16	.us	0 to 65536 (2^{16})	reg [n:0]
Bit Vector [n:0], n=16 to 31	unsigned int	32	.ui	0 to 4294967295 (2^{32})	reg [n:0]
Bit Vector [n:0], n=32 to 63	unsigned long long int	64	.ui64	0 to 1.84e19 (2^{64})	reg [n:0]

** Bit Vector cannot be more than 64 bits, or ERROR will be generated in C++ template in data type

Verilog in DLL-Block : Verilog .DLL C++ Main

- Verilog in DLL-Block
 - Verilog is initially converted to C++ using Verilator
 - Verilog C++ .DLL Main template is used to convert SPICE data according to typed ports and parameters
 - Thus, the code essentially runs with the C++ code (converted by Verilator) and with a .dll C++ main to interface with SPICE node data
 - SPICE node of the DLL block essentially acts as a voltage source in SPICE, following the range and nature of the declared data type

Verilog Source

```
module datatype_dll ( in0bool, in1char, i
// You will probably want to flush out th
  input reg in0bool;
  input byte in1char;
  input byte unsigned in2uchar;
  input shortint in3short;
  input shortint unsigned in4ushort;
  input integer in5int;
  input integer unsigned in6uint;
  input shortreal in7float32;
  input real in8float64;
  input longint in9int64;
  input longint unsigned in10uint64;
  input reg [0:0] in11bitvector;
  output reg out12bool;
  output byte out13char;
  output byte unsigned out14uchar;
  output shortint out15short;
  output shortint unsigned out16ushort;
  output integer out17int;
  output integer unsigned out18uint;
  output shortreal out19float32;
  output real out20float64;
  output longint out21int64;
  output longint unsigned out22uint64;
  output reg [0:0] out23bitvector;

// Implement the module here
```

Verilog .DLL C++ Main

```
extern "C" __declspec(dllexport) void datatype_dll()
{
  if(!*instance)
    *instance = new Vdatatype_dll;
  (*instance)->in0bool = data[0].b;
  (*instance)->in1char = data[1].c;
  (*instance)->in2uchar = data[2].uc;
  (*instance)->in3short = data[3].s;
  (*instance)->in4ushort = data[4].us;
  (*instance)->in5int = data[5].i;
  (*instance)->in6uint = data[6].ui;
  (*instance)->in7float32 = data[7].f;
  (*instance)->in8float64 = data[8].d;
  (*instance)->in9int64 = data[9].i64;
  (*instance)->in10uint64 = data[10].ui64;
  (*instance)->in11bitvector = data[11].b;

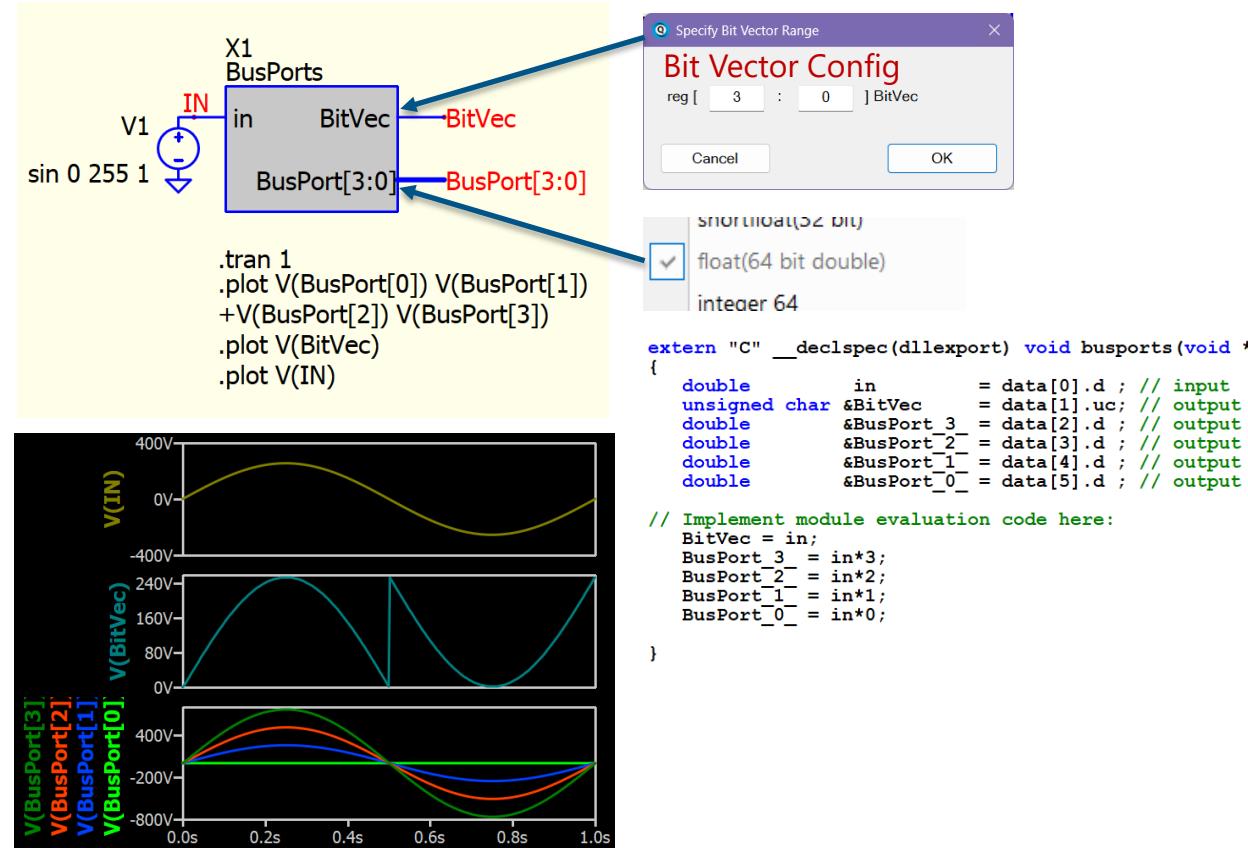
  (*instance)->eval();

  data[12].b = (*instance)->out12bool;
  data[13].c = (*instance)->out13char;
  data[14].uc = (*instance)->out14uchar;
  data[15].s = (*instance)->out15short;
  data[16].us = (*instance)->out16ushort;
  data[17].i = (*instance)->out17int;
  data[18].ui = (*instance)->out18uint;
  data[19].f = (*instance)->out19float32;
  data[20].d = (*instance)->out20float64;
  data[21].i64 = (*instance)->out21int64;
  data[22].ui64 = (*instance)->out22uint64;
  data[23].b = (*instance)->out23bitvector;
}
```

Bit Vector and Bus Ports in C++

Qspice : \01 Overview\Bus Format Ports (Cpp)\

- Bit Vector and Bus Ports
 - This is a C++ example
 - Bit Vector is configured as reg [3:0] ; however, in C++ template, in default Qspice convert it to 8 bit unsigned char
 - From the simulation results, it can confirmed that BitVec is range from 0 to 255 (2^8) instead of 15 (2^4)
 - It is because C++ doesn't have reg, which only available in Verilog. C++ only declare to its closest 1, 8, 16, 32, 64 bits unsigned integer
 - BusPort is configured with bus syntax [3:0]
 - Data Type can be defined as whatever we want, and in C++ template, 4 output ports (double) are declared
 - But in schematic, only one port is required to be declared
 - Bus ports require bus net to get each bus line data

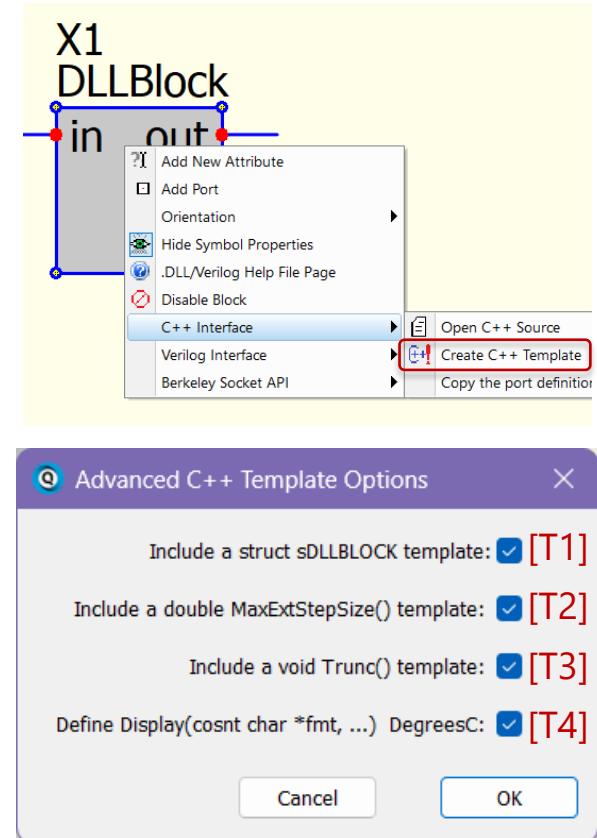


Ø-Device (.DLL)

Part 2: C++ Template

Ø-Device – Create C++ Template

- Ø-Device – Create C++ Template
 - To write in C++, right click Ø-Device, C++ Interface > Create C++ Template
- Advanced C++ Template Options
 - [T1] Struct template
 - Declare the structure (member variables) which can be used between functions
 - member variable, e.g. inst->[variable name]
 - [T2] MaxExtStepSize()
 - Control maximum time step during simulation
 - Special usage : Return -1e308 to abort simulation and goes to the next step (if any)
 - [T3] Trunc()
 - Temporary reduce timestep to Temporal TOLERance (TTOL) based on condition comparison from hypothetical evaluation
 - [T4] Display()
 - Qspice built-in functions and variables pointers



Ø-Device (Template)

Qspice : \02 Template\DLLex (C++ template)\

```
// Automatically generated C++ file on Wed Aug 27 00:23:11 2025
```

```
// To build with Digital Mars C++ Compiler:  
// dmc -mn -WD dllib_x1.cpp kernel32.lib
```

```
#include <malloc.h>
```

```
extern "C" __declspec(dllexport) void (*Display)(const char *format, ...) = 0; // works like printf()  
extern "C" __declspec(dllexport) void (*EXIT)(const char *format, ...) = 0; // print message like printf()  
extern "C" __declspec(dllexport) const double *DegreesC = 0; // pointer to current circuit  
extern "C" __declspec(dllexport) const int *StepNumber = 0; // pointer to current step num  
extern "C" __declspec(dllexport) const int *NumberSteps = 0; // pointer to estimated number  
extern "C" __declspec(dllexport) const char *const *InstanceName = 0; // pointer to address of instance name  
extern "C" __declspec(dllexport) const char *const *QUnit = 0; // pointer to QUnit  
extern "C" __declspec(dllexport) const bool *ForKeeps = 0; // pointer to whether being evaluated  
extern "C" __declspec(dllexport) const bool *HoldICs = 0; // pointer to whether instance  
extern "C" __declspec(dllexport) const void *GUI_HWND = 0; // pointer to Window handle of GUI  
extern "C" __declspec(dllexport) const double *CRTTime = 0; // pointer to CRT time  
extern "C" __declspec(dllexport) const double *CRTDelta = 0; // pointer to CRT delta  
extern "C" __declspec(dllexport) const int *IntegrationOrder = 0; // integration order  
extern "C" __declspec(dllexport) const char *InstallDirectory = 0; // install directory  
extern "C" __declspec(dllexport) double (*Eng atof)(const char **string) = 0;  
extern "C" __declspec(dllexport) const char *(*BinaryFormat)(unsigned int data) = 0; // Binary format  
extern "C" __declspec(dllexport) const char *(*EngFormat)(double x, const char *units, int numDgts) = 0; // Engineering format  
extern "C" __declspec(dllexport) int (*DFFT)(struct sComplex *u, bool inv, unsigned int N, double scale) = 0; // Discrete Fourier Transform  
extern "C" __declspec(dllexport) void (*bzero)(void *ptr, unsigned int count) = 0;
```

```
union uData
```

```
{  
    bool b;  
    char c;  
    unsigned char uc;  
    short s;  
    unsigned short us;  
    int i;  
    unsigned int ui;  
    float f;  
    double d;  
    long long int i64;  
    unsigned long long int ui64;  
    char *str;  
    unsigned char *bytes;
```

```
// int DllMain() must exist and return 1 for a process to load the .DLL  
// See https://docs.microsoft.com/en-us/windows/win32/dllmain for more information.  
int __stdcall DllMain(void *module, unsigned int reason, void *reserved) { return 1; }
```

```
// #undef pin names lest they collide with names in any header file(s) you might include.  
#undef in  
#undef out
```

[T4] Display()

Functions and Variables

```
[T2] MaxExtStepSize()  
To determine maximum time step
```

```
[T3] Trunc()  
To determine if any next timestep
```

```
struct sDLLEX_X1  
{  
    // declare the structure here  
};
```

[T1] Struct template

Define pointer data used throughout the code

```
extern "C" __declspec(dllexport) void dllex_x1(struct sDLLEX_X1 **opaque, double t, union uData *data)  
{  
    double in = data[0].d; // input  
    double K = data[1].d; // input parameter  
    double *out = data[2].d; // output  
  
    if(!*opaque) // if(!*opaque) only run at beginning  
    {  
        *opaque = (struct sDLLEX_X1 *) malloc(sizeof(struct sDLLEX_X1));  
        bzero(*opaque, sizeof(struct sDLLEX_X1));  
    }  
    struct sDLLEX_X1 *inst = *opaque;  
  
    // Implement module evaluation code here:  
}  
  
extern "C" __declspec(dllexport) double MaxExtStepSize(struct sDLLEX_X1 *inst, double t)  
{  
    return 1e308; // implement a good choice of max timestep size that depends on struct sDLLEX_X1  
}  
  
extern "C" __declspec(dllexport) void Trunc(struct sDLLEX_X1 *inst, double t, union uData *data, double *timestep)  
{ // limit the timestep to a tolerance if the circuit causes a change in struct sDLLEX_X1  
    const double ttol = 1e-9; // ins default tolerance  
    if(*timestep > ttol)  
    {  
        struct sDLLEX_X1 tmp = *inst;  
        dllex_x1(&(tmp), t, data);  
        if(tmp != *inst) // implement a meaningful way to detect if the state has changed  
            *timestep = ttol;  
    }  
}  
  
extern "C" __declspec(dllexport) void Destroy(struct sDLLEX_X1 *inst)  
{  
    free(inst);  
}
```

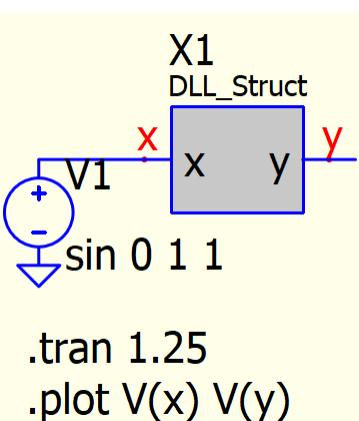
[T4] destroy()

Call this function upon termination of the simulation

Ø-Device (Template) [T1] : Struct()

Qspice : \02 Template\DLL_Struct\

- struct : Structure
 - Structure is utilized to store and manipulate data within the DLL functions
 - This example demonstrate a structure named sDLL_STRUCT contains a single member float z



Output Window

```
X1: dll_struct() - Inst->z = 0.993794
X1: dll_struct() - Inst->z = 0.994940
X1: dll_struct() - Inst->z = 0.995970
X1: dll_struct() - Inst->z = 0.996884
X1: dll_struct() - Inst->z = 0.997680
X1: dll_struct() - Inst->z = 0.998360
X1: dll_struct() - Inst->z = 0.998922
X1: dll_struct() - Inst->z = 0.999367
X1: dll_struct() - Inst->z = 0.999694
X1: dll_struct() - Inst->z = 0.999903
X1: dll_struct() - Inst->z = 0.999995
X1: dll_struct() - Inst->z = 1.000000

Total elapsed time: 4.76094 seconds.
X1: Destory fcn is executed :
X1: Destory() - Inst->z = 1.000000
```

```
struct sDLL_STRUCT
{
    // declare the structure here
    float z;
};

extern "C" __declspec(dllexport) void dll_struct(struct sDLL_STRUCT **opaque)
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLL_STRUCT *) malloc(sizeof(struct sDLL_STRUCT));
        bzero(*opaque, sizeof(struct sDLL_STRUCT));
    }
    struct sDLL_STRUCT *inst = *opaque;
    y = x*x;
    inst->z = y;
    Display("dll_struct() - Inst->z = %f\n",inst->z);
}

// Implement module
y = x*x;
inst->z = y;
Display("dll_struct() - Inst->z = %f\n",inst->z);

extern "C" __declspec(dllexport) void Destory(struct sDLL_STRUCT *inst)
{
    Display("Destory fcn is executed : \n");
    Display("Destory() - Inst->z = %f\n",inst->z);
    free(inst);
}
```

y cannot be pass outside dll_struct(), but by inst->z=y, this value can pass to other function

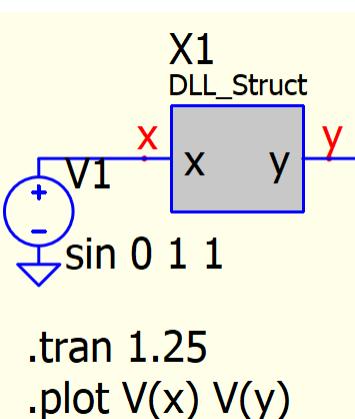
inst-z in Destory()

Ø-Device (Template) [T1] : Struct()

Qspice : \02 Template\DLL_Struct\

- struct : Structure

- Structure is utilized to store and manipulate data within the DLL functions
- This example demonstrate
 - A structure named sDLL_STRUCT contains a single member float z
 - An example to show how to declare a function maths() to be used



```
Output Window
```

X1: dll_struct() - Inst->z = 0.993794
X1: dll_struct() - Inst->z = 0.994940
X1: dll_struct() - Inst->z = 0.995970
X1: dll_struct() - Inst->z = 0.996884
X1: dll_struct() - Inst->z = 0.997680
X1: dll_struct() - Inst->z = 0.998360
X1: dll_struct() - Inst->z = 0.998922
X1: dll_struct() - Inst->z = 0.999367
X1: dll_struct() - Inst->z = 0.999694
X1: dll_struct() - Inst->z = 0.999903
X1: dll_struct() - Inst->z = 0.999995
X1: dll_struct() - Inst->z = 1.000000

Total elapsed time: 4.76094 seconds.

