Device Parameters Extraction Within Silvaco Simulation Software

Dragica Vasileska

Arizona State University

- Device parameter extraction is an important ingredient in the device simulation process as parameters are needed for higher circuit level models and for verification of the device models and calibration with experimental data.
- In this set of slides we give comprehensive overview of the parameter extraction statements within Silvaco simulation framework.

Device Extraction

Device extraction always deals with a “log-file” that contains I-V information produced by a device simulator (such as ATLAS), and therefore deals almost exclusively in curves.

Extract allows the user to construct a curve using separate X and Y axes. For each axis, the user can choose the voltage or current on any electrode, the capacitance or conductance between any two electrodes, or the transient time for AC simulations. The axes may be manipulated individually, such as multiplication or division by a constant, or axes may be combined in algebraic functions.

The Curve

The basic element is always the curve. Once the curve is constructed, it can be used as is, by saving it to a file for use by TONYPLOT, or as an OPTIMIZER target, or it can be used as the basis for further extraction.

To construct a curve representing voltage on electrode “emitter1” (on the X axis) versus current on electrode “base2", write:

```
extract name="iv" curve(v."emitter1", i."base2")
```

The first variable specified inside the parentheses becomes the X axis of the curve; the second variable becomes the Y axis. The v.“name” and i.“name” syntax is used for any electrode name — just insert the proper name of the electrode. The electrode name must have been defined previously (such as in the device deck, or previous to that in an ATHENA input deck using the “electrode” statement, or interactively in DEVEDIT). Electrode names may contain spaces but must always be quoted.

Transient time is represented by keyword time:

```
extract name="It curve" curve(time, i."anode")
```

For Device temperature curves use:

```
extract name="VdT" curve(v."drain", temperature)
```

For extracting a frequency curve use:

```
extract name="Idf" curve(i."drain", frequency)
```

To extract a capacitance or conductance curve, use this syntax:

```
extract name="cv" curve(c."electrode1""electrode2", v."electrode3")
```

and

```
extract name="gv" curve(g."electrode1""electrode2", v."electrode3")
```

For other electrical parameters use the following syntax:

```
extract name="IdT" curve(elect."parameter“,v.”drain")
```

It is also possible to shift or manipulate curve axes. Each axis is manipulated separately. The simplest form of axis manipulation is algebra with a constant:

```
extract name="big iv" curve(v."gate"/50, 10*i."drain")
```

Any constant expression can be multiplied, divided, added, or subtracted to each axis.
Curve axis may also be combined algebraically, similar to TONYPLOT’s function capability:

```
extract name="combine" curve(i."collector", i."collector"/i."base")
```

All electrode values (current, voltage, capacitance, conductance) may be combined in any form this way.

Another curve type is deriv() used to return the derivative (dydx). For example, statement below will create the curve of dydx gate bias and drain current plotted against and X axis of gate bias.

```
extract name="dydx" deriv(v."gate", i."drain") outfile="dydx.dat"
```

It is also possible to calculate dydx to the nth derivative as below:

```
extract name="dydx2" deriv(v."gate", i."drain", 2) outfile="dydx2.dat"
```

To find local maxima and minima on a curve, the section of the curve X axis can be limited. The following statement extracts the maximum drain current where gate bias is between the limits of 0.5 volts and 2.5 volts.

```
extract name="limit" max(curve(v."gate", i."drain",x.min=0.5 x.max=2.5)) outf="limit.dat"
```

In addition, there are several operators which apply to curve axes. They are:

```
abs(axis)
log(axis)
log10(axis)
sqrt(axis)
atan(axis)
-axis
```

For instance:

```
extract curve(abs(i."drain"), abs(v."gate"))
```

The operators may be combined, i.e., log10(abs(axis)). These operators work on curve axes from process simulation as well.

Curve Manipulation

A number of curve manipulation primitives exist:

```
min(curve)
max(curve)
ave(curve)
minslope(curve)
maxslope(curve)
slope(line)
xintercept(line)
yintercept(line)
area from curve
area from curve where x.min=X1 and x.max=X2
x.val from curve where y.val=k
y.val from curve where x.val=k
x.val from curve where y.val=k and val.occno=n
y.val from curve where x.val=k and val.occno=n
grad from curve where y.val=k
grad from curve where x.val=k
```

For instance, using the BJT curve example, the user could find the maximum of Ic/Ib vs Ic, or maximum beta, by writing:

```
extract name="max beta" max(curve(i."collector", i."collector"/i."base"))
```

max(), min(), and ave() all work on the Y axis of the curve.

The sloped lines and intercepts often work together. The primitives minslope() and maxslope() can be thought of as returning a line. Extracting a line by itself has no meaning, so three other operators take a line as input. The operators are slope(), which returns the slope of the line, and xintercept() and yintercept(), which return the value where the line intercepts the corresponding axis.

For instance, a Vt test for MOS devices looks at a curve of Vg (x) versus Id (y), and finds the X intercept of the maximum slope. Such a test would look like:

```
extract name="vt" xintercept(maxslope(curve(abs(v."gate", abs(i."drain"))))
```

Some Vt tests take off Vd/2 from the resulting value. You could write:

```
extract name="vt" xintercept(maxslope(curve(abs(v."gate",
abs(i."drain"))))) - ave(v."drain")/2
```

Note that the last example uses:
ave(v."drain")/2

The min(), max(), and ave() operators can be used on both curves,

```
extract name="Iave" ave(curve(v."gate", i."drain"))
```

and also on individual curve axes,

```
extract name="Iave" ave(i."drain")
```

or even on axis functions:

```
extract name="Icb max" max(i."collector"/i."base")
```