X1: Destory fcn is executed :

X1: Destory() - Inst->z = 1.000000

```
// Forward declaration of the math function
float maths(float val);

struct sDLL_STRUCT
{
    // declare the structure here
    float z;
};

extern "C" __declspec(dllexport) void dll_struct(struct sDLL_STRUCT **opaque)
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLL_STRUCT *) malloc(sizeof(struct sDLL_STRUCT));
        bzero(*opaque, sizeof(struct sDLL_STRUCT));
    }
    struct sDLL_STRUCT *inst = *opaque;

    // Implement module code
    y = maths(x);
    inst->z = y; // y cannot be pass outside dll_struct(), but by inst->z=y, this value can pass to other function
    Display("dll_struct() - Inst->z = %f\n",inst->z);
}

// Function prototype for the math function
float maths(float val)
{
    return (val * val);
}

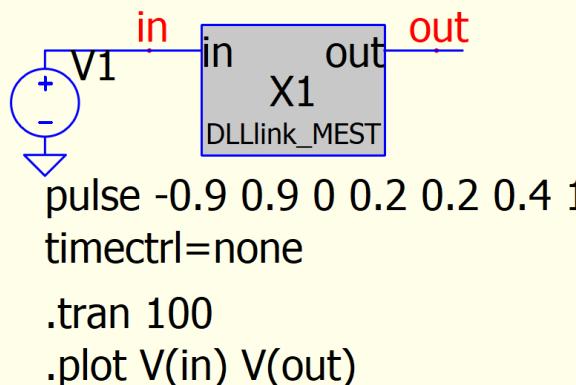
extern "C" __declspec(dllexport) void Destroy(struct sDLL_STRUCT *inst)
{
    Display("Destory fcn is executed : \n");
    Display("Destory() - Inst->z = %f\n",inst->z);
    free(inst);
}
```

inst-z in Destory()

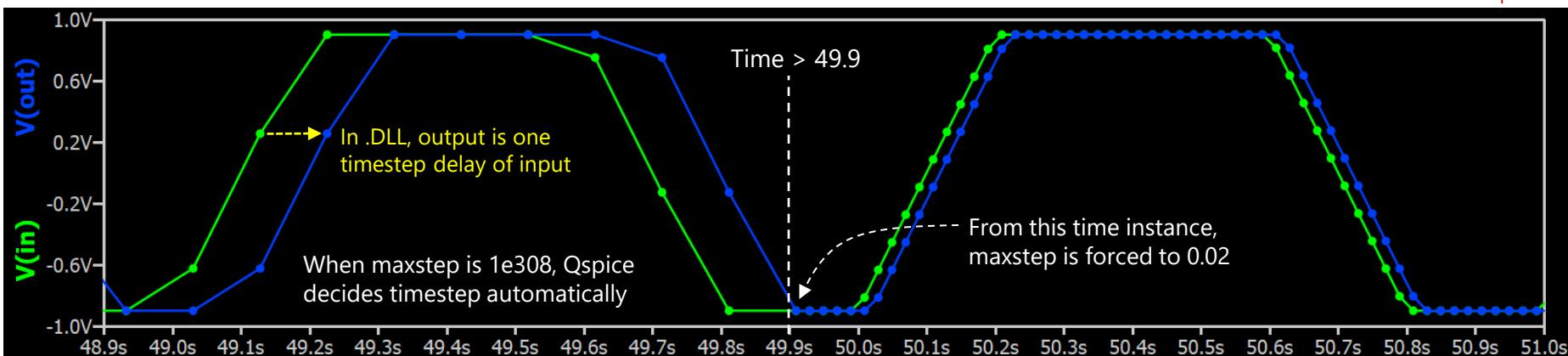
Ø-Device (Template) [T2] : MaxExtStepSize()

Qspice : \02 Template\DLLlink_MaxExtStepSize\

- `MaxExtStepSize()`
 - Return value in `MaxExtStepSize()` determines maxstep
 - Return `1e308` equivalent maxstep is infinite (default)
 - This example demonstrates change of maxstep from default to 0.02 during simulation



```
struct sDLLLINK_MEST  
{  
    // declare the structure here  
    float t; ← assign an inst->t  
};  
  
extern "C" __declspec(dllexport) void dlllink_mest(struct  
{  
    double in = data[0].d; // input  
    double &out = data[1].d; // output  
  
    if(!*opaque)  
    {  
        *opaque = (struct sDLLLINK_MEST *) malloc(sizeof(  
            bzero(*opaque, sizeof(struct sDLLLINK_MEST));  
    }  
    struct sDLLLINK_MEST *inst = *opaque;  
  
    // Implement module evaluation code here:  
    out = in; ← inst->t = t as MaxExtStepSize() will  
    inst->t = t; ← not accept t but only inst->t  
};  
  
extern "C" __declspec(dllexport) double MaxExtStepSize()  
{  
    //return 1e308; // implement a good choice of max ti  
    if (inst->t > 49.9)  
        return 0.2e-1;  
    else  
        return 1e308; ← If condition  
    }  
    • Before t<49.9s, maxstep set to max  
    • After t>49.9s set maxstep=0.02
```

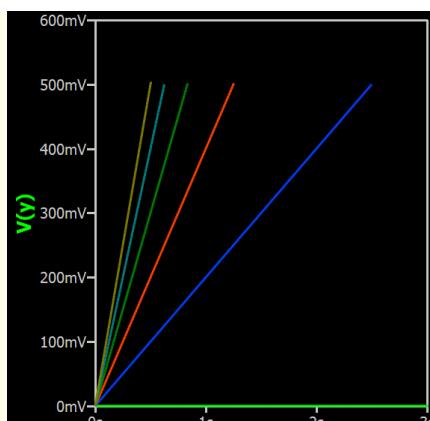
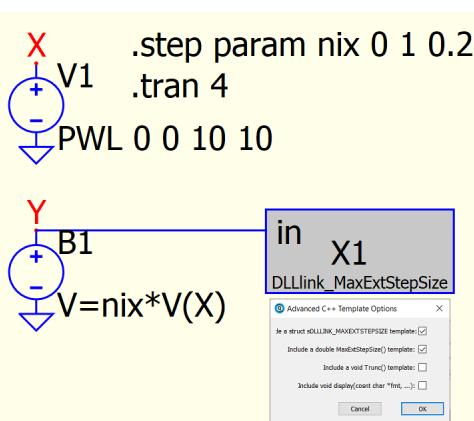


Ø-Device (Template) [T2] : MaxExtStepSize()

Qspice : \02 Template\DLLlink_MaxExtStepSize_Abort\

- MaxExtStepSize()

- A special usage (abort simulation)
 - Have MaxExtStepSize() to return -1e308 can abort simulation and goes to the next step (if any)
 - ** Specifically -1e308, no other number



```
struct sDLLLINK_MAXEXTSTEPSIZE
{
    // declare the structure here
    double x; Declare pointer inst->x
};

extern "C" __declspec(dllexport) void dlllink_maxextste
{
    double in = data[0].d; // input

    if(!*opaque)
    {
        *opaque = (struct sDLLLINK_MAXEXTSTEPSIZE *) mal
        bzero(*opaque, sizeof(struct sDLLLINK_MAXEXTSTEP
    }
    struct sDLLLINK_MAXEXTSTEPSIZE *inst = *opaque;

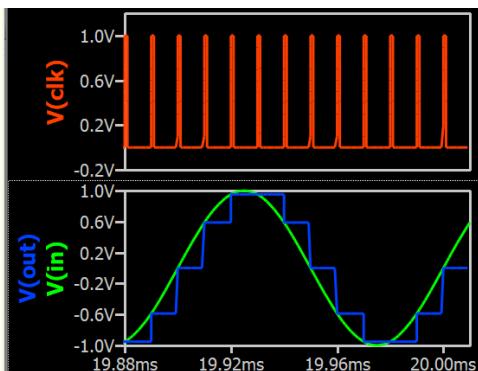
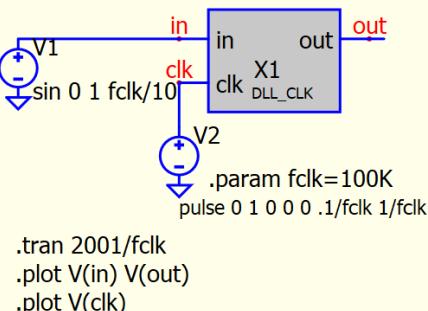
// Implement module evaluation code here:
    inst->x = in; Pointer is required to pass value of "in" to other
} function, in this case, to EaxExtStepSize()
}

extern "C" __declspec(dllexport) double MaxExtStepSize(
{
    if (inst->x > 0.5) {return -1e308;}
    return 1e308; // implement a good choice of max time
}
```

Ø-Device (Template) [T3] : Trunc(), Example#1 - Rising Edge Detection

Qspice : \02 Template\DLL_CLK\

- Advanced C++ Template Options
 - Include a void **Trunc() template**
 - Include a **struct** template
- Purpose of this example
 - Limit timestep with Trunc() if the circuit causes a change
 - Struct to store instance variable which only used within C++ code (pointer to pass variable between functions)
 - Demonstrate how to implement rising edge detection and sampling time calculation



```
struct sDLL_CLK
{
    // declare the structure here
    bool last_clk;
    double lastT;
};

extern "C" __declspec(dllexport) void dll_clk(struct sDLL_CLK
{
    bool clk = data[0].b; // input
    double in = data[1].d; // input
    double &out = data[2].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLL_CLK *) malloc(sizeof(struct sDLL_CLK));
        bzero(*opaque, sizeof(struct sDLL_CLK));
    }
    struct sDLL_CLK *inst = *opaque;

    // Implement module evaluation code here:
    if (clk & !inst->last_clk){
        double T = t - inst->lastT; // T : sampling period ca
        inst->lastT = t;
        out = in;
    }
    inst->last_clk = clk;
}

extern "C" __declspec(dllexport) void Trunc(struct sDLL_CLK
{ // limit the timestep to a tolerance if the circuit ca
const double ttol = 1e-9; ← Line#74, next slide
    if(*timestep > ttol)
    {
        double &out = data[2].d; // output

        // Save output vector
        const double _out = out;

        struct sDLL_CLK tmp = *inst;
        dll_clk(&(tmp), t, data);
        // if(tmp != *inst) // implement a meaningful way to detect
        // *timestep = ttol;
        if((tmp.last_clk != inst->last_clk) & !inst->last_clk)
            *timestep = ttol;

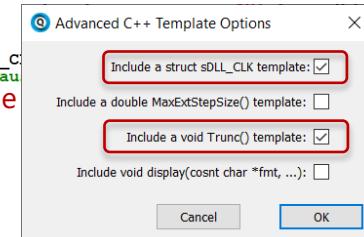
        // Restore output vector
        out = _out;
    }
}
```

[1] Declare structure pointers
inst->last_clk : to store last clock
inst->last : to store last rising edge time

User code

[2] detect rising edge (last_clk=0, clk=1)

[3] Calculate time between two rising edge (T)

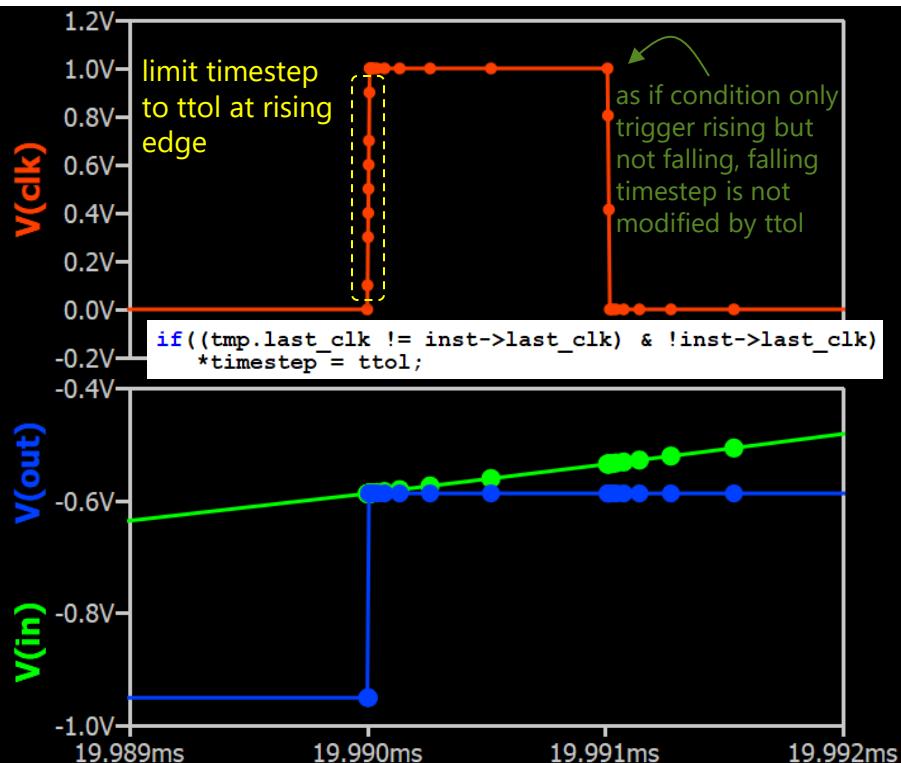


[4] if the circuit causes a change, limit the timestep to *timestep=ttol
User needs to define what the change (if condition) to force *timestep=ttol

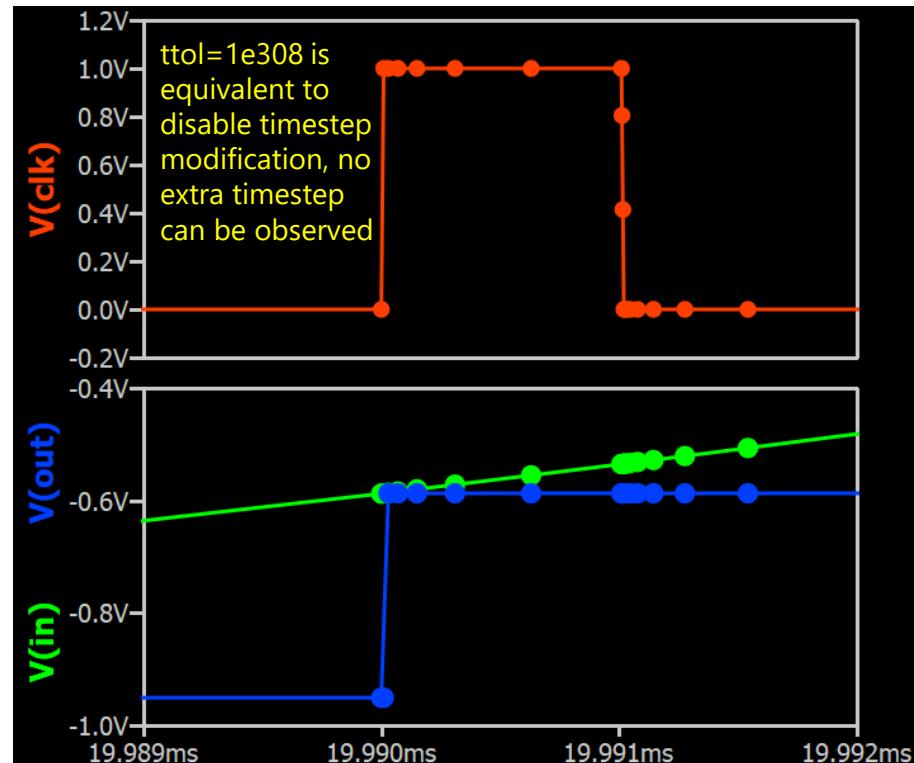
\emptyset -Device (Template) [T3] : Trunc() and ttol

Qspice : \02 Template\DLL_CLK\

Line#74 : const double ttol=1e-9;



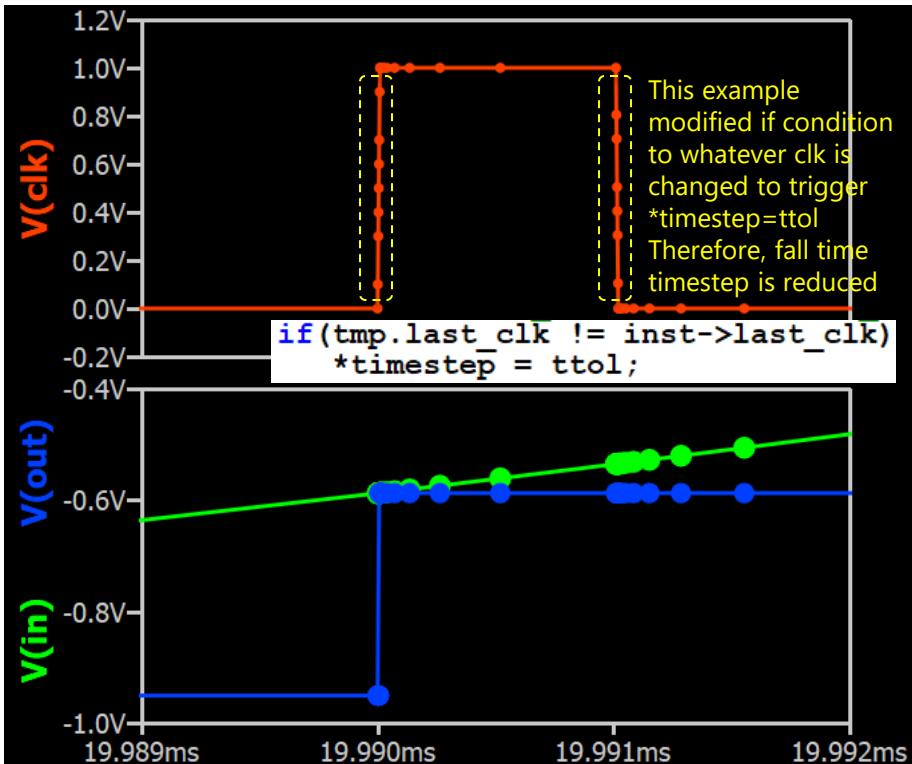
Line#74 : const double ttol=1e308;



Ø-Device (Template) [T3] : Trunc() and ttol

Qspice : \02 Template\DLL CLK

Line#74 : const double ttol=1e-9;



Comment of Trunc() template

- Most critical in Trunc() for user to modify are these 2 lines

```
const double ttol = 1e-9;  
// if(tmp != *inst) // implement a meaningful way to detect if the state has changed  
// *timestep = ttol;
```

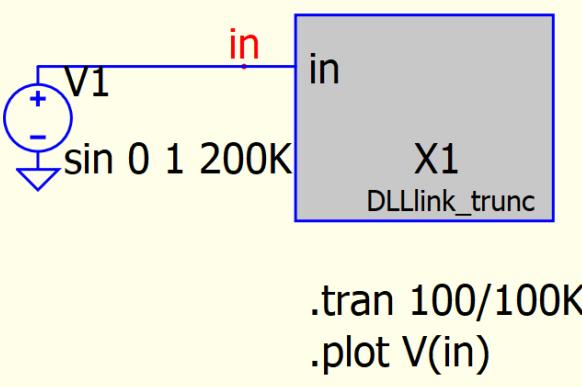
- ttol : limit Temporal tolerance timestep
- if (condition) : condition where timestep temporary force to ttol
 - In Trunc(), struct tmp is created for the purpose to compare *inst
 - User has to utilize tmp and *inst to detect if the state has changed

Ø-Device (Template) [T3] : Trunc() and ttol – Example#2 Zero-crossing

Qspice : \02 Template\DLLlink_Trunc\

- Trunc() example

- This example is to reduce simulation timestep at signal zero-crossing
- Code keynote
 - Use `inst->x = in` to pass data to Trunc()
 - In Trunc(), use if condition to force ttol when `tmp.x` within ± 0.1



```
struct sDLLLINK_TRUNC
{
    // declare the structure here
    double x;
};

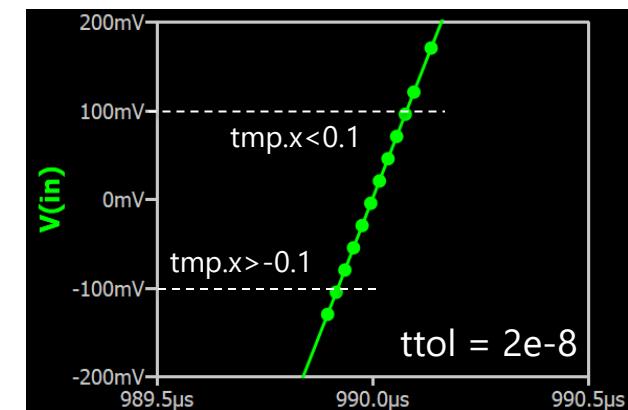
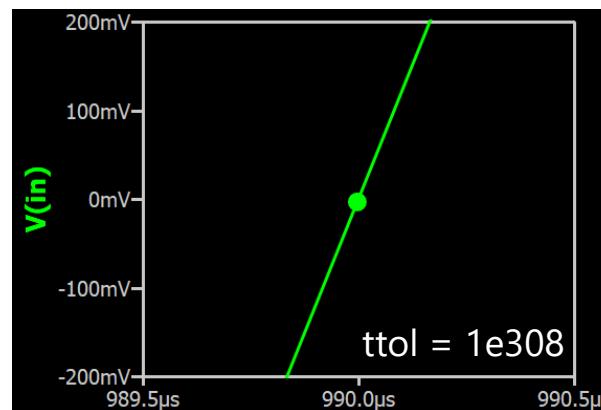
extern "C" __declspec(dllexport) void dll
{
    double in = data[0].d; // input

    if (!*opaque)
    {
        *opaque = (struct sDLLLINK_TRUNC *) bzero(*opaque, sizeof(struct sDLLLINK_TRUNC));
        struct sDLLLINK_TRUNC *inst = *opaque;

        // Implement module evaluation code here:
        inst->x = in;
    }
}
```

```
const double ttol = 2e-8;
if (*timestep > ttol)
{
    // Save output vector
    struct sDLLLINK_TRUNC tmp = *inst;
    dlllink_trunc(&tmp, t, data);
    // if(tmp != *inst) // implement a measure
    //     *timestep = ttol;
    if (tmp.x < 0.1 & tmp.x > -0.1)
        *timestep = ttol;
}
```

** If use `inst->x` to replace `tmp.x`, couldn't achieve this pattern



Ø-Device (Template) [T4] : Function and variable pointers

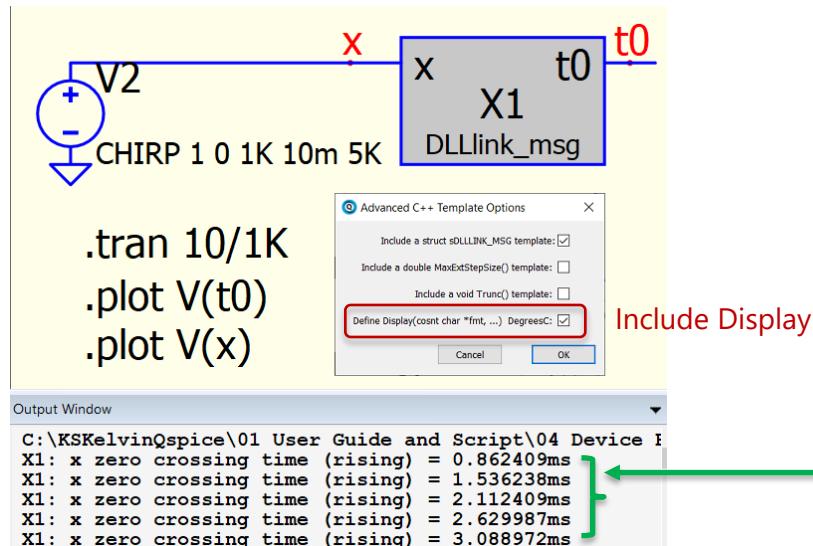
Qspice : \02 Template\Template Pointers\01 Set of Linked Functions

```
● extern "C" __declspec(dllexport) void (*Display)(const char *format, ...) = 0; // works like printf()
● extern "C" __declspec(dllexport) void (*EXIT)(const char *format, ...) = 0; // print message like printf() but exit(0) afterward
● extern "C" __declspec(dllexport) const double *DegreesC = 0; // pointer to current circuit temperature
● extern "C" __declspec(dllexport) const int *StepNumber = 0; // pointer to current step number
● extern "C" __declspec(dllexport) const int *NumberSteps = 0; // pointer to estimated number of steps
● extern "C" __declspec(dllexport) const char* const *InstanceName = 0; // pointer to address of instance name
● extern "C" __declspec(dllexport) const char *QUX = 0; // path to QUX.exe
● extern "C" __declspec(dllexport) const bool *ForKeeps = 0; // pointer to whether being evaluated non-hypothetically
● extern "C" __declspec(dllexport) const bool *HoldICs = 0; // pointer to whether instance initial conditions are being held
● extern "C" __declspec(dllexport) const void *GUI_HWND = 0; // pointer to Window handle of QUX.exe
● extern "C" __declspec(dllexport) const double *CKTtime = 0;
● extern "C" __declspec(dllexport) const double *CKTdelta = 0;
● extern "C" __declspec(dllexport) const int *IntegrationOrder = 0;
● extern "C" __declspec(dllexport) const char *InstallDirectory = 0;
● extern "C" __declspec(dllexport) double (*EngAtof)(const char **string) = 0;
● extern "C" __declspec(dllexport) const char *(*BinaryFormat)(unsigned int data) = 0; // BinaryFormat(0x1C) returns "0b00011100"
● extern "C" __declspec(dllexport) const char *(*EngFormat )(double x, const char *units, int numDgts) = 0; // EngFormat(1e-6, "s", 6) returns "1us"
● extern "C" __declspec(dllexport) int (*DFFT)(struct sComplex *u, bool inv, unsigned int N, double scale) = 0; // Discrete Fast Fourier Transform
```

Ø-Device : Function Pointers – Display()

Qspice : \02 Template\Template Pointers\Display\Display()\

- Display()
 - Display() works like printf() in output window during simulation
 - Device instance name is included in the console return
 - Display() is suppressed during Trunc() and hypothetically evaluation called by Trunc()
 - This example detect zero crossing of input x and output its time (simulation time) with Display()



```
struct sDLLLINK_MSG
{
    // declare the structure here
    double last_x;
};

extern "C" __declspec(dllexport) void dlllink_msg(struct sDLLLINK_
{
    double x = data[0].d; // input
    double &t0 = data[1].d; // output

    if (!*opaque)
    {
        *opaque = (struct sDLLLINK_MSG *) malloc(sizeof(struct sDLLLINK_MSG));
        bzero(*opaque, sizeof(struct sDLLLINK_MSG));
    }
    struct sDLLLINK_MSG *inst = *opaque;

    // Implement module evaluation code here:
    if (x>=0 & inst->last_x<0)
    {
        t0 = t;
        Display("x zero crossing time (rising) = %fms\n", t0/1e-3);
    }
    inst->last_x = x;           Display a string
}
```

Ø-Device : Function Pointers – Display() in Trunc() and hypothetical

- Display in Trunc() and hypothetically evaluation
 - As Display() is suppressed during Trunc() and hypothetically evaluation, below are three methods to do a console return
 - Use printf() - #include <stdio.h>
 - If you run the simulation from the console
 - FILE pointer
 - Put a FILE pointer in the instance structure and have Trunc() output write to that file (most reliable way)
 - Use the old method – declare a display() function
 - \Template Pointers\Display\display() - template before 29June24\

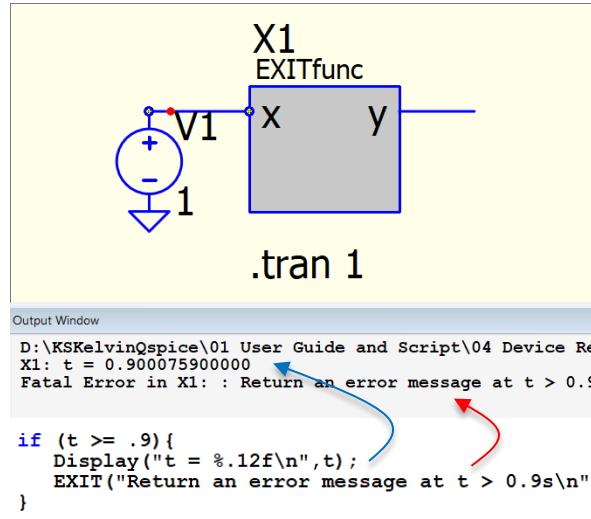
```
#include <stdio.h>
#include <malloc.h>
#include <stdarg.h>
#include <time.h>

void display(const char *fmt, ...)
{
    msleep(30);
    fflush(stdout);
    va_list args = { 0 };
    va_start(args, fmt);
    vprintf(fmt, args);
    va_end(args);
    fflush(stdout);
    msleep(30);
}
```

Ø-Device : Function Pointers – Display() and EXIT()

Qspice : \02 Template\Template Pointers\EXIT\

- Display() function
 - Print message like printf()
- EXIT() function
 - Print message like printf() but exit(0) afterward
 - A "Fatal Error" message is returned in output window and halt the simulation



Ø-Device : Variable Pointers – *DegreesC, *StepNumber, *NumberSteps, ...

Qspice : \02 Template\Template Pointers\DegreesC Step Instance QUX\

- *DegreesC
 - Pointer to current circuit temperature
- *StepNumber
 - Pointer to current step number (.step)
- *NumberSteps
 - Pointer to estimated number of steps (.step)
- *InstanceName
 - Pointer to address of instance name
- *QUX
 - Char pointer to path to QUX.exe

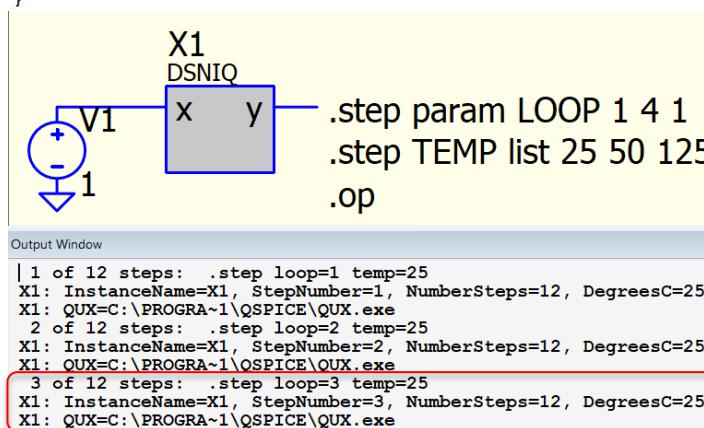
```
extern "C" __declspec(dllexport) const double *DegreesC = 0; // pointer to current circuit temperature
extern "C" __declspec(dllexport) const int *StepNumber = 0; // pointer to current step number
extern "C" __declspec(dllexport) const int *NumberSteps = 0; // pointer to estimated number of steps
extern "C" __declspec(dllexport) const char* const *InstanceName = 0; // pointer to address of instance name
extern "C" __declspec(dllexport) const char *QUX = 0; // path to QUX.exe

extern "C" __declspec(dllexport) void dsniq(struct sDSNIQ **opaque, double t
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDSNIQ *) malloc(sizeof(struct sDSNIQ));
        bzero(*opaque, sizeof(struct sDSNIQ));

        Display("InstanceName=%s, StepNumber=%d, ", *InstanceName, *StepNumber);
        Display("NumberSteps=%d, DegreesC=%0.f\n", *NumberSteps, *DegreesC);
        Display("QUX=%s\n", QUX);
    }
    struct sDSNIQ *inst = *opaque;

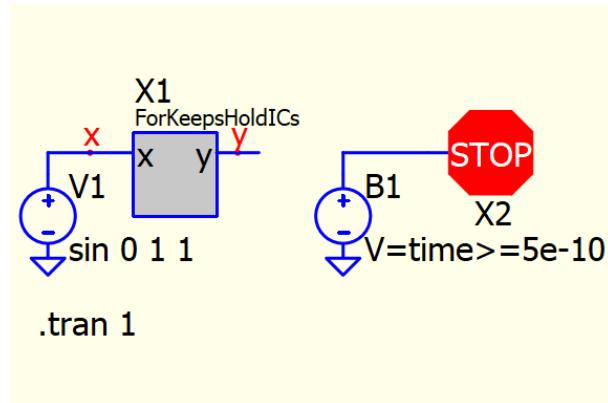
    // Implement module evaluation code here:
}
```



Ø-Device : Variable Pointers – *ForKeeps, *HoldICs

Qspice : \02 Template\Template Pointers\ForKeeps HoldICs\

- *ForKeeps : pointer to whether being evaluated non-hypothetically
 - Non-hypothetical = 1
 - Hypothetical = 0
- *HoldICs : pointer to whether instance initial conditions are being held
- These pointers are useful in transient simulation, to identify which instance is the first attempt in transient simulation (ForKeeps is true)



```
#include <malloc.h>

extern "C" __declspec(dllexport) int (*Display)(const char *format, ...) = 0;
extern "C" __declspec(dllexport) const double *DegreesC = 0;
extern "C" __declspec(dllexport) const int *StepNumber = 0;
extern "C" __declspec(dllexport) const int *NumberSteps = 0;
extern "C" __declspec(dllexport) const char* const *InstanceName = 0;
extern "C" __declspec(dllexport) const char *QNX = 0;
extern "C" __declspec(dllexport) const bool *ForKeeps = 0;
extern "C" __declspec(dllexport) const bool *HoldICs = 0;
extern "C" __declspec(dllexport) int (*DFFT)(struct sComplex *u, bool inv, uns

extern "C" __declspec(dllexport) void forkeepsholdics(struct sFORKEEPSHOLDICS
{
    double x = data[0].d; // input
    double dy = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sFORKEEPSHOLDICS *) malloc(sizeof(struct sFORKEEPSHOLDICS));
        bzero(*opaque, sizeof(struct sFORKEEPSHOLDICS));
    }
    struct sFORKEEPSHOLDICS *inst = *opaque;

    // Implement module evaluation code here:
    Display("t=%f, ForKeeps=%d, HoldICs=%d\n", t, *ForKeeps, *HoldICs);
}
```

Display("t=%f, ForKeeps=%d, HoldICs=%d\n", t, *ForKeeps, *HoldICs);