The user can also find the Y value on a curve for a given X value, and the other way round. For example, to find the collector current (Y) for base voltage 2.3 (X), use:
extract name="Ic[Vb=2.3]" y.val from curve(abs(v."base"),
abs(i."collector")) where x.val = 2.3

EXTRACT uses linear interpolation if necessary. If more than one point on the curve matches the condition, the first one is taken, unless the following syntax is used to specify the occurrence of the condition. This example would find the second Y point on the curve matching an X value of 2.3:

```
extract name="Ic[Vb=2.3]" y.val from curve(abs(v."base",abs(i."collector"))
where x.val = 2.3 and val.occno =2
```

The condition used for finding an intercept can be a value or an expression and therefore use the min(), max(), and ave() operators. The following command creates a transient time against drain-gate capacitance curve and calculate the intercepting time where the capacitance is at its minimum value:

```
extract name="t at Cdrain-gate[Min]" x.val from curve(time, c."drain""gate")
where y.val=min(c."drain""gate")
```

In addition to finding intercept points on curves, it is possible to calculate the gradient at the intercept, specified by either a Y or X value, as shown below:

```
extract name="slope_at_x" grad from curve(v."gate", i."drain") where x.val=1.5

extract name="slope_at_y" grad from curve(v."gate", i."drain") where y.val=0.001
```

It is also possible to find the area under a specified curve for either the whole curve or as below, between X limits:

```
extract name="iv area" area from curve(v."gate", c."drain""gate")
where x.min=2 and x.max=5
```

General Curve Examples

Curve Creation

The following command extracts a curve of collector current against base voltage and places the output in icvb.dat.

```
extract name="IcVb curve" curve(i."collector", v."base") outfile="icvb.dat"
```

Min Operator with Curves
The following command calculates the minimum value for a curve of drain current against internal gate voltage.

```
extract name="Vgint[Min]" min(curve(i."drain", vint."gate"))
```

Max Operator with Curves
The following command calculates the maximum value for a curve of base voltage against basecollector capacitance.

```
extract name="Cbase-coll[Max]" max(curve(v."base", c."base""collector"))
```

Ave Operator with Curves
The following command calculates the average value for a curve of drain current against gate-drain conductance.

```
extract name="Ggate-drain[Ave]" ave(curve(i."drain", g."gate""drain"))
```

X Value Intercept for Specified Y
The following command creates a frequency against drain current curve and calculate the intercepting frequency for a drain current of 1.5-e6.

```
extract name="Freq at Id=1.5e-6" x.val from curve(frequency, i."drain") where y.val=1.5e-6
```

Y Value Intercept for Specified X
The following command creates a drain voltage against device temperature curve and calculates the intercepting temperature for a drain voltage of 5V.

```
extract name="T at Vd=5" y.val from curve(v."drain", temperature) where x.val=5.0
```

Abs Operator with Axis
The following command creates a curve of absolute gate voltage against absolute optical wavelength

(log, log10 and sqrt also available).

```
extract name="Vg-optW curve" curve(abs(v."gate"), abs(elect."optical.wavelength"))
```

Min Operator with Axis Intercept
The following command creates a transient time against gate-drain capacitance curve and calculate the intercepting time where the capacitance is at its minimum value.

```
extract name="t at Cgate-drain[Min]" x.val from curve(time, c."gate""drain")
where y.val=min(c."gate""drain")
```

Max Operator with Axis Intercept
The following command creates a collector current against collector current divided by base current curve and calculate the intercepting collector current where Ic/Ib is at a maximum value.

```
extract name="Ic at Ic/Ib[Max]" x.val from curve(i."collector",
i."collector"/i."base") where y.val=max(i."collector"/i."base")
```

Second Intercept Occurrence
The following command creates a gate voltage against source photo current curve and calculates the second intercept of gate voltage for a source photo current of 2e-4.

```
extract name="2nd Vg at Isp=2e-4" x.val from curve(v."gate", elect."source
photo current") where y.val=2e-4 and val.occno=2
```

Gradient at Axis Intercept:
This statement create a probe Itime against drain current curve and find the gradient at the point where probe Itime is at a maximum.

```
extract name="grad_at_maxTime" grad from curve(probe."Itime",
i."drain") where y.val=max(probe."Itime")
```

Axis Manipulation with Constants
The following command creates a gate voltage divided by ten against total gate capacitance multiplied by five. (Add and subtract also available).

```
extract name="Vg/10 5*C-gg curve" curve(v."gate"/10, 5*c."gate""gate")
```

X Axis Interception of Line Created by Maxslope Operator
The following command calculates the X axis intercept for the maximum slope of a drain current against gate voltage curve.

```
extract name="Xint for IdVg" xintercept(maxslope(curve(i."drain",
v."gate")))
```

Y Axis Interception of Line Created by Minslope Operator
The following command calculates the Y axis intercept for the minimum slope of a substrate current against drain voltage.

```
extract name="Yint for IsVd" yintercept(minslope(curve(i."substrate",
v."drain")))
```

Axis Manipulation Combined with Max and Abs Operators
The following command calculates the maximum value of drain-gate resistance.

```
extract name="Rdrain-gate[Max]" max(1.0/(abs(g."drain""gate")))
```

Axis Manipulation Combined with Y Value Intercept
The following command creates a gate voltage against drain-gate resistance and calculates the intercepting drain-gate resistance for a gate voltage of 0V.

```
extract name="Rdrain-gate at Vg=0" y.val from curve (v."gate", 1.0/abs(g."drain""gate"))
where x.val=0.0
```

Derivative
The statement below creates the curve of dydx gate bias and drain current plotted against and X axis

of gate bias

```
extract name="dydx" deriv(v."gate", i."drain") outfile="dydx.dat"
```

This further example calculates to the 2nd derivative.

```
extract name="dydx2" deriv(v."gate", i."drain", 2) outfile="dydx2.dat"
```