Output Window

```
C:\KSKelvinQspice\01 User Guide and Script\04 I
X1: t=0.0000000000000000, ForKeeps=0, HoldICs=1
X1: t=0.0000000000000000, ForKeeps=1, HoldICs=1 ← 1st attempt in transient
X1: t=0.0000000002500000, ForKeeps=1, HoldICs=0
X1: t=0.0000000005000000, ForKeeps=1, HoldICs=0

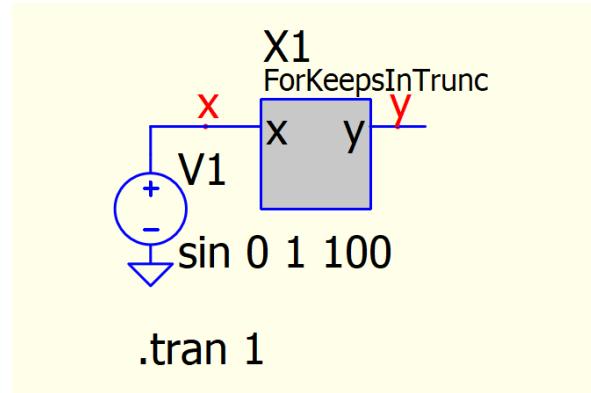
Total elapsed time: 0.0905704 seconds.
```

1st attempt in transient simulation : ForKeeps is true

Ø-Device : Variable Pointers – *ForKeeps

Qspice : \02 Template\Template Pointers\ForKeeps in Trunc\

- *ForKeeps
 - Pointer to whether being evaluated non-hypothetically
 - If Trunc() is included for the temporal timestep (TTOL), the main function can be called multiple times, and being called through Trunc() is hypothetical
 - Display() only print result in non-hypothetical evaluation, but printf() or fprintf() in the main evaluation outputs results regardless of whether it is called through Trunc() or not
 - *ForKeeps helps to identify whether the main evaluation is called hypothetically or not



Output Window

```
>>> In main() : t = 0.0041125298, ForKeeps = 1
>>> In Trunc(): t = 0.0042725298, ForKeeps = 0
>>> In Trunc(): t = 0.0041225298, ForKeeps = 0
>>> In main() : t = 0.0041225298, ForKeeps = 1
>>> In Trunc(): t = 0.0041425298, ForKeeps = 0
>>> In main() : t = 0.0041425298, ForKeeps = 1
>>> In Trunc(): t = 0.0041825298, ForKeeps = 0
>>> In Trunc(): t = 0.0041525298, ForKeeps = 0
>>> In main() : t = 0.0041525298, ForKeeps = 1
```

If *ForKeeps is 0, if it is a hypothetical call

```
struct sFORKEEPSINTRUNC
{
    // declare the structure here
    double last_y;
};

extern "C" __declspec(dllexport) void forkeepsintrun
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sFORKEEPSINTRUNC *) malloc(s
            bzero(*opaque);
    }
    struct sFORKEE
    // Implement modu

    if (x > 0.5) y = 1;
    else y = 0;

    if (*ForKeeps) printf(">>> In main() : ");
    printf("t = %.10f, ForKeeps = %d\n", t,*ForKeeps);
    inst->last_y = y;
}

extern "C" __declspec
{
    return 1e308; // in
}

extern "C" __declspec(dllexport) void Trunc(struct s
{
    // limit the timestep to a tolerance if the circui
    const double ttol = 1e-5; // ins default toleranc
    if(*timestep > ttol)
    {
        printf(">>> In Trunc(): ");
        struct sFORKEEPSINTRUNC tmp = *inst;
        forkeepsintrunc(&(tmp), t, data);
        // if(tmp != *inst) // implement a meaningful way
        // *timestep = ttol;
        if(tmp.last_y != inst->last_y)
            *timestep = ttol;
    }
}
```

If *ForKeeps is set, it indicate a standard main call

In this example, this printf() is always printed

If *ForKeeps is 0, if it is a hypothetical call

Ø-Device : Variable Pointers – *GUI_HWND (Windows Message Box)

Qspice : \02 Template\Template Pointers\WinMsgBox\

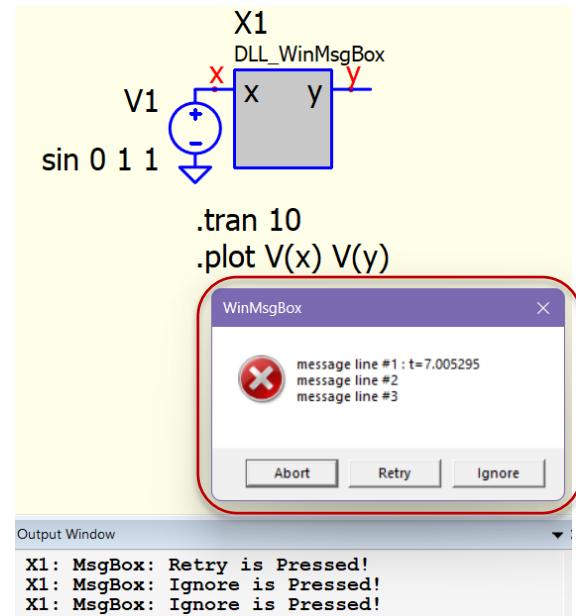
- Windows Message Box

- Add this definition : "extern "C" __declspec(dllexport) void *GUI_HWND = 0;"
 - This is an undocumented feature and mentioned in revision history
 - Reference example from RDunn (Robert Dunn)
<https://github.com/robdunn4/QSpice/tree/main/Miscellany>

Quote RDunn message

<https://forum.qorvo.com/t/using-windows-message-boxes-in-dll-code-gui-hwnd/22854>

- GUI_HWND is a Windows handle to the QSpice schematic window if the simulation was launched from the GUI. If the simulation was launched from a command line (i.e., with QSpice64.exe or QSpice80.exe), this is a null value.
- We can use GUI_HWND to display a Windows message box and get a user response from within a DLL.



Ø-Device : Variable Pointers – *GUI_HWND (Windows Message Box)

Qspice : \02 Template\Template Pointers\WinMsgBox\dll_winmsgbox.cpp

```
#include <malloc.h>
#include <Windows.h>
#include <stdio.h>

extern "C" __declspec(dllexport) int (*Display)(const char *format, ...) = 0; // works !
extern "C" __declspec(dllexport) const double *DegreesC = 0; // pointer
extern "C" __declspec(dllexport) const int *StepNumber = 0; // pointer
extern "C" __declspec(dllexport) const int *NumberSteps = 0; // pointer
extern "C" __declspec(dllexport) const char* const *InstanceName = 0; // pointer
extern "C" __declspec(dllexport) const char *QUX = 0; // path to
extern "C" __declspec(dllexport) const bool *ForKeeps = 0; // pointer
extern "C" __declspec(dllexport) const bool *HoldICs = 0; // pointer
extern "C" __declspec(dllexport) int (*DFET)(struct sComplex *, bool inv, unsigned int
extern "C" __declspec(dllexport) void *GUI_HWND = 0;

union uData
{
    bool b;
    char c;
    unsigned char uc;
    short s;
    unsigned short us;
    int i;
    unsigned int ui;
    float f;
    double d;
    long long int i64;
    unsigned long long int ui64;
    char *str;
    unsigned char *bytes;
};

// int DllMain() must exist and return 1 for a process to load the .DLL
// See https://docs.microsoft.com/en-us/windows/win32/dllmain for more information.
int __stdcall DllMain(void *module, unsigned int reason, void *reserved) { return 1; }

void bzero(void *ptr, unsigned int count)
{
    unsigned char *first = (unsigned char *) ptr;
    unsigned char *last = first + count;
    while(first < last)
        *first++ = '\0';
}

// #undef pin names lest they collide with names in any header file(s) you might include
#undef x
#undef y

struct sDLL_WINMSGBOX
{
    // declare the structure here
    double lastX;
};

/* Forward declaration of message box function */
int MsgBox(double t);
```

```
extern "C" __declspec(dllexport) void dll_winmsgbox(struct sDLL_WINMSGBOX **opaque, dou
{
    double x = data[0].d; // input
    double y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLL_WINMSGBOX *) malloc(sizeof(struct sDLL_WINMSGBOX));
        bzero(*opaque, sizeof(struct sDLL_WINMSGBOX));
    }
    struct sDLL_WINMSGBOX *inst = *opaque;

    // Implement module evaluation code here:

    // y set to high at x zero crossing [rising direction]
    y = false; // Initialize output signal 'y' to false (low state)
    if(inst->lastX < 0 & x >= 0){ // Detect rising edge zero-crossing
        y = true; // Set output 'y' to true (high state) on rising edge
    }

    // Show message box when zero-crossing occurs and capture user response
    int userChoice = MsgBox(t); // 't' is passed as a parameter
    switch (userChoice) // Handle user action
    {
        case IDABORT: // User clicked "Abort"
            Display("MsgBox: Abort is Pressed!\n");
            break;
        case IDRETRY: // User clicked "Retry"
            Display("MsgBox: Retry is Pressed!\n");
            break;
        case IDIGNORE: // User clicked "Ignore"
            Display("MsgBox: Ignore is Pressed!\n");
            break;
    }
    inst->lastX = x; // Update lastX for next zero-crossing detection
}

// Displays a modal message box with Abort/Retry/Ignore options
int MsgBox(double t)
{
    char msg[512];
    sprintf(msg,
        " message line #1 : t=%lf\n" // Line 1: Shows time 't'
        " message line #2\n" // Line 2: Placeholder for additional info
        " message line #3\n" // Line 3: Placeholder + extra spacing
        ,t);
    // show the msgbox, return result
    int userChoice = MessageBoxA(
        static_cast<HWND>(GUI_HWND), // Parent window handle (cast from global GUI_HWND)
        msg, // Formatted message text
        "WinMsgBox", // Title of the message box
        MB_ABORTRETRYIGNORE | MB_ICONERROR | MB_APPLMODAL // Flags:
        // - Buttons: Abort/Retry/Ignore
        // - Icon: Red error symbol (X)
        // - Behavior: Blocks parent app until dismissed
    );
    return userChoice; // Return IDABORT (3), IDRETRY (4), or IDIGNORE (5)
}
```

Ø-Device : Variable Pointers – *GUI_HWND (Windows Message Box) MessageBoxA() Reference – Buttons

- MessageBoxA : Buttons
 - MB_OK (use case : basic notification)
 - OK → Return IDOK (1)
 - MB_OKCANCEL (use case : confirmations)
 - OK → Return IDOK (1) | Cancel → Return IDCANCEL (2)
 - MB_ABORTRETRYIGNORE
 - Abort → Return IDABORT (3) | Retry → Return IDRETRY (4) | Ignore → Return IDIGNORE (5)
 - MB_YESNO (use case : binary choices)
 - Yes → Return IDYES (6) | No → Return IDNO (7)
 - MB_YESNOCANCEL (use case : optional cancellations)
 - Yes → Return IDYES (6) | No → Return IDNO (7) | Cancel → Return IDCANCEL (2)
 - MB_RETRYCANCEL (use case : temporary failures)
 - Retry → Return IDRETRY (4) | Cancel → Return IDCANCEL (2)

Ø-Device : Variable Pointers – *GUI_HWND (Windows Message Box) MessageBoxA() Reference – Icon Flags, Modality Flags

- MessageBoxA : Icon Flags
 - MB_ICONERROR (Red "X" symbol) : Critical errors
 - MB_ICONWARNING (Yellow "!" triangle) : Warnings
 - MB_ICONINFORMATION (Blue "i" icon) : Informational messages
 - MB_ICONQUESTION (Question mark, now deprecated by Microsoft, avoid)
- MessageBoxA : Modality Flags (Behavior Control)
 - MB_APPLMODAL : Blocks only the current app (default)
 - MB_SYSTEMMODAL : Blocks all apps (rare; use sparingly)
 - MB_TASKMODAL : Like MB_APPLMODAL, but safer for threads
- Remark:
 - Return values are integers (e.g., IDOK, IDYES), but you can compare them directly with the constants.
 - Icons and buttons are combined with | (bitwise OR).

Ø-Device : Variable Pointers – *CKTtime and *CKTdelta

Qspice : \02 Template\Template Pointers\CKT\

- ***CKTtime**
 - *CKTtime represents the simulation time in non-hypothetical evaluation
 - In a non-hypothetical evaluation with *ForKeeps=1, t and *CKTtime are identical
 - In hypothetical evaluations (e.g. Trunc() or ckt(&(&tmp), t, data), where t is hypothetical, *CKTtime can still return a non-hypothetical value

- ***CKTdelta**
 - *CKTdelta represents the delta time between two simulation steps
 - In a non-hypothetical evaluation with *ForKeeps=1, *CKTdelta is the time difference between the current and previous timestamps i.e. t – inst->lastT
 - In hypothetical evaluations, *CKTdelta equals the difference between the hypothetical time t and *CKTtime

Non-hypothetical evaluation

```
>m: *ForKeeps=1
>m: t=0.00009400000; *CKTtime=0.00009400000
>m: dT=t-inst->lastT=0.00002000000
>m: *CKTdelta=0.00002000000
>m: *CKTnoncon=0;

>m: *ForKeeps=0 [In Hypothetical]
>m: t=0.000013400000; *CKTtime=0.00009400000
>m: *CKTdelta=0.00004000000
>m: *CKTnoncon=0;
>t: *ForKeeps=0 [In Trunc()]
>t: t=0.000013400000; *CKTtime=0.00009400000
>t: *CKTdelta=0.00004000000
>t: *CKTnoncon=0;

>m: *ForKeeps=0 [In Hypothetical]
>m: t=0.000010400000; *CKTtime=0.00009400000
>m: *CKTdelta=0.00001000000
>m: *CKTnoncon=0;
>t: *ForKeeps=0 [In Trunc()]
>t: t=0.000010400000; *CKTtime=0.00009400000
>t: *CKTdelta=0.00001000000
>t: *CKTnoncon=0;

>m: *ForKeeps=1
>m: t=0.000010400000; *CKTtime=0.000010400000
>m: dT=t-inst->lastT=0.00001000000
>m: *CKTdelta=0.00001000000
>m: *CKTnoncon=0;
```

Hypothetical evaluation

```
extern "C" __declspec(dllexport) void ckt(struct sCKT **opaque, d
{
    double x = data[0].d; // input
    double y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sCKT *) malloc(sizeof(struct sCKT));
        bzero(*opaque, sizeof(struct sCKT));
    }
    struct sCKT *inst = *opaque;

// Implement module evaluation code here:
    double dT = t - inst->lastT;

    if (*ForKeeps){
        display("m: *ForKeeps=%d\n", *ForKeeps);
        display("m: t=%f; *CKTtime=%f\n", t, *CKTtime);
        display("m: dT=%f=inst->lastT=%f\n", dT);
        display("m: *CKTdelta=%f\n", *CKTdelta);
        display("m: *CKTnoncon=%d\n", *CKTnoncon);
    }else{
        display("m: *ForKeeps=%d [In Hypothetical]\n", *ForKeeps);
        display("m: t=%f; *CKTtime=%f\n", t, *CKTtime);
        display("m: *CKTdelta=%f\n", *CKTdelta);
        display("m: *CKTnoncon=%d\n", *CKTnoncon);
    }

    if (*ForKeeps && t > 0.00001){
        EXIT("Time exceeded 0.00001s\n");
    }

    inst->lastT = t;
    inst->lastX = x;
}

extern "C" __declspec(dllexport) double MaxExtStepSize(struct sCK
{
    return 1e308; // implement a good choice of max timestep size
}

extern "C" __declspec(dllexport) void Trunc(struct sCKT *inst, do
{
    // limit the timestep to a tolerance if the circuit causes a ch
    const double ttol = 1e-6; // ins default tolerance
    if(*timestep > ttol)
    {
        struct sCKT tmp = *inst;
        ckt(&tmp, t, data);

        display("t: *ForKeeps=%d [In Trunc()]\n", *ForKeeps);
        display("t: t=%f; *CKTtime=%f\n", t, *CKTtime);
        display("t: *CKTdelta=%f\n", *CKTdelta);
        display("t: *CKTnoncon=%d\n", *CKTnoncon);

        if(tmp.lastX != inst->lastX)
            *timestep = ttol;
    }
}
```

Ø-Device : Variable Pointers – *IntegrationOrder

Qspice : \02 Template\Template Pointers\IntegrationOrder\

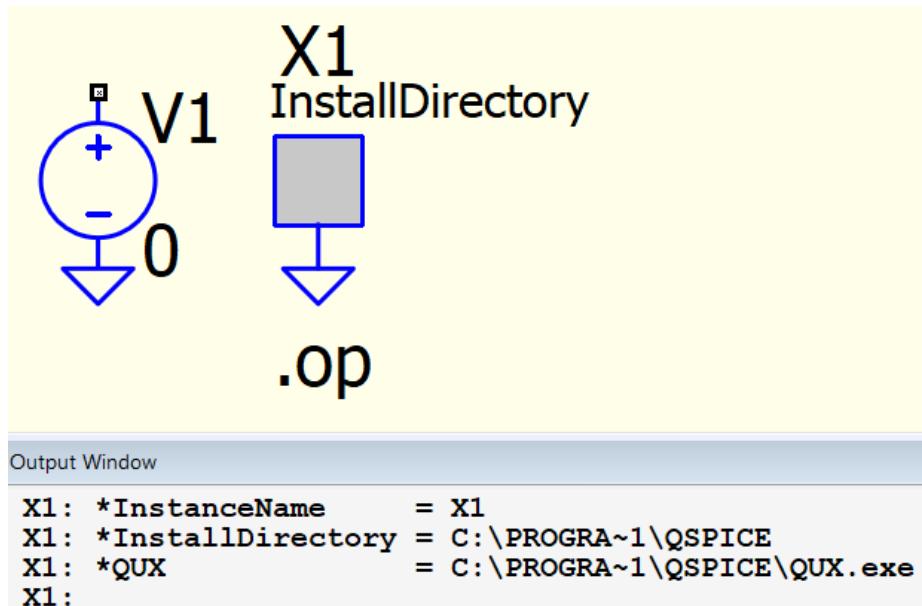
- *IntegrationOrder
 - Return value of **.option MAXORD** (Maximum integration order)

Ø-Device : Variable Pointers – *InstallDirectory

Qspice : \02 Template\Template Pointers\InstallDirectory\

- *InstallDirectory
 - Return Qspice installation directory path
- *InstanceName
 - Pointer to address of instance name
- *QUX
 - Char pointer of full path to QUX.exe

```
// Implement module evaluation code here:  
Display ("*InstanceName      = %s\n",*InstanceName );  
Display ("*InstallDirectory = %s\n",InstallDirectory);  
Display ("*QUX            = %s\n",QUX);
```



Ø-Device : Function Pointers – EngAtof() – Multiples metric formats

Qspice : \02 Template\Template Pointers\EngAtof\01 EngAtof\

- EngAtof()
 - double (*EngAtof)(const char **string)=0;
 - access to user-defined parameters, mathematical expressions, and SPICE metric multipliers
 - This is an example to read an input string that contains multiple numerical inputs with metric multiplier

```
const char *pArgs = args;
y1 = EngAtof (&pArgs);
y2 = EngAtof (&pArgs);
y3 = EngAtof (&pArgs);
y4 = EngAtof (&pArgs);
Display ("% .0e, %.0e, %.0e, %.0e\n", y1, y2, y3, y4);
```

X1
EngAtof

```
char *args="1u 1m 1 1k"
.ytran 1
.plot V(y1) V(y2) V(y3) V(y4)
```

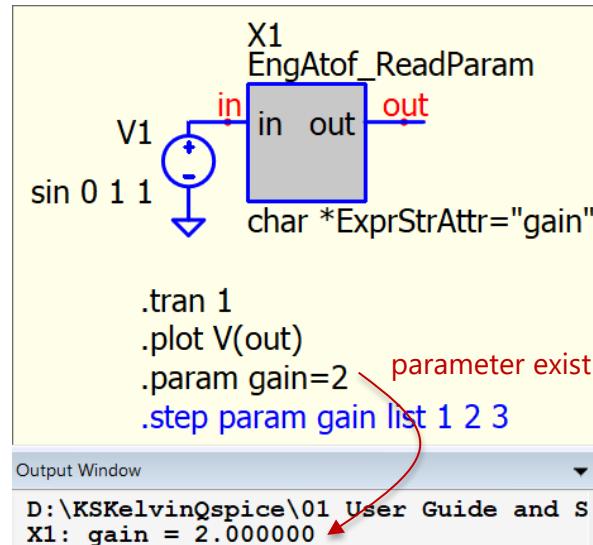
Output Window

```
D:\KSKelvinQspice\01 User Guid
X1: 1e-06,1e-03,1e+00,1e+03
```

Ø-Device : Function Pointers – EngAtof() – Read Parameters

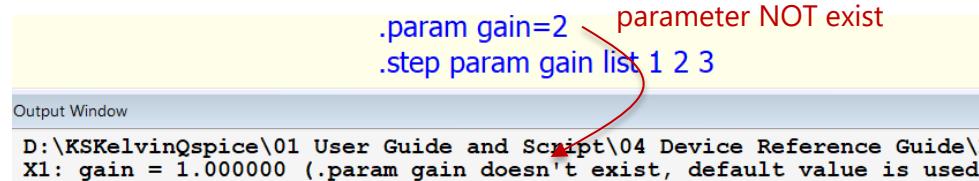
Qspice : \02 Template\Template Pointers\EngAtof\02a Read Param\

- EngAtof() : Read .param
 - EngAtof() can access to user-defined parameters
 - **param_name** is parameter name to search
 - **const char *pExpr = param_name** copy parameter name to point pExpr
 - **double value = EngAtof(&pExpr)** takes the parameter name as input and convert it value to a floating-point number
 - Return NaN if parameter doesn't exist
 - After EngAtof() operation, parameter name string is removed from pExpr. That why I copy the parameter name string as I can keep the parameter name string for Display purpose



```
// Check if the instance has not been initialized
if(!inst->init)
{
    // Parameter: "gain"
    const char *param_name = "gain";
    const char *pExpr = param_name;
    double value = EngAtof(&pExpr);
    double default_val = 1;
    if(!isnan(value))
        Display("%s = %f\n", param_name, value);
    else
    {
        Display("%s = %f (.param %s doesn't exist",
               value = default_val;
    }
    // Store the parameter value in the instance
    inst->DLLgain = value;

    inst->init = true; // Mark instance as initialized
}
```



Ø-Device : Function Pointers – EngAtof() – Read Parameters

Qspice : \02 Template\Template Pointers\EngAtof\02b Read Param - getparameter()\

- EngAtof() : Function to get parameter value from C++ in Qspice
 - This is the helper function to get parameter value
 - Usage : <value> = get_parameter("pName",0);
 - <value> is return value
 - "pName" is the parameter name expected in .param from netlist
 - 0 is default value if parameter name cannot be found in .param from netlist

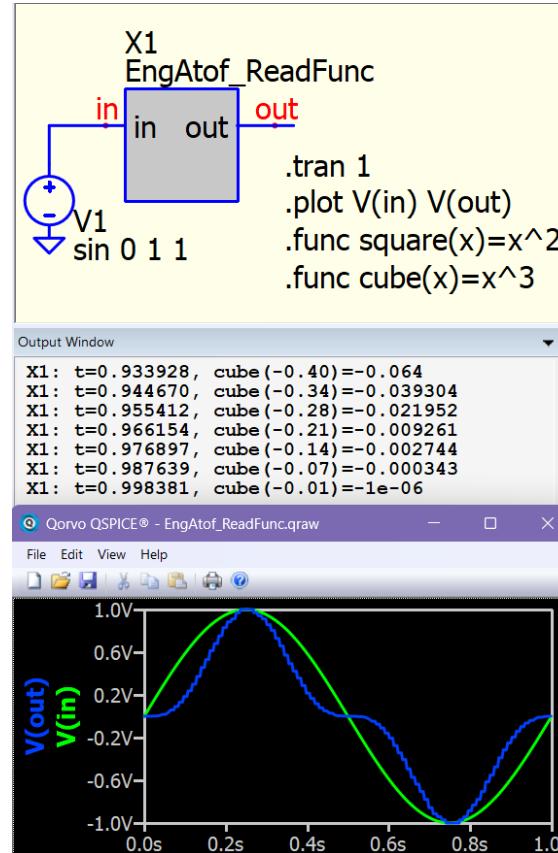
```
// Helper function to get parameter value
double get_parameter(const char *param_name, double default_val) {
    // Set pExpr to point to parameter name
    const char *pExpr = param_name;    // Pointer to the parameter name string
    double value = EngAtof(&pExpr);   // Convert parameter value to double using EngAtof function

    // Check if the conversion resulted in a valid number (not NaN)
    if(!isnan(value)) {
        // Display the parameter value if valid
        Display("%s = %g\n", param_name, value);
    } else {
        // Display warning and use default value if parameter doesn't exist
        Display("%s = %g (.param %s doesn't exist, default value is used)\n", param_name, default_val, param_name);
        value = default_val; // Use the provided default value
    }
    return value; // Return the parameter value (or default if not found)
}
```

Ø-Device : Function Pointers – EngAtof() – Read Functions

Qspice : \02 Template\Template Pointers\EngAtof\03a Read Func\

- EngAtof() : Read .func
 - **EngAtof()** can access to mathematical expressions in user-defined functions
 - **FnExpr** is char variable of function expression, **sprintf** [#include <cstdio>] is to format the function call
 - **Const char *pExpr = (const char*)&FnExpr** takes function expression as input and compute it result
 - Return NaN if function expression doesn't exist

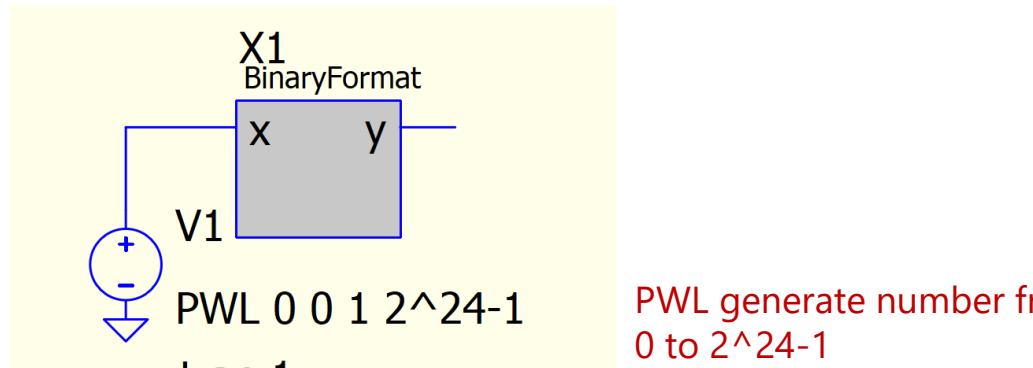


```
double sampleperiod = 0.01;
// Check if enough time has passed since the last
if(t - inst->Tprevious > sampleperiod)
{
    char FnExpr[128]; // buffer for expression string
    //Format a function call expression string, c1
    sprintf(FnExpr, "cube(%2f)", in);
    const char *pExpr = (const char*)&FnExpr; //(
    double value = EngAtof(&pExpr);
    if(isnan(value)){
        Display("t=%6f, %s=%g\n", t, FnExpr, value);
    }
    else{
        Display("t=%6f, %s function doesn't exist",
               value = 0;
    }
    // Set the output to the evaluated function value
    out = value;
    // Update the previous execution time to current
    inst->Tprevious = t;
}
```

Ø-Device : Function Pointers – BinaryFormat()

Qspice : \02 Template\Template Pointers\BinaryFormat\

- BinaryFormat()
 - const char *(**BinaryFormat**)(unsigned int data)
 - Function to convert an unsigned integer into binary number in string format



PWL generate number from
0 to 2²⁴-1
= 16777216-1 = 16777215

```
// Implement module evaluation code here:  
unsigned int dec = x;  
Display("x=%11.2f, dec=%8d, binary=%s\n", x, dec, BinaryFormat(dec));
```

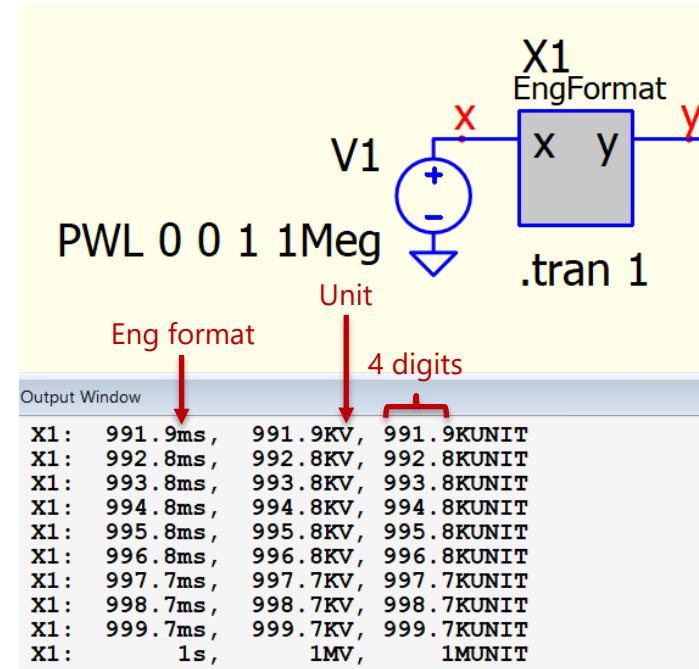
Output Window

```
X1: x=16640862.80, dec=16640862, binary=0b11111011110101101011110  
X1: x=16657246.80, dec=16657246, binary=0b11111100010101101011110  
X1: x=16673630.80, dec=16673630, binary=0b111111100110101101011110  
X1: x=16690014.80, dec=16690014, binary=0b111111101010101101011110  
X1: x=16706398.79, dec=16706398, binary=0b111111101110101101011110  
X1: x=16722782.79, dec=16722782, binary=0b111111110010101101011110  
X1: x=16739166.79, dec=16739166, binary=0b111111110110101101011110  
X1: x=16755550.79, dec=16755550, binary=0b111111111010101101011110  
X1: x=16771934.79, dec=16771934, binary=0b111111111110101101011110  
X1: x=16777215.00, dec=16777215, binary=0b11111111111111111111111111111111
```

Ø-Device : Function Pointers – EngFormat()

Qspice : \02 Template\Template Pointers\EngFormat\

- EngFormat()
 - const char *(*EngFormat)(double x, const char *units, int numDgts)
 - x : numerical number
 - units : characters
 - numDgts : number of digits
 - Function to return char string of numerical input with metric multiplier (n, u, m, K, M, ...)
 - Characters in "units" concatenates to numerical converted string

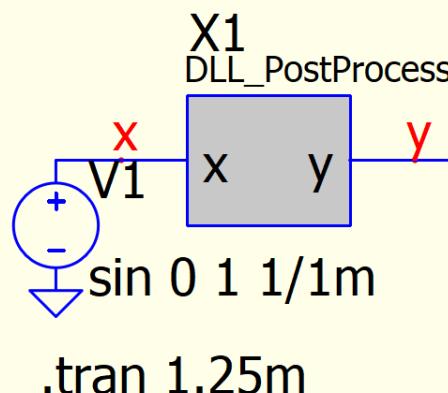


```
// Implement module evaluation code here: // EngFormat(1e-6, "s", 6) returns "1μs"
y = x;
Display("%8s, %8s, %10s\n", EngFormat(t, "s", 4), EngFormat(x, "V", 4), EngFormat(x, "UNIT", 4));
}
```

Ø-Device : [5] Post Processing Function (Undocumented)

Qspice : \02 Template\DLL_PostProcess\

- Post Processing in DLL (not available through template creation)
 - Add this definition : "extern "C" __declspec(dllexport) void PostProcess(struct sMYDLL *inst)"
 - it will be called at the end of the simulation (all steps complete and data file closed)
 - Rename **sMYDLL** as the name of struct if pointer variable is in used
 - This is an undocumented feature and mentioned in revision history



Output Window

C:\KSKelvinQspice\01 User Guide and Script\04 Device
X1: Last output is -1.000000
X1: Simulation Completed!
Total elapsed time: 0.0179553 seconds.

```
struct sDLL_POSTPROCESS
{
    // declare the structure here
    float lastY;
};

extern "C" __declspec(dllexport) void dll_postprocess(struct sDLL_POSTPROCESS **opaque)
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLL_POSTPROCESS *) malloc(sizeof(struct sDLL_POSTPROCESS));
        bzero(*opaque, sizeof(struct sDLL_POSTPROCESS));
    }
    struct sDLL_POSTPROCESS *inst = *opaque;

    // Implement module evaluation code here:
    y = -x;
    inst->lastY = y;
}

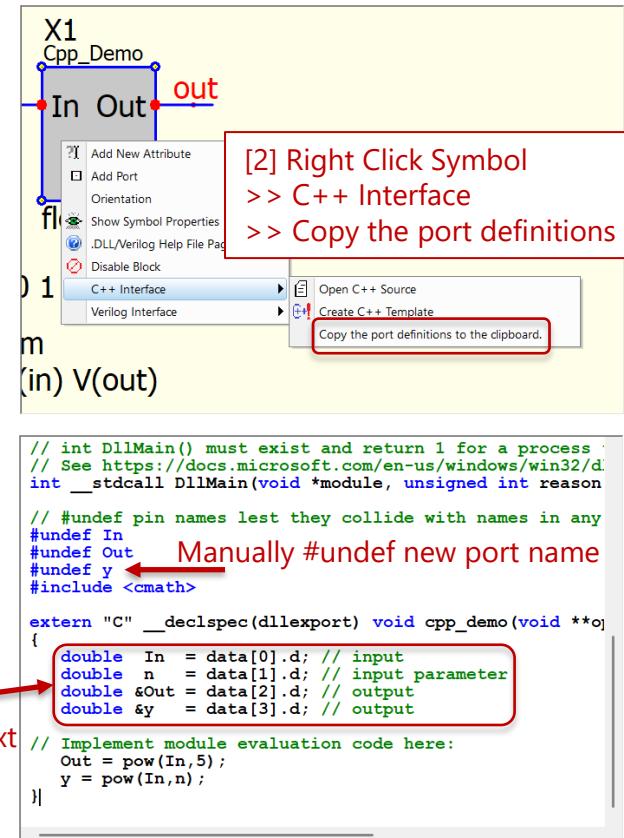
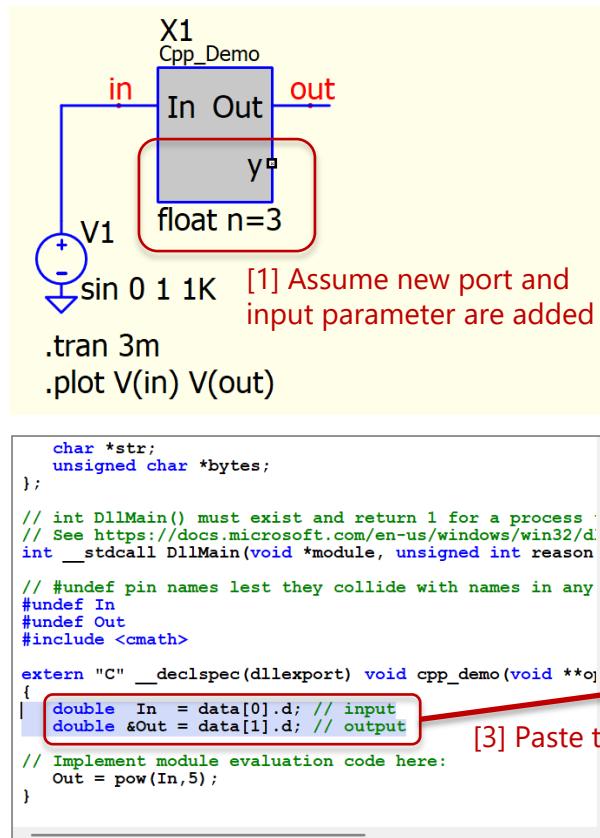
extern "C" __declspec(dllexport) void PostProcess(struct sDLL_POSTPROCESS *inst)
{
    Display("Last output is %f\n",inst->lastY);
    Display("Simulation Completed!");
}
```

Ø-Device (.DLL)

Part 3: Supplementary

Ø-Device : Workflow of Pins and Input Parameters Change

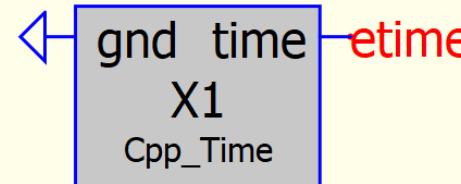
- Change of DLL Symbol
 - If the DLL model name is changed, it requires recreating the DLL template, as multiple function names need to be changed
 - If the pins or input parameters change, the user has the option to copy the new port definitions into the .cpp code instead of recreating the DLL template



Ø-Device : Simulation time and call from directory

Qspice : cpp_time.cpp

- Time
 - **t** is simulation time in C++ code



.tran 1
.plot V(etime)

```
// #undef pin names lest they collide with
// #undef time

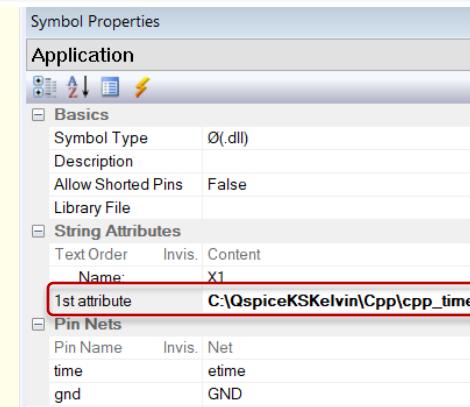
extern "C" __declspec(dllexport) void cpp_
{
    double &time = data[0].d; // output

    // Implement module evaluation code here:
    time = t;
}
```

- Dll from other directory
 - 1st attribute can accept absolute or relative path name
 - If space in directory path, add "" for string format
 - e.g. "C:\Qspice KSKelvin\Cpp\cpp_time"
 - To call for example
cpp_time.dll, not to include .dll
 - Don't attempt to modify .cpp when path included



.tran 1
.plot V(etime)

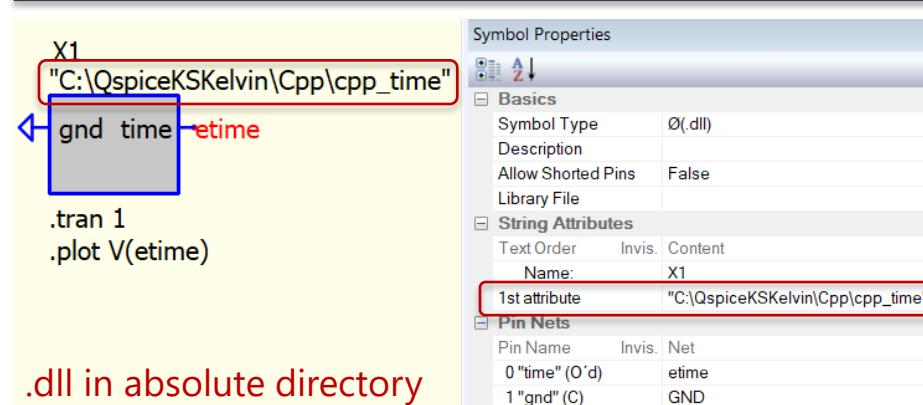
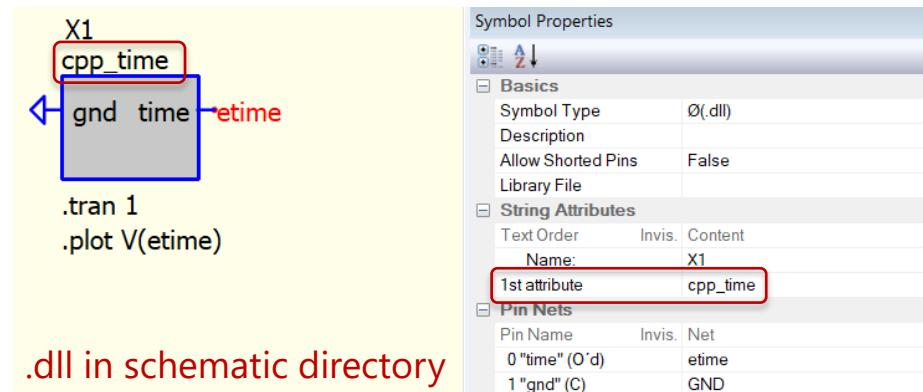


Ø-Device : DLL Directory

Qspice : \03 Supplementary\DLL-Directory\

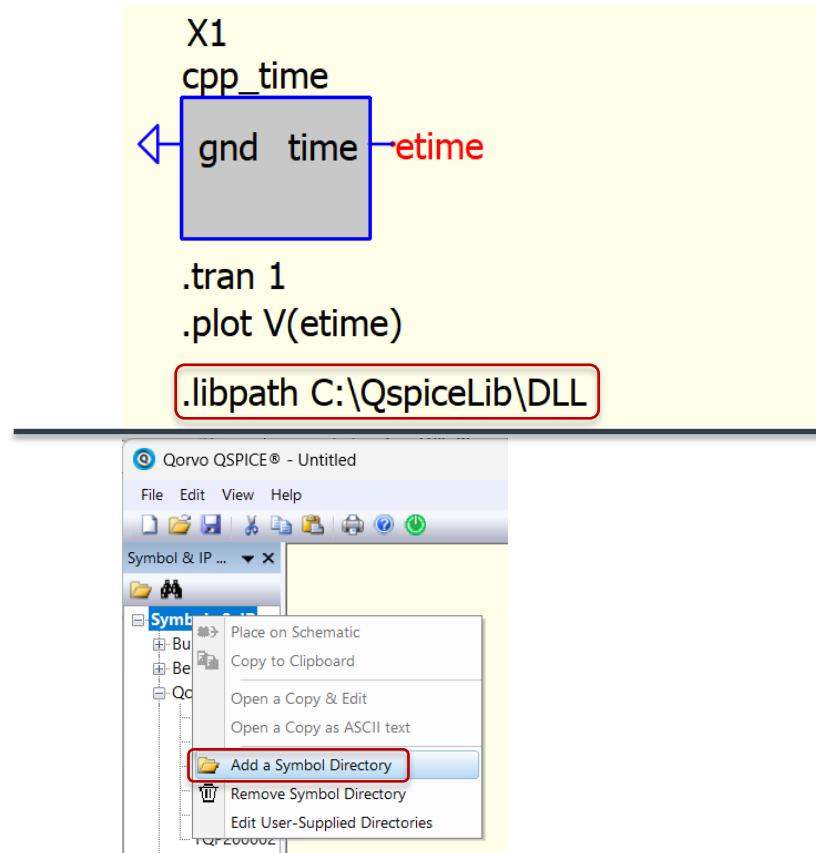
- DLL Directory

- 1st attribute of DLL symbol defines the .dll file to be called
- Assume .dll file name is
 - [filepath][dllname].dll
- .dll in schematic directory
 - 1st attribute is [dllname]
 - This definition allows dll symbol to open .cpp code file if it is also in schematic directory
- .dll in absolute directory
 - 1st attribute is [filepath][dllname]
 - If [filepath] contains space, must use " " to declare entire string path. If no space, " " can be omitted
 - Example : "C:\QspiceKSKelvin\Cpp\cpp_time"
 - This definition cannot link to open a .cpp
- .dll in subdirectory
 - 1st attribute is [subdirectory][dllname]
 - Example : Cpp\cpp_time



Ø-Device : .libpath for .dll path

- .libpath for .dll path
 - If .dll files are in directory path declared with .libpath, Qspice can search into the .libpath directory for the .dll file
 - The DLL symbol 1st attribute doesn't need [filepath] but only [dllname]
- Two ways to declare .libpath
 - Assume .dll file is in C:\QspiceLib\DLL
 - **Method #1**
 - Add .libpath C:\QspiceLib\DLL into schematic
 - **Method #2**
 - View > Symbols & IP
 - Select : Add a Symbol Directory
 - Add C:\QspiceLib\DLL
 - Select : Edit User-Supplied Directories to confirm path is added successfully
 - Now, .libpath C:\QspiceLib\DLL is automatically add into netlist by default



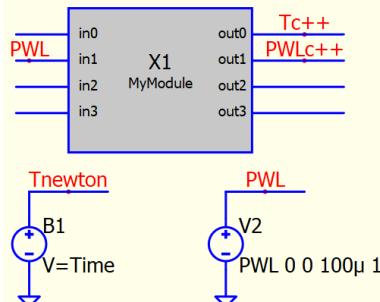
Ø-Device : Simulation time – Analog and DLL Time

Qspice : Analog and DLL Time.qsch

- Analog and DLL Time
 - Exact time "Time" is analog Newton-Raphson iteration which can be called from Behavioral source $V=Time$
 - .DLL "t" is dll time from very last timestep
- In this example
 - out0 output .DLL time "t", which is a different by a timestep to analog time "Tnewton" : Timestep = $V(Tnewton)-V(Tc++)$
 - In the same way, for out1=in1, .DLL output is last timestep data sample and therefore, out1 not exactly equal in1 at same time!

```
extern "C" __declspec(dllexport) void mym
{
    double in0 = data[0].d; // input
    double in1 = data[1].d; // input
    double in2 = data[2].d; // input
    double in3 = data[3].d; // input
    double out0 = data[4].d; // output
    double out1 = data[5].d; // output
    double out2 = data[6].d; // output
    double out3 = data[7].d; // output
}

// Implement module evaluation code here:
out0=t;
out1=in1;
```



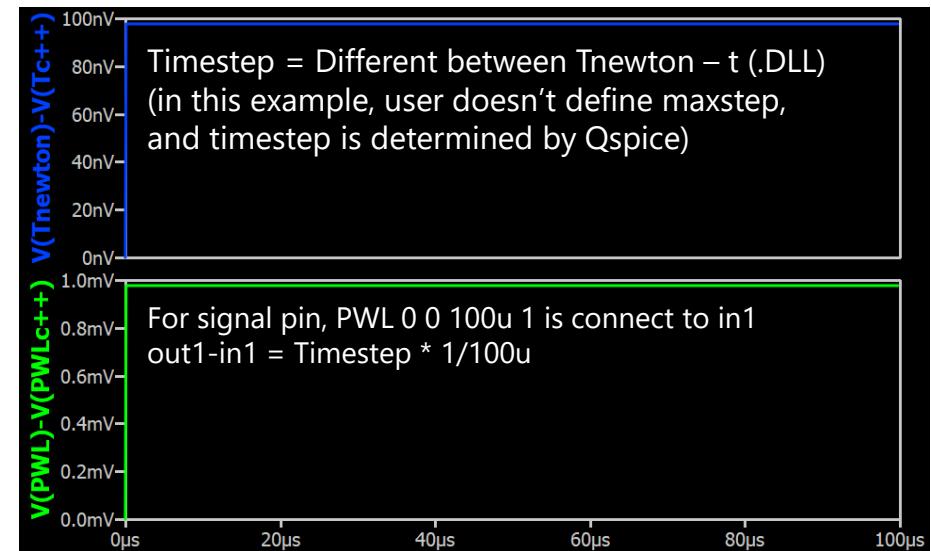
According to Mike Engelhardt

1. Tnewton is the time in the analog Newton-Raphson iteration
2. The .DLL gets the time from the very last timestep

.tran 100 μ
.plot V(PWL)-V(PWLc++)
.plot V(Tnewton)-V(Tc++)
.plot V(x)

For verification

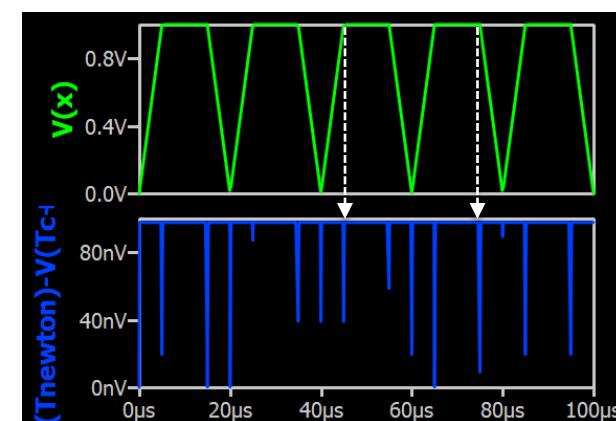
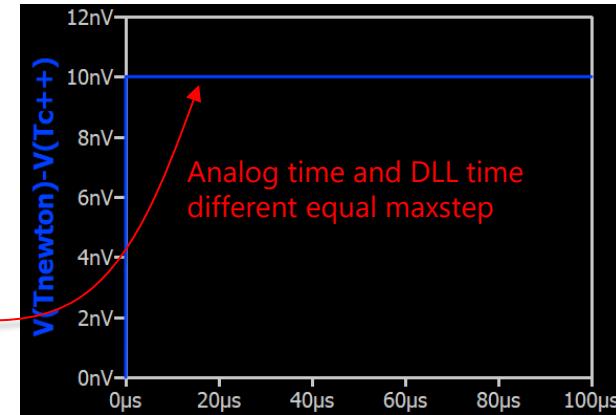
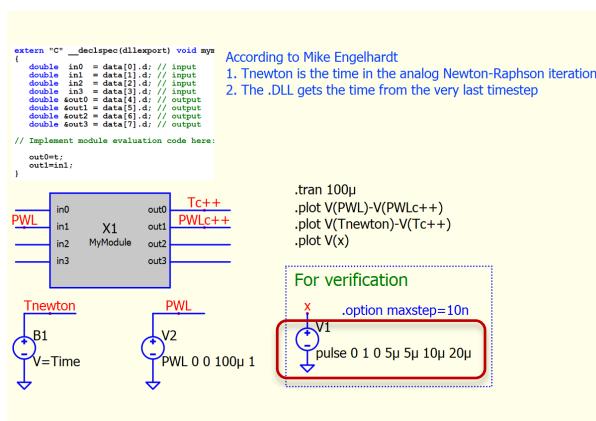
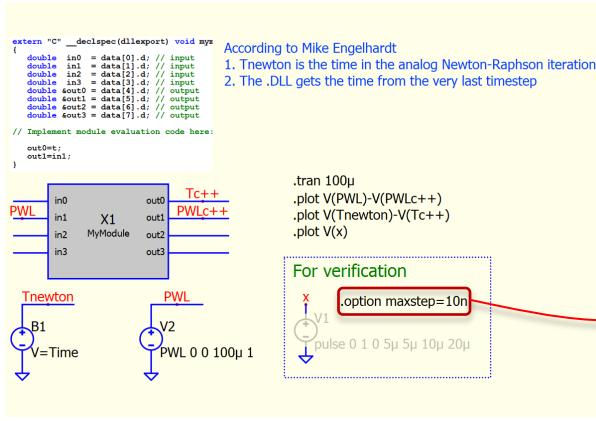
x
V1
.option maxstep=10n
pulse 0 1 0 5 μ 5 μ 10 μ 20 μ



Ø-Device : Simulation time – Analog and DLL Time

Qspice : Analog and DLL Time.qsch

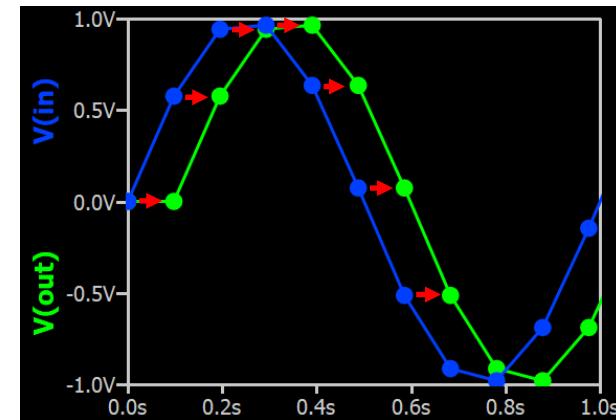
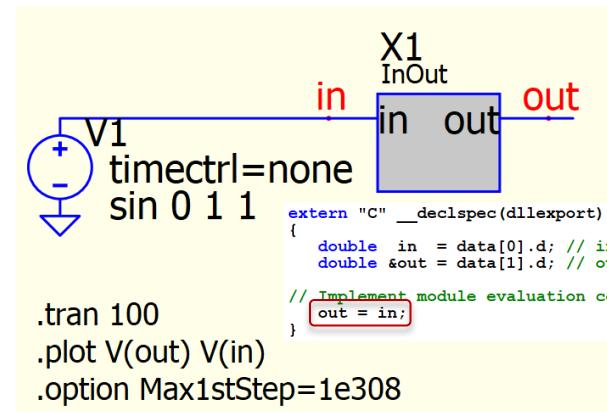
- Change of Timestep #1
 - Different between DLL t and analog time can be affected by assigning max time step with .option maxstep
 - In this example, time different is clamped to equal maxstep setting



DLL Input and Output Timing Relationship

Qspice : parent.InOut.qsch | inout.cpp

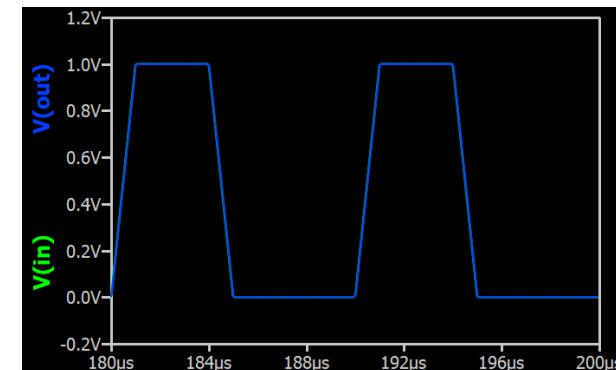
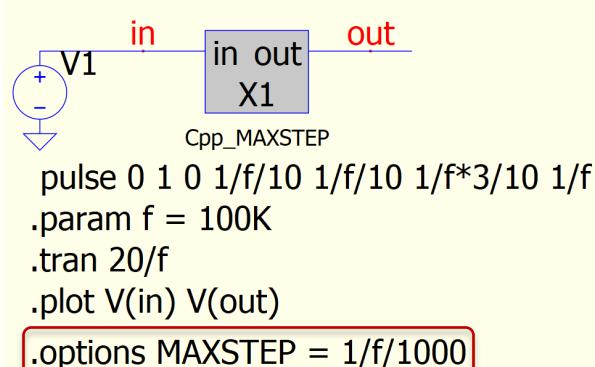
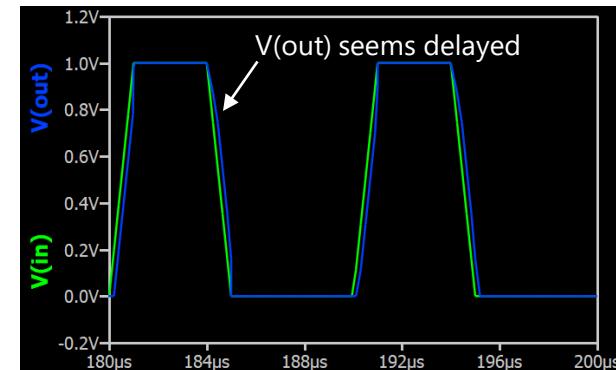
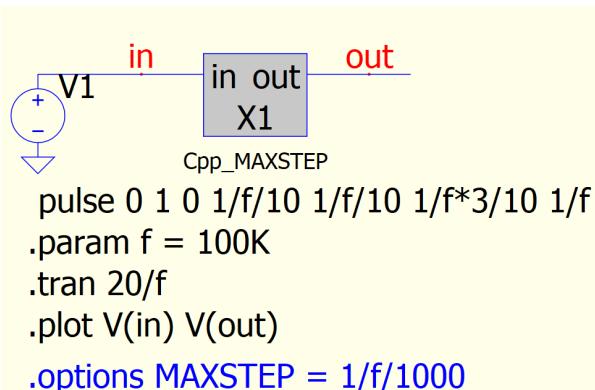
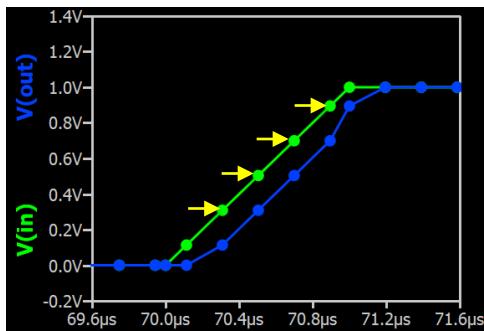
- DLL Input and Output
 - Output of Ø-Device is returned with one timestep of delay
 - This example compare in and out relationship with a "out=in;" C++ code
 - Timestep in this simulation is at about ~0.1s, therefore, V(out) is delayed by 0.1s
 - If any device change timestep (SPICE is varying timestep throughout simulation), this delay can vary over simulation



DLL Input and Output Timing Relationship with MAXSTEP

Qspice : Cpp_MAXSTEP.qsch

- DLL Input and Output
 - This delay may result in visually strange output, which can be visually improved by limiting the maximum time step or TTOL parameter
 - .options MAXSTEP=<value>



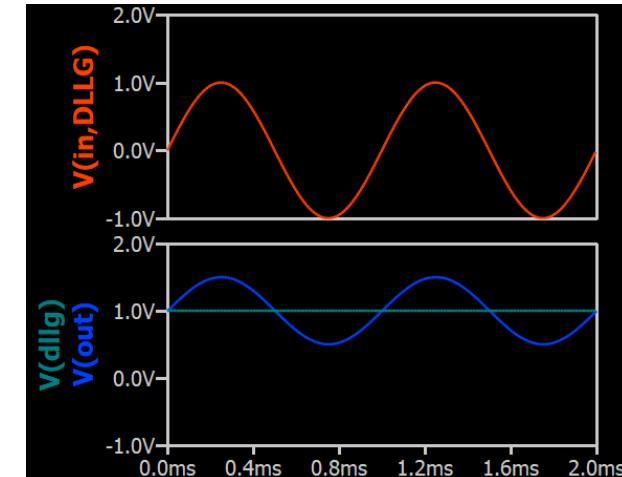
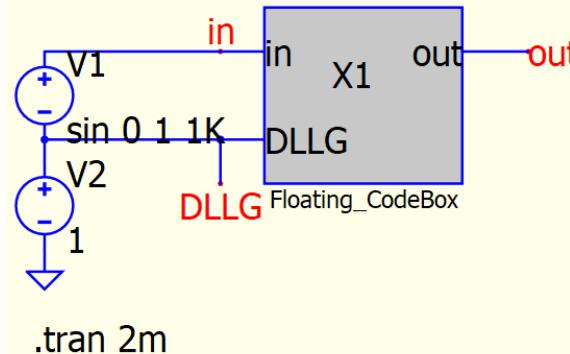
DLL's GND in Ø-Device

Qspice : Floating_CodeBox.qsch

- Two Purposes of DLL's Gnd
 - Floating operation where Ø-Device reference is not 0
 - Only output but no input port is defined (previous slide)

Example of floating ground operation

```
// Implement module evaluation code here:  
out = in * 0.5;
```

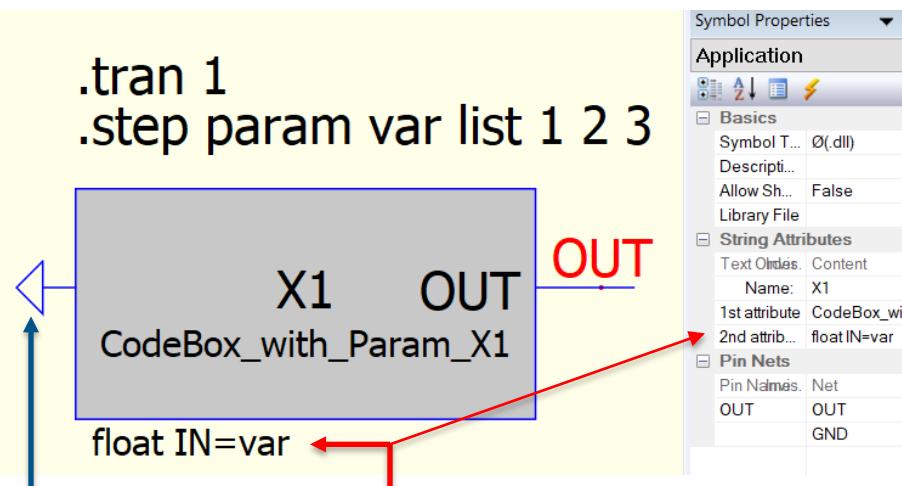


Engelhardt

The inputs and outputs of a .DLL go through a converter. If there's no DLL GND specified, the converters operate ground referenced. If you give a .DLL a GND, then inputs and output are referred to that .DLL GND port. It lets you run your logic hot decked as one might in an offline converter.

Ø-Device with Input Parameter (No Input Port)

Qspice : CodeBox_with_Param.qsch



[1] For Input parameter, in creating hierarchical block

1. Right click the block > select Add attribute
2. [data type] [Input Port name] = [parameter] or <val>

As GND is needed if hierarchical doesn't have input port

- Add Port
- Right click Port > Data Type > DLL's GND

```
// #undef pin names lest they collide with names in ^
#define OUT

extern "C" __declspec(dllexport) void codebox_with_p(
{
    double IN = data[0].d; // input parameter
    double &OUT = data[1].d; // output

    // Implement module evaluation code here:
    OUT = IN;
}
```

** Important note!! Whenever you change input port, parameter or output port, you should recreate and copy code to a new C++ template, as Qspice requires particular order for index in data[]

- Right click the block > C++ Interface > Create C++ Template
- In this example, float IN=var will auto generate as input parameter
double IN = data[0].d;

Variables : Global and Instance/Member Variables

Qspice : Variable-OneDLLBlock.qsch | dllvar.cpp

- Global and Instance/Member Variables
 - The C++ code in this example increments variables a and b by 1 every second
 - The instance/member variable **a** is output to y1, while the global variable **b** is output to y2
 - If the schematic consists of only one DLL block, y1 and y2 will be identical
 - However, when multiple DLL blocks are used, the global variables will update each other, leading to unexpected results in the output

```
float b; ← Global variable

struct sDLLVAR
{
    // declare the structure here
    float a; ← Member variable
    float lastT;
};

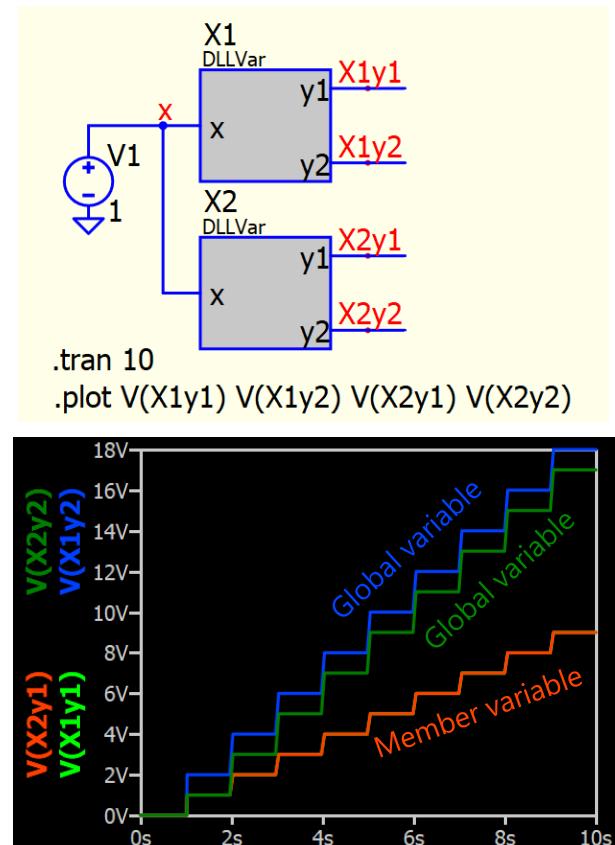
extern "C" __declspec(dllexport) void
{
    double x = data[0].d; // input
    double &y1 = data[1].d; // output
    double &y2 = data[2].d; // output

    if(!*opaque)
    {
        *opaque = (struct sDLLVAR *) malloc(sizeof(struct sDLLVAR));
        (*opaque).lastT = 0;
    }
    struct sDLLVAR *inst = *opaque;

    // Implement module evaluation code !
    if (t - inst->lastT > 1)
    {
        inst->a = inst->a + 1;
        y1 = inst->a;

        b = b + 1;
        y2 = b;

        inst->lastT = t; Increase a and b
    }
}
```



Variables : Global and Instance/Member Variables – Initialization

Qspice : (Folder - Variable Initialization p1) Variable-Initialization.qsch | dllvar.cpp

- Global Var Initialization
 - Initial value can be assigned directly in declaration
- Member Var Initialization
 - Choose method 1 or 2
 - This code include both, actual implementation only need one
 - Method 1 : Assign Initial Value to member
 - `(*opaque)->[var] = [initial];`
 - Put this within if(!*opaque)
 - Method 2 : Initial Status Flag
 - Set an initial status flag (e.g. float init) and use if condition to check this initial status flag
 - If initial status is true, enter the if condition and setup all instances with their respective initial values

```
float b=3; ← Global variable Initialization

struct sDLLVAR
{
    // declare the structure here
    float a;
    float init; ← Instance variable Initialization
    float lastT; with a status variable (method 2)
};

extern "C" __declspec(dllexport) void dllvar(s
{
    double x = data[0].d; // input
    double &y1 = data[1].d; // output
    double &y2 = data[2].d; // output

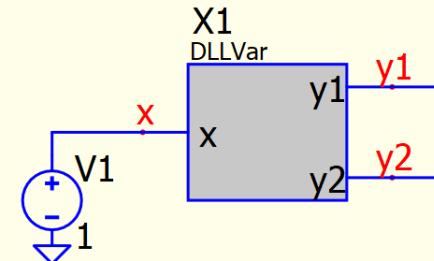
    if(!*opaque)
    {
        *opaque = (struct sDLLVAR *) malloc(sizeof(struct sDLLVAR));
        bzero(*opaque, sizeof(struct sDLLVAR));

        // assign initial value (method 1)
        (*opaque)->a = 1; Initialize (method 1)
    }
    struct sDLLVAR *inst = *opaque;

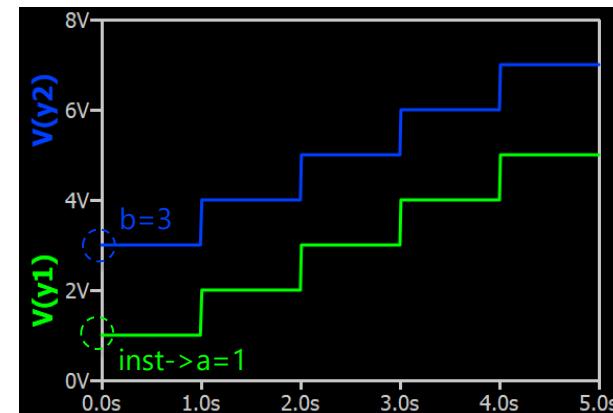
    // Implement module evaluation code here:

    // assign initial value (method 2)
    if (inst->init == 0){
        inst->a = 1;
        inst->init = 1;
    }

    y1 = inst->a;
    y2 = b; If condition for inst->init to
            help initialize (method 2)
}
```



.tran 5
.plot V(y1) V(y2)



Variables : Global and Instance/Member Variables – Initialization

Qspice : (Folder - Variable Initialization p2) Variable-Initialization.qsch | dllvar.cpp

- Member Var Initialization
 - Method 3 : Constructor Initialization
 - Use constructor initialization
 - With new and delete operators for memory allocation
 - Few lines of Qspice standard C++ template need to be commented, e.g. malloc(), bzero() and free()

```
struct sDLLVAR
{
    // declare the structure here
    float a;
    float lastT;

    sDLLVAR() : a(1.0),lastT(0.0){}; // Initial value declaration

extern "C" __declspec(dllexport) void dllvar(st)
{
    double x = data[0].d; // input
    double &y1 = data[1].d; // output
    double &y2 = data[2].d; // output

    if(!*opaque)
    {
        /*opaque = (struct sDLLVAR *) malloc(sizeof(struct sDLLVAR));
        bzero(*opaque, sizeof(struct sDLLVAR));*/
        // assign initial value (method 3)
        *opaque = new sDLLVAR();
    }
    struct sDLLVAR *inst = *opaque;

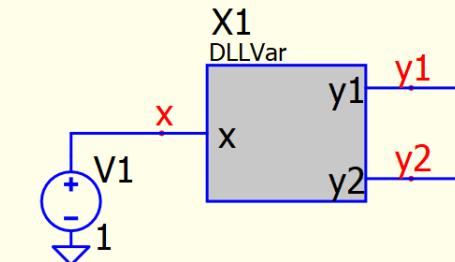
    // Implement module evaluation code here:
    y1 = inst->a;
    y2 = b;

    if (t - inst->lastT > 1){
        inst->a = inst->a + 1;
        y1 = inst->a;

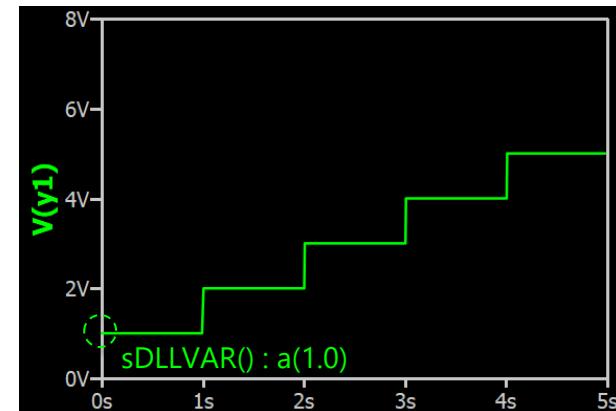
        b = b + 1;
        y2 = b;

        inst->lastT = t;
    }
}

extern "C" __declspec(dllexport) void Destroy(st)
{
    //free(inst);
    delete inst;
}
```



.tran 5
.plot V(y1) V(y2)



Variables : Local Variable

Qspice : Local-Variable.qsch | freq2voltage.cpp

- Local Variable

- Local variables are only accessible within the scope of that function, which can't be accessed or modified outside of it
- It is useful in Qspice for variable which is not required to store between each timestep of simulation

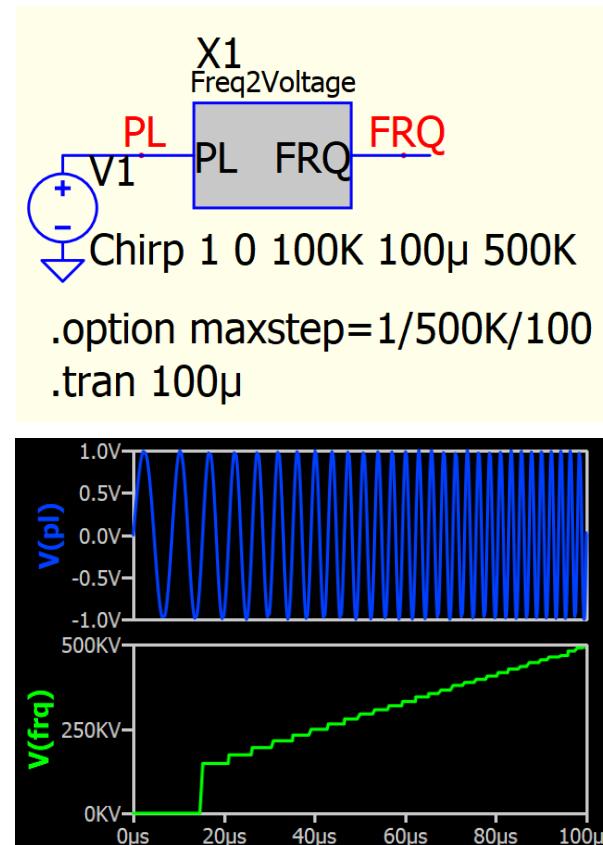
```
struct sFREQ2VOLTAGE
{
    // declare the structure here
    float lastT;
    float lastPL;
};

extern "C" __declspec(dllexport) void freq2voltage(
{
    double PL = data[0].d; // input
    double &FRQ = data[1].d; // output

    if(!*opaque)
    {
        *opaque = (struct sFREQ2VOLTAGE *) malloc(sizeof(struct sFREQ2VOLTAGE));
        bzero(*opaque, sizeof(struct sFREQ2VOLTAGE));
    }
    struct sFREQ2VOLTAGE *inst = *opaque;

    // Implement module evaluation code here:

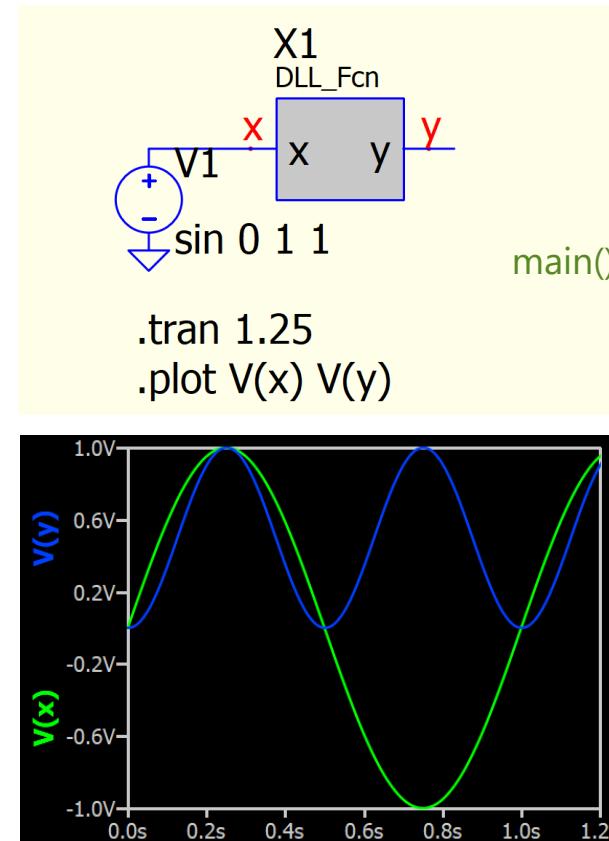
    // rising edge at 0V
    if (inst->lastPL < 0 & PL > 0){
        if (inst->lastT != 0){
            float T = t - inst->lastT;
            FRQ = 1/T;
        }
        inst->lastT = t;
    }
    inst->lastPL = PL;
}
```



Functions : Declare User Functions

Qspice : \03 Supplementary\DLL_Fcn\01-Basic\

- Declare User Functions
 - C compilers read code from top to bottom. A function call in main() needs to know the function's signature (return type and parameters) to validate the call
 - A function written after main() requires a forward declaration
 - If a function is written before main(), a forward declaration is not required



```
// #undef pin names lest they collide with n
#undef x
#undef y
Forward declaration
// Forward declaration of the math function
float maths(float val);

extern "C" __declspec(dllexport) void dll_fc
{
    double x = data[0].d; // input
    double &y = data[1].d; // output

    // Implement module evaluation code here:
    y = maths(x);
}

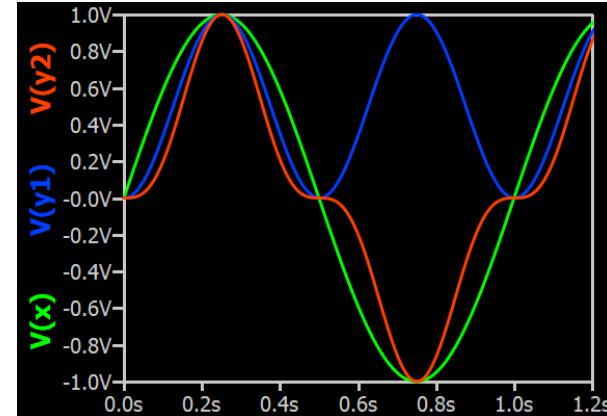
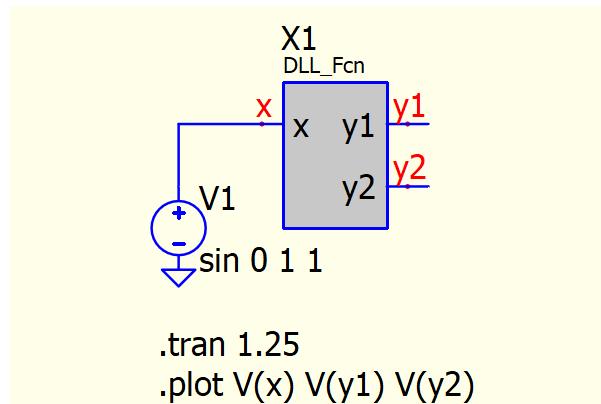
// Function prototype for the math function
float maths(float val)
{
    return (val * val);
}
```

Function written after main()

Functions : Return multiple values with Pointers

Qspice : \03 Supplementary\DLL_Fcn\02-MultipleOutputs-Pointer\

- Function return multiple values – Pointer
 - Function in C cannot directly return with multiple values
 - A common way is to pass pointers as parameters to "return" multiple values



```
// Function - using pointers
void maths(double val, double *square, double *cube)
{
    *square = val * val;      // Calculate square and store via pointer
    *cube = val * val * val;  // Calculate cube and store via pointer
}

extern "C" __declspec(dllexport) void dll_fcn(void **opaque, double t, union uData *data)
{
    double x = data[0].d; // input
    double &y1 = data[1].d; // output
    double &y2 = data[2].d; // output

    // Implement module evaluation code here:
    // Call maths function with input x, store results in y1 and y2
    maths(x, &y1, &y2);
}
```

Functions : Return multiple values with Structure (struct)

Qspice : \03 Supplementary\DLL_Fcn\03-MultipleOutputs-Struct\

- Function return multiple values – Structure

```
// Define a struct to hold multiple results
typedef struct{
    double square;
    double cube;
}CalculationResult;

// Function that calculates square and cube, returning them in a struct
CalculationResult maths(double val)
{
    CalculationResult result;
    result.square = val * val;      // Calculate square and store struct
    result(cube = val * val * val; // Calculate cube and store struct
    return result;
}

extern "C" __declspec(dllexport) void dll_fcn(void **opaque, double t, union
{
    double x = data[0].d; // input
    double &y1 = data[1].d; // output
    double &y2 = data[2].d; // output

    // Implement module evaluation code here:

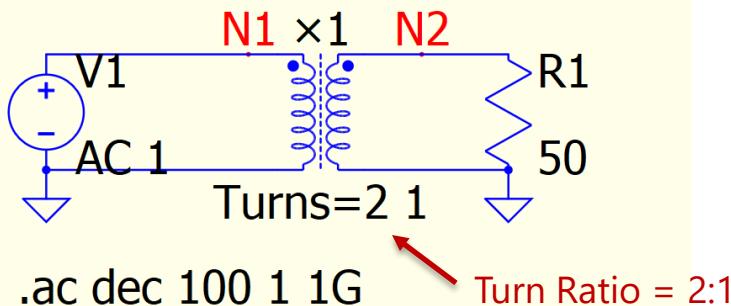
    // Call maths function and get struct with both results
    CalculationResult results = maths(x);
    y1 = results.square; // Assign square result from struct to first output
    y2 = results(cube); // Assign cube result from struct to second output
}
```

**x-Device
Transformer**

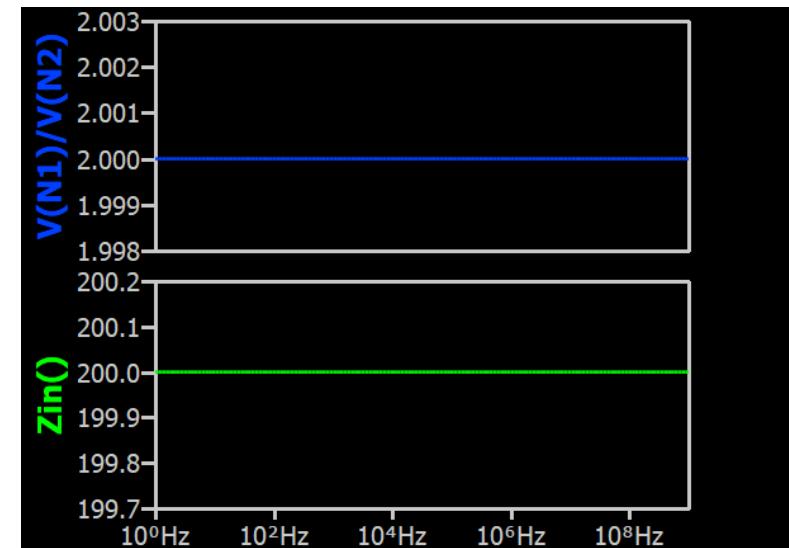
✗-Device : Transformer

Qspice : Transformer - Basic.qsch

- ✗-Device: Transformer
 - Syntax: $\times nnn \text{ «PRI+ PRI- SEC1+ SEC1- SEC2+ SEC2 [...]» } <\text{TURN}=N1 N2 N3 ...>$ [Additional Instance Parameters]
 - Ideal Transformer Equation (Inductance $\rightarrow \infty$)
 - $n = \frac{N_1}{N_2} = \frac{v_1}{v_2}$ and $Z_{N1} = n^2 Z_{N2}$



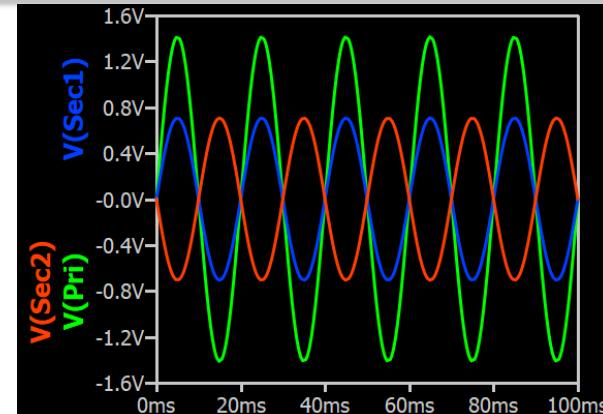
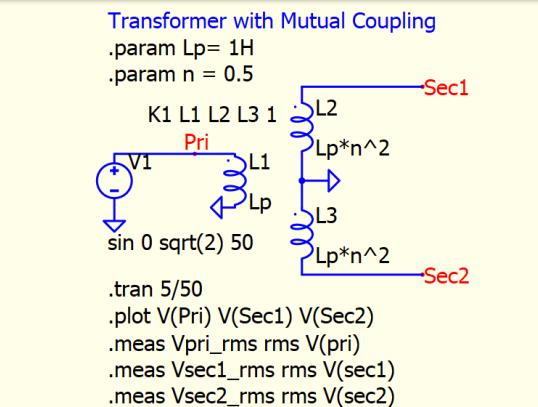
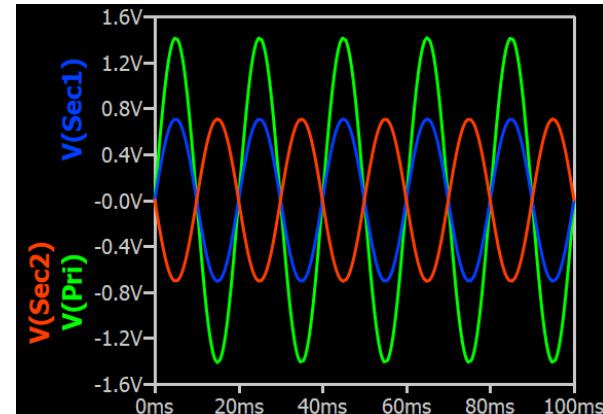
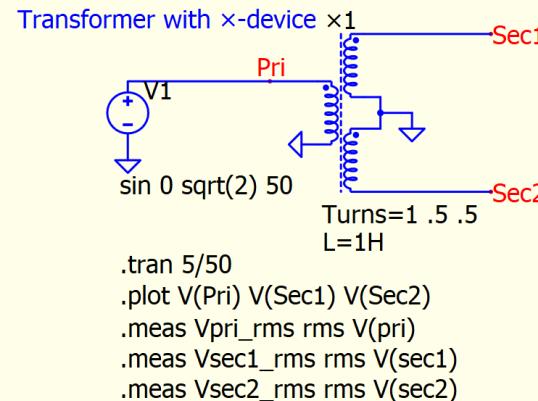
```
.ac dec 100 1 1G
.func Zin() V(N1)/-(I(V1))
.plot Zin()
.plot V(N1)/V(N2)
```



\times -Device : Transformer

Qspice : Transformer - Multiple Windings.qsch | Mutual - Multiple Windings.qsch

- Multiple Windings
 - \times -Device supports multiple winding by defining turns ratio
 - Example demonstrate using \times -Device or mutual inductance for one primary and two secondary windings transformer



x-Device : Transformer Instance Params

Transformer Instance Parameters

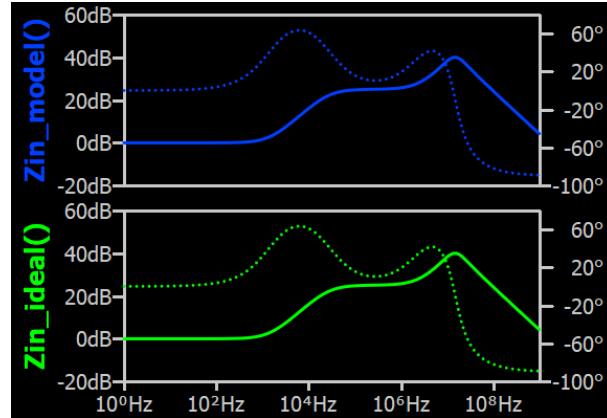
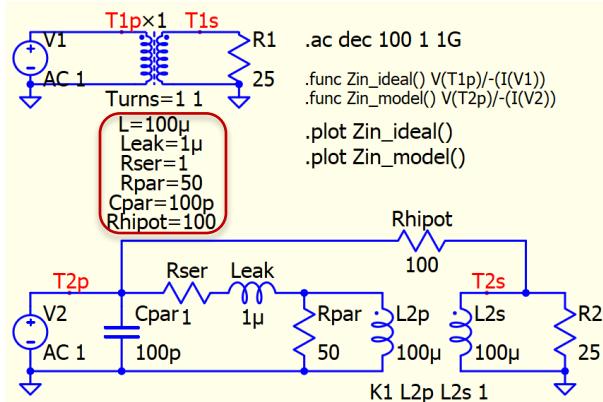
Name	Description	Units	Default
C	Primary shunt capacitance(aka CPAR)	F	0.0
ISAT ¹	Current that drops inductance to SATFRAC of zero-current value	A	Infinite
K ¹	Alternate means of specifying LEAK		1
L	Primary zero-current inductance	H	Infinite
LEAK	Leakage inductance(aka LLEAK)	H	0
LSAT ¹	Inductance asymptotically approached in saturation	H	10% of L
RHIPOT	DC resistance to primary	Ω	Infinite
RPAR ¹	Primary parallel resistance	Ω	Infinite
RSER ¹	Primary series resistance	Ω	0.0
SATFRAC ¹	Fractional drop in L at ISAT		0.7
TURNS	List of relative number of turns		

^{1]} ISAT, K, LSAT, RPAR, RSER, and SATFRAC are ignored unless L is given positive definite and finite.

Instance Params : C, L, Leak, Rser, Rpar, Rhipot, Isat, Lsat, Satfrac

Qspice : Transformer - Cpar L Leak Rser Rpar Rhipot.qsch | Transformer - Isat Lsat Satfrac.qsch

- C, L, Leak, Rser, Rpar, Rhipot
 - L : Primary Inductance
 - C (Cpar) : Primary shunt capacitance
 - Leak : Leakage Inductance
 - Rser : Series Resistance
 - Rpar : Parallel Resistance
 - Rhipot : DC resistance to primary



- ISAT, LSAT, SATFRAC
 - Isat : Current that drops inductance to SATFRAC*L
 - Satfrac : Fractional drop in L at Isat
 - Lsat : Inductance asymptotically approached in saturation
 - These parameters are ignored unless L is given

